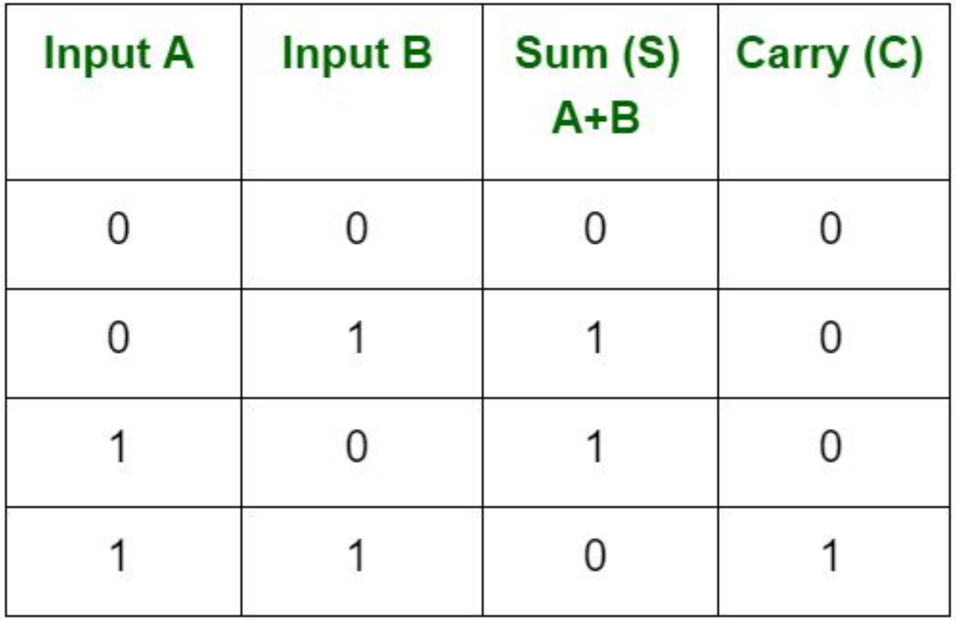
Binary Arithmetic

- Computer works only with binary number system.
- Although a computer accepts decimal input provides decimal output, but internally it works on binary.
- Operations to be done on binary data:
  1. Binary addition
  2. Binary subtraction

Binary Addition

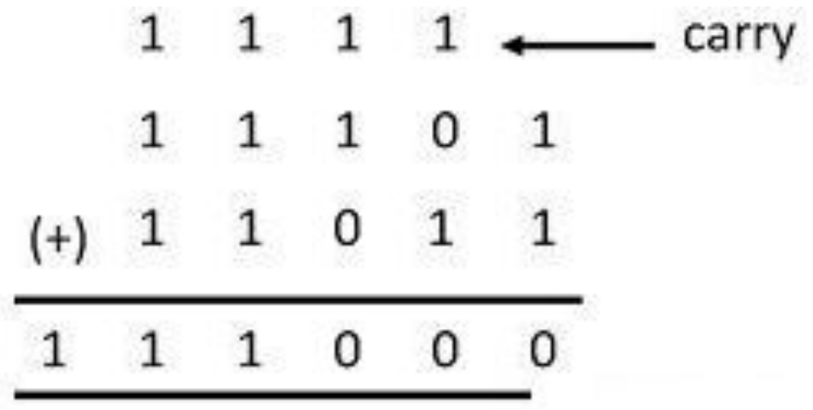
Truth table to add two binary numbers:



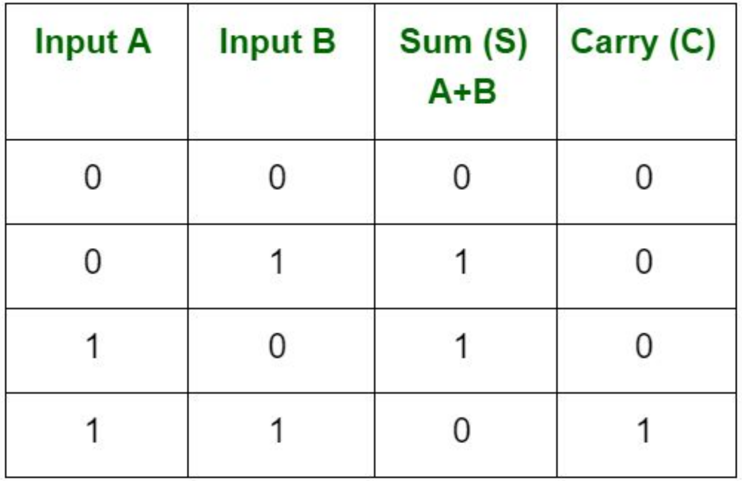
Input A	Input B	Sum (S) A+B	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary Addition - Example

Add 11101 and 11011



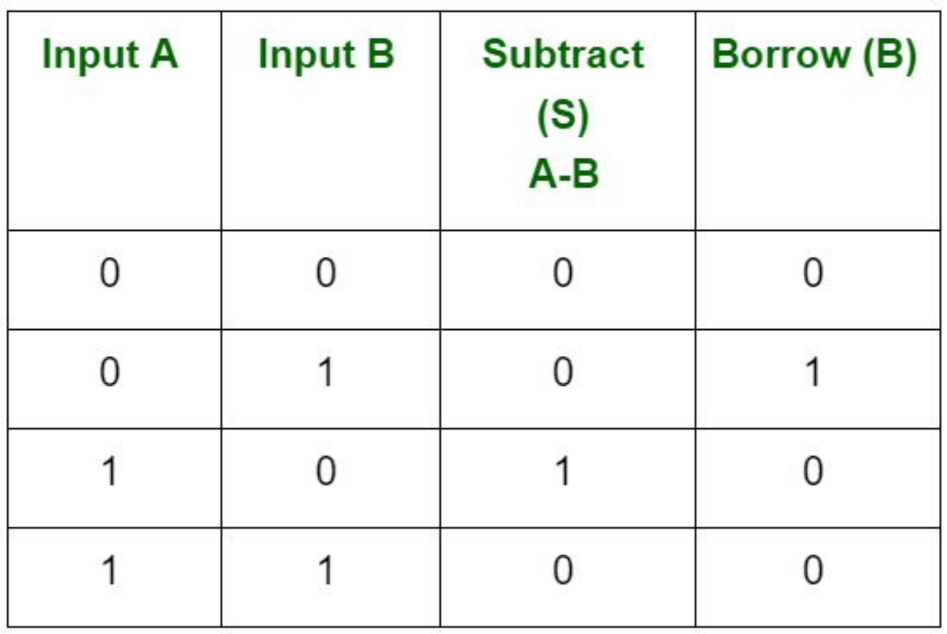
$$\begin{array}{rccccccl}
 & & 1 & 1 & 1 & 1 & & \leftarrow \text{carry} \\
 & & 1 & 1 & 1 & 0 & 1 & \\
(+) & & 1 & 1 & 0 & 1 & 1 & \\
\hline
 & 1 & 1 & 1 & 0 & 0 & 0 & \\
\hline
\end{array}$$



Input A	Input B	Sum (S) A+B	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary Subtraction

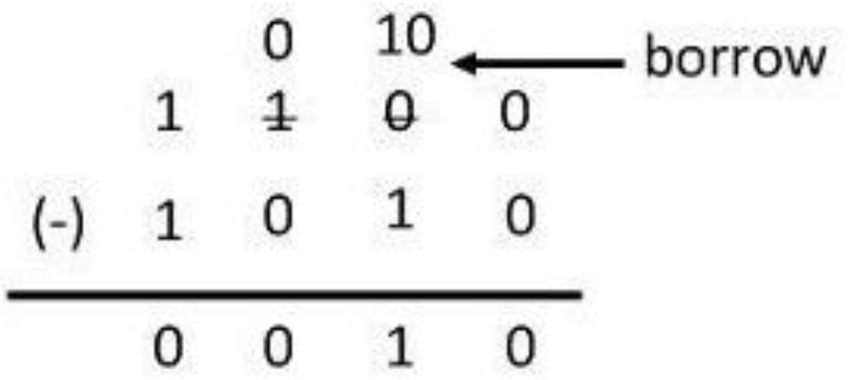
Truth table to subtract two binary numbers:

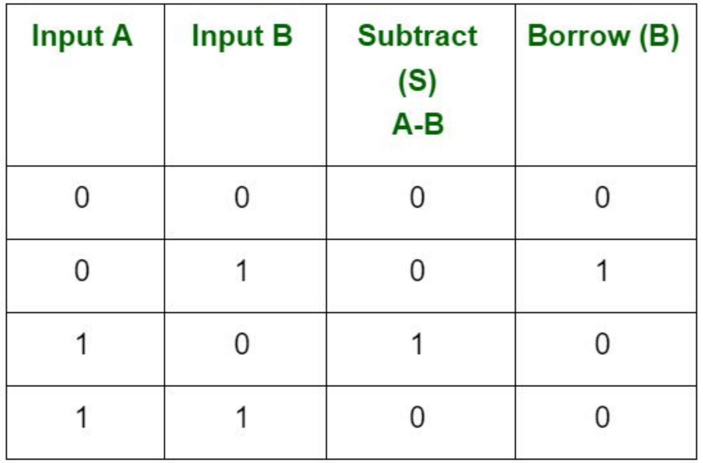


Input A	Input B	Subtract (S) A-B	Borrow (B)
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Binary Subtraction-Example

Subtract: 1100 - 1010





Input A	Input B	Subtract (S) A-B	Borrow (B)
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

1’s Complement and 2’s Complement

1s complement and 2s complement are way of representing the signed binary numbers.

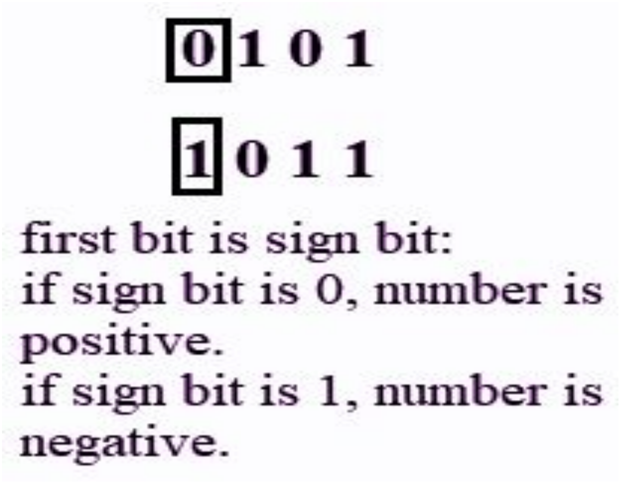
In general, the binary number can be represented in two ways.

1. **Unsigned Binary Numbers:** Using unsigned binary number representation, only positive binary numbers can be represented.
2. **Signed Binary Numbers:** Using signed binary number representation both positive and negative numbers can be represented.

1's Complement and 2's Complement

2. Signed Binary Numbers: In signed binary number representation, MSB of the number is a sign bit. For positive numbers, the sign bit is 0 and for negative number, the sign bit is 1. There are three different ways the signed binary numbers can be represented.

1. Signed Magnitude Form
2. **1's Complement Form**
3. **2's Complement Form**



$\boxed{0}\,1\,0\,1$

$\boxed{1}\,0\,1\,1$

first bit is sign bit:
if sign bit is 0, number is positive.
if sign bit is 1, number is negative.

1’s Complement Representation

1’s complement of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.

Example:

1's complement of "0111" is "1000"

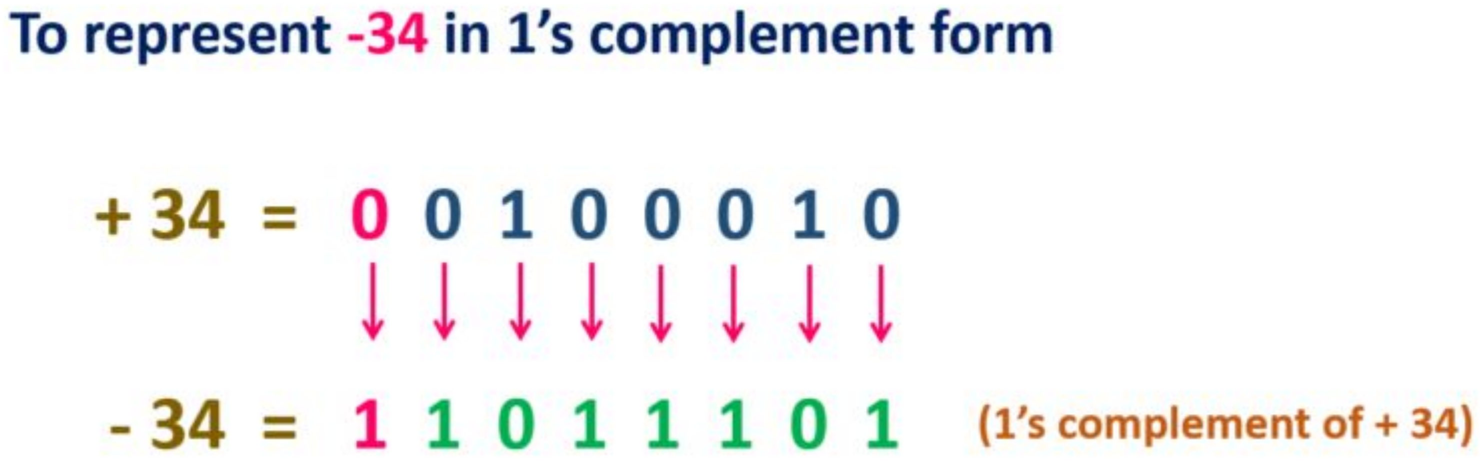
1's complement of "1100" is "0011"

1’s Complement Representation

In this, the representation of the positive number is same as the negative number. But the representation of the negative number is different.

Example: if we want to represent -34 in 8-bit 1’s complement form, then first write the positive number (+34). And invert all 1s in that number by 0s and 0s by 1s in that number. The corresponding inverted number represents the -34 in 1’s complement form. It is also called 1s complement of the number +34.

1’s Complement Representation Example



To represent -34 in 1’s complement form

+ 34 = 0 0 1 0 0 0 1 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

- 34 = 1 1 0 1 1 1 0 1 (1’s complement of + 34)

1’s Complement Representation

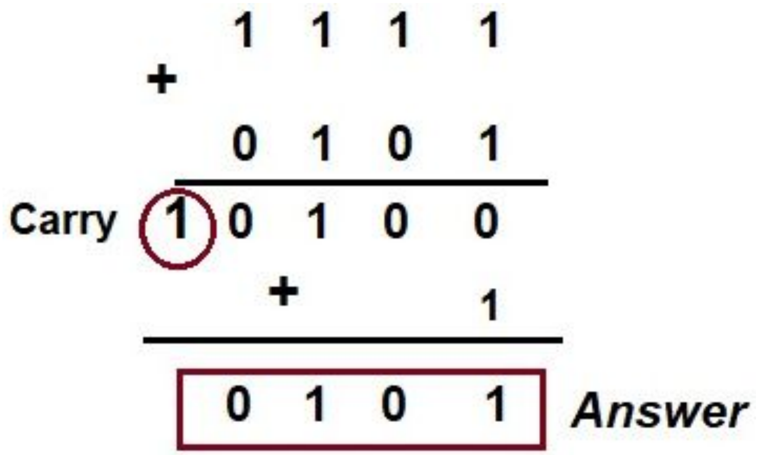
1. Write the 1's complement of the subtrahend.
2. Then add the 1's complement subtrahend with the minuend.
3. If the result has a carryover, then add that carry over in the least significant bit.
4. If there is no carryover, then take the 1's complement of the resultant, and it is negative.

1’s Complement Representation Example

Subtract (1010)2 from (1111)2 using 1’s complement method.

Step-1: Find the 1’s complement of 1010. The required 1’s complement will be 0101.

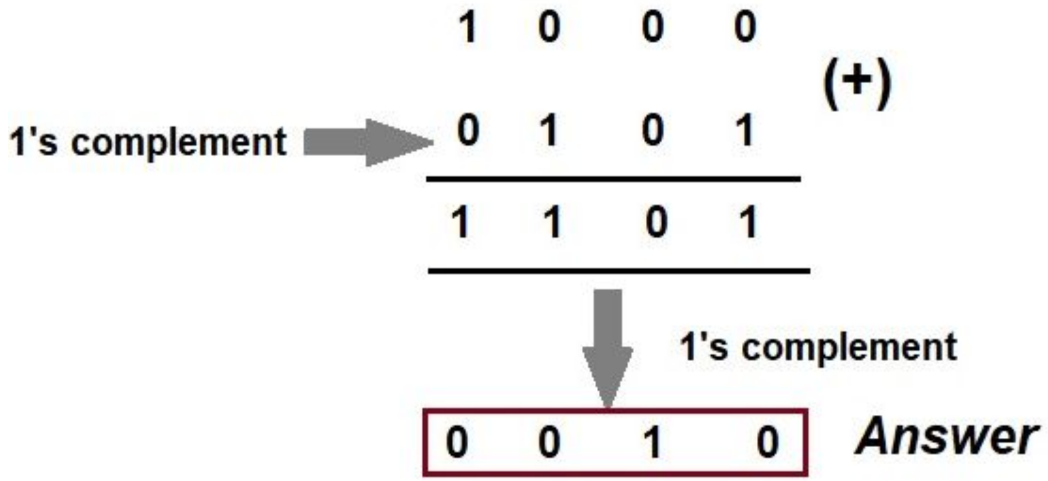
Step-2: In this step, we need to add the value calculated in step-1 to 1111. This is shown below.



1’s Complement Representation Example

Subtract (1010)2 from (1000)2 using 1’s complement method.

=> The 1’s complement of (1010)2 is (0101)2. Now, we will add this with the smaller number and finally take 1’s complement of the result to get the answer.



1’s Complement Representation Example

Practice Example 1: 10101 - 00111

Practice Example 2: 10101 - 10111

2’s Complement Representation

2’s complement of binary number is 1’s complement of given number plus 1 to the least significant bit (LSB).

For example:

2’s complement of binary number 10010 is (01101) + 1 = 01110.

2’s Complement Representation

Example-1 − Find 2’s complement of binary number 10101110.

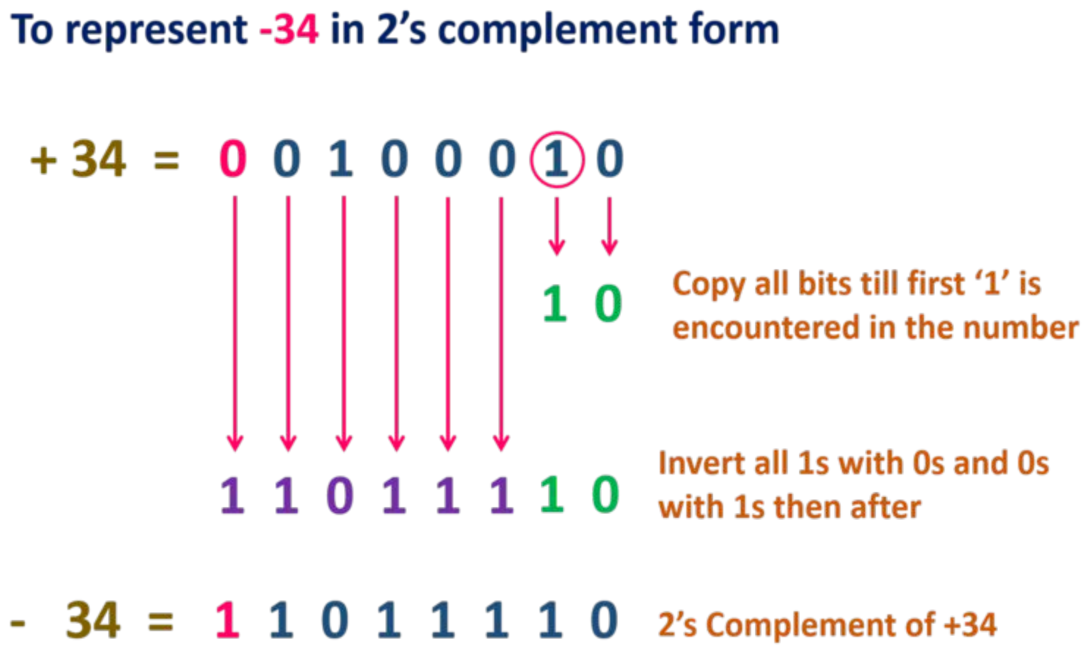
Simply invert each bit of given binary number, which will be 01010001. Then add 1 to the LSB of this result, i.e., 01010001+1=01010010 which is answer.

Example-2 − Find 2’s complement of binary number 10001.001.

Simply invert each bit of given binary number, which will be 01110.110 Then add 1 to the LSB of this result, i.e., 01110.110+1=01110.111 which is answer.

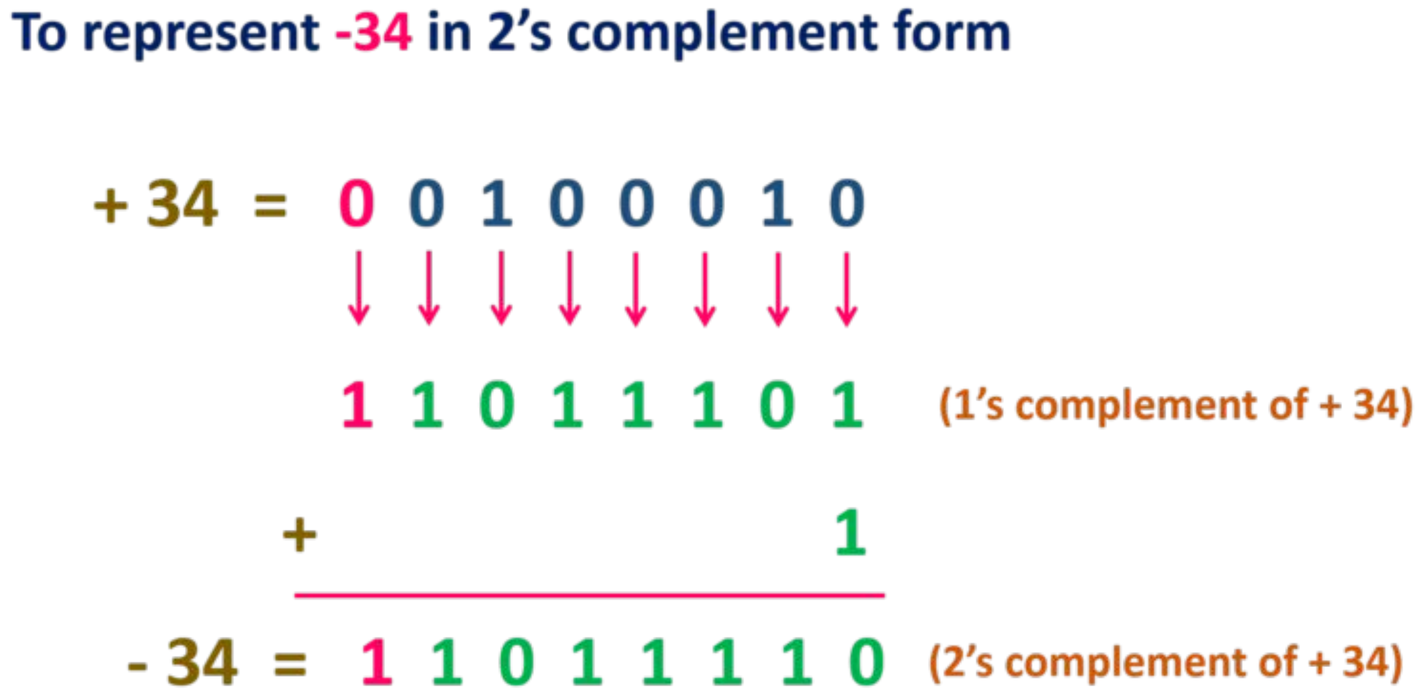
2’s Complement Representation

But the representation of the **negative number** is different.
Example: if we want to represent -34 in 2’s complement form then


To represent -34 in 2’s complement form
+ 34 = 0 0 1 0 0 0 1 0
1 0
Copy all bits till first ‘1’ is encountered in the number
1 1 0 1 1 1 1 0
Invert all 1s with 0s and 0s with 1s then after
- 34 = 1 1 0 1 1 1 1 0
2’s Complement of +34

2's Complement Representation

The second way of representing -34 in 2's complement form is



To represent -34 in 2's complement form

+ 34 = 0 0 1 0 0 0 1 0

$\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$

1 1 0 1 1 1 0 1 (1's complement of + 34)

+ 1

- 34 = 1 1 0 1 1 1 1 0 (2's complement of + 34)

2's Complement Representation

Example 1: 7 - 2 = ?

=> 7 + (-2)

2 -> 0 0 1 0

1 1 1 0 (2's complement of 2 i.e. -2)

```
  7  ->   0 1 1 1
(-2) -> +1 1 1 0
         --------
          0 1 0 1   (5)
```

Sign bit is 0, so the number is positive

2's Complement Representation

Example 2: 3 - 6 = ?

=> 3 + (-6)

6 -> 0 1 1 0

1 0 1 0 (2's complement of 6 i.e. -6)

```
   3  ->   0 0 1 1
(-6) -> +1 0 1 0
          -------
           1 1 0 1
```

Sign bit is 1, so the number is negative

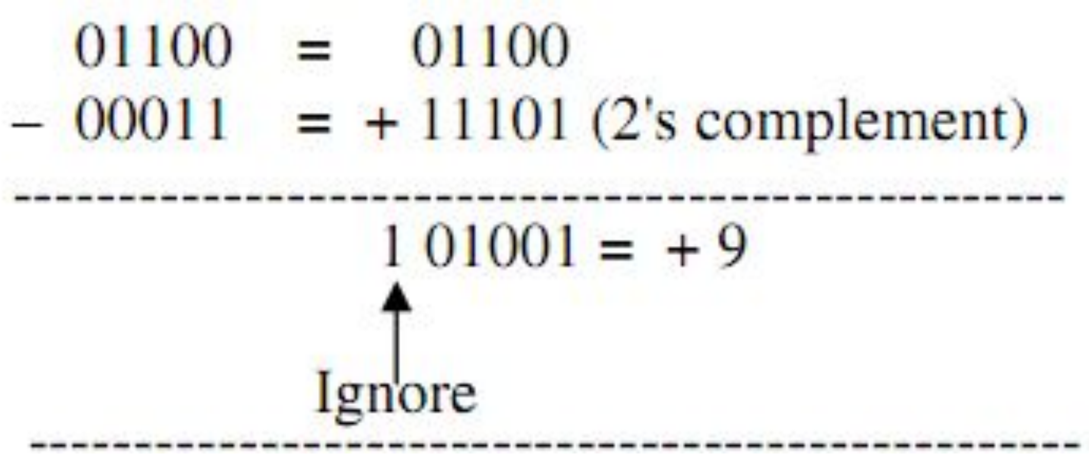
Practice Example

Example 1: Subtraction of 01100-00011 using 2's complement method

Example 2: Perform the operation 5 - 4 using 2’s complement.

Practice Example

Example1: Subtraction of 01100-00011 using 2's complement method



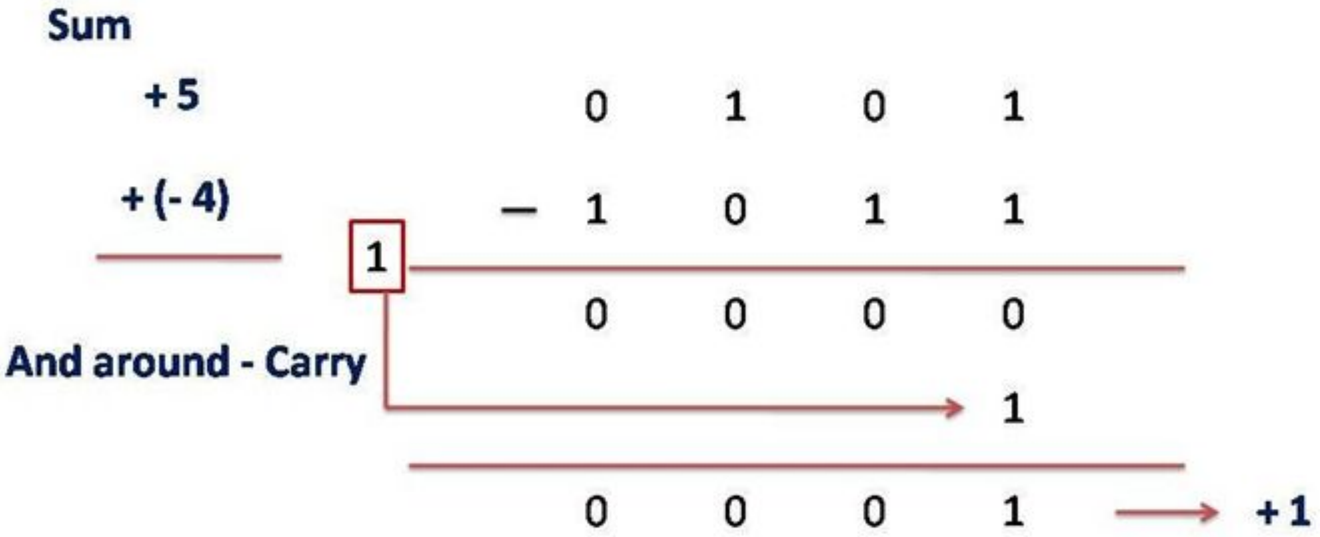
```
   01100   =    01100
 – 00011   =  + 11101 (2's complement)
----------------------------------------
              1 01001 =  + 9
              ↑
           Ignore
----------------------------------------
```

If a last carry is produced discard the carry and the answer is provided by the remaining bits that is positive that is,

$(1001)_2 = (+ 9)_{10}$.

Practice Example

Example2: Perform the operation 5- 4 using 2's complement.



Boolean Algebra

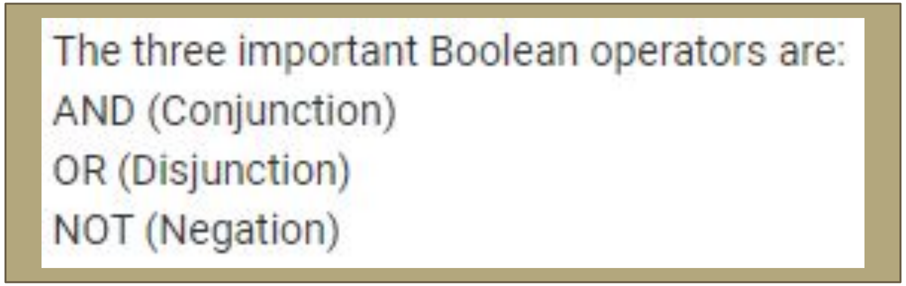
It is a mathematical system that defines a series of logical operations(AND,OR,NOT) performed on a set of variable (a,b,c,...).

In expression, $Y=A+1$

Y is a function having value 0 or 1.

A is a variable having value 0 or 1.

1 is a constant.



The three important Boolean operators are:

AND (Conjunction)

OR (Disjunction)

NOT (Negation)

Boolean Algebra

Complement: It is represented by “bar” over a letter. For example, complement of A will be $\bar{A}$

So, If A= 0 , $\bar{A} = 1$

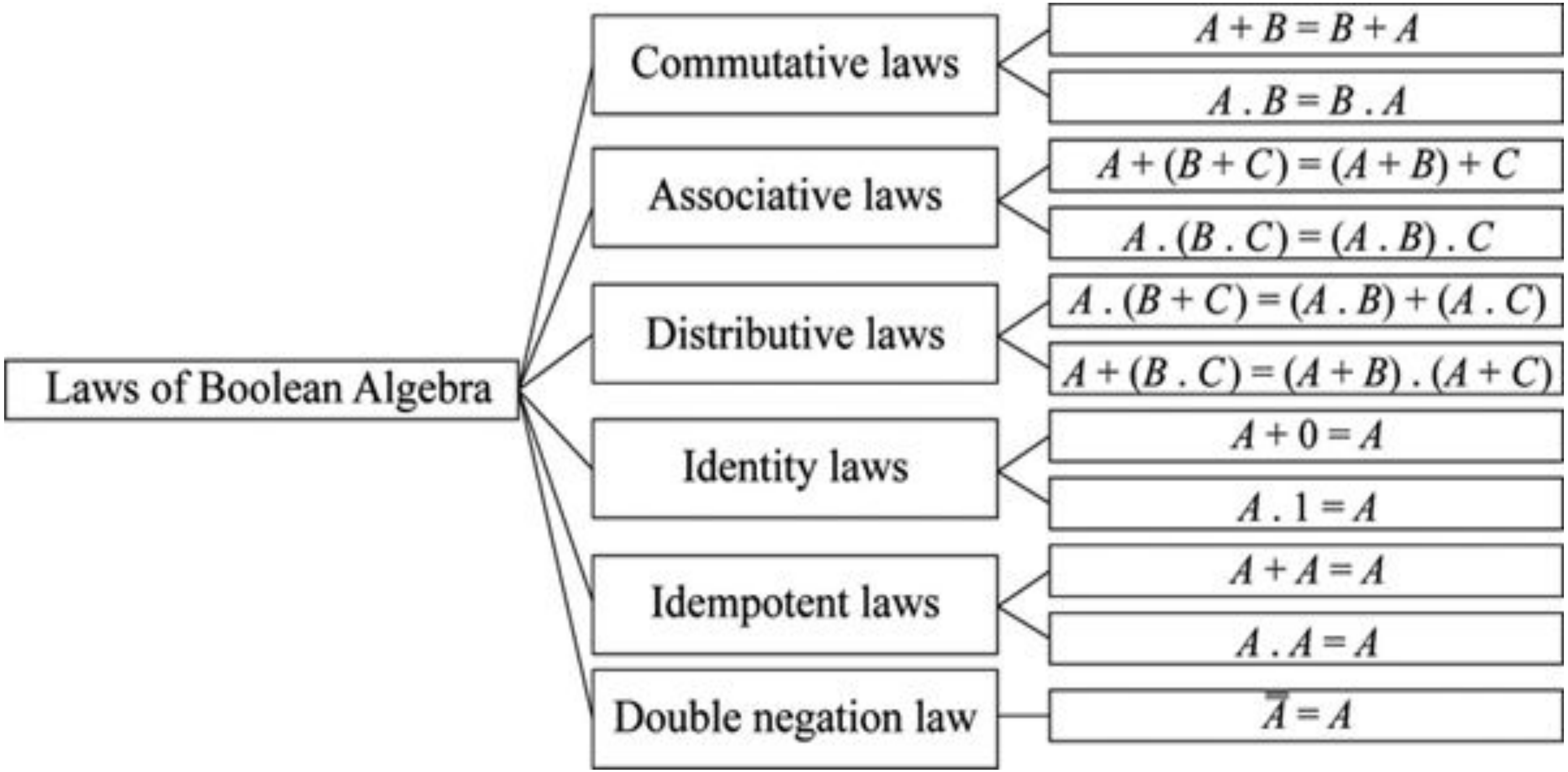
If A= 1 , $\bar{A} = 0$

Boolean function: Boolean expression is used to describe Boolean functions.

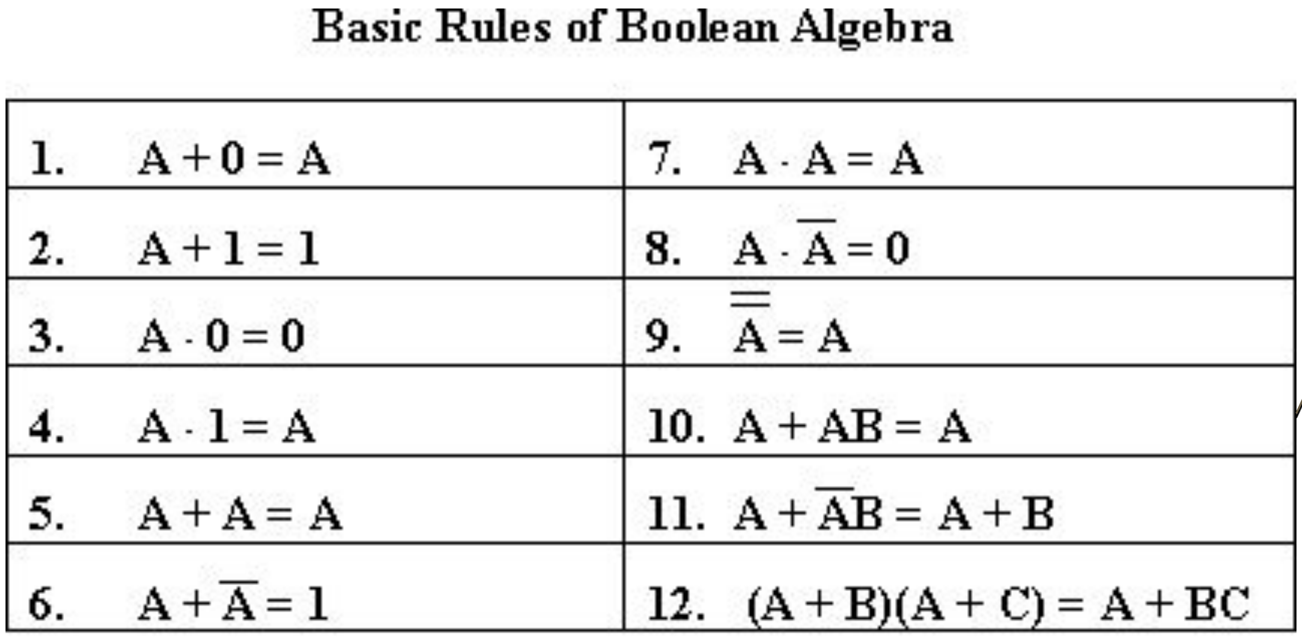
Example: $F(A, B, C, D) = A + \overline{BC} + ADC$

Boolean Function — Boolean Expression

Boolean Laws



Boolean Laws



Basic Rules of Boolean Algebra

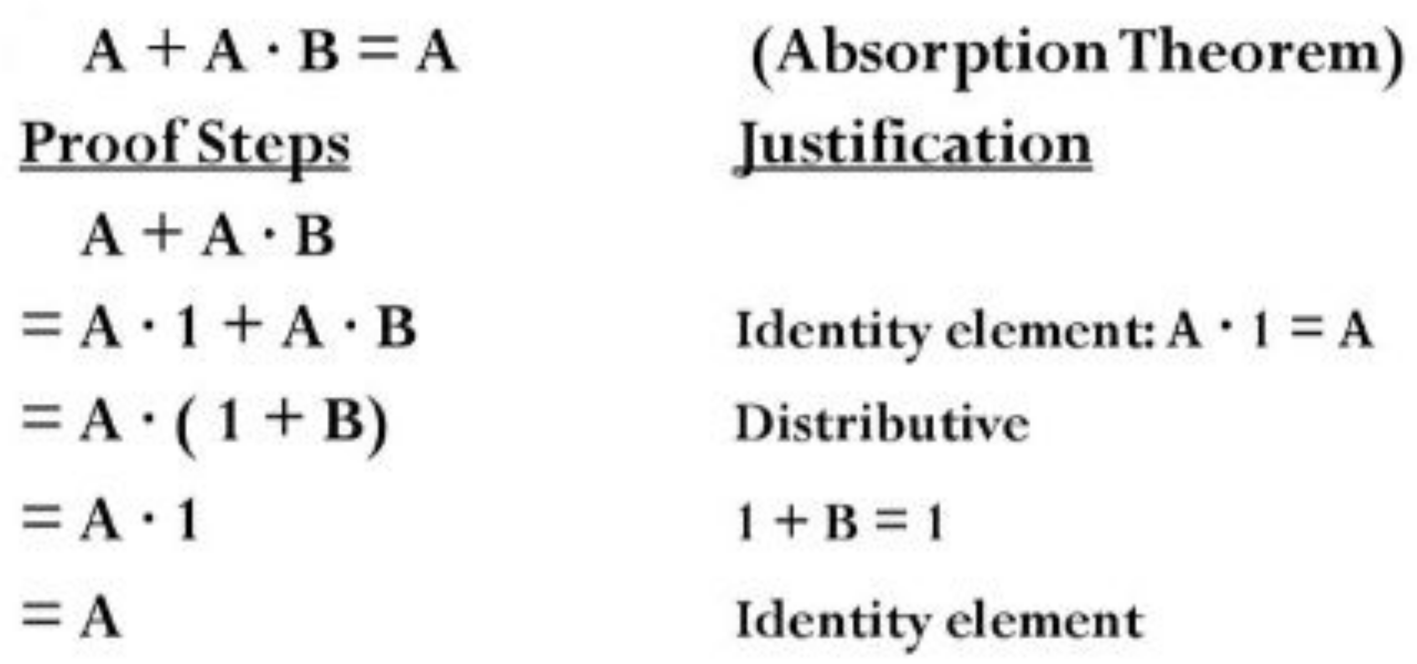
1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

$$\begin{tikzcd}
A + AB \arrow[d] \\
A(1 + B) \arrow[d] \\
A(1) \arrow[d] \\
A
\end{tikzcd}$$

DeMorgan's Theorem

$$\overline{(AB)} = (\overline{A} + \overline{B}) \qquad \overline{(A + B)} = (\overline{A}\ \overline{B})$$

Example 1



$A + A \cdot B = A$ (Absorption Theorem)

Proof Steps	Justification
$A + A \cdot B$	
$= A \cdot 1 + A \cdot B$	Identity element: $A \cdot 1 = A$
$= A \cdot (1 + B)$	Distributive
$= A \cdot 1$	$1 + B = 1$
$= A$	Identity element

Example 2

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

$$\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC \text{(Distributive law)} \\
&= A + AC + AB + BC \quad (AA = A) \\
&= A(1 + C) + AB + BC \qquad (1 + C = 1) \\
&= A.\, 1 + AB + BC \\
&= A(1 + B) + BC \qquad (1 + B = 1) \\
&= A.\, 1 + BC \qquad (A\, .\, 1 = A) \\
&= A + BC
\end{aligned}$$

Sum of Product (SOP)

- The sum-of-products (**SOP**) form is a method (or form) of simplifying the Boolean expressions of logic gates.
- Sum and product derived from the symbolic representations of the OR and AND functions.
- OR (+) , AND (.) , addition and multiplication.

$$f(A,B,C) = \underbrace{ABC \overset{\overset{\text{Sum}}{\downarrow}}{+} A'BC'}_{\text{Product terms}}$$

Product of Sum (POS)

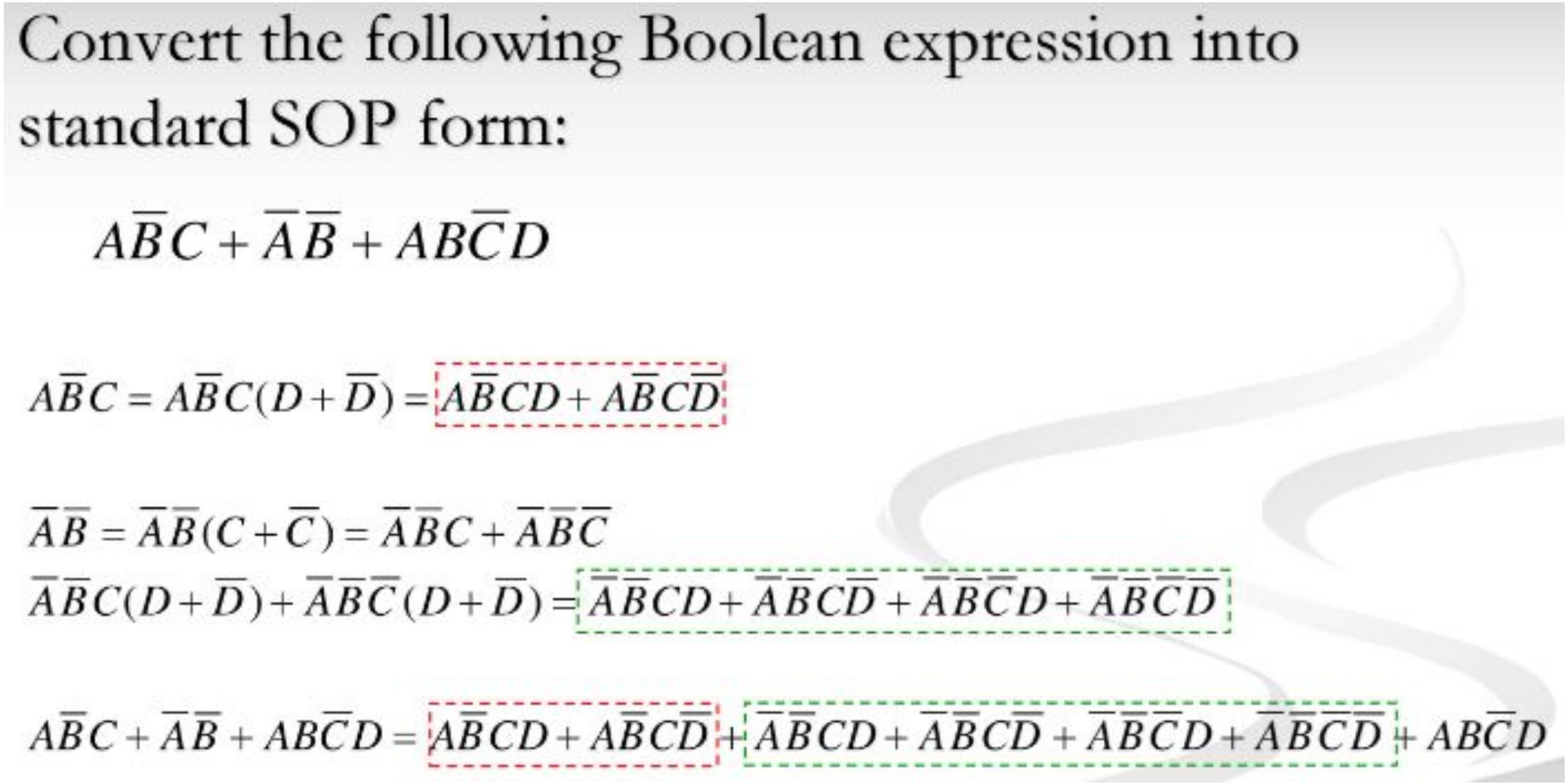
- When two or more sum terms are multiplied by a Boolean OR operation.
- Sum terms are defined by using OR operation and the product term is defined by using AND operation.

$$f(A,B,C) = \underbrace{(A'+B) \overset{\text{Product}}{.} (B+C')}_{\text{Sum terms}}$$

Canonical form

- In SOP or POS form, all individual terms do not involve all literals.
- For example AB + A’BC the first product term do not contain literal C.
- If each term in SOP or POS contain all literals then the expression is known as standard or canonical form.

Standard SOP



Convert the following Boolean expression into standard SOP form:

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

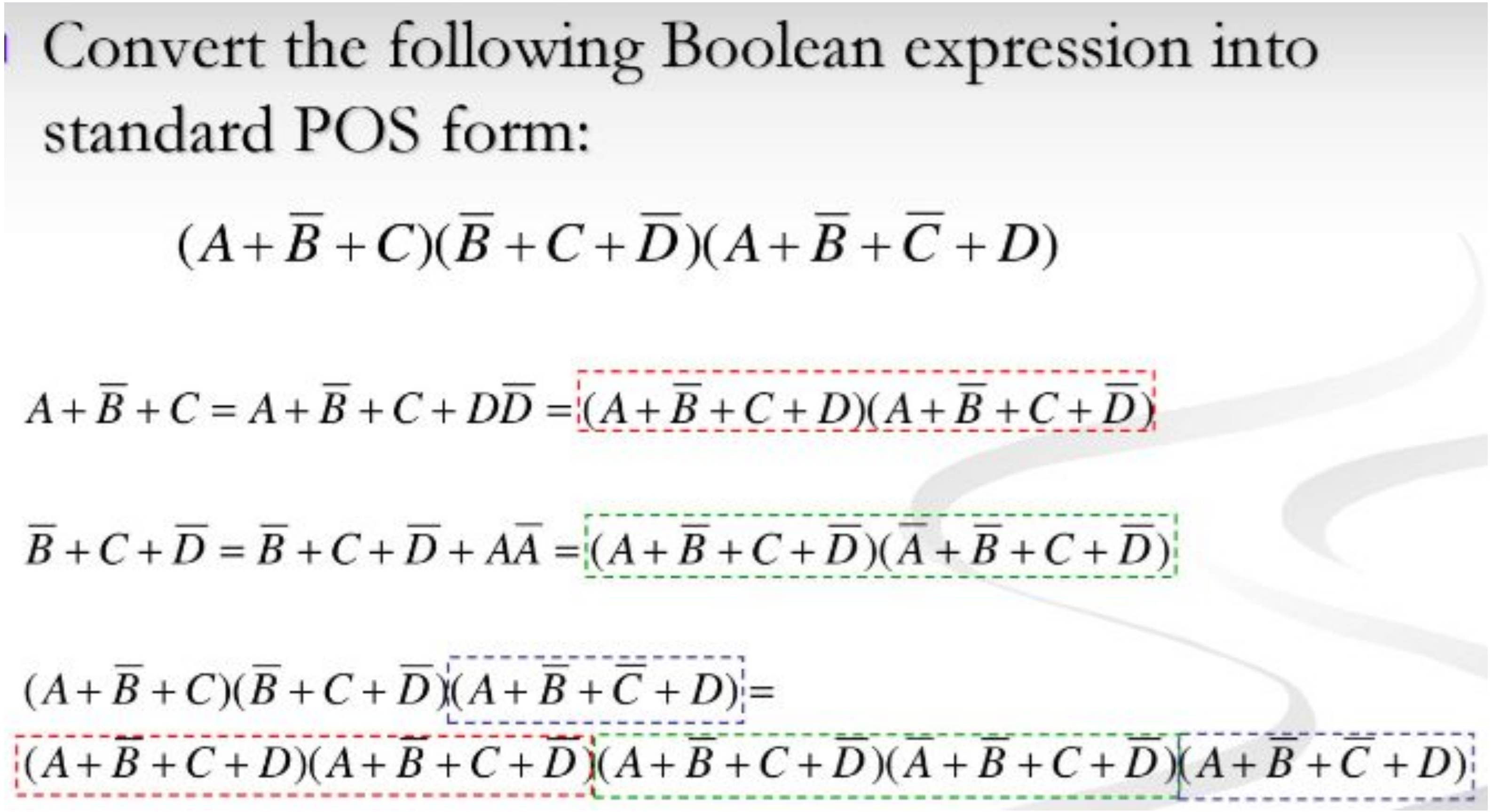
$$A\overline{B}C = A\overline{B}C(D+\overline{D}) = \boxed{A\overline{B}CD + A\overline{B}C\overline{D}}$$

$$\overline{A}\,\overline{B} = \overline{A}\,\overline{B}(C+\overline{C}) = \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$\overline{A}\,\overline{B}C(D+\overline{D}) + \overline{A}\,\overline{B}\,\overline{C}(D+\overline{D}) = \boxed{\overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}}$$

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D = \boxed{A\overline{B}CD + A\overline{B}C\overline{D}} + \boxed{\overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}} + AB\overline{C}D$$

Standard POS



Convert the following Boolean expression into standard POS form:

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

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

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

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

$$\boxed{(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})}\boxed{(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})}\boxed{(A+\overline{B}+\overline{C}+D)}$$

Minterm and Maxterm

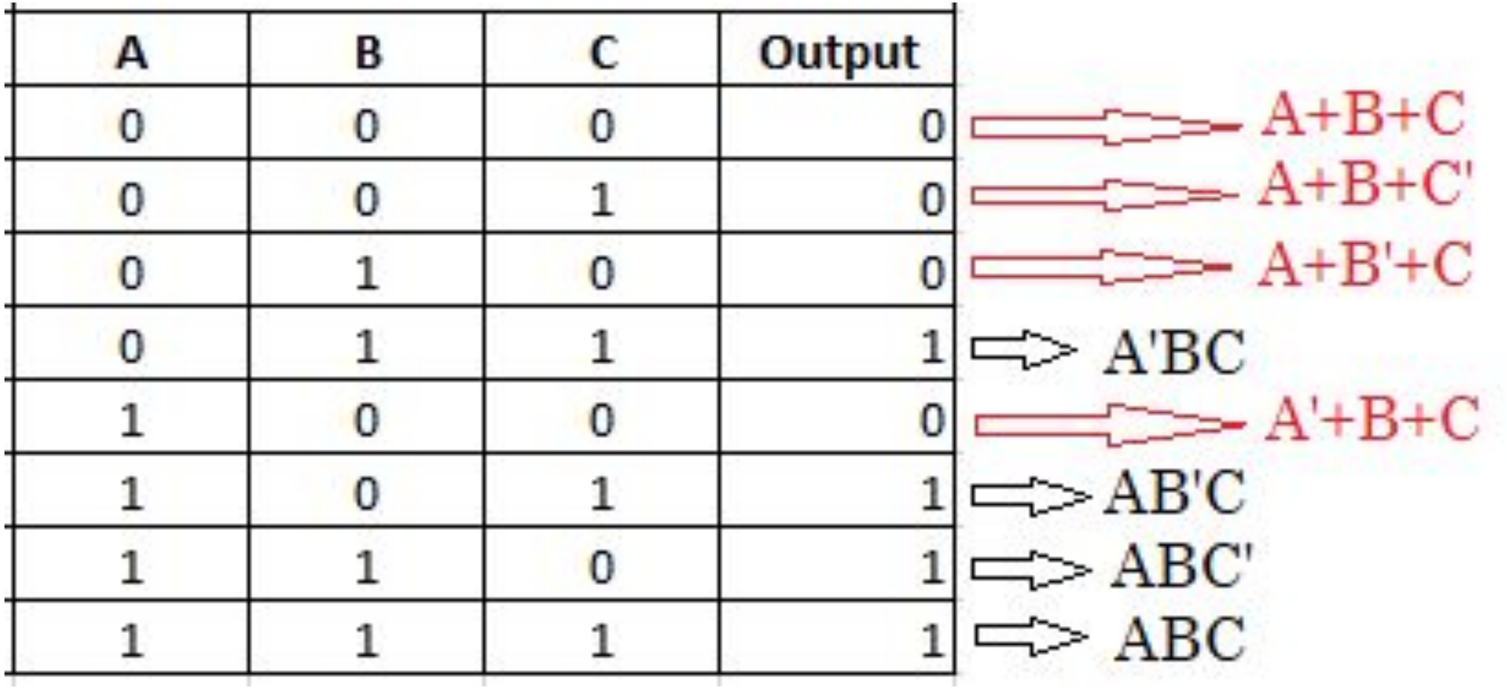
Minterm is **product term** in which the Boolean variable is either normal or in complemented form.
Each individual termin standard SOP form is called Minterm.

Maxterm is a **sum term** in which the Boolean variable is either normal or in complemented form.
Each individual termin standard POS form is called Maxterm.

Minterm and Maxterm

- Each individual term in the POS form is called **Maxterm**.
- Each individual term in the SOP form is called **Minterm**.
- In Minterm, we look for the functions where the output results is “1”.
- while in Maxterm we look for function where the output results is “0”.
- We perform **Sum of minterm** also known as Sum of products (SOP) .
- We perform **Product of Maxterm** also known as Product of sum (POS).

Minterm and Maxterm example

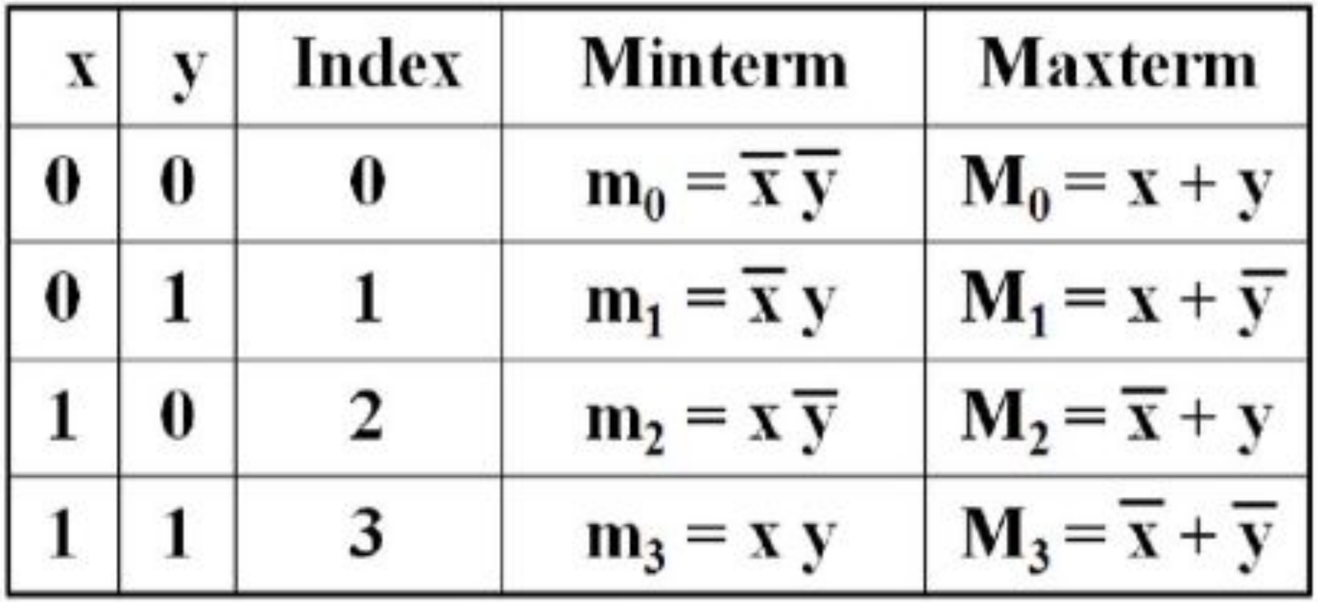


A	B	C	Output	
0	0	0	0	A+B+C
0	0	1	0	A+B+C'
0	1	0	0	A+B'+C
0	1	1	1	A'BC
1	0	0	0	A'+B+C
1	0	1	1	AB'C
1	1	0	1	ABC'
1	1	1	1	ABC

SOP Expression: A’BC+AB’C+ABC’+ABC
POS Expression: (A+B+C)(A+B+C’)(A+B’+C)(A’+B+C)

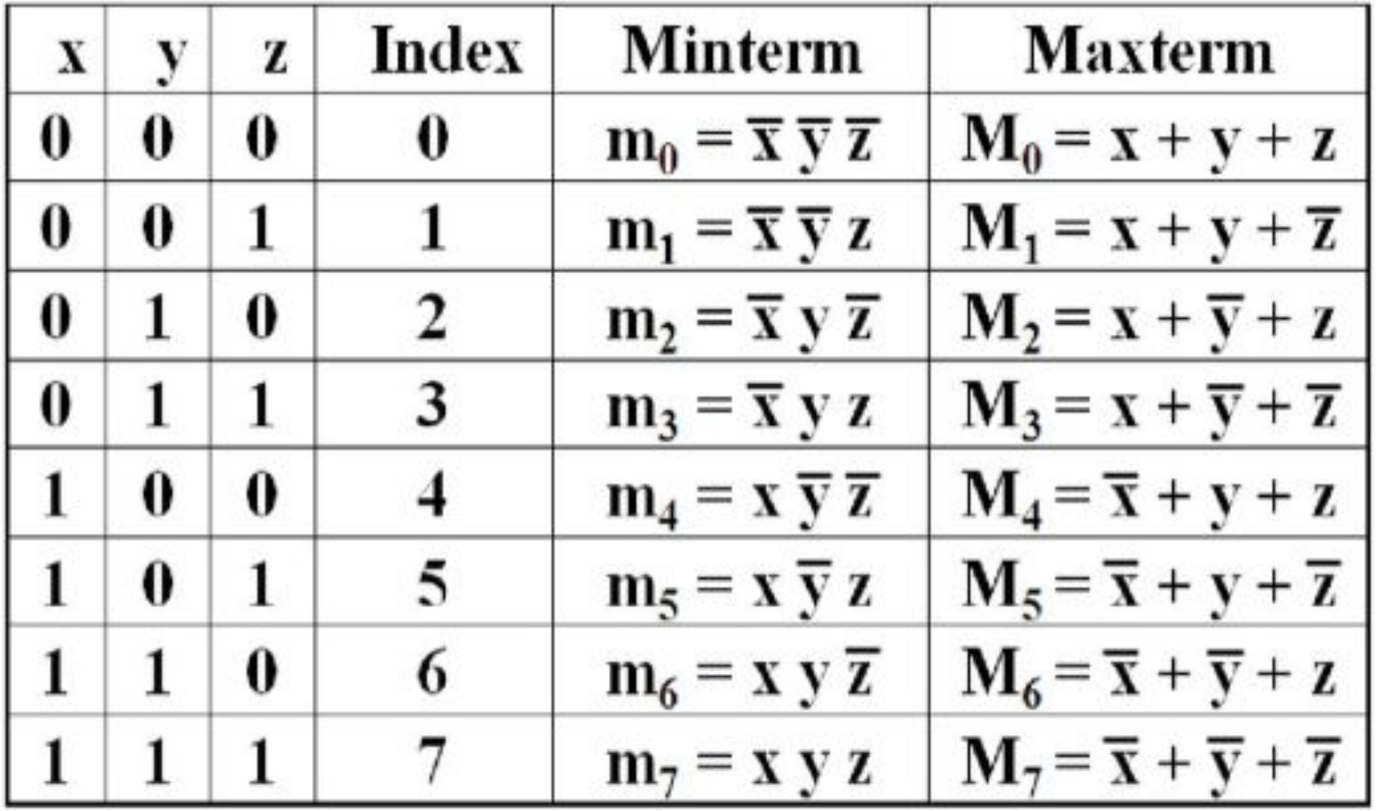
Minterm and Maxterm for 2 variable

Two variable minterms and maxterms.



x	y	Index	Minterm	Maxterm
0	0	0	$m_0 = \overline{x}\,\overline{y}$	$M_0 = x + y$
0	1	1	$m_1 = \overline{x}\,y$	$M_1 = x + \overline{y}$
1	0	2	$m_2 = x\,\overline{y}$	$M_2 = \overline{x} + y$
1	1	3	$m_3 = x\,y$	$M_3 = \overline{x} + \overline{y}$

Minterm and Maxterm for 3 variable



x	y	z	Index	Minterm	Maxterm
0	0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$
0	0	1	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$
0	1	0	2	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$
0	1	1	3	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$
1	0	0	4	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$
1	0	1	5	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$
1	1	0	6	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$
1	1	1	7	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$

Example 1: **Express the Boolean function F=A+BC as minterm**

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

By complement law A+A'=1. Adding all the input variables in the product terms we get

F=(AB+AB')(C+C')+ ABC+ A'BC

F= ABC+ABC'+AB'C+ABC'+AB'C'+ABC+A'BC

There are two ABCs and ABC'. So let us consider one ABC and ABC'

F= ABC+ ABC'+AB'C+ AB'C'+ A'BC

In minterm the '1' input is written as such and the input '0' is complimented.

F= 111+110+101+100+011

F= m7+m6+m5+m4+m3

F=∑m(3,4,5,6,7)

Example 2: Express the Boolean function F=A+BC as maxterm

F=A+(BC)

After removing the bracket and expanding we get

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

By complement law CC'=BB'=1. So after adding all input terms in the expression we get,

F= (A+B+CC').(A+C+BB')

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

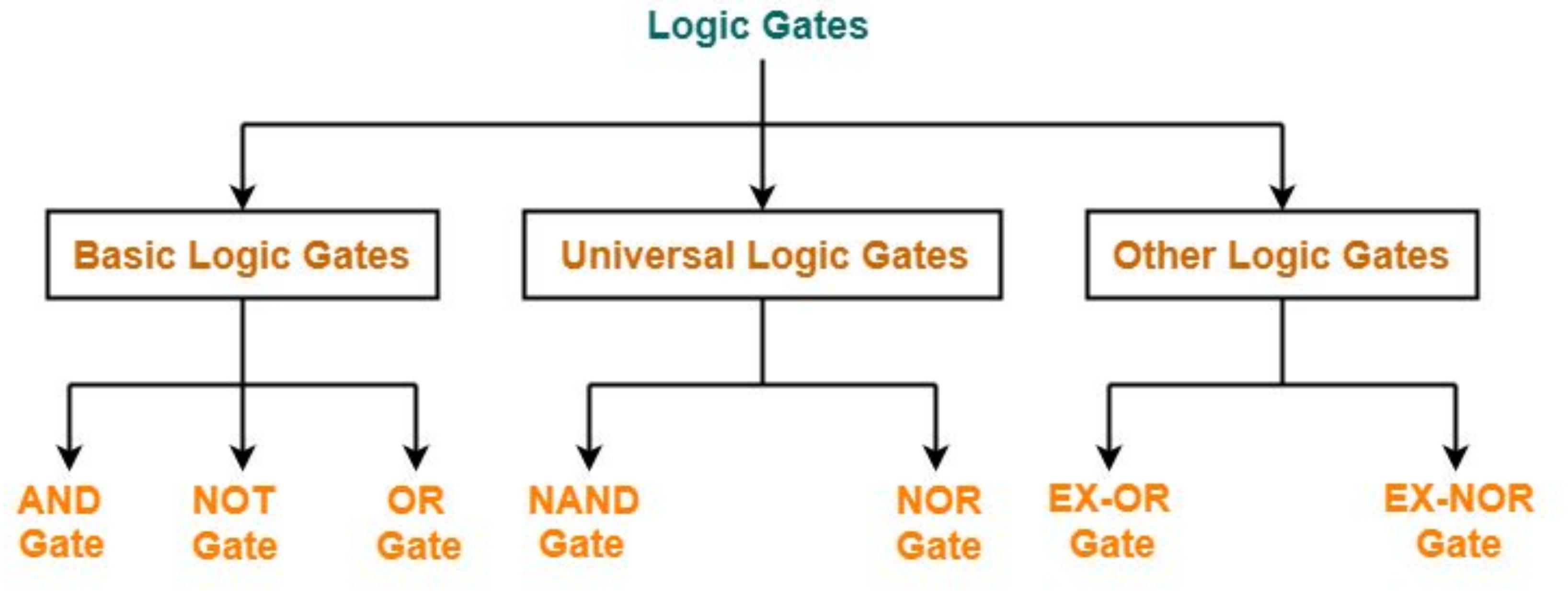
There are two (A+B+C)s. So let us consider one (A+B+C)

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

In Maxterm the input ‘0’ is written as such and the input ‘1’ is complemented.

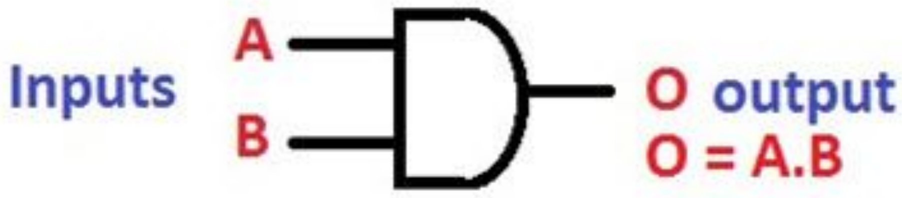
F=(0+0+0).(0+0+1).(0+1+0) = M0.M1.M2 = πM(0,1,2)

Logic gates

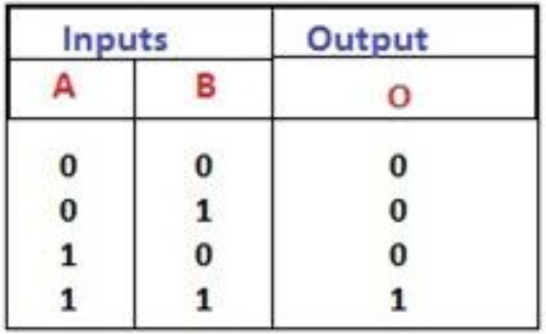


Types of Logic Gates

Logic gates: AND Gate



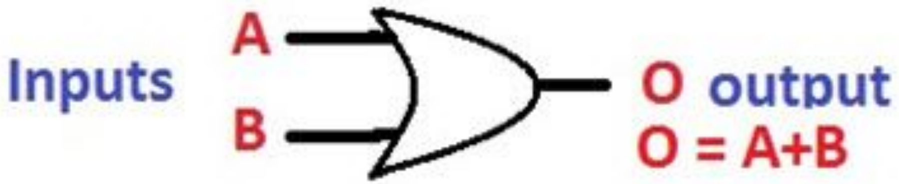
Symbol



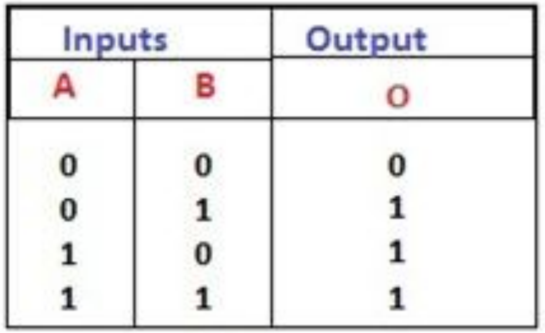
Inputs		Output
A	B	O
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

Logic gates: OR Gate



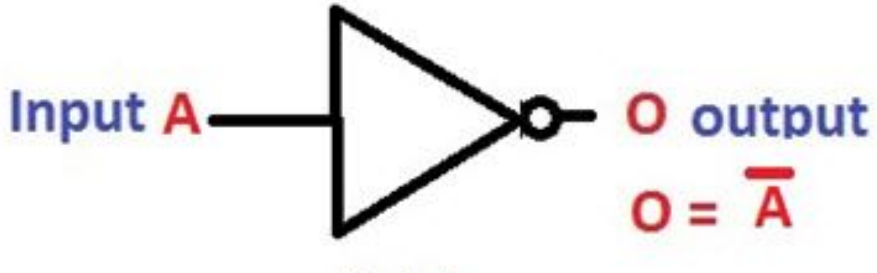
Symbol



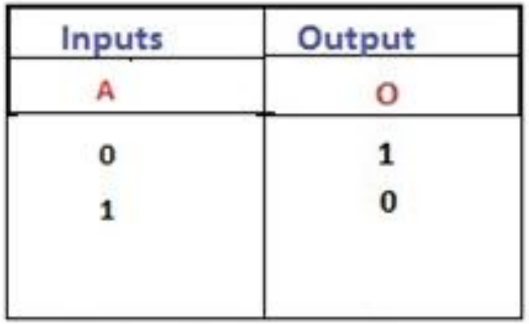
Inputs		Output
A	B	O
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

Logic gates: NOT Gate



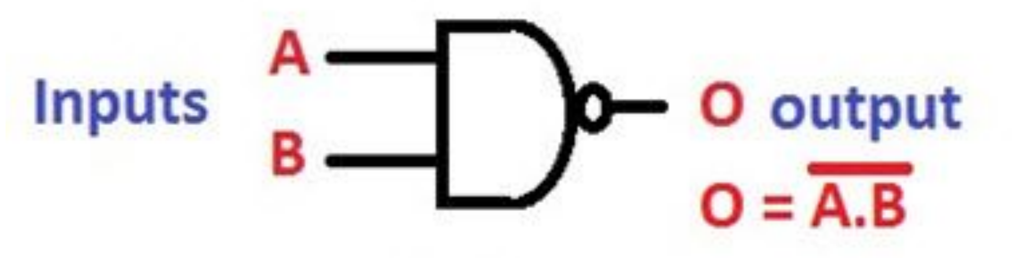
Symbol



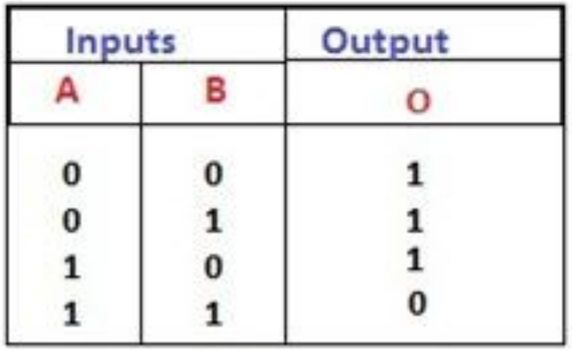
Inputs	Output
A	O
0	1
1	0

Truth table

Logic gates: NAND Gate



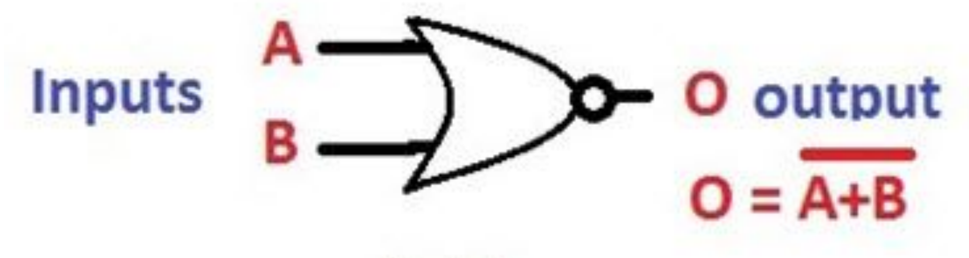
Symbol



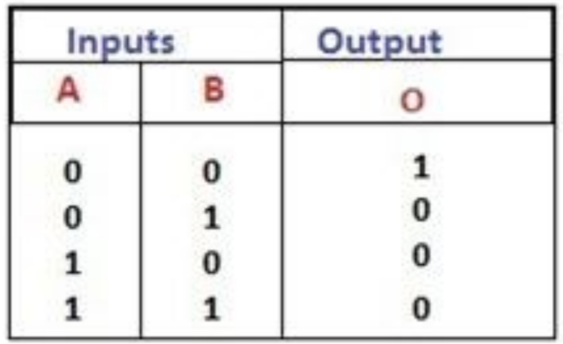
Inputs		Output
A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

Logic gates: NOR Gate



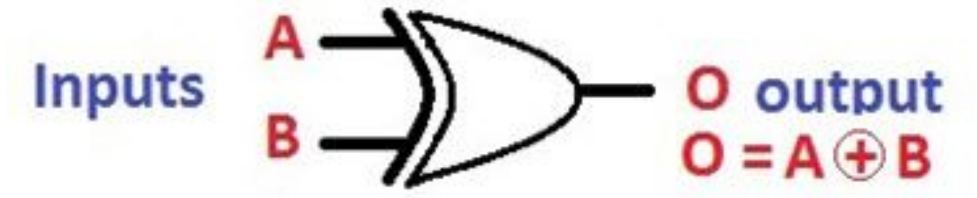
Symbol



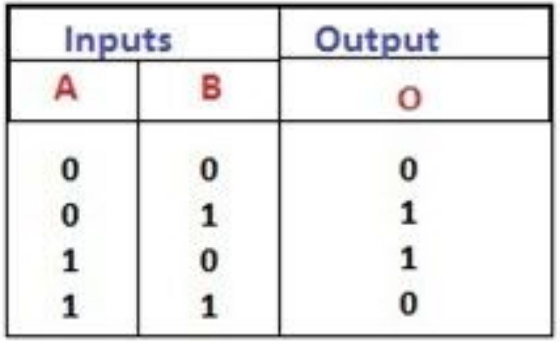
Inputs		Output
A	B	O
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

Logic gates: Exclusive-OR Gate



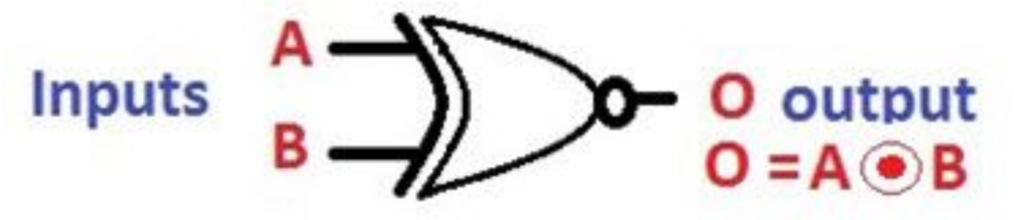
Symbol



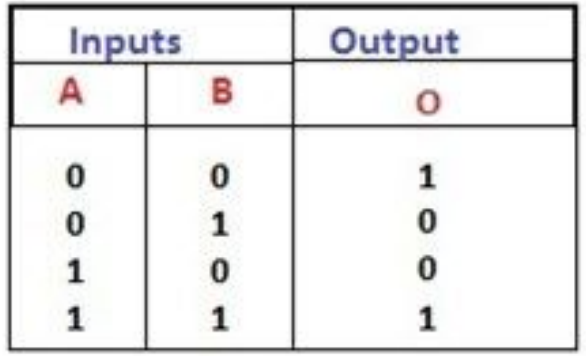
Inputs		Output
A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

Logic gates: Exclusive-NOR Gate



Symbol



Inputs		Output
A	B	O
0	0	1
0	1	0
1	0	0
1	1	1

Truth table

Adder

- An adder is **a device that will add together two bits and give the result as the output**.
- The bits being added together are called the "addends".
- Adders can be concatenated in order to add together two binary numbers of an arbitrary length.
- There are two kinds of adders - **half adders and full adders**.

Half adder

- Half Adder is the digital logic circuit that is used to implement the **binary addition**.

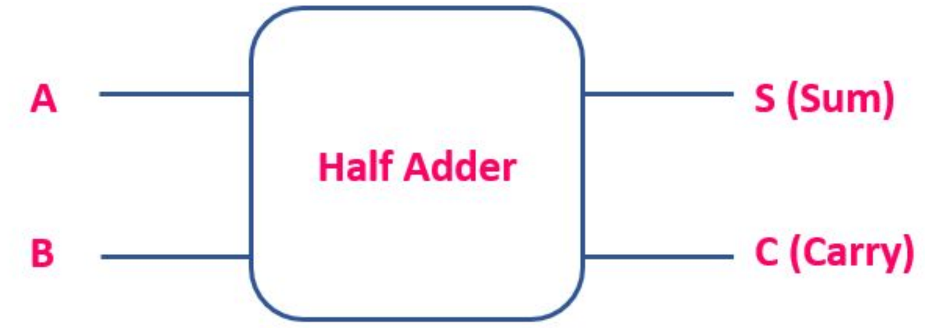
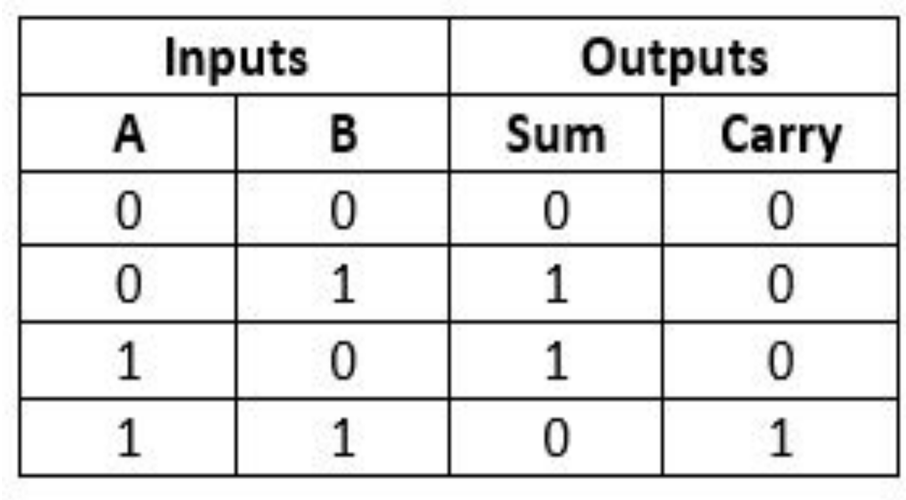


Fig: Block diagram

Half adder circuit

- It adds the two bits and generates the sum bit (S) and carry bit (C) as an output.



Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig: Truth Table

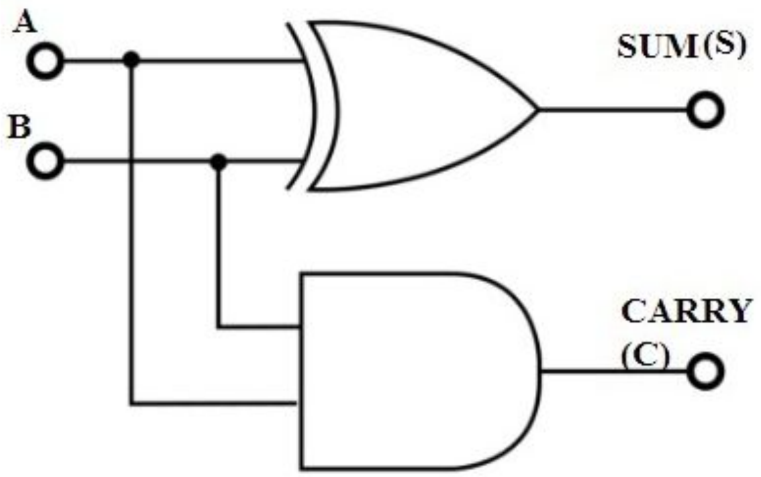


Fig: Half adder circuit

Half adder

$$S = A \oplus B$$

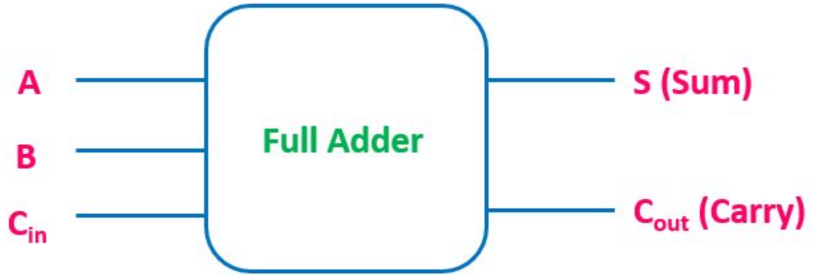
$$C = A \cdot B$$

The main disadvantage of this circuit is that it can only add two inputs and if there is any carry, it is neglected. Thus, the process is incomplete.

To overcome this difficulty Full Adder is designed.

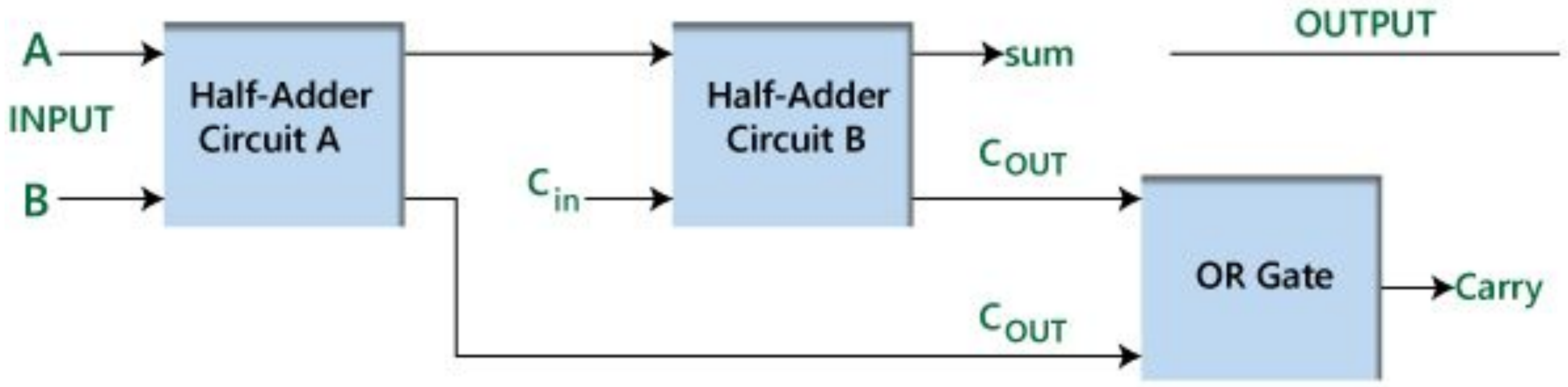
Full adder

The full adder is the combinational circuit that adds the two bits along with the incoming carry (Cin) and generates the sum bit (S) and an outgoing carry bit (Cout) as an output.

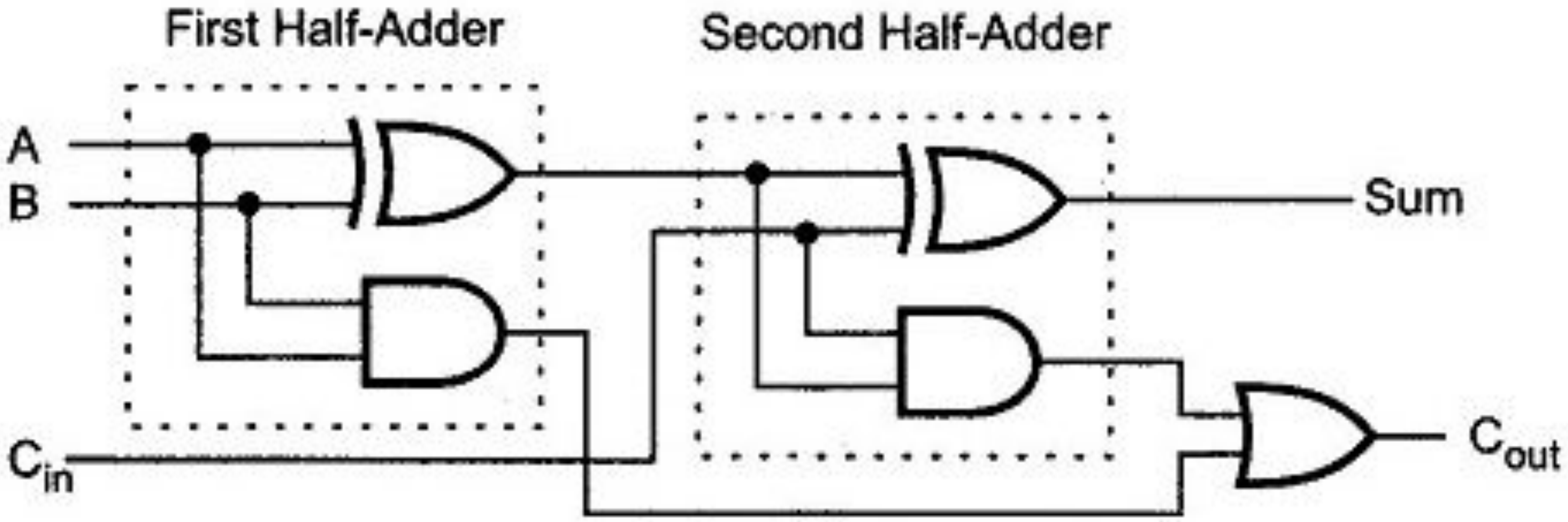


Full adder

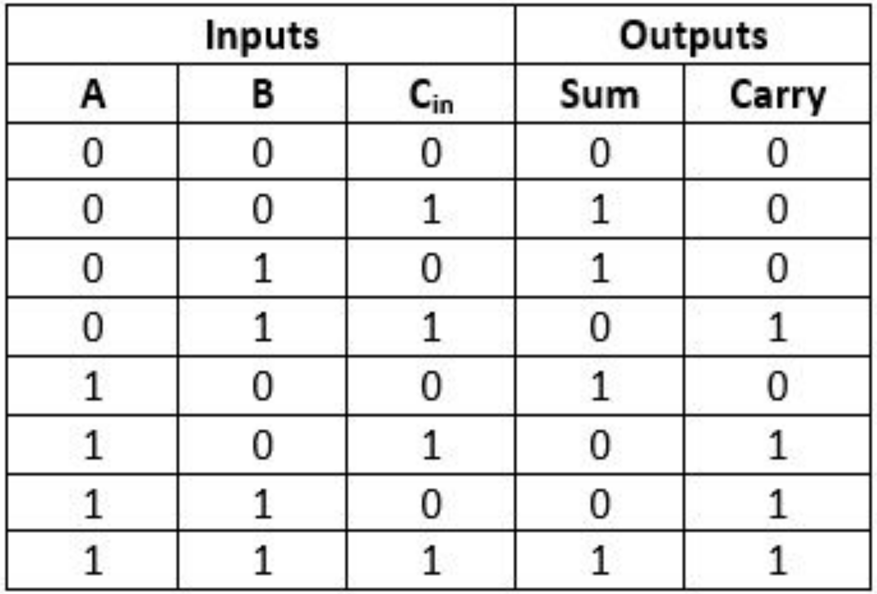
A full-adder can also be implemented with two half-adders and one OR gate



Full adder



Full adder truth table and sum expression



Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean expression for the sum:

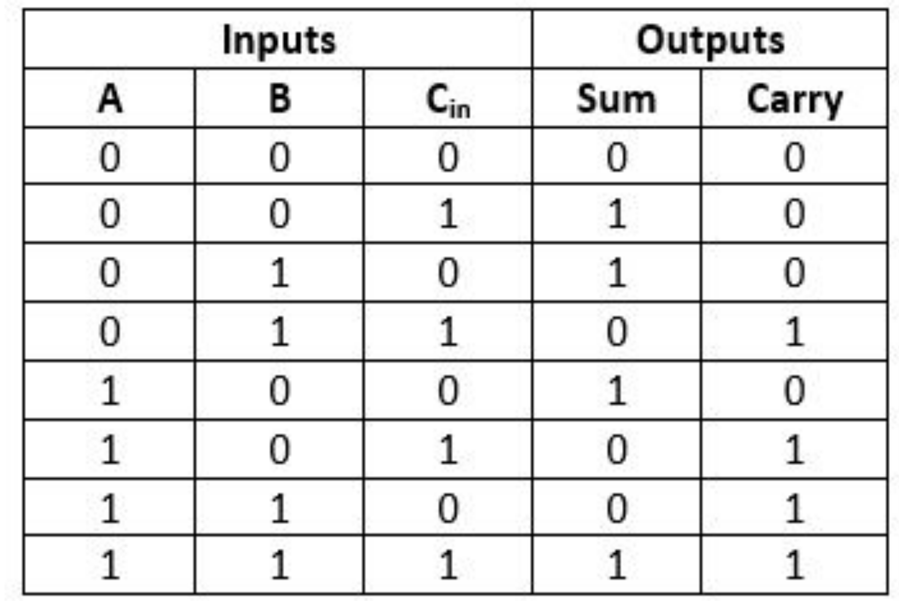
$$S = \overline{A}\,\overline{B}\,C_{in} + \overline{A}\,B\,\overline{C_{in}} + A\,\overline{B}\,\overline{C_{in}} + A\,B\,C_{in}$$

$$= \overline{A}\,(\overline{B}\,C_{in} + B\,\overline{C_{in}}) + A\,(\overline{B}\,\overline{C_{in}} + B\,C_{in})$$

$$= \overline{A}\,(B \oplus C_{in}) + A\,(\overline{B \oplus C_{in}})$$

$$S = A \oplus B \oplus C_{in}$$

Full adder truth table and carry expression



Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean expression for carry:

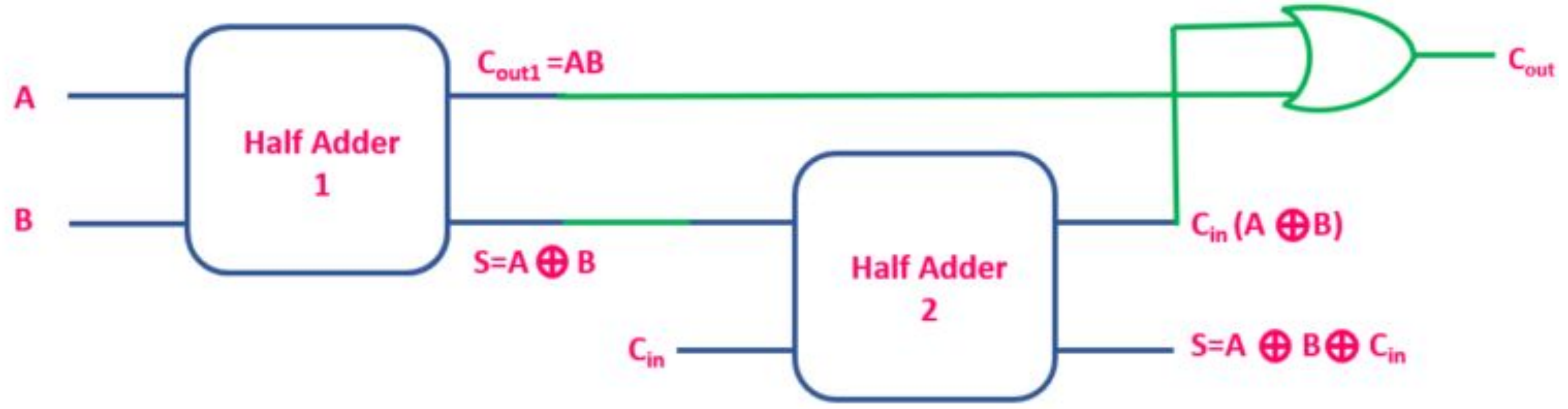
$$C_{out} = \overline{A}\,B\,C_{in} + A\,\overline{B}\,C_{in} + A\,B\,\overline{C_{in}} + A\,B\,C_{in}$$

$$= A\,B\,(C_{in} + \overline{C_{in}}) + C_{in}\,(\overline{A}\,B + A\,\overline{B})$$

$$= A\,B + C_{in}\,(A \oplus B)$$

$$C_{out} = AB + BC_{in} + AC_{in}$$

Full adder circuit with Boolean expression for sum and carry


A
B
Half Adder 1
C_{out1} =AB
S=A ⊕ B
C_{in}
Half Adder 2
C_{in} (A ⊕B)
S=A ⊕ B⊕ C_{in}
C_{out}