

TSN1101

Computer Architecture and Organization

Section A (Digital Logic Design)

Lecture 02


Introduction to Digital Logic and
Boolean Algebra

Introduction to Digital Logic and Boolean Algebra

– Part 1 of 3

Introductory Digital Concepts
- Digital Vs Analog Systems

TOPIC COVERAGE

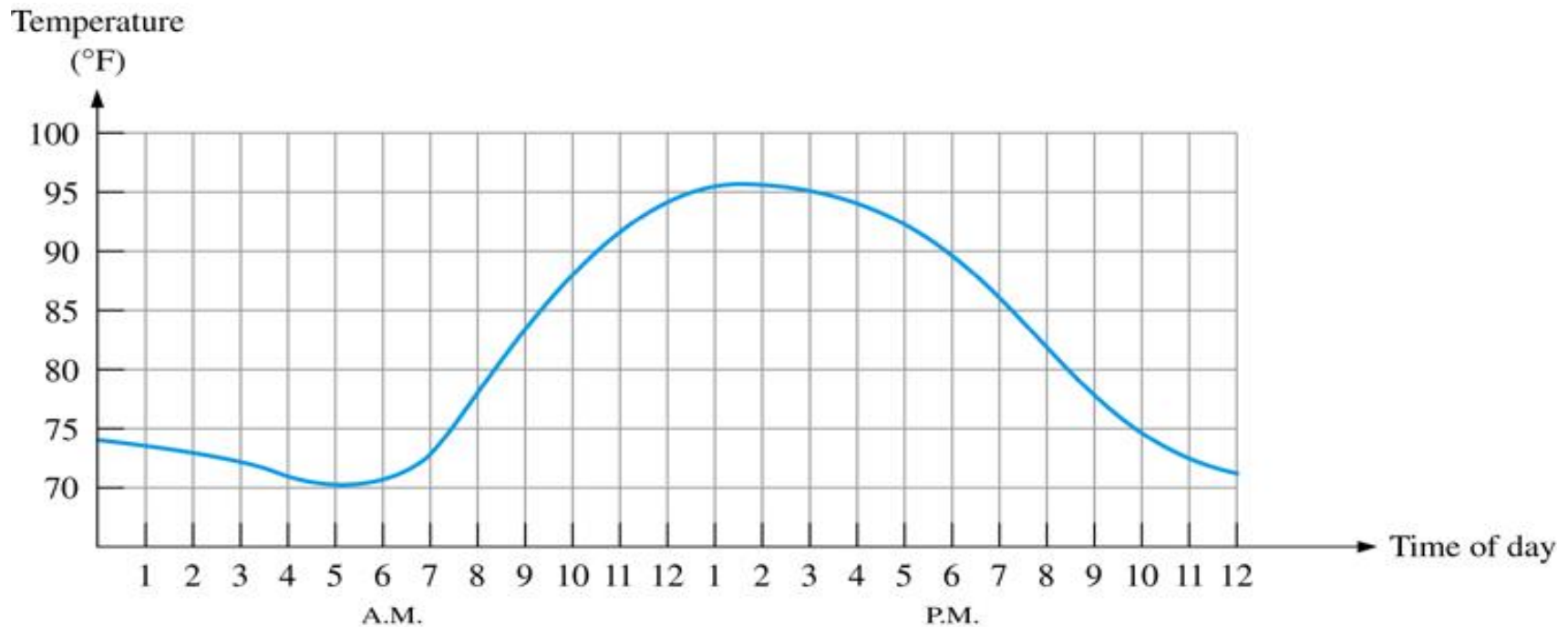
- PART 1 of 3

- ❑ Analog Vs Digital Quantities
- ❑ Analog Vs Digital Electronic Systems – Examples
- ❑ Advantages and Limitations of Digital Systems
- ❑ Hybrid Systems
- ❑ Logic levels – Positive and Negative Logic
- ❑ Digital Waveforms – Period, Frequency, Duty cycle
- ❑ Timing Diagram
- ❑ Types of Data Transfer – Serial Vs Parallel

Analog Quantities

An **analog quantity** is one having continuous values.

Examples: Temperature, Pressure, Level, Position, Volume, Voltage, Current



Graph of an Analog Quantity – Temperature Vs Time

Digital Quantities

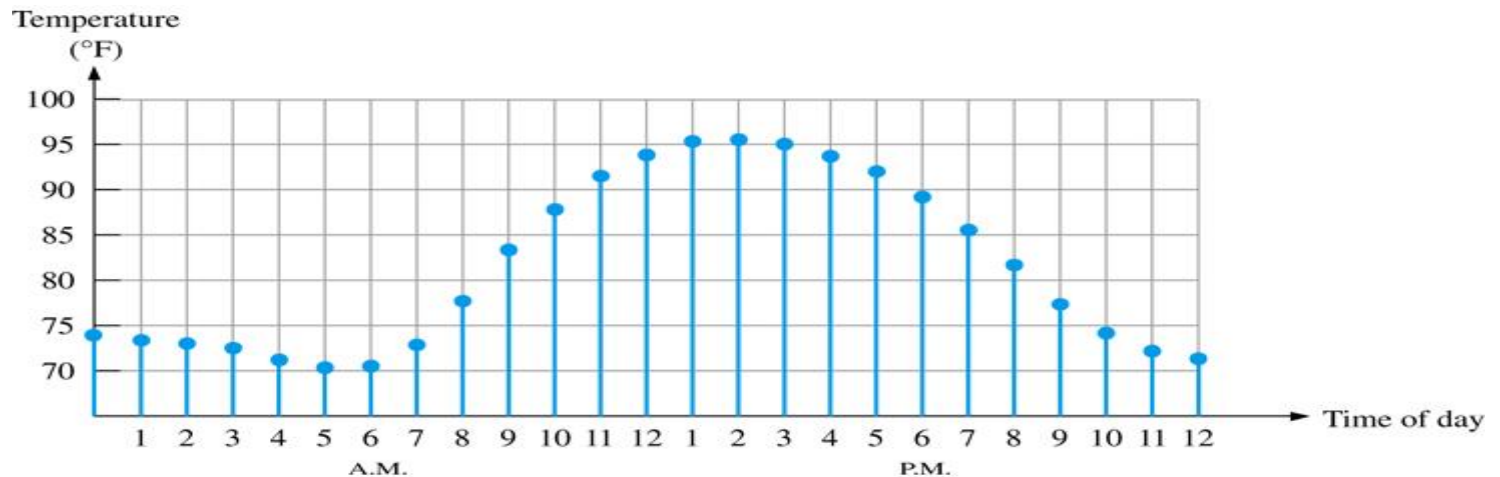
A **digital quantity** is one having a discrete set of values.

Examples: Digital Watch reading – (*Time of the day in minutes/seconds*)

Number of coins

Human population of a city (it changes with the time)

People travel from/to the city

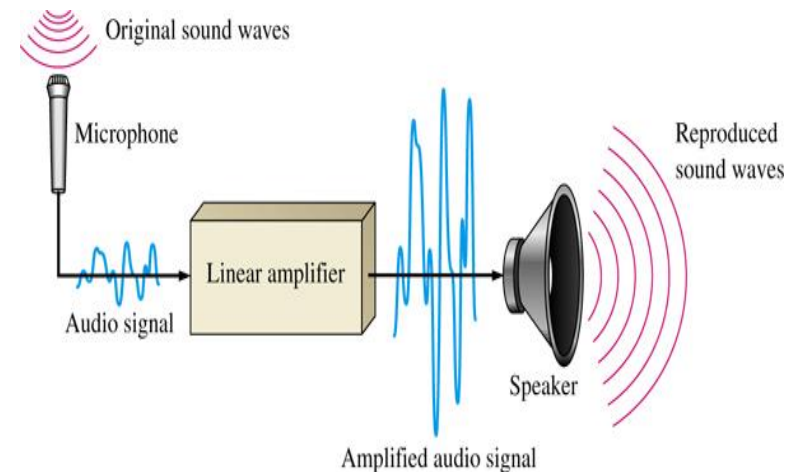


Sampled-value representation of Temperature Vs Time

Analog Electronic System

– An Example

- Analog system
 - *A combination of devices that manipulate values represented in an analog form*
 - Here the variable is allowed to take any value in a specified range.
 - **An example:** A basic audio public address system.
Sound through a microphone causes voltage changes in proportion to the amplitude of the sound waves.



Digital System

- Examples

- Digital system
 - *A combination of devices that manipulate values represented in digital form.*

Examples:

- Digital Computer
- Handheld Calculator
- Digital Watch
- Telephone system
- Digital audio and video equipment



Analog Watch and Digital Watch

Advantages of Digital over Analog

- ❑ Data Processing and Transmission – more efficient and reliable
- ❑ Data Storage – more compact storage and greater accuracy and clarity in reproduction
- ❑ Ease of design – In switching circuits, only the range in which the voltage or current fall is important not the exact values
- ❑ Accuracy and precision are easier to maintain – In analog systems, voltage and current signals are affected by temperature, humidity but in digital systems, information does not degrade
- ❑ Easy Programmable operation
- ❑ Less affected by noise - since exact value is not important in digital systems
- ❑ Ease of fabrication on IC chips – analog devices cannot be economically integrated.

Limitations of Digital Techniques

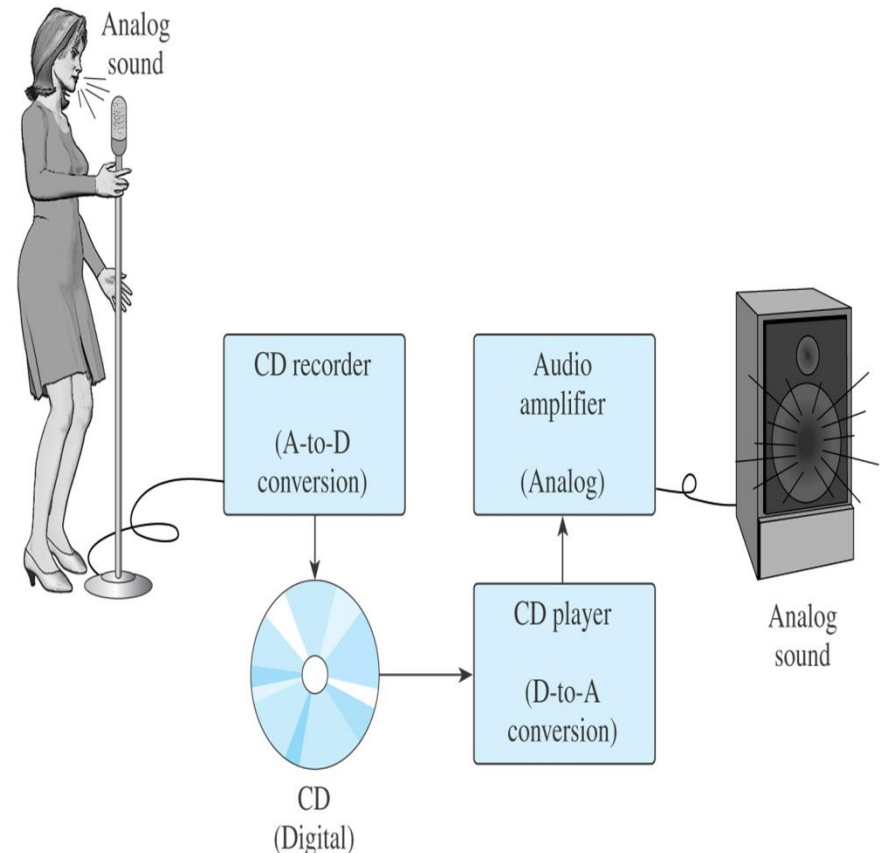
- The real world is analog
- The analog nature of the world requires a time consuming conversion process:
 - Convert analog inputs to digital
 - Process (operate on) the digital information
 - Convert the digital output back to analog

Digital and Analog electronics together

- Examples

Example 1: The audio CD is a typical hybrid (combination) system.

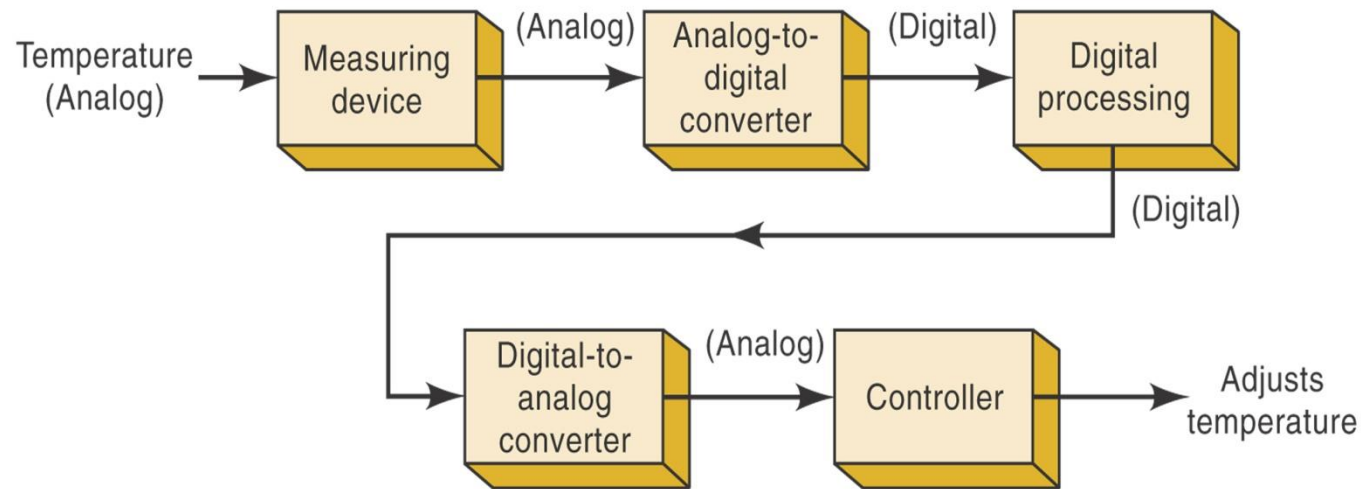
- Analog sound is converted into analog voltage.
- Analog voltage is changed into digital through an ADC in the recorder.
- Digital information is stored on the CD .
- At playback the digital information is changed into analog by a DAC in the CD player.
- The analog voltage is amplified and used to drive a speaker that produces the original analog sound.



Digital and Analog electronics together

-Examples

Example 2: Temperature Control System



Binary Digits and Logic Levels

- Positive Logic

HIGH = 1 Low = 0

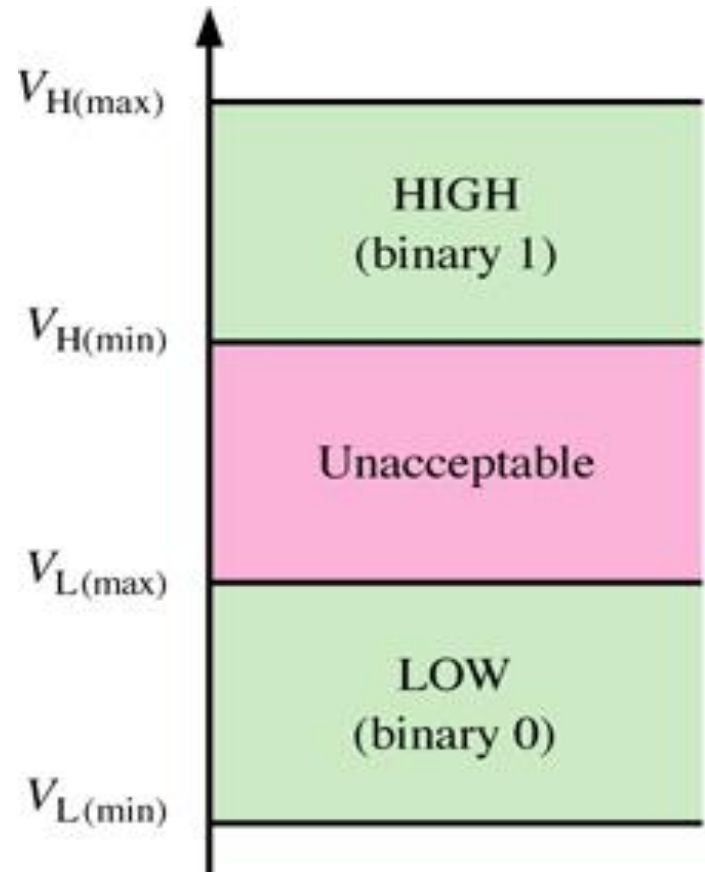
Logic Levels - The voltages used to represent a 1 and a 0

Ex: For TTL , HIGH=2V to 5 V

LOW=0 V to 0.8 V

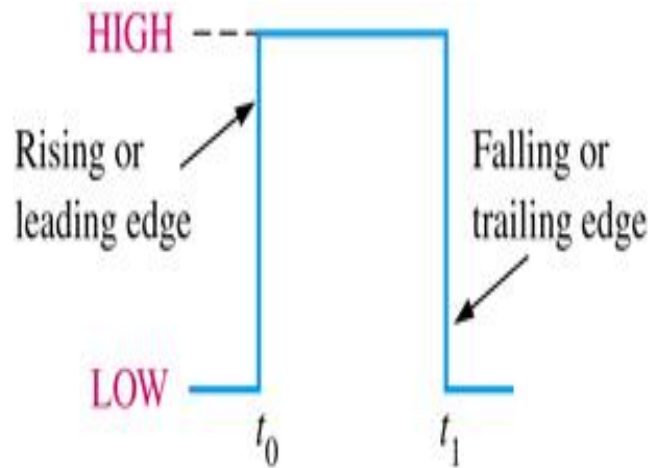
- Negative logic

High =0 Low =1

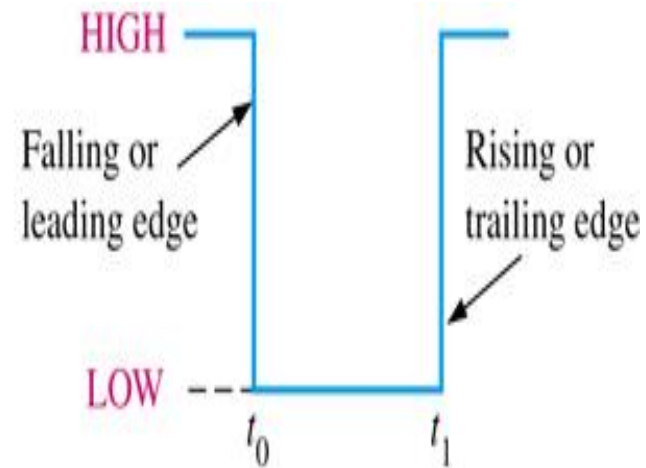


Digital Waveforms

– Ideal Pulse



(a) Positive-going pulse



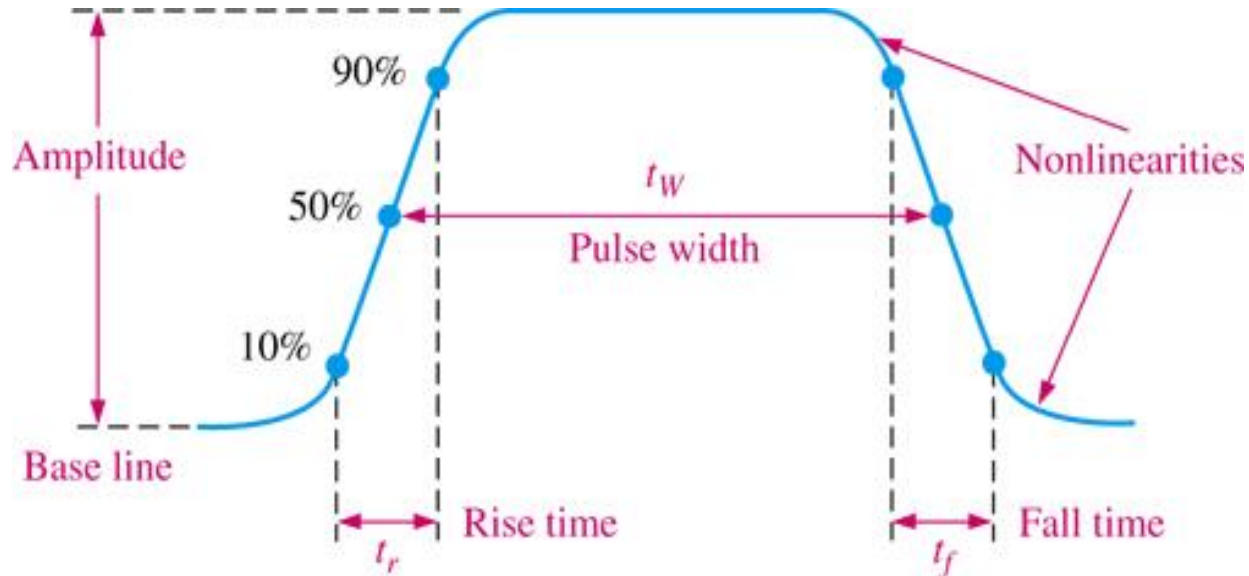
(b) Negative-going pulse

IDEAL PULSES

- Digital Waveforms is made up of a series of pulses
- Digital Waveforms consist of voltage levels that are changing back and forth between HIGH and LOW states.

Digital Waveforms

– Non-ideal pulse



NONIDEAL PULSE CHARACTERISTICS

Rise Time – measured from 10% of the pulse amplitude to 90% of the pulse amplitude

Fall Time - measured from 90% of the pulse amplitude to 10% of the pulse amplitude

Pulse Width – Time interval between the 50% points on the rising and falling edges

Digital Waveforms-Characteristics

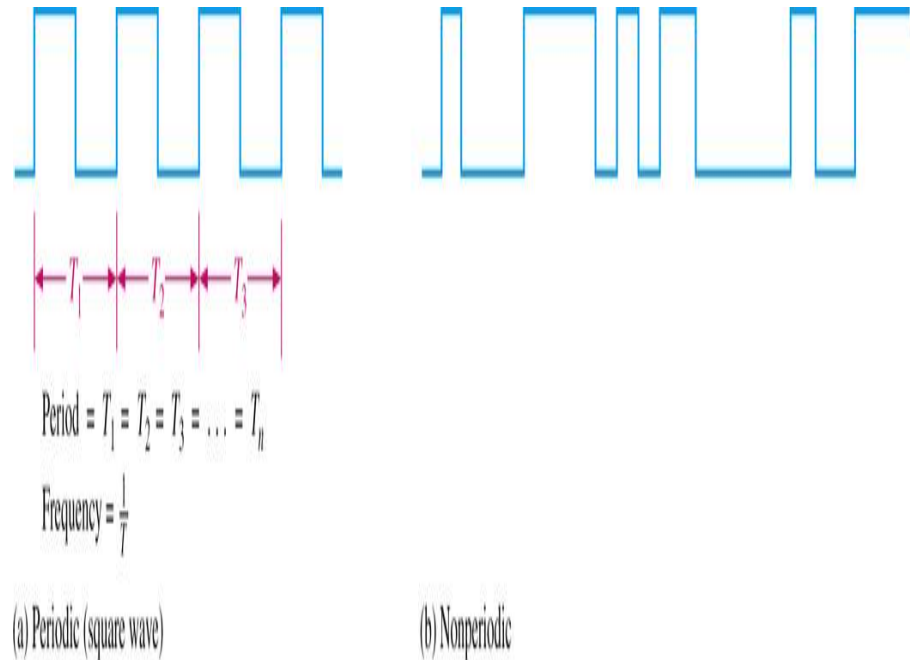
- Periodic Vs Non-periodic

- Periodic pulse waveform

One that repeats itself at a fixed interval, called a period

- Non-periodic pulse waveform

Composed of pulses of randomly differing time interval between pulses (pulse width)



Periodic Digital Waveforms-Characteristics

- Period Vs Frequency

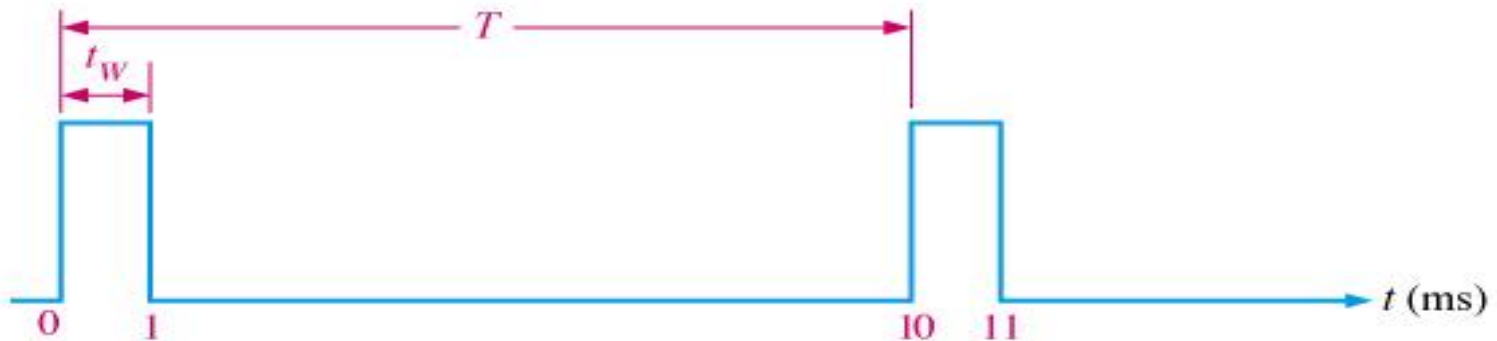
- Frequency (f) is the rate at which it repeats itself - measured in cycles per second or Hertz (Hz)
- Period (T) is the time required for a periodic waveform to repeat itself
 - measured in seconds
- Relationship between frequency and period
$$f = 1/T$$
$$T = 1/f$$

Periodic Digital Waveforms-Characteristics

- Duty Cycle

Duty cycle is the ratio of the pulse width to the period and expressed as a percentage

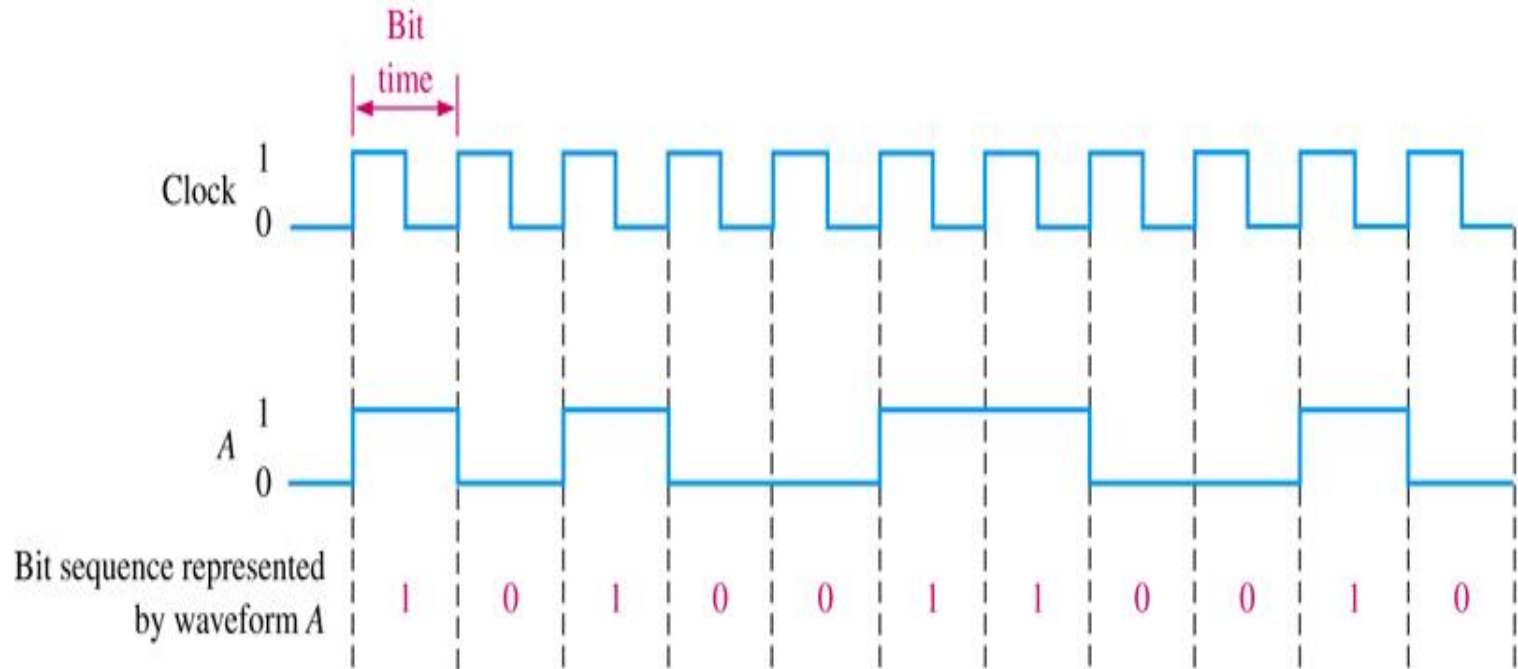
$$\text{Duty cycle} = (t_w / T) \times 100\%$$



Example: For the above periodic waveform, determine the following
a. Period b. Frequency c. duty cycle

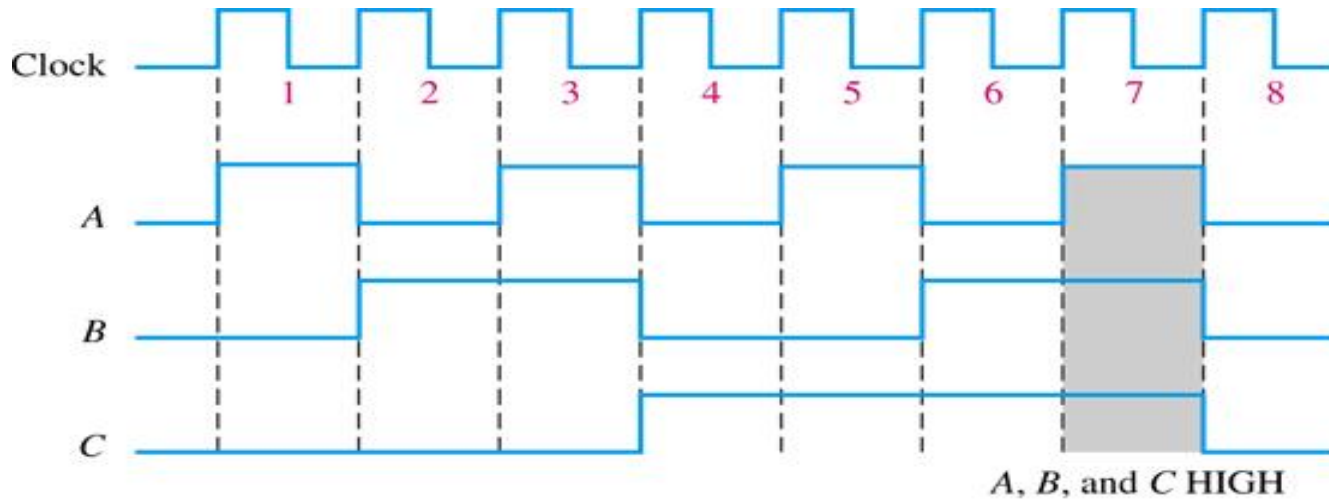
Solution: Period = 10 ms, Frequency = 100 Hz , Duty cycle = 10%

Representation of Bit Sequence



- The clock is a periodic waveform in which each interval between pulses (period) equals the time for one bit (bit time)
- Here waveform A level change occurs at the leading edge of the clock waveform

Timing Diagrams

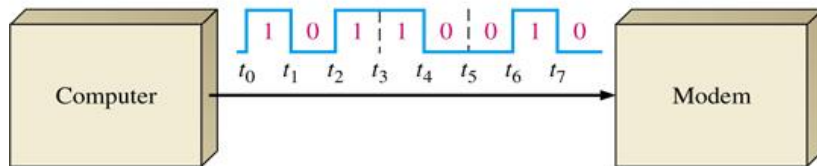


- a graph of digital waveforms showing the actual relationship of two or more waveforms and how each waveform changes in relation to the others
- show voltage versus time.
- Horizontal scale represents regular intervals of time beginning at time zero.
- used to show how digital signals change with time.

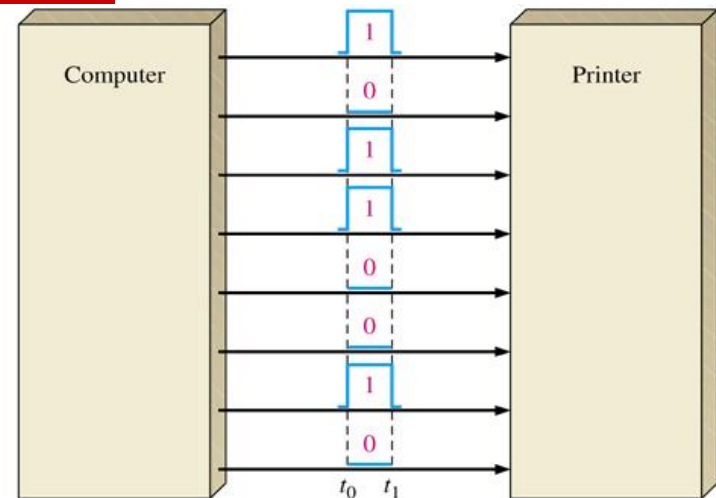
Here waveforms A, B, and C are HIGH only during bit time 7

Binary Data Transfer

- Two types



(a) Serial transfer of 8 bits of binary data from computer to modem. Interval t_0 to t_1 is first.

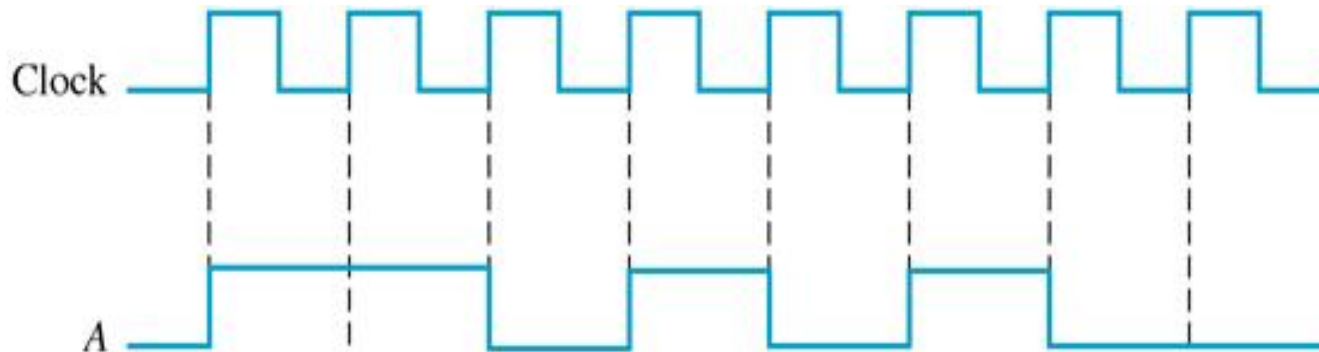


(b) Parallel transfer of 8 bits of binary data from computer to printer. The beginning time is t_0 .

- **Serial Transfer** - Sent one bit at a time along a single line
Advantage: only one line is required
Disadvantage: It takes longer to transfer a given number of bits
- **Parallel Transfer** - all the bits in a group are sent out on separate lines at the same time
Advantage: Speed of transfer – more
Disadvantage: More lines are required

Binary Data Transfer

- Example



Problem:

Determine the total time required to serially transfer the eight bits contained in waveform A and indicate the sequence of bits. The 100kHz is used as reference. What is the total time to transfer the same eight bits in parallel

Solution:

Period $T = 1/f = 10$ microseconds

Total time required for serial transfer = $8 T = 80$ microseconds

Bit Sequence = 11010100

Total time required for parallel transfer = $1T = 10$ microseconds

Problems

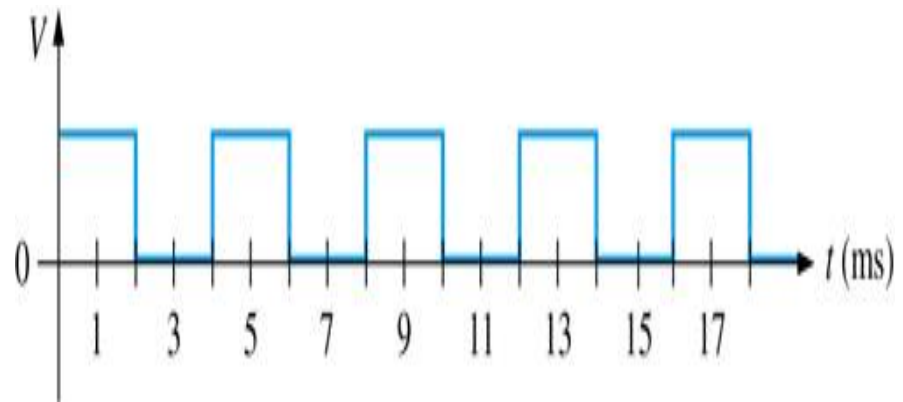
Problem:

Determine the

- Period
- Frequency
- Duty cycle
- Determine the waveform is periodic or non-periodic

Solution:

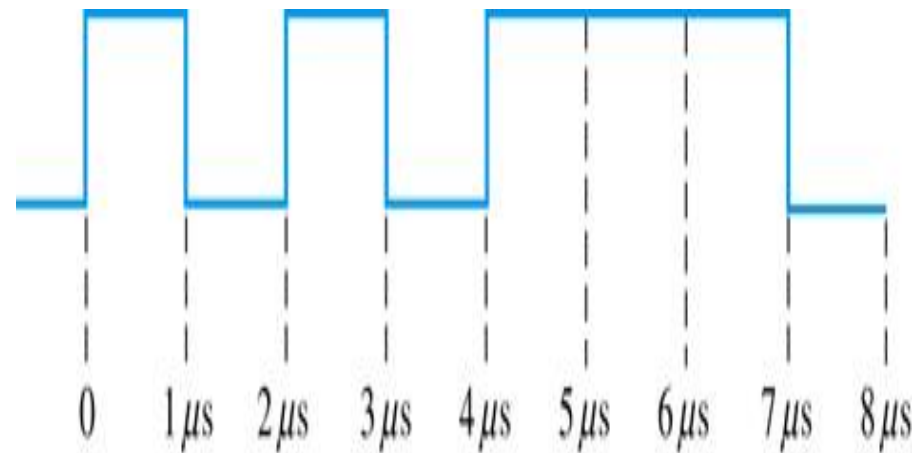
- Period = 4 ms
- Frequency = 250 Hz
- Duty cycle = $(2\text{ms}/4\text{ms}) \times 100\%$
= 50 %
- The waveform is periodic since it repeats at a fixed interval



Problems

Problem:

- Determine the bit sequence
- Determine the total serial transfer time for the eight bits
- Determine the total parallel transfer time



Solution:

- 10101110
- Each bit time = 1 microsecond
Serial transfer time = (8 bits) \times 1 microsecond/bit = 8 microseconds
Parallel transfer time = 1 bit time = 1 microsecond

Introduction to Digital Logic and Boolean Algebra

– Part 2 of 3

Logic Gates

- NOT, AND, OR, NAND, NOR, XOR, XNOR

TOPIC COVERAGE

- PART 2 of 3

- Logic Gates - NOT, AND, OR, NAND, NOR, XOR, XNOR
 - Standard Logic Symbols
 - Truth Tables
 - Logic expression
 - Logical operation
 - Timing Diagram
 - Application examples
- IC Gates
 - DIP, Pin configurations

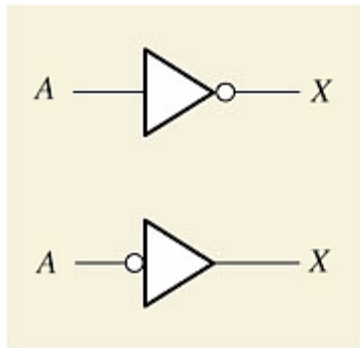
Logic Gates - Introduction

- ❑ Logic gates are the building blocks of computers.
- ❑ Most of the functions in a computer are implemented with logic gates used on a very large scale.
- ❑ For example, a Microprocessor, which is the main part of the computer, is made of up of hundreds of thousands of logic gates

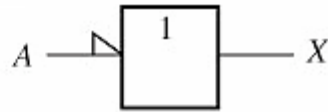
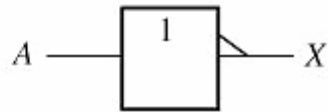
The Inverter (NOT gate)

The Inverter (NOT gate)

- Symbol, Truth Table, Boolean Expression, Logical Operation, Timing Diagram



Distinctive shape symbols



Rectangular outline symbols

Basic Logical Function:

NOT gate can have only one input and performs **logical inversion or complementation**.

Logical operation:

The output of an inverter is always the complement (opposite) of the input.

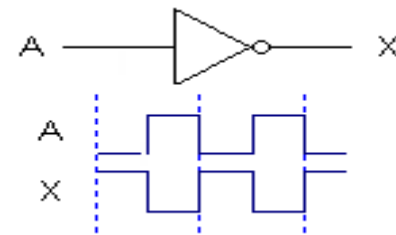
A	X
0	1
1	0

Truth table

0 = LOW
1 = HIGH

$$X = \overline{A}$$

Boolean expression



Pulsed waveforms/
Timing Diagram

The Inverter (NOT gate)

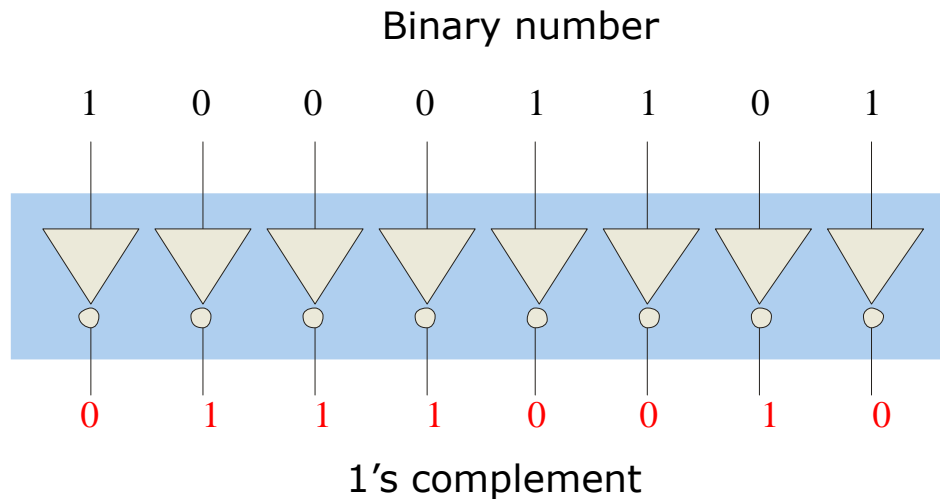
- Problem

The output of the INVERTER is connected to the input of a second INVERTER. Determine the output of the second INVERTER for the input A

The Inverter

- An example application

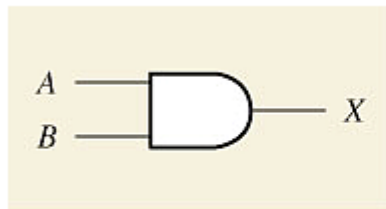
A group of inverters can be used to form the 1's complement of a binary number



The AND Gate

The AND Gate

- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol

Basic Logical Function:

An AND gate can have two or more inputs and performs **logical multiplication**.

Logical Operation:

The output of an AND gate is HIGH only when all inputs are HIGH.

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

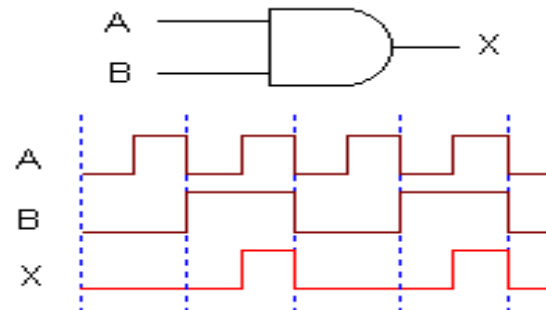
$$X = AB$$

Boolean expression

Truth table

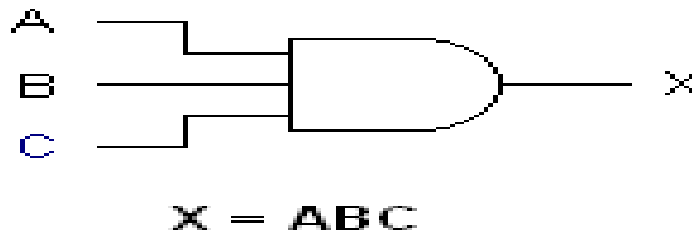
(2-input AND gate)

0 = LOW
1 = HIGH

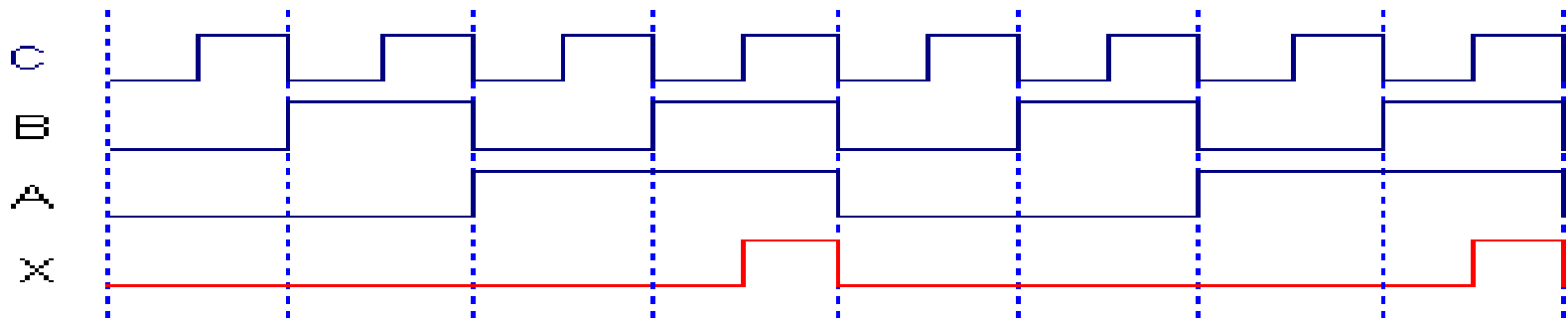


Timing Diagram

The AND gate (3-inputs)

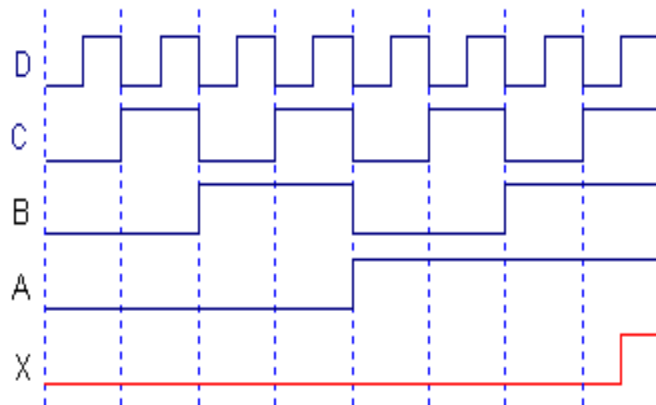
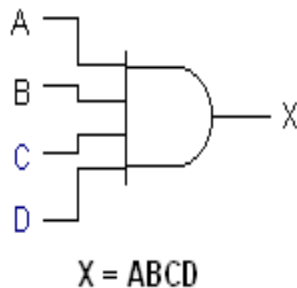


A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



3-Input AND Gate

The AND gate (4-inputs)



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 total number of possible combinations of binary inputs to a gate is determined by the following formula:

$$N = 2^n$$

where N is the number of possible input combinations and n is the number of input variables.

For two input variables:

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

For three input variables:

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

For four input variables:

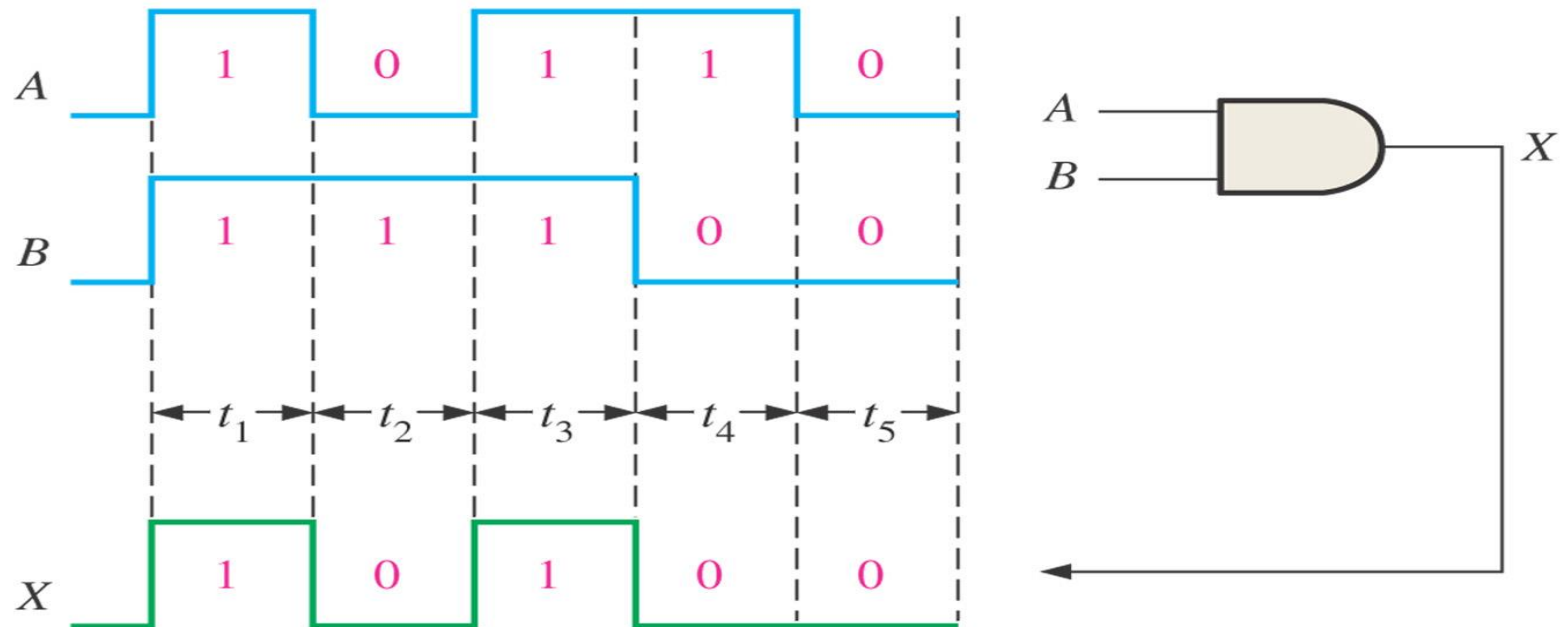
$$N = 2^4 = 16 \text{ combinations}$$

4-Input AND Gate

The AND gate

- Problem 1

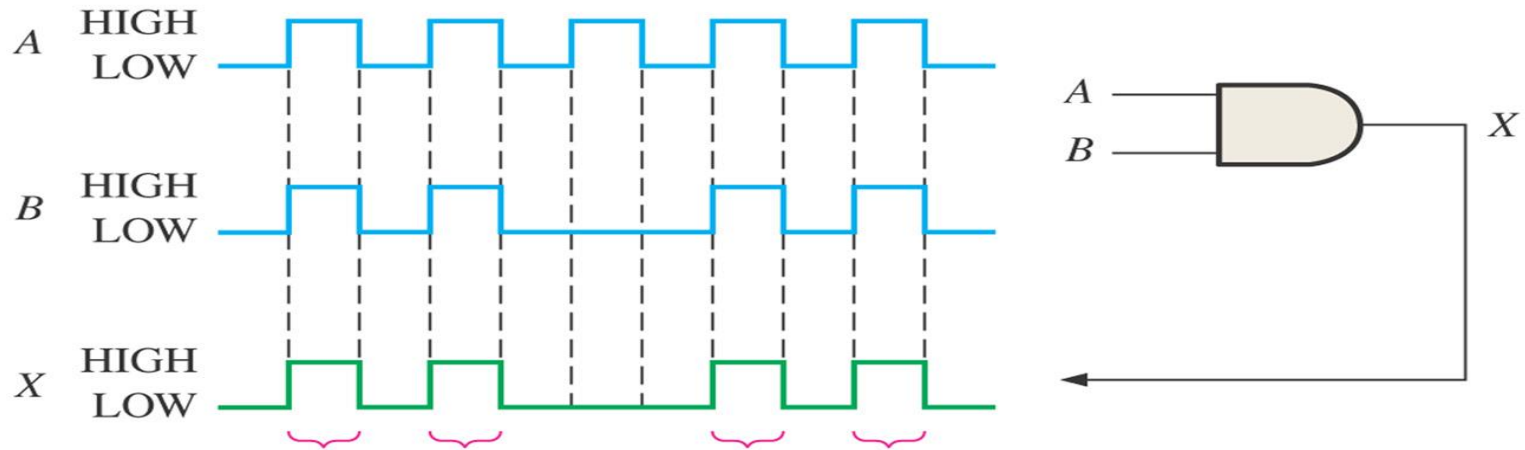
If two waveforms, A and B are applied to AND gate inputs as shown in figure below, what is the resulting output waveform?



The AND gate

- Problem 2

If two waveforms, A and B are applied to AND gate inputs as shown in figure below, what is the resulting output waveform?

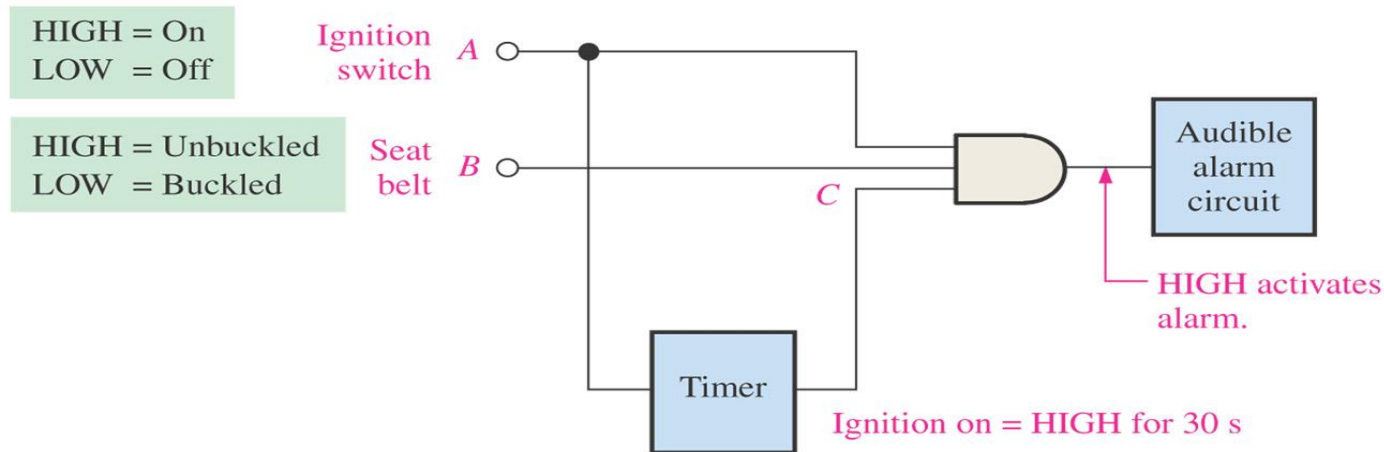


A and B are both HIGH during these four time intervals.
Therefore X is HIGH.

The AND gate

- An example application

- A common application of the AND gate is to enable (to allow) the passage of a signal (pulse waveform) from one point to another at certain times and to inhibit (prevent) the passage at other times.
- Example : A seat belt Alarm system



An AND gate is used to detect when the ignition switch is ON and the seat belt is unbuckled. If the ignition switch is on, a HIGH is produced on input A of the AND gate. If the seat belt is not properly buckled, a HIGH is produced on input B. Also, when the ignition switch is turned on, a timer is started that produces a HIGH on input C for 30s. If all three conditions exist- that is, if the ignition is on and the seat belt is unbuckled and timer is running- the output of AND gate is HIGH and audible alarm is energized to remind the driver.

The AND gate

- An example application

- 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.

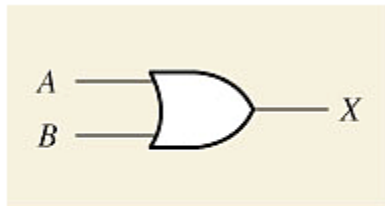
Example: If the binary number 10100011 is ANDed with the mask 00001111, what is the result?

Ans: 00000011

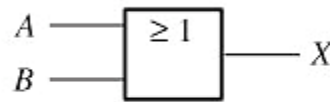
The OR Gate

The OR Gate

- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol

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

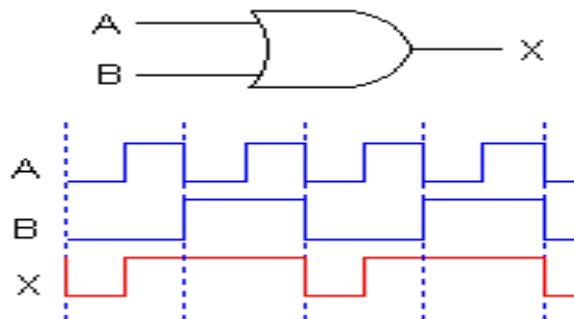
Truth table

(2-input OR gate)

0 = LOW
1 = HIGH

$$X = A + B$$

Boolean expression



Timing Diagram

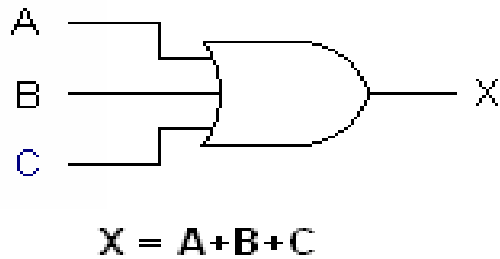
Basic Logical Function:

The OR gate can have two or more inputs and performs **logical addition**.

Logical Operation:

The output of an OR gate is LOW only when all inputs are LOW.

The OR gate (3-inputs)

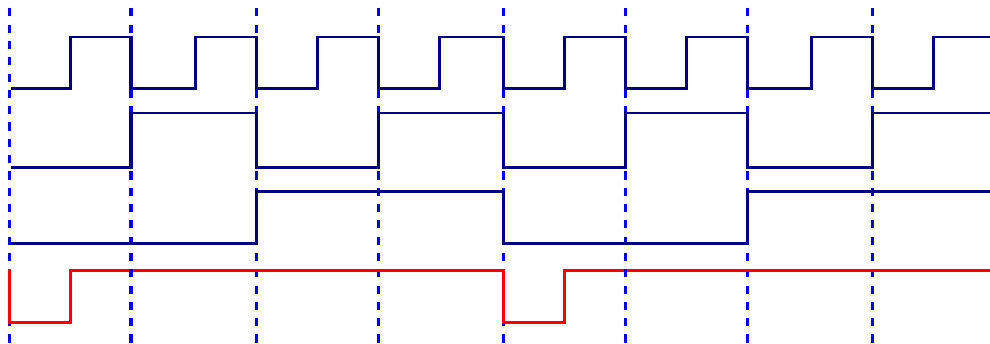


A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Note:

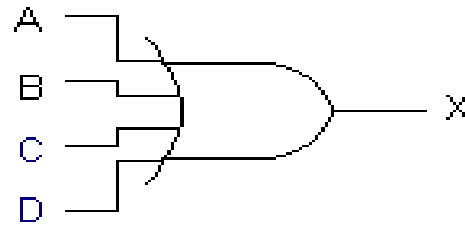
Boolean addition differs from binary addition in the cases where two or more 1s are added.

There is no carry in Boolean addition.

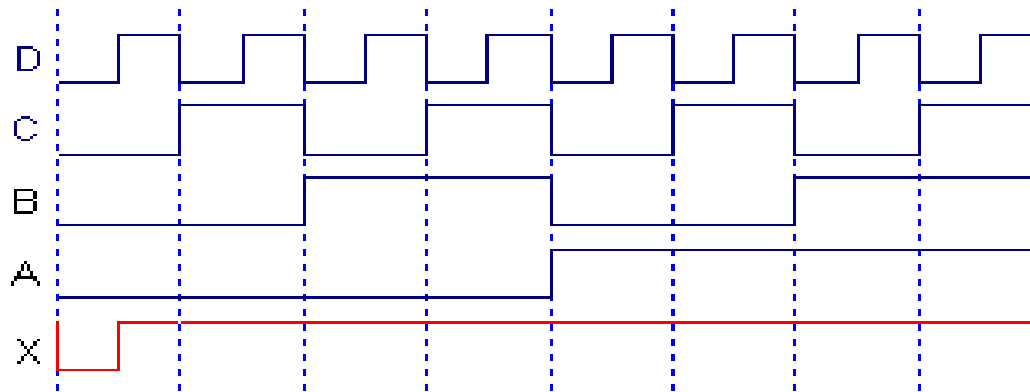


3-Input OR Gate

The OR gate (4-inputs)



$$X = A + B + C + D$$



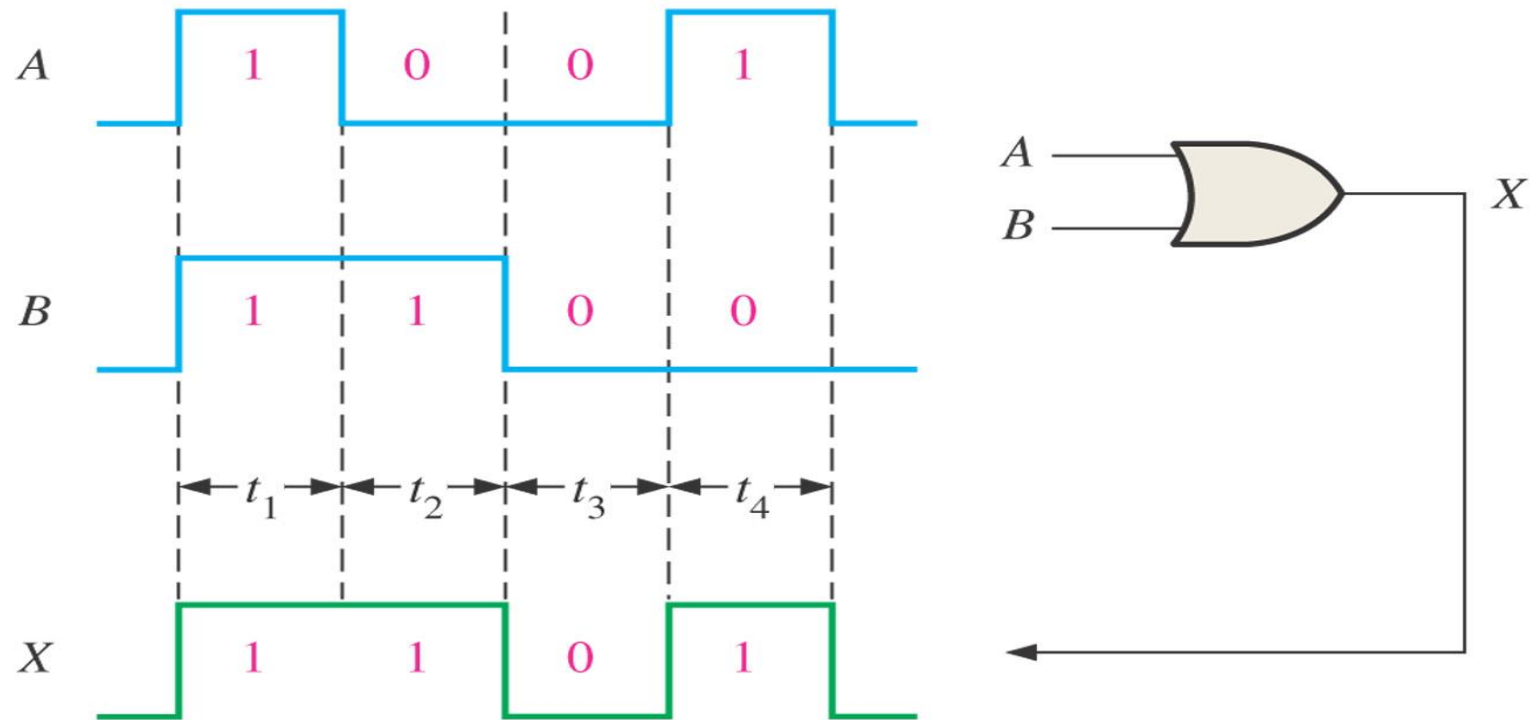
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4-Input OR Gate

The OR gate

- Problem 1

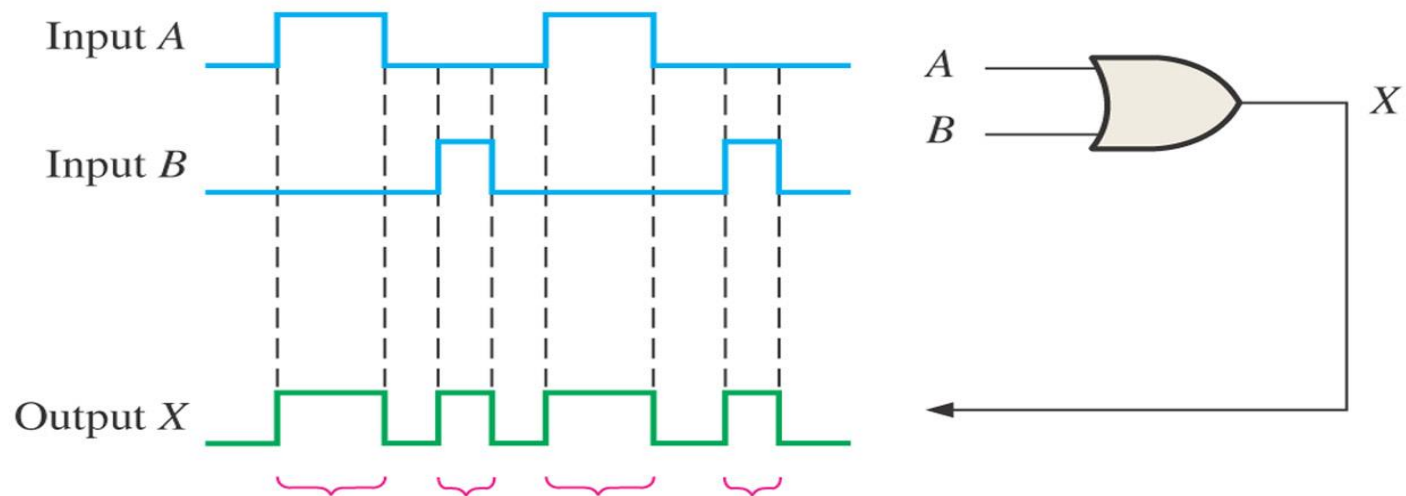
If two waveforms, A and B are applied to OR gate inputs as shown in figure below, what is the resulting output waveform?



The OR gate

- Problem 2

If two waveforms, A and B are applied to OR gate inputs as shown in figure below, what is the resulting output waveform?



When either input or both inputs are HIGH,
the output is HIGH.

The OR gate

- An example application

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

Example: What will be the result if you OR an ASCII upper case letter with the binary number 010 0000?

Ans: The resulting letter will be ASCII lower case.

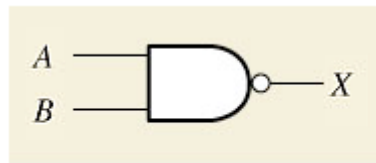
For example,

Take the 7 bit ASCII code for A = 100 0001 (OR) 010 0000 = 110 0001 (a)
7 bit ASCII code for Z = 101 1010 (OR) 010 0000 = 111 1010 (z)

The NAND Gate

The NAND Gate

- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol

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

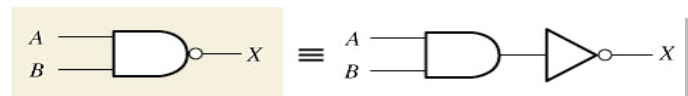
Truth table

(2-input NAND Gate)

0 = LOW
1 = HIGH

$$X = \overline{AB}$$

Boolean expression

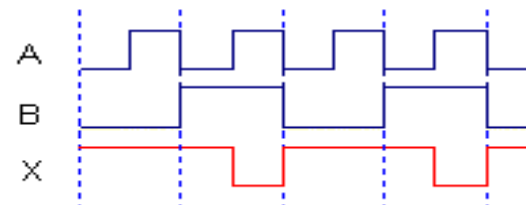
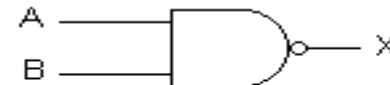


Basic Logical Function:

The NAND gate can have two or more inputs and performs **inverse** of **logical multiplication**.

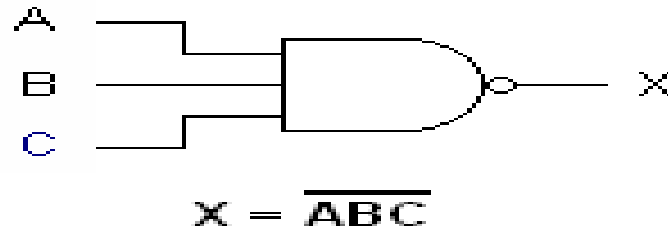
Logical Operation:

The output of a NAND gate is **LOW** only when all inputs are **HIGH**

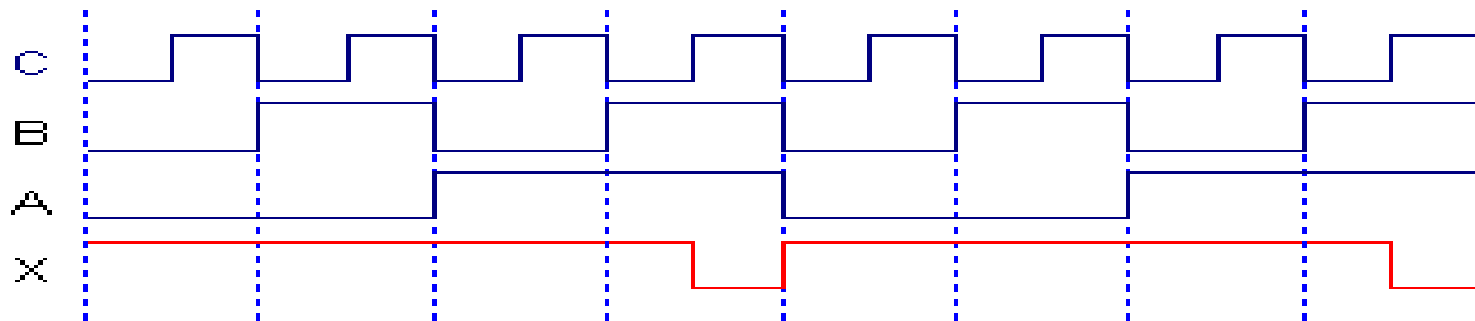


Timing Diagram

The NAND gate (3-inputs)

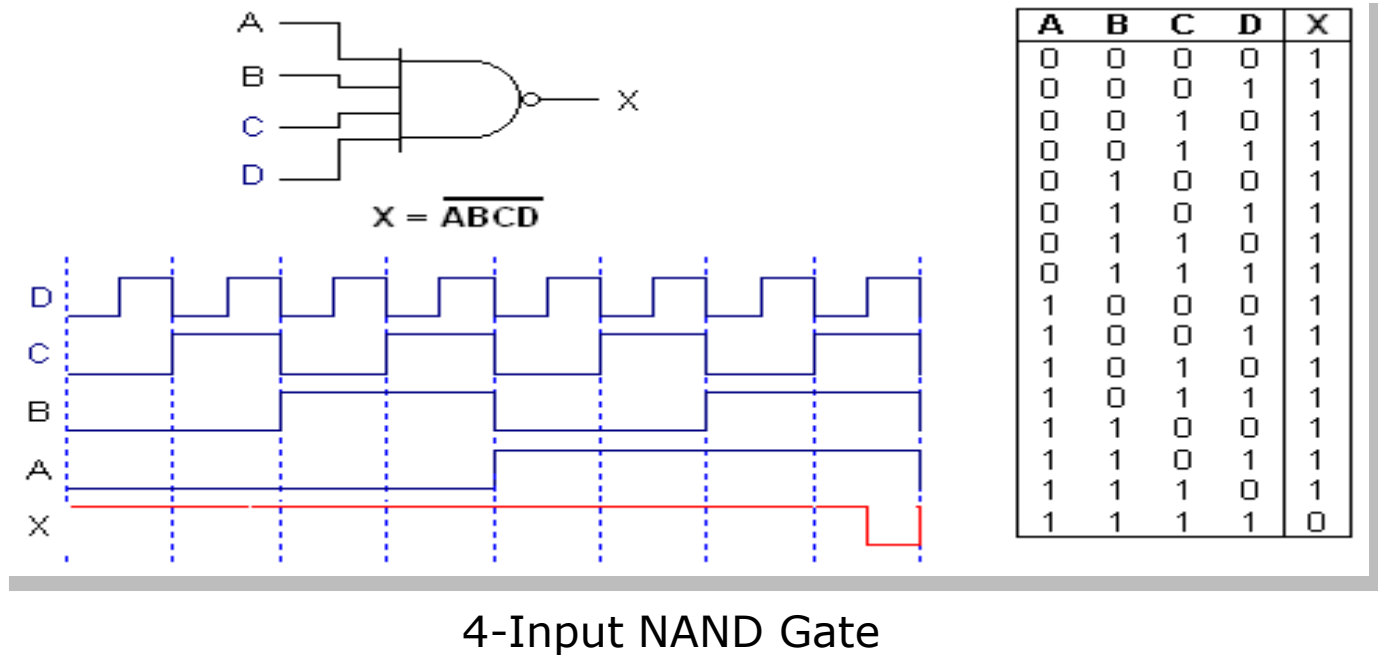


A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



3-Input NAND Gate

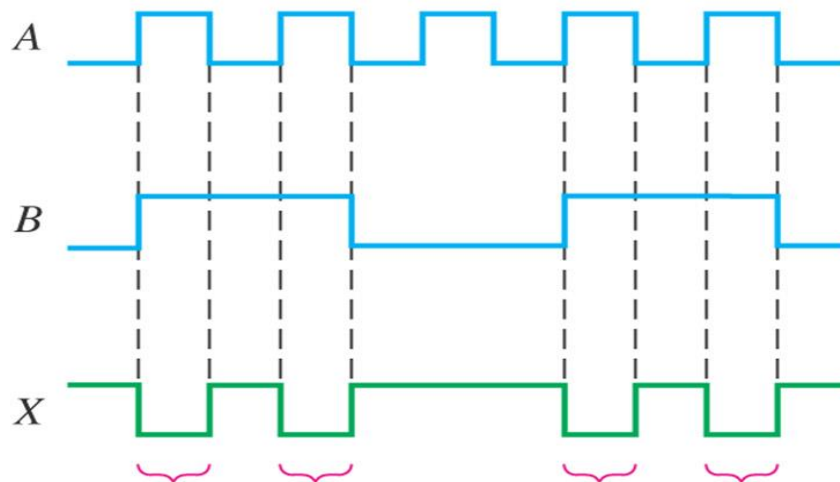
The NAND gate (4-inputs)



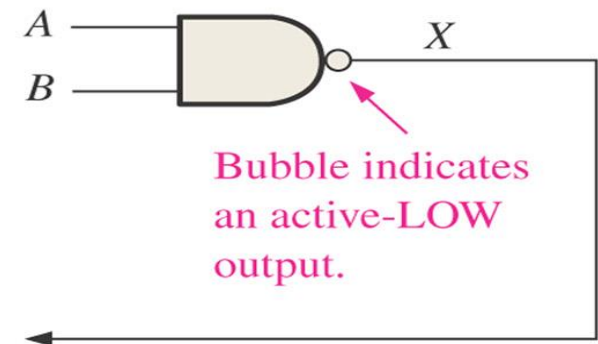
The NAND gate

- Problem

If two waveforms, A and B are applied to NAND gate inputs as shown in figure below, what is the resulting output waveform?



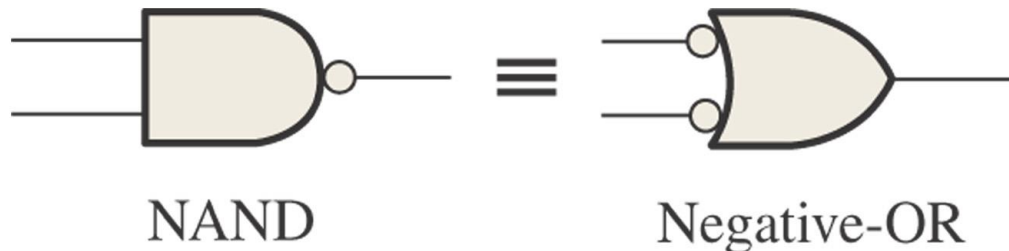
A and B are both HIGH during these four time intervals. Therefore X is LOW.



The NAND Gate

- Negative-OR Equivalent operation

- A NAND gate produces HIGH output, when one or more inputs are LOW.
- From this view point, a NAND gate can be used for an OR operation that requires one or more LOW inputs to produce a HIGH output. This aspect of NAND operation is referred to as **negative-OR**

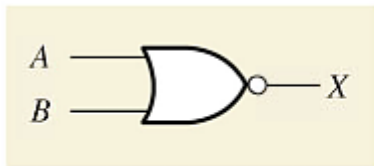


Standard symbols representing the two equivalent operations of a NAND gate

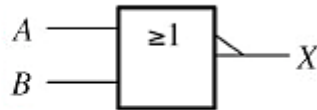
The NOR Gate

The NOR Gate

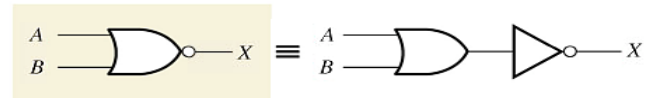
- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol



Basic Logical Function:

The NOR gate can have two or more inputs and performs **inverse** of **logical addition**.

Logical Operation:

The output of a NOR gate is HIGH only when all inputs are LOW

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

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

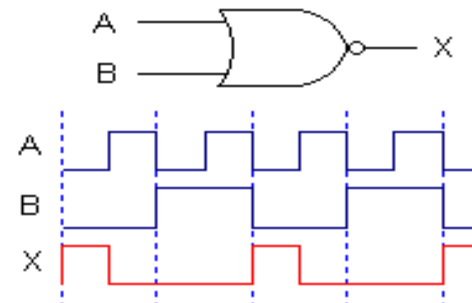
Boolean expression

Truth table

(2-input NOR Gate)

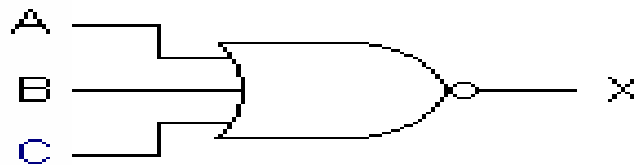
0 = LOW

1 = HIGH



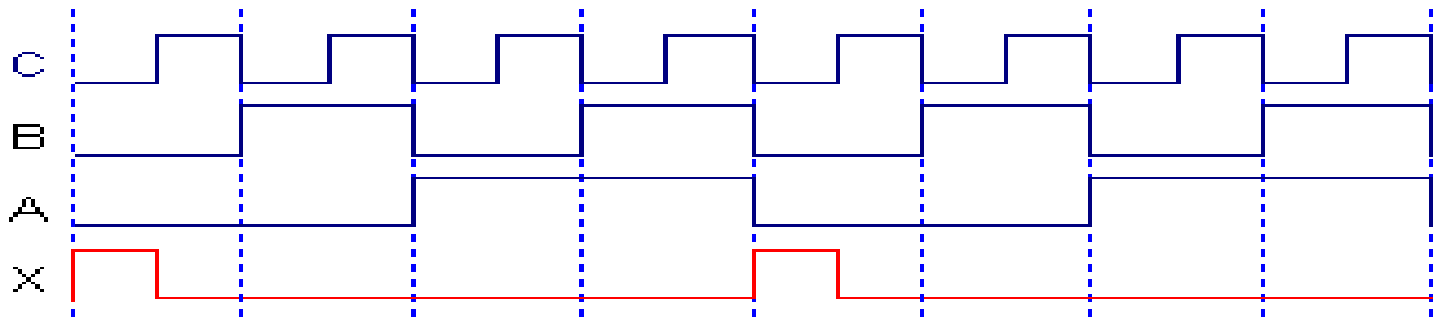
Timing Diagram

The NOR gate (3-inputs)



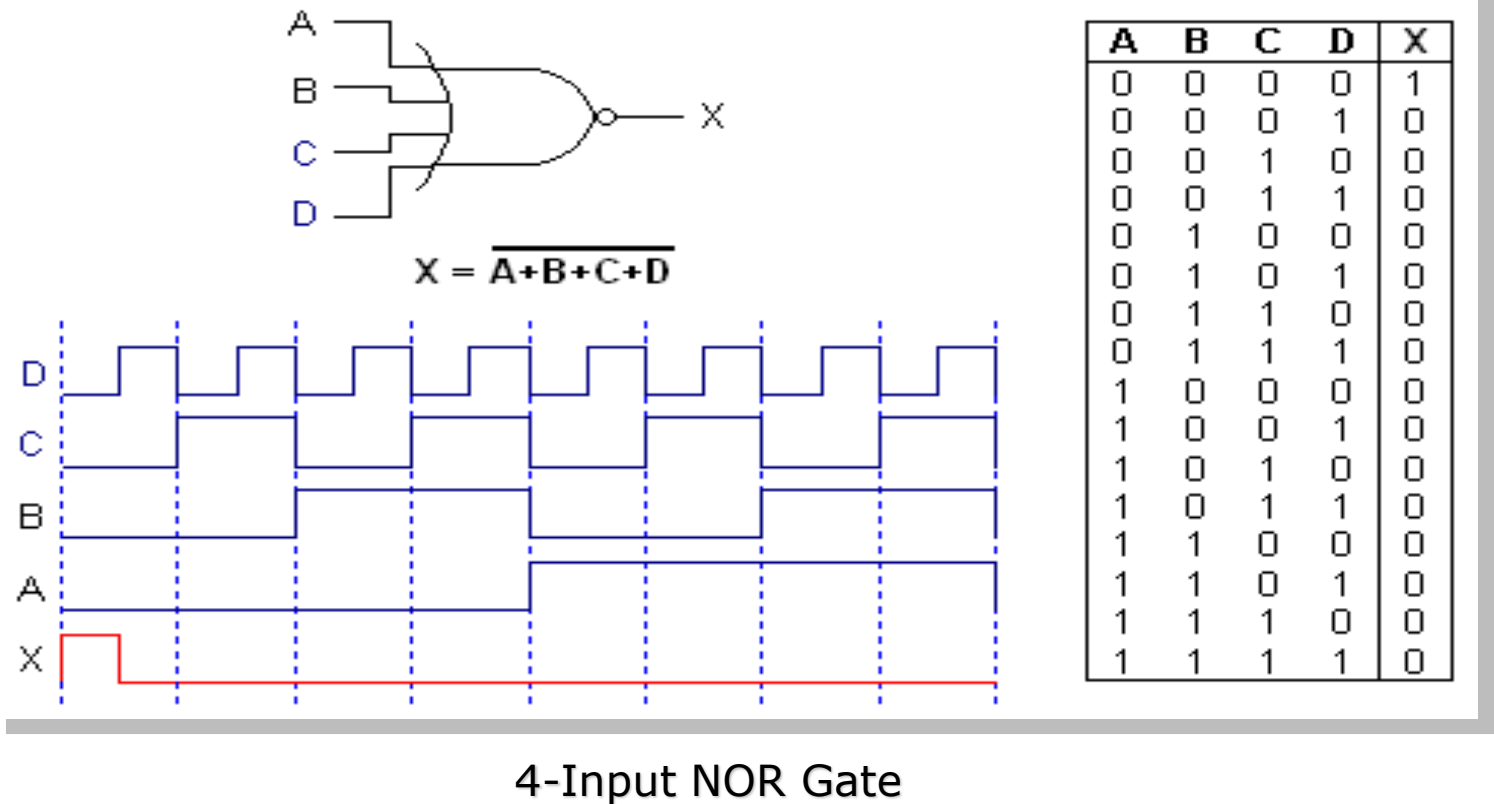
$$X = \overline{A+B+C}$$

A	B	C	X
0	0	0	1
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	0



3-Input NOR Gate

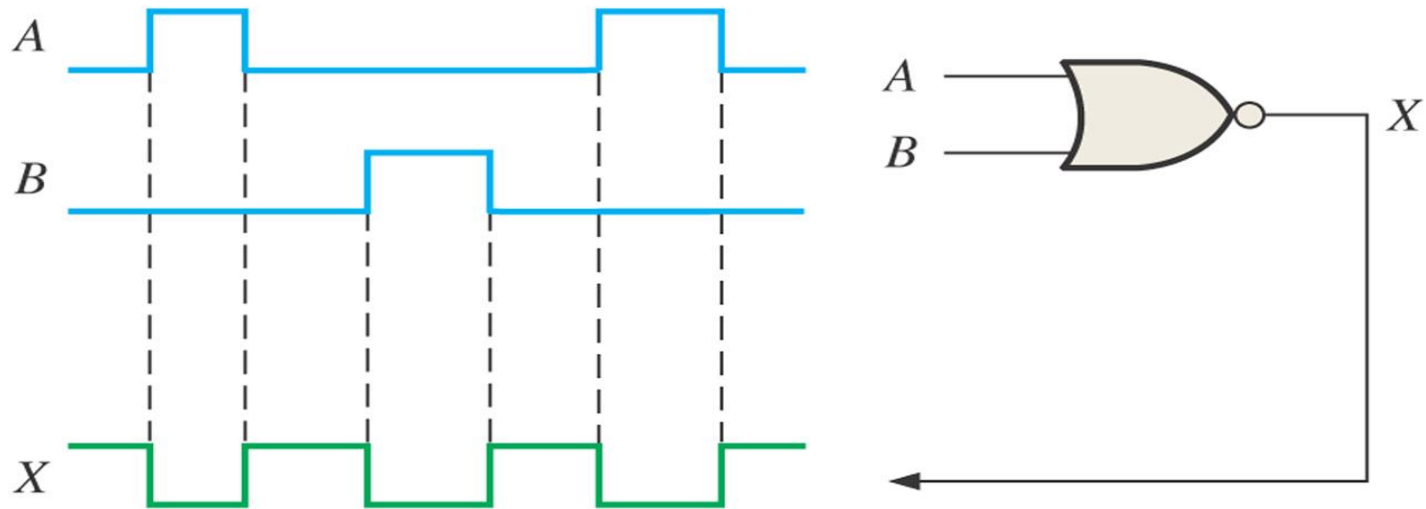
The NOR gate (4-inputs)



The NOR gate

- Problem

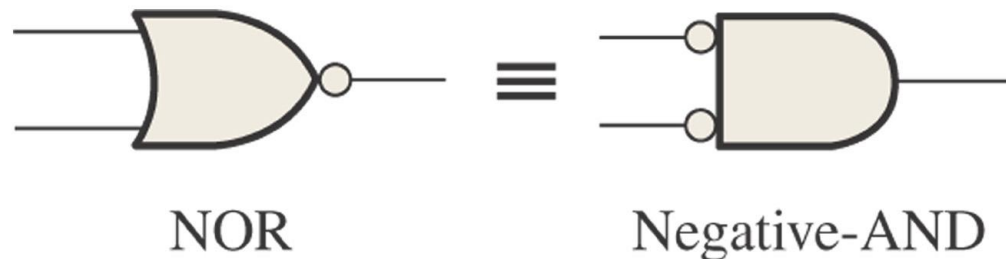
If two waveforms, A and B are applied to NOR gate inputs as shown in figure below, what is the resulting output waveform?



The NOR Gate

- Negative-AND Equivalent operation

- ❑ A NOR gate produces HIGH output, when all inputs are LOW.
- ❑ From this view point, a NOR gate can be used for an AND operation that requires all LOW inputs to produce a HIGH output. This aspect of NOR operation is referred to as **negative-AND**

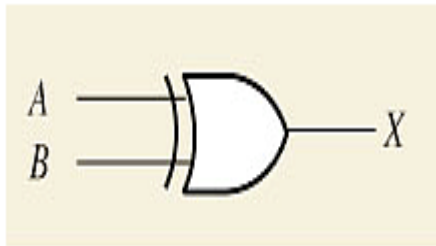


Standard symbols representing the two equivalent operations of a NOR gate

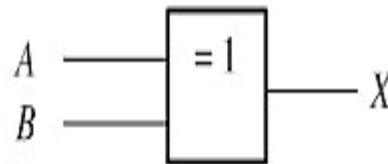
The XOR Gate

The XOR (Exclusive-OR) Gate

- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol

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

Truth table

(2-input XOR gate)

0 = LOW
1 = HIGH

$$X = A \oplus B$$

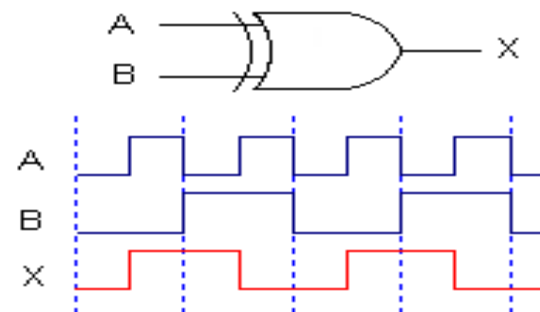
Boolean expression

Basic Logical Function:

The XOR gate can have two or more inputs and performs **ODD function** (output is equal to 1 if the input variables have an odd number of 1's).

Logical Operation:

The output of XOR gate is HIGH whenever the two inputs are different (2-input XOR gate)

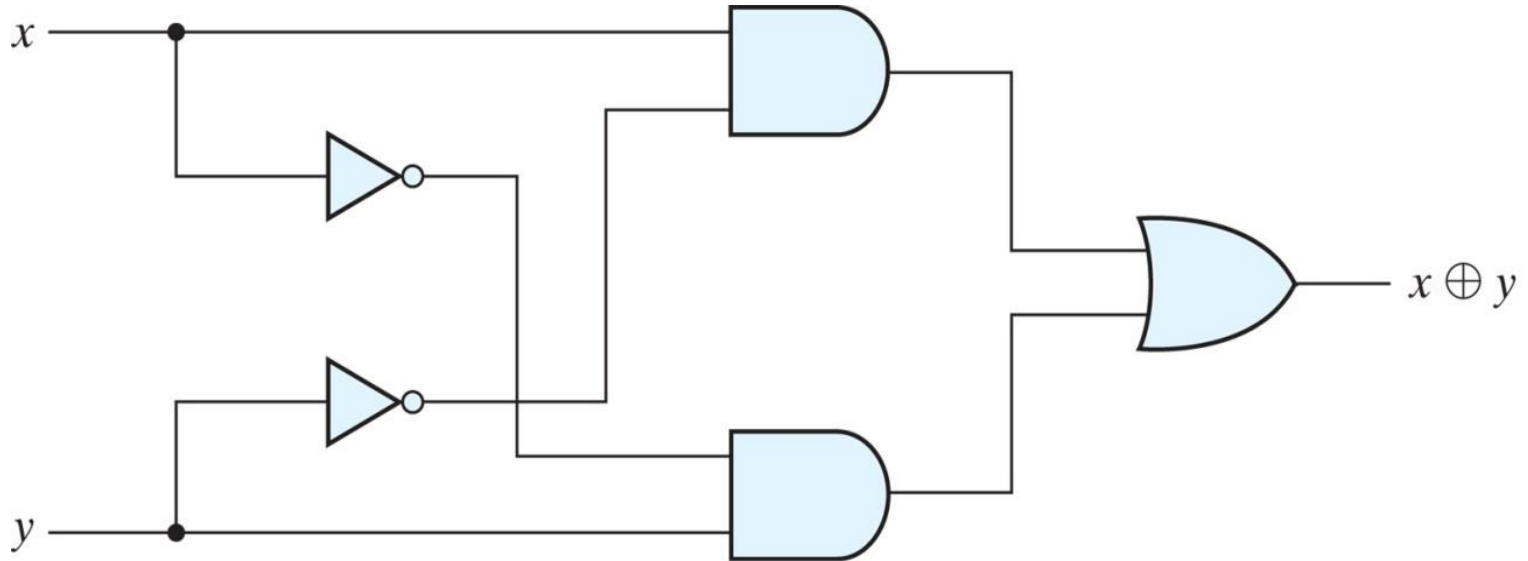


Timing Diagram

The XOR gate (2-inputs)

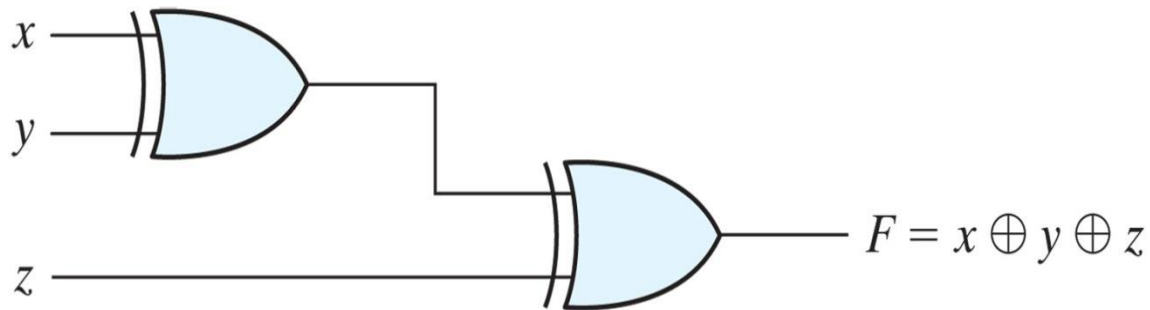
- using AND-OR-NOT Gates

$$x \oplus y = \bar{x}y + x\bar{y}$$

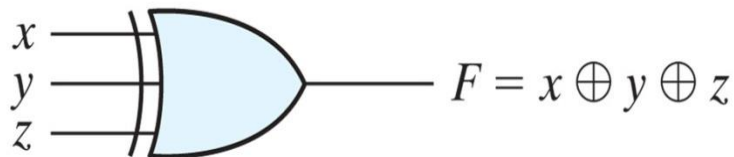


(a) Exclusive-OR with AND-OR-NOT gates

The XOR gate (3-inputs)



(a) Using 2-input gates



(b) 3-input gate

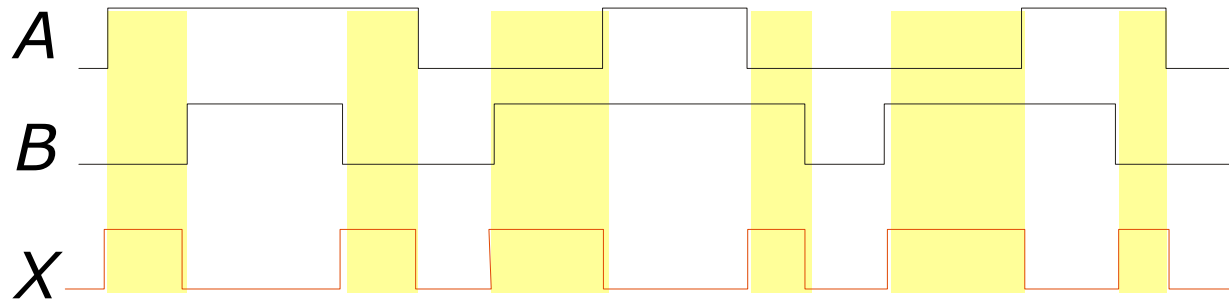
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

The XOR gate

- Problem

If two waveforms, A and B are applied to XOR gate inputs as shown in figure below, what is the resulting output waveform?



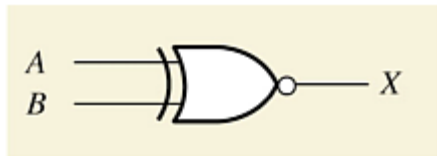
If the A and B waveforms are both inverted for the above waveforms, how is the output affected?

Ans: There is no change in the output.

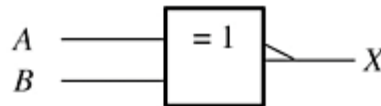
The XNOR Gate

The XNOR (Exclusive-NOR) Gate

- Symbol, Truth Table, Boolean Expression, Logical operation, Timing Diagram



Distinctive shape symbol



Rectangular outline symbol

Basic Logical Function:

The XNOR gate can have two or more inputs and performs **EVEN function** (output is equal to 1 if the input variables have an even number of 1's).

Logical Operation:

The output of XNOR gate is HIGH whenever the two inputs are identical (2-input XNOR gate - can be called as equivalence gate)

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

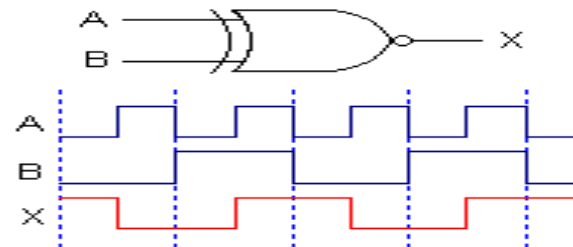
$$X = \overline{A \oplus B}$$

Boolean expression

Truth table

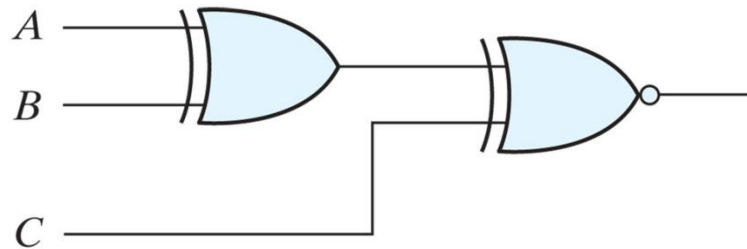
(2-input XNOR Gate)

0 = LOW
1 = HIGH



Timing Diagram

The XNOR gate (3-inputs)



3-input even function

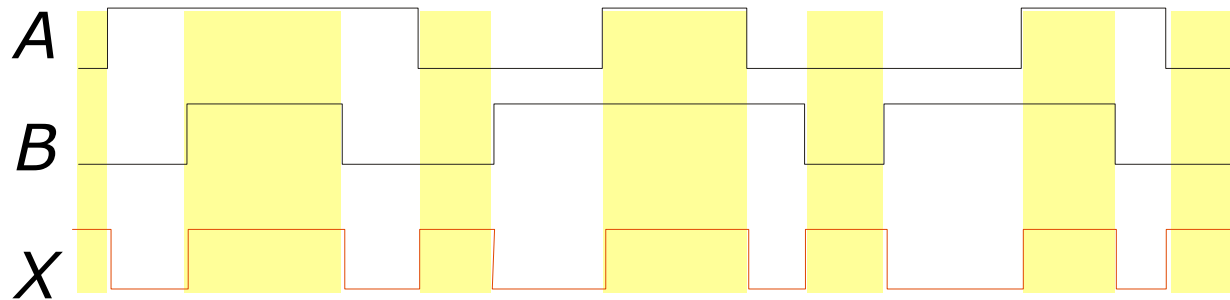
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table

The XNOR gate

- Problem


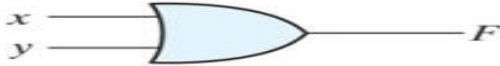



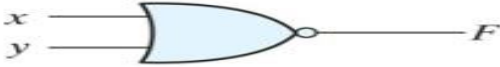
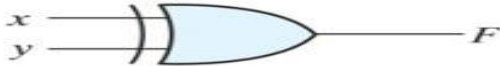

If two waveforms, A and B are applied to XNOR gate inputs as shown in figure below, what is the resulting output waveform?



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

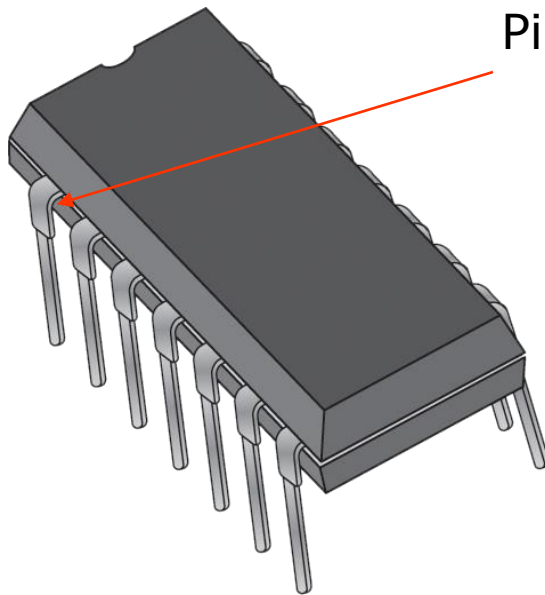
Ans: The output will be inverted

Basic Logic Gates (Summary)

Name	Graphic symbol	Algebraic function	Truth table															
AND		$F = x \cdot y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inverter		$F = x'$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
Buffer		$F = x$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	
NAND		$F = (xy)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

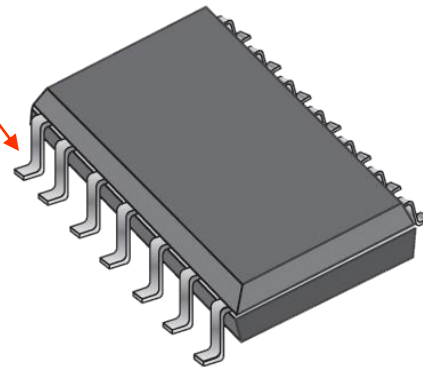
Integrated Circuit (IC) Packages

DIP and surface mount chips



Dual in-line package

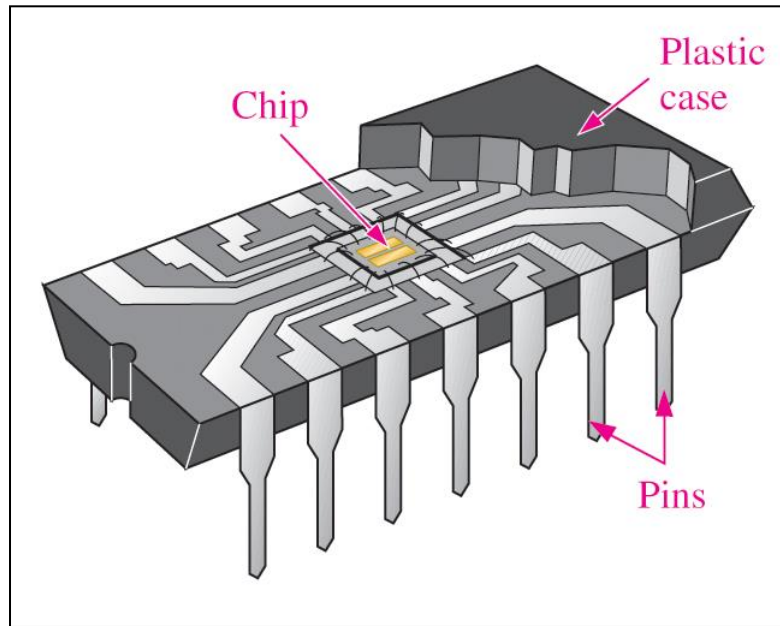
Pin 1



Small outline IC (SOIC)

Dual-In-line Package Chip

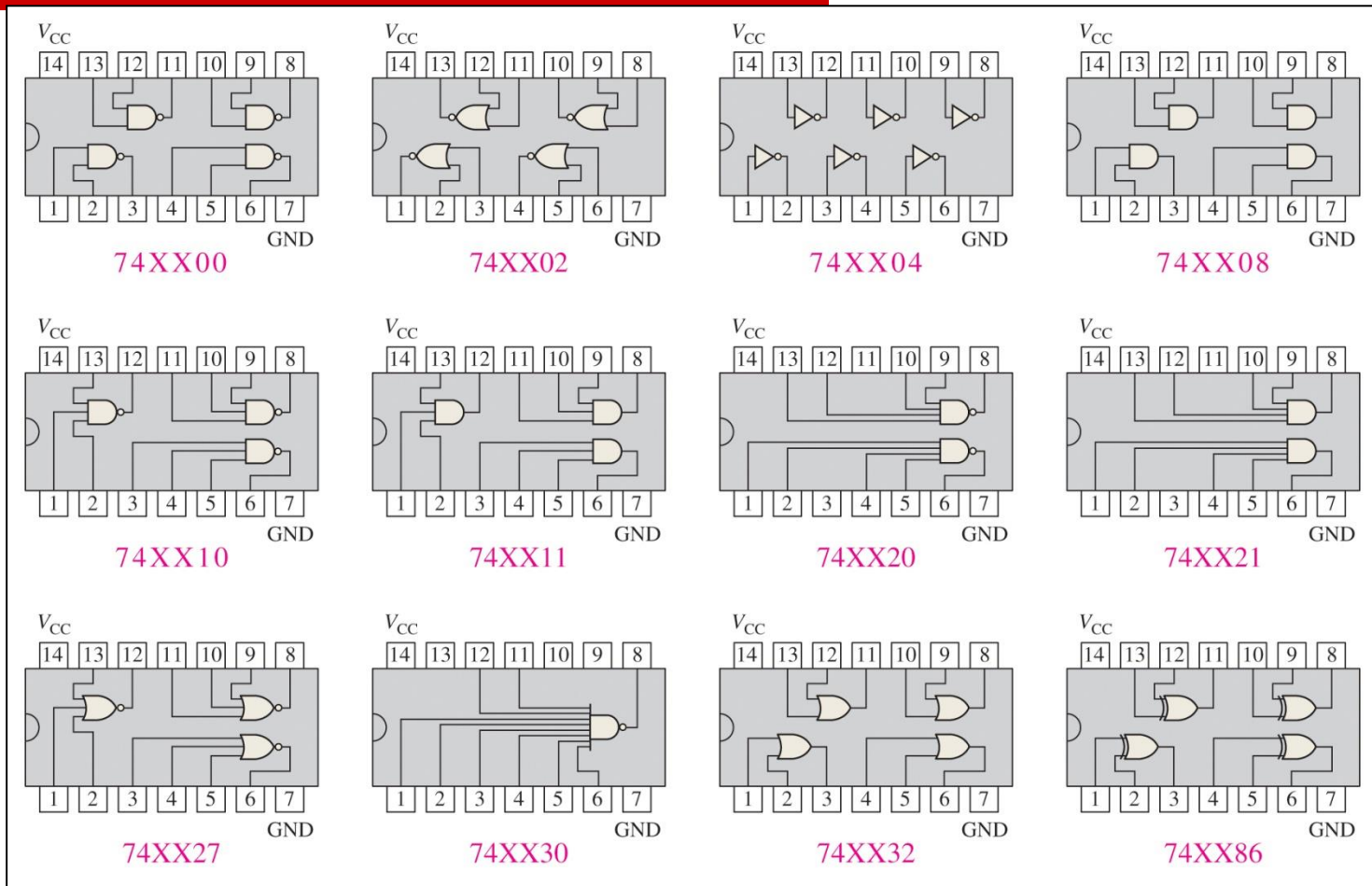
Cutaway view of a DIP (**D**ual-**I**n-line **P**ackage) chip:



The TTL series, available as DIPs, are popular for laboratory experiments with logic.

Pin configuration diagrams

- Common fixed-function IC gates



Introduction to Digital Logic and Boolean Algebra

– Part 3 of 3

Boolean Algebra

- **Laws, Rules and Theorems**
 - **Analysis of Logic Circuits**
 - **Simplification**
-

TOPIC COVERAGE

- PART 3 of 3

- Boolean Algebra
 - Laws and Rules of Boolean Algebra
 - Demorgan's theorems
- Boolean Analysis of Logic circuits
 - Evaluation of logic circuit output
 - Constructing a truth table for a logic circuit
- Simplification using Boolean Algebra

BOOLEAN ALGEBRA

- Rules, Laws, and Theorems

Boolean Algebra

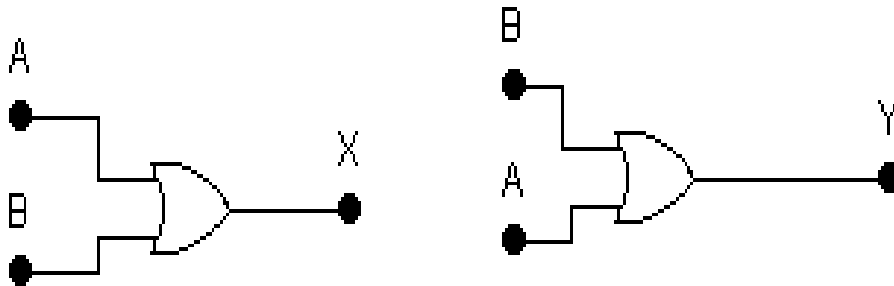
- Boolean algebra is the mathematics of digital systems developed by George Boole in 1854.
- A basic knowledge of Boolean algebra is necessary to the study and analysis of logic circuits.
- Variable, complement and literal are terms used in Boolean algebra.
 - A **variable** is a symbol used to represent logical quantity. Any single variable can have a 1 or a 0 value.
 - The **complement or inverse of a variable** is indicated by bar or prime symbol
 - A **literal** is a variable or its complement.

Laws of Boolean Algebra

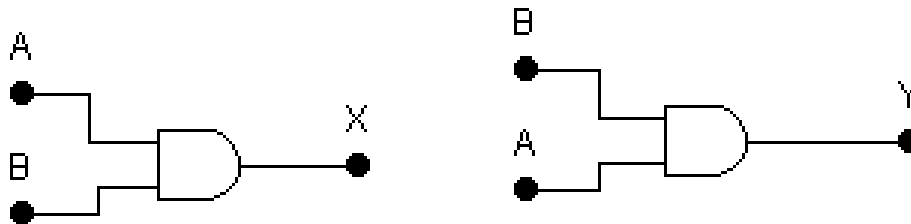
- ☐ Commutative Law
- ☐ Associative Law
- ☐ Distributive Law

Laws of Boolean Algebra

□ **Commutative Law** of Addition: $A + B = B + A$

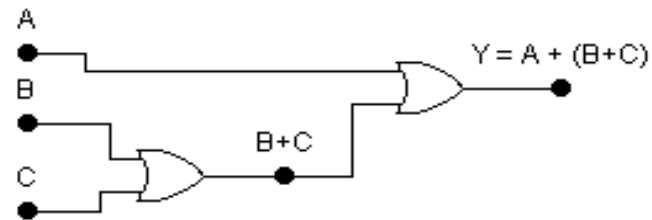
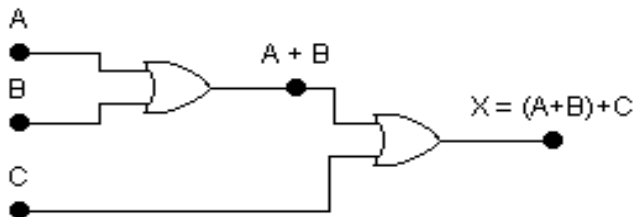


□ **Commutative Law** of Multiplication: $A \cdot B = B \cdot A$

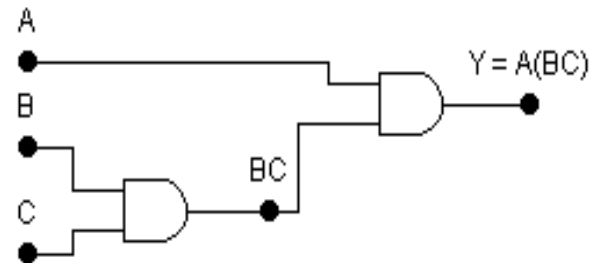
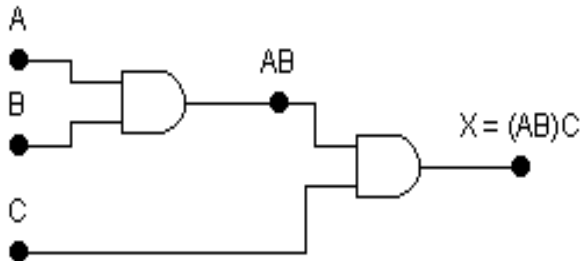


Laws of Boolean Algebra

□ **Associative Law** of Addition: $A + (B + C) = (A + B) + C$



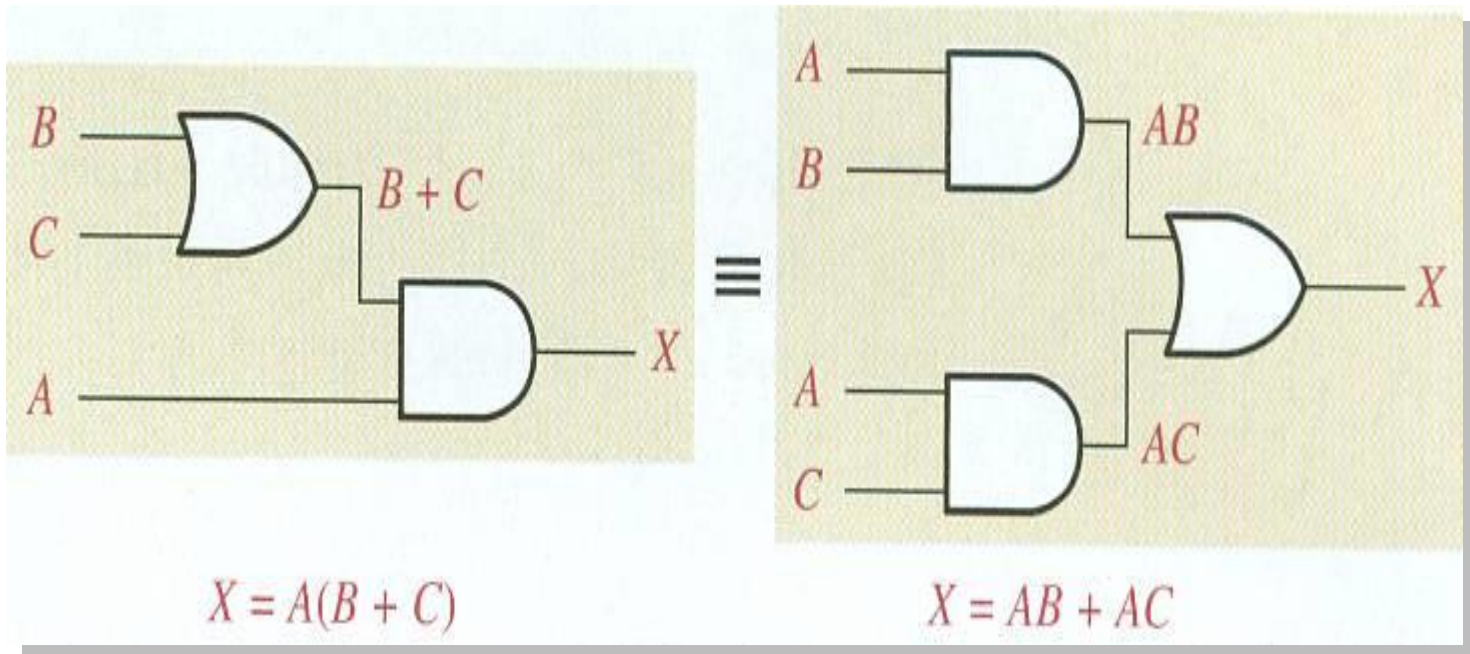
□ **Associative Law** of Multiplication: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$



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. $A + \bar{A} = 1$

7. $A \cdot A = A$

8. $A \cdot \bar{A} = 0$

9. $\bar{\bar{A}} = A$

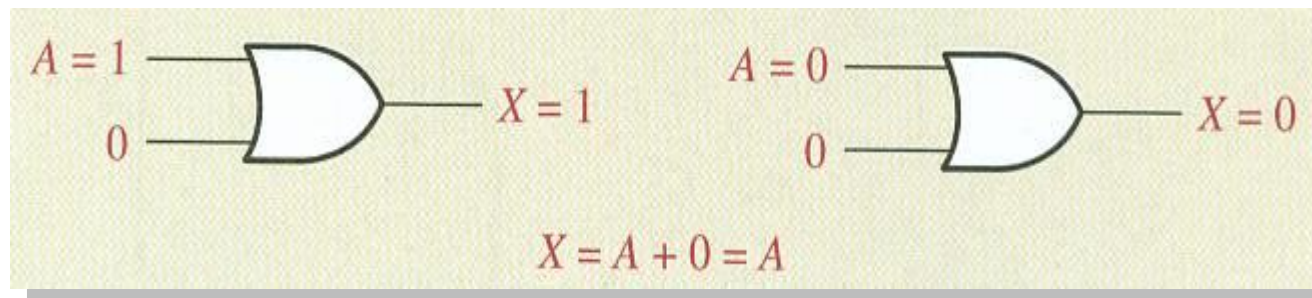
10. $A + AB = A$

11. $A + \bar{A}B = A + B$

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

Rules of Boolean Algebra

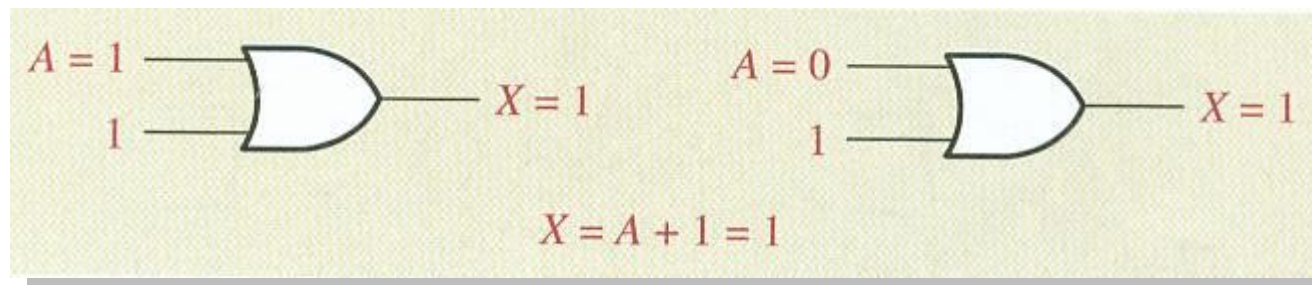
□ Rule 1



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

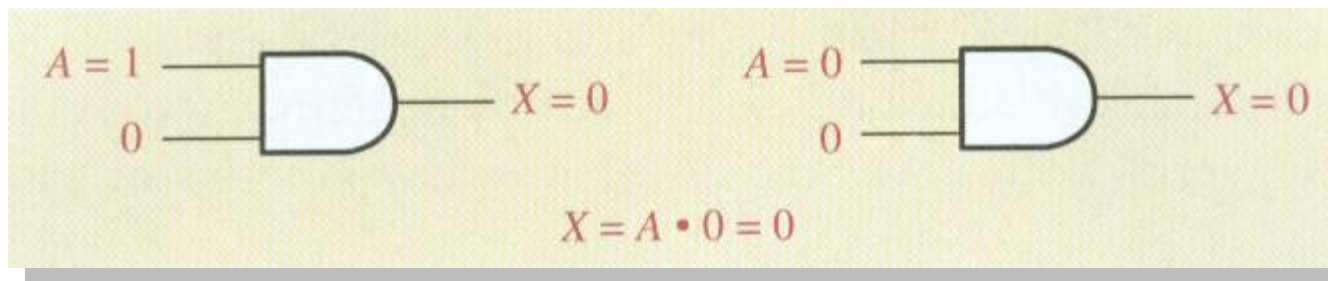
OR Truth Table

□ Rule 2



Rules of Boolean Algebra

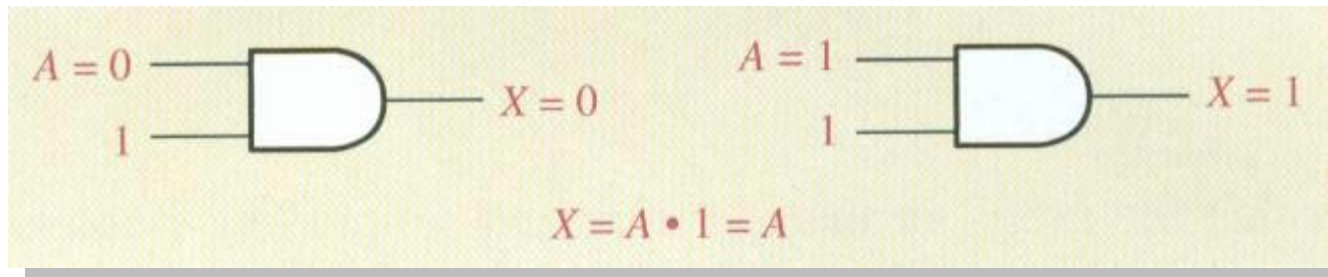
□ Rule 3



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

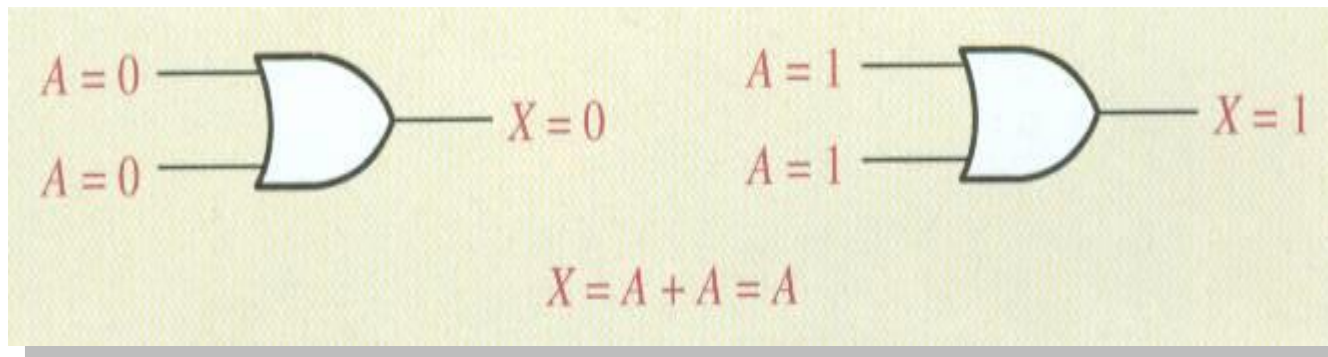
AND Truth Table

□ Rule 4



Rules of Boolean Algebra

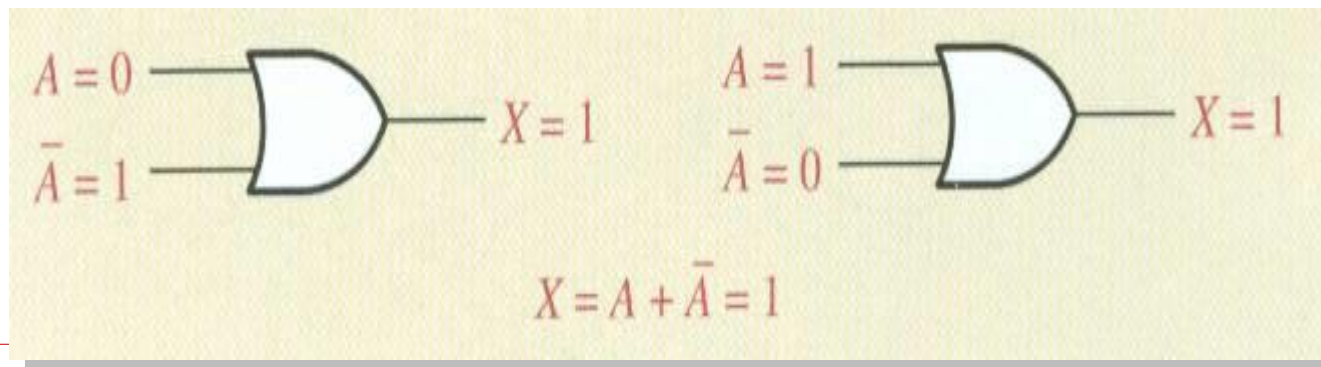
□ Rule 5



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

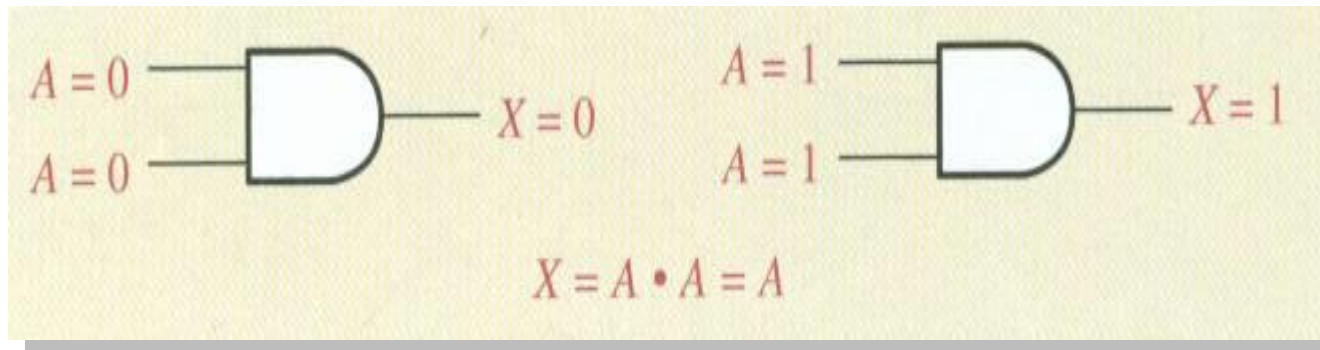
OR Truth Table

□ Rule 6



Rules of Boolean Algebra

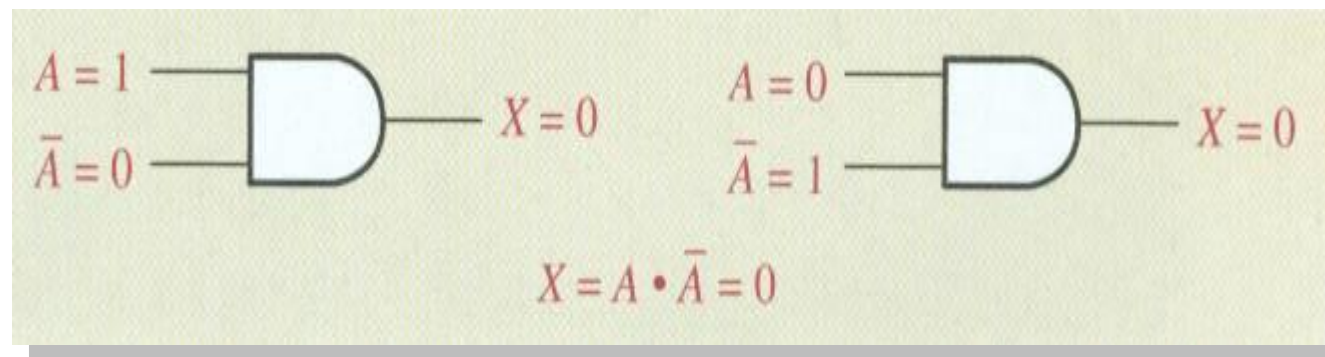
□ Rule 7



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

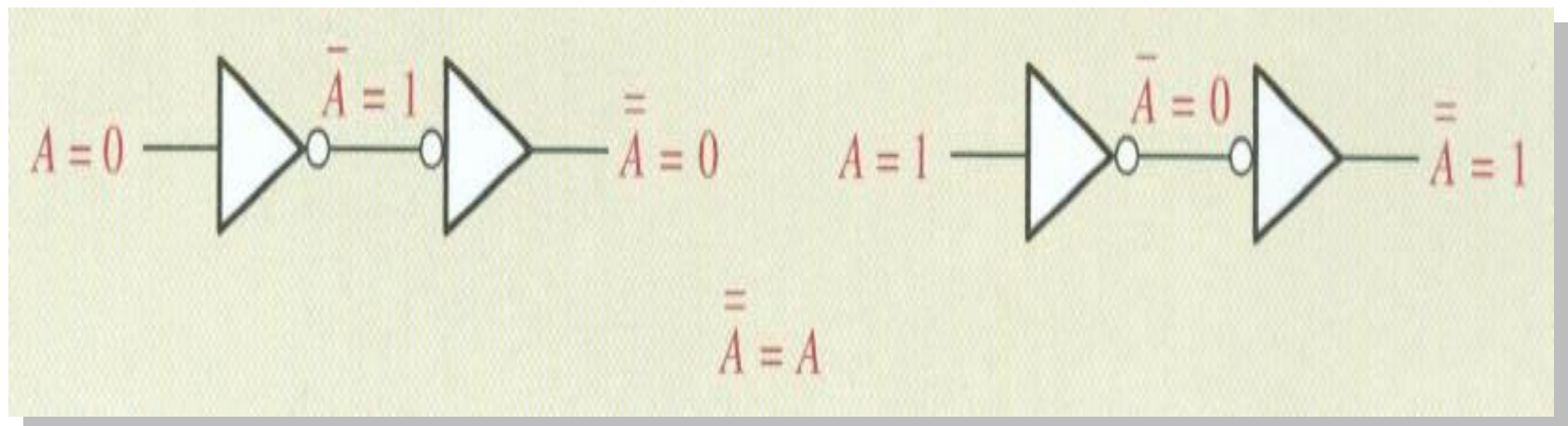
AND Truth Table

□ Rule 8



Rules of Boolean Algebra

□ Rule 9

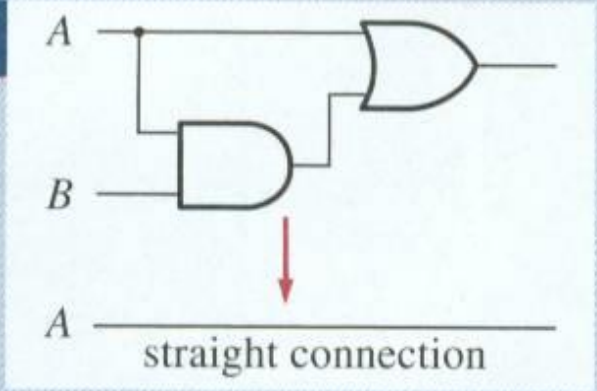


Rules of Boolean Algebra

□ Rule 10: $A + AB = A$

<i>A</i>	<i>B</i>	<i>AB</i>	<i>A + AB</i>
0	0	0	0
0	1	0	0
1	0	0	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

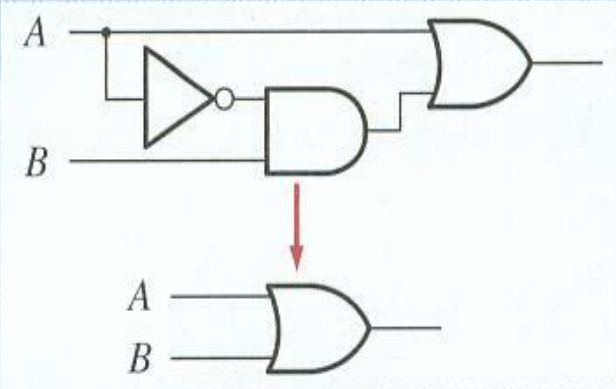
OR Truth Table

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

↑ 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

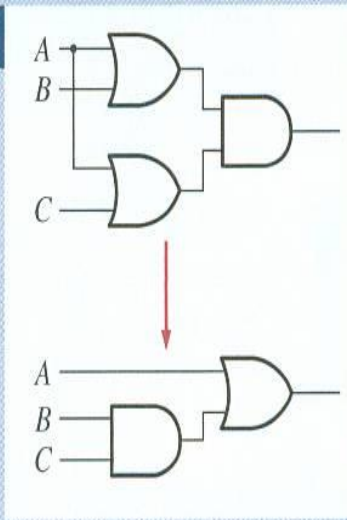
OR Truth Table

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

De Morgan's Theorems

De Morgan's first theorem:

➤ **The complement of a product of variables is equal to the sum of the complements of the variables.**

Stated in another way,

➤ The complement of two or more variables **AND**ed is equivalent to the **OR** of the complements of the individual variables.

➤ The formula for expressing this theorem for two variables is:

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

De Morgan's Theorems

De Morgan's second theorem:

➤ **The complement of a sum of variables is equal to the product of the complements of the variables.**

Stated in another way,

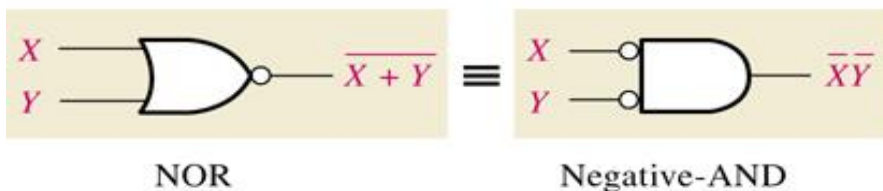
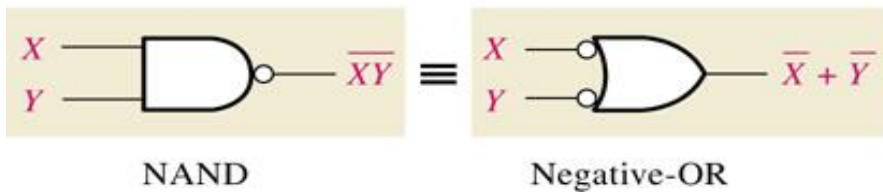
➤ The complement of two or more variables **O**red is equivalent to the **A**ND of the complements of the individual variables

➤ The formula for expressing this theorem for two variables is:

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

De Morgan's Theorem

- De Morgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and equivalency of the NOR and negative-AND gates.
- These theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted



Inputs		Output	
X	Y	\overline{XY}	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Inputs		Output	
X	Y	$\overline{X + Y}$	$\overline{X} \overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Gate equivalencies and corresponding truth tables

BOOLEAN ANALYSIS OF LOGIC CIRCUITS

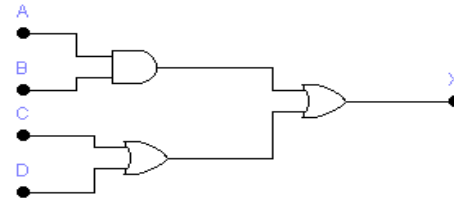
Boolean Analysis of Logic Circuits

The purpose of this section is to

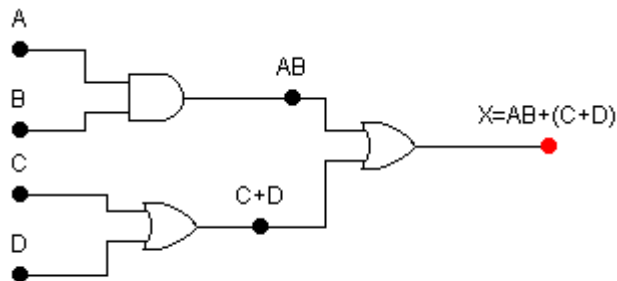
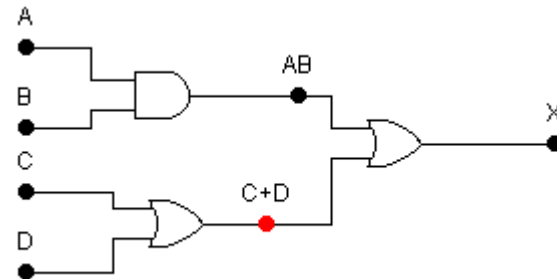
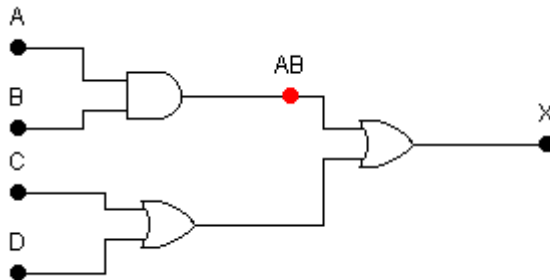
- ❑ Determine the boolean expression for a combination of gates.
- ❑ Evaluate the logic operation of a circuit from the boolean expression.
- ❑ Arrive at the simplified boolean algebra expression with the given logic.

Determination of the boolean expression is done one gate at a time starting at the inputs and simplification is done using Boolean laws and rules

Example 1 : Problem



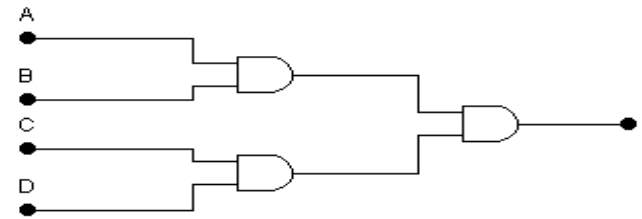
Solution: One gate at a time starting with the inputs



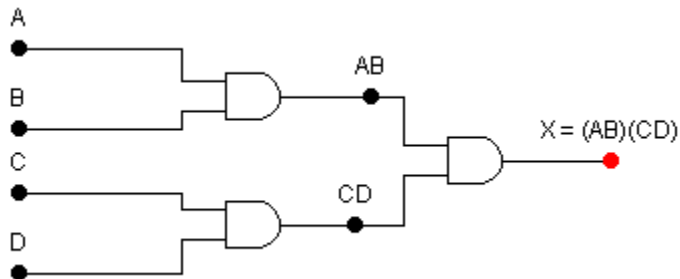
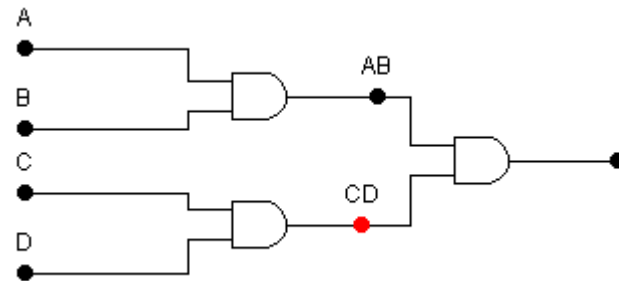
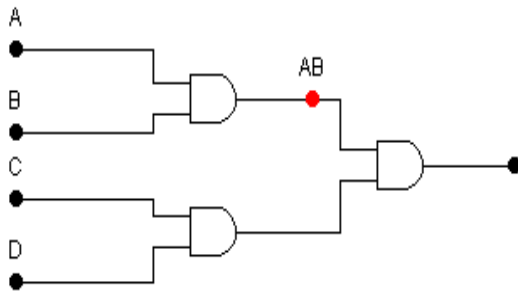
$$X = AB + (C + D)$$

$$X = AB + C + D$$

Example 2 : Problem



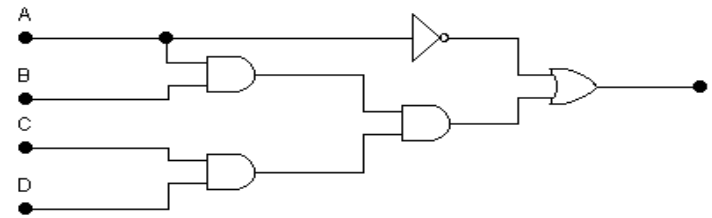
Solution:



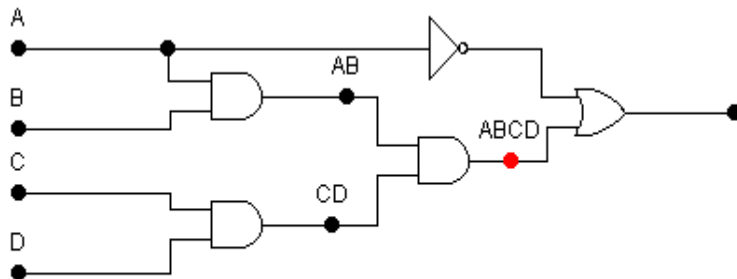
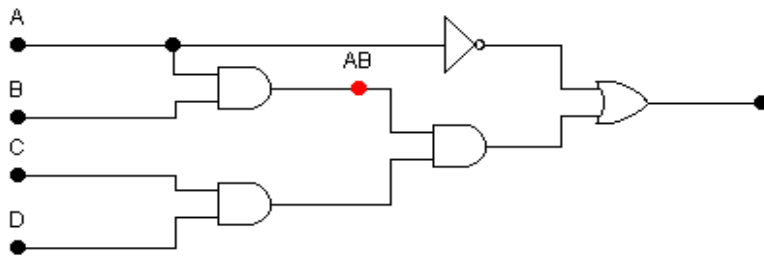
$$X = (AB)(CD)$$

$$X = ABCD$$

Example 3 : Problem

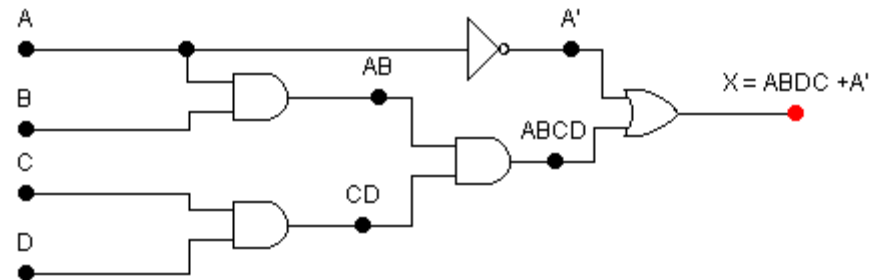
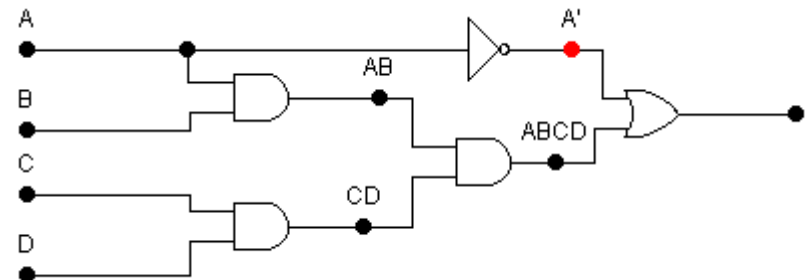
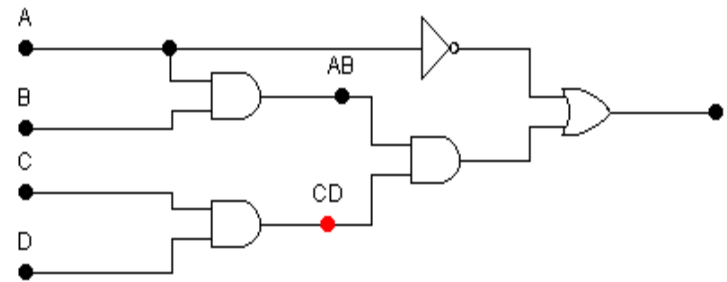


Solution:



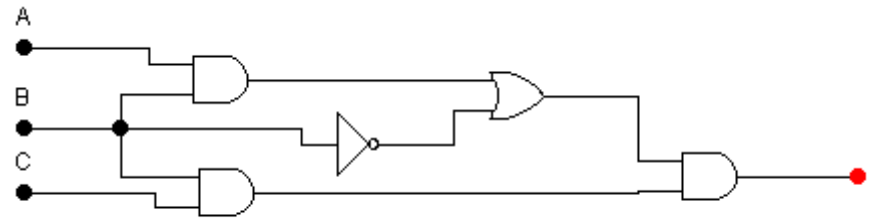
$$X = ABCD + \overline{A}$$

$$X = \overline{A} + BCD$$

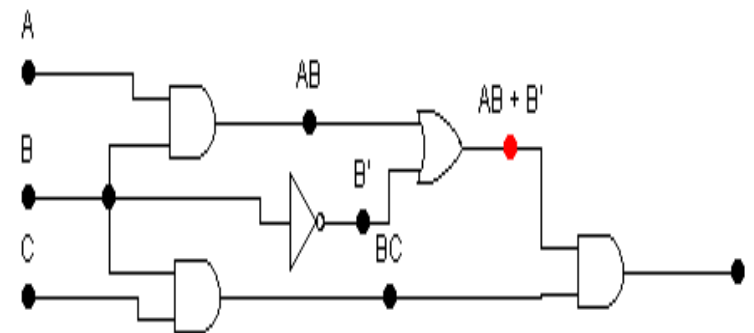
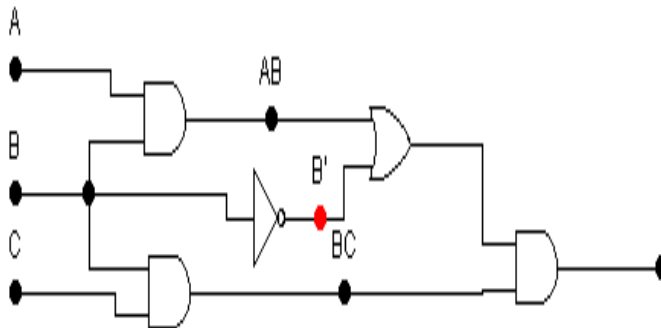
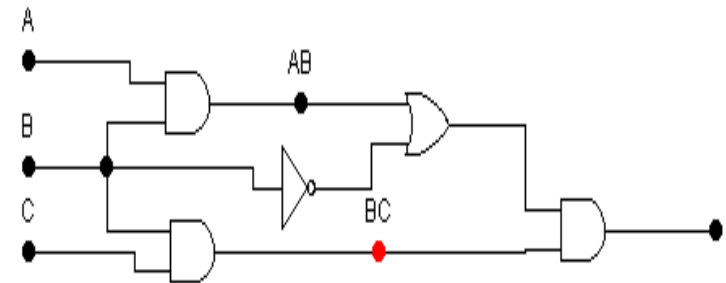
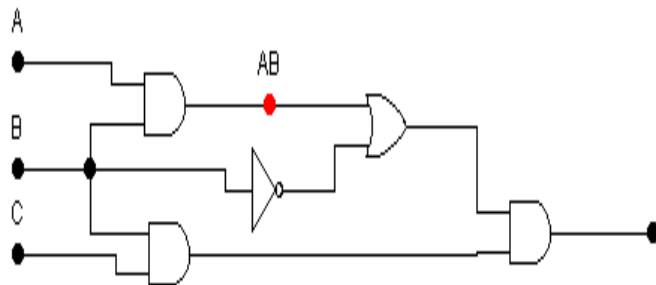


$$X = ABDC + A'$$

Example 4 : Problem

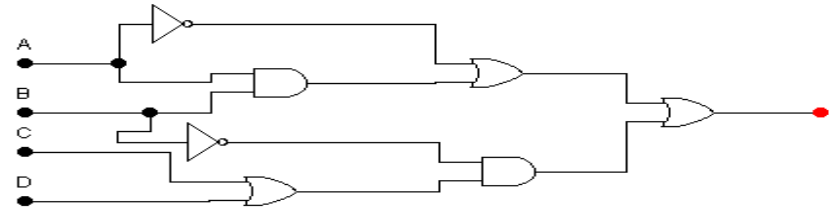


Solution:

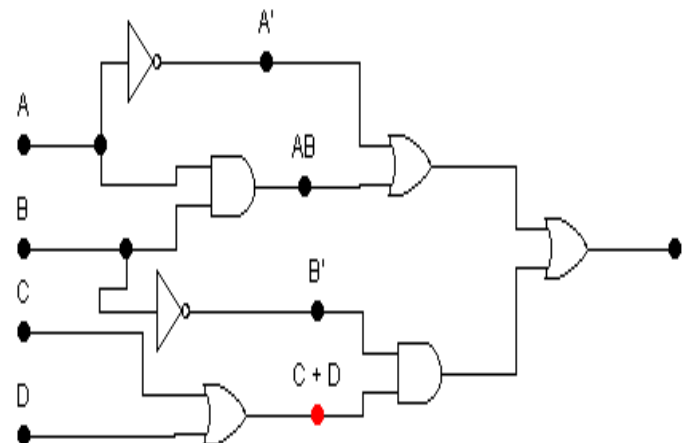
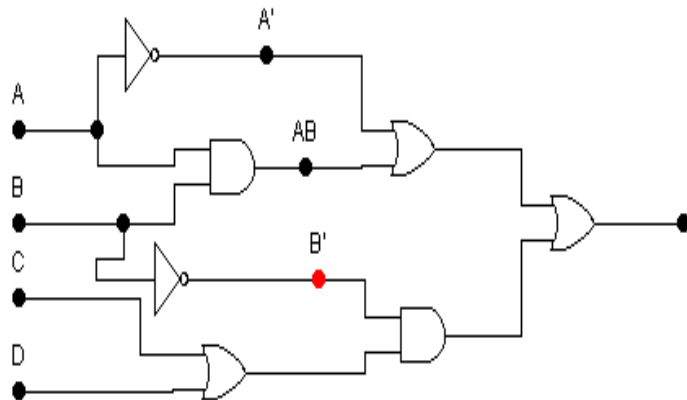
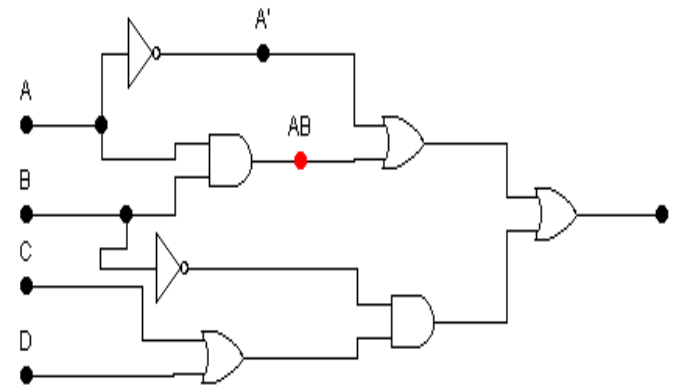
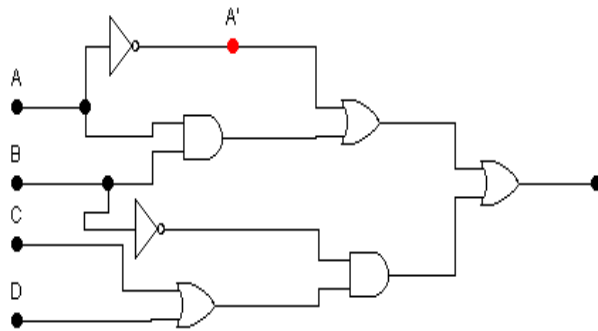


$$X = ABC$$

Example 5 : Problem

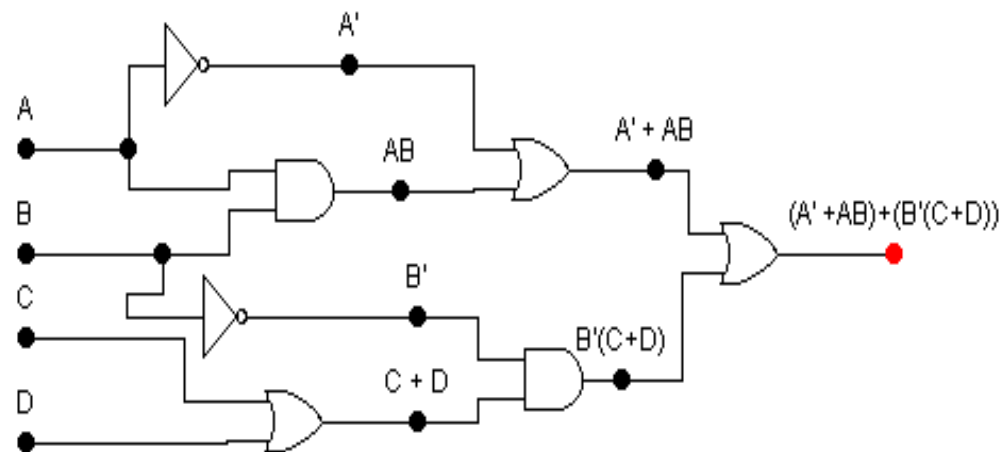
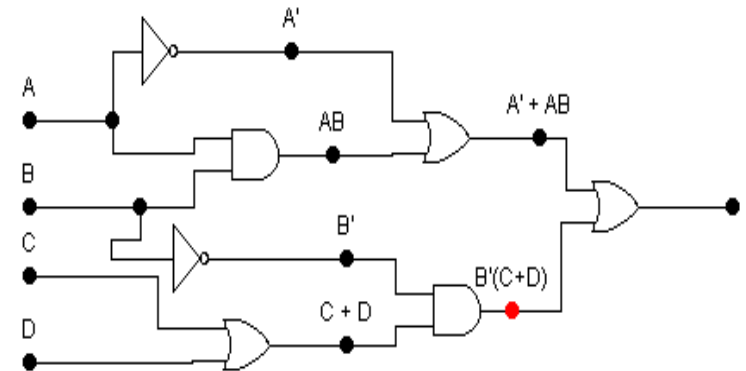
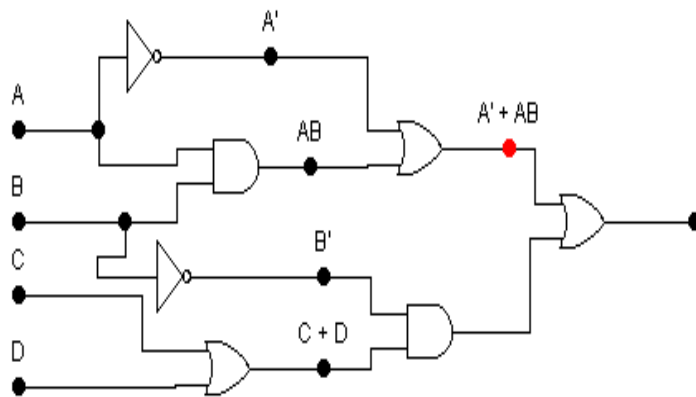


Solution:



Example 5 : Problem

Solution (continued):



Example 5 : Problem

Solution (continued):

Simplification:

$$X = (\bar{A} + AB) + (\bar{B}(C+D))$$

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

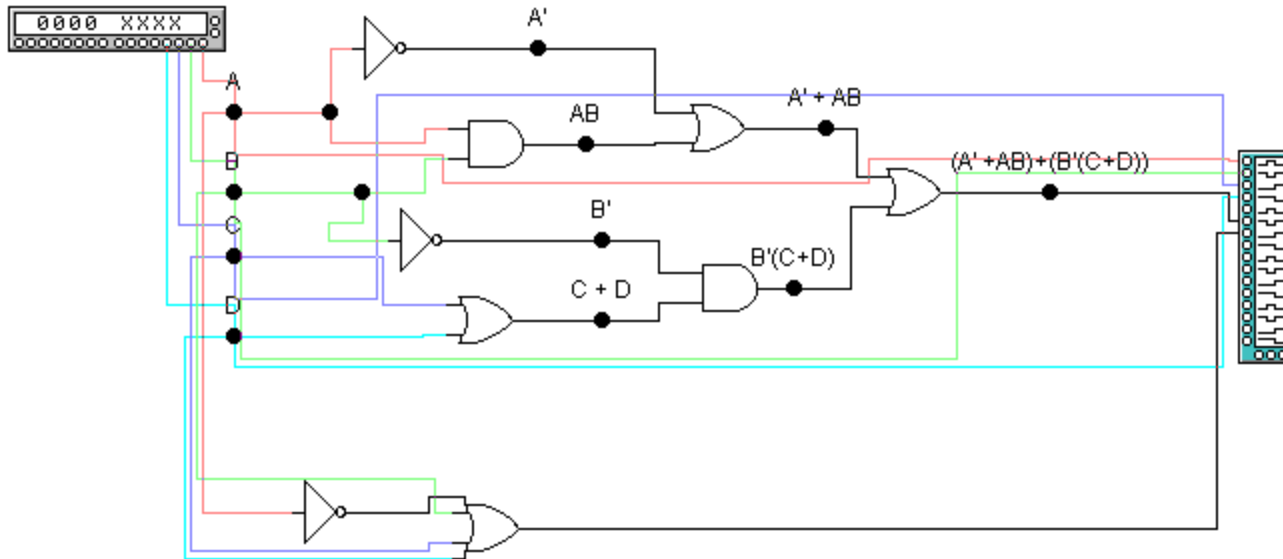
$$X = (\bar{A} + B) + (\bar{B}C + \bar{B}D)$$

$$X = \bar{A} + B + \bar{B}C + \bar{B}D$$

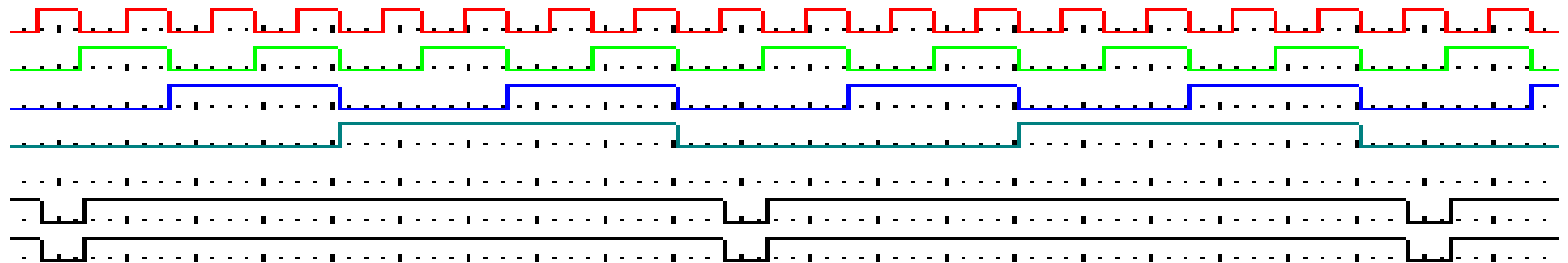
$$X = \bar{A} + B + C + \bar{B}D$$

$$X = \bar{A} + B + C + D$$

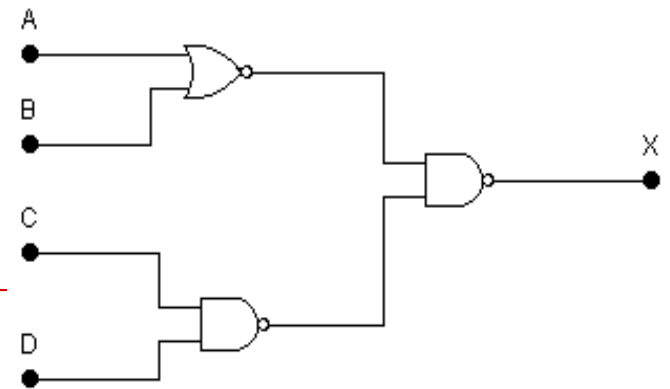
Example 5 : Problem



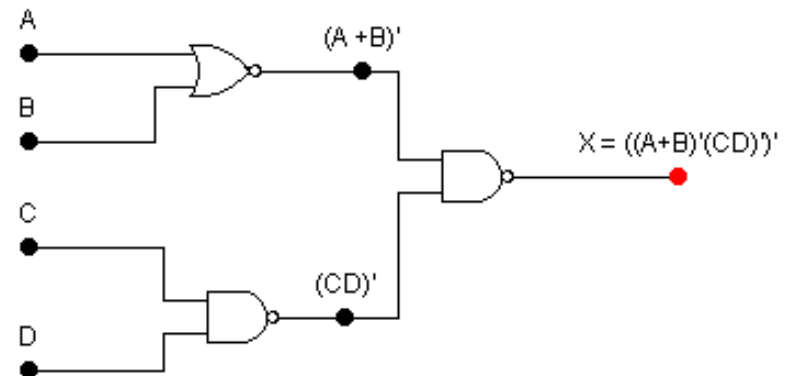
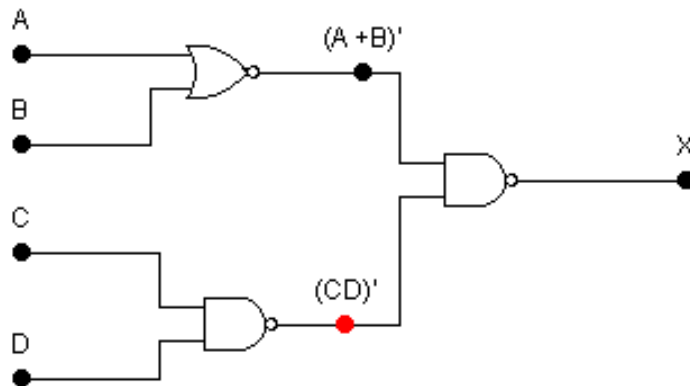
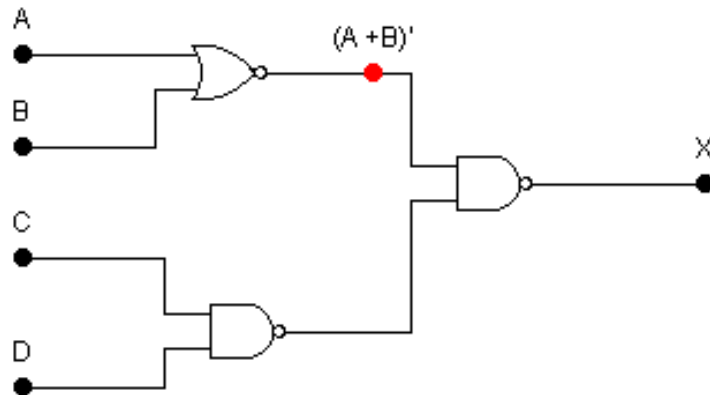
The circuits are different but the outputs are the same



Example 6 : Problem

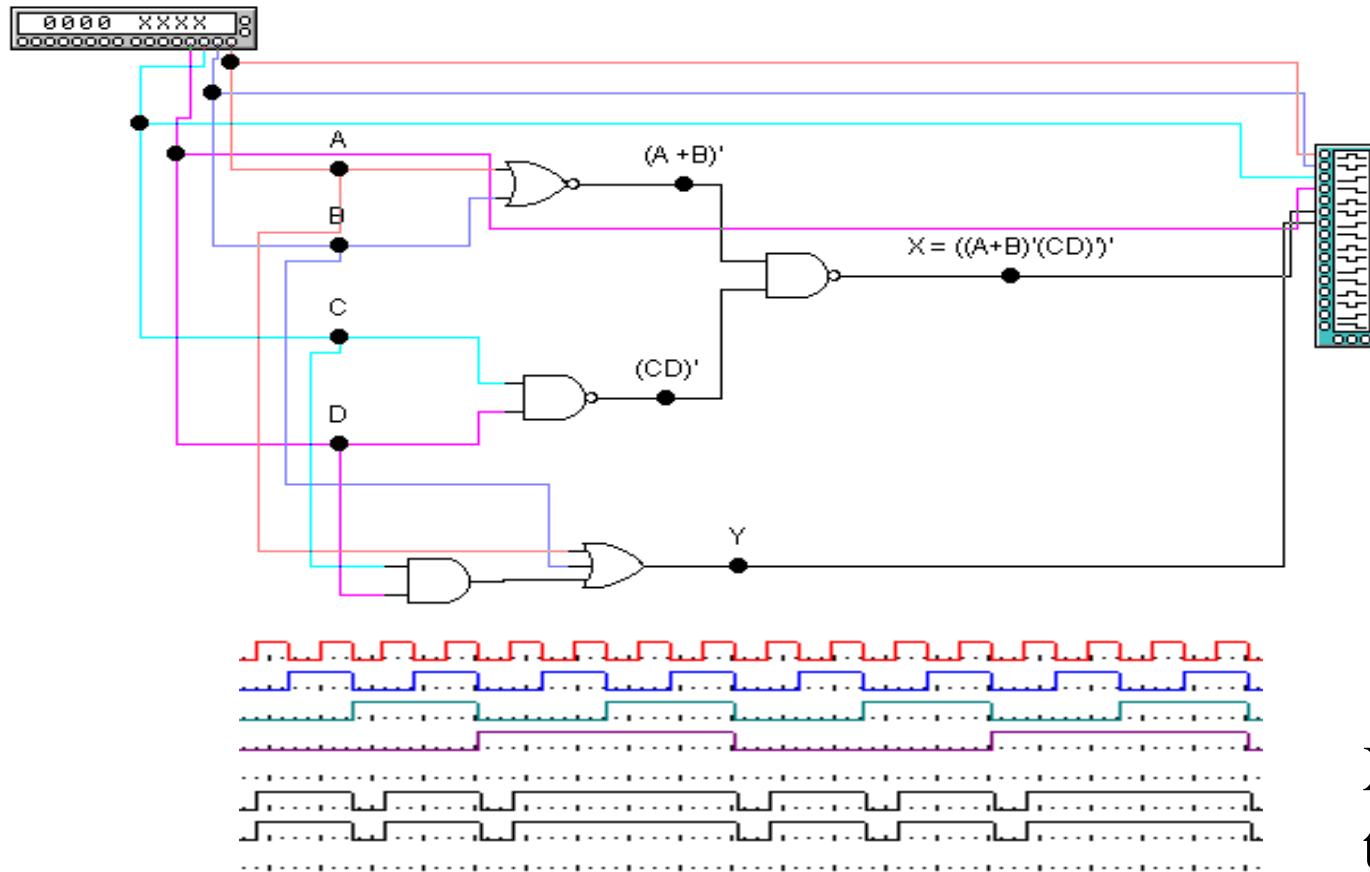


Solution:



$$\overline{\overline{(A+B)(CD)}} = \overline{\overline{(A+B)}} + \overline{\overline{CD}} \\ = A + B + CD$$

Example 6 : Problem



X and Y are
the same

SIMPLIFICATION USING BOOLEAN ALGEBRA TECHNIQUES

Problem 1

Simplify: $A'B + A'B'C'D' + ABCD'$

$$\begin{aligned} A'B + A'B'C'D' + ABCD' &= A'(B + B'C'D') + ABCD' \\ &= A'(B + C'D') + ABCD' \\ &= B(A' + ACD') + A'C'D' \\ &= B(A' + CD') + A'C'D' \\ &= A'B + BCD' + A'C'D' \end{aligned}$$

Problem 2

Simplify: $F = A' B C + A B' C + A B C' + A B C$

$$F = \bar{A}BC + A\bar{B}C + AB(\bar{C} + C)$$

$$F = \bar{A}BC + A\bar{B}C + AB(1)$$

$$F = \bar{A}BC + A\bar{B}C + AB$$

$$F = \bar{A}BC + A\bar{B}C + AB + ABC$$

$$F = \bar{A}BC + AC(\bar{B} + B) + AB$$

$$F = \bar{A}BC + AC + AB$$

$$F = \bar{A}BC + AC + AB + ABC$$

$$F = BC(\bar{A} + A) + AC + AB$$

$$**$F = BC + AC + AB$**$$

Problem 3

Simplify:

$$W = [M + N'P + (R + ST)'] [M + N'P + R + ST]$$

$$\text{Assume } X = M + N'P \quad Y = R + ST$$

$$W = (X + Y')(X + Y)$$

$$W = XX + XY + Y'X + Y'Y$$

$$W = X \cdot 1 + XY + XY' + 0$$

$$W = X + X(Y + Y') = X + X \cdot 1 = X$$

$$\mathbf{W = M + N'P}$$

Problem 4

Express the complement $f'(w,x,y,z)$ of the following expression in a simplified form.

$$\begin{aligned}f(w,x,y,z) &= wx(y'z + yz') \\f'(w,x,y,z) &= w' + x' + (y'z + yz')' \\&= w' + x' + (y'z)'(yz')' \\&= w' + x' + (y + z')(y' + z) \\&= w' + x' + yy' + yz + z'y' + z'z \\&= w' + x' + 0 + yz + z'y' + 0 \\&= \mathbf{w' + x' + yz + y'z'}\end{aligned}$$

Simplification using Boolean Algebra

– More Problems

□ Using Boolean Algebra Techniques, simplify the following expressions :

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

2. $[AB'(C + BD) + A'B']C$

3. $A'BC + AB'C' + A'B'C' + AB'C + ABC$

4. $(AB + AC)' + A'B'C$

References

Slides adopted from the books

- ❑ Thomas L.Floyd, “Digital Fundamentals,” 11th Edition, Prentice Hall, 2014 (ISBN10:0132737965/ISBN13:9780132737968)
- ❑ M.Morris Mano and Michael D. Ciletti, " Digital Design," 5th Edition, Pearson Education International, 2013 (ISBN: 9780132774208)
- ❑ Ronald J.Tocci, Neal S.Widmer, and Gregory L.Moss, "Digital Systems- Principles and Application"- 11th Edition, Pearson Education International, 2011 (ISBN: 9780135103821)