

[Unit-1] Digital Logical Circuit

Digital Computers

A Digital computer can be considered as a digital system that performs various computational tasks.

The first electronic digital computer was developed in the late 1940s and was used primarily for numerical computations.

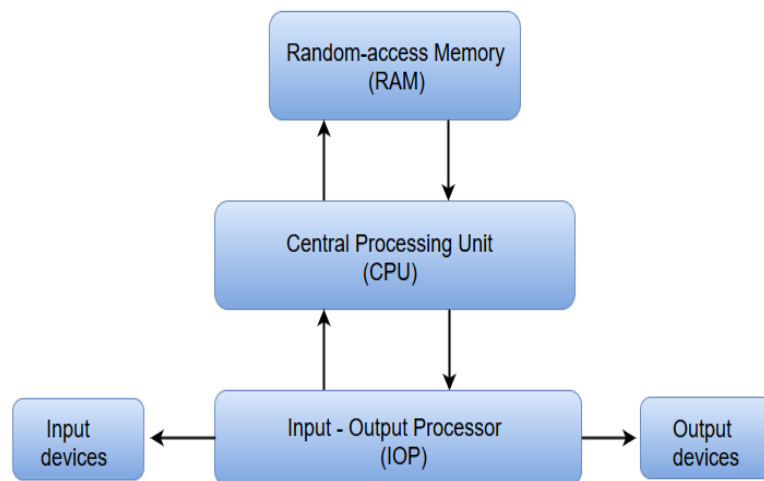
By convention, the digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit.

A computer system is subdivided into two functional entities: Hardware and Software.

The hardware consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.

The software of the computer consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

Block diagram of a digital computer:

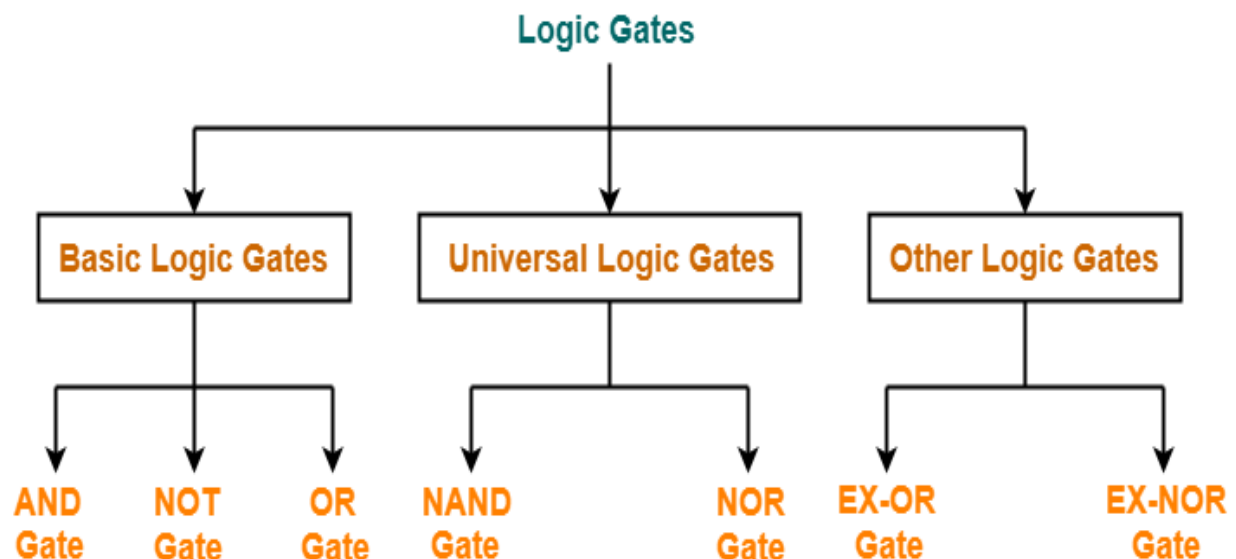


- The Central Processing Unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and a control circuit for fetching and executing instructions.
- The memory unit of a digital computer contains storage for instructions and data.

- The Random Access Memory (RAM) for real-time processing of the data.
- The Input-Output devices for generating inputs from the user and displaying the final results to the user.
- The Input-Output devices connected to the computer include the keyboard, mouse, terminals, magnetic disk drives, and other communication devices.

Logic Gates

- The logic gates are the main structural part of a digital system.
- Logic Gates are a block of hardware that produces signals of binary 1 or 0 when input logic requirements are satisfied.
- Each gate has a distinct graphic symbol, and its operation can be described by means of algebraic expressions.
- The seven basic logic gates includes: AND, OR, XOR, NOT, NAND, NOR, and XNOR.
- The relationship between the input-output binary variables for each gate can be represented in tabular form by a truth table.
- Each gate has one or two binary input variables designated by A and B and one binary output variable designated by x.

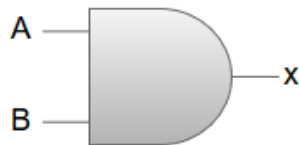


Types of Logic Gates

AND GATE:

The AND gate is an electronic circuit which gives a high output only if all its inputs are high. The AND operation is represented by a dot (.) sign.

AND Gate:



Algebraic Function: $x = AB$

Truth Table:

A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:

The OR gate is an electronic circuit which gives a high output if one or more of its inputs are high. The operation performed by an OR gate is represented by a plus (+) sign.

OR Gate:



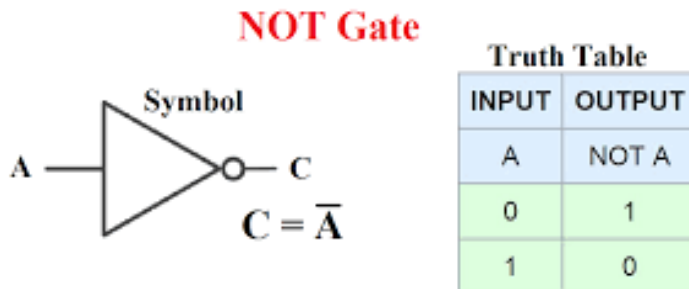
Algebraic Function: $x = A + B$

Truth Table:

A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

The NOT gate is an electronic circuit which produces an inverted version of the input at its output. It is also known as an **Inverter**.



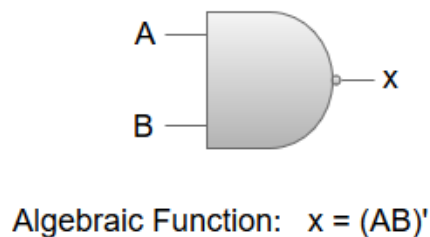
ProjectBot123.com

NAND GATE:

The NOT-AND (NAND) gate which is equal to an AND gate followed by a NOT gate. The NAND gate gives a high output if any of the inputs are low. The NAND gate is represented by a AND gate with a small circle on the output. The small circle represents inversion.

NAND Gate:

Truth Table:



A	B	x
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

The NOT-OR (NOR) gate which is equal to an OR gate followed by a NOT gate. The NOR gate gives a low output if any of the inputs are high. The NOR gate is represented by an OR gate with a small circle on the output. The small circle represents inversion.

NOR Gate:



Algebraic Function: $x = (A+B)'$

Truth Table:

A	B	x
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive-OR/ XOR GATE:

The 'Exclusive-OR' gate is a circuit which will give a high output if one of its inputs is high but not both of them. The XOR operation is represented by an encircled plus sign.

XOR Gate:



Algebraic Function: $x = A \oplus B$
or
 $x = A'B + AB'$

Truth Table:

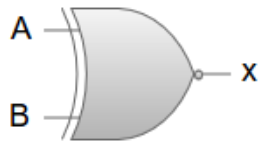
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE-NOR/Equivalence GATE:

The 'Exclusive-NOR' gate is a circuit that does the inverse operation to the XOR gate. It will give a low output if one of its inputs is high but not both of them. The small circle represents inversion.

Exclusive-NOR Gate:

Truth Table:






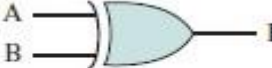


Algebraic Function: $x = (A \oplus B)'$

or

$$x = A'B' + AB$$

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \cdot B$ or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \overline{A}$ or $F = A'$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{AB}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$F = A \oplus B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

De Morgan's Theorems

Boolean e Morgan has suggested two theorems which are extremely useful in Boolean Algebra. The two theorems are discussed below.

Theorem 1

- The left hand side (LHS) of this theorem represents a NAND gate with inputs A and B, whereas the right hand side (RHS) of the theorem represents an OR gate with inverted inputs.
- This OR gate is called as **Bubbled OR**.

Table showing verification of the De Morgan's first theorem –

A	B	\overline{AB}	\overline{A}	\overline{B}	$\overline{A} + \overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Theorem 2

- The LHS of this theorem represents a NOR gate with inputs A and B, whereas the RHS represents an AND gate with inverted inputs.
- This AND gate is called as **Bubbled AND**.

A	B	$\overline{A+B}$	\overline{A}	\overline{B}	$\overline{A} \cdot \overline{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

Boolean algebra

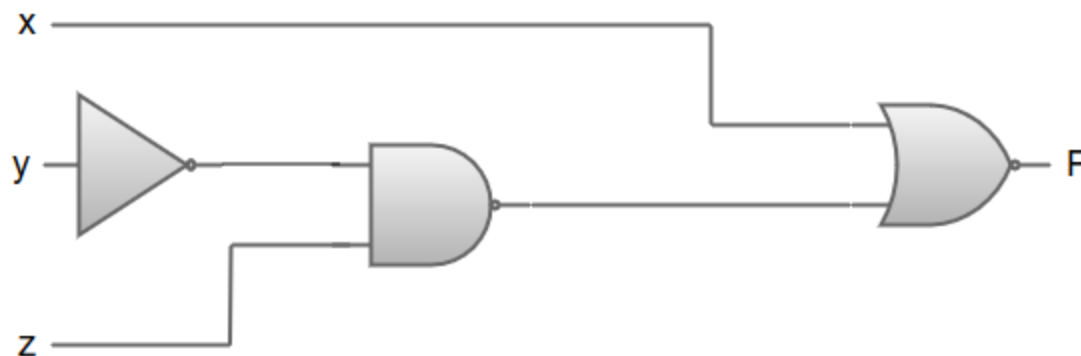
Boolean algebra can be considered as an algebra that deals with binary variables and logic operations. Boolean algebraic variables are designated by letters such as A, B, x, and y. The basic operations performed are AND, OR, and complement.

The Boolean algebraic functions are mostly expressed with binary variables, logic operation symbols, parentheses, and equal sign. For a given value of variables, the Boolean function can be either 1 or 0. For instance, consider the Boolean function:

$$F = x + y'z$$

The logic diagram for the Boolean function $F = x + y'z$ can be represented as:

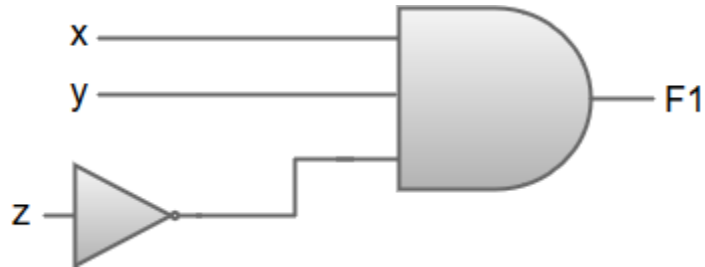
$$\mathbf{F = x + y'z}$$



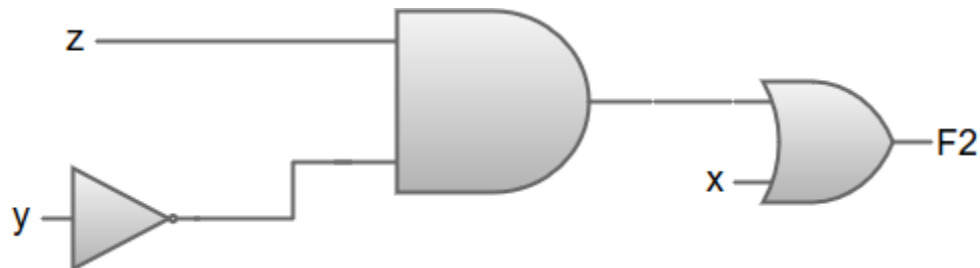
Examples of Boolean algebra simplifications using logic gates

In this section, we will look at some of the examples of Boolean algebra simplification using Logic gates.

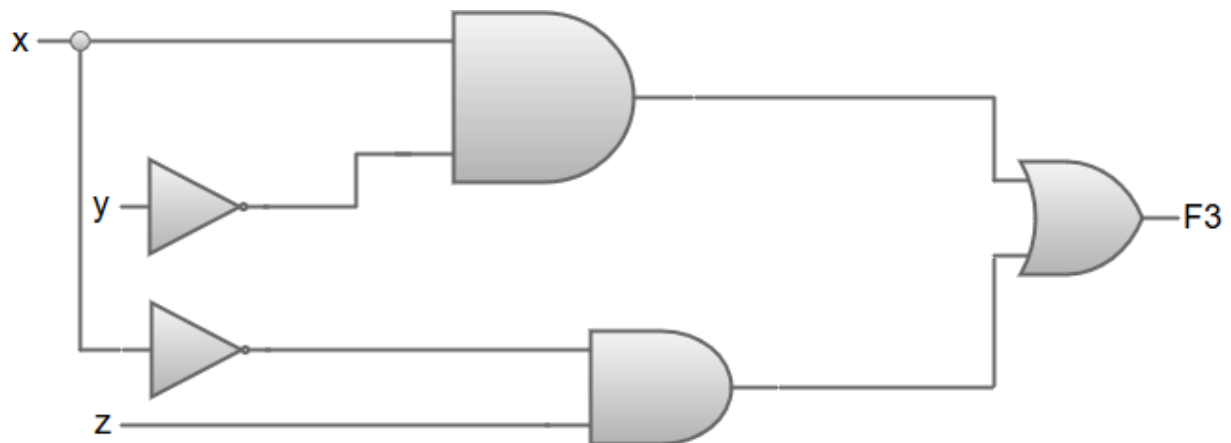
1. $F1 = xyz'$



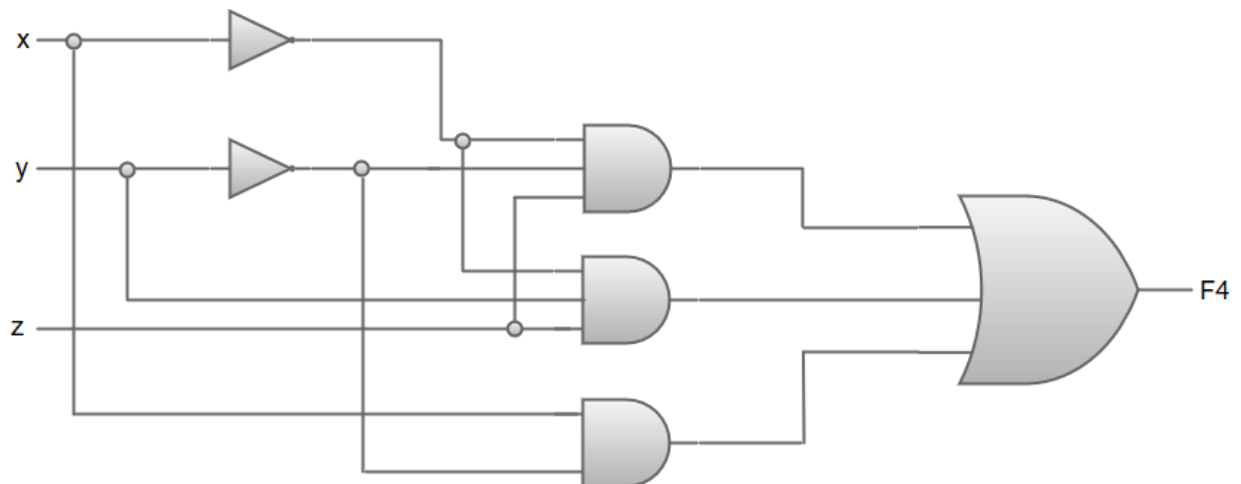
2. $F2 = x + y'z$



3. $F3 = xy' + x'z$



4. $F4 = x'y'z + x'yz + xy'$



→CANONICAL OR STANDARD FORMS

There are two ways to represent boolean function, one is standard (canonical) sum of products form (SOP), and another is the standard product of sums form (POS). These forms are important because functional comparison can be made using these forms and they are also useful to minimize the boolean functions.

➤ 1. Sum of products form (SOP)

It is expression in which one or more product terms are logically added. In this kind of expression variables may or may not be complemented.

Ex.

Term $X.Y.Z$ are the examples of product term, logical addition of these two product terms $X.Y.Z. + X'Y'Z'$ are known as sum of product expression.

Ex1. $F = A + BC$

Ex2. $F = ABC + A'BC + AB'C + ABC'$

Ex3. $F = XYZ + X'Y'Z' + XY' + Z$

Min terms

Sum of products function is also known as MIN terms. Each minterms obtained from an AND terms of the n variables, with each variable being primed if the corresponding bit of binary number is 0 and unprimed if a 1.

➤ 2. Products of Sum form (POS)

It is expression in which one or more product terms are logically multiplied. In this kind of expression variables may or may not be complemented.

Ex.

Term $X+Y+Z$ and $X'+Y'+Z'$ are sum terms, then logical multiplication of these two sum terms $(X+Y+Z).(X'+Y'+Z')$ are known as product of sum expression.

Ex1. $F=(A+B).C$

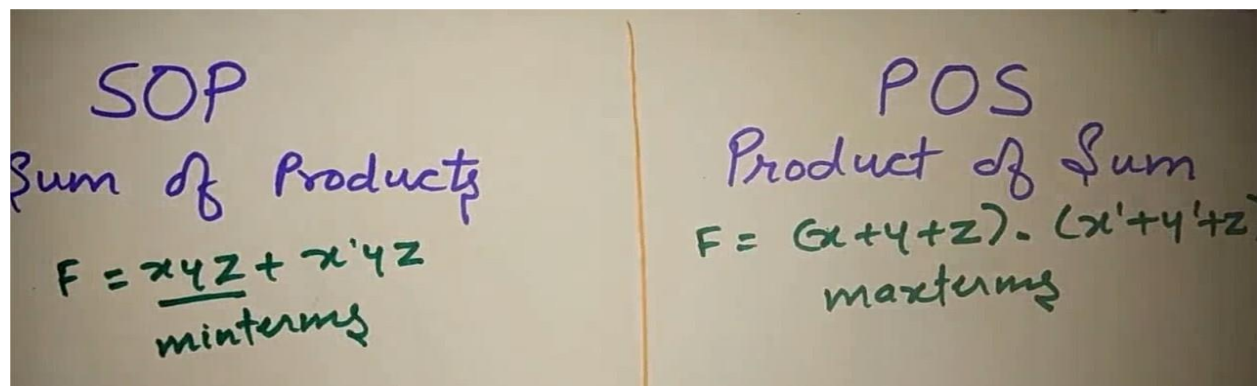
Ex2. $F=(ABC).(A'BC).(AB'C).(ABC')$

Ex3. $F=(XYZ).(X'Y'Z').(XY').(Z)$

MAX terms

Products of Sum function is also known as MAX terms. Each maxterms obtained from an OR terms of the n variables, with each variable being primed if the corresponding bit of binary number is 1 and unprimed if a 0.

1.Min terms and Max Terms in SOP and POS.



2. Truth table in SOP and POS. [Ex.(SOP) $0 \rightarrow X'$ and $1 \rightarrow X$]

Sum of Products

$F = \underline{x y z} + x' y z$
minterms

x	y	F
0	0	1
0	1	0
1	0	1
1	1	0

$x' y'$
 $x' y$
 $x y'$
 $x y$

POS

Product of Sum

$F = (x + y + z) \cdot (x' + y)$
maxterms

3. SOP $\rightarrow 0$ is a compliment and 1 is don't. POS $\rightarrow 1$ is a compliment and 0 is don't.

Sum of Products

$F = \underline{x y z} + x' y z$
minterms

x	y	F
0	0	1
0	1	0
1	0	1
1	1	0

$x' y'$
 $x' y$
 $x y'$
 $x y$

POS

Product of Sum

$F = (x + y + z) \cdot (x' + y)$
maxterms

$x + y$
 $x + y'$
 $x' + y$
 $x' + y'$

4. SOP is symbol Σ and POS are Π symbol to complete statement.

Products
 $x'y'z$
 xyz

POS
Product of Sum
 $F = (x+y+z) \cdot (x'+y'+z)$
maxterms

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

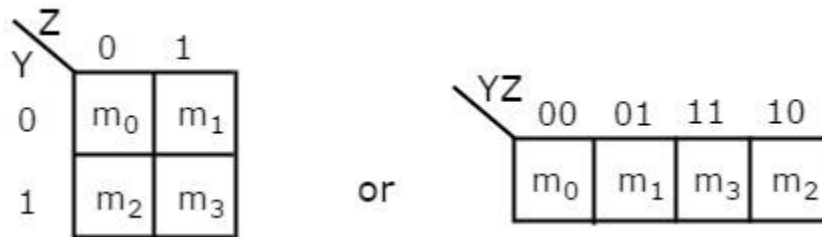
SOP $F = \Sigma(0, 2)$
 $F = x'y' + xy'$
POS $F = \Pi(1, 3)$
 $F = (x+y)(x'+y')$

→ KARNAUGH MAP (K-MAP)

- The boolean function may be simplified using algebra method, but this method is sometime difficult because it lacks specific rules for predicting each succeeding step.
- The map method provides a simple and straight forward procedure for minimizing Boolean functions.
- This method is regarded either as a pictorial form of a truth table or as an extension of venn diagram.
- This method is known as a karnaugh map (K-MAP) or as Veitch Diagram.
- Two, three and four variable k-map are specification.

1.K-MAP for 2 variable

The number of cells in 2 variable K-map is four, since the number of variables is two. The following figure shows **2 variable K-Map**.



As shown in the first row is for a A' and the second row for A. similarly ,the first column is for B' and the second column for B.

Example : $\rightarrow F = AB + B'A + A'B$

2 Variables K-Map

$F = \bar{A}\bar{B} + \bar{B}A + \bar{A}B$ ✓

Handwritten K-maps and simplification:

Top left K-map (A/B):

A \ B	0	1
0	00	01
1	10	11

Top right K-map (A/B):

A \ B	0	1
0	0	1
1	2	3

Bottom left K-map (A/B):

A \ B	\bar{B}	B
\bar{A}	$\bar{A}\bar{B}$	$\bar{A}B$
A	$A\bar{B}$	AB

Handwritten equation: $2^n = 2^2 = 4$

2. K-MAP for 3 variable

The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows **3 variable K-Map**.

X \ YZ	00	01	11	10
0	m ₀	m ₁	m ₃	m ₂
1	m ₄	m ₅	m ₇	m ₆

There is only one possibility of grouping 8 adjacent min terms.

3. K-MAP for 4 variable

The number of cells in 4 variable K-map is sixteen, since the number of variables is four. The following figure shows **4 variable K-Map**.

WX \ YZ	00	01	11	10
00	m ₀	m ₁	m ₃	m ₂
01	m ₄	m ₅	m ₇	m ₆
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
10	m ₈	m ₉	m ₁₁	m ₁₀

There is only one possibility of grouping 16 adjacent min terms.

Binary Table

Sr.no.	BINARY	OCTAL	DECIMAL	HEXA DECIMAL
0	0000	0	0	0
1	0001	1	1	1
2	0010	2	2	2
3	0011	3	3	3
4	0100	4	4	4
5	0101	5	5	5
6	0110	6	6	6
7	0111	7	7	7
8	1000		8	8
9	1001			9
10	1010			A
11	1011			B
12	1100			C
13	1101			D
14	1110			E
15	1111			F

Rule for k-map

- 1) Groups may not include any cell containing a zero.(Only 1)
- 2) Groups may be horizontal or vertical, but not diagonal.
- 3) Groups must contain 1, 2, 4, 8, or in general 2^n cells. ...
- 4) Each group should be as large as possible.
- 5) Each cell containing a one must be in at least one group.
- 6) Groups may overlap.
- 7) Groups may wrap around the table.

The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together adjacent cells containing *ones*

1. Groups may not include any cell containing a **zero**

B \ A	0	1	
	0	0	
1	1		

WRONG ✗

B \ A	0	1	
	0	0	
1	1	1	

RIGHT ✓

2. Groups may be horizontal or vertical, but not diagonal.

B \ A	0	1	
	0	0	1
1	1		0

WRONG ✗

B \ A	0	1	
	0	0	1
1	1	1	

RIGHT ✓

3. Groups must contain 1, 2, 4, 8, or in general 2^n cells.

That is if $n = 1$, a group will contain two 1's since $2^1 = 2$.

If $n = 2$, a group will contain four 1's since $2^2 = 4$.

B \ A	0	1	
	0	1	1
1	0	0	

Group of 2
RIGHT ✓

C \ AB	00	01	11	10	
	0	0	1	1	1
1	0	0	0	0	

Group of 3
WRONG ✗

B \ A	0	1	
	0	1	1
1	1	1	

Group of 4
RIGHT ✓

C \ AB	00	01	11	10	
	0	1	1	1	1
1	0	0	0	1	

Group of 5
WRONG ✗

4. Each group should be as large as possible.

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

RIGHT ✓

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

WRONG ✗

(Note that no Boolean laws broken, but not sufficiently minimal)

5. Each cell containing a **one** must be in at least one group.

AB \ C	00	01	11	10
0	0	0	1	1
1	0	0	0	1

Group I

Group II

1 present in at least one group.

6. Groups may overlap.

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups overlapping.

RIGHT ✓

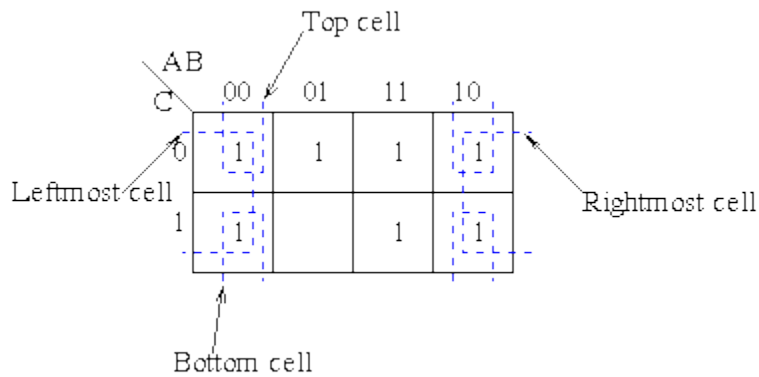
AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups not overlapping.

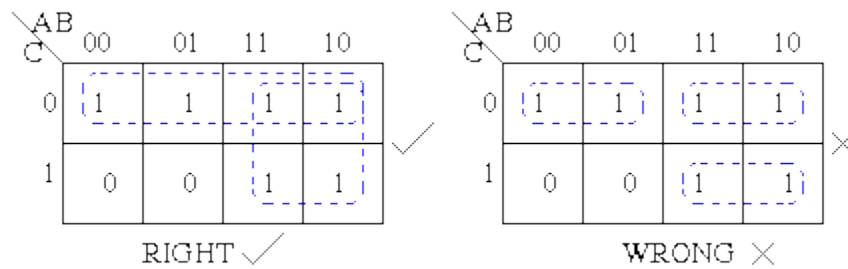
WRONG ✗

7. Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be

grouped with the bottom cell.

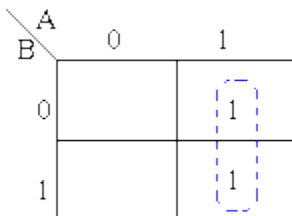


8. There should be as few groups as possible, as long as this does not contradict any of the previous rules.



Example 1:

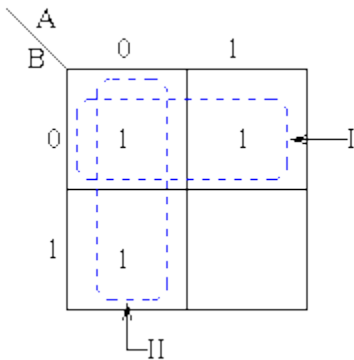
Consider the following map. The function plotted is: $Z = f(A,B) = A\bar{B} + AB$



Ans. Draw the circuit

Example 2:

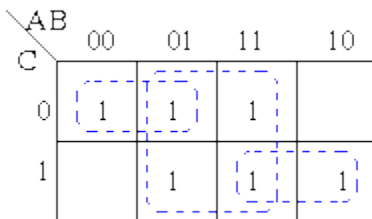
Consider the expression $Z = f(A,B) = \bar{A}\bar{B} + A\bar{B} + \bar{A}B$ plotted on the Karnaugh map:



Ans. Draw the circuit

Example 3:

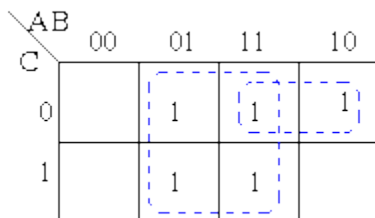
$$Z = f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}B + AB\bar{C} + AC$$



Ans. Draw the circuit

Example 4:

$$Z = f(A,B,C) = \bar{A}B + B\bar{C} + BC + A\bar{B}\bar{C}$$



Ans. Draw the circuit

Example 5: $F(x,y,z) = \Sigma(2,3,4,5)$

		yz		y	
x		00	01	11	10
0				1	1
1		1	1		

z

$\Sigma(2,3,4,5) = x'y + xy'$

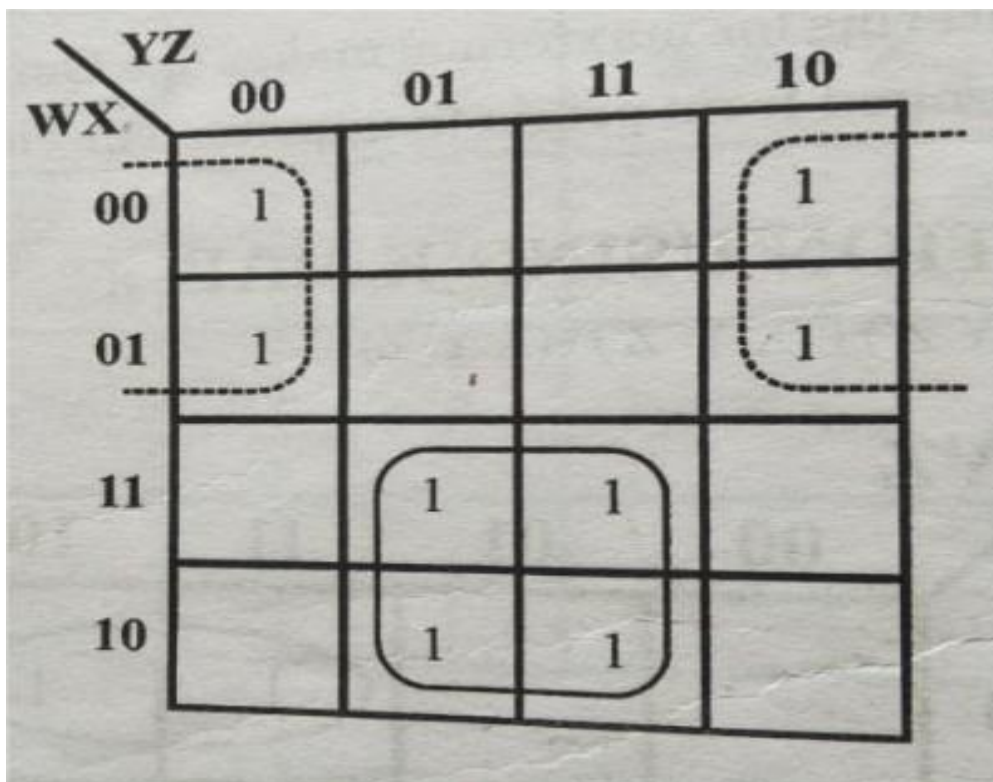
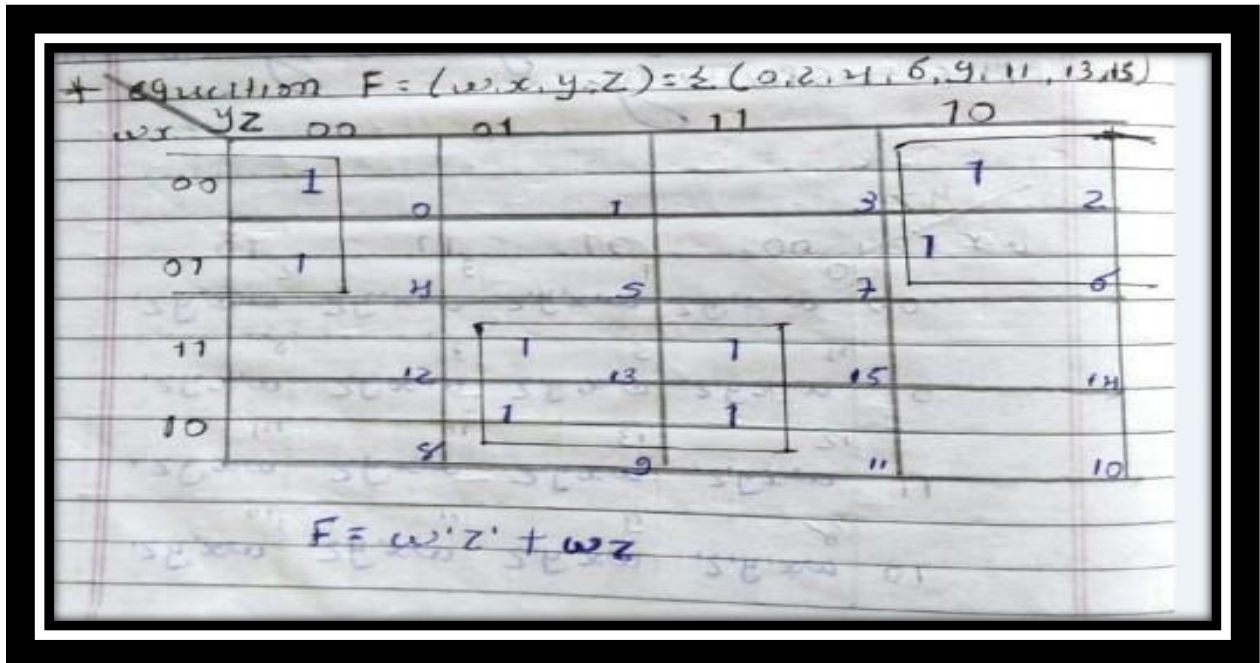
Example 6: $F(x,y,z) = \Sigma(3,4,6,7)$

		yz		y	
x		00	01	11	10
0				1	
1		1		1	1

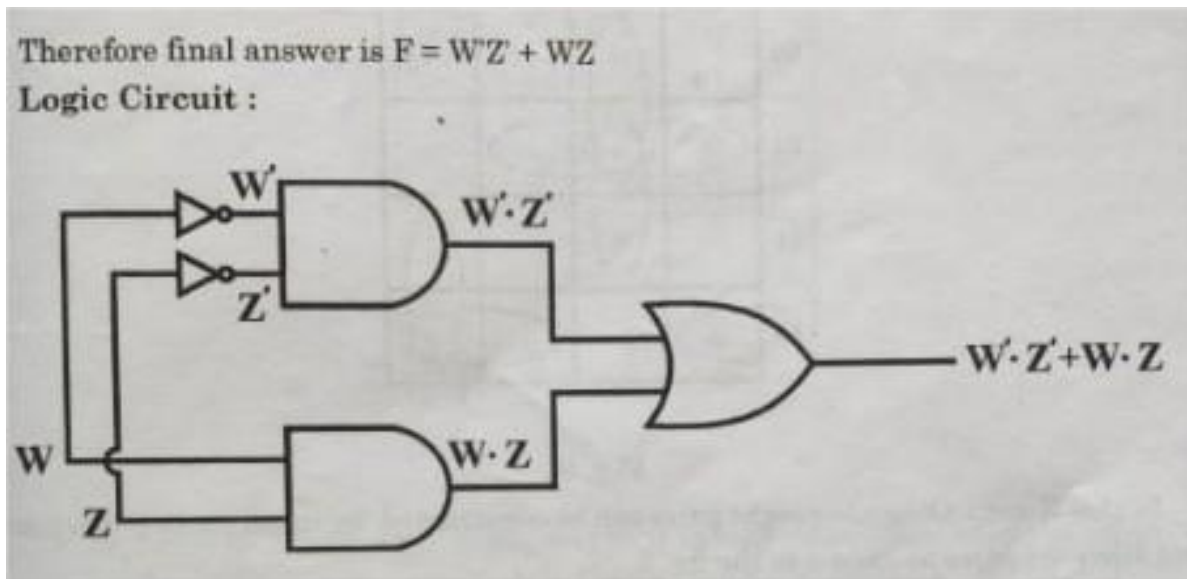
z

$\Sigma(3,4,6,7) = yz + xz'$

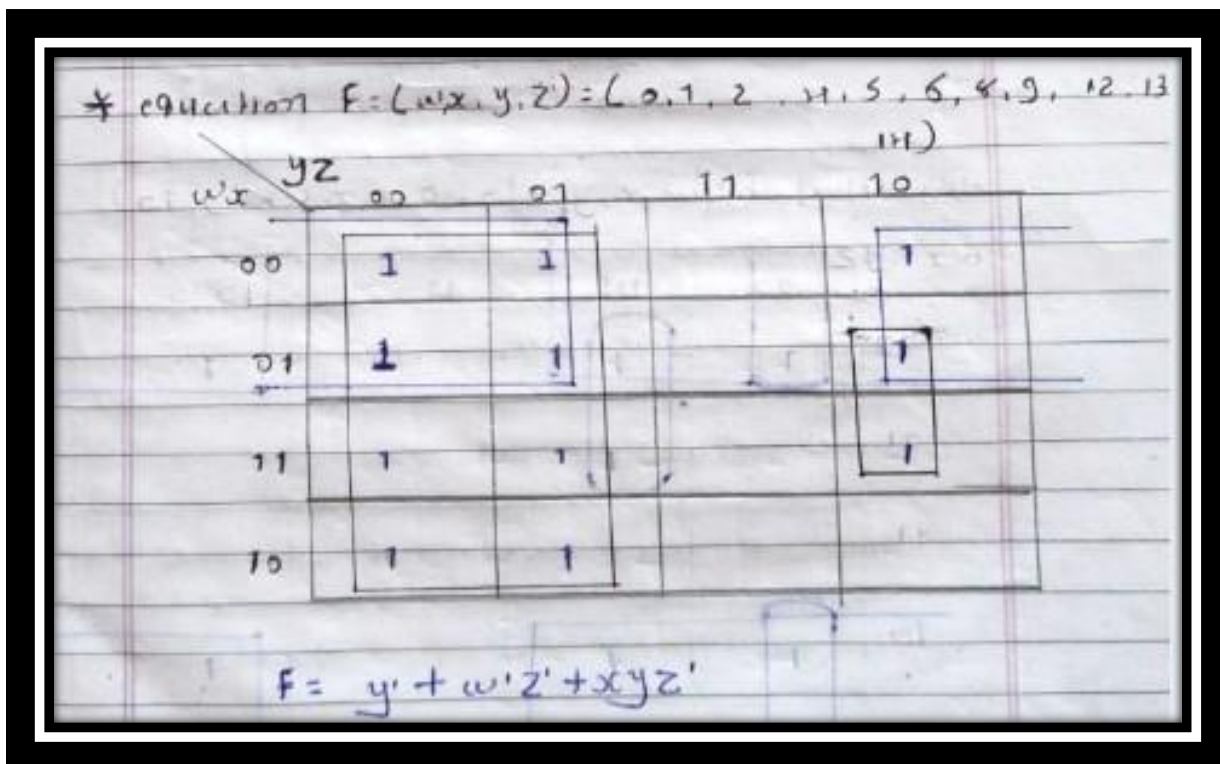
Example 7: Simplified the equation $F(W,X,Y,Z) = \sum (0,2,4,6,9,11,13,15)$ using k-map and draw the logic circuit for the result that obtained from the k-map.



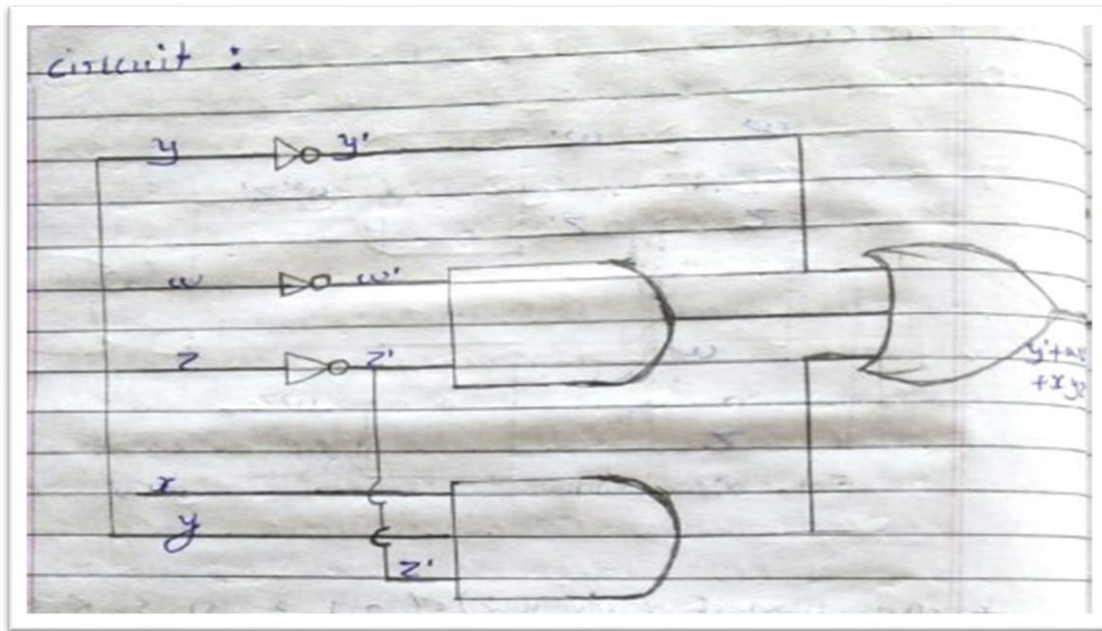
Circuit → $F = W'Z' + WZ$



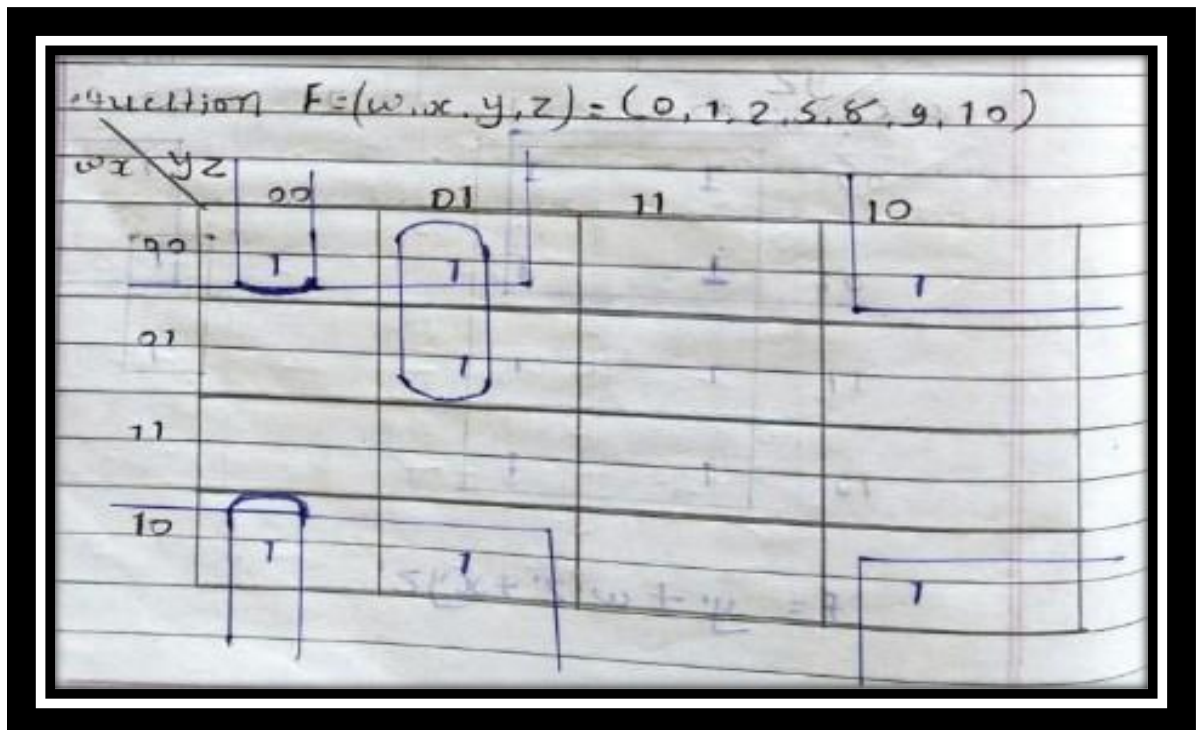
Example 8: Simplified the equation $F(W,X,Y,Z) = \sum (0,1,2,4,5,6,8,9,12,13,14)$ using k-map and draw the logic circuit for the result that obtained from the k-map.



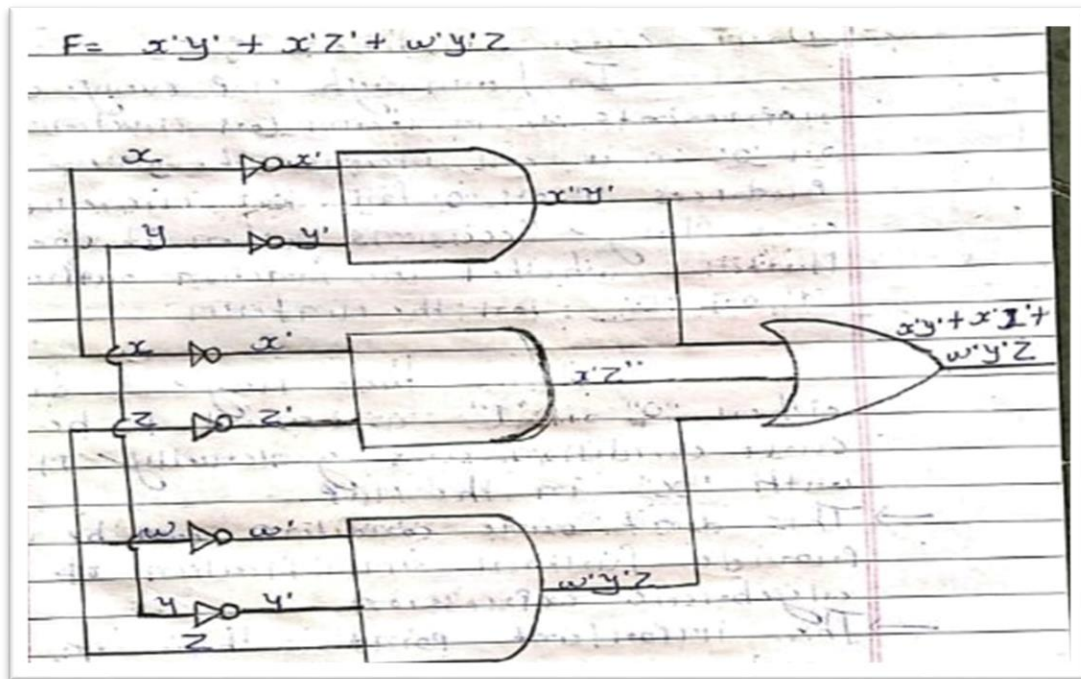
Circuit → $F = Y' + W'Z' + XYZ'$



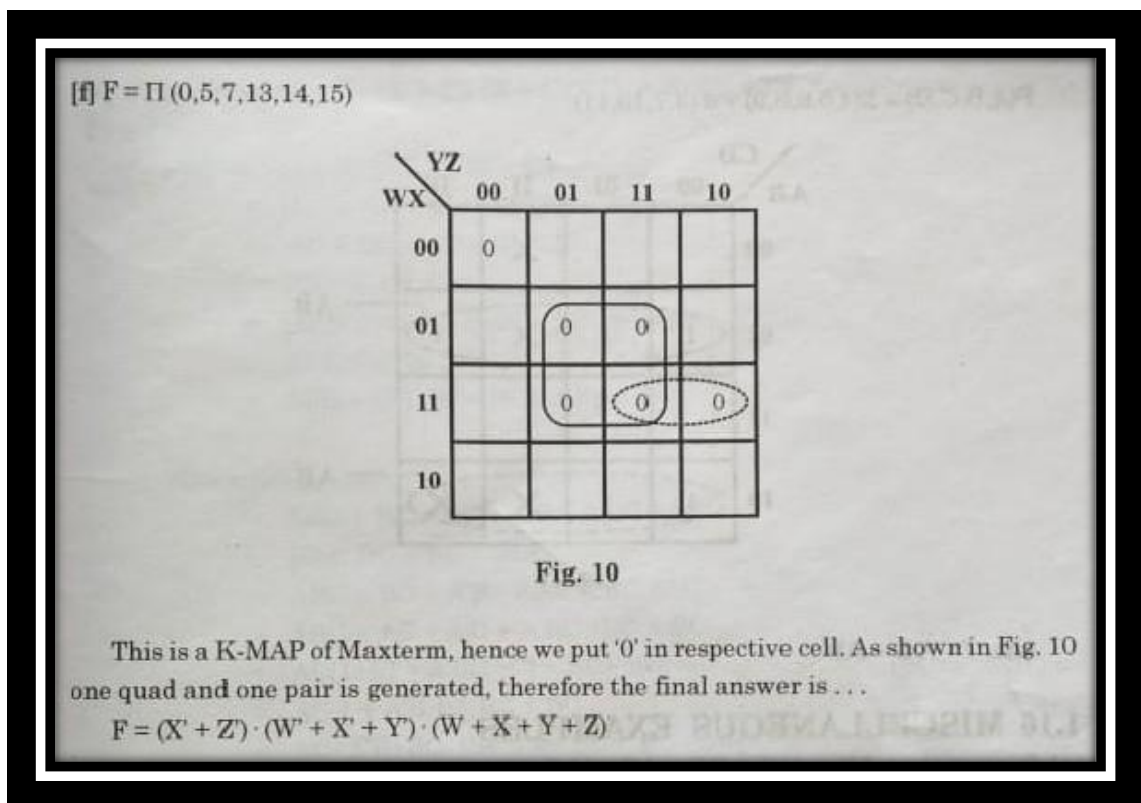
Example 9: Simplified the equation $F(W,X,Y,Z) = \sum (0,1,2,4,5,8,9,10)$ using k-map and draw the logic circuit for the result that obtained from the k-map.



Circuit → $F = X'Y' + X'Z' + W'Y'Z$



Example 10: $F = \Pi(0,5,7,13,14,15)$



Don't care Condition:

- In k-map every cell represents a minterm (or maxterm).
- The "Don't care" condition says that we can use the blank cells of a K-map to make a group of the variables.
- To make a group of cells, we can use the "don't care" cells as either 0 or 1, and if required, we can also ignore that cell.
- We mainly use the "don't care" cell to make a large group of cells.
- The cross(\times) symbol is used to represent the "don't care" cell in K-map.

Don't Care Conditions

- Simplify the Boolean function
 $F(w, x, y, z) = \sum(1, 3, 7, 11, 15)$ which has the don't-care conditions: $d(w, x, y, z) = (0, 2, 5)$

YZ \ WX	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

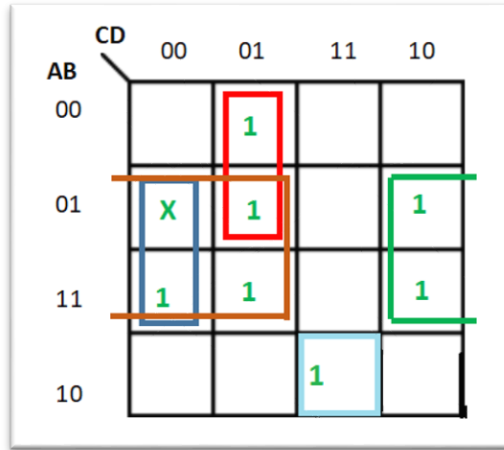
A red box highlights the top row (YZ=00) with cells (WX=00, YZ=00) containing 'X', (WX=01, YZ=00) containing '1', (WX=11, YZ=00) containing '1', and (WX=10, YZ=00) containing 'X'. A red arrow points from the label 'W'X'' to the top row. A green box highlights the third column (WX=11) with cells (WX=11, YZ=00) containing '1', (WX=11, YZ=01) containing '1', (WX=11, YZ=11) containing '1', and (WX=11, YZ=10) containing '1'. A green arrow points from the label 'YZ' to the third column.

$$F = W'X' + YZ$$

Example-1:

Minimise the following function in SOP minimal form using K-Maps:

$$f = m(1, 5, 6, 11, 12, 13, 14) + d(4)$$



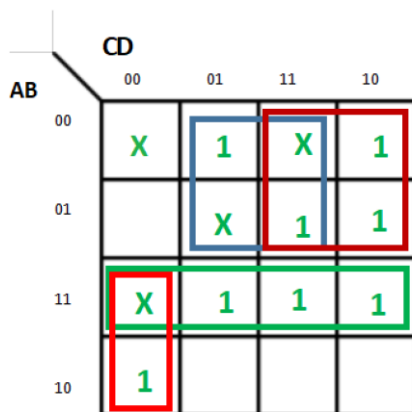
Therefore, SOP minimal is,

$$f = BC' + BCD' + A'C'D + AB'CD$$

Example-2:

Minimise the following function in SOP minimal form using K-Maps:

$$F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(0, 3, 5, 12)$$



Therefore, SOP minimal is,

$$f = AC'D' + A'D + A'C + AB$$

Example-3:

Minimise the following function in POS minimal form using K-Maps:

$$F(A, B, C, D) = m(0, 1, 2, 3, 4, 5) + d(10, 11, 12, 13, 14, 15)$$

Explanation:

Writing the given expression in POS form:

$$F(A, B, C, D) = M(6, 7, 8, 9) + d(12, 13, 14, 15)$$

		CD			
		00	01	11	10
AB	00				
	01			0	0
	11	X	X	X	X
	10	0	0		

Therefore, POS minimal is, $F = (A' + C)(B' + C')$

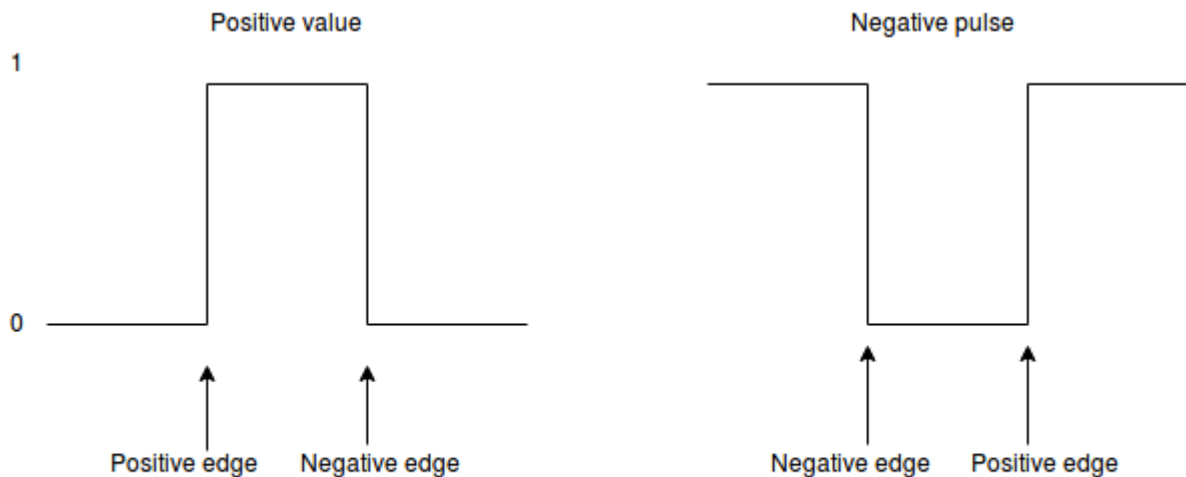


Sequential and Combinational circuits

Clock pulse

- ❖ A pulse start from the initial value of '0', goes momentarily to '1', and after a short while, returns to its initial '0' value.
- ❖ A clock pulse is either positive or negative.
- ❖ A positive clock source remains at '0' during the interval between pulses and goes to 1 during the occurrence of a pulse.
- ❖ The pulse goes through two signal transition: from '0' to '1' and return from '1' to '0'.

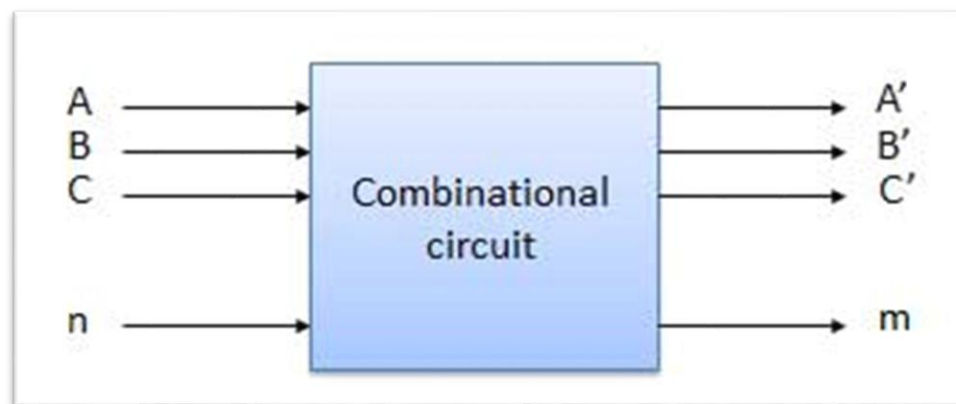
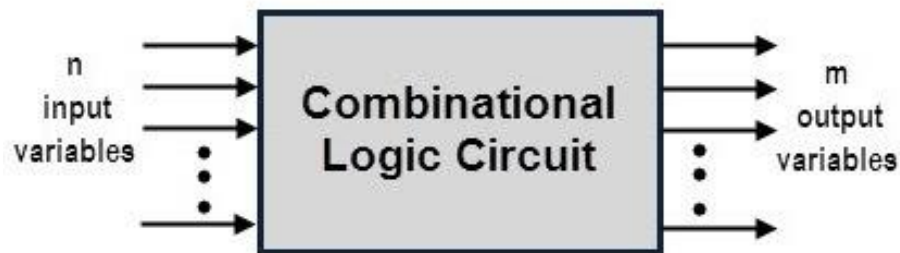
❖ Definition of clock pulse transition:



- ❖ The positive transition is defined as a positive edge and the negative transition as a negative edge.

Combinational circuits:

- A combinational circuit is a connected arrangement of logic gates with a set of input and outputs.
- The 'n' binary input variable come from an external source; the 'm' binary output variable go to an external destination, and in between there is an interconnection of logic gates.
- A combinational circuit transforms binary information from the given input data to the required output data.
- Combinational circuit are employed in digital computer for generating binary control decision and for providing digital components required for data processing.



- **Ex.** Multiplexer, De-multiplexer, Encoder, Decoder, Half-adder, Full-adder, etc. that are used for arithmetic operation and Boolean operation.

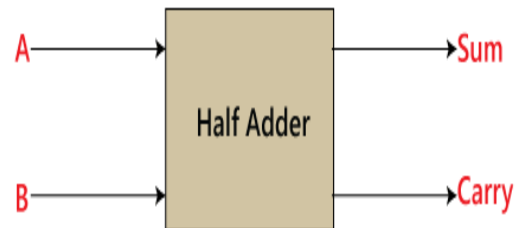
Arithmetic Circuits

Digital computers perform a variety of arithmetic operations, like addition, subtraction in this section we mainly discuss the Adder and subtraction circuit.

[1]Half - Adder

A Half-adder circuit needs two binary inputs and two binary outputs. The input variable shows the augend and addend bits whereas the output variable produces the sum and carry. We can understand the function of a half-adder by formulating a truth table. The truth table for a half-adder is:

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

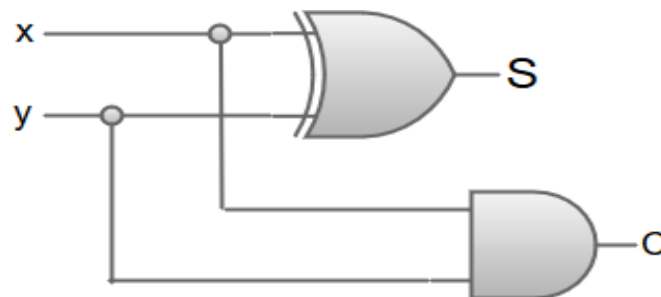


- 'x' and 'y' are the two inputs, and S (Sum) and C (Carry) are the two outputs.
- The Carry output is '0' unless both the inputs are 1.
- 'S' represents the least significant bit of the sum.

The simplified sum of products (SOP) expressions is:

$$S = x'y + xy', C = xy$$

The logic diagram for a half-adder circuit can be represented as:



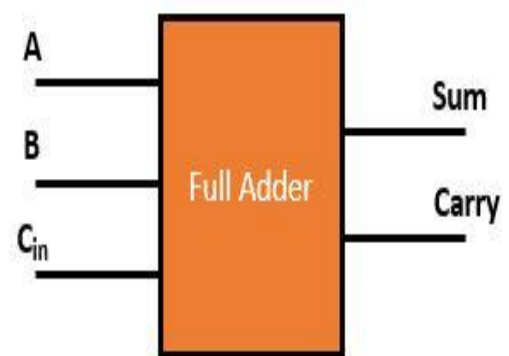
$$S = x'y + xy'$$

$$C = xy$$

[2]Full - Adder

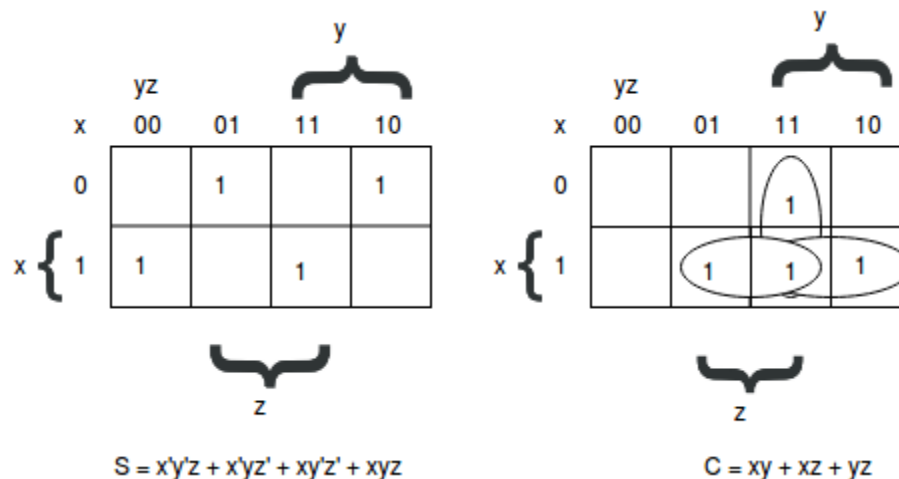
This circuit needs three binary inputs and two binary outputs. The truth table for a full-adder is:

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

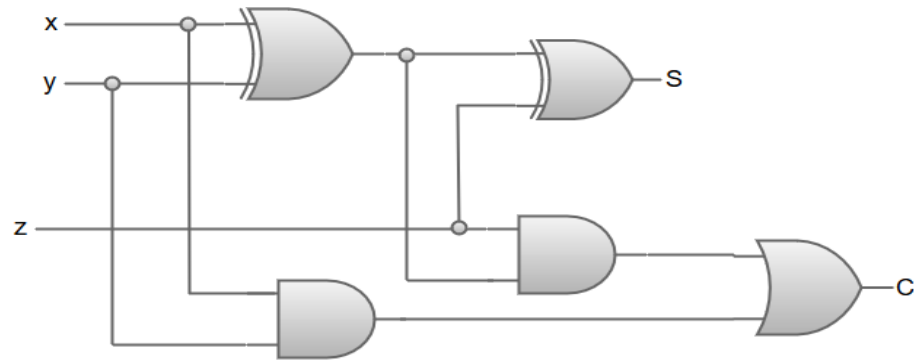


- Two of the input variable 'x' and 'y', represent the two significant bits to be added.
- The third input variable 'z', represents the carry from the previous lower significant position.
- The outputs are designated by the symbol 'S' for sum and 'C' for carry.
- The eight rows under the input variables designate all possible combinations of 0's, and 1's that these variables may have.
- The input-output logical relationship of the full-adder circuit may be expressed in two Boolean functions, one for each output variable.
- Each output Boolean function can be simplified by using a unique map method.

Maps for a full-adder:



The logic diagram for a full-adder circuit can be represented as:



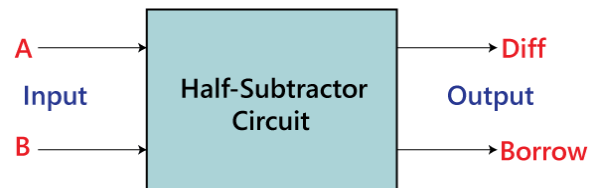
$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$

[3] Half Subtractor

The half subtractor is also a building block for subtracting two binary numbers. It has two inputs and two outputs. This circuit is used to subtract two single bit binary numbers A and B. The '**diff**' and '**borrow**' are two output states of the half subtractor.

Block diagram



Truth Table

Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The SOP form of the **Diff** and **Borrow** is as follows:

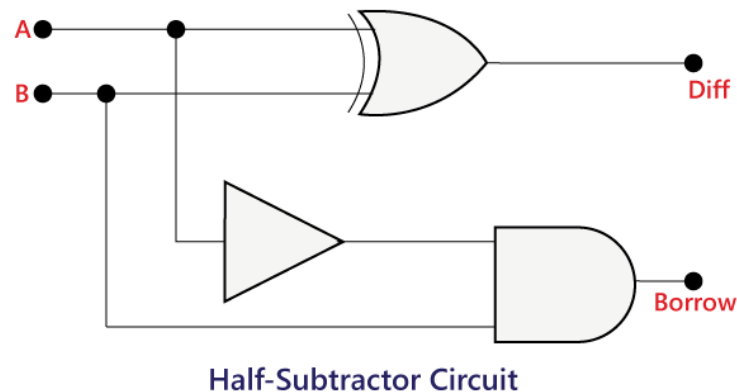
$$\text{Diff} = A'B + AB'$$

$$\text{Borrow} = A'B$$

In the above table,

Half-Subtractor logical circuit

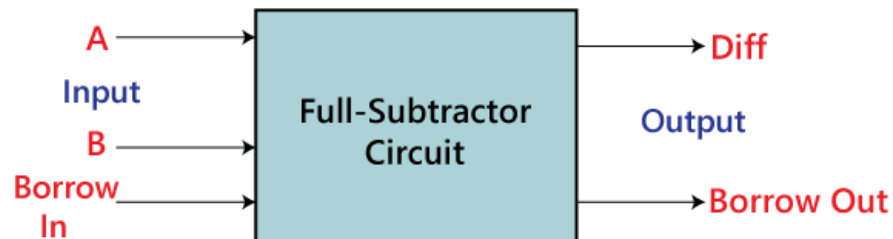
So, the Half Subtractor is designed by combining the 'XOR', 'AND', and 'NOT' gates and provide the Diff and Borrow.



[4] Full Subtractor

The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.

Block diagram



Truth Table

Inputs			Outputs	
A	B	Borrow _{in}	Diff	Borrow
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

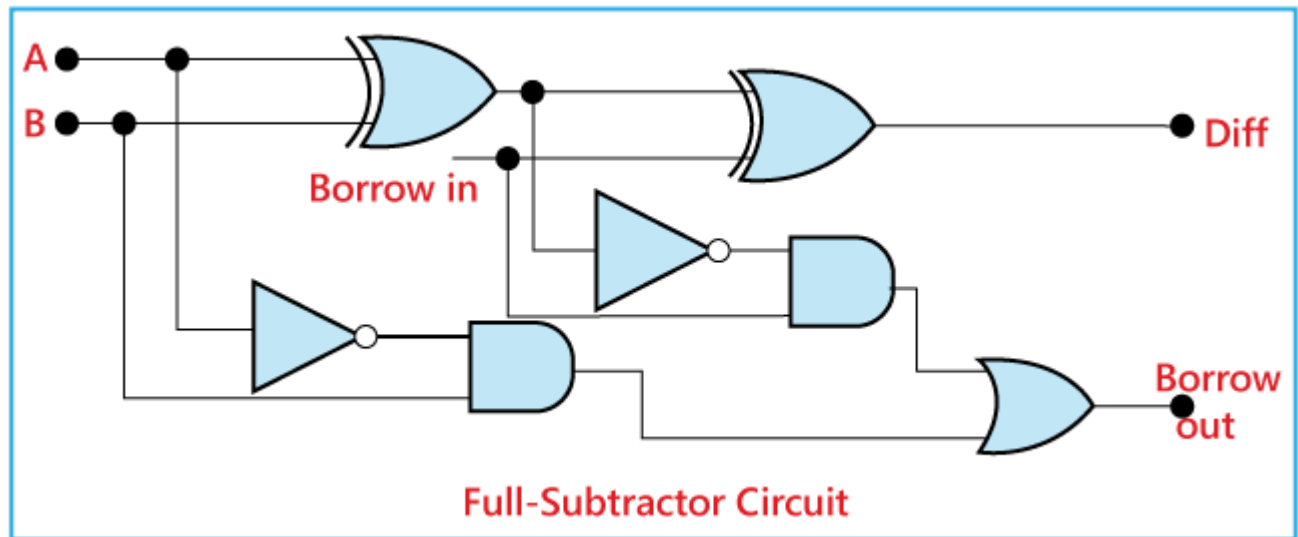
The SOP form can be obtained with the help of K-map as:

x \ yz	00	01	11	10
	0	1	0	1
1	1	0	1	0

$$\text{Diff} = xy'z' + x'y'z + xyz + x'yz'$$

x \ yz	00	01	11	10
	0	1	1	1
1	1	0	1	0

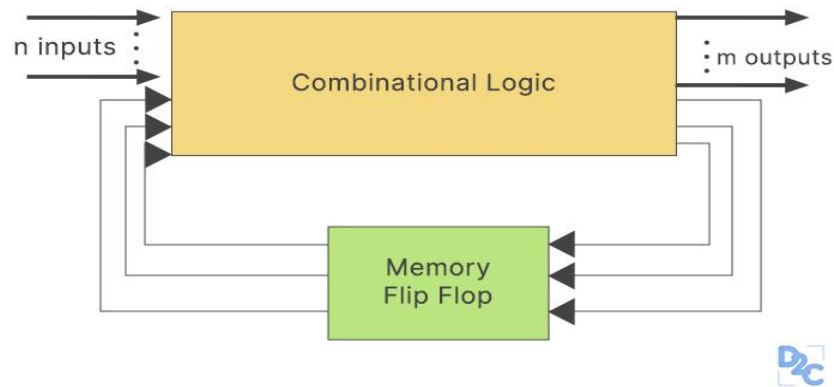
$$\text{Borrow} = x'z + x'y + yz$$



Sequential circuits:

- Sequential circuits are those which are dependent on clock cycles and depend on present as well as past inputs to generate any output.

Block diagram



- Flip-flops, Registers, Counters, etc. that are used to store data.


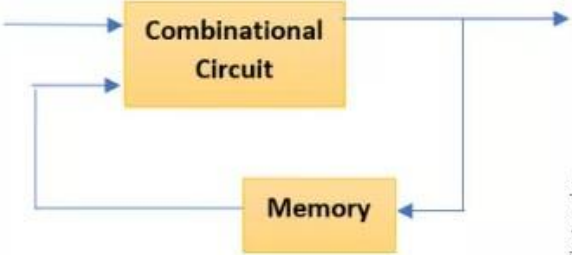
Combinational Circuit	Sequential Circuit
Output only depends on the present input	Output depends on present input and past output
Memory element is absent	Memory element is present
No clock signal is applied	Clock signal is required
	
Example - Half Adder, Full Adder, Multiplexer	Examples - Flipflop, Counters, Registers

Image Credits: Electronics

❖ Flip-Flop

Flip-flop is a circuit that maintains a state until directed by input to change the state. A basic flip-flop can be constructed using four-NAND or four-NOR gates.

Types of flip-flops:

- 1) SR Flip Flop
- 2) Clocked SR Flip Flop.
- 3) D Flip Flop
- 4) JK Flip Flop
- 5) JK –Master Slave Flip Flop.
- 6) T Flip Flop

Logic diagrams and truth tables of the different types of flip-flops are as follows:

[1] S-R Flip-flop/Basic Flip-Flop

Flip flops are an application of logic gates. A flip-flop circuit can remain in a binary state indefinitely (as long as power is delivered to the circuit) until directed by an input signal to switch states.

S-R flip-flop stands for SET-RESET flip-flops.

The SET-RESET flip-flop consists of two NOR gates and also two NAND gates.

These flip-flops are also called S-R Latch.

The design of these flip flops also includes two inputs, called the SET [S] and RESET [R]. There are also two outputs, Q and Q'.

Logic Diagram and Truth Table

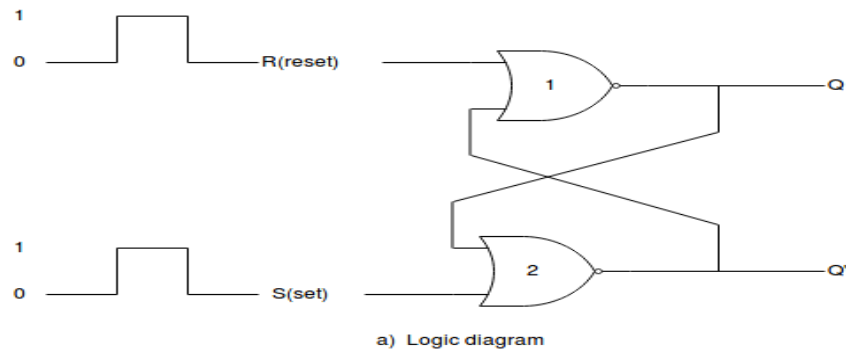


fig: Basic flip-flop circuit with NOR gates

S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

b) Truth table

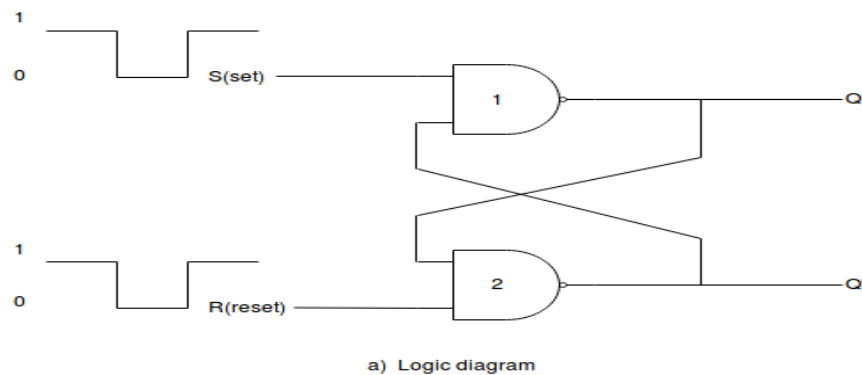


fig: Basic flip-flop circuit with NAND gates

S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

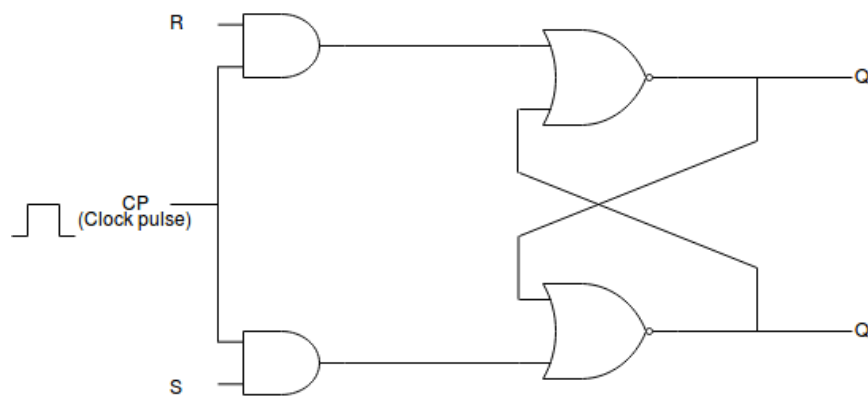
b) Truth table

[2] Clocked S-R Flip-Flop

The operation of a basic flip-flop can be modified by providing an additional control input that determines when the state of the circuit is to be changed.

The limitation with a S-R flip-flop using NOR and NAND gate is the invalid state. This problem can be overcome by using a stable SR flip-flop that can change outputs when certain invalid states are met, regardless of the condition of either the Set or the Reset inputs.

Logic Diagram and Truth Table



a) Logic diagram

fig: Clocked SR flip flop

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Intermediate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Intermediate

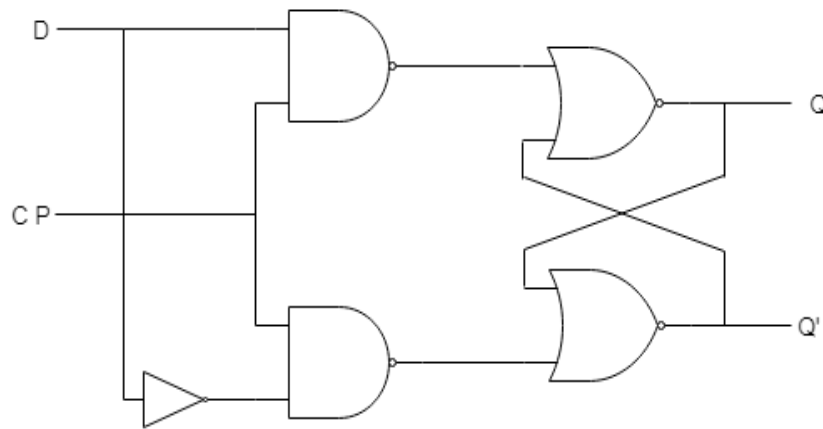
b) Truth table

A clock pulse is given to the inputs of the AND Gate. If the value of the clock pulse is '0', the outputs of both the AND Gates remain '0'.

[3] D Flip-Flop

D flip-flop is a slight modification of clocked SR flip-flop.

Logic Diagram and Truth Table



(a) Logic diagram with Nand gates

Q	D	Q (t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(c) Transition table

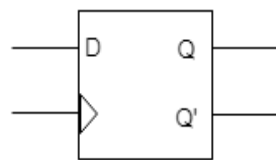


fig. Clocked D flip flop

From the above figure, you can see that the D input is connected to the S input and the complement of the D input is connected to the R input.

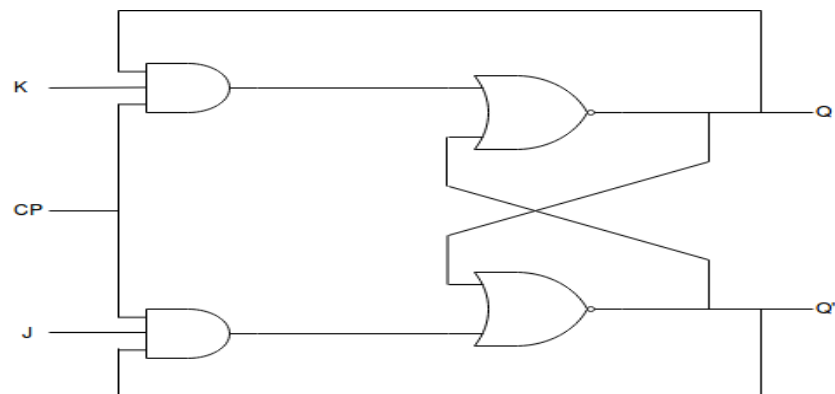
When the value of CP is '1' (HIGH), the flip-flop moves to the SET state if it is '0' (LOW), the flip-flop switches to the CLEAR state.

[4] J-K Flip-Flop

J-K flip-flop can be considered as a modification of the S-R flip-flop.

The main difference is that the intermediate state is more refined and precise than that of an S-R flip-flop.

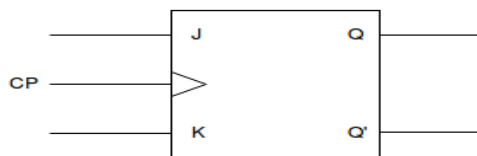
Logic Diagram and Truth Table



a) Logic diagram

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

b) Transition table



c) Graphical symbol

fig. Clocked JK flip flop

The characteristics of inputs 'J' and 'K' is same as the 'S' and 'R' inputs of the S-R flip-flop.

J stands for SET, and 'K' stands for CLEAR.

When both the inputs J and K have a HIGH state, the flip-flop switches to the complement state, so, for a value of $Q = 1$, it switches to $Q=0$, and for a value of $Q = 0$, it switches to $Q=1$.

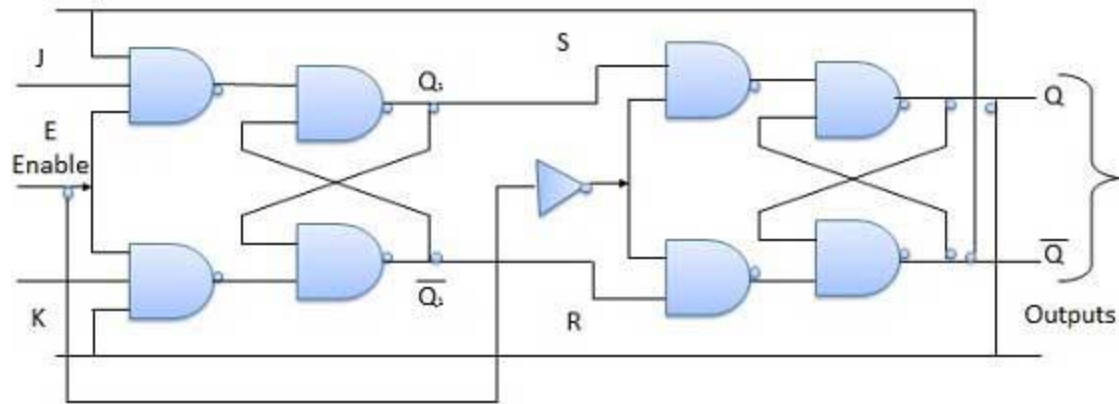
[5] Master Slave J-K Flip-Flop

Master slave JK FF is a cascade of two S-R FF with feedback from the output of second to input of first.

Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level.

Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive.

Logic Diagram



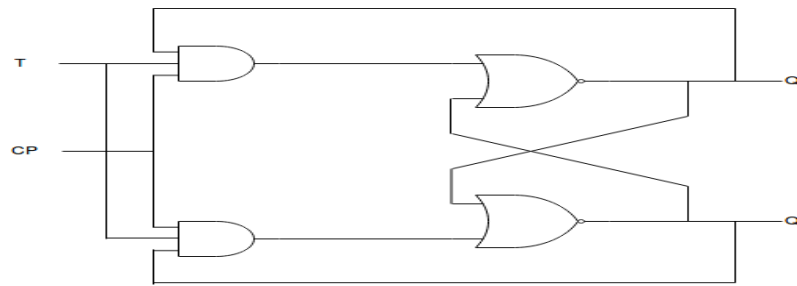
Truth Table

Inputs			Outputs		Comments
E	J	K	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	Q_n	\overline{Q}_n	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	\overline{Q}_n	Q_n	Toggle

[6] T Flip-Flop

T flip-flop is a much simpler version of the J-K flip-flop.

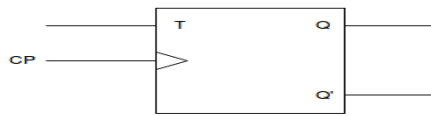
Logic Diagram and Truth Table



a) Logic diagram

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

b) Transition table



c) Graphical symbol

fig. Clocked T flip flop

Both the J and K inputs are connected and are also called as a single input J-K Flip-flop.

Triggering of Flip-Flops

The state of the flip-flop is changed by a momentary change in the input signal. This momentary change is known as Trigger, and the transition it causes is said to triggering the flip-flop.

Pulses trigger clocked flip-flops.

A pulse start from the initial value of '0', goes momentarily to '1', and after a short while, returns to its initial '0' value.

A clock pulse is either positive or negative.

A positive clock source remains at '0' during the interval between pulses and goes to 1 during the occurrence of a pulse.

The pulse goes through two signal transition: from '0' to '1' and return from '1' to '0'.