⇒ Number Systems

1. Binary - base = 2
2. Decimal - 10
3. Octal - 8
4. Hexa Decimal - 16

0 - 0 - 0000
1 - 2^0 - 0001
2 - 2^1 - 0010
3 - 2^1 2^0 - 0011
4 - 2^2 - 0100
5 - 2^2 2^0 - 0101
6 - 2^2 2^1 - 0110
7 - 2^2 2^1 2^0 - 0111
8 - 2^3 - 1000
9 - 1001
10 - 1010
11 - 1011
12 - 1100
13 - 1101
14 - 1110
15 - 1111
16 - 10000
17 - 10001
18 - 10010
19 - 10011
20 - 10100
21 - 10101
22 - 10110
23 - 10111
24 - 11000
25 - 11001
26 - 11010
27 - 11011
28 - 11100
29 - 11101
30 - 11110
31 - 11111

Scanned with CamScanner

1. Binary to decimal conversion

$(01011.1)_2 - (\ )_{10}$

$1\times2^0 + 1\times2^1 + 1\times2^2 + 0\times2^3 + 01\times2^4 + 0\times2^5$

$1+2+4+16 = 23$

(MSB) Most significant bit — least significant bit (LSB)

$(10101)_2 - (\ )_{10}$

$1+2^2+2^4 = 16+4+1 = 21$

$(10100)_2 \Rightarrow 2^2+2^4 = 16+4 = 20$

$(10101.1011)_2 = (\ )_{10}$

$12.5 \quad \frac{25}{100} \times \frac{1}{100} \quad \frac{5}{2}$

$6.25 \quad \frac{12.5}{25} \quad \frac{25}{100} \times \frac{1}{100} \quad \frac{16}{4}$

After binary decimal

$1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4} = \frac{1}{2} + 0 + \frac{1}{8} + \frac{1}{16}$

$= 0.5 + 0.125 + 0.0625$

$1+2^2+2^4 = 16+4+1 = 21$

$= 21.51875$

2. Decimal to Binary

$(25)_{10} = (\ )_2$

2	25	1
2	12	0
2	6	0
2	3	1
2	1	1
	0	

$= 11001$

$(53)_{10} = (\ )_2$

2	53	1
2	26	0
2	13	1
2	6	0
2	3	1
	1	

110101

$(25.65)_{10} = (\ )_2$

→ Number left to the decimal

After the decimal point

$0.65 \times 2 = 1.3 \quad 1$

$0.3 \times 2 = 0.6 \quad 0$

$0.6 \times 2 = 1.2 \quad 1$

$0.2 \times 2 = 0.4 \quad 0$

$0.4 \times 2 = 0.8 \quad 0$

$0.8 \times 2 = 1.6 \quad 1$

$11001.101001\ldots$

Scanned with CamScanner

$(25.25)_{10} = (\ )_2$

$0.25 \times 2 = 0.5 \quad 1$

$0.5 \times 2 = 1 \quad 1$ ↓

11001.11

0.125

$0.125 \times 2 = 0.25 \quad 0$

$0.25 \times 2 = 0.5 \quad 0$

$0.5 \times 2 = 1 \quad 1$ ↓

0.001

3. Octal to Decimal

$(2516)_8 = (\ )_{10}$

$6 \times 8^0 + 1 \times 8^1 + 5 \times 8^2 + 2 \times 8^3$

$6 + 8 + 320 + 1024 = 1358$

→ $(124)_8 = (\ )_{10}$

$4 \times 8^0 + 2 \times 8^1 + 1 \times 8^2$

$64 + 16 + 4 = 84$

→ $(124.241)_8 = (\ )_{10}$

↓

84

$2 \times 8^{-1} + 4 \times 8^{-2} + 1 \times 8^{-3}$

$0.25 + 0.0625 + 0.00195\ldots$

84.31445

4. Decimal to Octal

→ $(55)_{10} = (\ )_8$

8 | 55	7	↑
8 | 6	6	↑
0		

$= (67)_8$

→ $(255)_{10} = (\ )_8$

8 | 255	7	
8 | 31	7	↑
8 | 3	3	↑
0		

$= (377)_8$

Scanned with CamScanner

$(55.8)_{10} = (\ )_8$

$0.8 \times 8 = 6.4 - 6$

$0.4 \times 8 = 3.2 - 3$

$0.2 \times 8 = 1.6 - 1$

$0.6 \times 8 = 4.8 - 4$

$0.2 \times 8 = 1.6 - 1$

$(67.63141..)_8$

$0.25)_{10}$

$0.25 \times 8 = 2.$

5. Hexadecimal to Decimal

0-9, A, B, C, D, E, F

→ $(27CD)_{16} = (\ )_{10}$

$D \times 16^0 + C \times 16^1 + 7 \times 16^2 + 2 \times 16^3$

$13 + 192 + 1792 + 8192 = 10189$

→ $(C5269)_{16} = (\ )_{10}$

$9 \times 16^0 + 6 \times 16^1 + 2 \times 16^2 + 5 \times 16^3 + C \times 16^4$

$9 + 96 + 512 + 20480 + 786432$

→ $(27C.4A6)_{16} = (\ )_{10}$

$C \times 16^0 + 7 \times 16^1 + 2 \times 16^2 + 4 \times \frac{1}{16} + 10 \times \frac{1}{16^2} + 6 \times \frac{1}{16^3}$

$192 + 112 + 512 + 0.25 + 0.0390 + 0.001464$

→ $(0.245B)_{16} = (\ )_{10}$

$2 \times \frac{1}{16} + 4 \times \frac{1}{16^2} + 5 \times \frac{1}{16^3} + 6 \times \frac{1}{16^4}$

$0.125 + 0.0156 + 0.00122 + 0.00009155$

Scanned with CamScanner

6. Decimal to Hexadecimal

→ $(55)_{10} = (\ )_{16}$

$16 \overline{)55}\ 7 \uparrow$ $\quad 3$

$\Rightarrow (37)_{16}$

→ $(255)_{10} = (\ )_{16}$

$16 \overline{)255}\ 15 \uparrow$ $\quad 15$

$(FF)_{16}$

→ $(165)_{10} = (\ )_{16}$

$16 \overline{)165}\ 5 \uparrow$ $\quad 10$

$(A5)_{16}$

→ $(165.125)_{10} = (\ )_{16}$

$0.125 \times 16 = 2$

$(A5.2)_{16}$

→ $(0.65)_{10} = (\ )_{16}$

$0.65 \times 16 = 10.4$ A

$0.4 \times 16 = 6.4$ 6

$0.4 \times 16 = 6.4$ 6 ⋮

$= (0.A66)_{16}$

7. Binary to Hexadecimal

→ 1010 1011 0100 1

0001 | 0101 | 0110 | 1001

1 5 6 9 $= 1569$

→ 0101 | 1101 | 0110 | 1011

5 13 6 11 $= 5D6B$

D B

→ 1010110.1101101

0101 | 0110.1101 | 1010

5 6 D A $= 56DA$

Reverse process for Hexadecimal to Binary

Scanned with CamScanner

8. Binary to octal

→ 10|101|101|101 = $(555)_8$
5 5 5

→ 001|111|011|001|001 = $(17311)_8$
1 7 3 1 1

→ 001|010|110.110|110|100 = $(126.664)_8$
1 2 6 6 6 4

Hexadecimal ⟷ Binary ⟷ Octal

Complements

1's complement :-
Interchange zero's and one's

011010 → 100101

2's complement = 1's complement + 1

→ 011010 → 2's → 100101
+ 1
100110

in arithematic 1+1 = 0 with additive carry 1

→ 1001 → 0110
+ 1
0111

→ 00000001 → (0000) 11111110
+1
11111111

→ 100000 → (0000) 011111
+ 1
100000

Scanned with CamScanner

→ 01011010

For any number, to find the 2's complement scan the bits from Right side until the 1 comes and complement the remaining

01011010 → 10100110

→ 100101 → 011011

$$\begin{array}{r} 5 = 101 \\ +3 = +\ 011 \\ \hline 1000 = 8 \end{array}$$

$$\begin{array}{r} 16 = 10000 \\ +17 = 10001 \\ \hline 100001 = 33 \end{array}$$

$$\begin{array}{r} 16 = 10000 \\ +14 = 01110 \\ \hline 11110 = 30 \end{array}$$

→ $\begin{array}{r} 5 \quad 1\overset{2}{0}1 \\ -3 \quad -011 \\ \hline 010 \end{array}$ In arithematic subtraction carry is 2.

1's complement addition

$$\begin{array}{r} 101 \\ +\ 100 \text{ (1's complement of 3)} \\ \hline 001 \\ +\ 1 \\ \hline 010 \end{array}$$

carry should be added

→ $\begin{array}{r} -1 \quad 111\ 111 \\ -2 \quad 000\ 100 \\ \hline 100 \\ 1 \\ \hline 101 \end{array}$

1+1+1 = 1 with carry 1

$$\begin{array}{r} 25 \quad 11001 \\ -17 \quad -10001 \\ \hline \end{array}$$

$$\begin{array}{r} 11001 \\ 01110 \\ +\ 1 \\ \hline 01000 \\ +\ 1 \\ \hline 01001 \\ +\ 1 \\ \hline 01010 \\ +\ 1 \\ \hline 01011 \end{array}$$

$$\begin{array}{r} 1000 \\ +\ 1 \\ \hline 1001 \end{array} = 8$$

Scanned with CamScanner

```
  3    011     011        If there is no carry
- 5  - 101     010        then it's result is negative
              -----       and it is in it's
               101        1's complement form
               010
               = -2

  2    010     010
- 7    111     000        NO carry
              -----
               010        complement => 101
                                      => -5

 17    10001   10001
-25    11001   00110
              -------
               10111   => 01000
                          -8
```

2's complement addition

```
  5    101     101
- 3    011     100 (2's complement
             (1)
              010 = 2
              -> carry should be neglected in
                 2's complement addition

  7    111     111
- 2    010   (1)110
              -----
               101 = 5

 25    11001   11001
-17    10001 (1)01111
              -------
               01000 = 8

                          If there is no carry then
  3    011     011        Result is in negative
- 5    101     011
              -----       and in it's 2's
               110        Complement form
```

Scanned with CamScanner

2 010 010
-7 111 001

011 → ~~1010~~ = 5

101 = 5 No carry ⇒ -5

17 10001 10001
-25 11001 00111

1000 = -8

80 10100000 10100000
-70 1000110 01111010

00010100 ⇒
Neglected 8020 = 10

70 1000110 1000110
-80 1010000 0110000

1110110
0001010
= -10

126 1111110
-86 0010110 1111110
0101010

0101000
32+8 = 40

86 1010110 1010110
-126 1111110 0000010

1011000 0101000
(64+16+8) 32+8 = 40
-40 (No carry)

Scanned with CamScanner

Representation of Negative numbers

(8bits)

+14 = 0 0001110

-14 = 1 0001110 sign magnitude (S.M)

-14 = 1 1110001 → 1's complement

-14 = 1 1110010 → 2's complement

+0 = 0 000 0000

-0 = 1 000 0000 sign magnitude

-0 = 1 111 1111 → 1's complement

-0 = 0000 0000 → 2's complement

If the no: of bits n=8, only (n-1)=7 bits are only assigned and one bit is fixed for sign

n=8 +127 to -127 → in sign mag. and 1s

-128 to +127 → in 2's complement

2's complement is prefferred over 1's complement and sign magnitude because of the following reason

In order to represent +0 and -0 in sign magnitude we need two seperate representations, whereas in 2's complement only one representation is sufficient to represent +0 and -0, that means we can represent one extra number in 2's complement compared to 1's complement and sign magnitude

Scanned with CamScanner

n=4

-7 to +7 → sign magnitude and 1's complement

-8 to +7 → 2's complement

For 'n' bits,

$-(2^{(n-1)}-1)$ to $+(2^{(n-1)}-1)$ sign magnitude and 1's complement

$-2^{(n-1)}$ to $+(2^{(n-1)}-1)$ in 2's complement

whenever the MSB is 1, that indicates it is a negative number

1001 = −7 111001 = −7

11001 = −7 1111001 = −7

Represent +42 in 2's complement

a) 0 0101010 101010 (64 16 4 2) 1010 0.1010 (16 4 2)

b) 1 0101011

c) 1 1010110 01010 10 (32 8 2)

d) None

→ −15

1 000 1111

1 1110001

Weighted and non-weighted codes:

8	4	2	1	
0	0	0	0	-0
0	0	0	1	-1
0	0	1	0	-2
0	0	1	1	-3
0	1	0	0	-4
0	1	0	1	-5
0	1	1	0	-6
0	1	1	1	-7
1	0	0	0	-8
1	0	0	1	-9

2	4	2	1	
0	0	0	0	-0
0	0	0	1	-1
0	0	1	0	-2
0	0	1	1	-3
0	1	0	0	-4
0	1	0	1	-5
0	1	1	0	-6
0	1	1	1	-7
1	1	1	0	-8
1	1	1	1	-9

6	4	-3	2	
0	0	0	0	-0
0	1	1	0	-1
0	0	0	1	-2
1	0	1	0	-3
0	1	0	0	-4
1	0	1	1	-5
1	0	0	0	-6
1	1	1	0	-7
1	0	0	1	-8
1	1	1	1	-9

Self complementory code

9's complement is obtained by subtracting that particular number from 9 or 99 or 999...

based on the number of digits.

3 → 9 - 3 = 6 25 → 99 - 25

Scanned with CamScanner

A code is said to be self complementary code, if 9's complement is obtained, by interchanging 1's and 0's

2421 — self complementary code

6 4 -3 2 is also a self complementary code

8421 is not a self complementary code

BCD → Binary coded Decimal Number.

If a number is to be represented in BCD form then it should be represented in 4 bit form

25	89	125
0010 0101	1000 1001	0001 0010 0101

8421 — BCD code

Non-weighted codes

Excess-3 (~~Gray~~) → by adding 3 to Binary

Binary	Excess-3
0000	0011
0001	0100
0010	0101
0011	0110
0100	0111

Scanned with CamScanner

Boolean Algebra

Properties

$0.0 = 0$	$0+0 = 0$
$0.1 = 0$	$0+1 = 1$
$1.0 = 0$	$1+0 = 1$
$1.1 = 1$	$1+1 = 1$

1) $A.0 = 0$
2) $A.1 = A$
3) $A.A = A$
4) $A.\bar{A} = 0$
5) $\bar{\bar{A}} = A$
6) $A+0 = A$
7) $A+A = A$
8) $A+\bar{A} = 1$
9) $A+1 = 1$
10) $(A+B)(A+C) = A+BC$
11) $(A+\bar{A}B) = (A+\bar{A}).(A+B)$
$\Rightarrow 1.(A+B) = A+B$
12) $(\bar{A}+AB) = \bar{A}+B$

$A+B = B+A$

$A(B+C) = AB+AC$

1) $AB+A(B+C)+B(B+C)$
$AB+AB+AC+B.B+BC$
$AB+B+AC+BC$
$AB+B+BC+AC$
$AB+B(1+C)+AC$
$AB+AC+B$
$(A+1)B+AC$
$B+AC$

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

Scanned with CamScanner

3. $a'bc + ab'c + abc' + abc$

$a'bc + ab'c + ab$

~~$(c(a'b + ab') + ab)$~~

$a'bc + ab + ab'c$

$(\bar{a}c + a)b + ab'c$

$(a + c)b + ab'c$

$ab + bc + ab'c$

$ab + c(b + ab')$

$ab + c(a + b) = ab + ac + bc$

$a'bc + abc + ab'c + abc + abc' + abc$

$bc(a' + a) + ac(b' + b) + ab(c' + c)$

$bc + ac + ab$

4. $a + a'b + a'b'c + a'b'c'd + a'b'c'd'e + \ldots$

$[a + a'(b + b'c + b'c'd + \ldots)$

$a + a'(b + b'(c + c'd + c'd'e + \ldots))]$

$a + a'b + a'b'c + a'b'c'd + a'b'c'd'e + \ldots$

$a + b + a'b'c + a'b'c'd + a'b'c'd'e + \ldots$

$a + b + b'c + a'b'c'd + \ldots$

$a + b + c + a'b'c'd + \ldots$

$a + b + b'c'd + c + \ldots$

$a + b + c + c'd + \ldots$

$a + b + c + d + a'b'c'd'e + \ldots$

$\vdots$

$a + b + c + d + e \ldots$

Scanned with CamScanner

5. $(x+y)(x+yz)+x'y'+x'z'$

$x + xyz + xy + yz + x'y' + z'z'$

$x + xy + xyz + yz + x'y' + z'z$

$x(1+y) + yz(1+x) + x'y' + x'z'$

$x + yz + x'z' + x'y'$

$x + z' + yz + x'y'$

$x + y' + z' + y$

$x + 1 + z'$

$= 1 + z'$

$\Rightarrow 1$

6. $a'bc' + a'b'c + a'bc + ab'c + abc$

$a'b' + c + a'bc + ab'c + abc$

$a'b' + c + a'bc + ac$

$a'b' + c(1 + a'b + a)$

$a'b' + c$

$a'b' + a'bc + ac$

$a'b' + ac + bc \rightarrow$ Irredundant expression

(or)

$a'b'c' + a'c + ac$

$a'b'c' + c$

$c + a'b'$

Consensus Theorem

$xy + x'z + yz = xy + x'z$

Proof: $xy + x'z + yz(x + x')$

$xy + x'z + xyz + x'yz$

$xy + x'z$

$\Rightarrow xy + x'zp + yzp = xy + x'zp$

Scanned with CamScanner

→ $xy + x'y' + yz = xy + x'y' + x'z$ (prove)

$xy + yz + x'y' + x'z$

$= xy + x'y' + x'z$

→ $xy + xy'z' + wxz' = xy + y'z'$

$xz' + xy + y'z' + wxz' \Rightarrow xy + y'z' + xz'$

$= xy + y'z'$

→ $a'c' + abd + bc'd + ab'd' + abcd' = a'c' + abd + ab'd' + acd$

$(a'c' + abd + c'bd + bc'd)$

$a'c' + abd + ab'd + abcd'$

$a'c' + abd + ab'd + abcd' + abc$

$a'c' + abd + adc + abcd' + ab'd$

$a'c' + ad + adc$

$a'c' + ad + c'd + adc$

$a'c' + ad + c'd + adc + ad$

$a'c' + ad + c'd + adc$

$abd + abcd'$

ac

$abc \quad ad + abcd'$

abc

$ad + abcd'$

$abc \quad a$

$a'c' + abd + bc'd + ab'd' + abcd' = a'c' + abd + ab'd' + acd$

$a'c' + abd + ad'(b' + bc)$

$a'c' + abd + ad'(b' + c)$

$a'c' + abd + ab'd' + acd'$

Scanned with CamScanner

→ $ab'c + a'bc' + bc$ ~~a+b~~

$ab'c + bc + ac + a'bc'$

$ab'c + bc + ac + a'bc'$

$c(a+b) + ac + a'bc'$

$ac + bc + a'bc'$

$ac + bc + a'bc' + a'b$

$ac + bc + a'b$

$ac + a'b$

→ $ab'c + a'bc' + ab$

$ab'c + ab + a'bc'$

$ab'c + ab + ac + a'bc'$

$a(c+b) + ac + a'bc'$

$ac + ab + a'bc'$

$ab + ac + a'bc' + a$

$ab'c + b(a'c' + a)$

$ab'c + bc' + ab$

$a(b + b'c) + bc'$

$a(b+c) + bc'$

$ab + ac + bc'$

$ac + bc'$

→ $ab'c + abc + a'c$

$a'b'c + c(ab + a')$

$a'b'c + c(b + a')$

$a'b'c + bc + a'c$

$a'b'c + bc$

$c(a'b' + b)$

$a'c + bc$

Scanned with CamScanner

1. $AB'C' + A'B'C' + A'BC' + A'B'C$

$B'C'(A + A') + A'BC' + A'B'C$

$B'C' + A'BC' + A'B'C$

$A'BC' + B'C' + A'C' + A'B'C$

$A'C' + B'(C' + A'C)$

$A'C' + B'(C' + A')$

$A'B' + A'C' + B'C'$

2. $ABC + A'BC + AB'C + ABC' + AB'C' + A'BC' + A'B'C'$

$BC + AB'C + ABC' + AB'C' + A'BC' + A'B'C'$

$BC + AB'C + ABC' + A'BC' + B'C'$

$BC + AB'C + B'C' + B'C'$

$BC + AB'C + C'$

$BC + AB' + C'$

$B + AB' + C'$

$A + B + C'$

3. $AB'C(BD + CDE) + AC'$

$AB'CD(B + CE) + AC'$

$AB'CDE + AC'$

$A(B'CDE + C')$

$A(B'DE + C')$

$AB'DE + C'A$

Scanned with CamScanner

Demorgan's Laws

$\overline{ABC} = \bar{A} + \bar{B} + \bar{C}$

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

4. $AB(C+BD')(AB)'$

$AB(C+BD')(A'+B')$

$(ABC+ABD')(A'+B')$

$AB(C+D')(A'+B')$

$AB(A'C+B'C+A'D'+D'B') = 0$

5. $A'B'C + A'B'C + A'B'C'D'$

$A'B'C + A'B'C'D'$

$A'B'C + A'B'D'$

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

6. $AB'C + (B'+C')(B'+D') + (A+C+D)'$

$AB'C + B' + B'D' + C'B' + C'D' + A'C'D'$

$AB'C + B' + C'B' + C'D' + A'C'D'$

$AB'C + B' + C'B' + C'D'$

$B'(AC+1)$

$B' + C'B' + C'D'$

$B' + C'D'$

Scanned with CamScanner

30/7/19

Minterm: A minterm is a product term which contains all the 'n' variables in complemented or uncomplemented form

In Minterm 0 is represented in complemented form
1 → uncomplemented form

Sum of products (SOP): sum of product terms

Min terms

Eg: $x'yz + xyz' + xyz$

Maxterm: A max term is a sum term which contains all the 'n' variables either in complemented or uncomplimented form.

0 → uncomplemented form
1 → complemented form

Product of sum (POS): product of sum terms

Maxterm

Eg: $(x+y+z)(x'+y'+z')(x'+y'+z)$

a b c	Minterm	Maxterm
0 0 0	$a'b'c'$	$a+b+c$
0 0 1	$a'b'c$	$a+b+c'$
0 1 0	$a'bc'$	$a+b'+c$
1 0 0	$ab'c'$	$a'+b+c$
0 1 1	$a'bc$	$a+b'+c'$
1 1 0	abc'	$a'+b'+c$
0 0 1	$ab'c$	$a'+b+c'$
1 1 1	abc	$a'+b'+c'$

Canonical SOP: Here, each product term should contain all the variables.

$x'y + xz' + yz$ (Not a canonical form)

$= x'y(z+z') + xz'(y+y') + yz(x+x')$

converting into canonical form

$= x'yz + x'yz' + xyz' + xy'z' + xyz + x'zy$

Redundant form

$= x'yz + x'yz' + xyz' + xy'z' + xy$

canonical form

Scanned with CamScanner

Canonical POS : Each sum term should contain all the variables.

$(x+y)(x+z')(y'+z)$ → Not a canonical form

$(\underbrace{x+y}_{A}+\underbrace{z\cdot z'}_{BC})(x+z'+yy')(y'+z+xx')$ } converting into canonical form

$A+BC = (A+B)(A+C)$ }

$=(x+y+z)(x+y+z')(x+y+z')(x+z'+y')$

$(y'+z+x)(y'+z+x')$

↓

canonical form

Logic Gates:

Basic gates : AND, OR, NOT

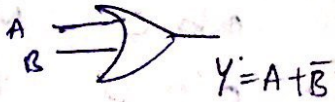
1. NOT (Inverter)

A	Y $\bar{A}$
0	1
1	0

2. AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

3. OR



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Universal Gates; NAND, NOR

1. NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

2. NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Scanned with CamScanner

When all the inputs in AND gate is high the output is high

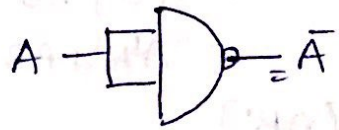
when atleast one inputs in OR gate is high the output is high

The output of NAND gate is high, when atleast one input is low

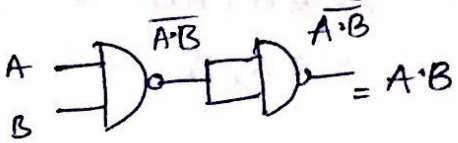
The output of NOR gates high, when all the inputs are low.

→ NAND as universal gate

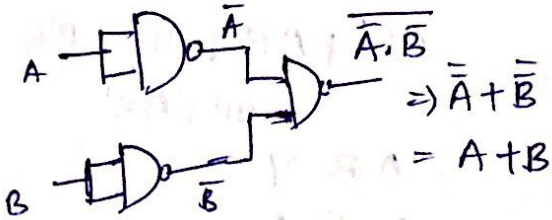
1) NOT



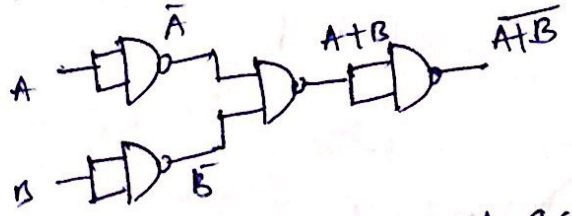
2) AND



3) OR

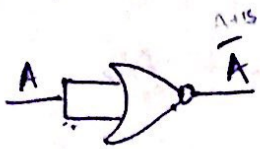


4) NOR

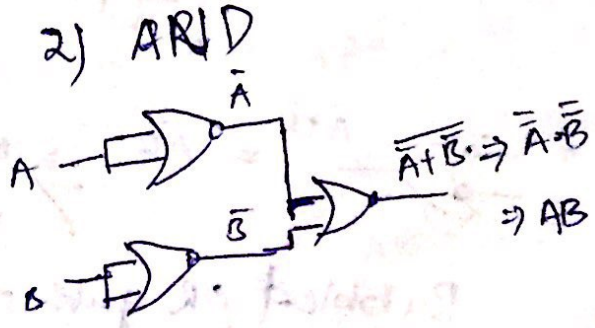


→ NOR as universal gate

1) NOT



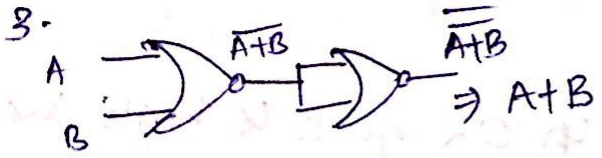
2) AND



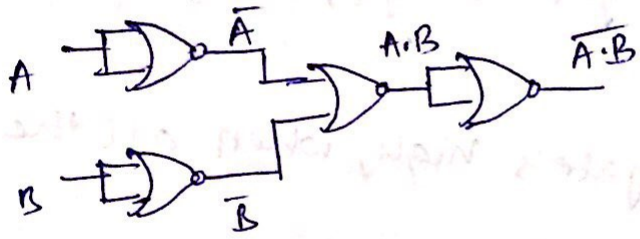
Scanned with CamScanner

OR

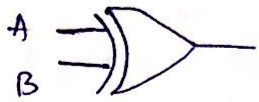
3.



4. NAND



Ex-OR gate

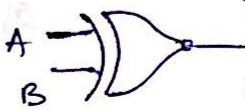


$Y = \bar{A}B + A\bar{B}$

$= A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR gate



$\overline{A \oplus B}$

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

$= A'B' + AB$

$(A'B)'(AB')'$

$(A+B')(A'+B)$

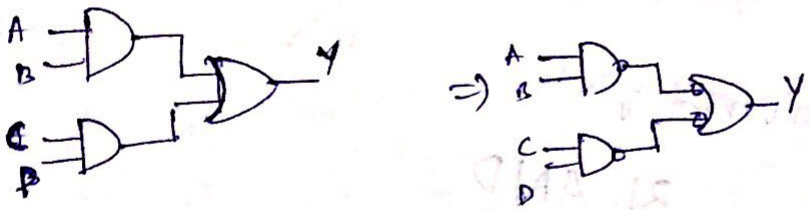
$AA' + AB + A'B' + B'B$

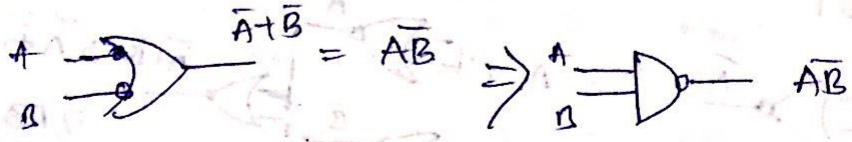
$= AB + A'B'$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

AND-OR → NAND-NAND

$Y = AB + CD$

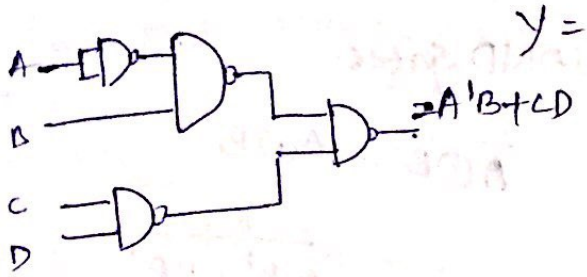




Bubbled OR gate = NAND gate

Scanned with CamScanner

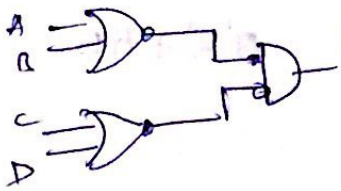
$y = A'B + CD$

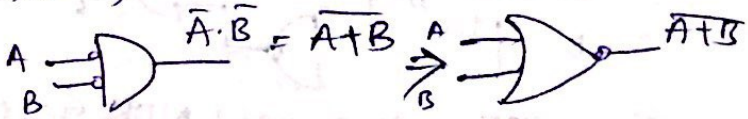


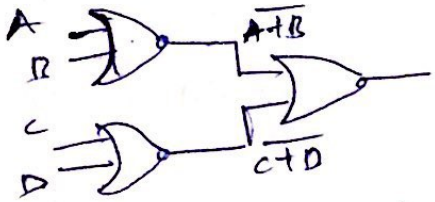
OR-AND → NOR-NOR

$F = (A+B)(C+D)$

Bubbled AND gate = NOR gate



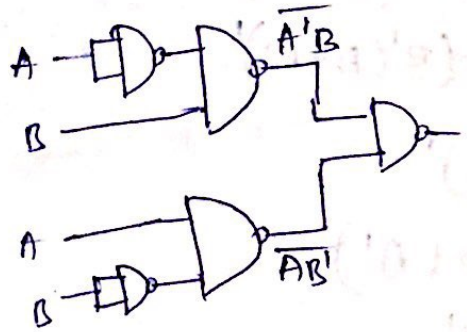




$\overline{\overline{A+B} + \overline{C+D}}$

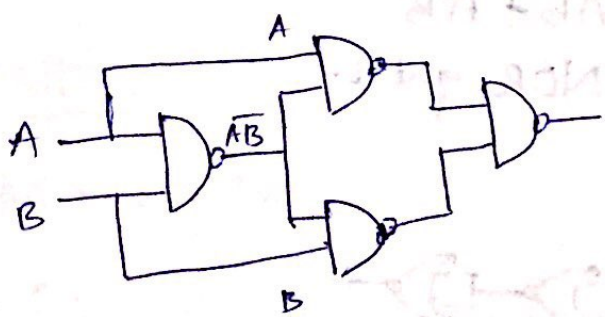
$(A+B)(C+D)$

Ex-OR gate using only NAND gate



$\overline{\overline{A'B}\cdot\overline{AB'}}$

$A'B + AB'$



$\overline{A\cdot\overline{AB}} \Rightarrow \bar{A} + AB$

$\overline{\overline{AB}\cdot B} = \bar{B} + AB$

$\overline{(\bar{A}+AB)(\bar{B}+AB)}$

$\overline{(\bar{A}+AB)} + \overline{(\bar{B}+AB)}$

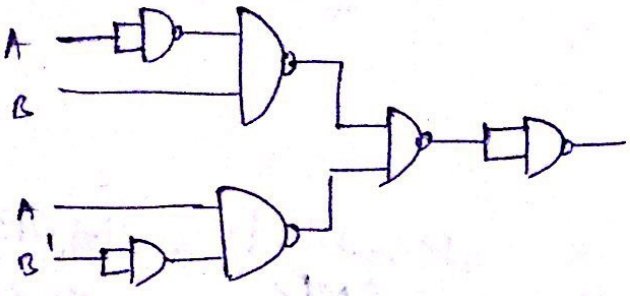
$(A'+B)' + (B'+A)'$

$AB' + A'B$

Scanned with CamScanner

Ex-NOR using only NAND gates.

$\overline{A \oplus B} = A \odot B$

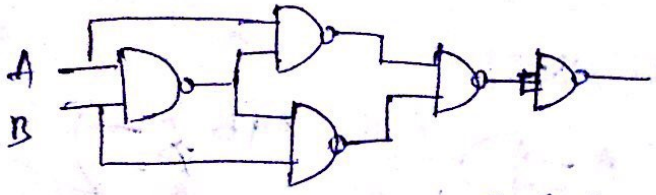


$\overline{AB' + BA'}$

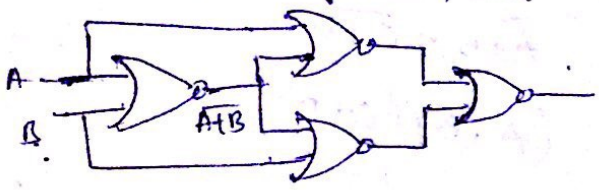
$(AB')'(BA')'$

$(A'+B)(B'+A)$

$A'B' + AB$



Ex-NOR using only NOR Gates



$\overline{A + \overline{A+B}} \cdot \Rightarrow (A') \cdot (\overline{A+B})' \quad B'(A+B)$

$\Rightarrow A'(A+B) + B'(A+B)$

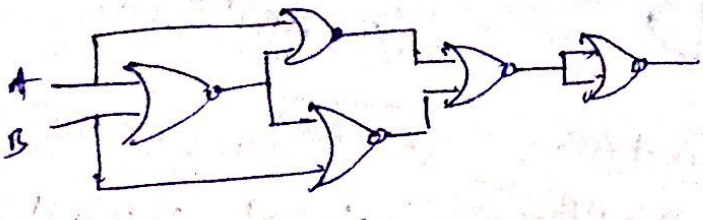
$\Rightarrow (A'(A+B))'(B'(A+B))'$

$(A'B)'(B'A)'$

$(A+B')(B+A')$

$AB + A'B'$

Ex-OR using only NOR gates.

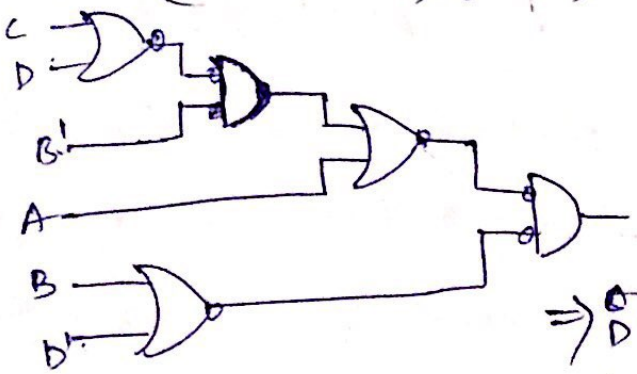


Scanned with CamScanner

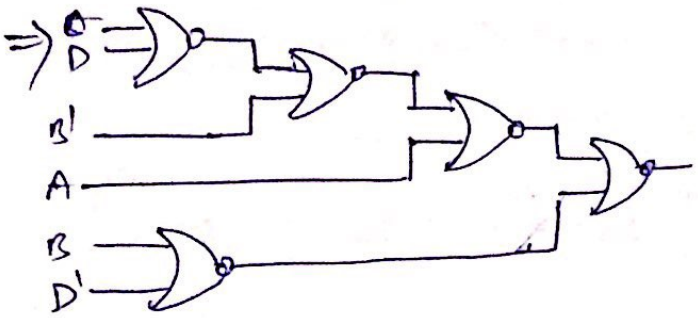
Multilevel NOR/NAND

1) NOR

$F = (A + B(C+D))(B+D')$

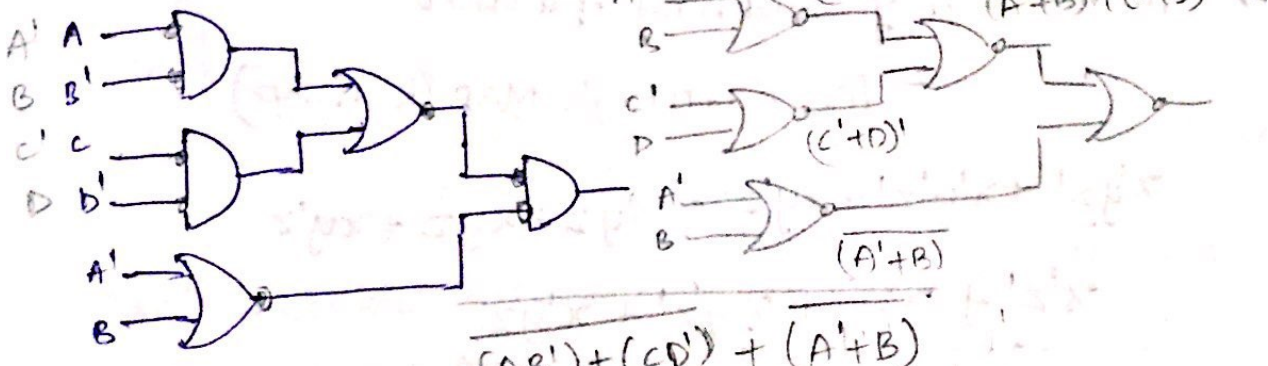


Bubbled AND = NOR



3-level NOR

$(AB'+CD')(A'+B)$



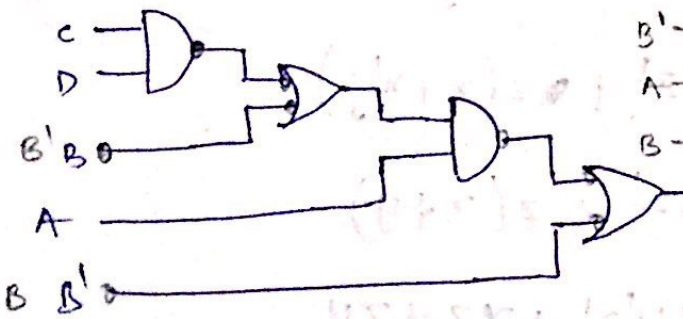
$\overline{(AB')+(CD')} + \overline{(A'+B)}$

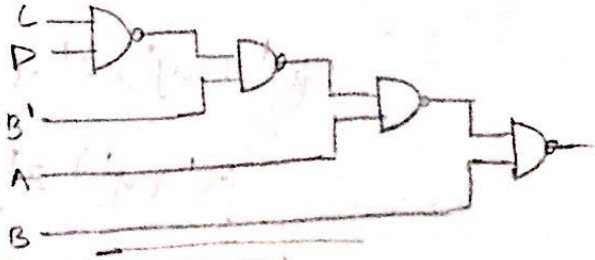
$(AB'+CD')(A'+B)$

1/8/19

2) NAND

$A(B+CD)+B'$





$\left(\overline{\overline{CD}B'}\right)'A\,B$

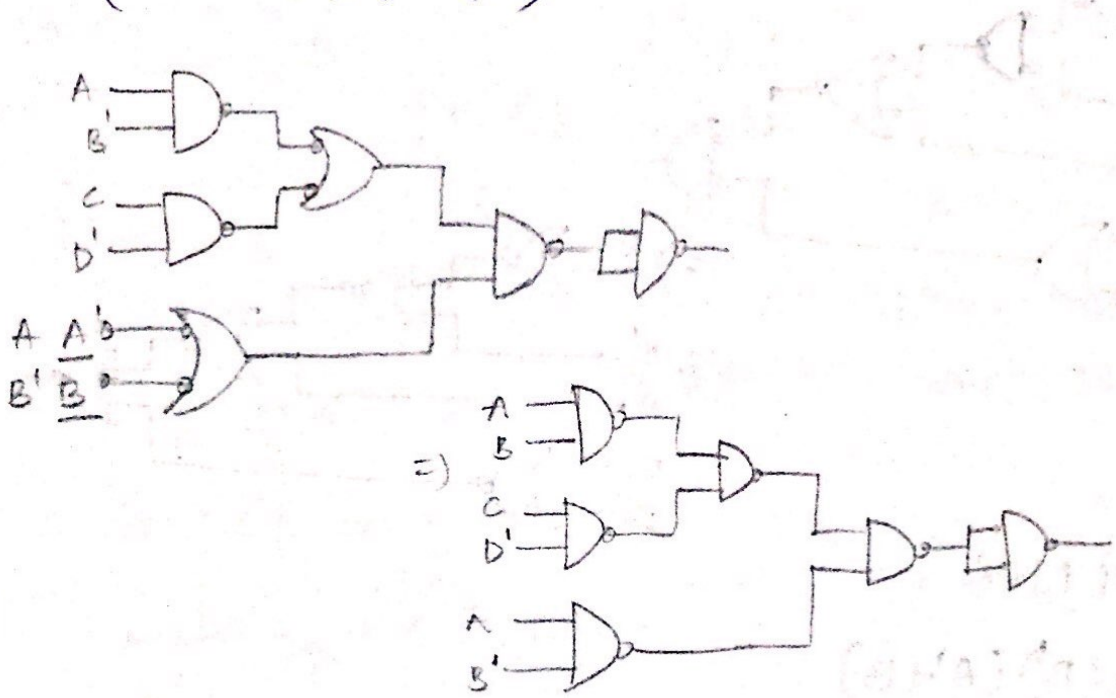
$(\overline{CD}B')'A + B'$

$(B+CD)A + B'$

Scanned with CamScanner

Mixed level NAND

$(AB'+CD')(A'+B)$



⇒ Minimisation of Switching function

- The Karnaugh Map (K-map)

$x'y'z' + x'y'z' + xy'z' + x'yz + xyz + xy'z$

$x'z' + xz + xy'z' + x'yz$

$x'x' + xy'z' + y'z' + xz + x'yz + yz$

$y'z' + yz + x'z' + xz$

$(x' + xy')z' + xz + x'yz$

$(x' + y')z' + xz(x + x'y)$

$(x' + y')z' + z(x + y)$

$x'z' + y'z' + xz + zy$

Scanned with CamScanner

② - $x'z' + xy' + yz$

③ - $x'y + y'z' + xyz$

②&③ → Minimal Expressions.

①, ②, ③ → Redundant expressions

An irredundant expression need not be minimal

A minimal expression may not be unique

K-Map

No. of cells $= 2^n$ where n = no. of variables.

$n = 4 \Rightarrow 2^n = 16$ cells.

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

In 'n' variable K-map each cell will be adjacent to 'n' other cells.

5 → 1, 4, 13, 7 ; 0 → 1, 4, 2, 8 ; 9 → 8, 13, 11, 1

Cell combinations will in powers of 2 $\Rightarrow 2^n = 2, 4, 8$

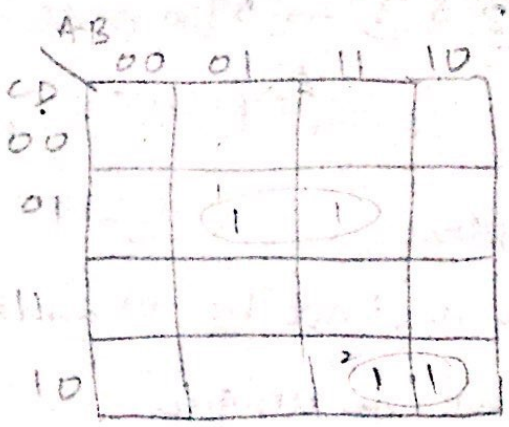
2 cells → pair ; 4 cells → quard ; 8 cells → octet

We can form a pair if they are adjacent.

Min terms in the K-map are represented by 1

Scanned with CamScanner

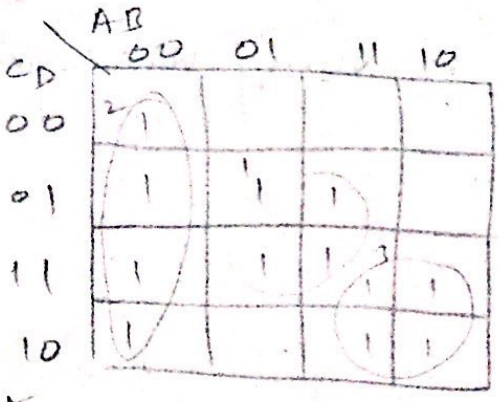
pair



when two adjacent cells are taken. The changing variable should be eliminated.

1. $BC'D$
2. ACD'

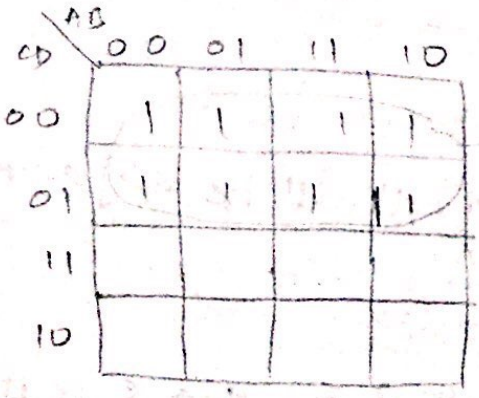
quad



In quad, each cell should be adjacent to two other cells which are considered.

1. BD
2. $A'B'$
3. AC

octet



→ C'

Is at four corners can also form a quad

Among the collection of octet each cell should be adjacent to 3 other cells

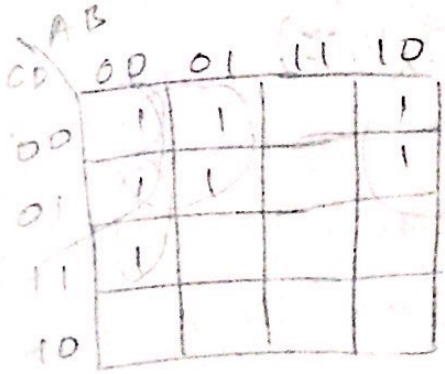
A collection of 2^m cells results in a product term which contains $(n-m)$ variables, where n is the no. of variables.

2^1 = pair $\Rightarrow 4-1 = 3$

2^2 = quad $\Rightarrow 4-2 = 2$

2^3 = octet $\Rightarrow 4-3 = 1$

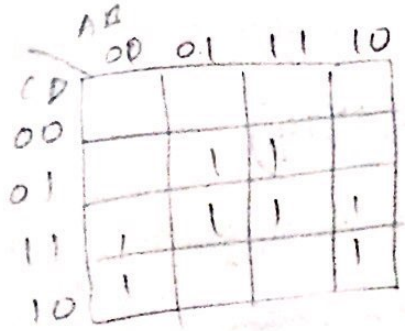
Scanned with CamScanner



then quads, if any remo
makit a pair

$F = A'C' + B'C' + A'B'D$

→



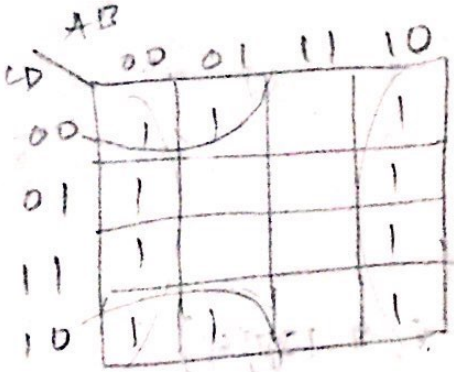
$BD + B'C$

Each minterm must be covered atleast once.

Always try to combine less no. of sub cubes.

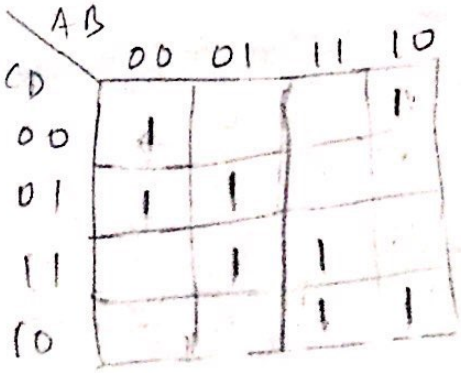
Each sub cube must be as large as possible.

→ $F(ABCD) = \Sigma(0,1,2,3,4,6,8,9,10,11)$



$B' + A'D'$

→ $= \Sigma(0,1,5,7,8,10,14,15)$

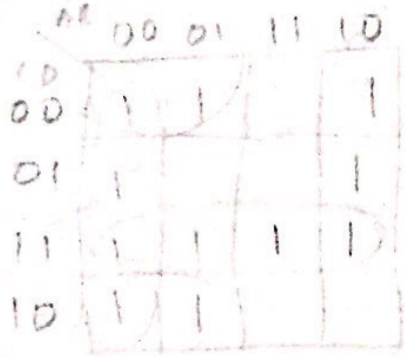


$= A'B'C' + A'BD + ABC + AB'C'$

$= A'C'D + BCD + ACD' + B'C'D'$

Scanned with CamScanner

→ $= \Sigma(0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 15)$

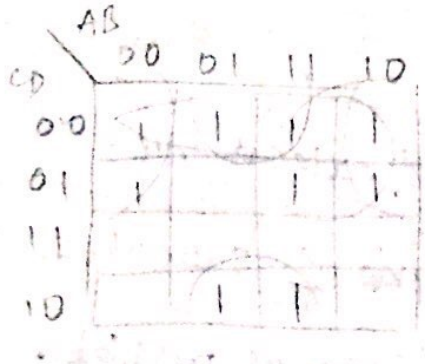


$A'B' + CD + A'D' + B'C'$

↓

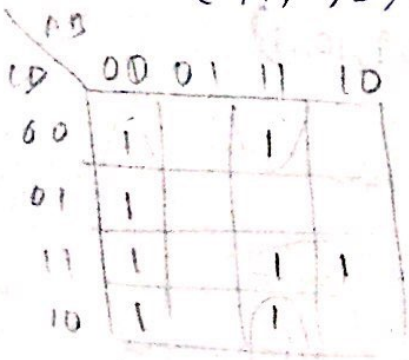
Redundant

→ $= \Sigma(0, 1, 4, 6, 8, 9, 12, 13, 14)$



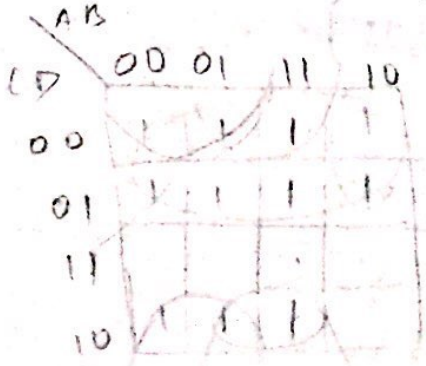
$B'C' + BD' + AC'$

→ $= \Sigma(0, 1, 2, 3, 11, 12, 14, 15)$



$A'B' + ACD + ABD'$

→ $= \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



$C' + A'D' + BD'$

Scanned with CamScanner

6/8/19

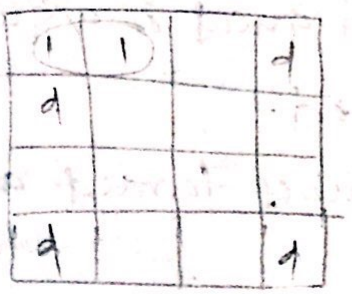
Don't cares

Represented by d ϕ x

If it is useful in simplification consider it or else don't care (neglect it) it.

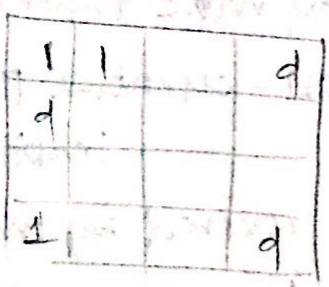
It is considered as minterm

Note: All minterms must be covered, don't cares may or may not be considered.



Here without quad also all minterms can be covered in pair so the answer is only 1 pair

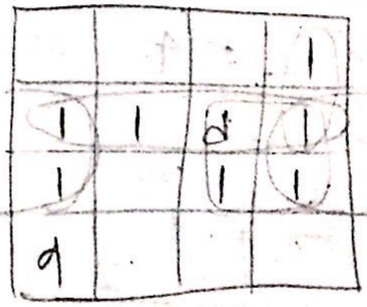
1 pair



1 quad, 1 pair ✓

2 pairs

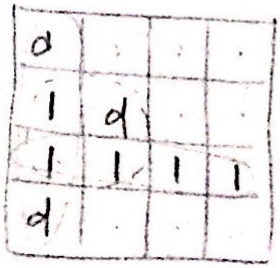
→ $f(abcd) = \Sigma(1,3,5,8,9,11,15) + d(2,13)$



3 quads

1 pair

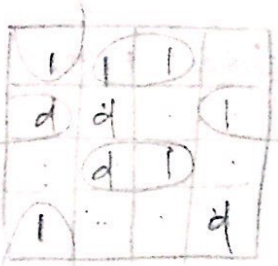
⇒ $f(abcd) = \Sigma(1,3,7,11,15) + d(0,2,5)$



2 quads

Scanned with CamScanner

$= \Sigma(0,2,4,9,12,15) + d(1,5,7,10)$



4 pairs

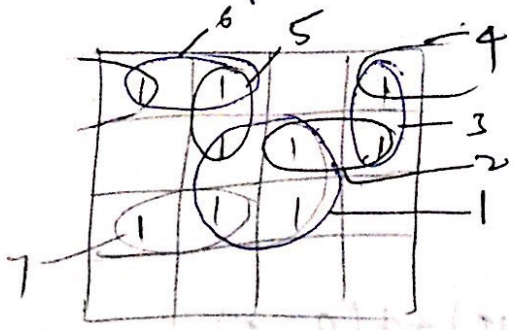
Essential Prime implicants

All the adjacent cell are combined wherever it is possible but a pair in quad or a quad in a octet are not considered.

Now what ever the (combinations ~~subcubes~~) formed are called prime implicants.

If each (all) ~~min~~term is covered by more than one prime implicant, then it is non-essential prime implicants.

If any of min terms are not covered by more than one prime implicant, then it is essential prime implicant.



7, 1 – Essential prime implicant

2, 3, 4, 5, 6 – non-essential prime implicants

Scanned with CamScanner

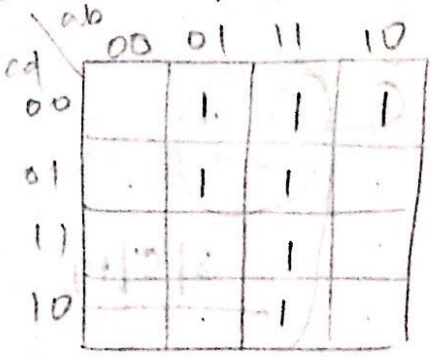
Essential prime implicant: 7/8/19

A prime implicant of a function F is said to be Essential prime implicant if it covers atleast any minterm which is not covered by any other prime implicant.

Procedure to find the Minimal expression

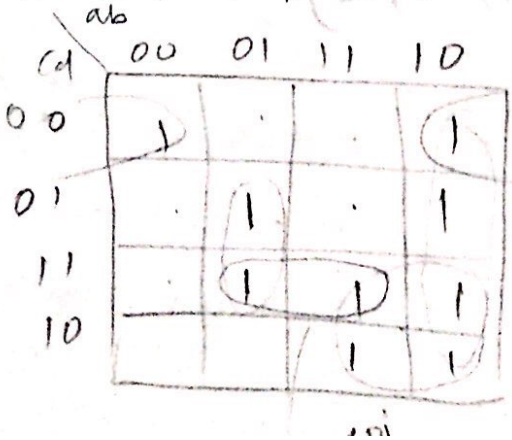
1. Identify all the essential prime implicants and include them in the Minimal expression.
2. If the essential prime implicant cover all the minterms then the expression will be unique.
3. otherwise choose minimum no. of additional prime implicants so that all the minterms are covered

1. $f(a,b,c,d) = \Sigma(4,5,8,12,13,14,15)$



$ab + bc' + acd'$

2. $\Sigma(0,5,7,8,9,10,11,14,15)$

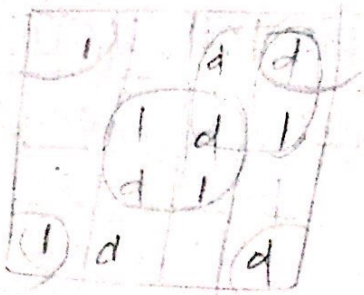


5 pi

4 epi

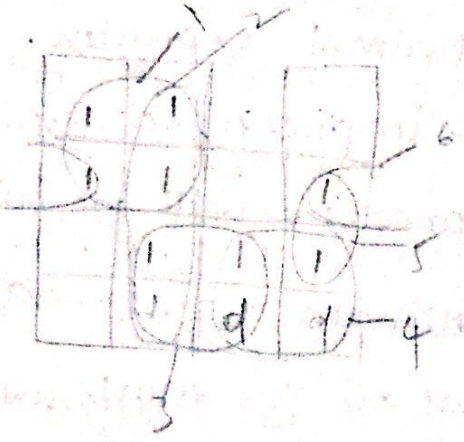
Scanned with CamScanner

3. $= \Sigma(0,2,5,9,15) + d(6,7,8,10,12,13)$



- 3 quads

4. $= \Sigma(0,1,4,5,6,7,9,11,15) + d(10,14)$

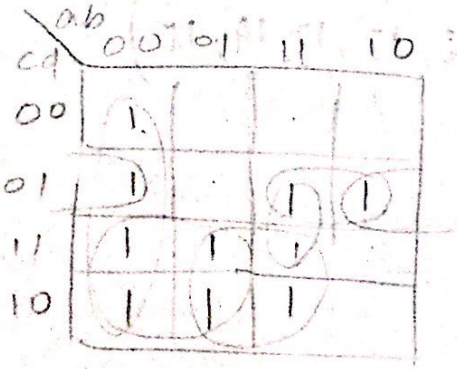


6 PI

1 epi

1 + 3 + 5 ⇒ minimal expression

$= \Sigma(0,1,2,3,6,7,9,13,14,15)$.



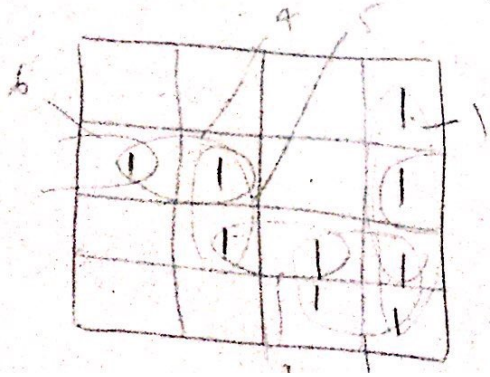
6 PI

2 epi

$a'b' + bc + c'da$

8|8|19

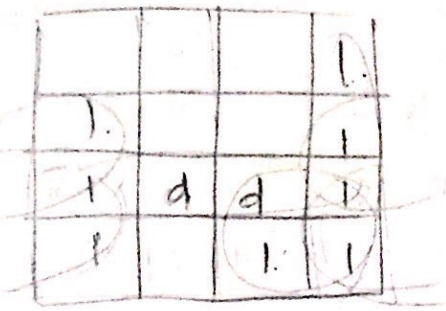
$= \Sigma(1,5,7,8,9,10,11,14,15)$ = (1) + (2)



In the minimal expression all the essential prime implicants must be covered.

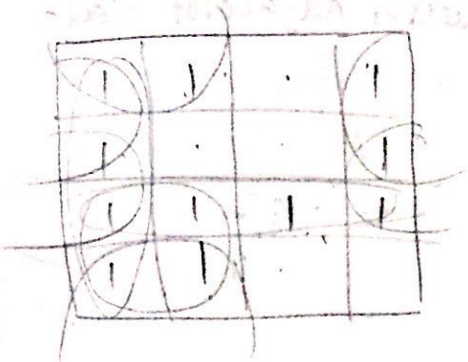
Scanned with CamScanner

$= \Sigma(1,2,3,8,9,10,11,14) + d(7,15)$



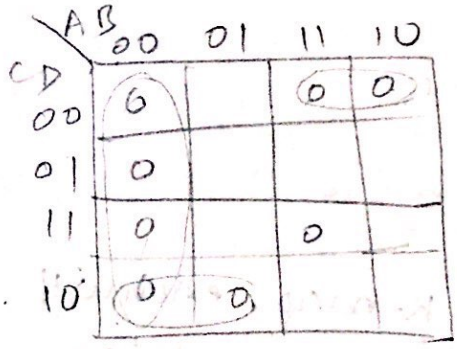
5 pi
4 epi

$= \Sigma(0,1,2,3,4,6,7,8,9,11,15)$



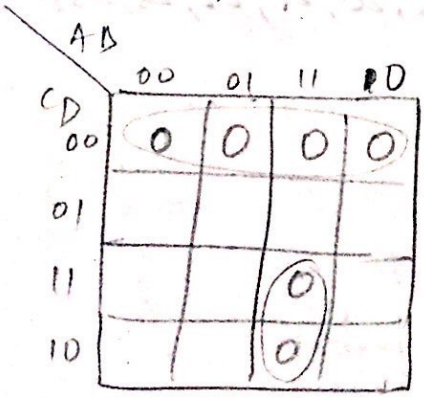
6 quads
3 epi

Product of Sums



$(A+B)\cdot(A'+C+D)\cdot(C'+D+A)\cdot(A'+B'+C'+D')$

$f(ABCD) = \pi(0,4,8,12,14,15)$



$(C+D)\cdot(A'+B'+C')$

Scanned with CamScanner

$= \pi(0,1,2,3,8,9,10,11)$

$\hookrightarrow = \Sigma(4,5,6,7,12,13,14,15)$

CD \ AB	00	01	11	10
00	0			0
01	0			0
11	0			0
10	0			0

If the max terms are given then the remaing cells are contain maxterms

In K-map cells are adjusted such that one variable changes when considered with adjacent cell.

Five variable K-map

$2^5 = 32 \Rightarrow$ cells

vwxyz

v=0

yz \ wx	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

v=1

yz \ wx	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

For each cell in 5 variable K-map, each will be adjacent to other 5 cells.

9/8/19

$+(vwxyz) = \Sigma(4,5,6,7,13,15,20,21,22,23,25,27,29$

v=0

yz \ wx	00	01	11	10
00		1		
01		1	1	
11		1	1	
10		1		

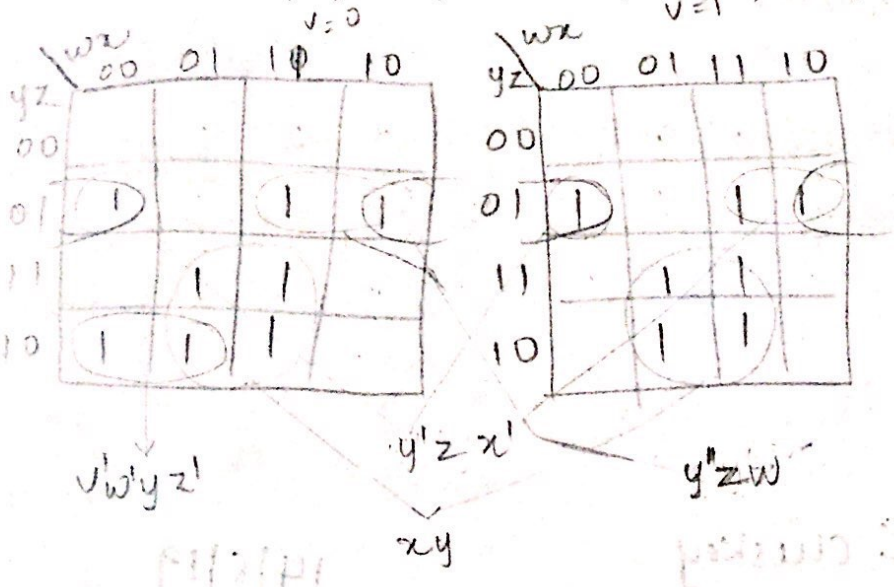
v=1

yz \ wx	00	01	11	10
00		1		
01		1	1	1
11		1	1	1
10		1		

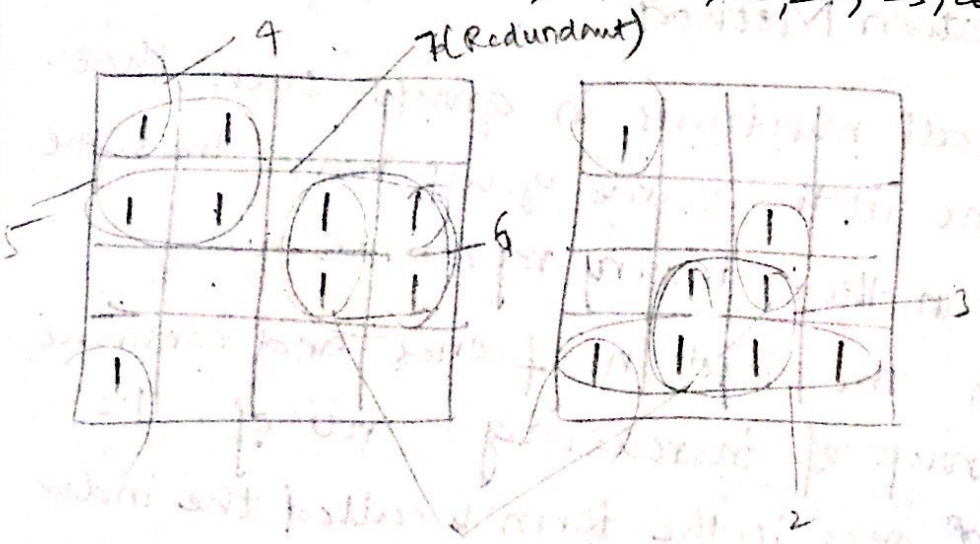
$w'x + xz + vwz$

Scanned with CamScanner

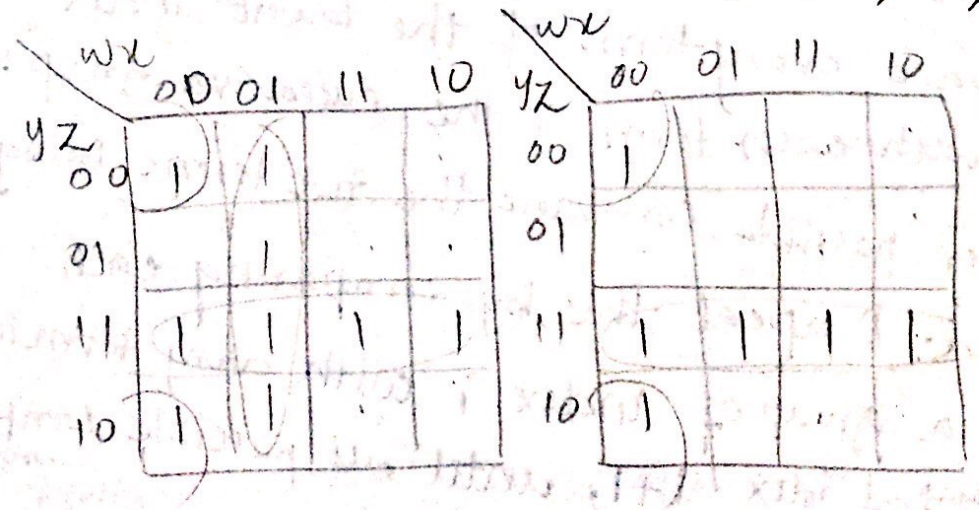
$= \Sigma(1, 2, 6, 7, 9, 13, 14, 15, 17, 22, 23, 25, 29, 30, 31)$



$= \Sigma(0, 1, 2, 4, 5, 9, 11, 13, 15, 16, 18, 22, 23, 26, 29, 30, 31)$



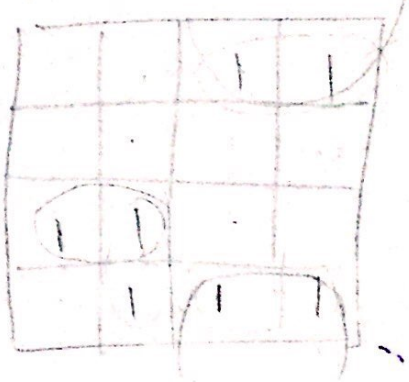
$= \Sigma(0, 2, 3, 4, 5, 6, 7, 11, 15, 16, 18, 19, 23, 27, 31)$

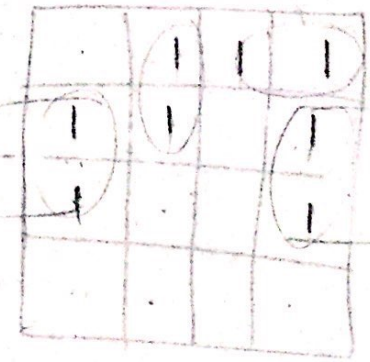


$yz + w'xv' + w'x'z'$

Scanned with CamScanner

$= \Sigma(3, 6, 7, 8, 10, 12, 14, 17, 19, 20, 21, 24, 25, 27, 28$





3 quads
3 pairs.

Quine MC Cluskey or Tabulation Method

14/8/19

Step 1. Arrange all minterms in groups, such that all terms in the same group have the same no. of 1's in their binary representation. Start with the least no. of ones and continue with groups of increasing no. of 1's. The no. of ones in the term is called the index of the term.

Step 2. Compare every term of the lowest index group with each term in the successive group: whenever possible, combine the two terms being compared. Repeat this by comparing each term in a group of index i with every term in the group of idex $i+1$, until all possible combinations are completed.

Scanned with CamScanner

Step 3. The terms generated in step-2 are now compared in the same fashion. The process continues until no further combinations are possible the remaining unchecked terms constitutes the set of prime implicants of the function.

→ $f(wxyz) = \Sigma(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)$

①	②	
0 ✓	(0,1) (1) ✓	
1 ✓	(0,2) (2) ✓	
2 ✓	(0,8) (8) ✓	
8 ✓	(1,5) (4) ✓	
5 ✓	(1,9) (8) ✓	→ Weight of the variable, that is being eliminated when the terms are combined
9 ✓	(2,10) (8) ✓	
10 ✓	(8,9) (1) ✓	Minterms are combined such that their difference is powers of 2's
7 ✓	(8,10) (2) ✓	
13 ✓	(5,7) (2) ✓	
15 ✓	(5,13) (8) ✓	
	(9,13) (4) ✓	
	(7,15) (8) ✓	
	(13,15) (4) ✓	

2^3 2^2 2^1 2^0
8 4 2 1
w x y z

③
A 0, 1, 8, 9 (1, 8) → (0,1) & (8,9) are combined because the difference blw 0;8 and 1,9 is same and is equal to 8 → also have equal weights

B 0, 2, 8, 10 (2, 8)

C 1, 5, 9, 13 (4, 8)

D 5, 7, 13, 15 (2, 8)

If the combinations selected contain the terms which are not in increasing order then that combination already exists

These can not be further combined

In ① ② ③ columns the the unchecked terms are the prime implicants.

Scanned with CamScanner

A 0, 1, 8, 9 (1,8) - 0 — $w'x'y'z' = x'y'$

B 0, 2, 8, 10 (2,8) — 2 — $w'x'yz' = x'z'$

C 1, 5, 9, 13 (4,8) - 9 — $wx'y'z = y'z$

D 5, 7, 13, 15 (2,8) - 15 — $wxyz = xz$

	0	1	2	5	7	8	9	10	13	15
A	x	x				x	x			
B	x		(x)			x		(x)		
C		x		x			x		x	
D				x	(x)				x	(x)

$F = B + D + A$

$= B + D + C$

→ $f(vwxyz) = \Sigma(13, 15, 17, 18, 19, 20, 21, 23, 25, 27, 29, 31) + d(1, 2, 12, 24)$

1 ✓ (1, 17) (16)

2 ✓ (2, 18) (16)

12 ✓ (12, 13) (1)

17 ✓ (17, 19) (2) ✓

18 ✓ (17, 21) (4) ✓

20 ✓ (17, 25) 18) ✓

24 ✓ (18, 19) (1)

13 ✓ (20, 21) (1)

19 ✓ (24, 25) (1)

21 ✓ (13, 15) 12) ✓

25 ✓ (13, 29) (16) ✓

15 ✓ (19, 23) 14) ✓

23 ✓ (19, 27) (8) ✓

27 ✓ (21, 23) (2) ✓

29 ✓ (21, 29) (8) ✓

31 ✓ (25, 27) (2) ✓

(25, 29) (4) ✓

(15, 31) (16) ✓

(23, 31) 18) ✓

(27, 31) (4) ✓

(29, 31) (2) ✓

Scanned with CamScanner

17, 19, 21, 23 (2,4) ✓
17, 19, 25, 27 (2,8) ✓
17, 21, 25, 29 (4,8) ✓

15, 15, 29, 31 (2, 16)
19, 23, 27, 31 (4, 8) ✓
21, 23, 29, 31 (2, 8) ✓
25, 27, 29, 31 (2, 4) ✓

17, 19, 21, 23, 25, 27, 29, 31 (2, 4, 8)

20 | 8 | 19

(16) (16) (1) (1) (1) (1)
(1, 17) (2, 18) (12, 13) (18, 19) (20, 21) (24, 25) B
E E D C
(2,16)
9 (13, 15, 29, 31) A ⟩ prime implicants.

(1, 17) — w'x'y'z

	13 ✓	15 ✓	17 ✓	18 ✓	19 ✓	20 ✓	21 ✓	23	25 ✓	27	29 ✓	31 ✓
✓A	×	(×)									(×)	(×)
✓B									(×)			
✓C						(×)	(×)					
✓D				×	(×)							
E	×											
F				×								
✓G			(×)									

Scanned with CamScanner

$f(vwxyz) = \Sigma(0,1,2,8,9,15,17,21,24,25,27,31)$

0 ✓	0,1 (1) ✓	0,1,8,9 (1,8) C
1 ✓	0,2 (2) H	1,9,17,25 (8,16) B
2 ✓	0,8 (8) ✓	8,9,24,25 (1,16) A
8 ✓	1,9 (8) ✓	
9 ✓	1,17 (16) ✓	
17 ✓	8,9 (1) ✓	
24 ✓	8,24 (16) ✓	
21 ✓	9,25 (16) ✓	
25 ✓	17,21 (4) G	
15 ✓	17,25 (8) ✓	
27 ✓	24,25 (1) ✓	
31 ✓	25,27 (2) F	
	15,31 (16) E	
	27,31 (4) D	

vwxyz
1100φ A = wx'y'

01000 v'w'y' = C

11X11

vwyz = D

	0	1	2	8	9	15	17	21	24	25	27	31
✓A				X	X				(X)	X		
B		X			X		X			X		
C	X	X		X	X							
D											X	X
✓E						(X)						X
F										X	X	
✓G							X	(X)				
✓H	X		(X)									

Scanned with CamScanner

21/8/19

	10	11	18	19	26
C			X	X	X
D			X	X	
E	X	X			
F	X	X		X	
G			X		X
H		X			
I		X			

Here all the minterms of D are covered in C so the row D can be eliminated. Wherein 10 and 11 columns are considered 11 can be eliminated because in order to cover the elements of 11 E, F, G and I should be taken.

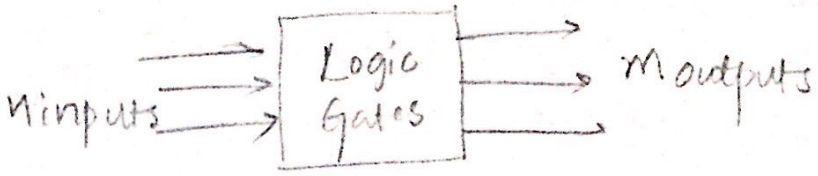
Digital circuits

- Combinational logic circuits
- Sequential logic circuits

a) Code convertors
b) Adders
c) Subtractors
d) Decoders
e) Comparators
f) Encoders
g) Multiplexers
h) Demultiplexers

Scanned with CamScanner

Any combinational logic circuit will have n inputs and m outputs



1. Code convertors: combinational code convertors converts the one code in to other codes.

BCD to Excess-3 code converter.

B_3	B_2	B_1	B_0		E_3	E_2	E_1	E_0
0	0	0	0	—	0	0	1	1
0	0	0	1	—	0	1	0	0
0	0	1	0	—	0	1	0	1
0	0	1	1	—	0	1	1	0
0	1	0	0	—	0	1	1	1
0	1	0	1	—	1	0	0	0
0	1	1	0	—	1	0	0	1
0	1	1	1	—	1	0	1	0
1	0	0	0	—	1	0	1	1
1	0	0	1	—	1	1	0	0

$E_3 = f(B_3\ B_2\ B_1\ B_0) + d(10-15)$

$= \Sigma(5,6,7,8,9)$

↓

E_3 is 1 at 5,6,7,8,9

Since the inputs are 4 the no of possible ways are 16. We are considering only upto 9, so the remaining are taken as don't cares.

Scanned with CamScanner

$E_2 = f() = \Sigma(1,2,3,4,9) + d(10-15)$

$E_1 = f() = \Sigma(0,3,4,7,8) + d(10-15)$

$E_0 = f() = \Sigma(0,2,4,6,8) + d(10-15)$

E_3

B_1B_0 \ B_3B_2	00	01	11	10
00			d	1
01		1	d	1
11		1	d	d
10		1	d	d

$E_3 = B_3 + B_2B_0 + B_2B_1$

E_2

B_1B_0 \ B_3B_2	00	01	11	10
00		1	d	
01	1		d	1
11	1		d	d
10	1		d	d

$B_2'B_0 + B_2'B_1 + B_1'B_0'B_2$

E_1

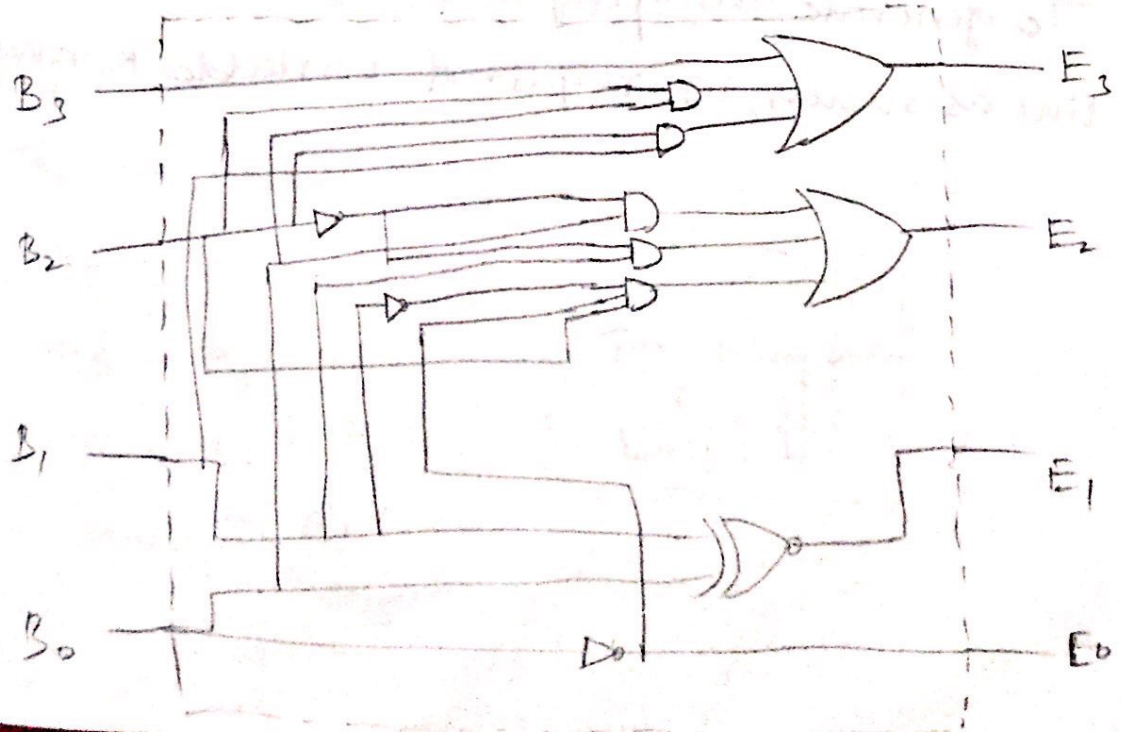
B_1B_0 \ B_3B_2	00	01	11	10
00	1	1	d	1
01			d	
11	1	1	d	d
10			d	d

$B_0'B_1' + B_0B_1$

E_0

B_1B_0 \ B_3B_2	00	01	11	10
00	1	1	d	1
01			d	
11			d	d
10	1	1	d	d

B_0'



Scanned with CamScanner

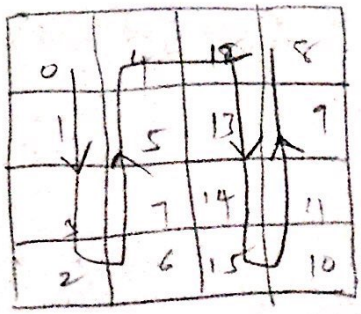
22/8/17

Binary to Gray code converter

Gray code is the mirror image code, first take the mirror image and add zeros above and ones below continue the process.

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
---
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
---
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

To generate the gray code take a zig-zag line as shown, in required variable K-map.



Scanned with CamScanner

Truthtable

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0	
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	- 3
0	0	1	1	0	0	1	0	- 2
0	1	0	0	0	1	1	0	- 6
0	1	0	1	0	1	1	1	- 7
0	1	1	0	0	1	0	1	- 5
0	1	1	1	0	1	0	0	- 4
1	0	0	0	1	1	0	0	- 12
1	0	0	1	1	1	0	1	- 13
1	0	1	0	1	1	1	1	- 14
1	0	1	1	1	1	1	0	- 15
1	1	0	0	1	0	1	0	- 10
1	1	0	1	1	0	1	1	- 11
1	1	1	0	1	0	0	1	- 9
1	1	1	1	1	0	0	0	- 8

$G_3 = \Sigma(8,9,10,11,12,13,14,15)$ → For these inputs the outputs are 1's

$G_2 = \Sigma(4,5,6,7,8,9,10,11)$

$G_1 = \Sigma(2,3,4,5,10,11,12,13)$

$G_0 = \Sigma(1,2,5,6,9,10,13,14)$

$G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

Scanned with CamScanner

G_3

B_1B_0 \ B_3B_2	00	01	11	10
00			1	1
01			1	1
11			1	1
10			1	1

B_3

G_2

B_1B_0 \ B_3B_2	00	01	11	10
00		1	1	1
01		1		1
11		1		1
10		1	1	

$B_3'B_2 + B_3B_2'$

G_1

B_1B_0 \ B_3B_2	00	01	11	10
00		1	1	
01		1	1	
11	1			1
10	1			1

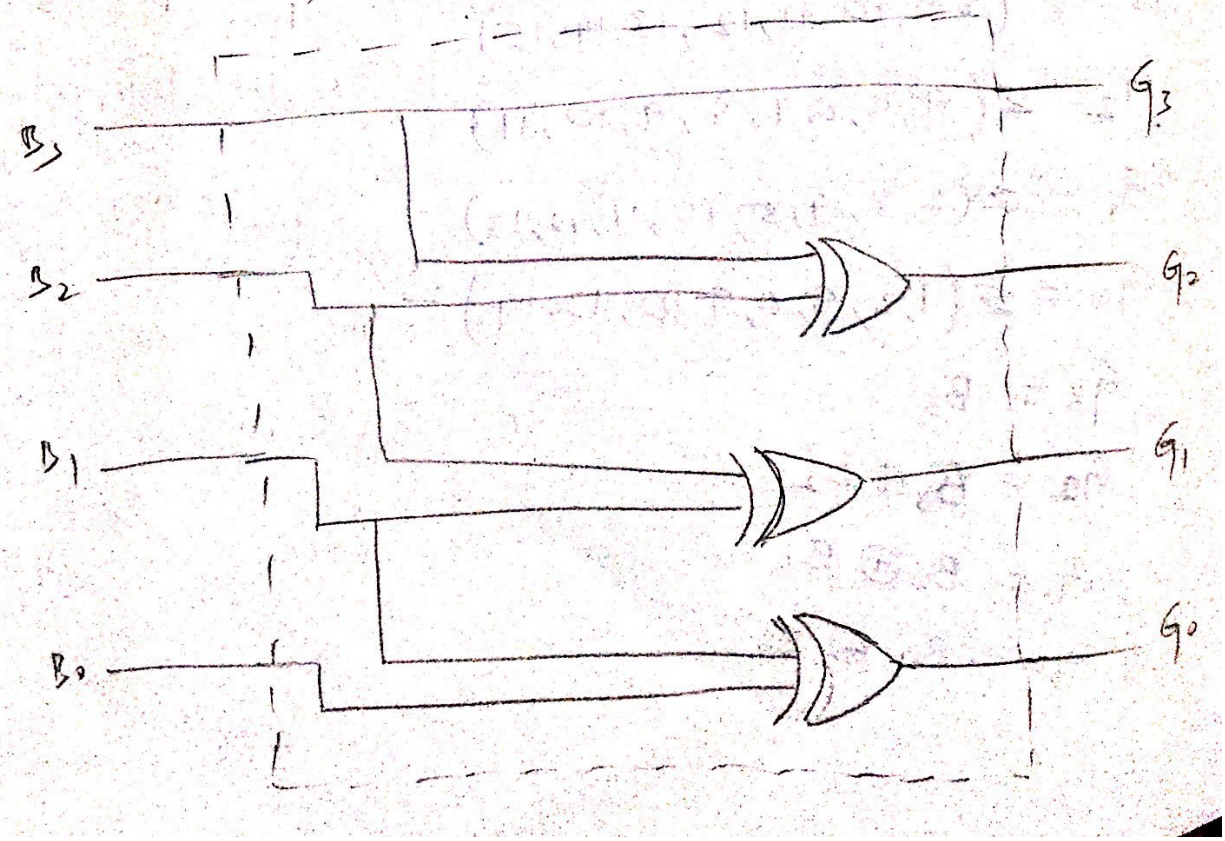
$B_2B_1' + B_1B_2'$

G_0

B_1B_0 \ B_3B_2	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$B_1'B_0 + B_1B_0'$

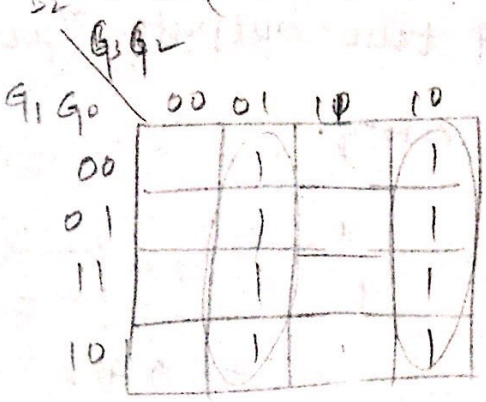
$G_3 = B_3$; $G_2 = B_3 \oplus B_2$; $G_1 = B_2 \oplus B_1$; $G_0 = B_1 \oplus B_0$



Scanned with CamScanner

Gray to Binary code converter

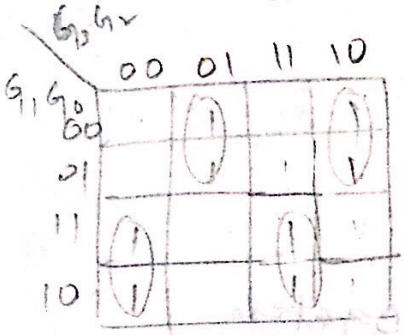
$B_2 = \Sigma(4,5,6,7,12,13,14,15)$



$G_3'G_2 + G_3G_2'$

$G_3 \oplus G_2$

$B_1 = \Sigma(2,3,5,4,13,14,11,10)$

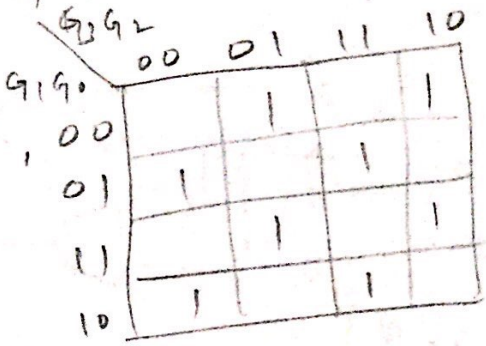


$G_3'G_2G_1' + G_1G_3'G_2'$

$+ G_3G_2G_1 + G_3G_2'G_1'$

$G_3'(G_2 \oplus G_1) + G_3(\overline{G_1 + G_2})$

$G_3 \oplus (G_1 \oplus G_2)$



$G_3'G_2G_1'G_0' + G_3G_2'G_1'G_0'$

$+ G_3'G_2'G_1'G_0 + G_3G_2G_1G_0$

$+ G_3'G_2G_1G_0 + G_3G_2'G_1G_0$

$+ G_3'G_2'G_1G_0' + G_3G_2G_1G_0'$

$G_0'G_1'(G_3 \oplus G_2) + G_0G_1'(\overline{G_2 \oplus G_3}) + G_1G_0(G_2 \oplus G_3)$

$+ G_0'G_1(\overline{G_2 \oplus G_3})$

$(G_3 \oplus G_2)(G_0'G_1' + G_0G_0) + (\overline{G_2 \oplus G_3})(G_0G_1' + G_0'G_1)$

$(G_1 \oplus G_2)(\overline{G_0 \oplus G_1}) + (\overline{G_2 \oplus G_3})(G_0 \oplus G_1)$

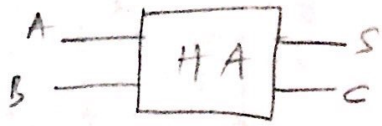
$= G_0 \oplus G_1 \oplus G_0 \oplus G_3$

Scanned with CamScanner

Adders:

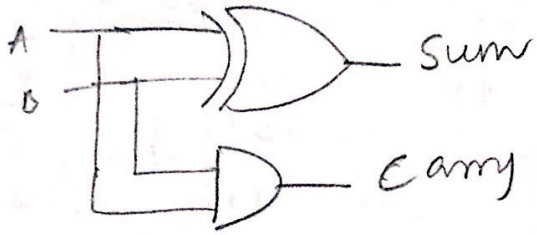
Half Adder: It is a combinational logic circuit, it consists two inputs and two outputs. Two outputs are sum and carry.

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

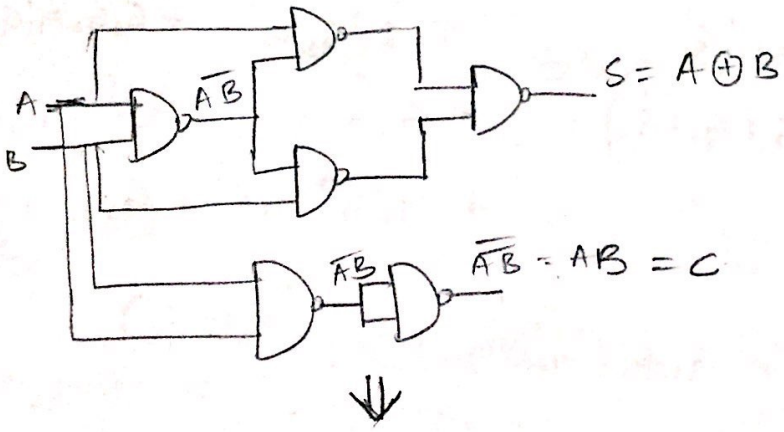


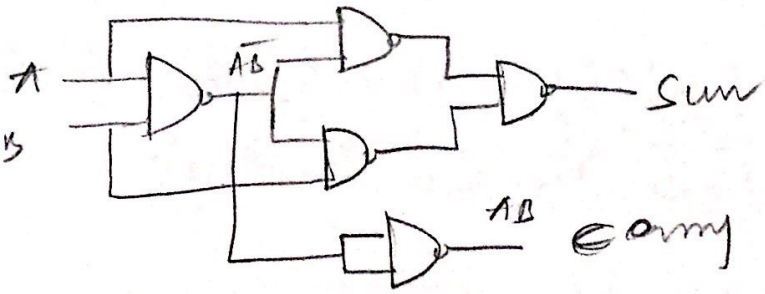
$S = A \oplus B$

$C = AB$



Half adder using only NAND gates.

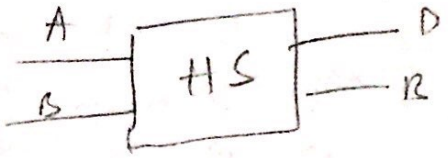




Scanned with CamScanner

Half subtractor: It is a combinational logic circuit, consists two inputs & two outputs. Two outputs are Difference and carry

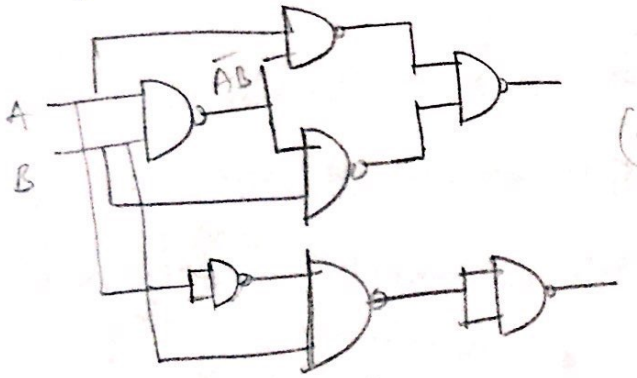
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



$D = A \oplus B$

$B = A'B$

Half subtractor using only NAND gates.

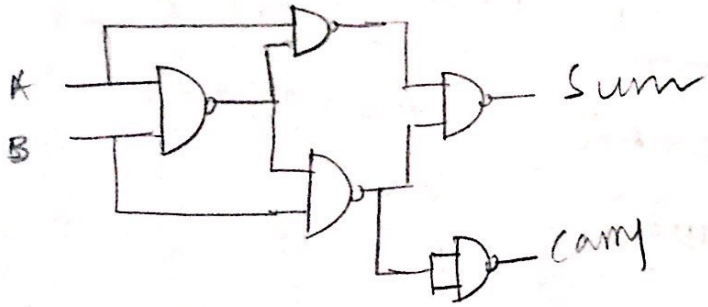


$(B(\bar{A}B))' \quad \overline{(A'+B')B}$

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

$AB + B'$

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

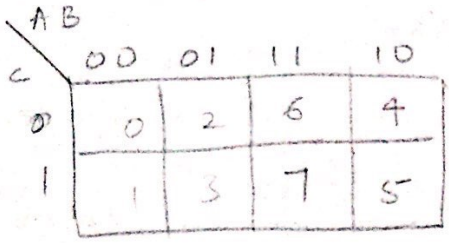


Full adder: It is a combinational logic circuit that consists three iputs and two outputs.

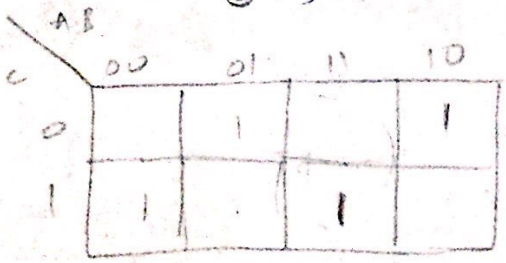
Two outputs are sum and carry.

Scanned with CamScanner

A	B	C	S	C
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



Sum

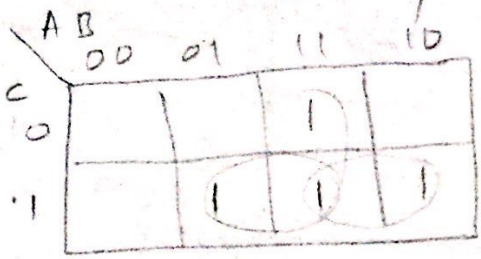


$A'B'C + A'BC' + ABC + AB'C'$

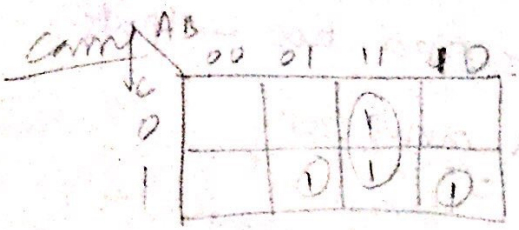
$A'(B \oplus C) + A(\overline{B \oplus C})$

$A \oplus B \oplus C$

Carry



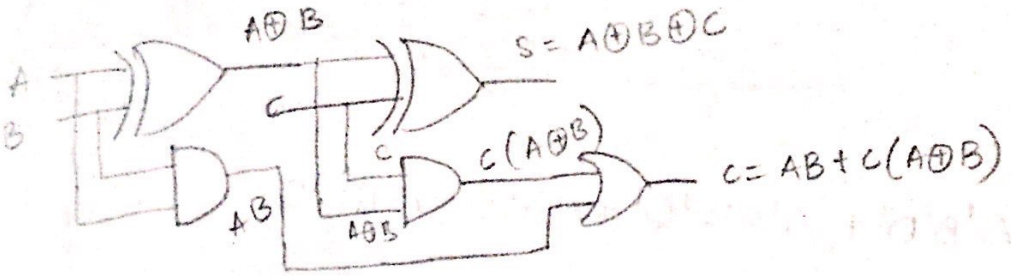
$AB + BC + AC$

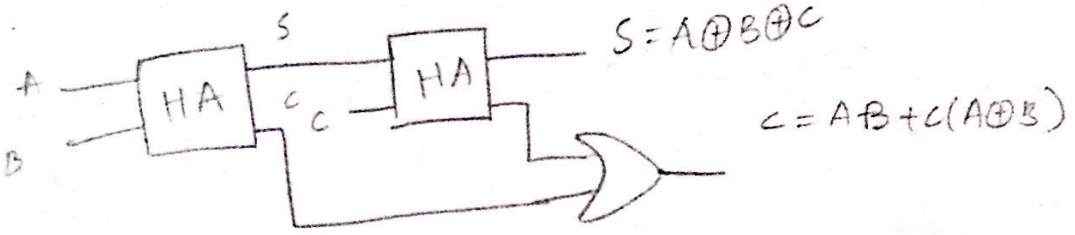


Scanned with CamScanner

23/8/19

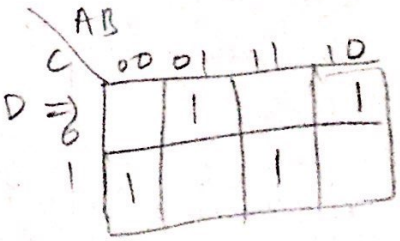
Full Adder using two Half Adders and one OR gate



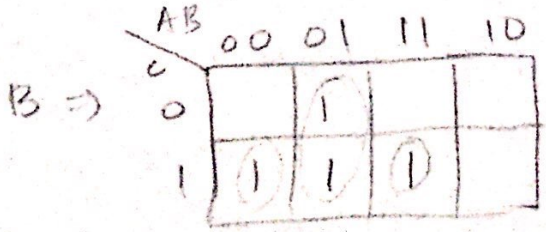


Full subtractor: It is a combinational logic circuit which consists three inputs and two outputs which are Difference and Barrow.

A	B	C	D	B
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



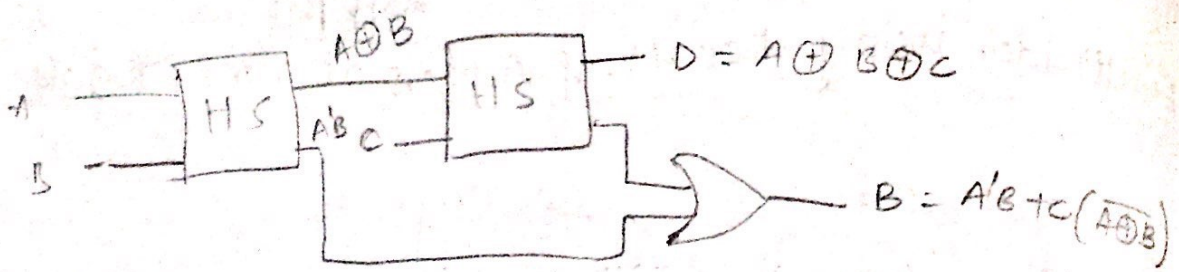
= A⊕B⊕C



A'B + A'B'C + ABC

A'B + C(A⊕B)'

Scanned with CamScanner



$f = A'B'D' + A'B'C'D' + A'BD + ABC'D$

$A'B'D' + BD(A' + C')$

$A'B'D' + A'BD + BDC'$

CD \ AB	00	01	11	10
00	1			
01		1	1	
11		1		
10	1			

$\rightarrow \pi\,(1,3,5,7,13,15)$

CD \ AB	00	01	11	10
00				
01	0	0	0	
11	0	0	0	
10				

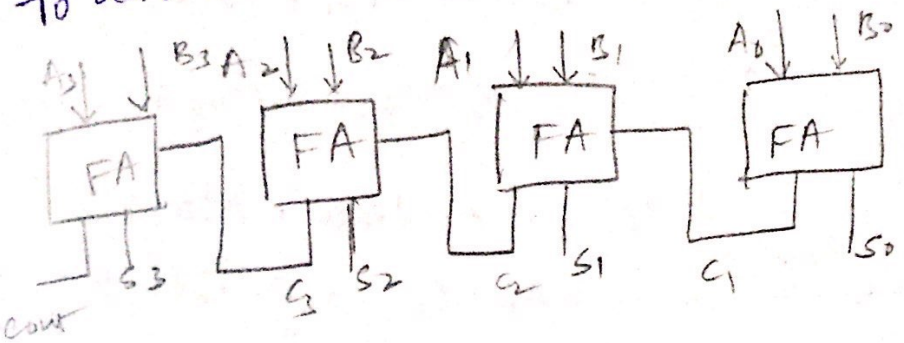
$(A + D')(B' + D')$

Scanned with CamScanner

4-bit Binary Parallel Adder

27/8/19

To add 2 bits and a carry full adder is required.



but it is a slow process

Magnitude Comparator

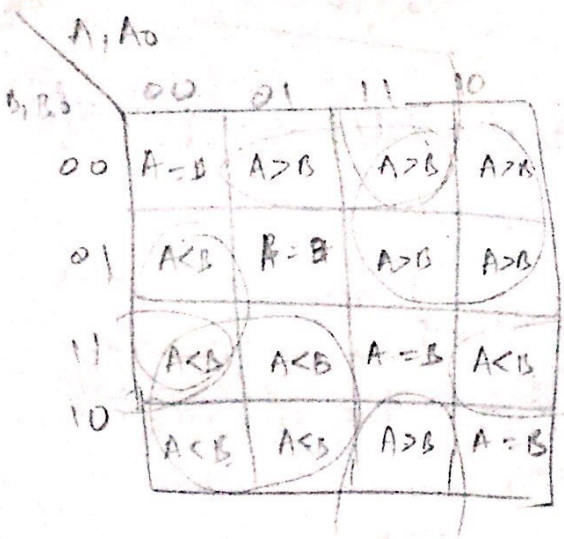
(n-2)

A_1 A_0	B_1 B_0	A>B	A=B	A<B
0 0	0 0	0	1	0
0 0	0 1	0	0	1
0 0	1 0	0	0	1
0 0	1 1	0	0	1
0 1	0 0	1	0	0
0 1	0 1	0	1	0
0 1	1 0	0	0	1
0 1	1 1			
1 0	0 0			
1 0	0 1			
1 0	1 0		1	
1 0	1 1			
1 1	0 0			
1 1	0 1			
1 1	1 0			
1 1	1 1			

→ 2 bit Magnitude comparator

For this 3 - K-maps are required to write the expressions for the outputs. Instead of that it can be done using only one K-map

Scanned with CamScanner



$A>B \Rightarrow A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

$A<B \Rightarrow A_1'B_1 + A_1'A_0B_0 + A_0'B_1B_0$

$A=B \Rightarrow A_1A_0'B_1B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1'B_0'$

$\Rightarrow A_1'B_1'(\overline{A_0 \oplus B_0}) + A_1B_1(A_0B_0 + A_0'B_0')$

$= A_1'B_1'(\overline{A_0 \oplus B_0}) + A_1B_1(\overline{A_0 \oplus B_0})$

$= (\overline{A_1 \oplus B_1})(\overline{A_0 \oplus B_0})$

$= (A_1 \odot B_1)(A_0 \odot B_0)$

4- bit magnitude comparator

No. of cells $= 2^{2n} \Rightarrow n=4 \Rightarrow 2^8$

$A = A_3A_2A_1A_0$

$B = B_3B_2B_1A_0$

$A_3 > B_3 \Rightarrow A>B$

$A_3 = B_3$; $A_2 > B_2 \Rightarrow A>B$

$A_2 = B_2$; $A_1 > B_1 \Rightarrow A>B$

$A_1 = B_1$; $A_0 > B_0 \Rightarrow A>B$

$A_3 \odot B_3 = x_3$

$A_2 \odot B_2 = x_2$

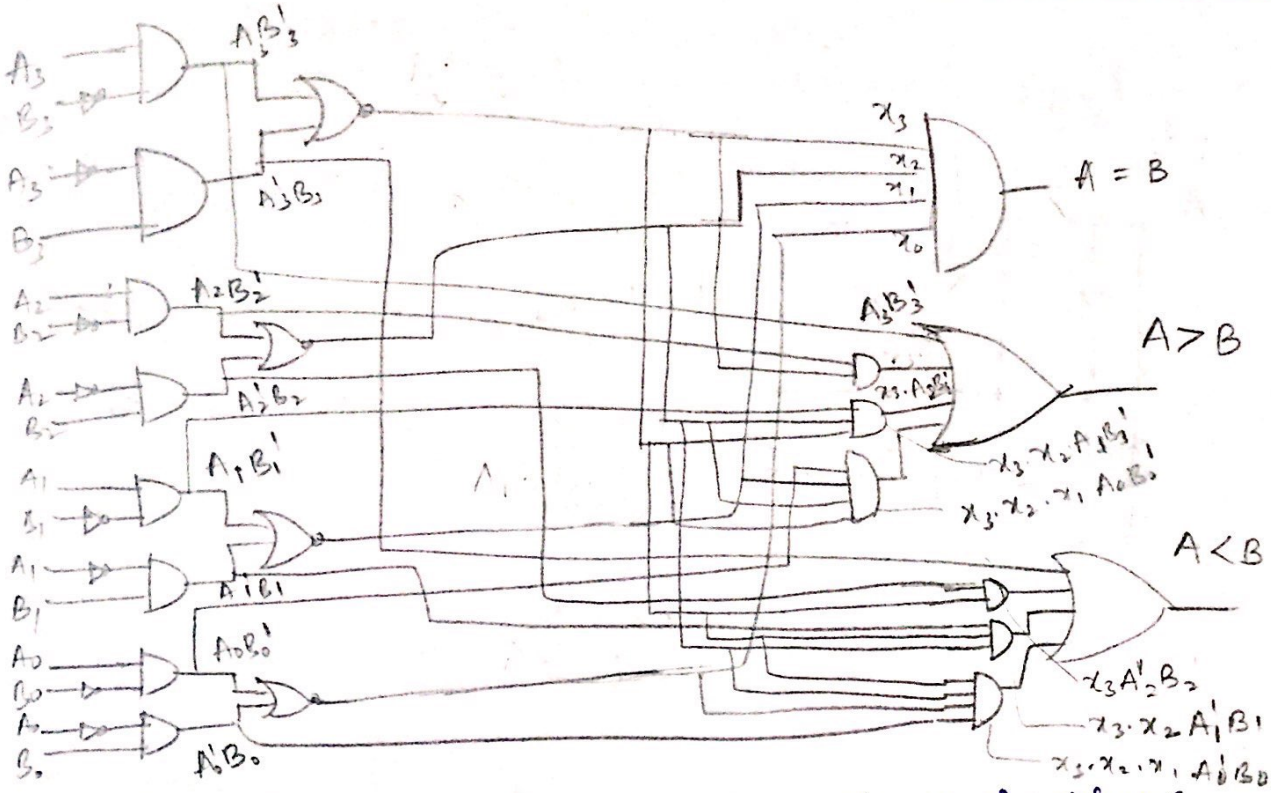
$A_1 \odot B_1 = x_1$

$A_0 \odot B_0 = x_0$

Outputs will be higher (1) when both the inputs are equal in XNOR gate

Output will be higher (1) when all the inputs are higher (1) in AND gate

Scanned with CamScanner



A_3	B_3	(F) $\Rightarrow A_3B_3'$
0	0	0
0	1	0
1	0	1
1	1	0

A_2B_2' is further checked only when A_3 is equal to B_3

$\Rightarrow x_3 \cdot A_2B_2'$

similarly $x_3 \cdot x_2 \cdot A_1B_1'$

$A > B \Rightarrow A_3B_3' + x_3 \cdot A_2B_2' + x_3 \cdot x_2 A_1B_1' + x_3 \cdot x_2 \cdot x_1 \cdot A_0B_0'$

$\Rightarrow A < B : A_3'B_3 + x_3 \cdot A_2'B_2 + x_3 \cdot x_2 A_1'B_1 + x_3 \cdot x_2 \cdot x_1 A_0'B_0$

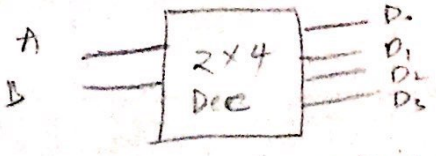
$A_1'B_1 + x_1 A_0'B_0$

$A_0'A_1' + A_0A_1(A_0'B_0) \Rightarrow A_1'A_0'D_0 +$

Decoder:

It is a combinational logic circuit, which consists of n inputs and m outputs

$m \leq 2^n$

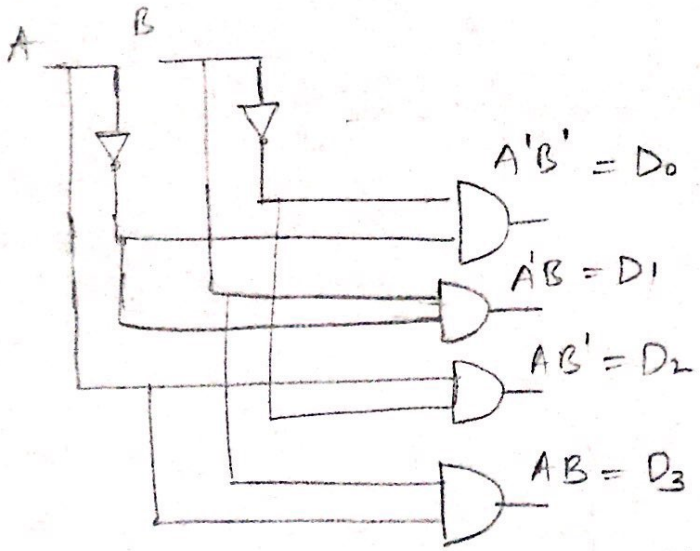


Block Diagram

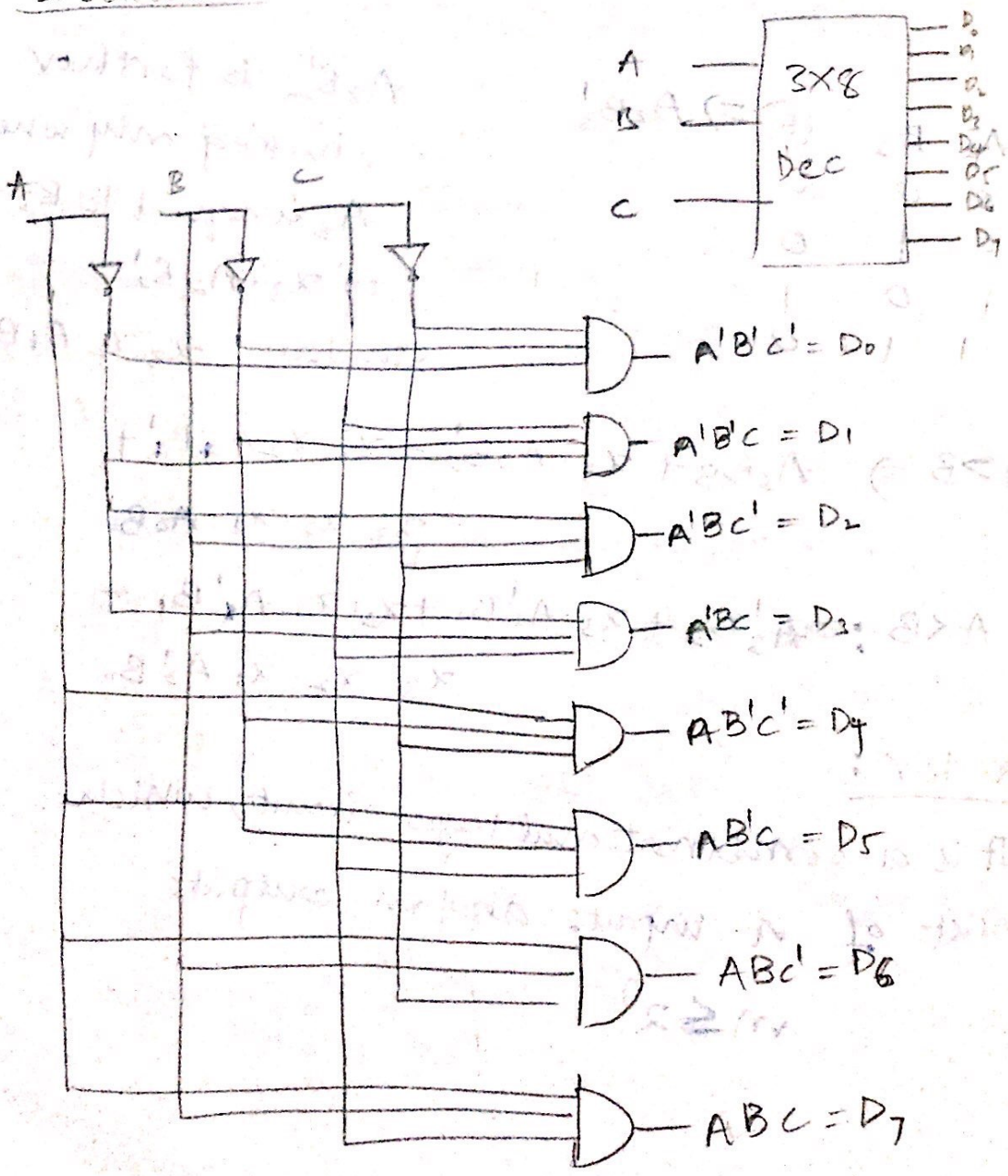
2×4 ⇒ 2 inputs, 4 outputs

Scanned with CamScanner

Logic diagram



3X8 decoder

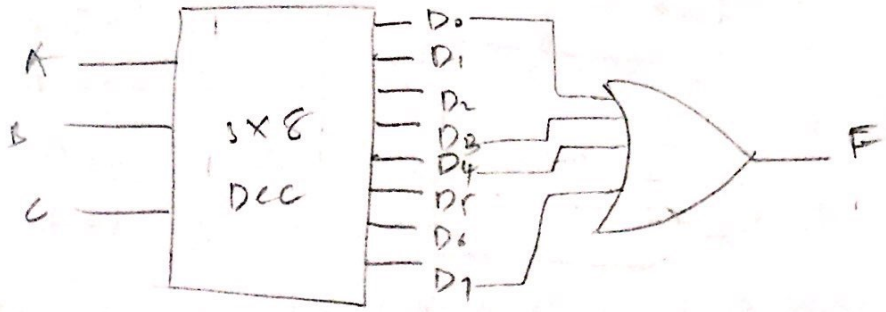


Scanned with CamScanner

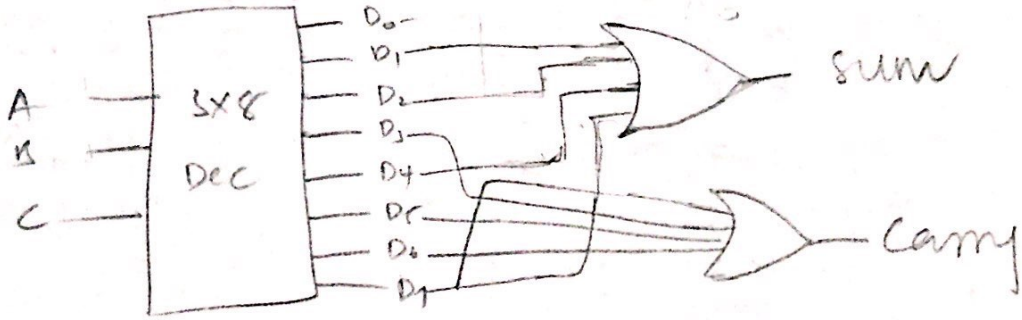
Boolean Function Implementation

$f(A,B,C) = \Sigma(1,2,4,7)$ → connect a OR gate to D_1, D_2, D_4 and D_7

$f(ABC) = \Sigma(0,3,4,7)$



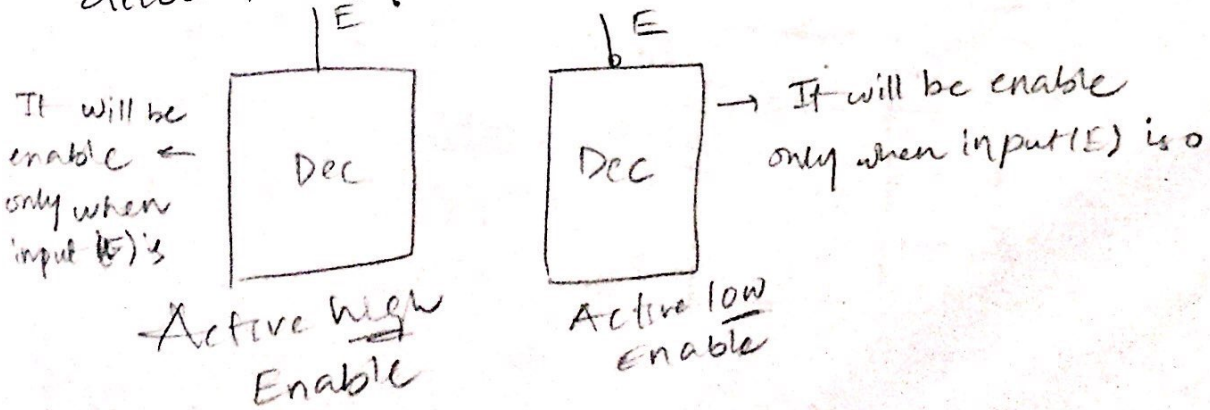
Implement full adder using a decoder



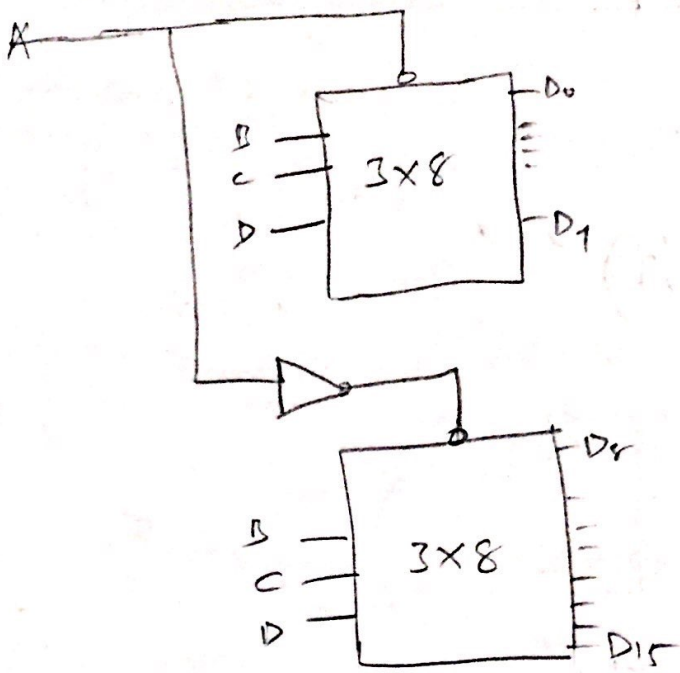
$f(ABC) = \pi(1,2,4,7)$

$f(ABC) = \Sigma(0,3,5,6)$ → Implement $\overline{F}$

Implement 4x16 decoder using Two 3x8 decoder and one inverter.

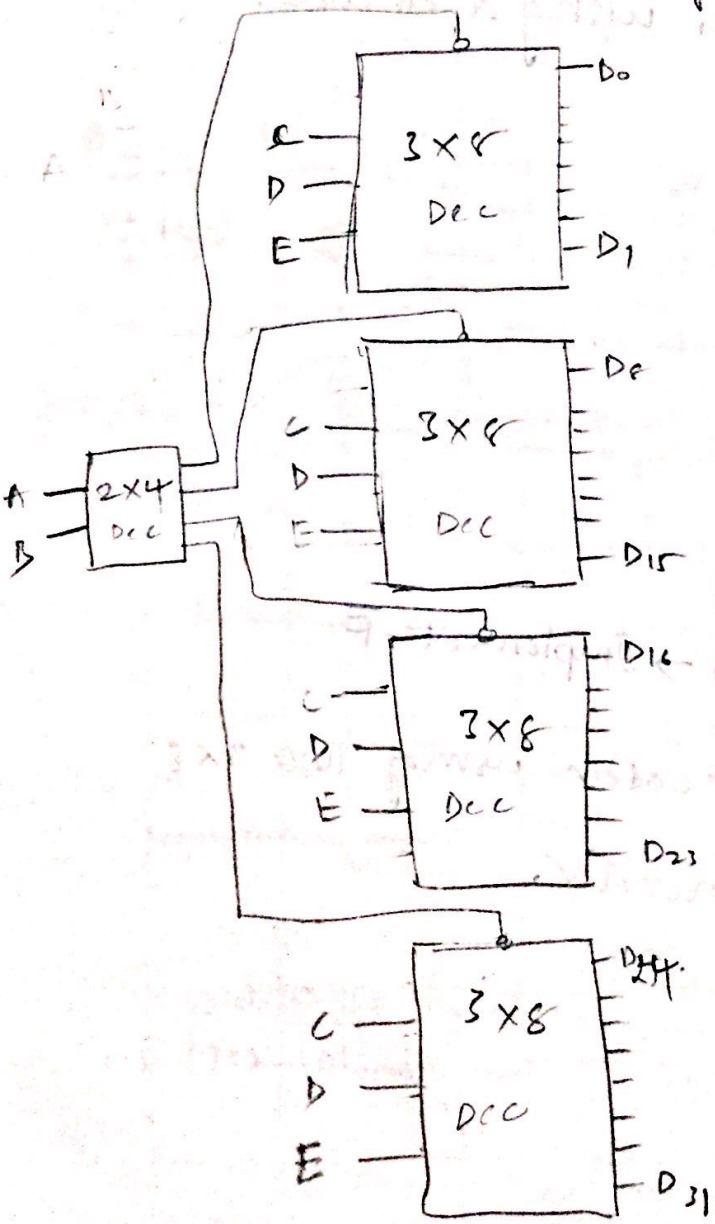


Scanned with CamScanner



A	B	C	D	
0	0	0	0	D_0
0	0	0	1	D_1
0	1	1	1	D_7
1	0	0	0	D_8
1	1	1	1	D_{15}

→ Implement 5×32 Decoder using 3×8 Decoders & we are allowed other decoders



A	B	C	D	E	
0	0	0	0	0	D_0
				1	D_1
0	0	1	1	1	D_7
0	1	0	0	0	D_8
				1	
0	1	1	1	1	D_{15}
1	0	0	0	0	D_{16}
				1	
1	0	1	1	1	D_{23}
1	1	0	0	0	D_{24}
				1	
1	1	1	1	1	D_{31}

Scanned with CamScanner

Decimal Decoders

3/9/19

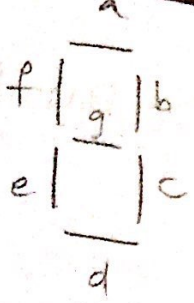
B_3	B_2	B_1	B_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

B_1B_0 \ B_3B_2	00	01	11	10
00	D_0	D_4	d	D_8
01	D_1	D_5	d	D_9
11	D_3	D_7	d	d
10	D_2	D_6	d	d

10 to 15 should be taken as don't cares

Scanned with CamScanner

BCD to seven segment decoder

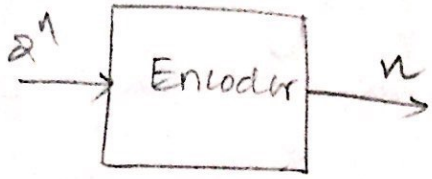


	B_3	B_2	B_1	B_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Scanned with CamScanner

Encoder (opposite of Decoder)

2^n – inputs ; n – outputs.



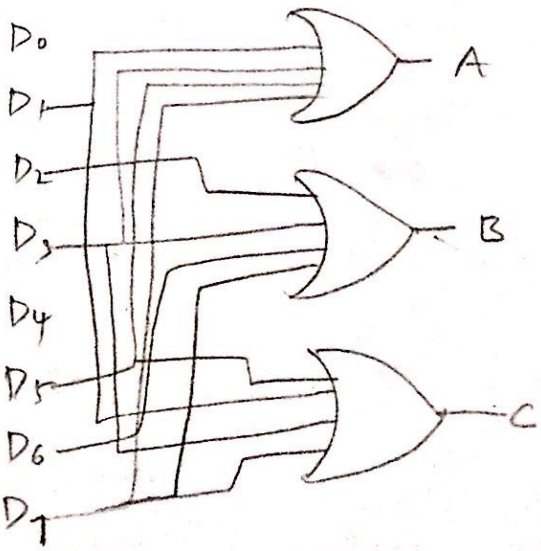
8 x 3 Encoder :

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	C	B	A
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$C = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$A = D_1 + D_3 + D_5 + D_7$



Disadvantages :

If two inputs are high simultaneously, we will get wrong output.

If all inputs are low the output is not mentioned.

Scanned with CamScanner

To overcome the disadvantages of Encoder

Priority Encoder, contd—

4x2 Encoder

V - Valid bit

D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	x	x	0
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

If D_2 and D_3 both inputs are in then the D_3 output will come because the priority will be given to D_3 as 3 is greater than 2

Y_1

D_2D_3 \ D_0D_1	00	01	11	10
00	d	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$Y_1 = D_2 + D_3$

Y_0

D_2D_3 \ D_0D_1	00	01	11	10
00	d	1	1	0
01	1	1	1	1
11	1	1	1	1
10	1			

$D_0D_2' + D_3$

V

D_2D_3 \ D_0D_1	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$V = D_0 + D_1 + D_2 + D_3$

If xx10 is considered for Y, x may be 0 or 1

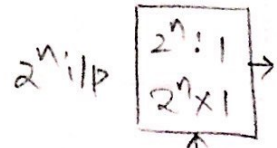
→ 0010 - 2
0110 - 6
1010 - 10
1100 - 12
} → 1

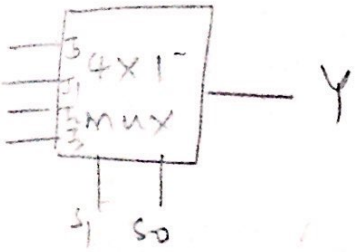
Scanned with CamScanner

Multiplexer

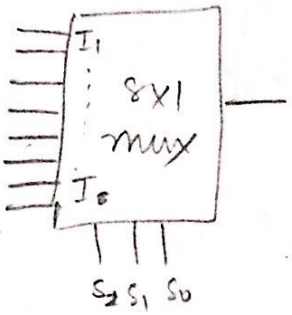
(mux → many inputs to single output)
(parallel to series converter)

Multiplexer will have 2^n inputs and one output
n - no of selection inputs. Multiplex is commonly known as mux.





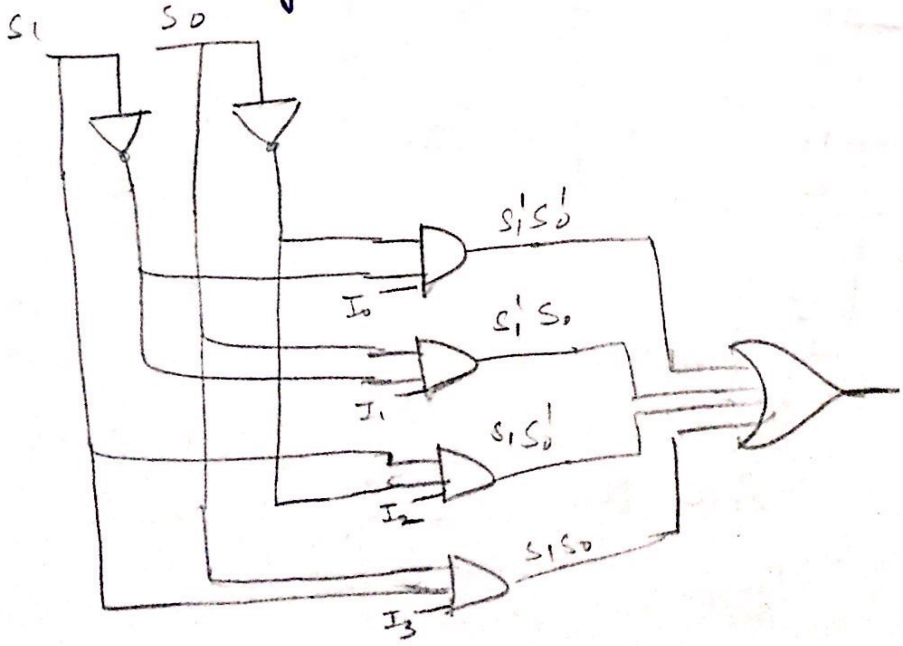
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

4/9/19

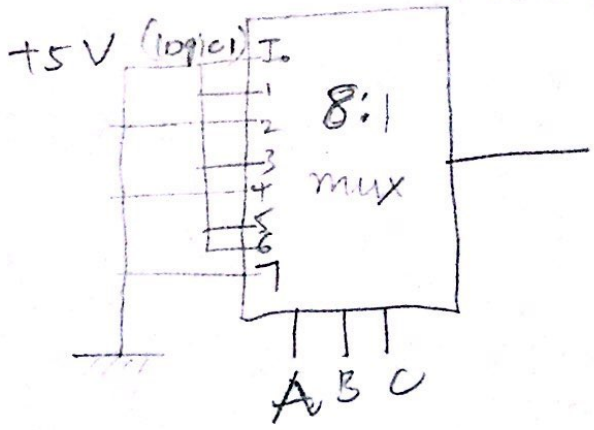
4x1 mux Logic diagram



Scanned with CamScanner

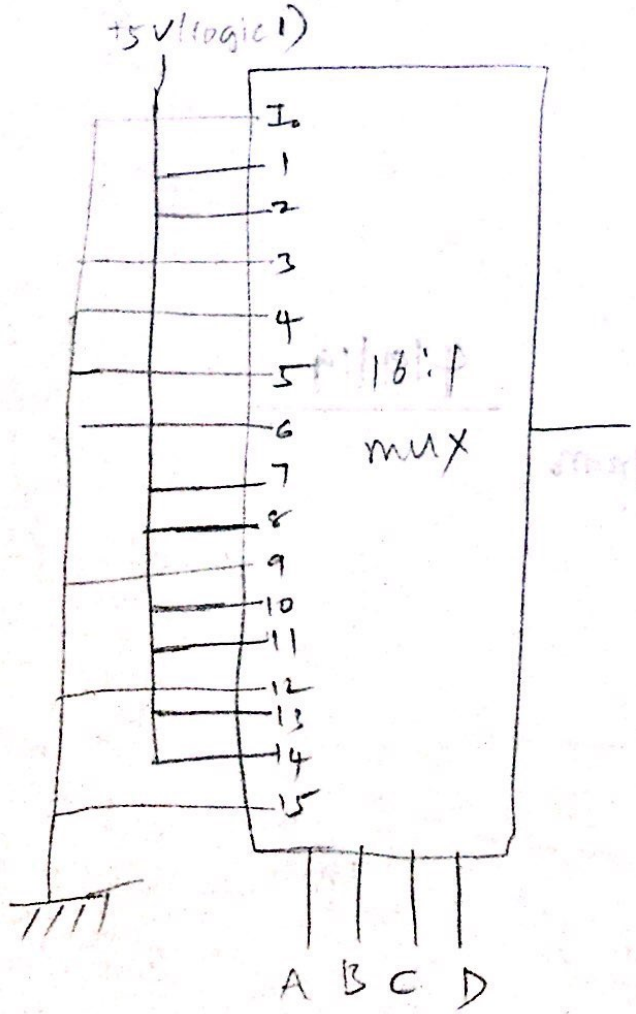
Boolean Function Implementation

$f(ABC) = \Sigma(1, 3, 5, 6)$



First find the size of mux
connect the given numbe
to logic 1 and remain
to the ground (logic 0

→ $f(ABCD) = \Sigma(1, 2, 7, 8, 10, 11, 13, 14)$



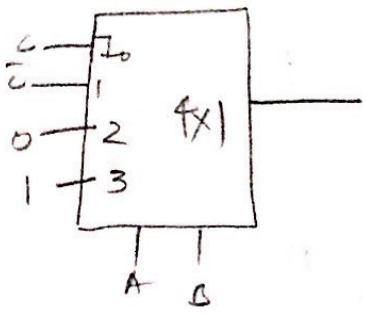
Scanned with CamScanner

case (i): Implementing n-variable Boolean Function with $2^n \times 1$ mux

case (ii): Implementing n-variable Boolean Function with $2^{n-1} \times 1$ mux

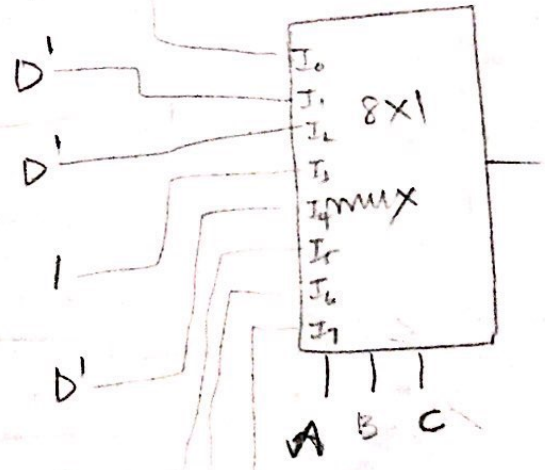
→ $f(ABC) = \Sigma(1, 2, 6, 7)$

A	B	C	f	
0	0	0	0	
0	0	1	1	C
0	1	0	1	
0	1	1	0	C'
1	0	0	0	
1	0	1	0	0
1	1	0	1	
1	1	1	1	1



→ $f(ABCD) = \Sigma(1, 2, 4, 6, 7, 8, 13, 14)$

A	B	C	D	f	
0	0	0	0	0	
0	0	0	1	1	D
0	0	1	0	1	
0	0	1	1	0	D'
0	1	0	0	1	
0	1	0	1	0	D'
0	1	1	0	1	
0	1	1	1	1	1
1	0	0	0	1	
1	0	0	1	0	D'
1	0	1	0	0	
1	0	1	1	0	0
1	1	0	0	0	
1	1	0	1	1	D
1	1	1	0	1	
1	1	1	1	0	D'



Scanned with CamScanner

5/9/19

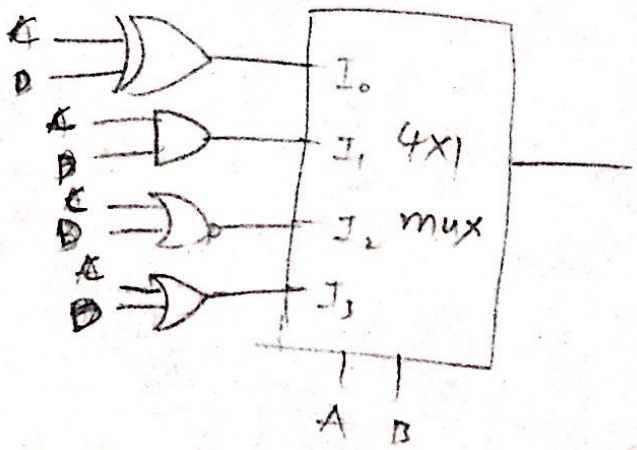
iii) Implementing 'n' variable Boolean function with $2^{n-2} \times 1$ mux

$f(ABCD) = \Sigma(1,2,7,8,13,14,15)$

A	B	C	D	F	
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	$C \oplus D$
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	$C \cdot D$
0	1	1	0	0	
0	1	1	1	1	
1	0	0	0	1	
1	0	0	1	0	$\overline{C+D}$
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	$C+D$
1	1	1	0	1	
1	1	1	1	1	

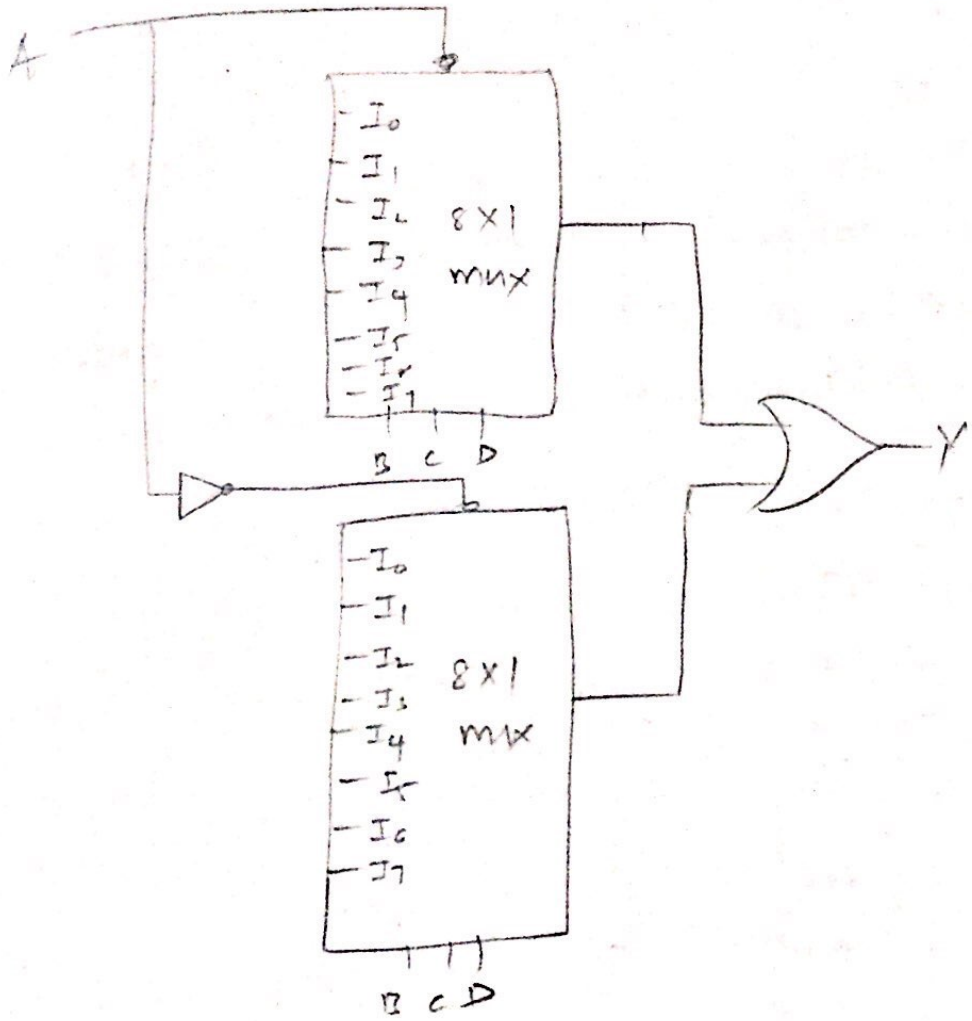
Represent F in terms of C and D

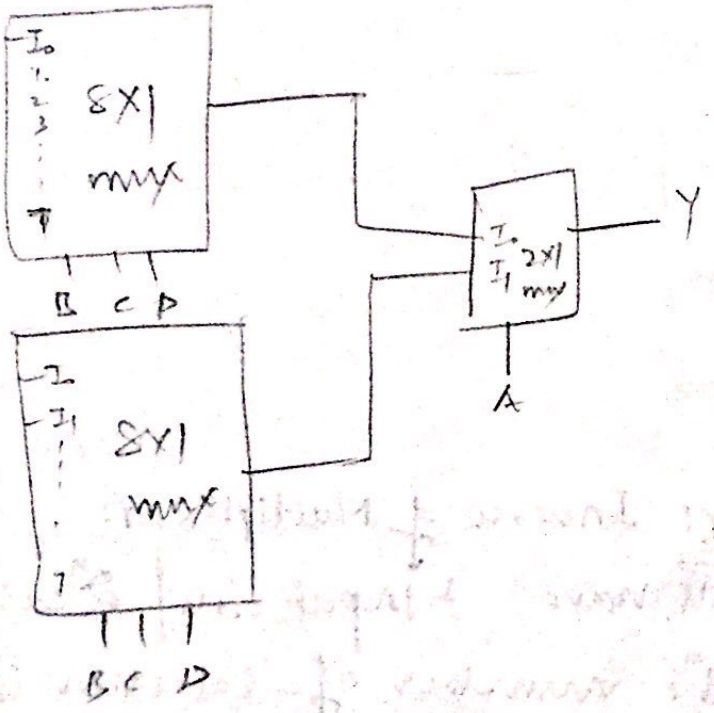
Now, A & B are the selection inputs.



Scanned with CamScanner

Implement 16x1 mux using two 8x1 muxs and one or gate

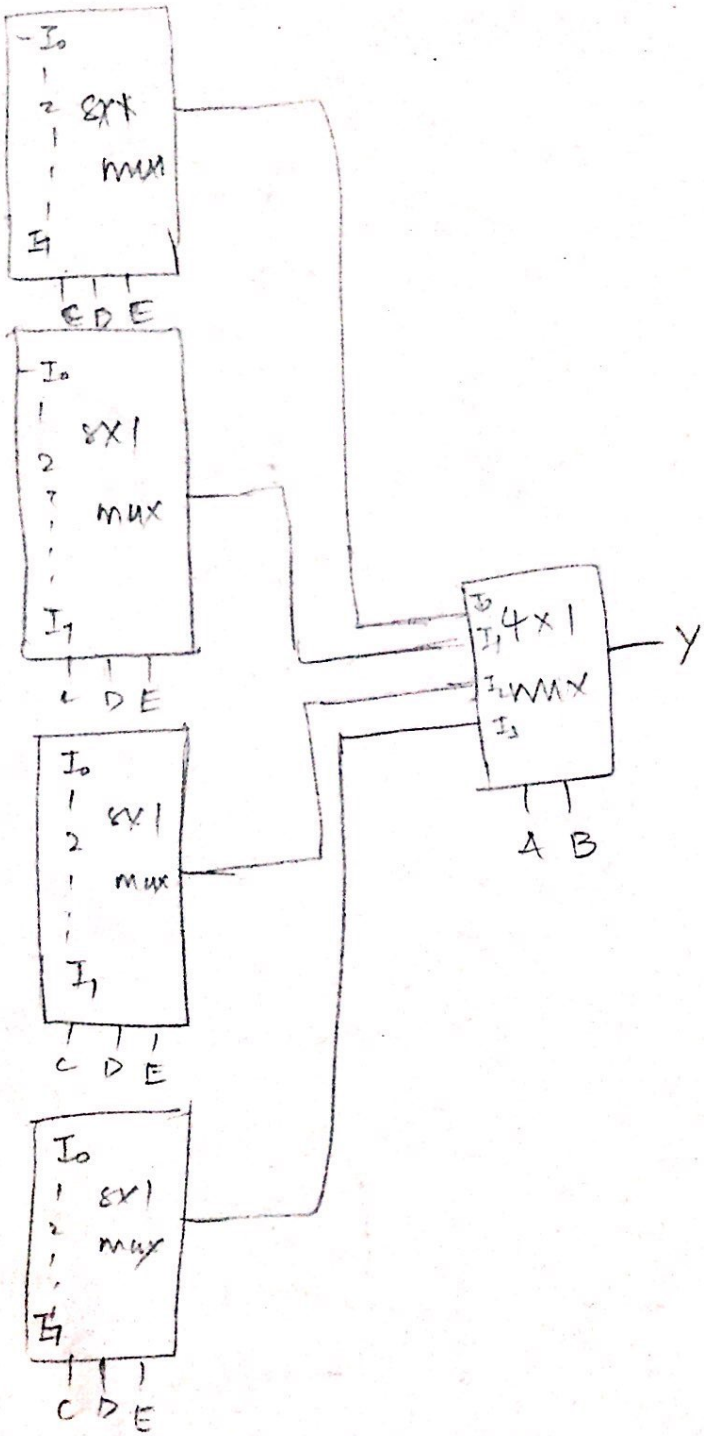




When enable inputs are not given then the mux will be enabled.

Scanned with CamScanner

Implement 32x1 mux using 4 8x1 muxs and any another mux



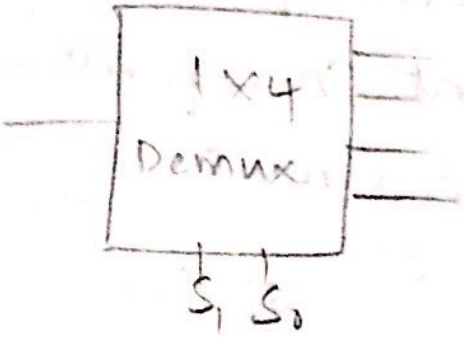
Demultiplexer: Inverse of Multiplexer.

Here it will have 1 input and 2^n outputs.

where n is the number of selection inputs.

1×2^n

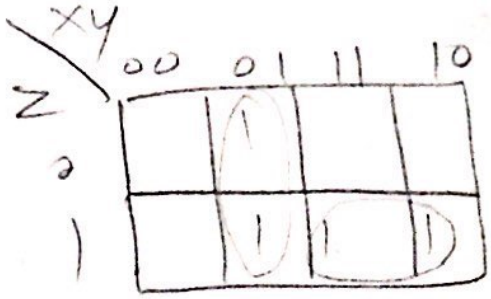
Scanned with CamScanner



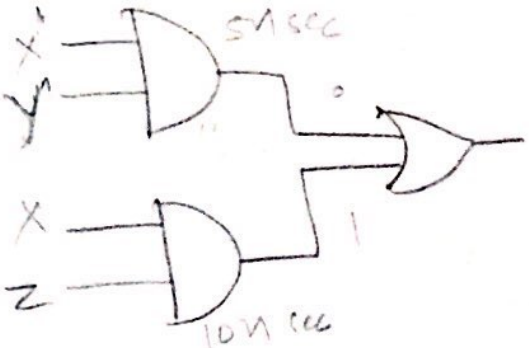
Hazards

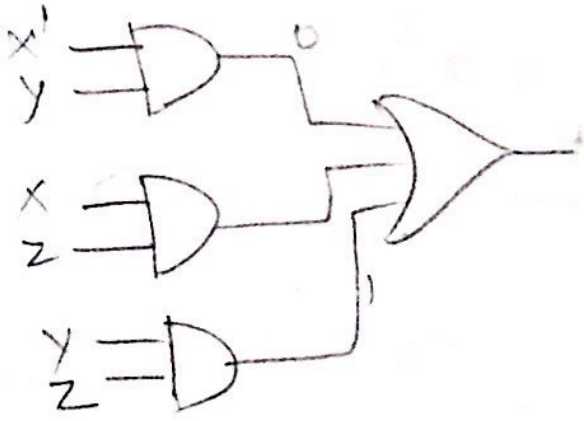
Getting momentary erorious output is called as Hazard. To avoid hazards we need to consider all the prime implicants.

$f(x,y,z) = \Sigma(2,3,5,7)$



$x'y + xz$





Scanned with CamScanner