CHAPTER ONE: DIGITAL SYSTEMS AND BINARY NUMBERS

Introduction:

A digital system is a combination of devices {mechanical, electrical, photo electronic,…,etc.} arranged to perform certain functions in which quantities are represented digitally.

Digital systems are used in communication, business transactions, traffic control, spacecraft guidance, medical treatment, weather monitoring, the Internet, and many other commercial, industrial, and scientific enterprises.

Number Systems and Number-Base Conversions:

1. Decimal Number Systems

It is said to be of base (10) since it uses 10 digits {0, 1, 2, 3, ……, 9}.

Ex. 1: $[7392]_{10} = 2\times10^0 + 9\times10^1 + 3\times10^2 + 7\times10^3 = 7392.$

Ex. 2: $[0.421]_{10} = 0\times10^0 + 4\times10^{-1} + 2\times10^{-2} + 1\times10^{-3} = 0.421.$

In General

For any numbers:

$$a_n a_{n-1} \ldots . a_1 a_0 . a_{-1} a_{-2} \ldots .. a_m = a_0\times N^0 + a_1\times N^1 + a_{n-1}\times N^{n-1} + a_n\times N^n + a_{-1}\times N^{-1} + a_{-2}\times N^{-2} + a_m\times N^m$$

Where; N is the base of the system.

2. Binary Number Systems

It is said to be of base (2) since it uses 2 digits {0, 1}.

Binary to Decimal Conversion

Ex. : $[11010.11]_2 = 0\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^3 + 1\times2^4 + 1\times2^{-1} + 1\times2^{-2} = (26.75)_{10}.$

In General

To convert r – base number system to decimal number:

$$a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} = a_0\times r^0 + a_1\times r^1 + a_2\times r^2 + a_{-1}\times r^{-1} + a_{-2}\times r^{-2} + a_{-3}\times r^{-3}$$

Ex. : $[4021.2]_5 = 1\times5^0 + 2\times5^1 + 0\times5^2 + 4\times5^3 + 2\times5^{-1} = (511.4)_{10}$.

Decimal to Binary Conversion

Ex. 1: Convert the decimal number [41] to binary number

The arithmetic process can be manipulated more conveniently as follows:

Integer	Remainder
$41 \div 2$	
$20 \div 2$	1
$10 \div 2$	0
$5 \div 2$	0
$2 \div 2$	1
1	0
	1 $\downarrow$

$\longleftarrow$

$(101001)_2$ = answer

So, $[41]_{10} = (101001)_2$

Ex. 2: Convert the decimal number [27.15] to binary number

The arithmetic process can be manipulated more conveniently as follows:

Integer	Remainder
$27 \div 2$	
$13 \div 2$	1
$6 \div 2$	1
$3 \div 2$	0
1	1
	1 $\downarrow$

$\longleftarrow$

$(11011)_2$ = answer

$[27]_{10} = (11011)_2$

Fraction	Coefficient
$0.15\times2 = 0.3$	0
$0.3\times2 = 0.6$	0
$0.6\times2 = 1.2$	1
$1.2\times2 = 0.4$	0
$0.4\times2 = 0.8$	0
$0.8\times2 = 1.6$	1
$1.6\times2 = 1.2$	1 $\downarrow$

$\longrightarrow$

$(0.0010011)_2$ = answer

$[0.15]_{10} = (0.0010011)_2$

So, $[27.15]_{10} = (11011.0010011)_2$

Note: Conversion from decimal integers to any base-*r* system is similar to this example, except that division is done by *r* instead of 2.

Ex. : Convert the decimal number [22.5] to 4 base number system

The arithmetic process can be manipulated more conveniently as follows:

Integer	**Remainder**	
$22 \div 4$		
$5 \div 4$	2	$\downarrow$
1	1	$\downarrow$
	1	$\downarrow$

$\longleftarrow$

$(112)_4$ = answer

$[22]_{10} = (112)_4$

Fraction	**Coefficient**	
$0.5 \times 4 = 2.0$	2	$\downarrow$
$2.0 \times 4 = 0.0$	0	$\downarrow$

$\longrightarrow$

$(0.20)_4$ = answer

$[0.5]_{10} = (0.2)_4$

So, $[22.5]_{10} = (112.2)_4$

H.W: Convert the following numbers:

1. $(645.34)_{10} \longrightarrow (\ )_2$
2. $(153.531)_{10} \longrightarrow (\ )_6$
3. $(11101.1101)_2 \longrightarrow (\ )_{10}$

3. Octal Number Systems

It is said to be of base (8) since it uses 8 digits {0, 1, 2,…..,7}.

Ex. 1: $[231]_8 = 1\times8^0+3\times8^1+2\times8^2 = (153)_{10}$.

Ex. 2: Convert the decimal number [245.5] to octal number system

The arithmetic process can be manipulated more conveniently as follows:

Integer	**Remainder**
245÷ 8	
30 ÷ 8	5 $\downarrow$
3	6
	3

$\longleftarrow$

$(365)_8$ = answer

$[245]_{10} = (365)_8$

Fraction	**Coefficient**
0.5×8= 4.0	4 $\downarrow$
4.0 × 8=0.0	0

$\longrightarrow$

$(0.40)_8$ = answer

$[0.5]_{10} = (0.4)_4$

So, $[245.5]_{10} = (365.4)_8$

4. Hexadecimal Number Systems

It is said to be of base (16) since it uses 16 digits {0, 1, 2,…..,9, A, B, C, D, E, F}.

Ex. 1 : $[2C.4A]_{16} = 12\times16^0+2\times16^1+4\times16^{-1}+10\times16^{-2} = (44.2)_{10}$.

Ex. 2 : Convert the decimal number [165.25] to hexadecimal number system

The arithmetic process can be manipulated more conveniently as follows:

Integer	**Remainder**
165÷ 16	
(10) A	5 $\downarrow$
	A

$\longleftarrow$

$(A5)_{16}$ = answer

$[165]_{10} = (A5)_{16}$

Fraction	**Coefficient**
$0.25 \times 16 = 4.0$	4
$4.0 \times 16 = 0.0$	0 $\downarrow$

$\overrightarrow{(0.40)_{16}}$ = answer

$[0.25]_{10} = (0.4)_{16}$

So, $[165.25]_{10} = (A5.4)_{16}$

H.W 1: Convert the following numbers:

1. $(33.22)_4 \longrightarrow (\ )_8$
2. $(BA.C)_{16} \longrightarrow (\ )_7$

H.W 2: In base [13], list the numbers between (4 and 40).

Conversion between Binary & Octal systems

The conversion between Binary and Octal is accomplished by partitioning the binary number into groups of three digits.

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ex. 1: Convert the octal number [63.4] to binary number system

$(63.4)_8 = (110011.100)_2$

Ex. 2: Convert the binary number (1011011.11011) to octal system

$(001011011.110110)_2 = (133.66)_8$

Conversion between Binary & Hexadecimal systems

The conversion between Binary and Hexadecimal is accomplished by partitioning the binary number into groups of four digits.

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

Ex. 1: Convert the Hexadecimal number [F67.19] to binary number system

$(F67.19)_{16} = (111101100111.00011001)_2$

Ex. 2: Convert the binary number (10100111011.0110101) to hexadecimal system

$(010100111011.01101010)_2 = (53B.6A)_{16}$

H.W: Convert the following numbers:

1. $(11011.1001)_2 \longrightarrow (\ )_8$
2. $(DF3.C5)_{16} \longrightarrow (\ )_2$

Arithmetic operations

Binary Arithmetic

1) Addition operation

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0; and carry 1 to the next column.

Ex. : Add; $(1011.01 + 101.101)_2$

```
1011.010   +
0101.101
__________
(10000.111)₂
```

2) Subtraction operation

0 – 0 = 0

0 – 1 = 1; and barrow 1 from the next column.

1 – 0 = 1

1 – 1 = 0

Ex. : Subtract; $(1000.01 - 11.001)_2$

```
1000.010    –
0011.001
__________
(101.001)₂
```

3) Multiplication operation

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Ex. : Multiply; $(11.01 \times 1.01)_2$

```
1101        ×
  101
__________
    1101     +
  00000
110100
__________
(100.0001)₂
```

4) Devision operation

$0 \div 0 =$ undefined

$1 \div 0 =$ undefined

$0 \div 1 = 0$

$1 \div 1 = 1$

Ex. : Divide; $(1101.01 \div 10)_2$

```
      110.101
   ____________
   110101      |  10
   10   -      |________
 ______________
     10
     10   -
 ______________
       0 10
         10   -
 ______________
        0 10
          10   -
 ______________
          00
```

So, $1101.01 \div 10 = (110.101)_2$

R – Base Arithmetic

Ex. : Evaluate the following:

1. $(42.51 + 15.3)_8$ & $(42.51 - 15.3)_8$

```
42.51                  42.51
         +                      –
15.30                  15.30
__________           __________

(60.01)8               (25.21)8
```

2. $(B3 + 4D)_{16}$ & $(B3 - 4D)_{16}$

```
B3                 B3
         +                  –
4D                 4D
__________       __________

(100)16            (66)16
```

H.W: Perform the following operations

1. $(50.27)_9 \div (15.28)_9$
2. $(44.56)_7 + (12.5)_6$
3. $(B3)_{13} - (6.55)_{13}$
4. $(33.24)_5 \times (14.21)_5$

Complements of Numbers

In digital systems, the complements are used to simplify the subtraction operation. There are two types of complements for r – base system:

- r 's complement.
- (r – 1)'s complement.

In binary system, there are 2's complement and 1's complement which represent the negative form of binary number.

1. The first complement (1's) are changed zeros to ones and ones to zeros.

Ex. : $(01001.1101)_2 \xrightarrow{1's} (10110.0010)_2$

2. The second complement (2's) can be either leaving least significant zeros and ones digit unchanged then replacing 1's to 0's and 0's to 1's; or by forming 1's complement and adding {1} for the least significant bit.

Ex. : $(110101)_2 \xrightarrow{2's} (\ )_{2's}$

$(110101)_2 \xrightarrow{1's} (001010\ )_{1's} + 1 = (001011)_{2's}$

Subtraction using Complements

In digital computer, if the subtraction implemented, we use the complements and addition as shown:

1) Convert the second number using 1's or 2's complement.
2) Replace the subtraction operation to addition operation.

Ex. 1: Perform the following operation using 2's complement.
$(1010100)_2 – (1000100)_2$

$(1000100)_2 \xrightarrow{1's} (0111011)_{1's} +1 = (0111100)_{2's}$

$$\begin{array}{r} 1010100 \\ + \quad 0111100 \\ \hline \overset{\curvearrowright}{1}\,0010000 \end{array}$$

….. $(0010000)_2$

Ignored

Ex. 2: Perform the following operation using 2's complement.
$(1010011.01)_2 - (0101100.10)_2$

$(0101100.10)_2 \xrightarrow{\text{1's}} (1010011.01)_{1's} + 1 = (1010011.10)_{2's}$

$$\begin{array}{r} 1010011.01 \\ + \\ 1010011.10 \\ \hline 1\;0100110.11 \end{array} \quad \ldots\ldots (0100110.11)_2$$

Ignored

Ex. 3: Perform the following operation using 1's complement.
$(1010111)_2 - (0110110)_2$

$(0110110)_2 \xrightarrow{\text{1's}} (1001001)_{1's}$

$$\begin{array}{r} 1010111 \\ + \\ 1001001 \\ \hline 1\;0100000 \\ \hookrightarrow 1 \quad + \\ \hline (100001)_2 \end{array}$$

Ex. 4: Find the 12's complement and 13's complement to the number $[B65.5C]_{13}$.

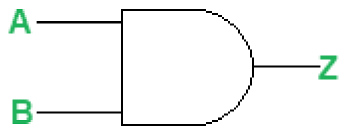
$$\begin{array}{r} CCC.CC \\ - \\ B65.5C \\ \hline (167.70)_{12's} \end{array} \quad + 1 = (167.71)_{13's}$$

Binary Logic Gates

1. AND Gate:

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

$\left.\right\}$ AND Gate Truth Table

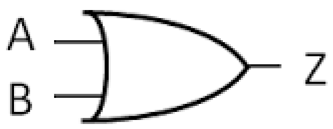


Z = A . B

2. OR Gate:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

$\left.\right\}$ OR Gate Truth Table

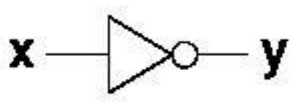


Z = A + B

3. NOT Gate:

x	y
0	1
1	0

$\left.\right\}$ NOT Gate Truth Table

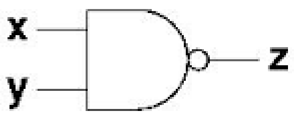


$y = \overline{x}$

4. NAND Gate:

x	y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NAND Gate Truth Table

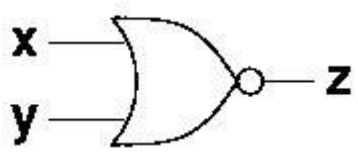


$z = \overline{x \cdot y}$

5. NOR Gate:

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

NOR Gate Truth Table

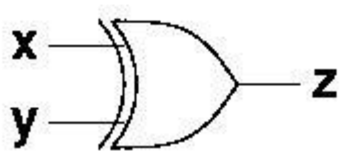


$z = \overline{x + y}$

6. Exclusive OR (Ex – OR) Gate:

x	Y	z
0	0	0
0	1	1
1	0	1
1	1	0

Ex - OR Gate Truth Table

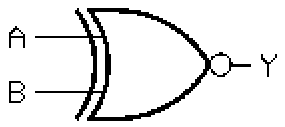


$z = x'y + xy' = x \oplus y$

7. Exclusive NOR (Ex – NOR) Gate:

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

Ex - NOR Gate Truth Table



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

CHAPTER TWO: BOOLEAN ALGEBRA

BASIC DEFINITIONS

Boolean algebra, like any other mathematical system, may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates. A *set* of elements is any collection of objects, usually having a common property. If S is a set, and x and y are certain objects, then the notation $x \in S$ means that x is a member of the set S and $y \notin S$ means that y is not an element of S.

A binary operator defined on a set S of elements is a rule that assigns, to each pair of elements from S, a unique element from S. As an example, consider the relation $a * b = c$. We say that $*$ is a binary operator if it specifies a rule for finding c from the pair (a, b) and also if $a, b, c \in S$. However, $*$ is not a binary operator if $a, b \in S$, and if $c \notin S$.

Boolean algebra can be used to help analyze a logic circuit and express its operation mathematically. There are four assumptions which are:

1. 0.0 = 0
   0.1 = 0
   1.0 = 0
   1.1 = 1
2. 0+0 = 0
   0+1 = 1
   1+0 = 1
   1+1 = 1
3. 0' = 1
   1' = 0
4. If x = 1 then x ≠ 0
   If x = 0 then x ≠ 1

Rules of Boolean Algebra

The basic thermoses and laws of Boolean Algebra:

1) Commutative Law

A . B = B .A

A + B = B +A

2) Associative Law

A . (B . C) = (A . B) . C

A + (B + C) = (A + B) + C

3) Distribution Law

A . (B + C) = (A . B) + (A . C)

A + (B . C) = (A + B) . (A + C)

4) A + 0 = A

A + 1 = 1

A . 0 = 0

A . 1 = A

5) A + A = A

A + A' = 1

A . A = A

A . A' = 0

6) Inversion Law

A'' = A

7) A + (A . B) = A

Proof : $A\ (\cancelto{1}{1+B}) = A \cdot 1 = A$

A + (A' . B) = A + B

Proof : $(\cancelto{1}{A+A'}) \cdot (A + B) = A + B$

8) DE Morgan's Theorem

A) $\overline{A \cdot B} = A' + B'$

Proof :

A	B	A . B	(A . B)'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	A'	B'	A'+B'
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

In General

$\overline{A \cdot B \cdot C \cdot D \cdot \ldots\ldots} = A' + B' + C' + D' + \ldots\ldots.$

B) $\overline{A + B} = A' . B'$

Proof :

A	B	A + B	(A + B)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

A	B	A'	B'	A'.B'
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

In General

$$\overline{A + B + C + D \ldots\ldots} = A' . B' . C' . D' . \ldots\ldots$$

Ex. : Prove that:

1. a'b + b'c + ac' = ab' + bc' + a'c

The left side:

= a'b(c + c') + (a + a')b'c + a(b + b')c'

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

$= ab'\underbrace{(c + c')}_{1} + bc'\,\underbrace{(a + a')}_{1} + a'c\,\underbrace{(b + b')}_{1}$

= ab' + bc' + a'c = The right side.

2. (x⊕y) ⊕z = x ⊕(y ⊕z)

The left side:

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

= xy'z' +x'yz' + ((x' + y).(x + y'))z

$= xy'z' + x'yz' + (\underbrace{xx'}_{0} + x'y' + xy + \underbrace{yy'}_{0})z$

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

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

= x (y ⊕z)' + x' (y ⊕z)

= x ⊕(y ⊕z) = The right side.

H.W 1: Simplify the following functions:

1. F_1 = AB'C + BC'D + ACD + BCD
2. F_2 = y + x'y'z + xzw' + xz'

H.W 2: Prove that:

ABC' + A'BC + ABC = B (A + C).

Canonical and Standard Forms

The logic function can be expressed by two standard form:

1. **Sum of Products (SoP) :** It is possible to implement the logic functions in sum of products method depending on the output, it the output equals logic 1. F = 1.
2. **Product of Sums (PoS):** It is possible to implement the logic functions in product of sums method depending on the output, it the output equals logic 0. F = 0.

Ex. 1: Find the canonical forms in SoP and PoS for the following function:

F(x,y,z) = xy + x'y' + z'

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

= xyz + xyz' + x'y'z + x'y'z' + (xy + xy' + x'y + x'y')z'

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

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

F (SoP) = $\sum(0,1,2,4,6,7)$

F (PoS) = $\prod(3,5)$

Ex. 2: Find the canonical forms in SoP and PoS for the following function:

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

= (A + B' + CC')(A + B + C')

= (A + B' + C)(A + B' + C')(A + B + C')

F (PoS) = $\prod(1,2,3)$

F (SoP) = $\sum(0,4,5,6,7)$

H.W : Convert the following logic function to SoP & PoS:

1. F = (x + y'z'w')(x'y' + y (z' + w))

2. F = $\overline{(A + B)'AB}$

CHAPTER THREE: GATE LEVEL MINIMIZATION

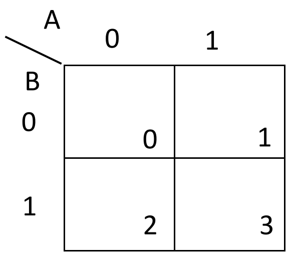
The Karnough Map Method

Karnough Map (K – M) is used to simplify the logic functions to reduce the Boolean functions with the aid of number of rules. Karnough Maps consist of 2^n cell depending on the (n) variables of the logic functions.

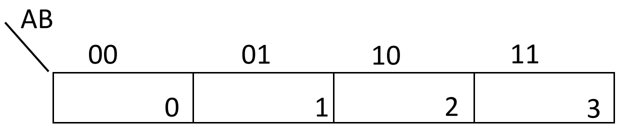
1) 2 – variables Karnough Map:

n = 2 $\Longrightarrow$ no. of cells = $2^n = 2^2 = 4$

For F(A, B)

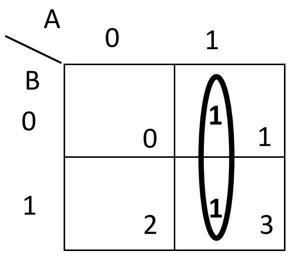


Or



Ex. : Using K – M to simplify the function :

$F = \sum(1,3)$

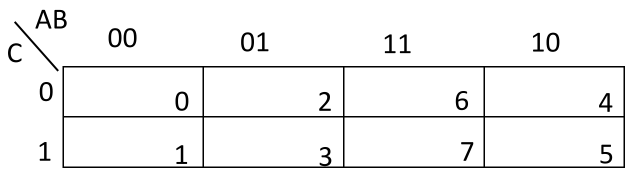


F = A

2) 3 – variables Karnough Map:

n = 3 $\Longrightarrow$ no. of cells = $2^n = 2^3 = 8$

For F(A, B, C)

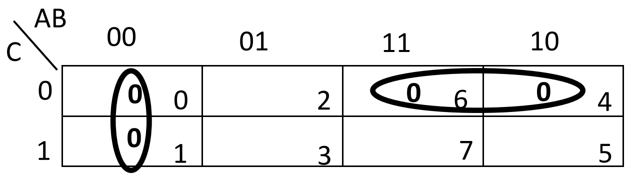


Or

B \ A	0	1
00	0	4
01	1	5
11	3	7
10	2	6

Ex. : Using K – M to simplify the function :

F = $\prod(0;1,4,6)$



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

3) 4 – variables Karnough Map:

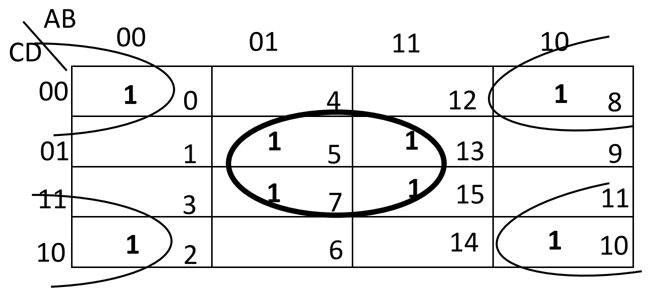
n = 4 $\Longrightarrow$ no. of cells = $2^n = 2^4 = 16$

For F(A, B, C, D)

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

Ex. : Using K – M to simplify the function :

F = $\sum(0,2,5,7,8,10,13,15)$



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

F = B'D' + BD = (B ⊕ D)'

4) 5 – variables Karnough Map:

n = 5 $\Longrightarrow$ no. of cells = $2^n = 2^5 = 32$

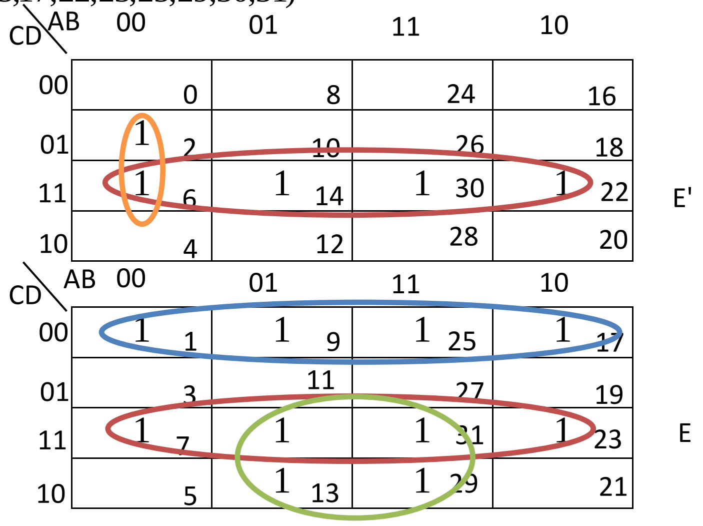
For F(A, B, C, D, E)

CD \ AB	00	01	11	10	
00	0	8	24	16	
01	2	10	26	18	
11	6	14	30	22	E'
10	4	12	28	20	

CD \ AB	00	01	11	10	
00	1	9	25	17	
01	3	11	27	19	
11	7	15	31	23	E
10	5	13	29	21	

Ex. : Using K – M to simplify the function :

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



CD \ AB	00	01	11	10	
00	0	8	24	16	
01	1 2	10	26	18	
11	1 6	1 14	1 30	1 22	E'
10	4	12	28	20	

CD \ AB	00	01	11	10	
00	1 1	1 9	1 25	1 17	
01	3	11	27	19	
11	1 7	1	1 31	1 23	E
10	5	1 13	1 29	21	

F = CD + C'D'E + BCE + A'B'DE'

H.W : Simplify the following logic functions using K - Maps:

1. F = A'B'C +ABC'D' + AD' + C

2. F = $\sum(1,2,5,8,10,14)$

Don't-Care Terms

The don't care terms will referred to the combination of variables will never occur. The don't care terms can be expressed by many symbols such as [ϕ, X, D]. When the don't care terms are appeared in the karnough maps, the resulting expression will be simplified.

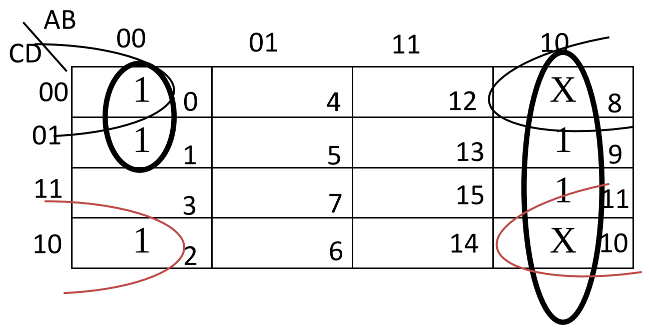
Ex. 1: Simplify the following expression:

F = A'B'C' +A'B'CD' + AB'D + $\sum\phi(ABC + AB'D')$

= A'B'C'(D + D') +A'B'CD' + AB'(C + C')D + $\sum\phi(ABC(D + D') + AB'(C + C')D')$

= A'B'C'D + A'B'C'D' +A'B'CD' + AB'CD + AB'C'D + $\sum\phi(ABCD + ABCD' + AB'CD' + AB'C'D')$

F= $\sum(0,1,2,9,11) + \sum\phi(8,10,14,15)$

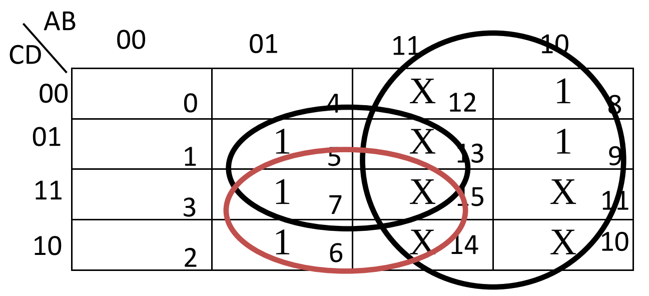


F = AB' + B'D' + A'B'C'

Ex. 2: Simplify the following function:

F= $\sum(5,6,7,8,9) + \sum\phi(10,11,12,13,14,15)$

F = A + BC + BD



H.W : Simplify the following logic functions:

1. $F_1 = x'y'z' + x'y'z' + xy'z'w + D(x'yzw' + xyw')$

2. $F_2 = \prod(1,4,5,9,10,11,14) + \prod D(0,2,7,15)$

NAND and NOR Implementation

NAND Gate and NOR Gate only can be used to implement the Boolean functions:

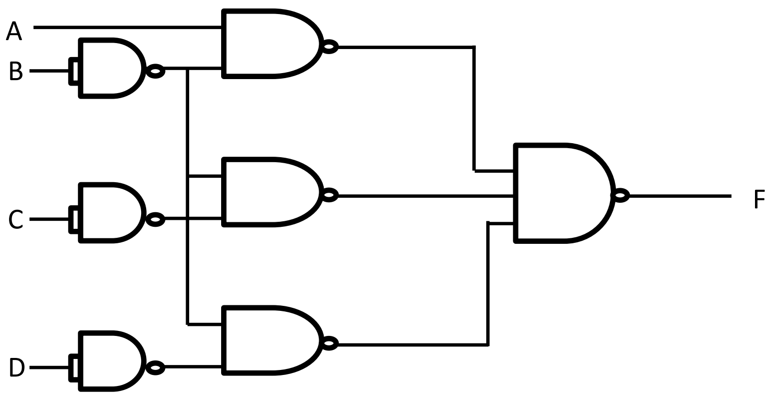
1) NAND Gate only:

Ex. : Simplify the following function using NAND Gate only:

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

$F = \overline{\overline{AB' + B'C' + B'D'}}$

$F = \overline{\overline{(AB')} \,.\, \overline{(B'C')} \,.\, \overline{B'D'}}$



2) N0R Gate only:

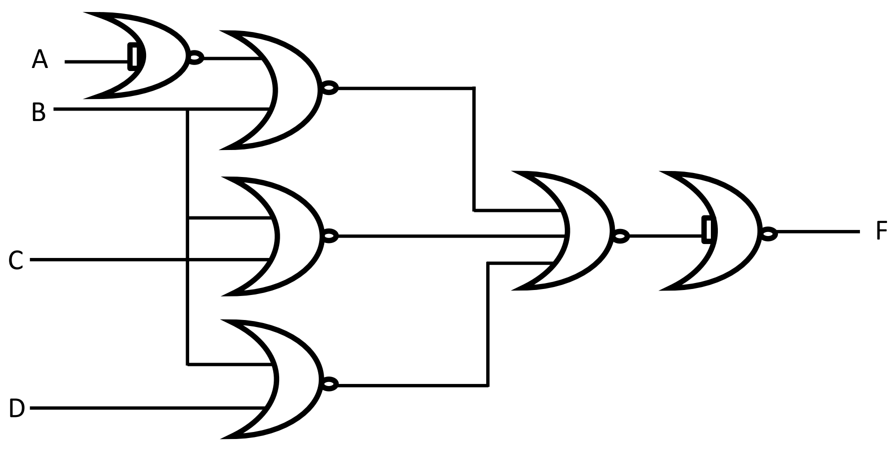
Ex. : Simplify the following function using NOR Gate only:

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

$F = \overline{\overline{AB'}} + \overline{\overline{B'C'}} + \overline{\overline{B'D'}}$

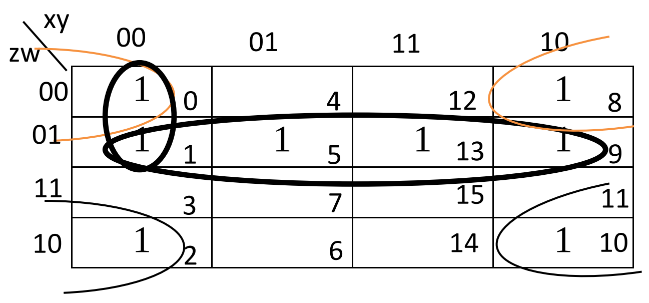
$F = \overline{\overline{(A'+B'')} + \overline{(B''+C'')} + \overline{(B''+D'')}}$

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



Ex. : Using NAND Gate only and NOR Gate only to simplify the following function:

F= $\sum(0,1,2,5,8,9,10,13)$

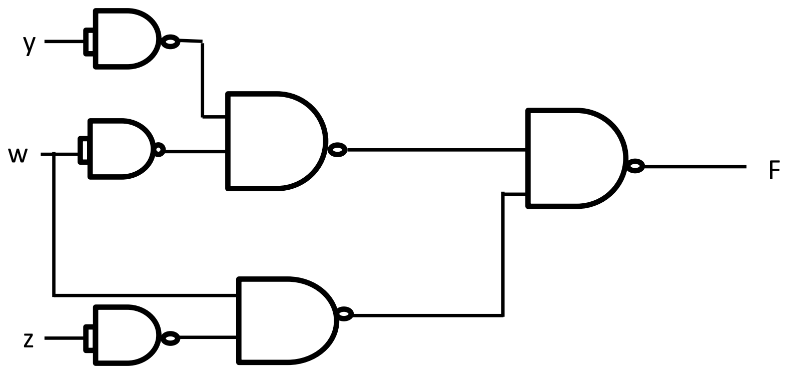


F = y'w' + z'w

Using NAND Gate:

$$F = \overline{\overline{y'w' + z'w}}$$

$$= \overline{\overline{y'w'} \cdot \overline{z'w}}$$

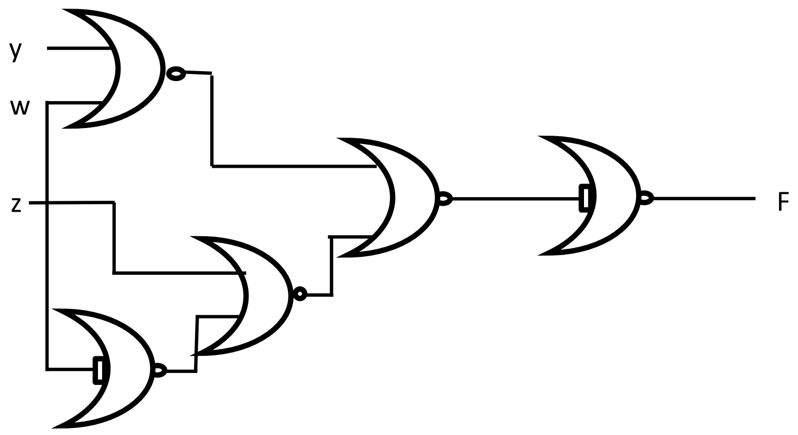


Using NOR Gate:

$$F = \overline{\overline{y'w'}} + \overline{\overline{z'w}}$$

$$= \overline{(y''+w'')} + \overline{(z''+w')}$$

$$= \overline{(y+w)} + \overline{(z+w')}$$

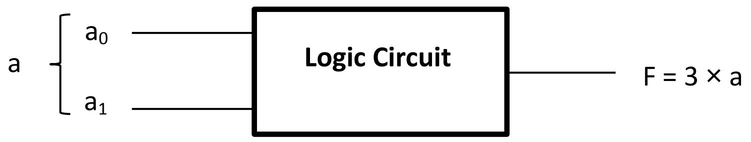


H.W : Implement the following logic functions using NAND and NOR Gates only:

1. $F_1 = (A' + B + C).(A + C)$
2. $F_2 = \sum(0,1,2,4,8,13,14,15)$
3. $F_3 = \prod(1,4,5,9,10,11,14)$

Logic Circuits

Ex. 1: Design a logic circuit that has two bits binary number, the outputs represent [3 × input number].



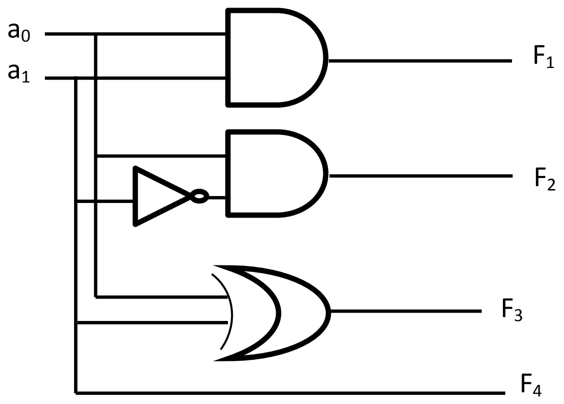
a_0	a_1	F_1	F_2	F_3	F_4
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	1	0
1	1	1	0	0	1

$F_1 = a_0a_1$

$F_2 = a_0a_1'$

$F_3 = a_0'a_1 + a_0a_1' = a_0 \oplus a_1$

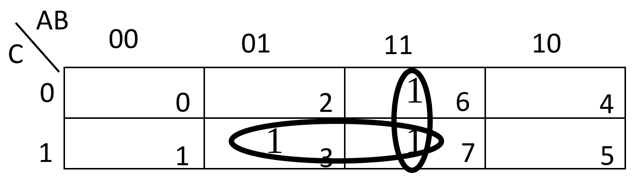
$F_4 = a_0'a_1 + a_0a_1 = a_1\underbrace{(a_0' + a_0)}_{1} = a_1$



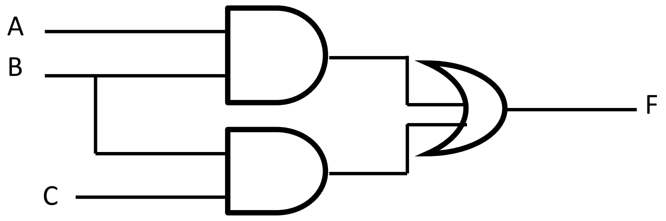
Ex. 2: Design a logic circuit using minimum numbers of logic gates. The logic circuit has three inputs and one output, the output is active only if two adjacent ones appear at the input.

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

$F = \sum(3,6,7)$



F = AB + BC



H.W : Design a logic circuit to produce an output F = 1 if and only if the input which is represented by 4 – bits binary number greater than (13) and less than (4).