Tishk International University
Science Faculty
IT Department

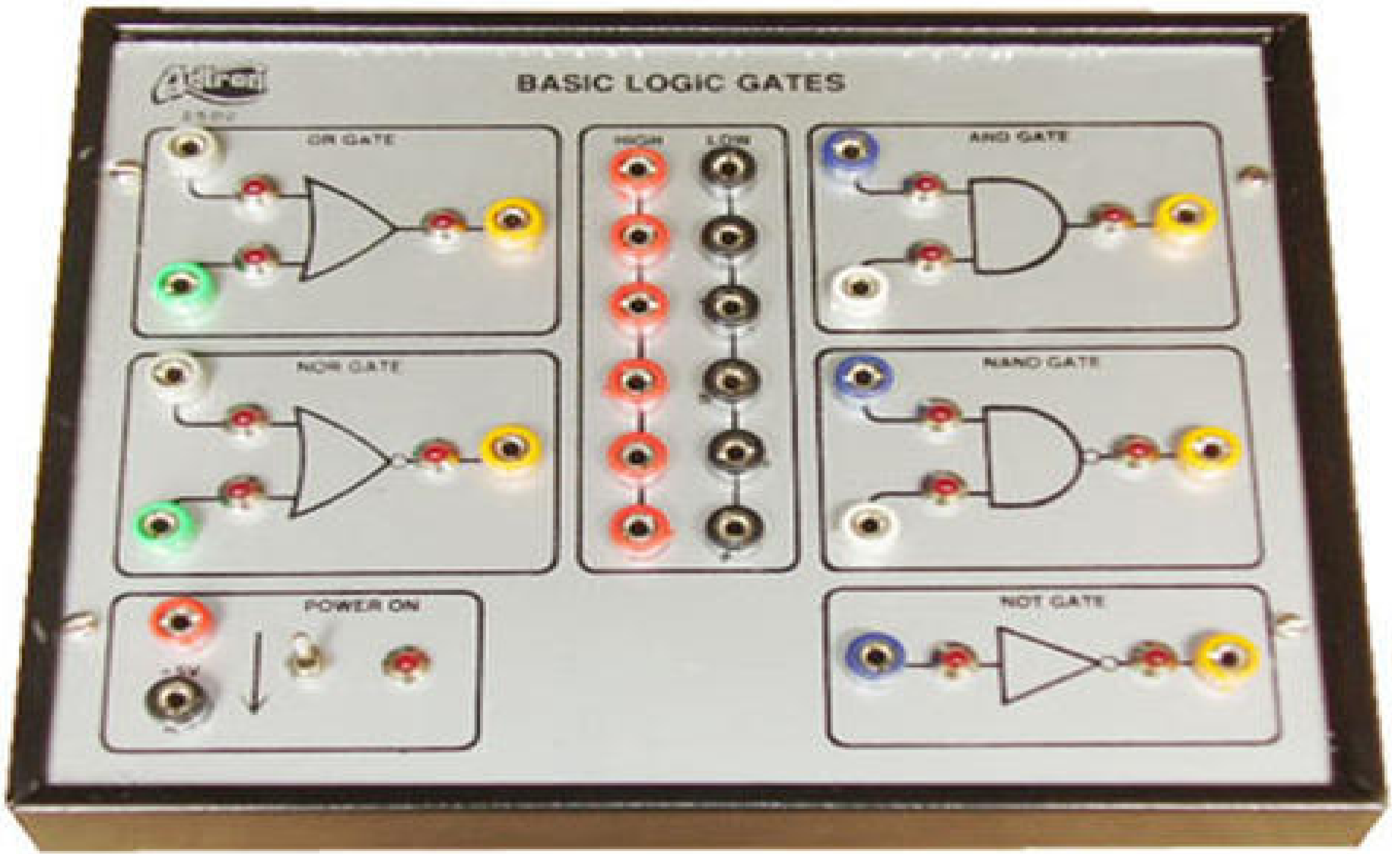
Logic Design

Lecture 02: Logic Gates and Boolean Algebra

2nd Grade –Fall Semester 2021-2022

Instructor: Alaa Ghazi

Lecture 2
Logic Gates and Boolean Algebra



Key Terms

Inverter A logic circuit that inverts or complements its inputs.

Truth table A table showing the inputs and corresponding output(s) of a logic circuit.

Timing diagram A diagram of waveforms showing the proper time relationship of all of the waveforms.

Boolean algebra The mathematics of logic circuits.

AND gate A logic gate that produces a HIGH output only when all of its inputs are HIGH.

Key Terms

OR gate A logic gate that produces a HIGH output when one or more inputs are HIGH.

NAND gate A logic gate that produces a LOW output only when all of its inputs are HIGH.

NOR gate A logic gate that produces a LOW output when one or more inputs are HIGH.

Exclusive-OR gate A logic gate that produces a HIGH output only when its two inputs are at opposite levels.

Exclusive-NOR gate A logic gate that produces a LOW output only when its two inputs are at opposite levels.

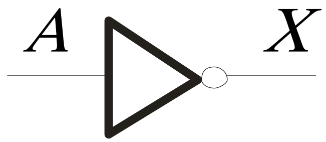
Binary Digits, Logic Levels, and Digital Waveforms

- The two binary digits are designated **0** and **1**
- They can also be called LOW and HIGH, where **LOW = 0** and **HIGH = 1**
- In order to practice with Logic Gates we can use:
  - **LogicCircuit**
  - **CEDAR Logic Simulator**
  - **Logisim**

Logic Gates

- Inverter
- AND Gate
- OR Gate
- NAND Gate
- NOR Gate
- Exclusive-OR Gate
- Exclusive-NOR Gate

The Inverter

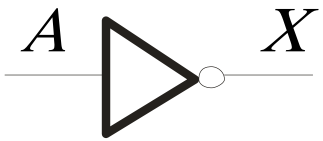


The inverter performs the Boolean **NOT** operation. When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW.

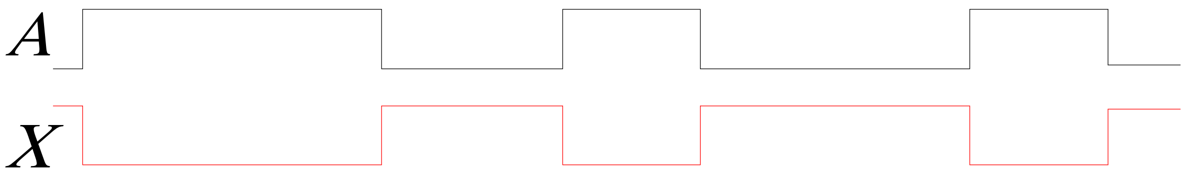
Input	Output
A	*X*
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

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

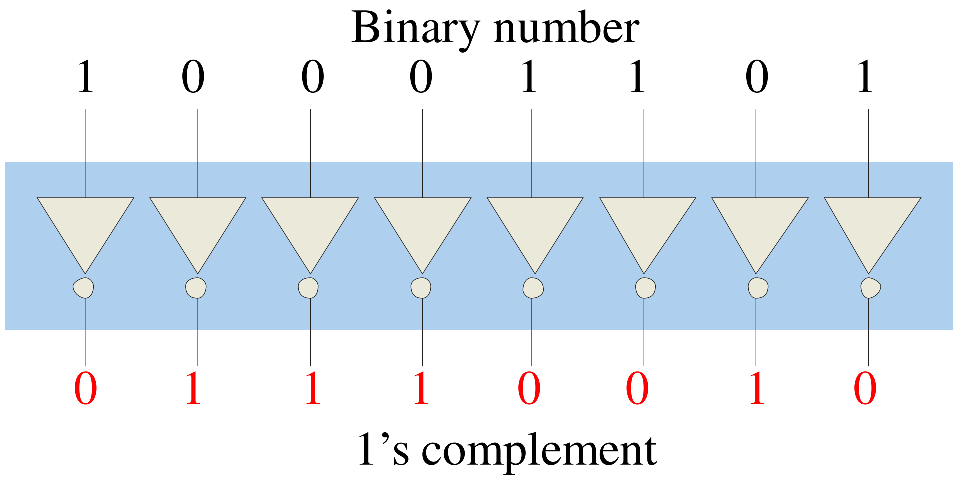
The Inverter



Example waveforms:



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



Truth Tables

- Total number of possible combinations of binary inputs

$$N = 2^n$$

- For two input variables:

$$N = 2^2 = 4 \text{ combinations}$$

- For three input variables:

$$N = 2^3 = 8 \text{ combinations}$$

- For four input variables:

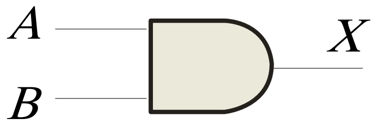
$$N = 2^4 = 16 \text{ combinations}$$

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

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

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The AND Gate

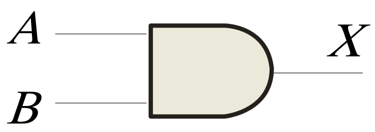


The **AND gate** produces a HIGH output when all inputs are HIGH; otherwise, the output is LOW. For a 2-input gate, the truth table is

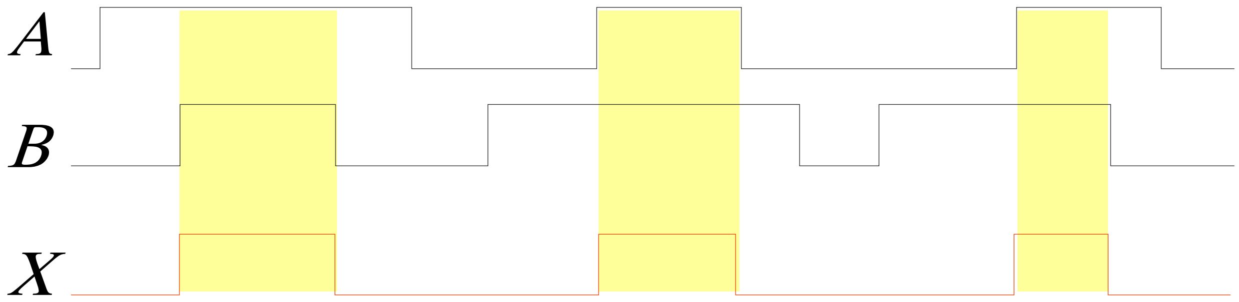
Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

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



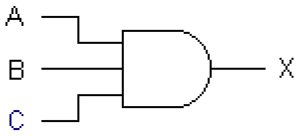
Example waveforms:



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.

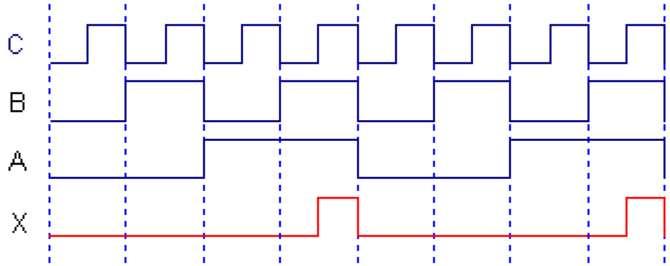
Example If the binary number 10100011 is ANDed with the mask 00001111, what is the result? 00000011

The AND Gate for more than 2 inputs

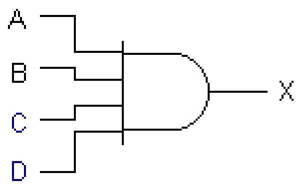


X = ABC

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

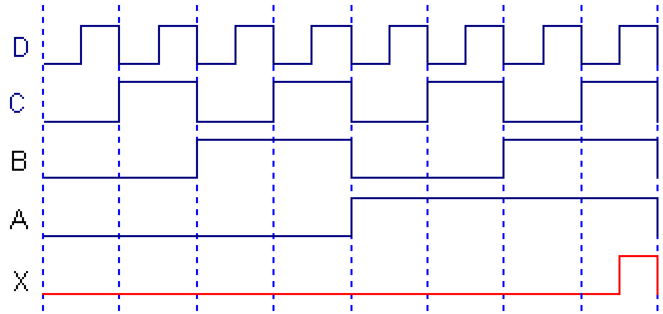


3-Input AND Gate



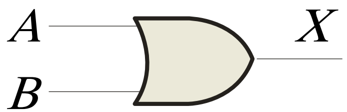
X = ABCD

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



4-Input AND Gate

The OR Gate

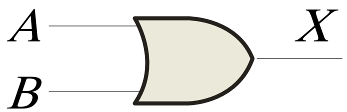


The **OR gate** produces a HIGH output if any input is HIGH; if all inputs are LOW, the output is LOW. For a 2-input gate, the truth table is

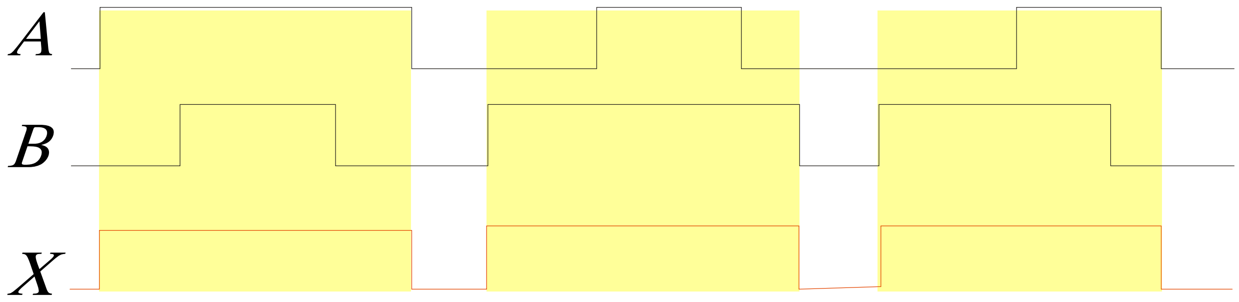
Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

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:

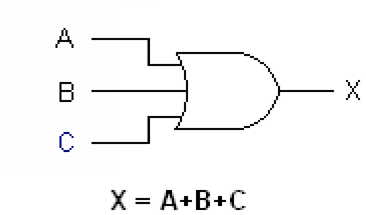


The OR operation can be used in computer programming to set certain bits of a binary number to 1.

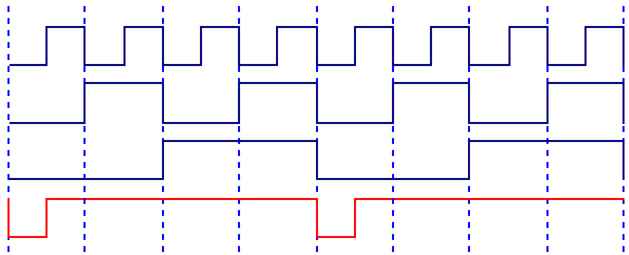
Example ASCII letters have a 1 in the bit 5 position for lower case letters and a 0 in this position for capitals. (Bit positions are numbered from right to left starting with 0.) What will be the result if you OR an ASCII letter with the 8-bit mask 00100000?

Solution The resulting letter will be lower case.

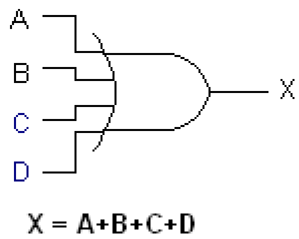
The OR Gate for more than 2 inputs



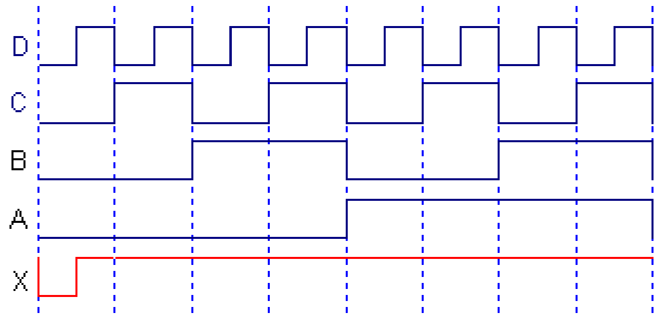
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



3-Input OR Gate

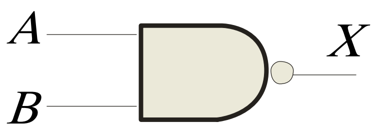


A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



4-Input OR Gate

The NAND Gate

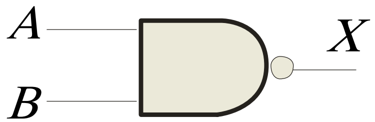


The **NAND gate** produces a LOW output when all inputs are HIGH; otherwise, the output is HIGH. For a 2-input gate, the truth table is

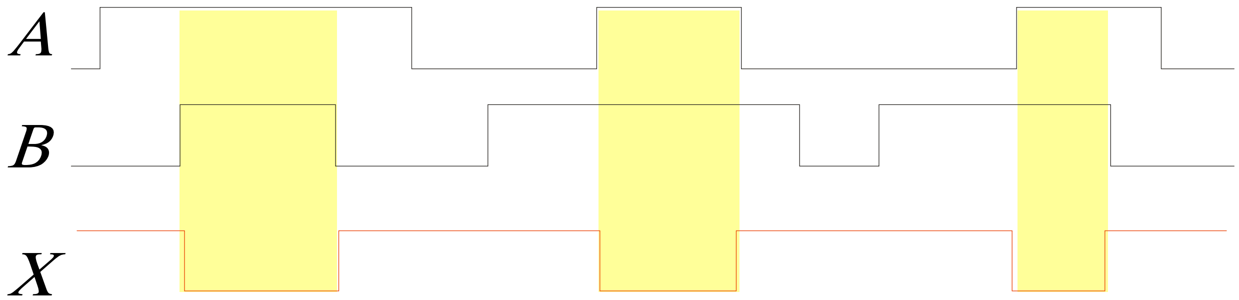
Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

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

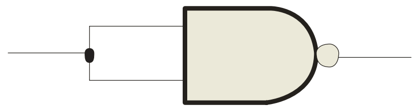


Example waveforms:

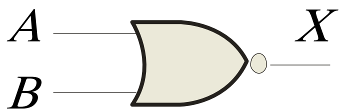


The NAND gate is particularly useful because it is a "universal" gate – all other basic gates can be constructed from NAND gates.

Question How would you connect a 2-input NAND gate to form a basic inverter?



The NOR Gate

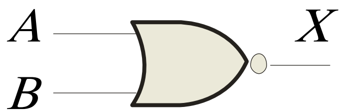


The **NOR gate** produces a LOW output if any input is HIGH; if all inputs are HIGH, the output is LOW. For a 2-input gate, the truth table is

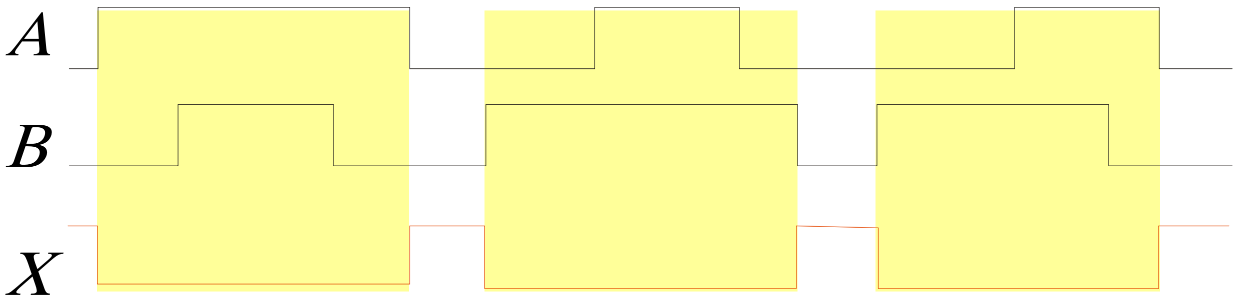
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

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



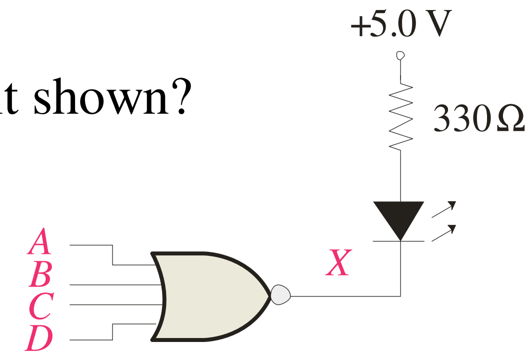
Example waveforms:



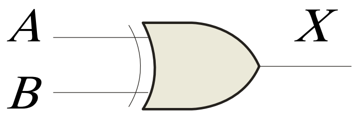
The NOR operation will produce a LOW if any input is HIGH.

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

Solution The LED will be on when any of the four inputs are HIGH.



The XOR Gate

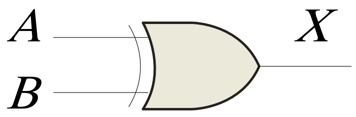


The **XOR gate** produces a HIGH output only when both inputs are at opposite logic levels. The truth table is

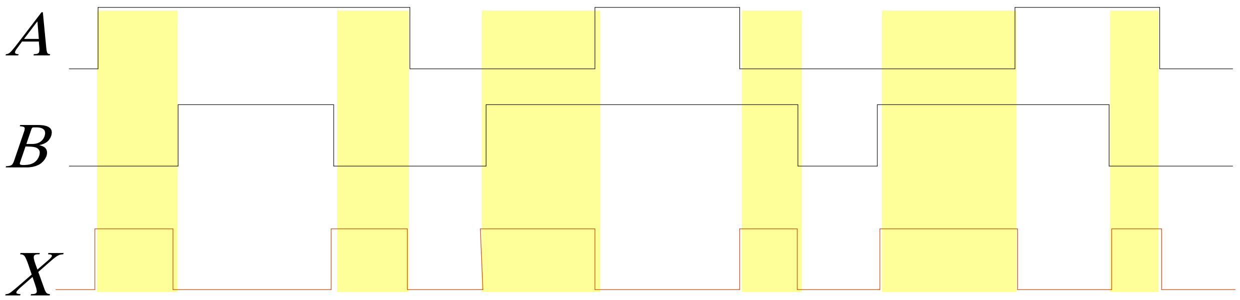
Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

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:

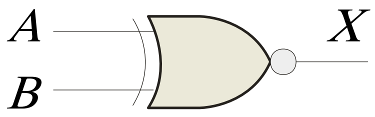


Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

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

There is no change in the output.

The XNOR Gate

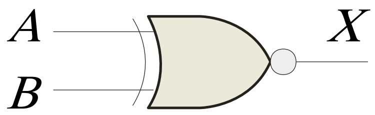


The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level. The truth table is

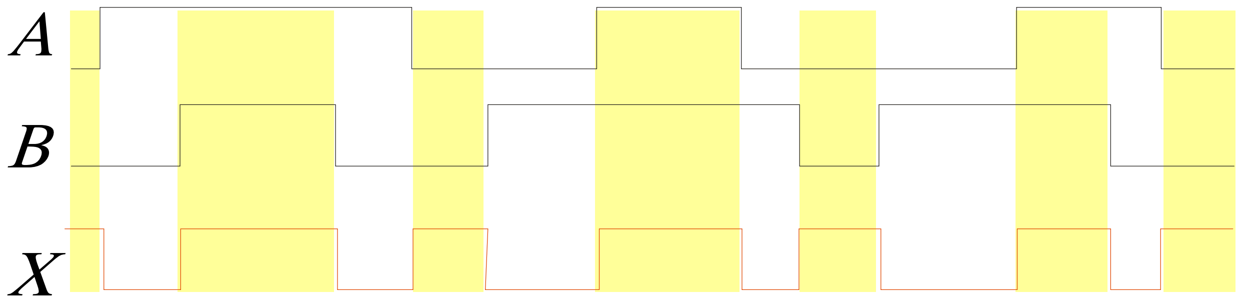
Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

The **XNOR** operation shown as $X = \overline{A}\overline{B} + AB$. 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 output will be inverted.

Boolean Operations and Expressions

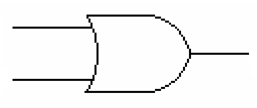
- Addition

  0 + 0 = 0

  0 + 1 = 1

  1 + 0 = 1

  1 + 1 = 1



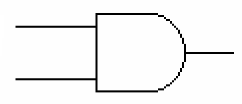
- Multiplication

  0 * 0 = 0

  0 * 1 = 0

  1 * 0 = 0

  1 * 1 = 1



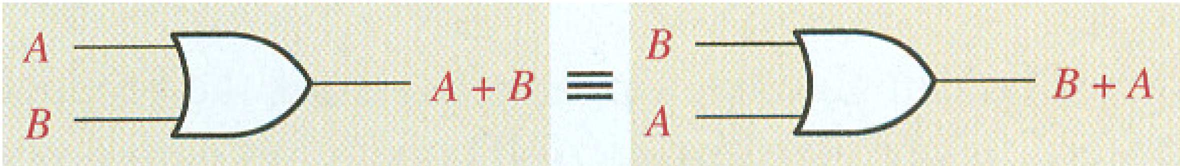
Laws Boolean Algebra

- Commutative Laws
- Associative Laws
- Distributive Law

Laws of Boolean Algebra

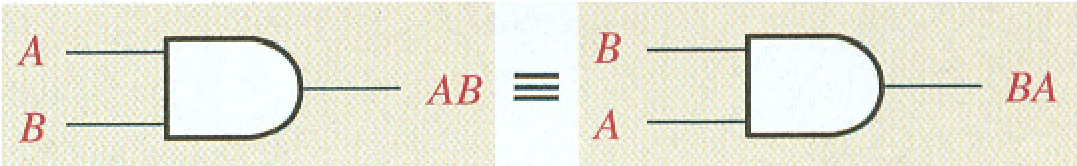
- Commutative Law of Addition:

$$A + B = B + A$$



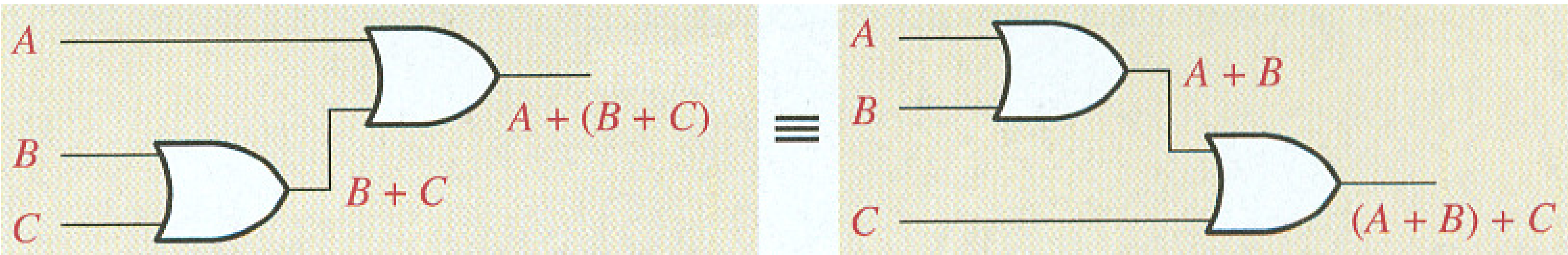
- Commutative Law of Multiplication:

A * B = B * A



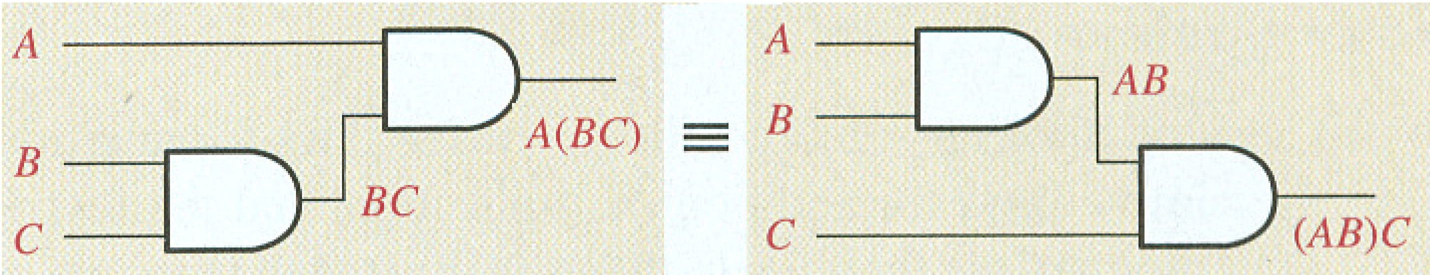
- Associative Law of Addition:

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



- Associative Law of Multiplication:

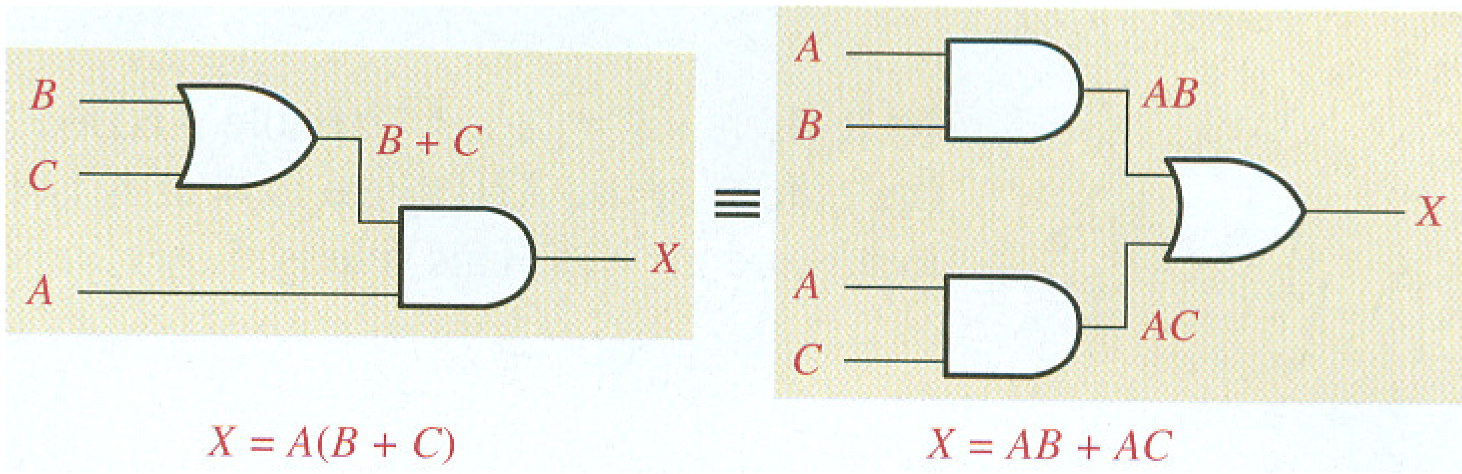
A * (B * C) = (A * B) * C



Laws of Boolean Algebra

- Distributive Law:

A(B + C) = AB + AC

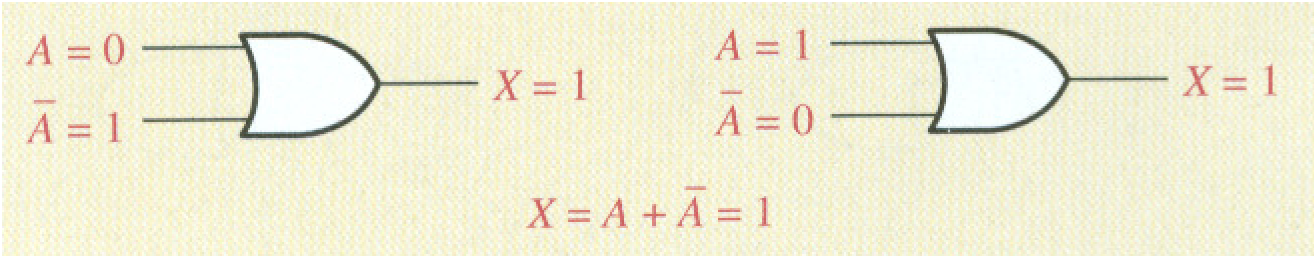


Rules of Boolean Algebra

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

Rules of Boolean Algebra

- Rule 6

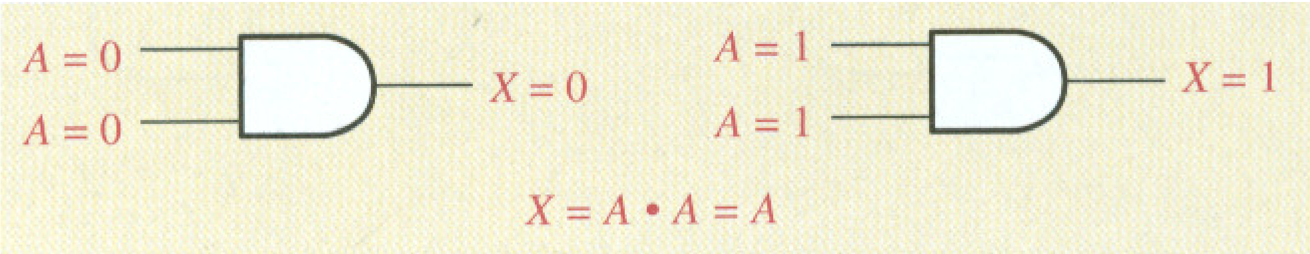


A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

OR Truth Table

Rules of Boolean Algebra

- Rule 7

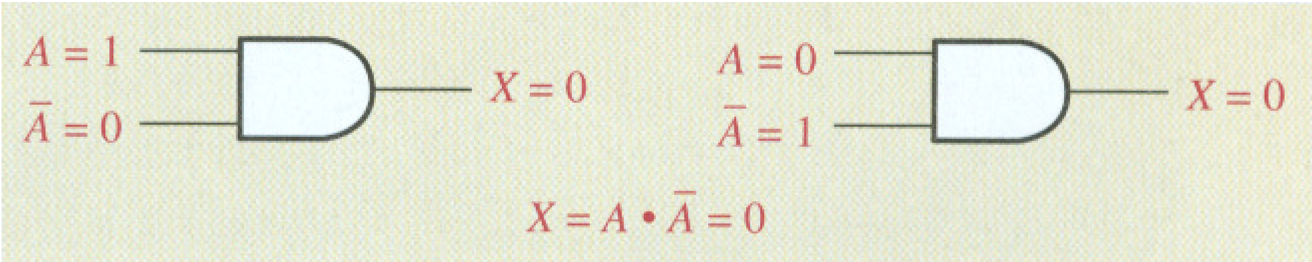


A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

AND Truth Table

Rules of Boolean Algebra

- Rule 8

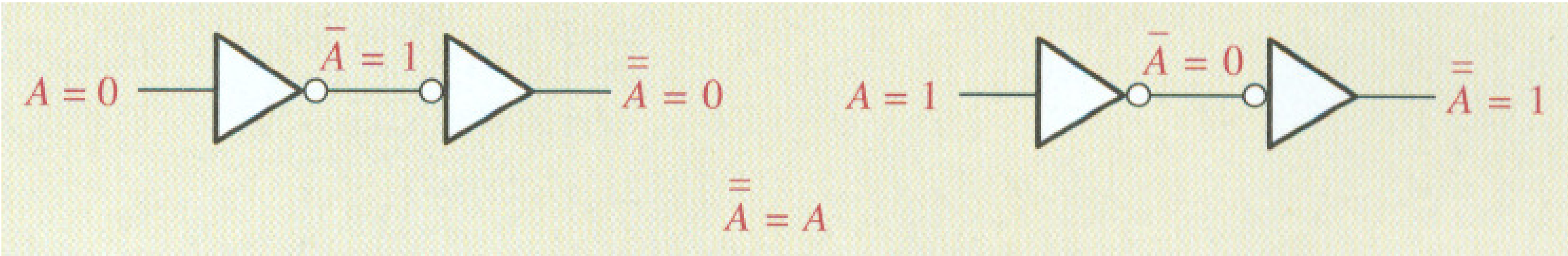


A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

AND Truth Table

Rules of Boolean Algebra

- Rule 9

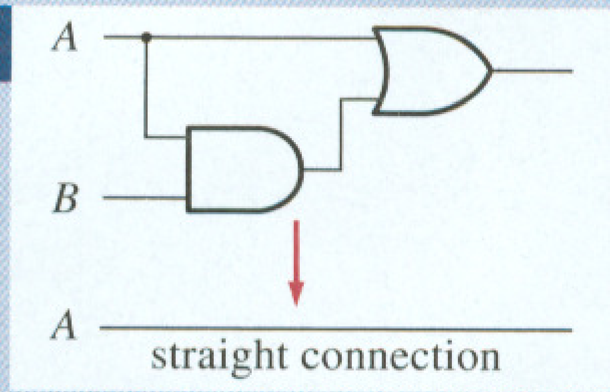


Rules of Boolean Algebra

- Rule 10: A + AB = A

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal (A column ↔ A + AB column)



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

AND Truth Table

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

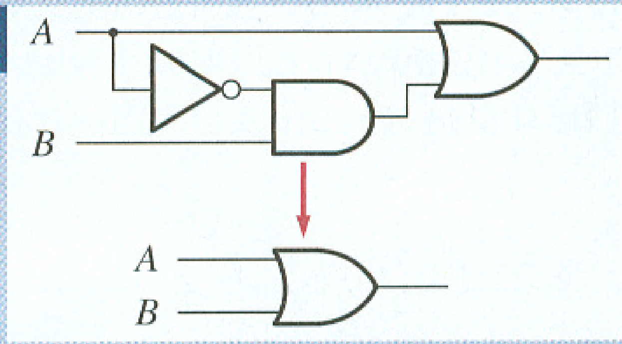
OR Truth Table

Rules of Boolean Algebra

- Rule 11: $A + \overline{A}B = A + B$

A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

$A + \overline{A}B$ and $A + B$: equal



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

AND Truth Table

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

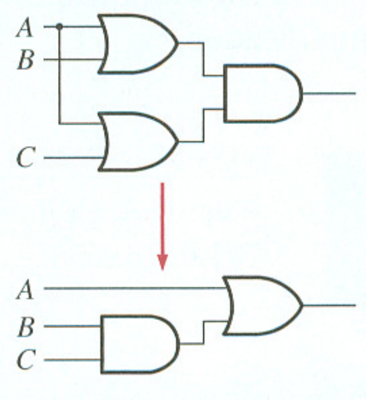
OR Truth Table

Rules of Boolean Algebra

- Rule 12: (A + B)(A + C) = A + BC

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

equal



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

AND Truth Table

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

OR Truth Table

DeMorgan's Theorems

- Theorem 1

$$\overline{XY} = \overline{X} + \overline{Y}$$

- Theorem 2

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$

Remember:

"Break the bar, change the sign"

DeMorgan's theorems are equally valid for use with three, four or more input variable expressions.

Example1:

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

Example2:

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

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

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

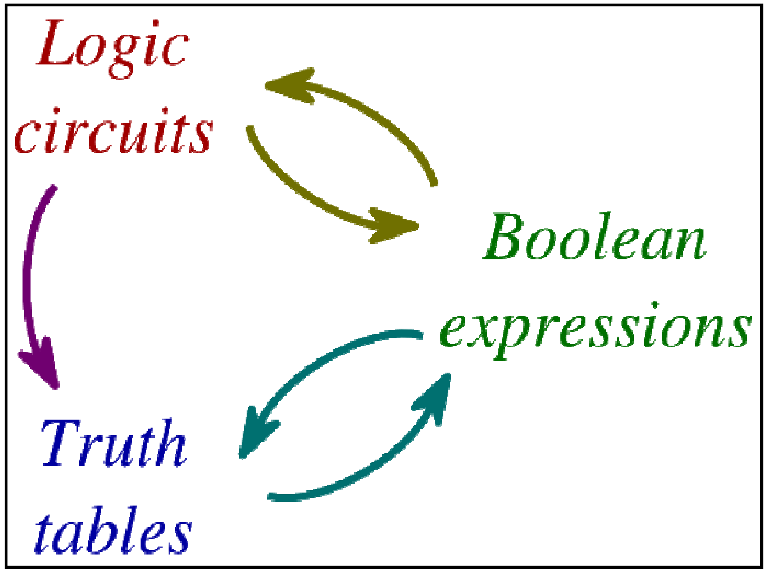
$$\overline{A}+B+C+D$$

Summary of the Rules of Boolean Algebra

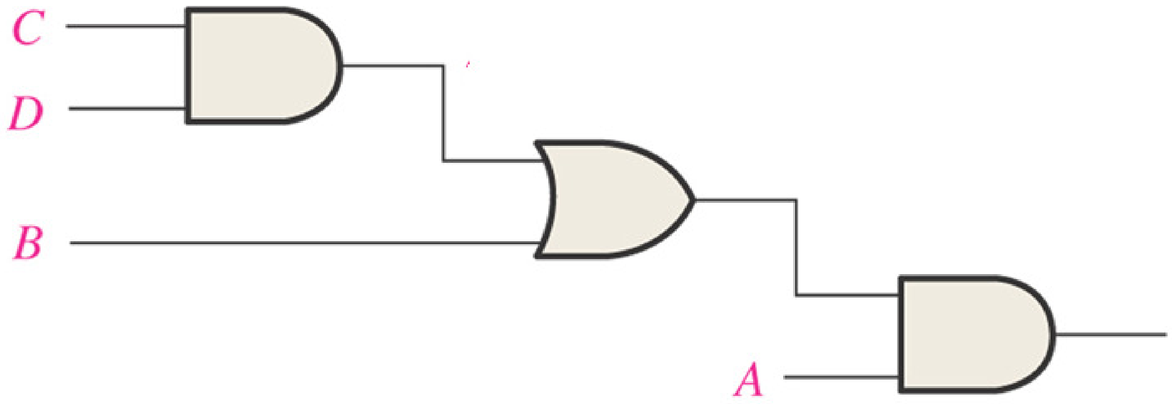
	AND Form	OR Form
Identify Law	$A.1 = A$	$A + 0 = A$
Zero and One Law	$A.0 = 0$	$A + 1=1$
Inverse Law	$A.\overline{A} = 0$	$A + \overline{A} =1$
Idempotent Law	$A.A = A$	$A + A = A$
Commutative Law	$A.B = B.A$	$A +B = B + A$
Associative Law	$A.(B.C) = (A.B).C$	$A + (B+C) = (A +B) + C$
Distributive Law	$A+(B.C)=(A+B).(A+C)$	$A.(B+C)= (A.B) + (A.C)$
Absorption Law	$A(A+B) = A$	$A + A.B =A$ $A +\overline{A} B = A+B$
DeMorgan's Law	$\overline{(A.B)} = \overline{A} + \overline{B}$	$\overline{(A+B)} = \overline{A}.\overline{B}$
Double Complement Law	$\overline{\overline{X}}=X$	

Relations Between Logic Forms

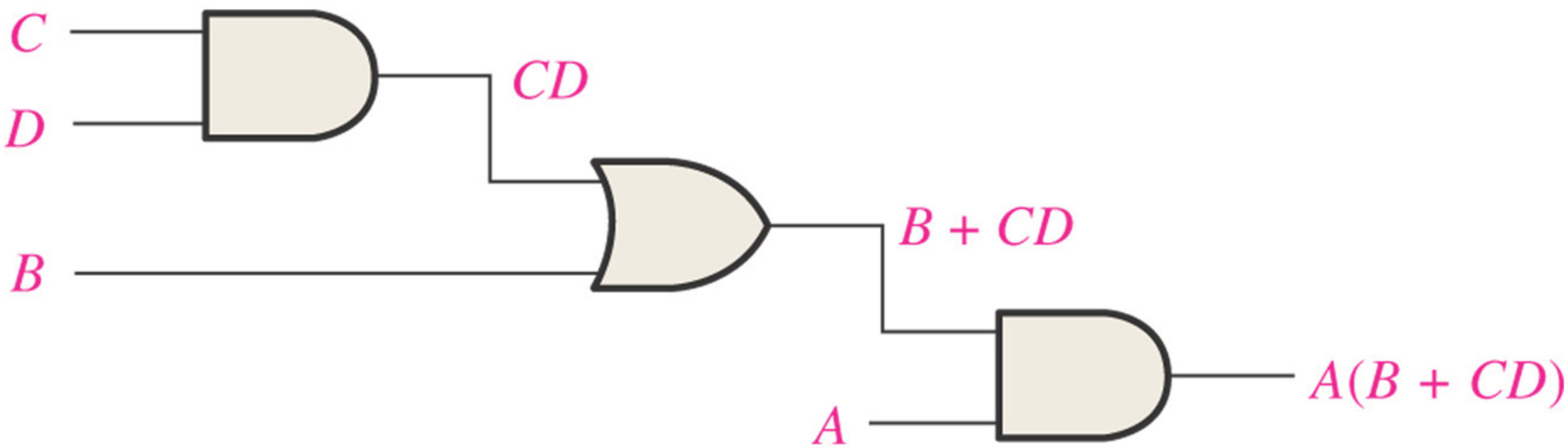
Boolean Expression to Truth-table: Evaluate expression for all input combinations and record output values.
Boolean Expression to Logic Circuit : Use AND gates for the AND operators, OR gates for the OR operators, and inverters for the NOT operator. Wire up the gates the match the structure of the expression.
Logic Circuit to Boolean Expression: Reverse the above process



Example: Find the Boolean Expression for the logic circuit below



Solution



Example: For the below Boolean Expression find out the Truth table and Logic Circuit

$$F = x + \overline{y}z$$

Solution:

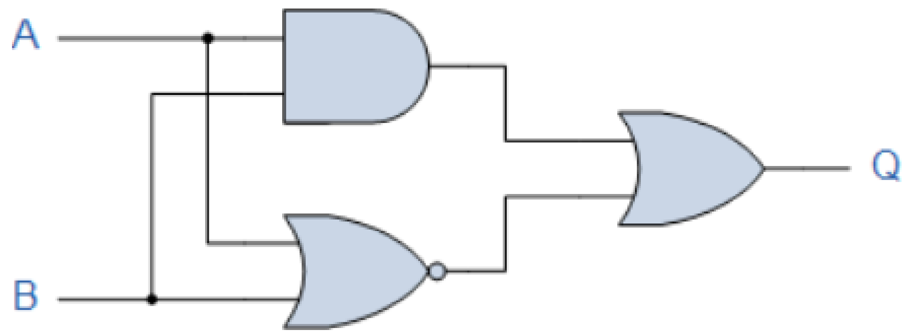
- **Truth Table**

  All possible combinations of input variables

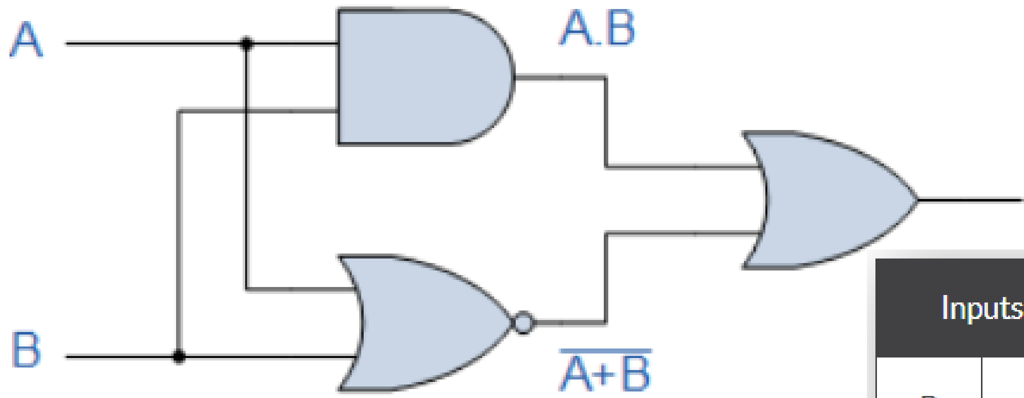
- **Logic Circuit**

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Example Find the Boolean Expression and Truth Table for below Logic Circuit.



Solution:



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

Inputs		Intermediates		Output
B	A	A.B	$\overline{A+B}$	Q
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1