XOR - XNOR Gates

Lesson Objectives:

In addition to AND, OR, NOT, NAND and NOR gates, exclusive-OR (XOR) and exclusive-NOR (XNOR) gates are also used in the design of digital circuits.
These have special functions and applications. These gates are particularly useful in arithmetic operations as well as error-detection and correction circuits.

XOR and XNOR gates are usually found as 2-input gates. No multiple-input XOR/XNOR gates are available since they are complex to fabricate with hardware.

The objectives of this lesson are to learn about:

1. XOR gates and XNOR gates
2. Their properties of operation and basic identities
3. Odd function and Even function
4. Parity generation and checking.

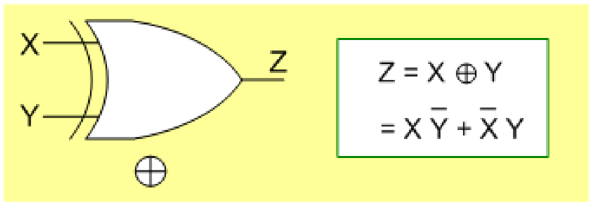
XOR Gate:

The exclusive-OR (XOR), operator uses the symbol ⊕, and it performs the following logic operation:

$X \oplus Y = X Y' + X' Y$

The graphic symbol and truth table of XOR gate is shown in the figure.

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0



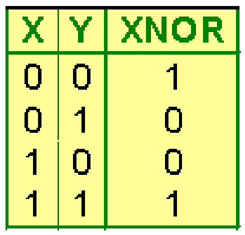
The result is 1 only when either X is equal to 1 or Y is equal to 1, but not when both X and Y are equal to 1.

XNOR Gate:

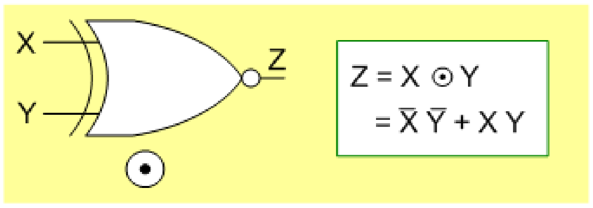
The exclusive-NOR (XNOR), operator uses the symbol ⊙, and it performs the following logic operation

$X \odot Y = X Y + X' Y' = (X \oplus Y)'$

The graphic symbol and truth table of XNOR (Equivalence) gate is shown in the figure.



X	Y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1



The result is 1 when either both X and Y are 0's or when both are 1's. That is why this gate is often referred to as the **Equivalence** gate.

The truth tables clearly show that the exclusive-NOR operation is the complement of the exclusive-OR.

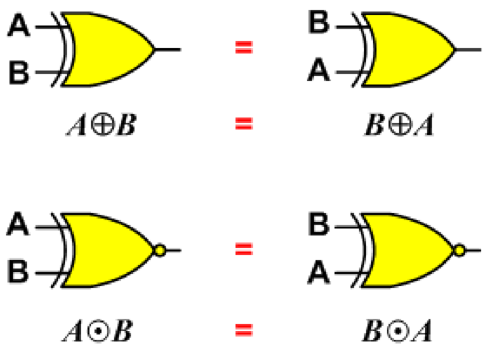
This can also be shown by algebraic manipulation as follows:

$$\begin{aligned}(X \oplus Y)' &= (X Y' + X' Y)' \\ &= (X Y')' (X' Y)' = (X' + Y) (X + Y') \\ &= (XY + X'Y') \\ &= X \odot Y\end{aligned}$$

Properties of XOR/XNOR Operations:

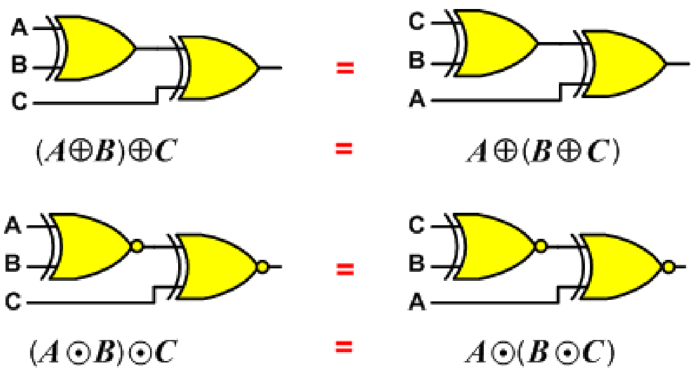
1- Commutativity

- $A \oplus B = B \oplus A$, and
- $A \odot B = B \odot A$



2- Associativity

- $A \oplus (B \oplus C) = (A \oplus B) \oplus C$, and
- $A \odot (B \odot C) = (A \odot B) \odot C$

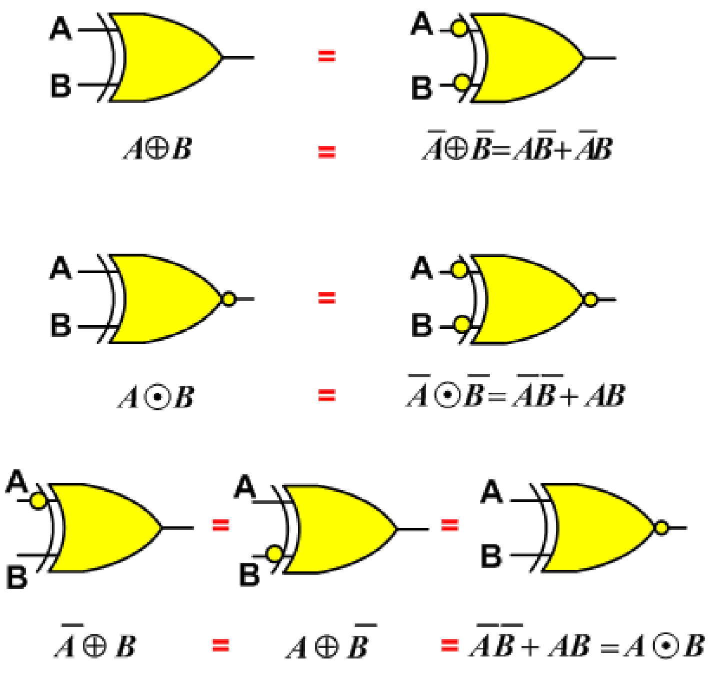

A
B
C
=
C
B
A
$(A\oplus B)\oplus C = A\oplus(B\oplus C)$
A
B
C
=
C
B
A
$(A\odot B)\odot C = A\odot(B\odot C)$

Basic Identities of XOR Operation:

Any of the following identities can be proven using either truth tables or algebraically by replacing the ⊕ operation by its equivalent Boolean expression:

- X ⊕ 0 = X
- X ⊕ 1 = X’
- X ⊕ X = 0
- X ⊕ X’ = 1
- X ⊕ Y’ = X’ ⊕ Y = (X ⊕ Y)’ = X ⊙ Y

The figure provides a graphical presentation of important XOR/XNOR rules and gate equivalence.

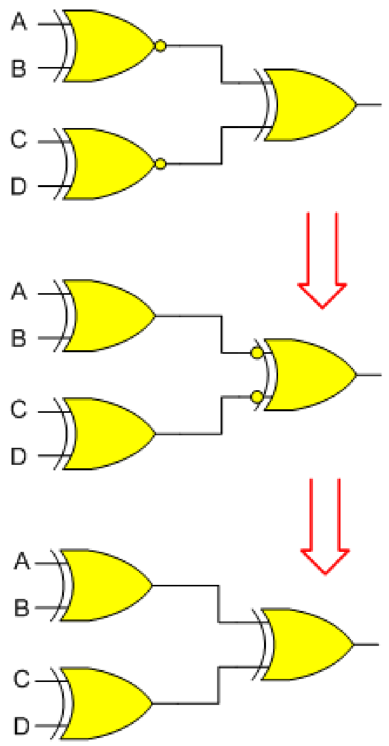

A
B
=
A
B
$A\oplus B = \bar{A}\oplus\bar{B} = A\bar{B}+\bar{A}B$
A
B
=
A
B
$A\odot B = \bar{A}\odot\bar{B} = \bar{A}\bar{B}+AB$
A
B
=
A
B
=
A
B
$\bar{A}\oplus B = A\oplus\bar{B} = \bar{A}\bar{B}+AB = A\odot B$

Example:

Show that **(A ⊙ B) ⊕ (C ⊙ D) = A ⊕ B ⊕ C ⊕ D**

Proving the above identity is easier done using graphical equivalence between gates as specified by the previous figure.

The following figure shows a step-by-step approach starting by the logic circuit corresponding to the left-hand-side of the identity and performing equivalent gate transformations till a circuit is reached that corresponds to the right-hand-side of the identity.



ODD Function:

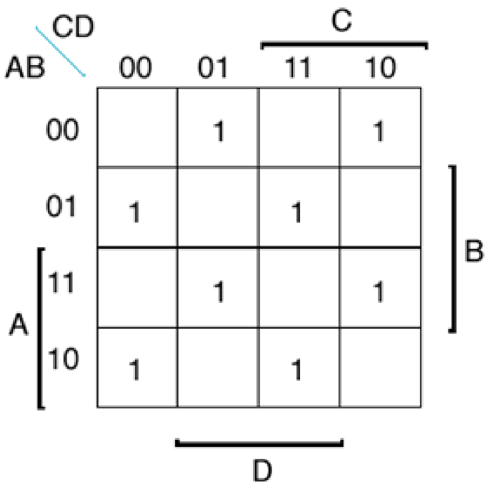
As shown in the K-map, **X ⊕ Y ⊕ Z = 1, IFF** *(if and only if)* the number of 1's in the input combination is ***odd***.

X	Y	Z	ODD
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

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

(a) X ⊕ Y ⊕ Z

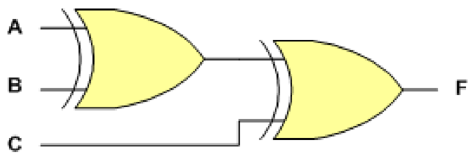
Likewise, **$A \oplus B \oplus C \oplus D = 1$, IFF** the number of 1's in the input combination is ***odd***.



(b) $A \oplus B \oplus C \oplus D$

In general, an exclusive-OR function of n-variables is an *odd function* which has a value of 1 **IFF** the number of 1's in the input combination is ***odd***, otherwise it has a value of 0.

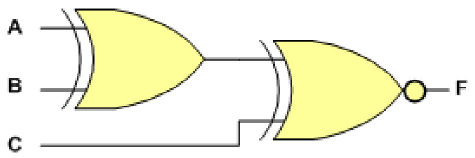
Since XOR gates are only designed with 2 inputs, the 3-input XOR function is implemented by means of two 2-input XOR gates, as shown in figure.



EVEN Function:

The complement of an odd function is an ***even function***. The even function is equal to 1 when the number of 1's in the input combination is even.

The complement of an odd function (an ***even function***) is obtained by replacing the output gate with an exclusive-NOR gate, as shown in figure.



Parity Generation and Checking:

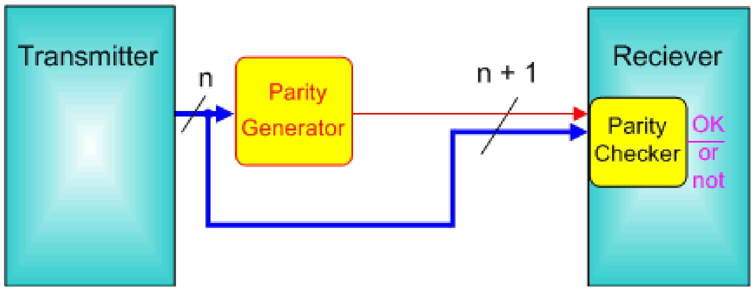
Exclusive-OR functions are very useful in systems using parity bits for error-detection.

A parity bit is used for the purpose of detecting errors during transmission of binary information.

A parity bit is an extra bit included with a binary message to make the **total number of 1's** in this message (including the parity bit) either **odd** or **even**.

The message, including the parity bit, is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted.

The circuit that generates the parity bit at the transmitter side is called a *parity generator*. The circuit that checks the parity at the receiver side is called a *parity checker.*



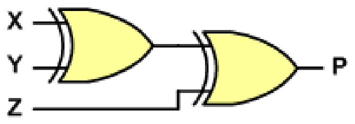
As an example, consider a 3-bit message to be transmitted together with an even parity bit. The table shows the ***truth table for the even parity generator***.

Three-Bit Message			Parity Bit
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

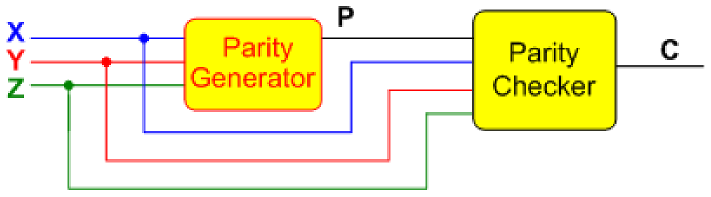
The three bits, **X, Y,** and **Z,** constitute the message and are the inputs to the *even parity generator* circuit whose output is the parity bit ***P***.

For even parity, whenever the message bits (**X, Y& Z**) have an odd number of 1's, the parity bit *P* must be 1. Otherwise, *P* must be 0.
Therefore, *P* can be expressed as a three-variable exclusive-OR function: $\mathbf{P = X \oplus Y \oplus Z}$

The logic diagram for the even parity generator circuit is shown in the figure.



The 4 bits (**X, Y, Z &** ***P***) are *transmitted* to their destination, where they are applied to a *parity-checker circuit* to check for possible errors in the transmission.



Since the information was transmitted with even parity, the received four bits must have an even number of 1's.

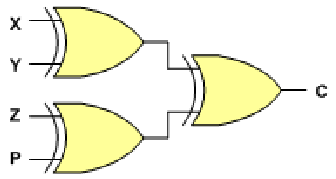
The parity checker generates an error signal (***C = 1***), whenever the received four bits have an odd number of 1's.

The table below shows the ***truth table for the even-parity checker.***

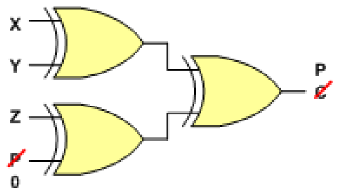
Four Bits Received				Parity Error Check
X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Obviously, the parity checker error output signal ***C*** is given by the following expression:
$C = X \oplus Y \oplus Z \oplus P$

The logic diagram of the even-parity checker is shown in the figure.



It is worth noting that the parity generator can also be implemented with the circuit of this figure if the input *P* is connected to logic-0 and the output is marked with P. This is because Z ⊕ 0 = Z, causing the value of *Z* to pass through the gate unchanged.



The advantage of this is that the same circuit can be used for both parity generation and checking.