Tishk International University
Science Faculty
IT Department

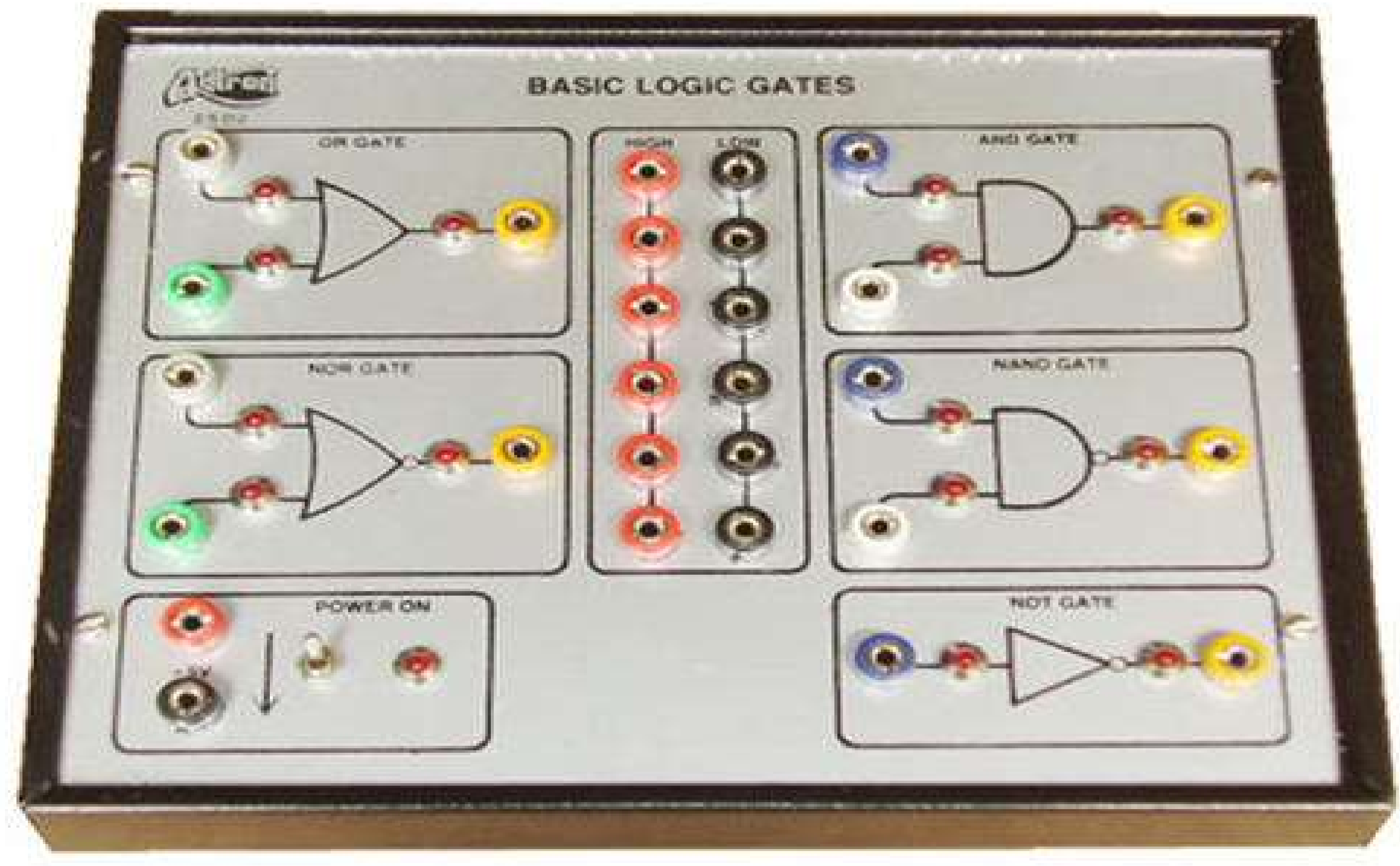
Logic Design

Lecture 02: Logic Gates and Boolean Algebra

2nd Grade

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

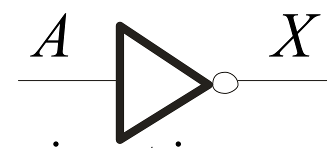
- The two binary digits are designated **0** and **1**
- They can also be called LOW and HIGH, where **LOW = 0** and **HIGH = 1**
- In order to practice with Logic Gates we can use:
  - **LogicCircuit**

    https://www.logiccircuit.org/downloads/LogicCircuit.Setup.2.22.07.22.zip

Logic Gates

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

The Inverter

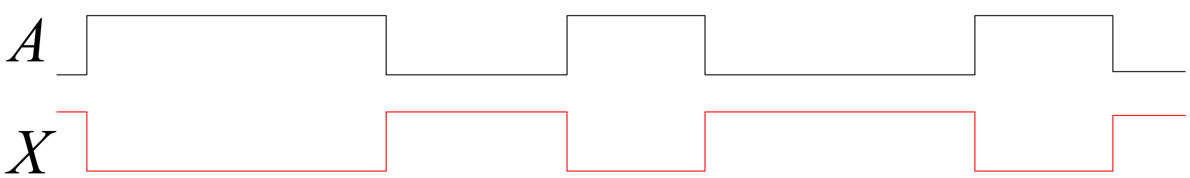


The inverter performs the Boolean **NOT** operation. When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW.

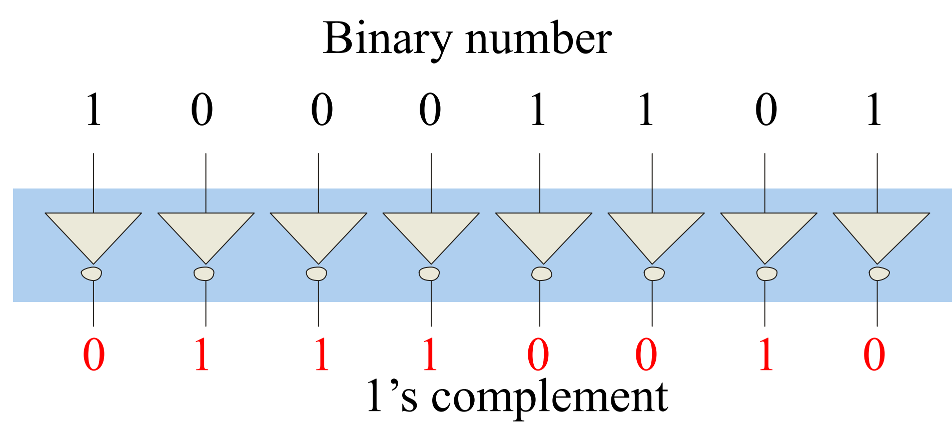
The **NOT** operation (complement) is shown with an overbar. Thus, the Boolean expression for an inverter is $X = \overline{A}$.

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

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:

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

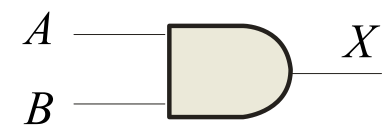
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

- For three input variables:

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

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

The AND Gate

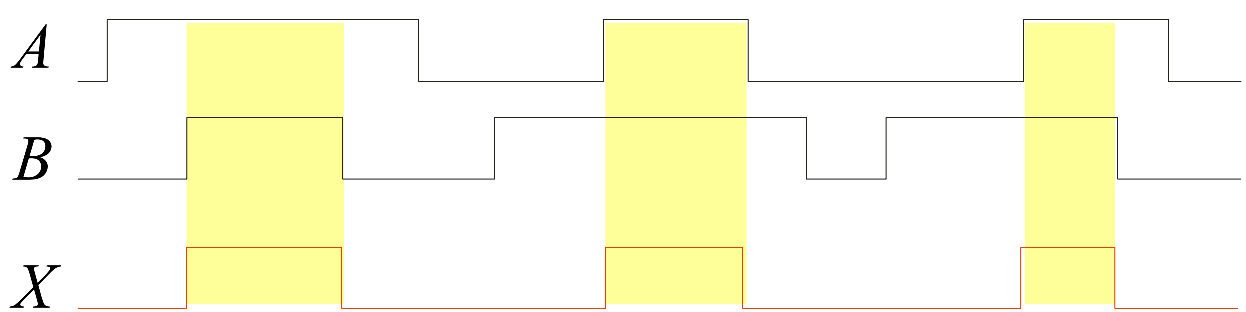


The **AND gate** produces a HIGH output when all inputs are HIGH; otherwise, the output is LOW. For a 2-input gate, the truth table is

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 $X = AB$.

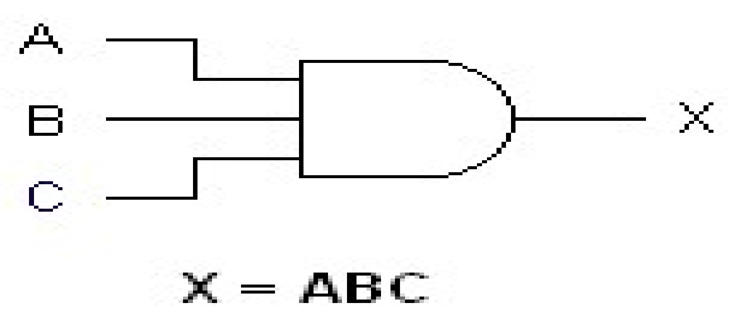
The AND operation is used in computer programming as a selective mask. If you want to retain certain bits of a binary number but reset the other bits to 0, you could set a mask with 1's in the position of the retained bits.

Example waveforms:

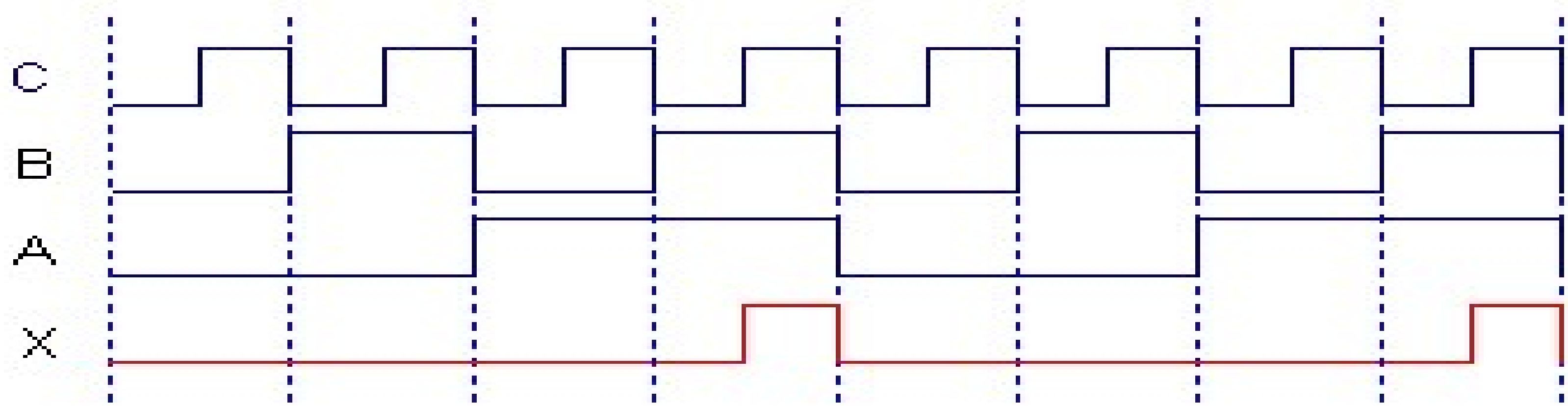


Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

The AND Gate for more than 2 inputs

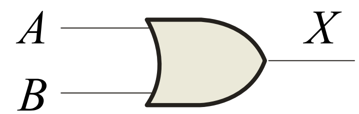


A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



3-Input AND Gate

The OR Gate

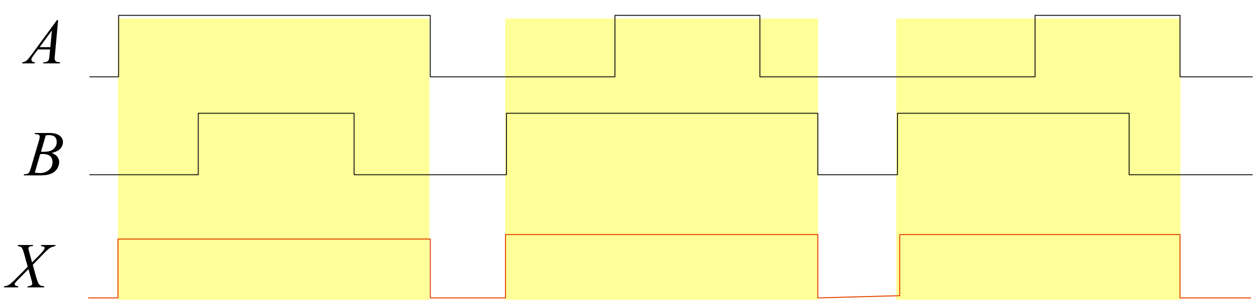


The **OR gate** produces a HIGH output if any input is HIGH; if all inputs are LOW, the output is LOW.

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

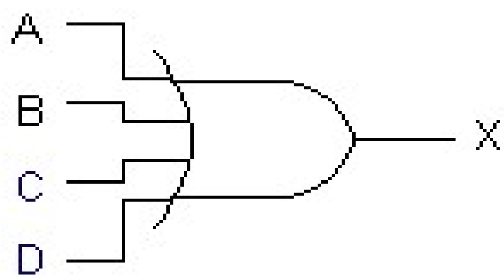
The OR operation can be used in computer programming to set certain bits of a binary number to 1.

Example waveforms:

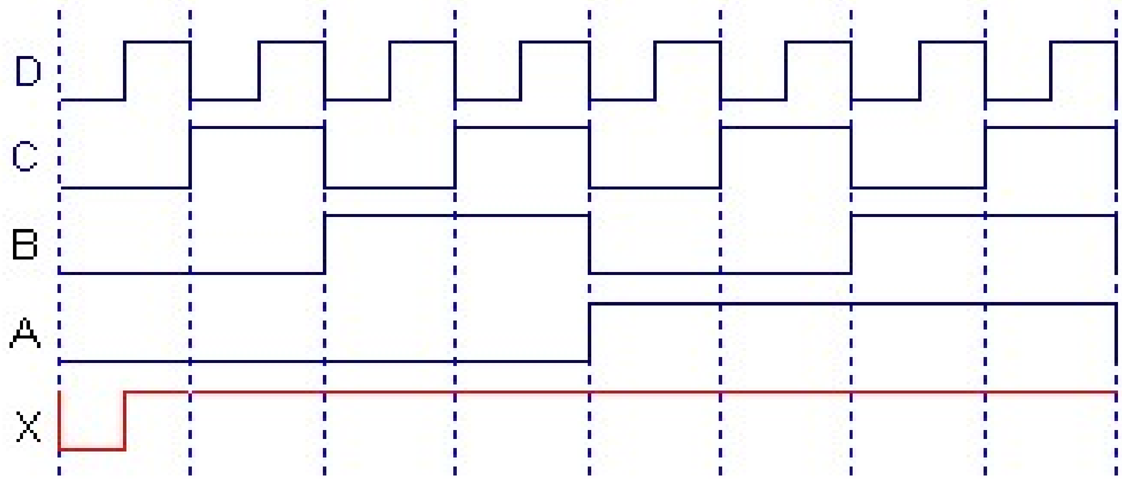


Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

The OR Gate for more than 2 inputs



X = A+B+C+D



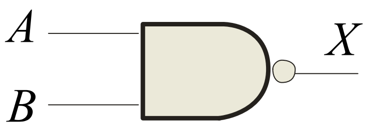
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4-Input OR Gate

The NAND Gate

The **NAND gate** produces a LOW output when all inputs are HIGH; otherwise, the output is HIGH.

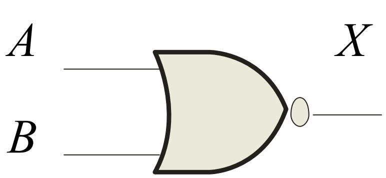
The NAND gate is particularly useful because it is a "universal" gate – all other basic gates can be constructed from NAND gates.



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

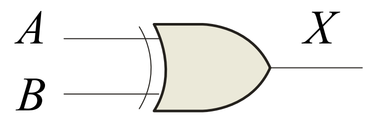
The NOR Gate

The **NOR gate** produces a LOW output if any input is HIGH; if all inputs are HIGH, the output is LOW. The NOR operation will produce a LOW if any input is HIGH.



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

The XOR Gate

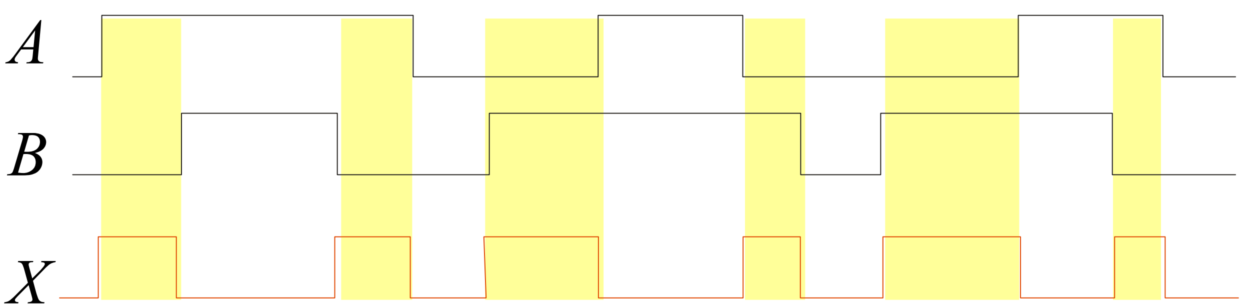


The **XOR gate** produces a HIGH output only when both inputs are at opposite logic levels.

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 $X = A \oplus B$.

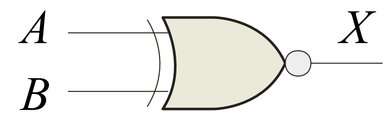
Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

Example waveforms:



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

The XNOR Gate

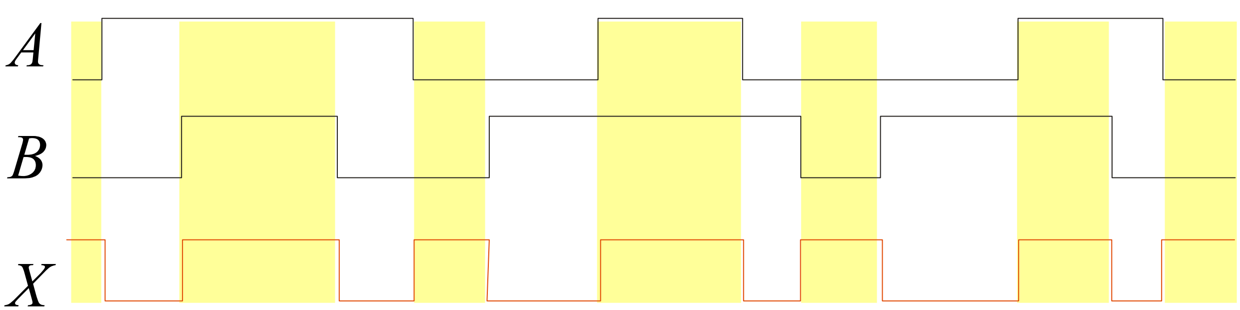


The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level.

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

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

Example waveforms:



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

Boolean Operations and Expressions

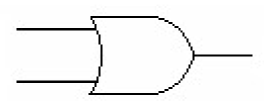
- Addition

  0 + 0 = 0

  0 + 1 = 1

  1 + 0 = 1

  1 + 1 = 1



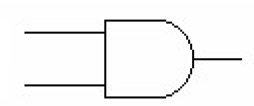
- Multiplication

  0 * 0 = 0

  0 * 1 = 0

  1 * 0 = 0

  1 * 1 = 1



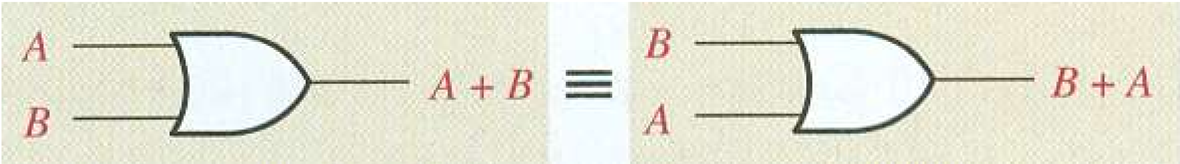
Laws of Boolean Algebra

- Commutative Laws
- Associative Laws
- Distributive Laws
- Self Rules
- Absorption Laws
- DeMorgan's Laws

Commutative Laws

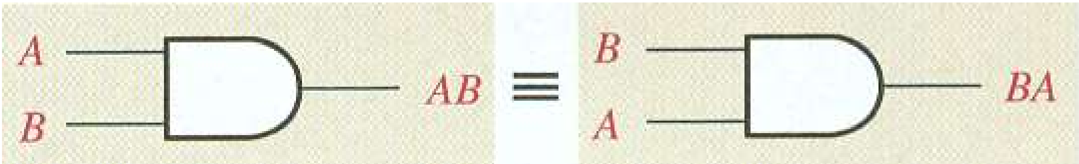
- Commutative Law of Addition:

A + B = B + A



- Commutative Law of Multiplication:

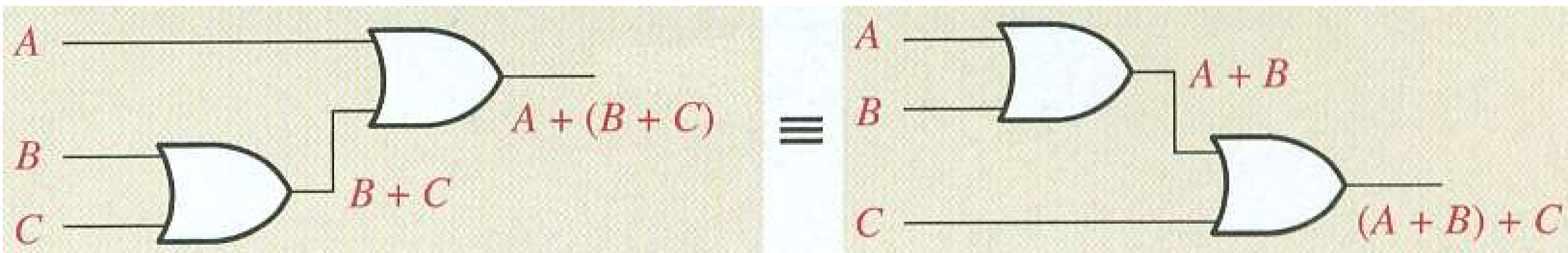
A * B = B * A



Associative Laws

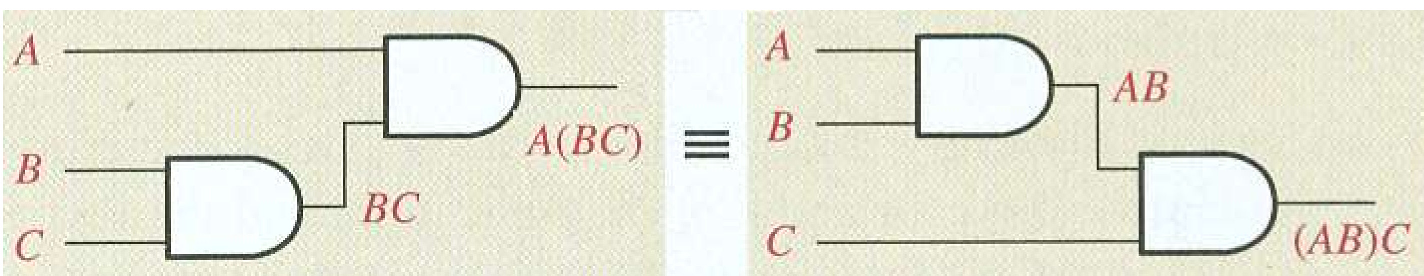
- Associative Law of Addition:

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



- Associative Law of Multiplication:

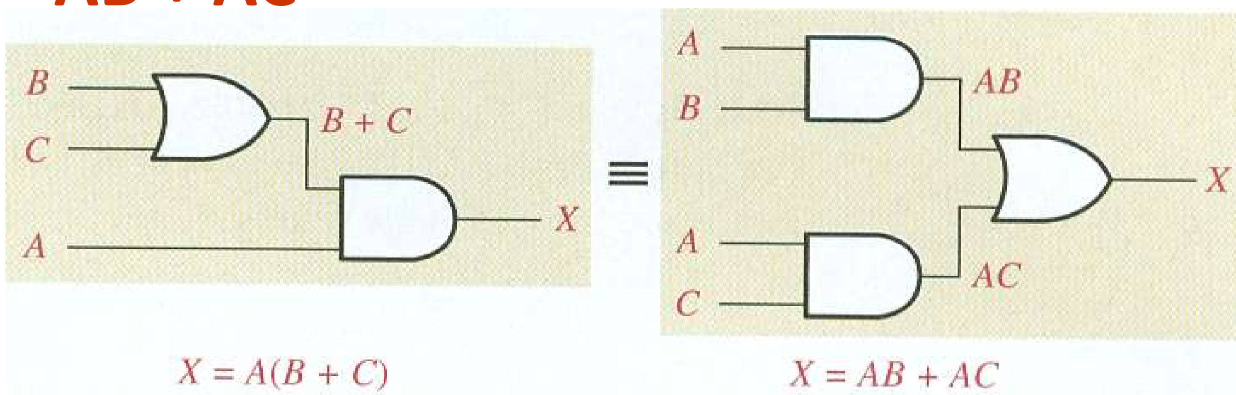
A * (B * C) = (A * B) * C



Distributive Laws

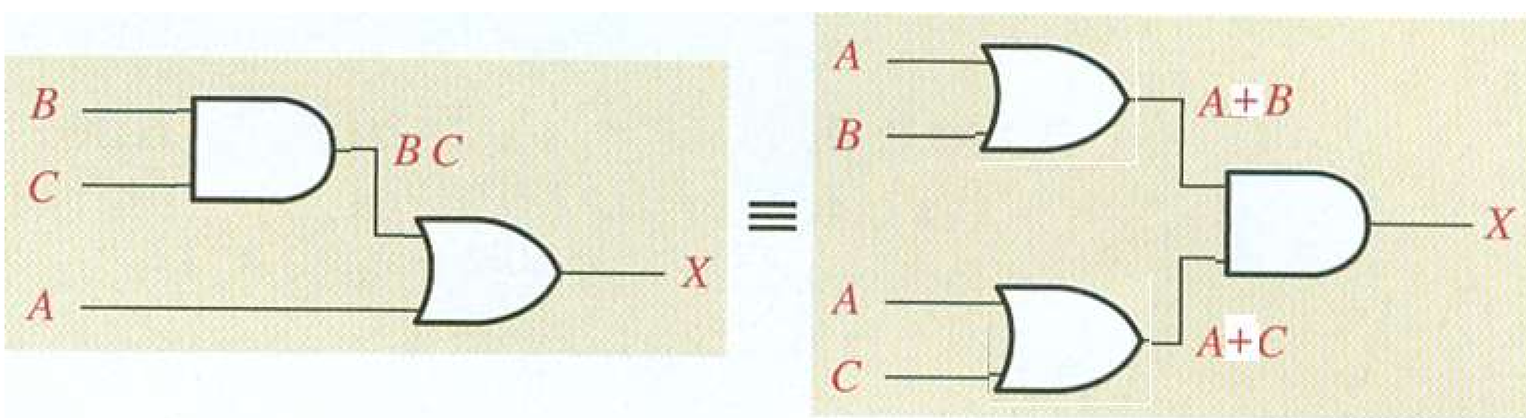
- Distributive Law of Multiplication over Addition:

A(B + C) = AB + AC



- Distributive Law of Addition over Multiplication :

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



Self Rules

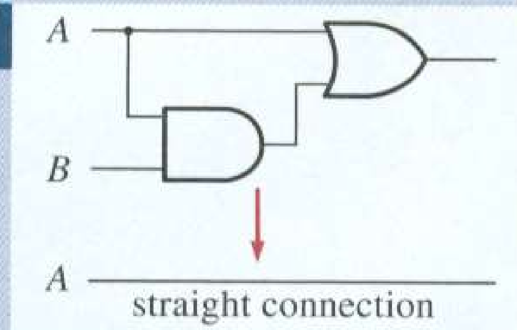
1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$

Absorption Laws

$A + AB = A$

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

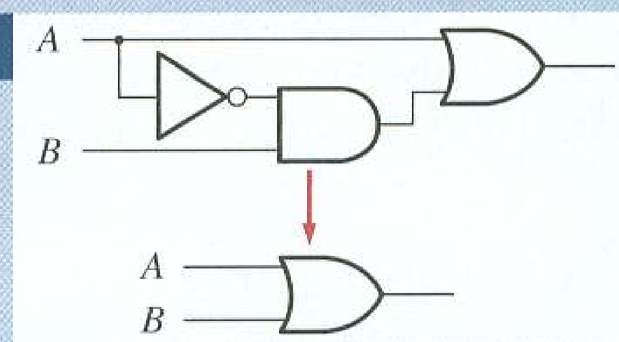
equal



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

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

equal



DeMorgan's Laws

- Law 1

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

- Law 2

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

Remember:

"Break the bar, change the sign"

DeMorgan's laws 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:

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

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

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

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

Summary of the Laws of Boolean Algebra

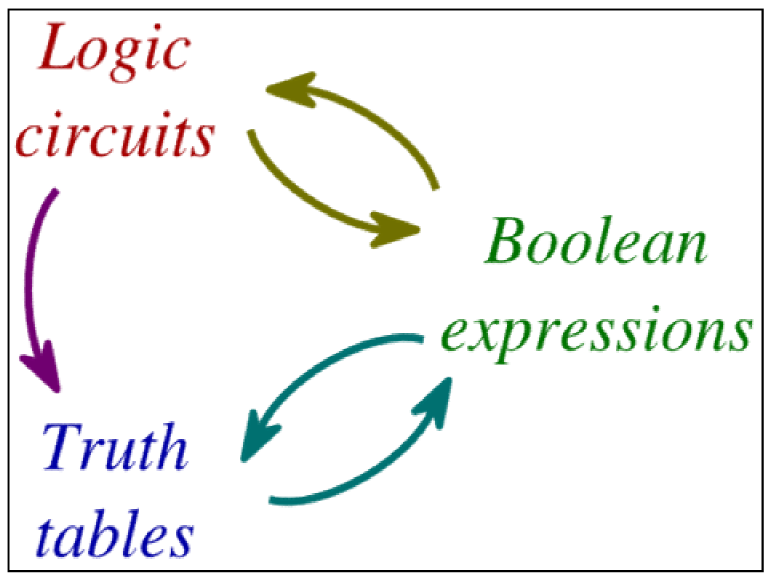
	AND Form	OR Form
Identify Law	$A.1 = A$	$A + 0 = A$
Zero and One Law	$A.0 = 0$	$A + 1 = 1$
Inverse Law	$A.\overline{A} = 0$	$A + \overline{A} = 1$
Idempotent Law	$A.A = A$	$A + A = A$
Commutative Law	$A.B = B.A$	$A + B = B + A$
Associative Law	$A.(B.C) = (A.B).C$	$A + (B+C) = (A + B) + C$
Distributive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

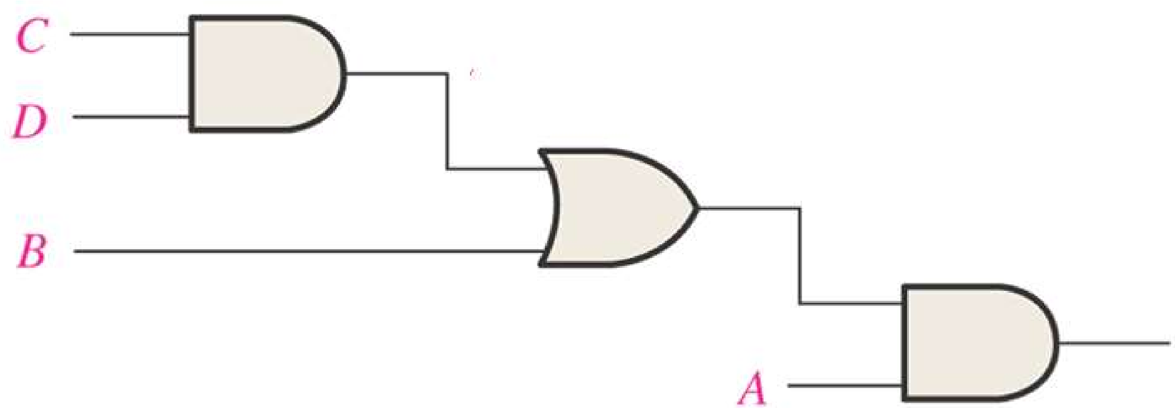
Boolean Expression to Truth-table: Evaluate expression for all input combinations and record output values.

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.

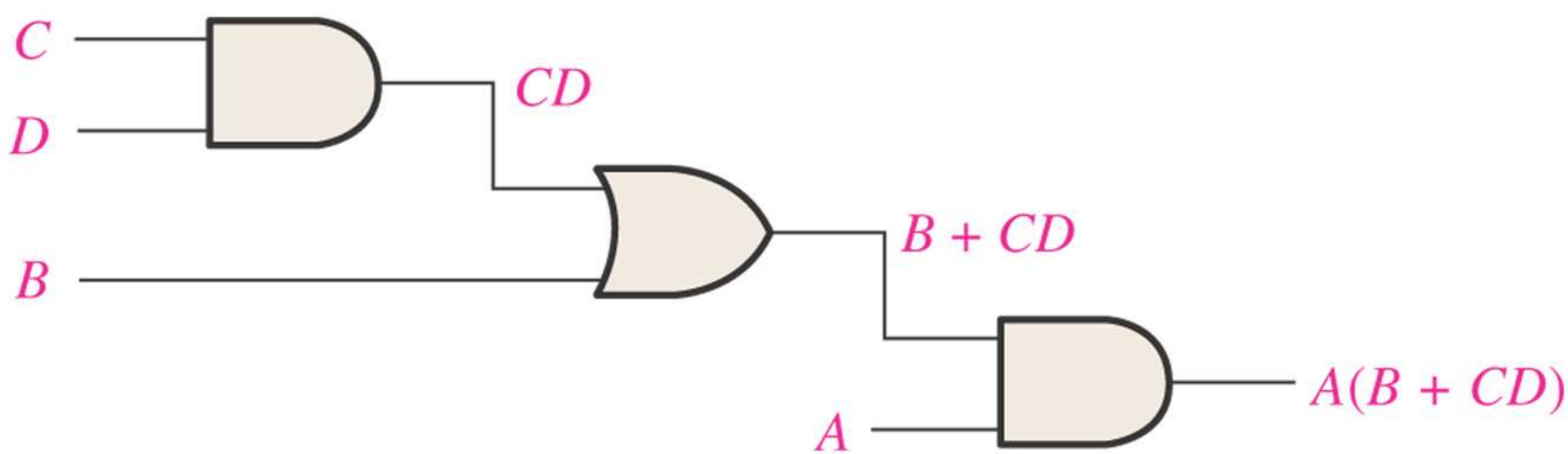
Logic Circuit to Boolean Expression: Reverse the above process



Example: Find the Boolean Expression for the logic circuit below



Solution

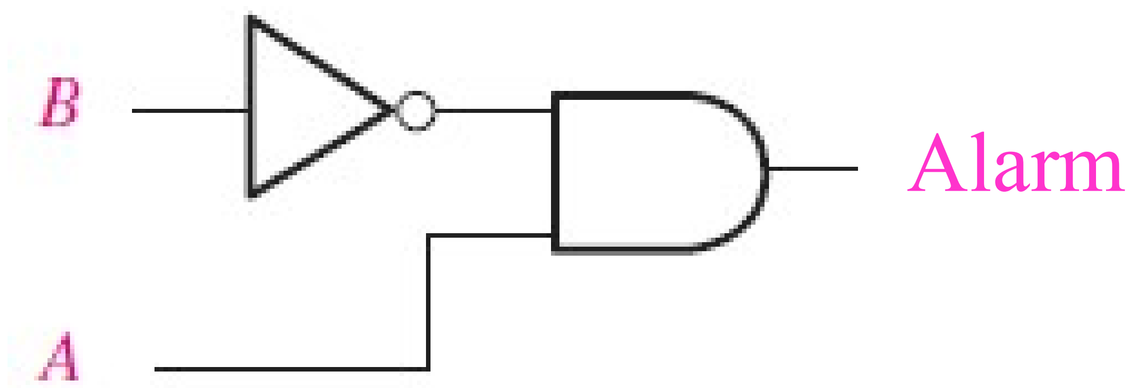


Example: Draw the circuit diagram of a simple seat belt alarm circuit , if both inputs Ignition switch is ON , and Seat belt is disconnected , then the output is high and the alarm is activated.

A = Ignition switch (ON is 1)

B = Seat belt (Connected is 1)

Solution:



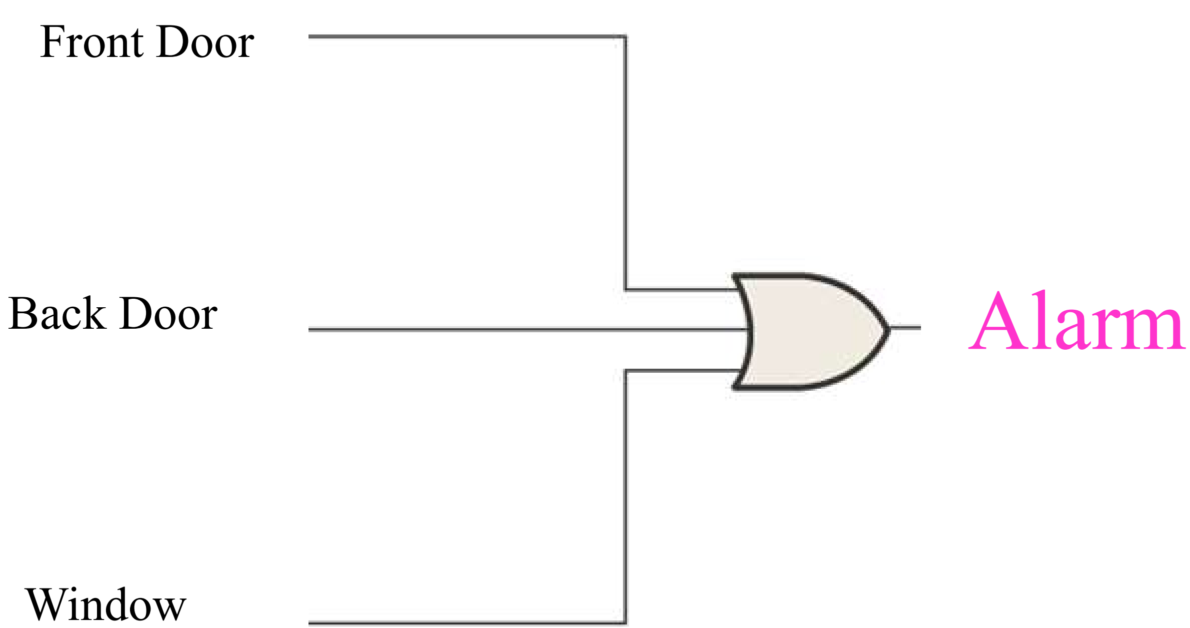
Example: Draw the circuit diagram of a simplified intrusion detection system, if any of the three inputs are high, then the output is high and the alarm is activated.

Front Door (Open =1)

Back Door (Open =1)

Window (Open =1)

Solution:



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

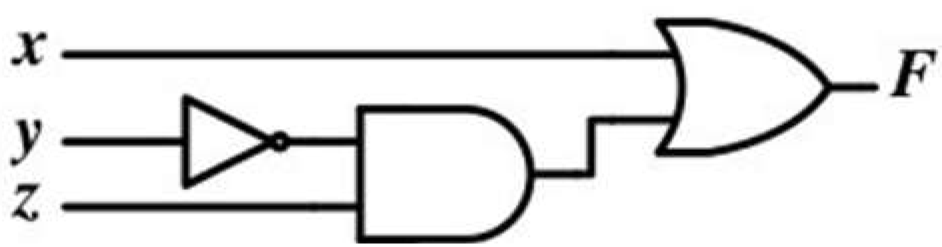
$$F = x + \overline{y}\,z$$

Solution:

- **Truth Table**

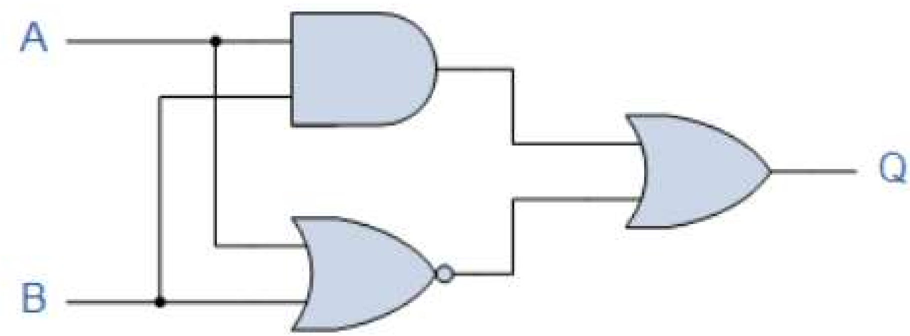
  All possible combinations of input variables

- **Logic Circuit**

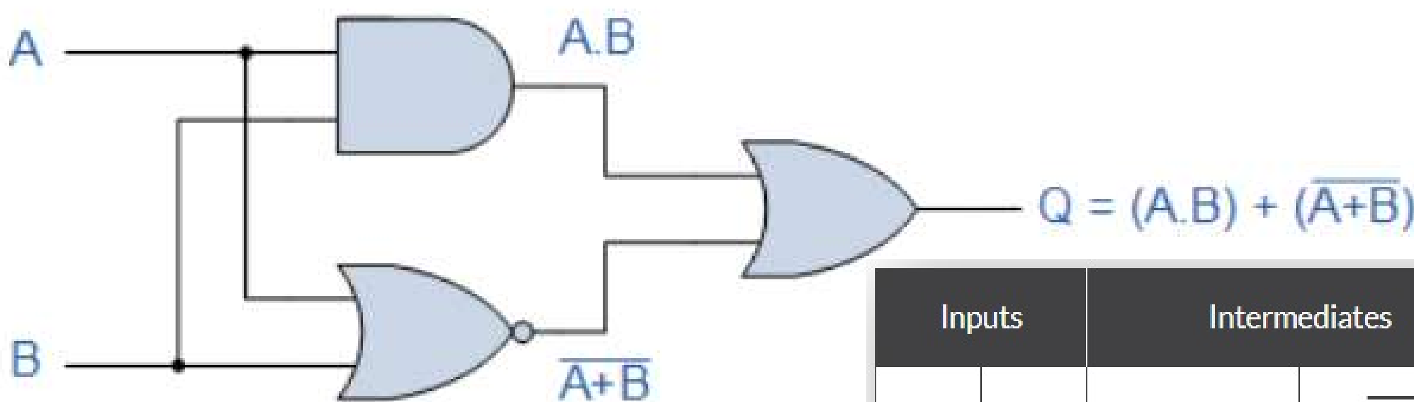


x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

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



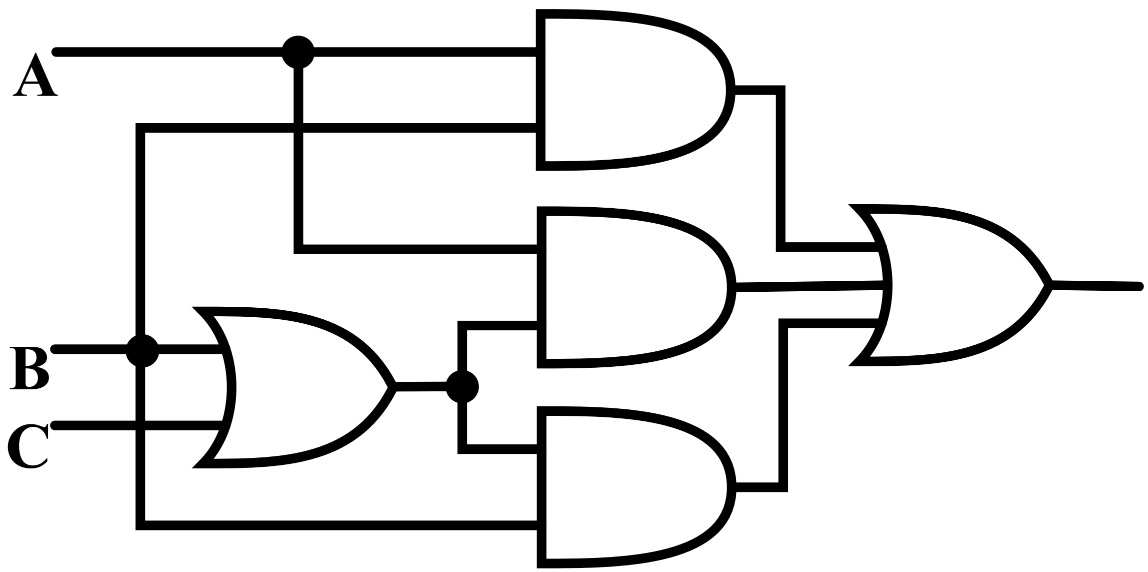
Solution:



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

Simplification Using Boolean Algebra

- A simplified Boolean expression uses the fewest gates possible to implement a given expression.

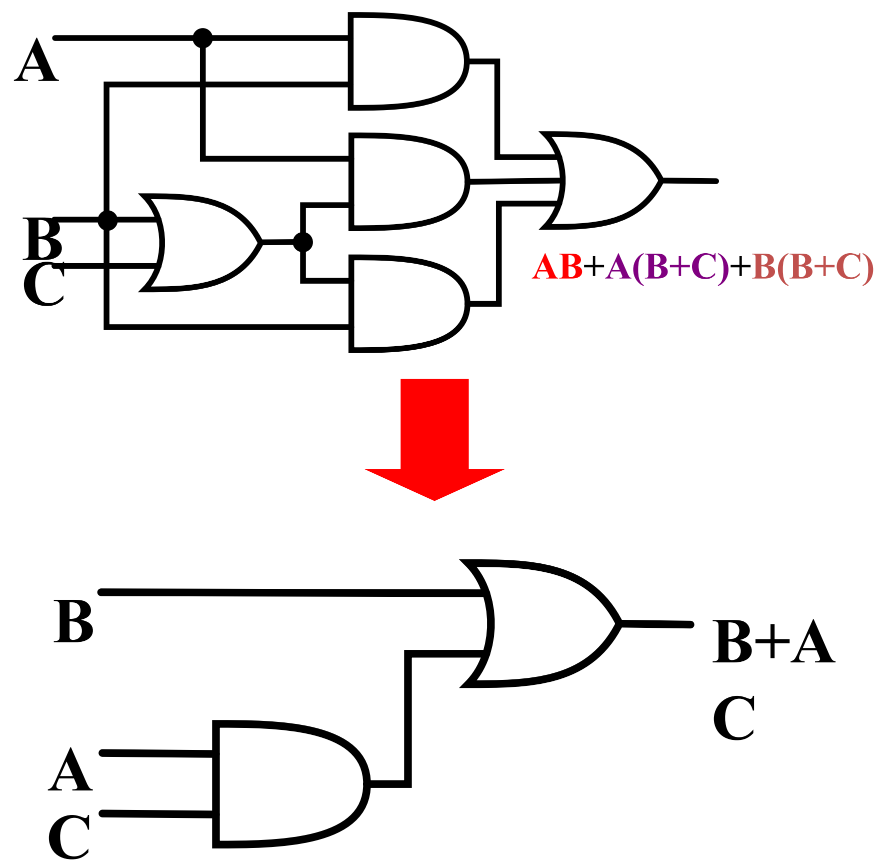


AB+A(B+C)+B(
B+C)

Example: Simplify the expression: AB+A(B+C)+B(B+C)

Solution:

- AB+A(B+C)+B(B+C)
  - (distributive law)
    - AB+AB+AC+BB+BC
  - (rule 7; BB=B)
    - AB+AB+AC+**B**+BC
  - (rule 5; AB+AB=AB)
    - **AB**+AC+B+BC
  - (rule 10; B+BC=B)
    - AB+AC+**B**
  - (rule 10; AB+B=B)
    - **B**+AC



Example: Simplify the below Expression using Boolean Expression

$$X = \overline{A\overline{B}.(A+C)} + \overline{A}B.\overline{[A+\overline{B}+\overline{C}]}$$

Solution:

$$
\begin{aligned}
X &= \overline{A\overline{B}\cdot(A+C)} + \overline{A}B\cdot\overline{A+\overline{B}+\overline{C}} \\
&= \overline{A\overline{B}} + \overline{A+C} + \overline{A}B\cdot(\overline{A}\cdot\overline{\overline{B}}\cdot\overline{\overline{C}}) \\
&= (\overline{A}+\overline{\overline{B}}) + \overline{A}\cdot\overline{C} + \overline{A}\,\overline{A}BBC \\
&= \overline{A} + B + \overline{A}\,\overline{C} + \overline{A}BC \\
&= \overline{A}(1+\overline{C}) + B + \overline{A}BC \\
&= \overline{A} + B + \overline{A}BC \\
&= \overline{A} + B(1+\overline{A}C) \\
&= \overline{A} + B \quad \leftarrow \textit{simplified equation}
\end{aligned}
$$

Example: Simplify the below Expression using Boolean Expression

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

Solution:

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

↓ Breaking longest bar

$$(\overline{\overline{A + BC}})\ (\overline{\overline{A\overline{B}}})$$

↓ Applying identity $\overline{\overline{A}} = A$ wherever double bars of equal length are found

$$(A + BC)(A\overline{B})$$

↓ Distributive property

$$AA\overline{B} + BCA\overline{B}$$

↓ Applying identity $AA = A$ to left term; applying identity $A\overline{A} = 0$ to B and $\overline{B}$ in right term

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

↓ Applying identity $A + 0 = A$

$$A\overline{B}$$