DIGITAL ELECTRONICS

Laws of Boolean Algebra

- Boolean Algebra is the mathematics we use to analyse digital gates and circuits.

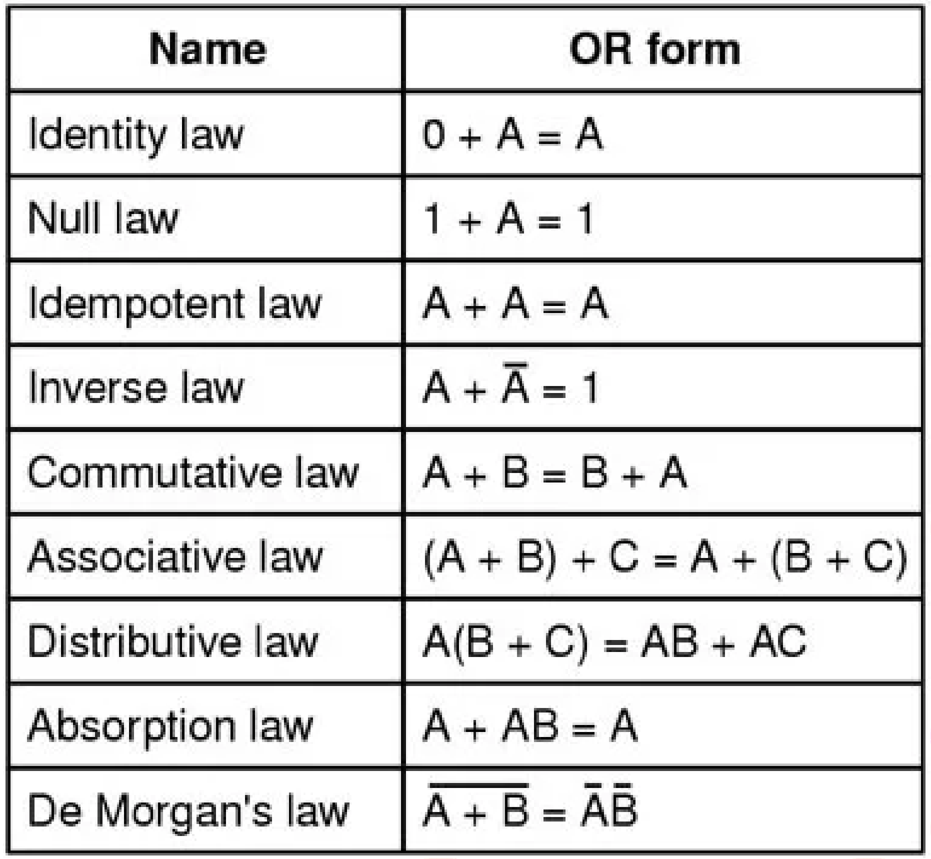
- We can use these “Laws of Boolean” to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required.

- Boolean Algebra is therefore a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions.

Laws of Boolean Algebra

Name	AND form
Identity law	$1A = A$
Null law	$0A = 0$
Idempotent law	$AA = A$
Inverse law	$A\overline{A} = 0$
Commutative law	$AB = BA$
Associative law	$(AB)C = A(BC)$
Distributive law	$A + BC = (A + B)(A + C)$
Absorption law	$A(A + B) = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$

Laws of Boolean Algebra



Name	OR form
Identity law	$0 + A = A$
Null law	$1 + A = 1$
Idempotent law	$A + A = A$
Inverse law	$A + \bar{A} = 1$
Commutative law	$A + B = B + A$
Associative law	$(A + B) + C = A + (B + C)$
Distributive law	$A(B + C) = AB + AC$
Absorption law	$A + AB = A$
De Morgan's law	$\overline{A + B} = \bar{A}\bar{B}$

Q. Find the simplified expression A’BC’+AC’.

a) B

b) A+C

c) (A+B)C’

d) B’C

TEACHERS ADDA BY TARGET ABHI 7877719287

Q. (X+Y`)(X+Z) can be represented by _____

a) (X+Y`Z)

b) (Y+X`)

c) XY`

d) (X+Z`)

TEACHERS ADDA BY TARGET ABHI 7877719287

Q. Evaluate the expression: $(X + Z)(X + XZ') + XY + Y$.

a) $XY+Z'$

b) $Y+XZ'+Y'Z$

c) $X'Z+Y$

d) $X+Y$

DIGITAL LOGIC

- In digital logic, the inputs and output of a function are in the form of binary numbers (boolean values) i.e., the values are either zero (0) or one (1).
- Therefore, digital logic is also known as ‘Boolean logic’. These inputs and output can be termed as ‘Boolean Variables’.
- The output boolean variable of a digital signal can be expressed in terms of input boolean variables which forms the ‘Boolean Expression’.

DIGITAL LOGIC

➢ **Representation of Boolean expression can be primarily done in two ways. They are as follows:**

1. **Sum of Products (SOP) form**
2. **Product of Sums (POS) form**

SUM OF PRODUCTS (SOP)

- It is one of the ways of writing a boolean expression.
- As the name suggests, it is formed by adding (OR operation) the product terms.
- These product terms are also called as ‘min-terms’. Min-terms are represented with ‘m’, they are the product(AND operation) of boolean variables either in normal form or complemented form.

Sum of Products (SOP)

Product of Sums (POS)

- As the name suggests, it is formed by multiplying(AND operation) the sum terms.
- These sum terms are also called as ‘max-terms’.
- Max-terms are represented with ‘M’, they are the sum (OR operation) of Boolean variables either in normal form or complemented form.

Teachers Adda by Target Abhi 7877719287

Product of Sums (POS)

KARNAUGH MAP(K-MAP) METHOD

- The K-map is a systematic way of simplifying Boolean expressions. With the help of the K-map method, we can find the simplest POS and SOP expression, which is known as the minimum expression.

- Just like the truth table, a K-map contains all the possible values of input variables and their corresponding output values.

- However, in K-map, the values are stored in cells of the array. In each cell, a binary value of each input variable is stored.

KARNAUGH MAP(K-MAP) METHOD

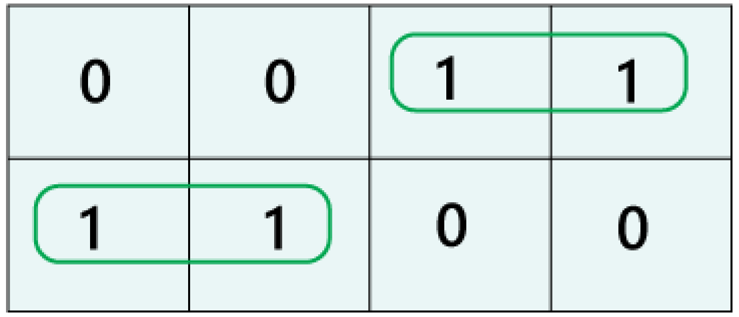
- The K-map method is used for expressions containing 2, 3, 4, 5 & 6. For a higher number of variables, there is another method used for simplification called the Quine-McClusky method.

- In K-map, the number of cells is similar to the total number of variable input combinations. For example, if the number of variables is three, the number of cells is $2^3=8$

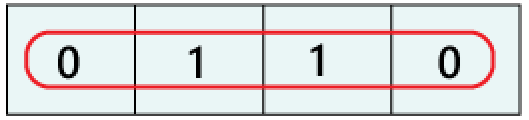
Karnaugh Map(K-Map) method

- The K-map takes the SOP and POS forms.
- The K-map grid is filled using 0's and 1's.
- The K-map is solved by making groups.

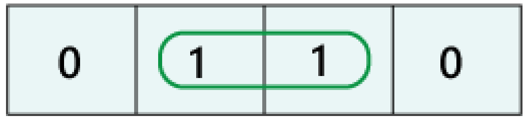
Teachers Adda by Target Abhi 7877719287



Notice that each group should have the largest number of 'ones'. A group cannot contain an empty cell or cell that contains 0.



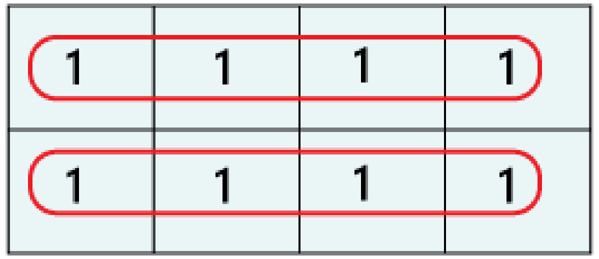
Incorrect



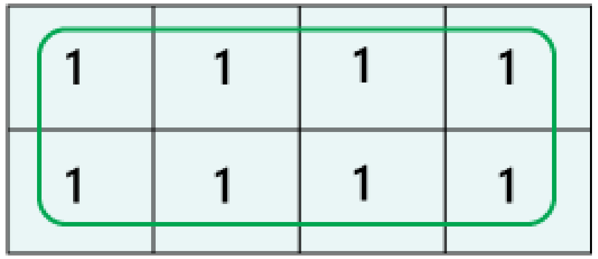
Correct

TEACHERS ADDA BY TARGET ABHI 7877719287

We group the number of ones in the decreasing order. First, we have to try to make the group of eight, then for four, after that two and lastly for 1.



Incorrect

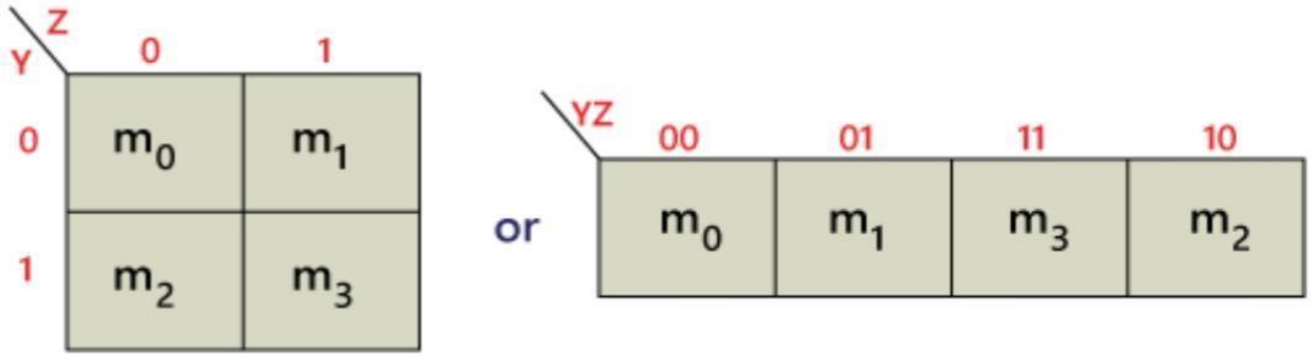


Correct

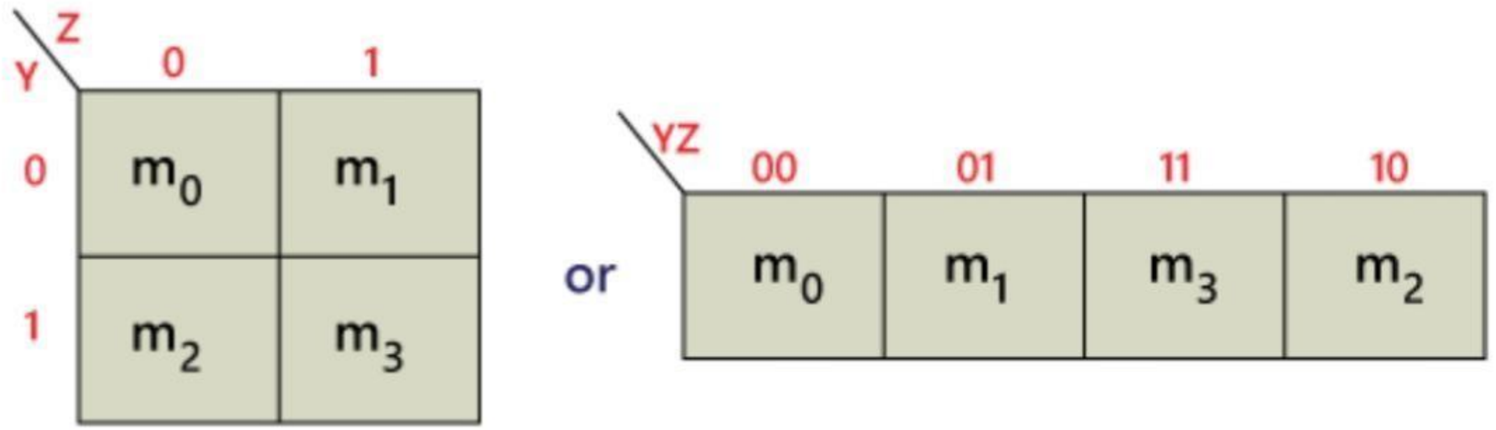
There are the following steps used to solve the expressions using K-map:

1. First, we find the K-map as per the number of variables.
2. Find the maxterm and minterm in the given expression.
3. Fill cells of K-map for SOP with 1 respective to the minterms.
4. Fill cells of the block for POS with 0 respective to the maxterm.
5. Next, we create rectangular groups that contain total terms in the power of two like 2, 4, 8, ... and try to cover as many elements as we can in one group.
6. With the help of these groups, we find the product terms and sum them up for the SOP form.

2 Variable K-map



There is a total of 4 cells in a 2-variable K-map. There are two variables in the 2-variable K-map. The following figure shows the structure of the 2-variable K-map:



The possible combinations of grouping 2 adjacent minterms are {(m_0, m_1), (m_2, m_3), (m_0, m_2) and (m_1, m_3)}.

TEACHERS ADDA BY TARGET ABHI 7877719287

3-VARIABLE K-MAP

- The 3-variable K-map is represented as an array of eight cells. In this case, we used A, B, and C for the variable.
- We can use any letter for the names of the variables. The binary values of variables A and B are along the left side, and the values of C are across the top.
- The value of the given cell is the binary values of A and B at left side in the same row combined with the value of C at the top in the same column.

AB \ C	0	1
00		
01		
11		
10		

AB \ C	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

TEACHERS ADDA BY TARGET ABHI 7877719287

The 4-Variable Karnaugh Map

- The 4-variable K-map is represented as an array of 16 cells.
- Binary values of A and B are along the left side, and the values of C and D are across the top.
- The value of the given cell is the binary values of A and B at left side in the same row combined with the binary values of C and D at the top in the same column.

TEACHERS ADDA BY TARGET ABHI 7877719287

AB \ CD	00	01	11	10
00				
01				
11				
10				

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

TEACHERS ADDA BY TARGET ABHI 7877719287

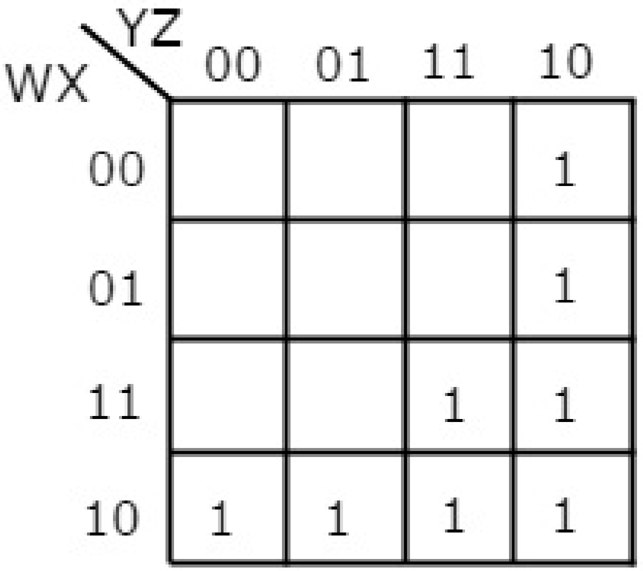
Simplification of boolean expressions using Karnaugh Map

As we know that K-map takes both SOP and POS forms. So, there are two possible solutions for K-map, i.e., minterm and maxterm solution.

Minterm Solution of K Map

There are the following steps to find the minterm solution or K-map:
Step 1: Firstly, we define the given expression in its canonical form.
Step 2: Next, we create the K-map by entering 1 to each product-term into the K-map cell and fill the remaining cells with zeros.
Step 3: Next, we form the groups by considering each one in the K-map.

Simplification of boolean expressions using Karnaugh Map



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

Teachers Adda by Target Abhi 7877719287

Let us **simplify** the following Boolean function, using K-map.

$$f(X,Y,Z)=\prod M(0,1,2,4)$$

TEACHERS ADDA BY TARGET ABHI 7877719287

Grouping of Cells for Simplification

The grouping of cells for simplification is done in the following ways:

1. Pairs: if two adjacent cells either in row or in column contain 1's, they are said to be in pairs

Grouping of Cells for Simplification

A \ B	0	1
0	00	01
1	10	11

Two-variable K-map

A \ B	0	1
0	0	1
1	2	3

Cell designation

A \ B	B´	B
A´	A´B´	A´B
A	AB´	AB

Cells with minterm

A \ B	B	B´
A	A + B	A + B´
A´	A´ + B	A´ + B´

Cells with maxterm

DON’T-CARE CONDITIONS

- Since the function may be either 0 or 1, we say that we don’t care what the function output is to be for this minterm.

- Minterms that may produce either 0 or 1 for the function are said to be don’t-care conditions and are marked with an in the map. These don’t-care conditions can be used to provide further simplification of the algebraic expression.

X

K-MAPS(WITH DON'T CARE CONDITION) QUESTIONS

$$F(A,B,C)=\Sigma\ m(0,1,6,7) + \Sigma\ d(3,4,5)$$

Combinational & Sequential Logic Circuits

INTRODUCTION

- In digital **circuit** theory, **sequential logic** is a type of **logic circuit** whose output depends not only on the present value of its input signals but on the sequence of past inputs, the input history as well. This is in contrast to **combinational logic**, whose output is a function of only the present input.

- **Difference between combinational and sequential circuit.**

- **Sequential circuits** are those which are dependent on clock cycles and depends on present as well as past inputs to generate any output.
- **Combinational Circuit** –
- In this output depends only upon present input.
- The sequential logic has memory while combinational logic does not.
- They employ a feedback loop to give output back to input. **Sequential** logic **circuits** is a form of binary **circuit**; its design employs one or more inputs and one or more outputs.

TEACHERS ADDA BY TARGET ABHI 7877719287

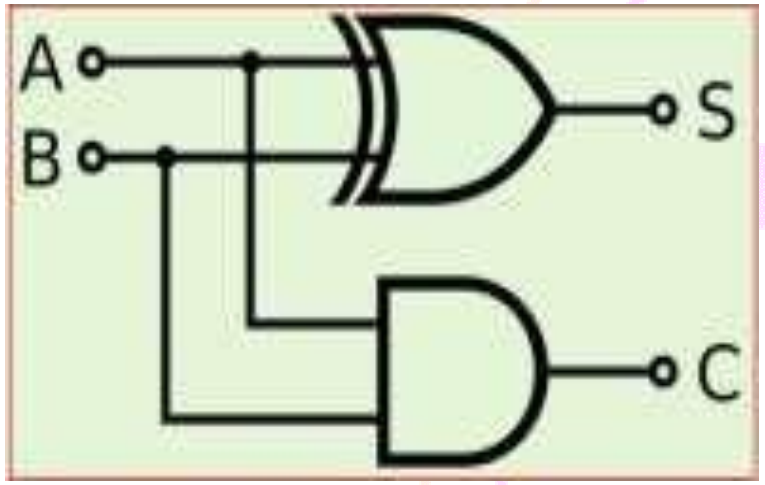
- A **combinational logic circuit** performs an operation assigned logically by a **Boolean** expression or truth table. Examples of common **combinational logic circuits** include: half adders, full adders, **multiplexers**, **demultiplexers**, encoders and decoders **.**

- A **Sequential** logic **circuits** is a form of binary **circuit**; its design employs one or more inputs and one or more outputs, whose states are related to some definite rules that depends on previous states. ... **Examples** of such **circuits** include clocks, flip-flops, bi-stables, counters, memories, and registers.
- There are two **types of sequential circuit**, synchronous and asynchronous. Synchronous **types** use pulsed or level inputs and a clock input to drive the **circuit**(with restrictions on pulse width and **circuit** propagation). Asynchronous **sequential circuits** do not use a clock signal as synchronous **circuits** do.
- **Flip flop** is a **sequential circuit** which generally samples its inputs and changes its outputs only at particular instants of time and not continuously. **Flip flop** is said to be edge sensitive or edge triggered rather than being level triggered like latches.

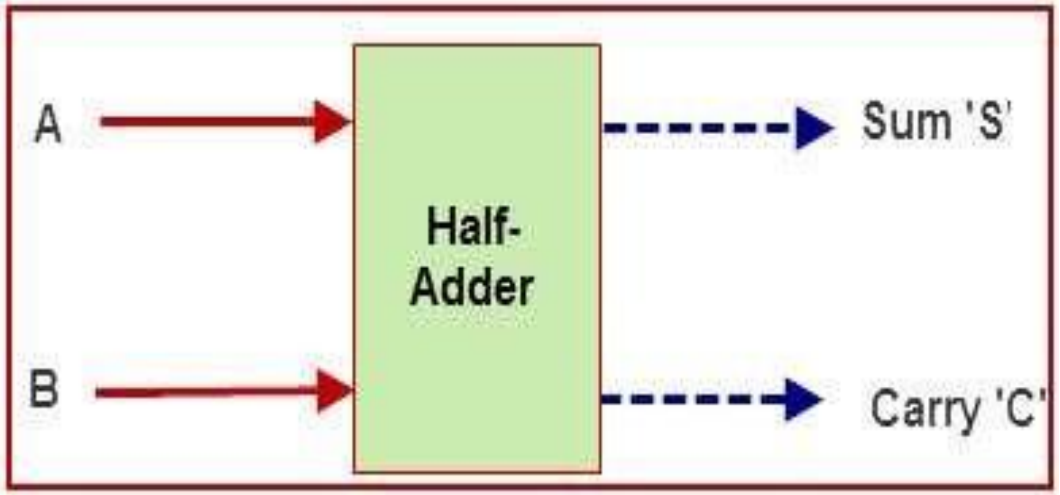
Half adder

- An **adder** is a digital circuit that performs addition of numbers. Half adder has only two inputs and two outputs. The **half adder** adds **two binary digits** called as augend and addend and produces two outputs as **sum and carry**; XOR is applied to both inputs to produce sum and AND gate is applied to both inputs to produce carry.
- By using half adder, you can design simple addition with the help of logic gates.

Half adder

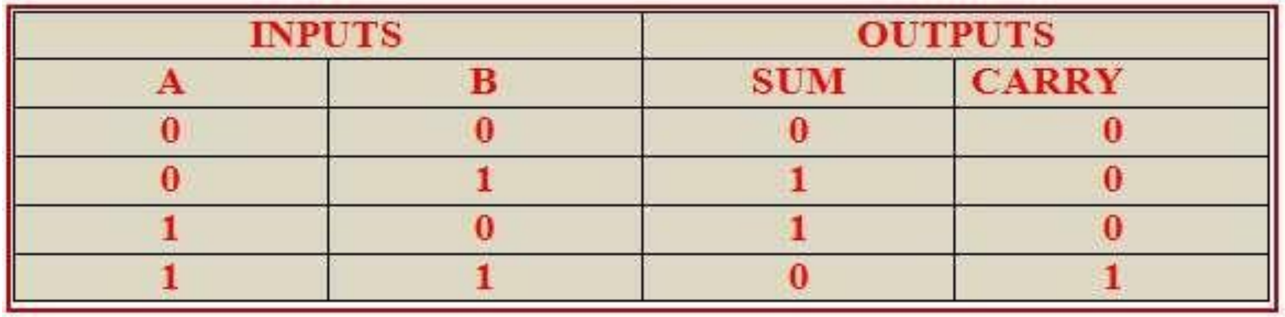


Circuit Implementation



Block Diagram

Truth Table



INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TEACHERS ADDA BY TARGET ABHI 7877719287

Why it is called as Half-adder ?

- The **half adder** can add only two input bits (A and B) and has nothing to do with the carry if there is any in the input. So if the input to a **half adder** have a carry, then it will neglect it and adds only the A and B bits. That means the binary addition process is not complete and that's why it is **called a half adder**

How Half Adder works ?

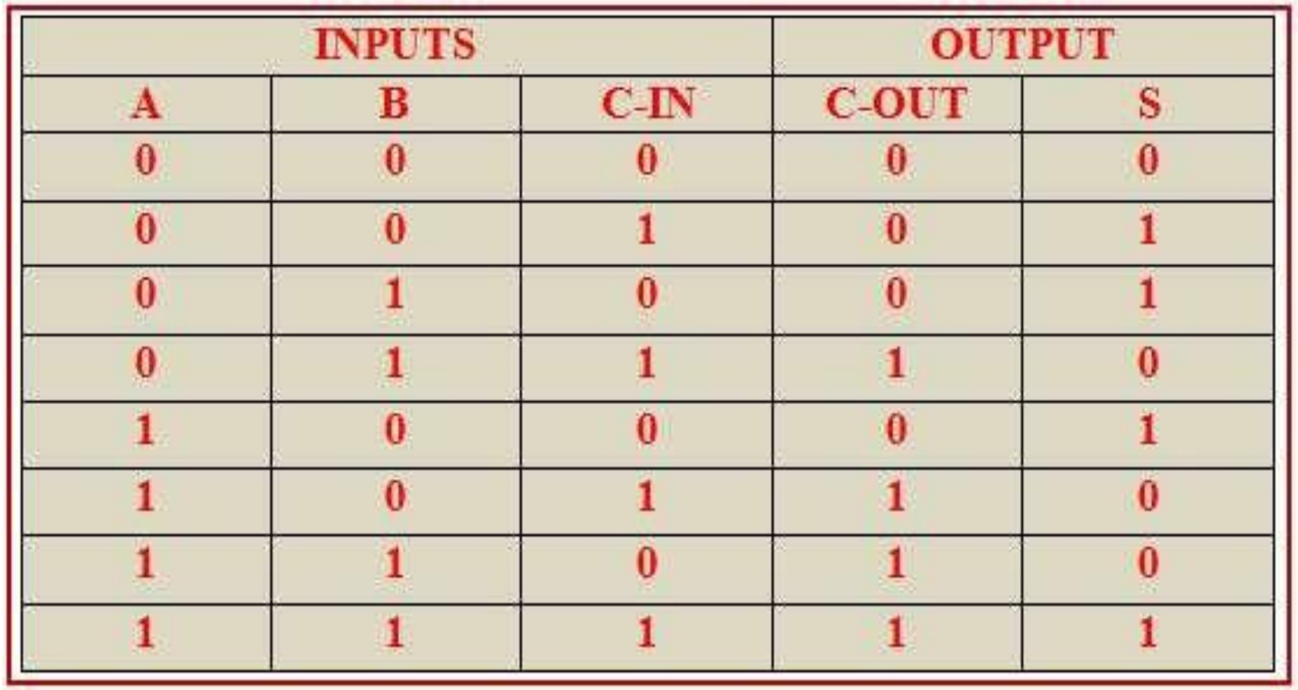
- **Half adder** is a simple combinational circuit used to add two single bits. It accepts two inputs and produce two outputs that is a sum output and a carry output. A **half adder** consists of two logic gates 1) XOR and 2) AND gate. And the carry operation performed by AND gate thus carry out put **will** be A+B.

FULL ADDER

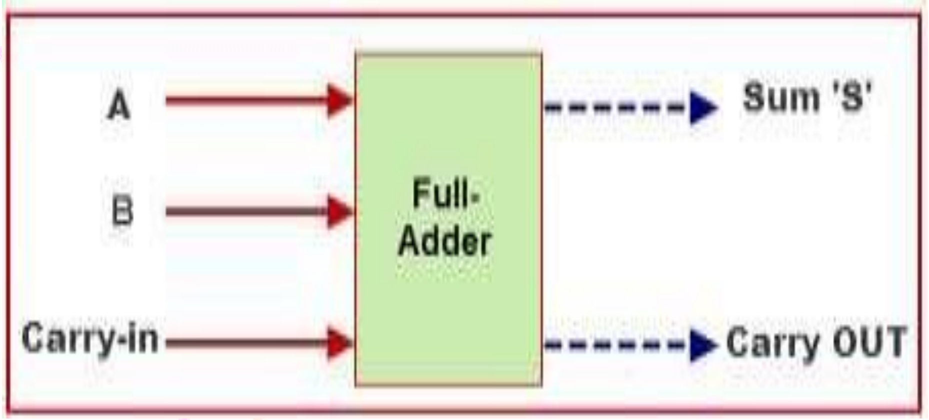
The full adder adds 3 one bit numbers, where two can be referred to as **operands and one can be referred to as bit carried in**. It produces 2-bit output and these can be referred to as output carry and sum.

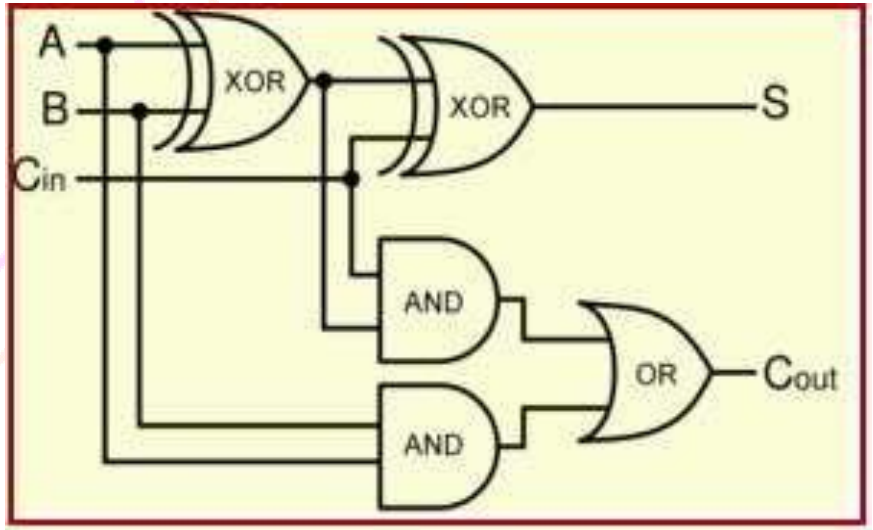
The full-adder has three inputs and two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. When a full-adder logic is designed, you string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.

FULL ADDER



INPUTS			OUTPUT	
A	B	C-IN	C-OUT	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





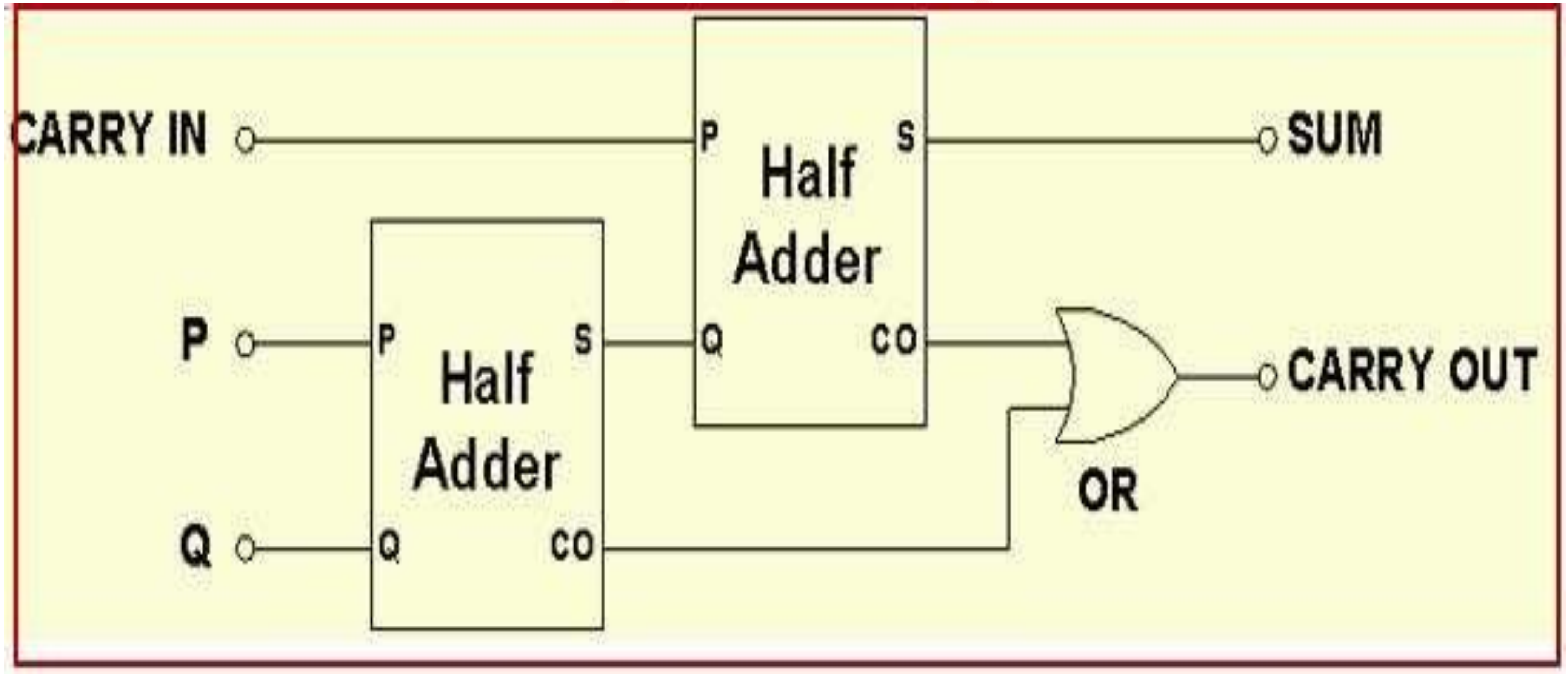
Circuit Implementation

•With the truth-table, the full adder logic can be implemented. You can see that the output S is an XOR between the input A and the half-adder, SUM output with B and C-IN inputs. We take C-OUT will only be true if any of the two inputs out of the three are HIGH.

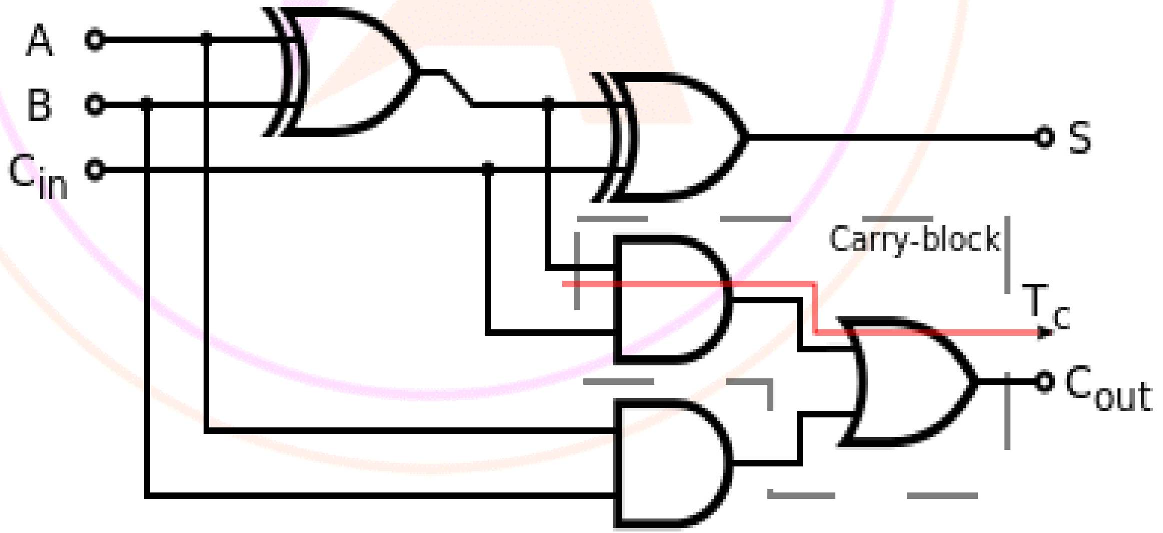
•So, we can implement a full adder circuit with the help of two half adder circuits. At first, half adder will be used to add A and B to produce a partial Sum and a second half adder logic can be used to add C-IN to the Sum produced by the first half adder to get the final S output.

• If any of the half adder logic produces a carry, there will be an output carry. So, C-OUT will be an OR function of the half-adder Carry outputs.

•The implementation of larger logic diagrams is possible with the above full adder logic a simpler symbol is mostly used to represent the operation. Given below is a simpler schematic representation of a full adder.

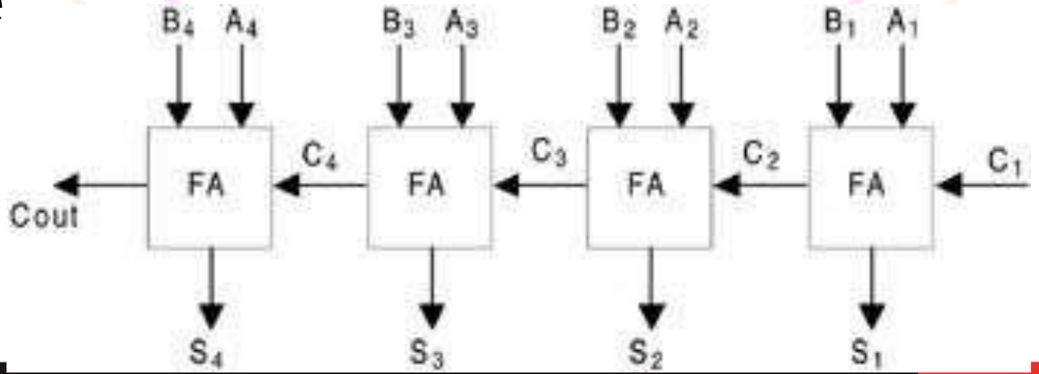

CARRY IN
P
S
Half Adder
SUM
P
P
Half Adder
S
Q
CO
CARRY OUT
OR
Q
Q
CO

- A **full adder** is a digital **circuit** that performs addition. **Full adders** are implemented with logic gates in hardware.
  A **full adder** adds three one-bit binary numbers, two operands and a carry bit. The **adder** outputs two numbers, a sum and a carry bit.
- The **Boolean expression for a full adder** is as follows.
- For the CARRY-OUT (Cout) bit: CARRY-OUT = A AND B OR Cin(A XOR B) = A.B + Cin(A ⊕ B)



4-Bit Parallel Binary adder :

- A **binary parallel adder** is a digital function that produces the arithmetic sum of two **binary** numbers in **parallel**.
- **FOUR-BIT** BINARY **PARALLEL ADDER** is a circuit in which two binary numbers each of n **bits** can be added by means of a full **adder** circuit. Consider the example that two **4-bit** binary numbers $B_4B_3B_2B_1$ and $A_4A_3A_2A_1$ are to be added with a carry input C_1.
- A group of four bits is called a nibble. A basic 4-bit parallel adder is implemented with four full-adder stages as shown in Figure



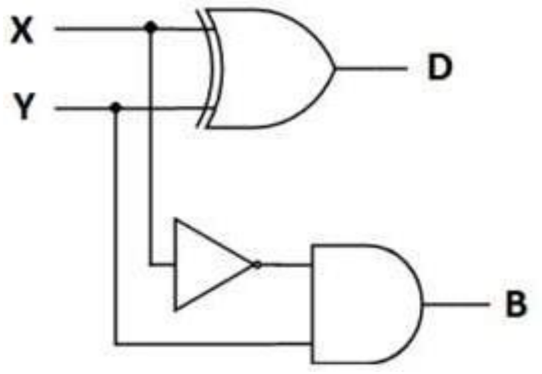
Half Substractor :

- A half subtractor is an arithmetic circuit that subtracts two bits and produces their difference. The circuit has two inputs minuend (X) and subtrahend (Y) and two output bits, one is the difference bit (D) and the other is the borrow bit (B).
- As like addition operation of 2 binary digits, which produces SUM and CARRY, the subtraction of 2 binary digits also produces two outputs which are termed as **difference and borrow**. The simplest possible subtraction of 2-bit binary digits consists of four possible operations, they are **0-0, 0-1, 1-0** and **1-1**. The operations **0-0, 1-0** and **1-1** produces a subtraction of 1-bit output whereas, the remaining operation **0-1** produces a 2-bit output. They are referred as **difference** and **borrow** bit respectively. This **borrow** bit is used for subtraction of the next higher pair bit.
- So, we can define half subtractor as a combinational circuit which is capable of performing subtraction of 2-bit binary digits is known as a half subtractor. Here, the binary digit from which the other digit is subtracted is called **minuend** and the binary digit which is to be subtracted is known as the **subtrahend**.

TEACHERS ADDA BY TARGET ABHI 7877719287

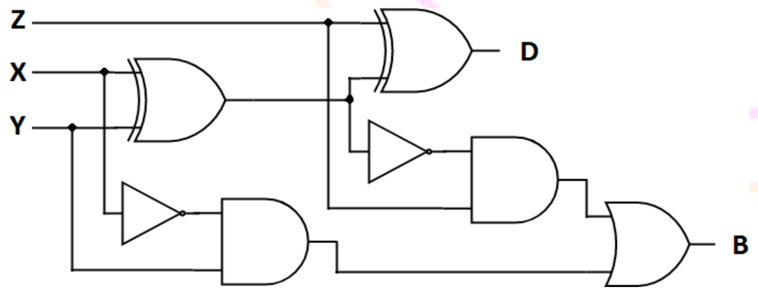
- It performs the operation X – Y. It should be noted that the weight of the output borrow bit is -2, while the weight of the output difference bit is +1.
- The truth table of the half subtractor is shown. The Boolean functions for the two outputs can be obtained directly from the truth table as:

  D = (XY + XY) = X ⊕Y
- **The half subtractor boolean expressions are :**
- **D = (X'Y + XY') = X ⊕ Y**
- **B = X'Y**



Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

A full subtractor

- A full subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant bit. The circuit has three inputs and two outputs.
- Input variables are minuend (X), subtrahend (Y), and previous borrow (Z); output variables are difference (D) and output borrow (B).
- It performs the operation **X – Y – Z**. It should be noted that the weight of the output borrow bit is -2, while the weight of the output difference bit is +1. The truth table of the full subtractor is shown.
- **The full subtractor boolean expressions are :**
- $(X'Y'Z + X'YZ' + XY'Z' + XYZ) = X \oplus Y \oplus Z$
- $(X'Y'Z + X'YZ' + X'YZ + XYZ) = X'(Y \oplus Z) + YZ$
- 

Inputs			Outputs	
X	Y	Z	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

TEACHERS ADDA BY TARGET ABHI 7877719287

- When there is a situation where the minuend and subtrahend number contains more significant bit, then the **borrow** bit which is obtained from the subtraction of 2-bit binary digits is subtracted from the next higher order pair of bits. In such situation, the subtraction involves the operation of 3 bits. Such situation of subtraction can't handle by a simple half subtractor. So, combining two half subtractor we can form another combinational circuit which can perform this type of operation. This circuit is known as the full subtractor.
- So we can define full subtractor as a combinational circuit which takes three inputs and produces two outputs **difference** and **borrow**. Above is the truth table of the full subtractor, we have used three input variables X, Y and Z which refers to the term **minuend, subtrahend** and **borrow** bit respectively. The two outputs **difference** and **borrow** are named as D and B respectively.
- The construction of **full subtractor circuit diagram** involves two half subtractor joined by an OR gate as shown in the above **circuit diagram of the full subtractor**. The two borrow bits generated by two separate half subtractor are fed to the OR gate which produces the final borrow bit. The final difference bit is the combination of the difference output of the first half adder and the next higher order pair of bits.