Binary of 45 – 101101

Sign and Magnitude representation: Negative numbers can be represented in many ways. One way is to use sign-magnitude. Eg: +2, -2 . Adding 1 to the left most in a binary number makes it a negative number and 0 makes it a positive number.

For example 0101101 represent +45 and 1101101 represents -45 if 6 digits of a binary number are considered and the leftmost digit represents the sign.

This is a problem as a number has two different representations, in case of 0. +0 ,-0 (0000, 1000) respectively and causes complications in digital systems . So we have complement notations (1’s complement and 2’s complement) to represent signed numbers.

One’s complement: It is the complement of that number. Eg : the one’s complement of 1010100 is 0101011.

1. (1111111 – 1010100) = 0101011

Two’s complement: One’s complement + 1

To represent -27: Its binary format for 27 is 00011011

Its 1’s complement is 11100100 + 1 = 11100101

###### BASE CONVERSIONS:

|  |  |
| --- | --- |
| BASE | REPRESENTATION |
| 2 | Binary |
| 8 | Octal |
| 10 | Decimal |
| 16 | HexaDecimal |

#### 1.Decimal to Binary

Decimal 10.25 to binary

2 10

2 5 - 0 (1010)

2 2 - 1

1 - 0

1010

0.25\* 2 = 0.50 -> 0 (01)

0.50\* 2 = 1.00 -> 1

1010.01 is the binary for 10.25

#### 2. Binary to Decimal:

1010.01 binary to decimal

0\*2^0 + 1\*2^1 + 0\*2^2 + 1\*2^3 + 0\*2^-1 + 1\*2^-2 = 0+2+0+8+0+0.25 = 10.25

#### 3. Decimal to Octal:

10.25 decimal to octal

8 10

1 - 2 (12)

0.25 \* 8 = 2.00

Answer is : 12.2 is the octal for decimal number 10.25

#### 4. Octal to Decimal

(12.2 )8 to decimal ?

2\*8^0+1\*8^1+2\*8^-1 = 2+8+0.25 = 10.25

#### 5. Hexadecimal to Binary

|  |  |
| --- | --- |
| Binary Equivalent | Hexadecimal |
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

(3A)16 to binary is => 00111010

#### 6. Binary to Hexadecimal:

Group them to groups of 4.

1111011011 is binary

0011 1101 1011 = 3DB is hexadecimal representation of binary number 1111011011.

#### Floating Point Representation:

#### To convert floating point (32-bit) number to decimal:

The floating point number has the elements:

1. Sign
2. Exponent
3. Mantissa

###### Sign:

Left most digit decides –ve number or +ve number

|  |  |
| --- | --- |
| Sign | Type |
| 1 | Negative number |
| 0 | Positive number |

###### Exponent(e):

The next 8 bits is the exponent.

###### Mantissa(m):

The remaining bits after excluding sign bit and exponent bits.

###### Scientific Notation:

1.xxxxxxxx \* 2^e

m

###### Bias:

In 8 bits, the number of exponent bits is 3. So, 2^(3 – 1) - 1 = 3 bits is bias

In 32 bits, the number of exponent bits is 8. So, 2^(8-1) - 1 = 127 bits is bias

In 64 bits, the number of exponent bits is 11. So, 2^(11-1) - 1 = 1023 bits is bias

If the exponent obtained is greater, then we will have a positive exponent (e).

Example 1:

0100 0011 0101 0100 0000 0000 0000 is a 32 bit number.

Sign bit is 0 ---------------- > so positive number.

Exponent is (10000110)2 to decimal we get 134, which is greater than 127.

Exponential Bias (e) = 134-127 = 7

Mantissa is 101 0100 0000 0000 0000 -----------------🡪 1\* 2^-1 + 0\*2^-2 + 1\*2^-3+0\*2^-4+1\*2^-5+………

= 0.65625

The decimal number is: (-1)^s \* (1+m)\*2^e

(-1)^0 \* (1+0.65625) \* 2^7 = 212 is the decimal number.

Example 2 : 0100 0001 1101 0000 0000 0000 0000 0000

1. 0 -> positive number
2. 1000 0011 -> 1 + 2 + 128 = 131

E = 131 – 127 = 4

1. 101 0000 0000 0000 0000 0000

0.5 + 0.125 = 0.625

4) (-1)^0 \* (1+0.625) \* 2^4 = 26

#### Convert decimal to floating point number(32-bit):

1. Sign(MSB)
2. Exponent(8-bit after MSB)
3. Mantissa(Remaining 23 bits)

#### Properties of Boolean Algebra:

|  |  |
| --- | --- |
| Law | Expression |
| Annulment Law | A.0 = 0  A+1 = 1 |
| Identity Law | A.1 = A  A+0 =A |
| Idempotent Law | A+A = A  A.A = A |
| Complement Law | A + A’ = 1  A . A’ = 0 |
| Double negation Law | ((A)’)’ = A |
| Commutative Law | A+B = B+A  A.B = B.A |
| Associative Law | A + (B+C) = (A+B) + C  A . (B . C) = (A . B) . C |
| Distributive Law | A . (B+C) = (A . B) + (A . C)  A + (B . C) = (A+B) . (A+C) |
| Absorption Law | A . (A+B) = A  A + AB = A |
| De Morgan Law | (A . B)’ = A’ +B’  (A+B)’ = A’ . B’ |

#### Representation of Boolean Algebra:

###### Canonical and Standard Forms:

Minterm or standard product (AND) mi: In this binary variable is

|  |  |
| --- | --- |
| Unprimed | 1 |
| Primed | 0 |

If minterm is xy’ means x=1 and y=0

For 2 variables minterms are:

|  |  |
| --- | --- |
| m 0 | = x’y’ |
| m 1 | = x’y |
| m 2 | = xy’ |
| m 3 | = xy |

Maxterm or Standard sum (OR) Mi:

|  |  |
| --- | --- |
| Unprimed | 0 |
| Primed | 1 |

|  |  |
| --- | --- |
| M 0 | = x+y |
| M 1 | = x+y’ |
| M 2 | = x’+y |
| M 3 | = x’+y’ |

For 3 variables x y and z:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | Minterms | | Maxterms | |
| X | Y | Z | Term | Designation | Term | Designation |
| 0 | 0 | 0 | x’y’z’ | m 0 | x+y+z | M 0 |
| 0 | 0 | 1 | x’y’z | m 1 | x+y+z’ | M 1 |
| 0 | 1 | 0 | x’yz’ | m 2 | x+y’+z | M 2 |
| 0 | 1 | 1 | x’yz | m 3 | x+y’+z’ | M 3 |
| 1 | 0 | 0 | xy’z’ | m 4 | x’+y+z | M 4 |
| 1 | 0 | 1 | xy’z | m 5 | x’+y+z’ | M 5 |
| 1 | 1 | 0 | xyz’ | m 6 | x’+y’+z | M 6 |
| 1 | 1 | 1 | xyz | m 7 | x’+y’+z’ | M 7 |

###### Relation between minterm and maxterm:

m 0 = ( M 0 )’ or M 0 = (m 0)’

###### Constructing BooleanFunctions:

Boolean functions expressed a sum of minterms and product of maxterms are said to be in canonical form.

Example: Express F = x+y’z in sum of minterms and product of maxterms.

F = x+y’z

= x(y+y’)+y’z

=xy+xy’+y’z

=xy(z+z’)+xy’(z+z’)+y’z(x+x’)

=xyz+xyz’+xy’z+xy’z’+x’y’z

F=m 7 + m 6 + m 5 + m 4 + m 1 is sum of minterms

= M0. M2 . M3 is product of maxterms.

###### Standard Forms:

###### Logic Gates:

AND, OR, NAND, NOR, XOR, Exclusive NOR (XOR)’, NOT , Buffer.

Universal Gates:

NOR Gate

NAND Gate

Universal Gates

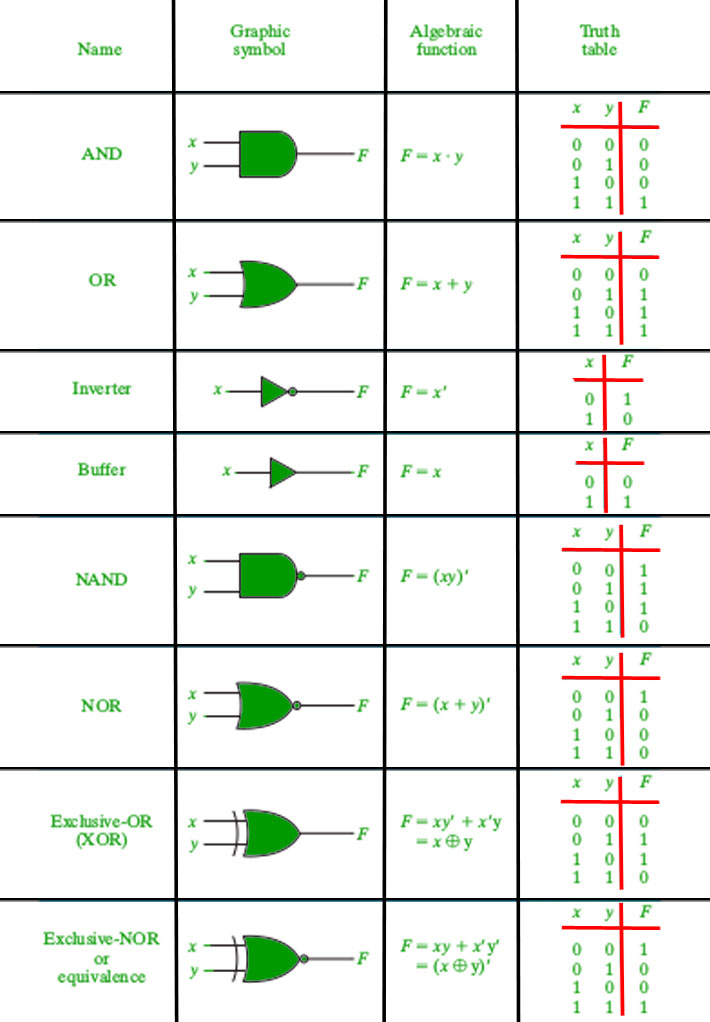
(A.B)’ = A’ + B’

(A.(A’+B’))’ = (AB’)’ = A’B

(B.(A’+B’))’ = (A’B)’ = AB’

(A’B.AB’)’

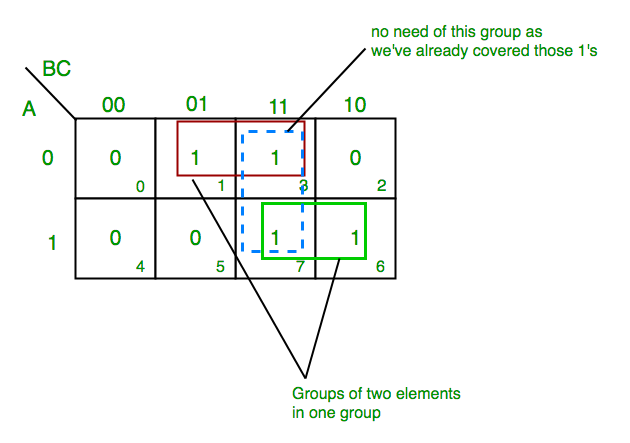
Construct XOR Gate using NAND Gate and NOR Gate.



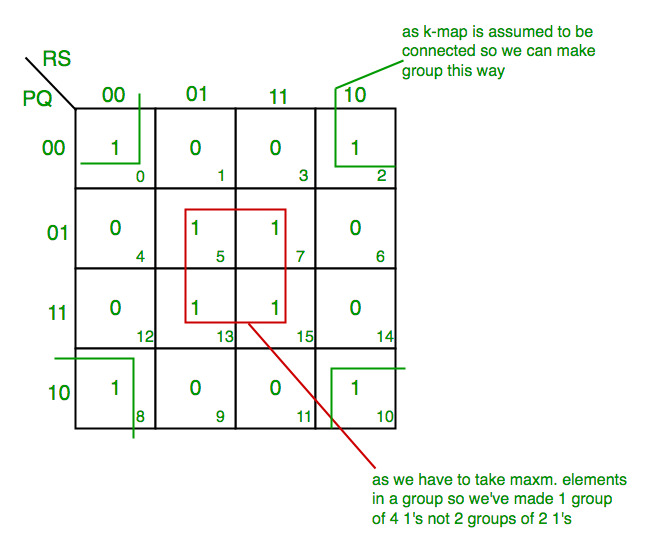
###### K- Map

SOP Form:

3-Variable: Z = ∑ A,B,C (1,3,6,7)

  
**Final expression (A’C+AB)**

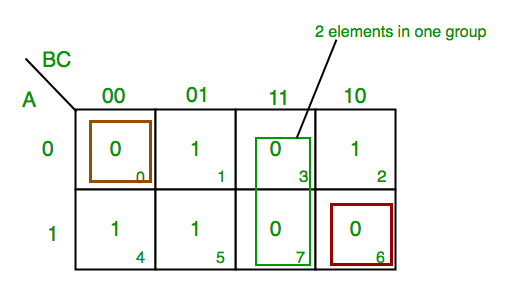
4-Variable Form : F(P,Q,R,S) = ∑(0,2,5,7,8,10,13,15)



**Final expression (QS+Q’S’)**

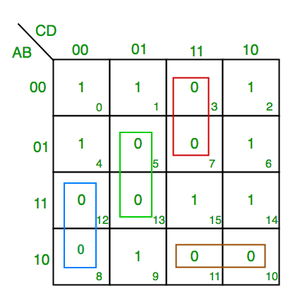
POS form:

K Map for 3 variables: F(A,B,C) = π(0,3,6,7)



**Final expression (A’ + B’ + C) (B’ + C’) (A + B + C)**

4 variables : F(A,B,C,D) = π(3,5,7,8,10,11,12,13)



**(C+D’+B’).(C’+D’+A).(A’+C+D).(A’+B+C’)**

**PITFALL–**  \*Always remember ***POS ≠ (SOP)’***

\*The correct form is (**POS of F)=(SOP of F’)’**