Chapter 4

Combinational Logical Circuit

Introduction

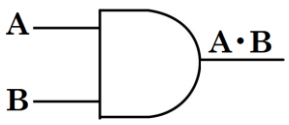
In digital circuit theory, **combinational logic** (sometimes also referred to as **time-independent logic**) is a type of digital logic which is implemented by Boolean circuits, where the output is a pure function of the present input only.

Combinational logic is used in computer circuits to perform Boolean algebra on input signals and on stored data. Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed using combinational logic. Other circuits used in computers, such as half adders, full adders, half subtractors, full subtractors, multiplexers, demultiplexers, encoders and decoders are also made by using combinational logic. (Wikipedia, 2019)

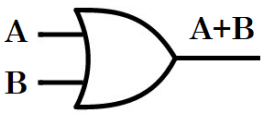
Logic Circuits Algebraically

Logic circuit can be built using basic logic gates such as AND, OR, NOT, NAND, and NOR gate. The inputs of a logic circuit are labeled as Boolean variables and the output is considered the Boolean expression or equation. For this, utilizing the Boolean variables and logic operation for each gate, we can easily determine the expression for the output. To generate a Boolean equation, an AND gate performs logic multiplication, an OR gate performs logic addition, and a NOT gate is the inversion of the input:

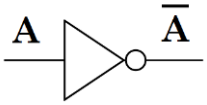
AND gate ➔ logic multiplication



OR gate ➔ logic addition

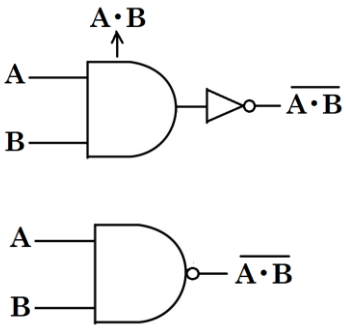


NOT gate ➔ Logic inversion

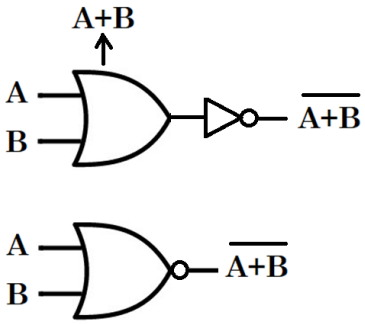


The other two logic gates, NAND and NOR gates, is the inversion the basic AND and OR gate:

NAND gate

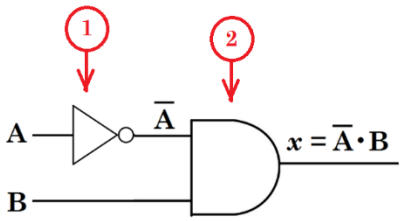


NOR gate



Example) Given the following logic circuits, find the Boolean equation of output ***x***:

a)



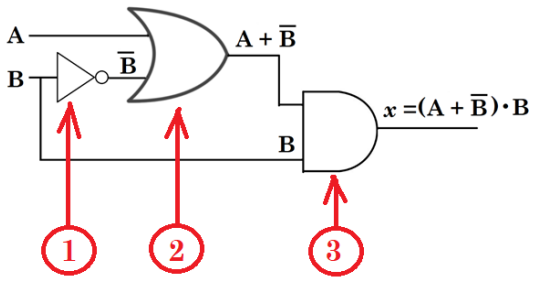
Steps:

1. Invert input A
2. Multiply $\bar{A}$ and B

Boolean equation of $x =$ $\boldsymbol{\bar{A} \cdot B}$

---

b)



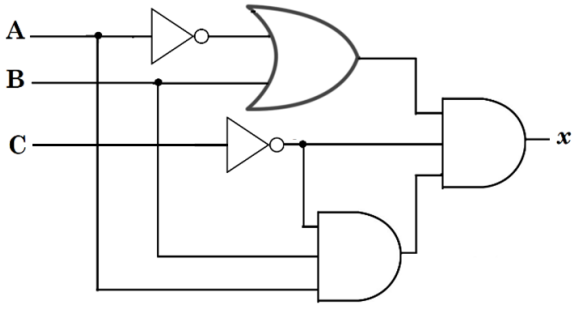
Steps:

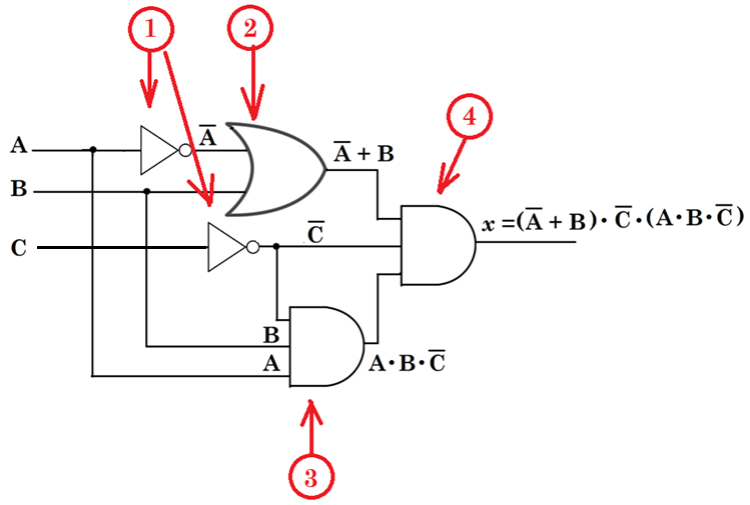
1. Invert input B
2. Add input A and $\bar{B}$
3. Multiply input $(A + \bar{B}) and\ B$

Boolean equation of $x =$ $\boldsymbol{(A + \bar{B}) \cdot B}$

---

c)





Steps:

1. Invert input A and C

2. Add input $\bar{A}$ and B

3. Multiply input A, B, and $\bar{C}$

4. Multiply input $(\overline{\boldsymbol{A}} + \boldsymbol{B})$, $\overline{\boldsymbol{C}}$ and $(\boldsymbol{A} \cdot \boldsymbol{B} \cdot \overline{\boldsymbol{C}})$

Boolean equation of x = $(\overline{\boldsymbol{A}} + \boldsymbol{B}) \cdot \overline{\boldsymbol{C}} \cdot (\boldsymbol{A} \cdot \boldsymbol{B} \cdot \overline{\boldsymbol{C}})$

Sum of Product (SOP) Form

The sum-of-products (SOP) form is a method (or form) of simplifying the Boolean expressions of logic gates. In this SOP form of Boolean function representation, the variables are operated by AND (product) to form a product term and all these product terms are ORed (summed or added) together to get the final function.

A sum-of-products form can be formed by adding (or summing) two or more product terms using a Boolean addition operation. Here the product terms are defined by using the AND operation and the sum term is defined by using OR operation.

The sum-of-products form is also called as Disjunctive Normal Form as the product terms are ORed together and Disjunction operation is logical OR. Sum-of-products form is also called as Standard SOP.

SOP form representation is most suitable to use them in FPGA (Field Programmable Gate Arrays).

SOP form can be obtained by

- Writing an AND term for each input combination, which produces HIGH output.
- Writing the input variables if the value is 1, and write the complement of the variable if its value is 0.
- OR the AND terms to obtain the output function.

Example)

$A\bar{B} + ABC + \bar{A}B\bar{C}$

Product of Sums (POS) Form

The product of sums form is a method (or form) of simplifying the Boolean expressions of logic gates. In this POS form, all the variables are ORed, i.e. written as sums to form sum terms.

All these sum terms are ANDed (multiplied) together to get the product-of-sum form. This form is exactly opposite to the SOP form. So this can also be said as "Dual of SOP form".

Here the sum terms are defined by using the OR operation and the product term is defined by using AND operation. When two or more sum terms are multiplied by a Boolean OR operation, the resultant output expression will be in the form of product-of-sums form or POS form.

The product-of-sums form is also called as Conjunctive Normal Form as the sum terms are ANDed together and Conjunction operation is logical AND. Product-of-sums form is also called as Standard POS.

POS form can be obtained by:

- Writing an OR term for each input combination, which produces LOW output.
- Writing the input variables if the value is 0, and write the complement of the variable if its value is 1.
- AND the OR terms to obtain the output function.

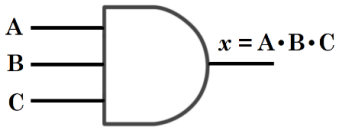
Example)

$(A + B)(A + \bar{B} + C)(\bar{A} + \bar{C})$

(Electronics Hub, 2019)

Implementing circuits from Boolean expressions

When the operation of a circuit is defined by a Boolean expression, we can draw a logic-circuit diagram directly from that expressions. For example, if we needed a circuit that was defined as $x = A \cdot B \cdot C$, we would immediately know that all that was needed was a three-input AND gate:



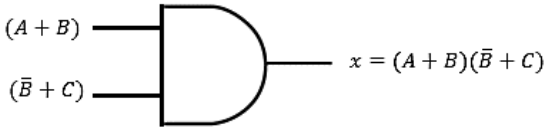
Example) Draw the circuit diagram to implement the expression:

a) $x = (A + B)(\bar{B} + C)$

Step 1) To draw this circuit, we identify the operation between the two terms in parenthesis:

Multiplication = AND gate

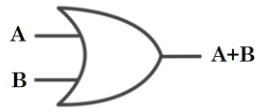
$$x = (A + B)(\bar{B} + C)$$

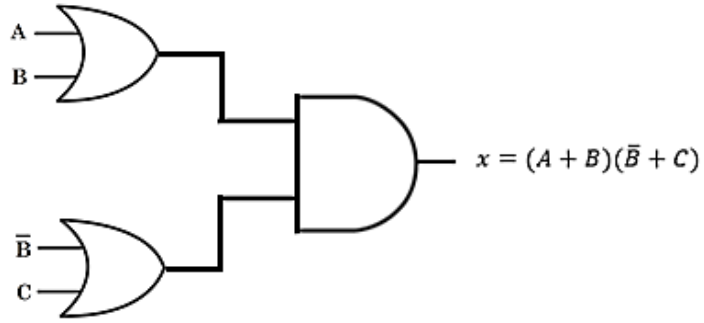


Step 2) Now, we can separate the parenthesis terms and identify the logic gate use in the operation:

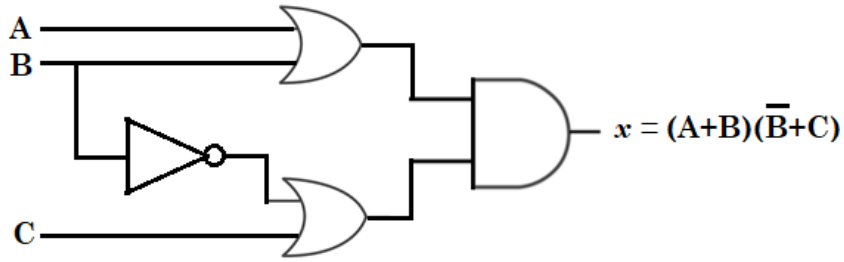
$(A + B)$
Addition of input A and B ➔ OR gate

$(\bar{B} + C)$
Addition of input $\bar{B}$ and B ➔ OR gate





Step 3) The last step is to add a NOT gate to invert input B



Exclusive-OR and Exclusive-NOR

There are two remaining gates of the primary electronics logic gates: *XOR*, which stands for *Exclusive OR*, and *XNOR*, which stands for *Exclusive NOR*. In an XOR gate, the output is HIGH if one, and only one, of the inputs is HIGH. If both inputs are LOW or both are LOW, the output is LOW.

X-OR and X-NOR gates are built using AND, OR, and NOT gates. We can find the logic circuit of the X-OR and X-NOR using its respective truth table. From the truth table, we can write the term when the output will be 1

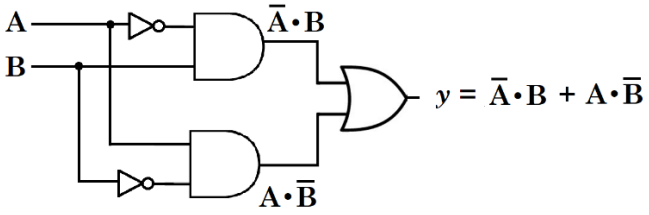
2-input X-OR Gate – Truth Table		
Input		Output
A	B	$Y = A \oplus B$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

To express the Boolean variable, the input that has the value 1 is represented as the Boolean variable while the value 0 is represented as the inverse of its Boolean variable. Also, if the inputs are of the same output, the inputs are multiplied (AND)

	Input	
	A	B
$\bar{A} \cdot B$	$0 = \bar{A}$	1 = B
$A \cdot \bar{B}$	1 = A	$0 = \bar{B}$

Now, we are going to join the outputs of the same truth table by adding (OR) them and the final output is SOP expression: $Output\ X - OR = y = \bar{A} \cdot B + A \cdot \bar{B}$

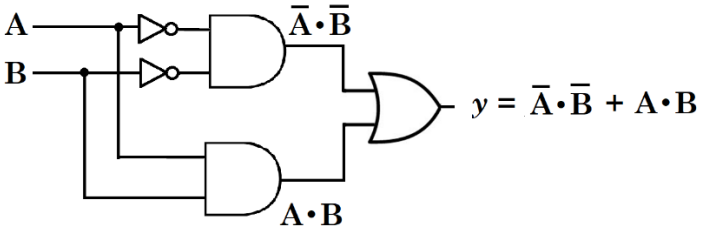
Once we have the SOP expression of the output Y, we can draw the logic circuit:



Following the same procedure, we can find the logic circuit of an X-NOR gate:

2-input X-NOR Gate – Truth Table		
Input		Output
A	B	$Y = \overline{A \oplus B}$
0 = $\bar{A}$ ← $\bar{A} \cdot \bar{B}$	0 = $\bar{B}$	**1**
0	1	**0**
1	0	**0**
1 = A ← $A \cdot B$	1 = A	**1**

$$Output\ X - NOR = y = \bar{A} \cdot \bar{B} + A \cdot B$$



References

Electronics Hub. (2019). Retrieved from Boolean Logic - SOP form, POS form: https://www.electronicshub.org/boolean-logic-sop-form-pos-form/

Wikipedia. (2019, May 1). Retrieved from Combinational Logic: https://en.wikipedia.org/wiki/Combinational_logic