Digital Logic & Digital Systems

4004CEM

Computer Architecture & Networks

Introduction to Logic Circuit Design

- Truth tables
- Boolean expressions
- Symbols of logic operators
- Basic logic gates
- Proof using truth tables
- Logic circuits and transmission formulae, equivalent circuits
- Standard results
- NAND and NOR gates
- XOR and XNOR gates

BITS: Logic 1 or Logic 0 (+5volts or 0volts)

Use Logic Circuits to store/manipulate them**.**

Boolean expressions to design and describe the circuits**.**

Example: Derive the logical expression for the output P so that P is 1 (true) if any one of the inputs A,B,C is true or all of the inputs are true.

Truth Table

required or actual output down the RHS for **every possible input combination**

A	B	C	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	*

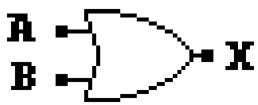
Boolean Expressions

P = 1	**if**	A=0 **AND** B=0 **AND** C=1
	OR	A=0 **AND** B=1 **AND** C=0
	OR	A=1 **AND** B=0 **AND** C=0
	OR	A=1 **AND** B=1 **AND** C=1

If the variable value is 0 the variable is written with a **bar over it or with an exclamation mark before it**.

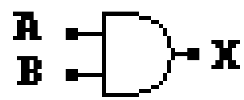
$$\mathbf{P} = \overline{\mathbf{A}}.\overline{\mathbf{B}}.\mathbf{C} + \overline{\mathbf{A}}.\mathbf{B}.\overline{\mathbf{C}} + \mathbf{A}.\overline{\mathbf{B}}.\overline{\mathbf{C}} + \mathbf{A}.\mathbf{B}.\mathbf{C}$$

Basic Gates



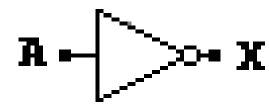
OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1



AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1



NOT

A	X
0	1
1	0

A I

For OR: the output is high if **either A OR B** is high

A B

I I

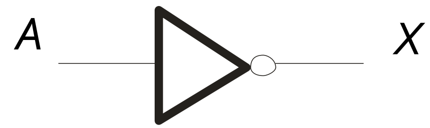
For AND: the output is high only if **A AND B** is high

For NOT: the output is **the inverse** of the input i.e. it is **NOT** the input

I >O

O>I

The Inverter

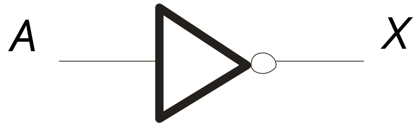


The inverter performs the Boolean **NOT** operation. When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW.

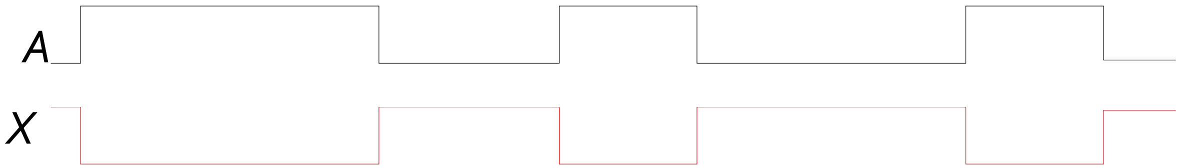
Input	Output
A	*X*
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

The **NOT** operation (complement) is shown with an overbar. Thus, the Boolean expression for an inverter is $X = \overline{A}$.

The Inverter

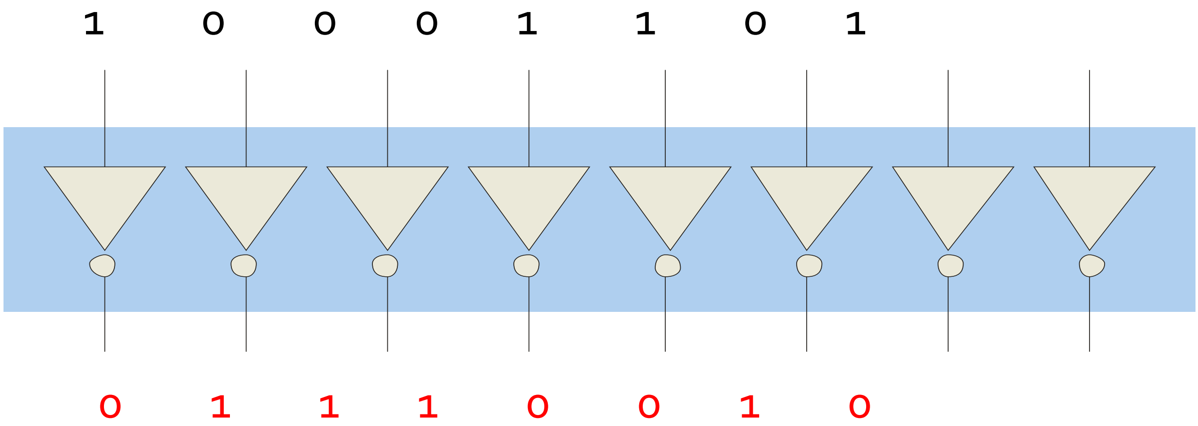


Example waveforms:



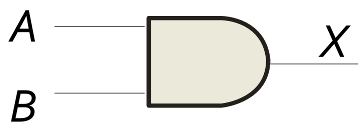
A group of inverters can be used to form the 1's complement of a binary number:

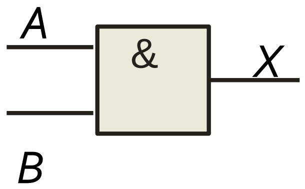
Binary number



1's complement

The AND Gate



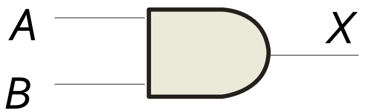


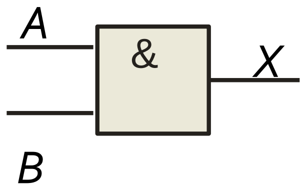
The **AND gate** produces a HIGH output when all inputs are HIGH; otherwise, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

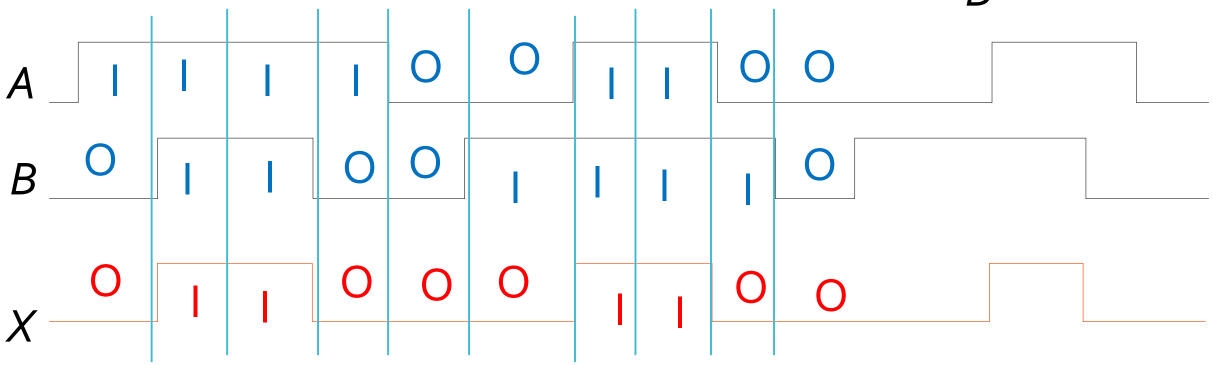
The **AND** operation is usually shown with a dot between the variables but it may be implied (no dot). Thus, the AND operation is written as $X = A \cdot B$ or $X = AB$.

The AND Gate





Example waveforms:



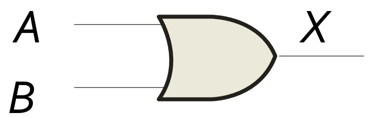
The AND operation is used in computer programming as a selective mask. If you want to retain certain bits of a binary number but reset the other bits to 0, you could set a mask with 1's in the position of the retained bits.

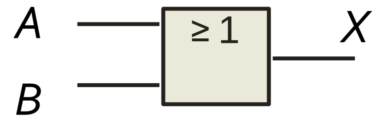
Example

If the binary number 10100011 is ANDed with the mask 00001111, what is the result?

00000011

The OR Gate





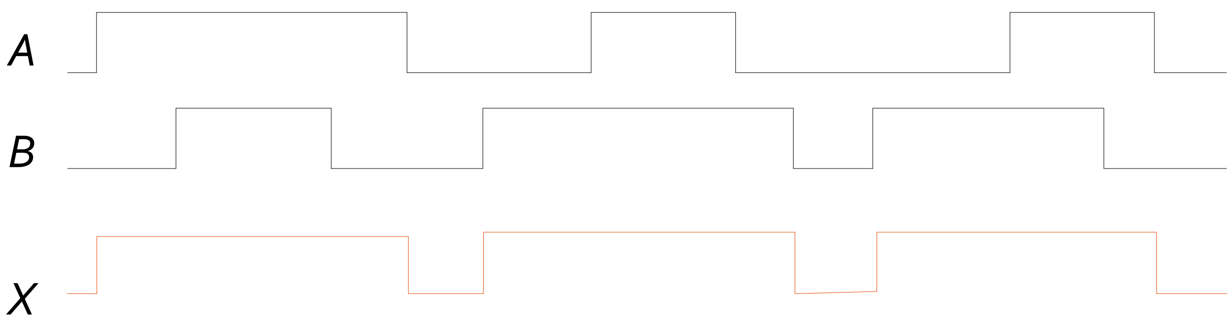
The **OR gate** produces a HIGH output if any input is HIGH; if all inputs are LOW, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

The **OR** operation is shown with a plus sign (+) between the variables. Thus, the OR operation is written as $X = A + B$.

The OR Gate

Example waveforms:

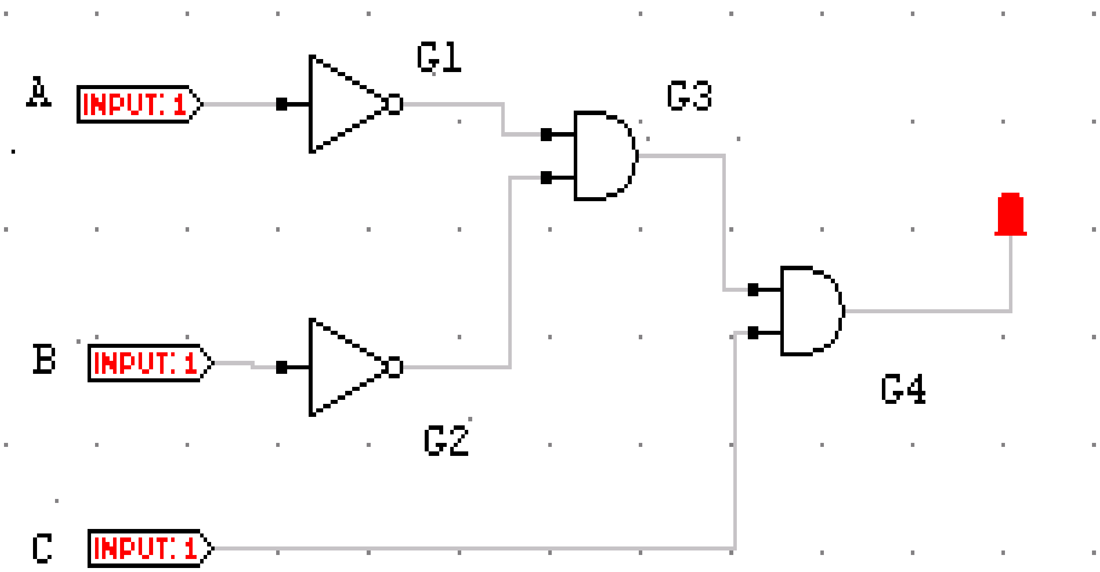


The OR operation can be used in computer programming to set certain bits of a binary number to 1.

Example ASCII letters have a 1 in the bit 5 position for lower case letters and a 0 in this position for capitals. (Bit positions are numbered from right to left starting with 0.) What will be the result if you OR an ASCII letter with the 8-bit mask 00100000?

Solution The resulting letter will be lower case.

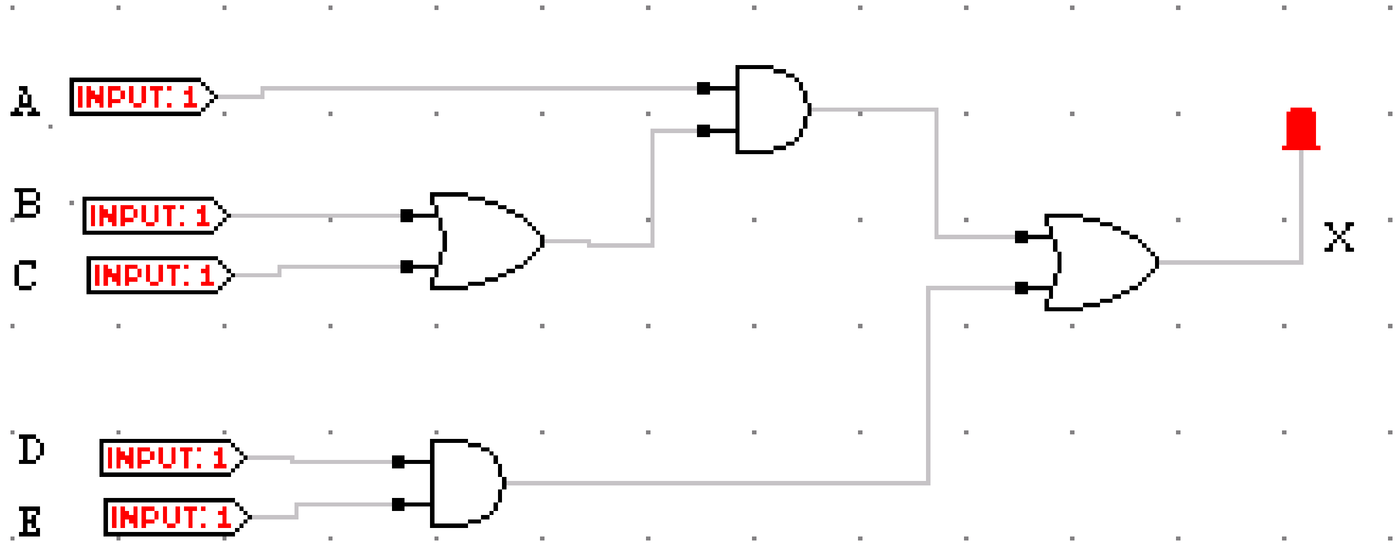
Proof Using Truth Tables



A	B	C	G1	G2	G3	G4
0	0	0	1	1	1	0
0	0	1	1	1	1	1
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0

Drawing Logic Circuits from Transmission Formulae

$$X = A.(B + C) + D.E$$



Equivalent Circuits

Complicated logic circuits can often be reduced to much simpler 'equivalent' circuits

Truth tables can be used to test equivalence

$$\mathbf{X} = \mathbf{A} + \mathbf{B.C} \quad \text{and} \quad \mathbf{Y} = (\mathbf{A} + \mathbf{B}).(\mathbf{A} + \mathbf{C})$$

A	B	C	BC	X	A+B	A+C	Y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

we are able to conclude that:

$$\mathbf{A} + \mathbf{B.C} = (\mathbf{A} + \mathbf{B}).(\mathbf{A} + \mathbf{C})$$

Standard Boolean Algebra rules:

Commutative Laws

A.B = B.A   A+B = B+A

Associative Laws

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

Distributive law

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

some simple obvious(!) results:

$$\mathbf{A}.\mathbf{A} = \mathbf{A}$$

$$\mathbf{A} + \mathbf{A} = \mathbf{A}$$

$$\mathbf{A}.\overline{\mathbf{A}} = 0$$

$$\mathbf{A} + \overline{\mathbf{A}} = 1$$

$$\mathbf{A}.1 = \mathbf{A}$$

$$\mathbf{A} + 1 = 1$$

$$\mathbf{A}.0 = 0$$

$$\mathbf{A} + 0 = \mathbf{A}$$

$$\overline{\overline{\mathbf{A}}} = \mathbf{A}$$

DeMorgan's Laws

Two further (less obvious!) results:

$$\overline{(\mathbf{A}+\mathbf{B})}=\overline{\mathbf{A}}.\overline{\mathbf{B}}$$

and

$$\overline{(\mathbf{A}.\mathbf{B})}=\overline{\mathbf{A}}+\overline{\mathbf{B}}$$

Note how the application of the inverse operator changes AND to OR and vice versa.

Quick method: to change an OR to an AND or vice versa, i) break the line ii) change the sign.

Simplifying Circuits

Example 1: To show that (A +B)(A + C) = A + B.C

Simplifying Circuits

Example 1: To show that (A +B)(A + C) = A + B.C

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

$$A.A+A.C+B.A+B.C=A.1+A.C+B.A+B.C$$

$$A.1+A.C+B.A+B.C=A.(1+C+B)+B.C$$

$$A.(1+C+B)+B.C=A.1+B.C$$

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

Example 2: To simplify $\mathbf{XY} + \mathbf{X\overline{Y}}$

Example 2: To simplify $\mathbf{XY} + \mathbf{X}\overline{\mathbf{Y}}$

$$\mathbf{XY} + \mathbf{X}\overline{\mathbf{Y}} = \mathbf{X}.(\mathbf{Y} + \overline{\mathbf{Y}})$$

$$\mathbf{XY} + \mathbf{X}\overline{\mathbf{Y}} = \mathbf{X}.(\mathbf{Y} + \overline{\mathbf{Y}}) = \mathbf{X}.1$$

$$\mathbf{XY} + \mathbf{X}\overline{\mathbf{Y}} = \mathbf{X}.1 = \mathbf{X}$$

Example 3:

To simplify the expression $(A + B).(A + C).(A + D)$

From example 1 $(A + B).(A + C) = (A + B.C)$

$$(A + B).(A + C).(A + D) = (A + B.C).(A + D)$$

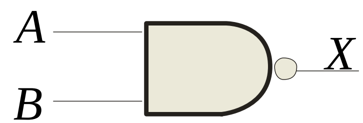
$$(A + B.C).(A + D) = A.A + A.D + B.C.A + B.C.D$$

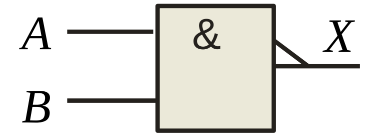
$$(A + B.C).(A + D) = A.(1 + D + B.C) + B.C.D$$

$$(A + B.C).(A + D) = A.1 + B.C.D$$

So $(A + B)(A + C)(A + D) = A + B.C.D$

The NAND Gate



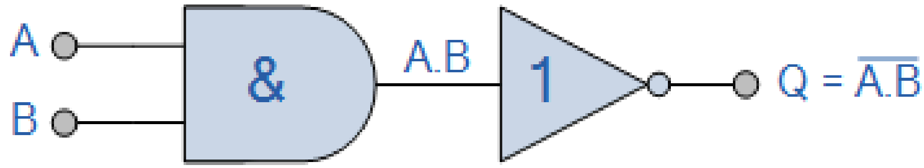


The **NAND gate** produces a LOW output when all inputs are HIGH; otherwise, the output is HIGH. For a 2-input gate, the truth table is

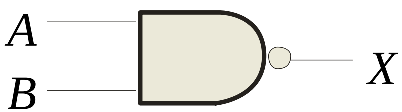
Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

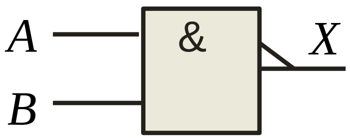
The **NAND** operation is shown with a dot between the variables and an overbar covering them. Thus, the NAND operation is written as $X = \overline{A \cdot B}$ (Alternatively, $X = \overline{AB}$.)

The NAND Gate

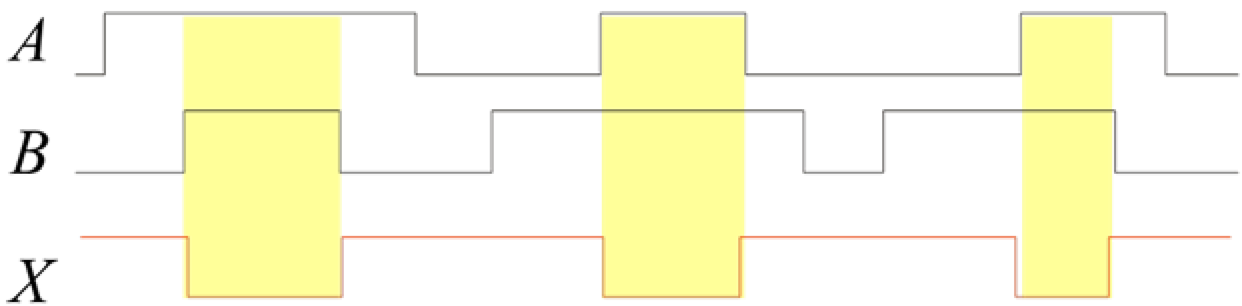


2-input "AND" gate plus a "NOT" gate

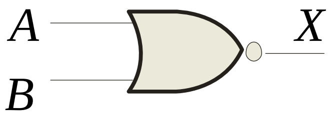


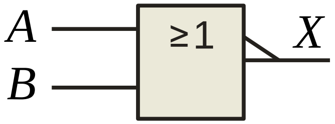


Example waveforms:



The NOR Gate



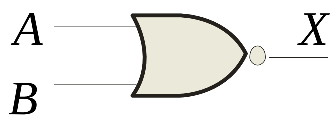


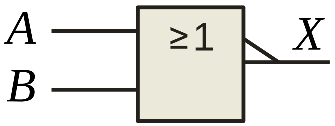
The **NOR gate** produces a LOW output if any input is HIGH; if all inputs are HIGH, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

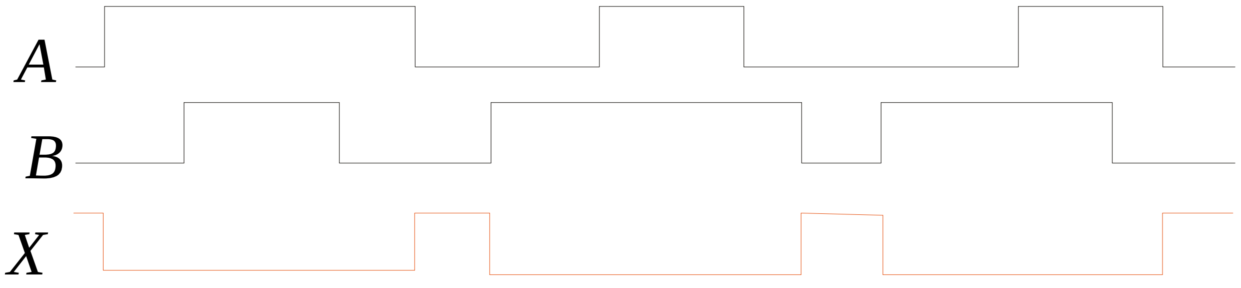
The **NOR** operation is shown with a plus sign (+) between the variables and an overbar covering them. Thus, the NOR operation is written as $X = \overline{A + B}$.

The NOR Gate





Example waveforms:

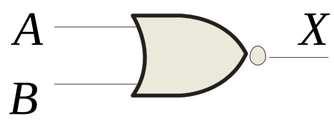


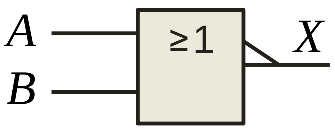
The NOR operation will produce a LOW if any input is HIGH.

Example When is the LED is ON for the circuit shown?

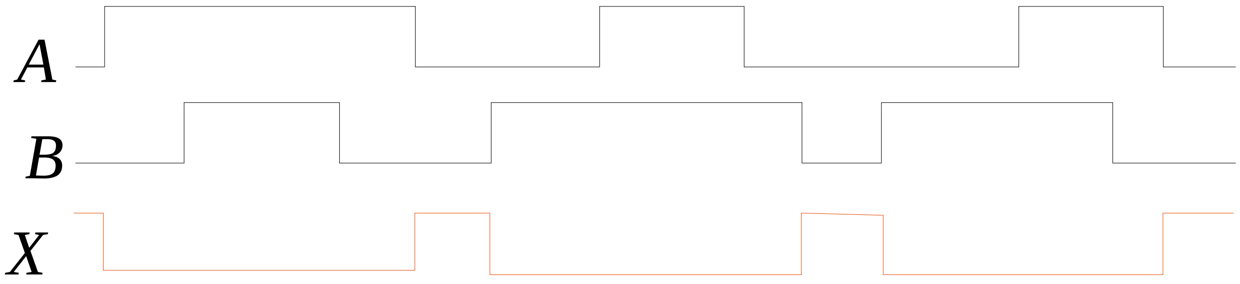
Solution

The NOR Gate



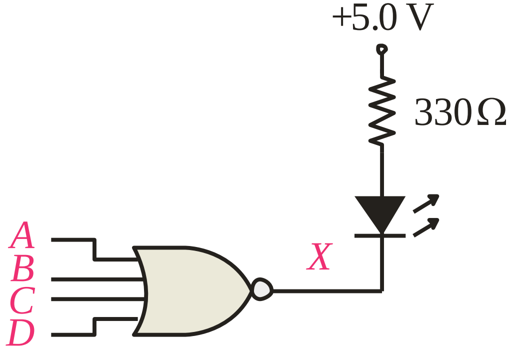


Example waveforms:



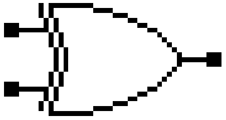
The NOR operation will produce a LOW if any input is HIGH.

Example When is the LED is ON for the circuit shown?



Solution The LED will be on when none of the four inputs are HIGH.

The XOR function



XOR

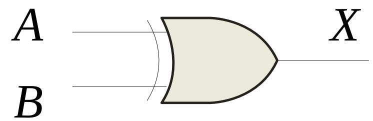
$$A \oplus B = A.\overline{B} + \overline{A}.B$$

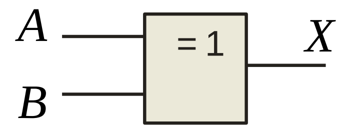
The truth table:

XOR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

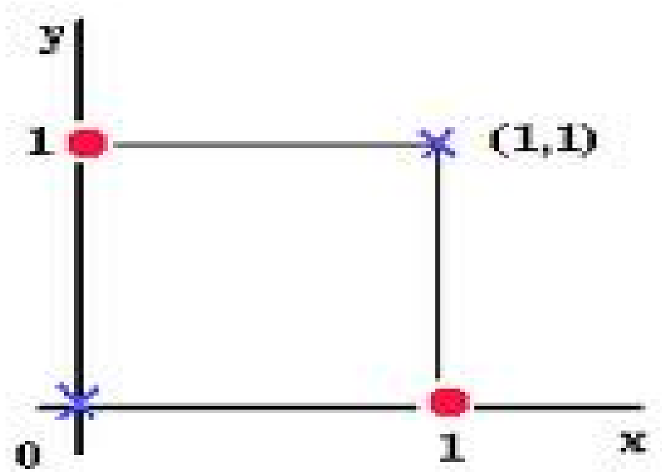
The XOR Gate





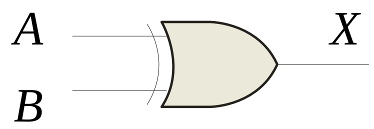
The **XOR gate** produces a HIGH output only when both inputs are at opposite logic levels. The truth table is

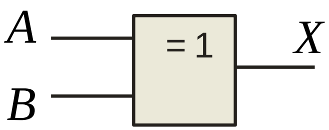
Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



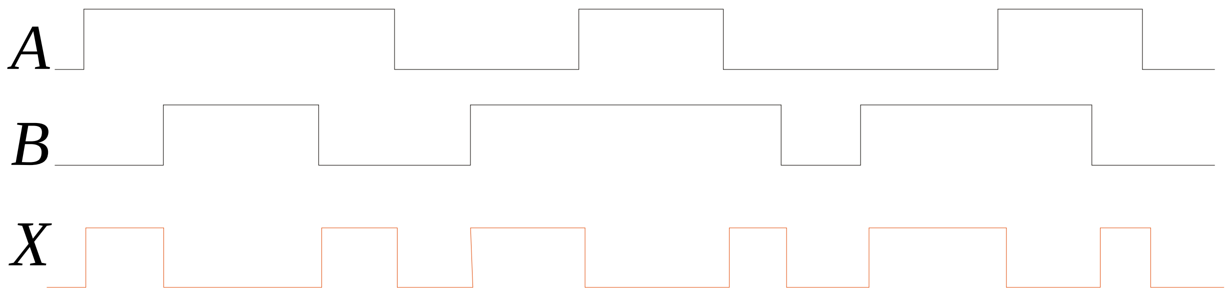
The **XOR** operation is written as $X = \overline{A}B + A\overline{B}$.
Alternatively, it can be written with a circled plus sign between the variables as $X = A \oplus B$.

The XOR Gate





Example waveforms:

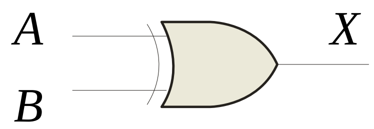


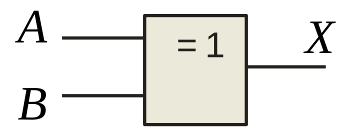
Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

Question

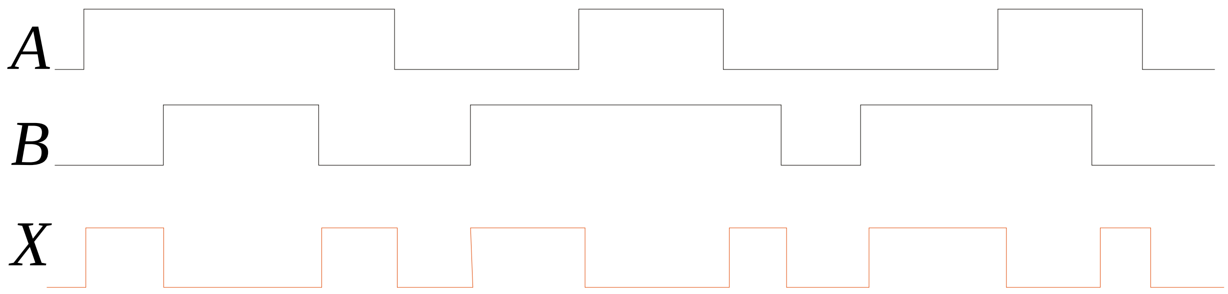
If the *A* and *B* waveforms are both inverted for the above waveforms, how is the output affected?

The XOR Gate





Example waveforms:



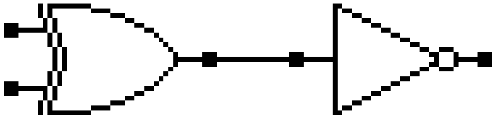
Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

Question

If the *A* and *B* waveforms are both inverted for the above waveforms, how is the output affected?

There is no change in the output.

The inverse of the XOR function (XNOR)



XNOR

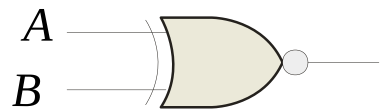
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

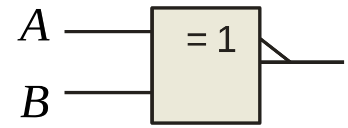
$\overline{A}.\overline{B} + A.B$

From the above truth table:

$$\overline{(\mathbf{A} \oplus \mathbf{B})} = \overline{\mathbf{A}}.\overline{\mathbf{B}} + \mathbf{A}.\mathbf{B}$$

The XNOR Gate



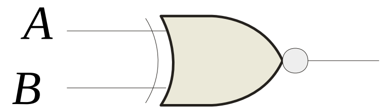


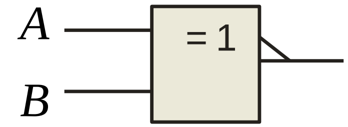
The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level. The truth table is

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

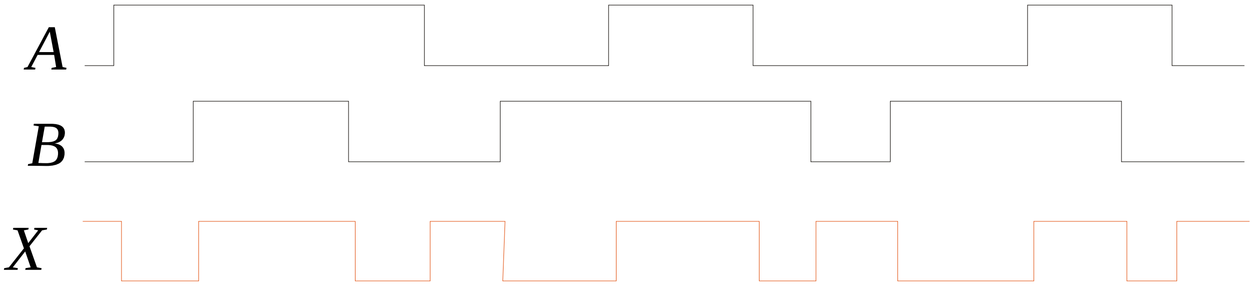
The **XNOR** operation shown as $X = \bar{A}\bar{B} + AB$. Alternatively, the XNOR operation can be shown with a circled dot between the variables. Thus, it can be shown as $X = A \odot B$.

The XNOR Gate





Example waveforms:

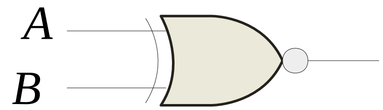


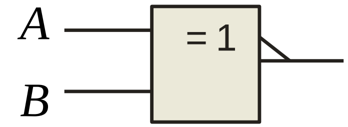
Notice that the XNOR gate will produce a HIGH when both inputs are the same. This makes it useful for comparison functions.

Question

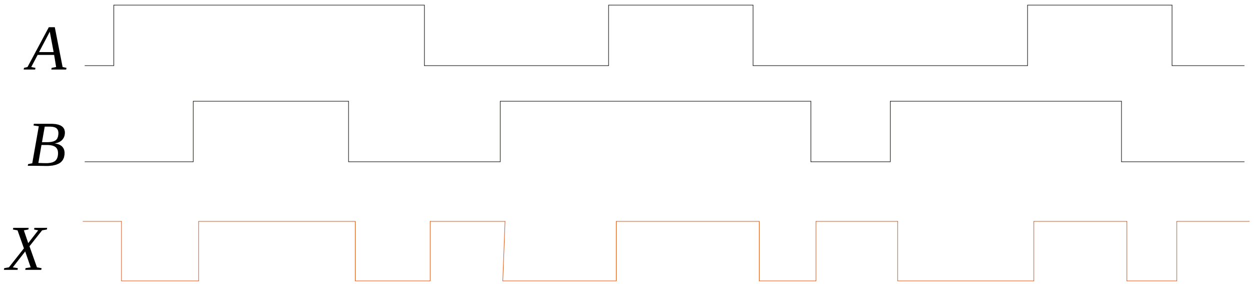
If the *A* waveform is inverted but *B* remains the same, how is the output affected?

The XNOR Gate





Example waveforms:



Notice that the XNOR gate will produce a HIGH when both inputs are the same. This makes it useful for comparison functions.

Question

If the *A* waveform is inverted but *B* remains the same, how is the output affected?

The output will be inverted.

END