MIT-WPU
॥ विश्वशान्तिर्भुवं ध्रुवा ॥

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Basics of Electrical and Electronics Engineering

ECE1022A

UNIT 3 – INTEGRATED CIRCUITS

1. DIGITAL ICS
2. ANALOG ICS

Fundamentals of Digital Electronics

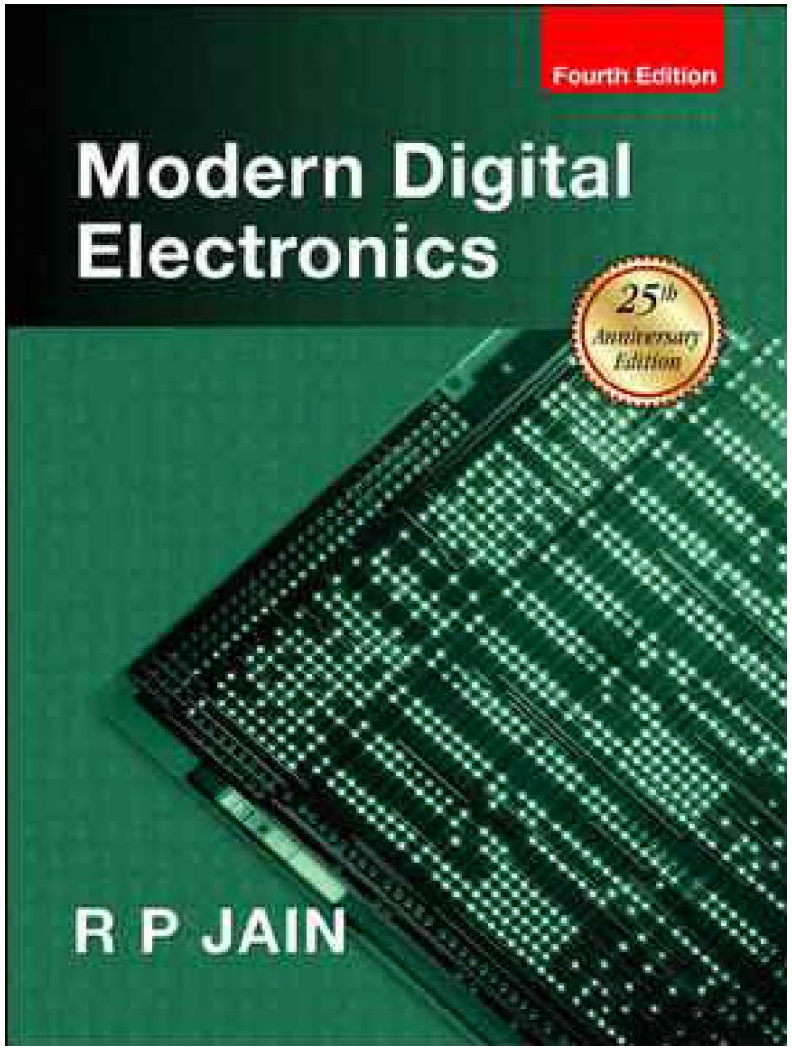
Contents:

Digital logic levels, basic gates and universal gates, Boolean algebra, Combinational Logic Circuits, De-Morgan's theorems, SOP, POS, K map, Half Adder, Full adder, flip flops: S R flip flop, JK flip flop, D flip flop, T flip flop, Shift registers, Introduction to Microcontrollers.

Reference Book:

- R.P. Jain, Modern Digital Electronics. New Delhi: Tata McGraw-Hill, 4th Edition, 2009
- Digital Fundamentals 11th Edition by Thomas L. Floyd

Reference book soft copy available in BEEE classroom

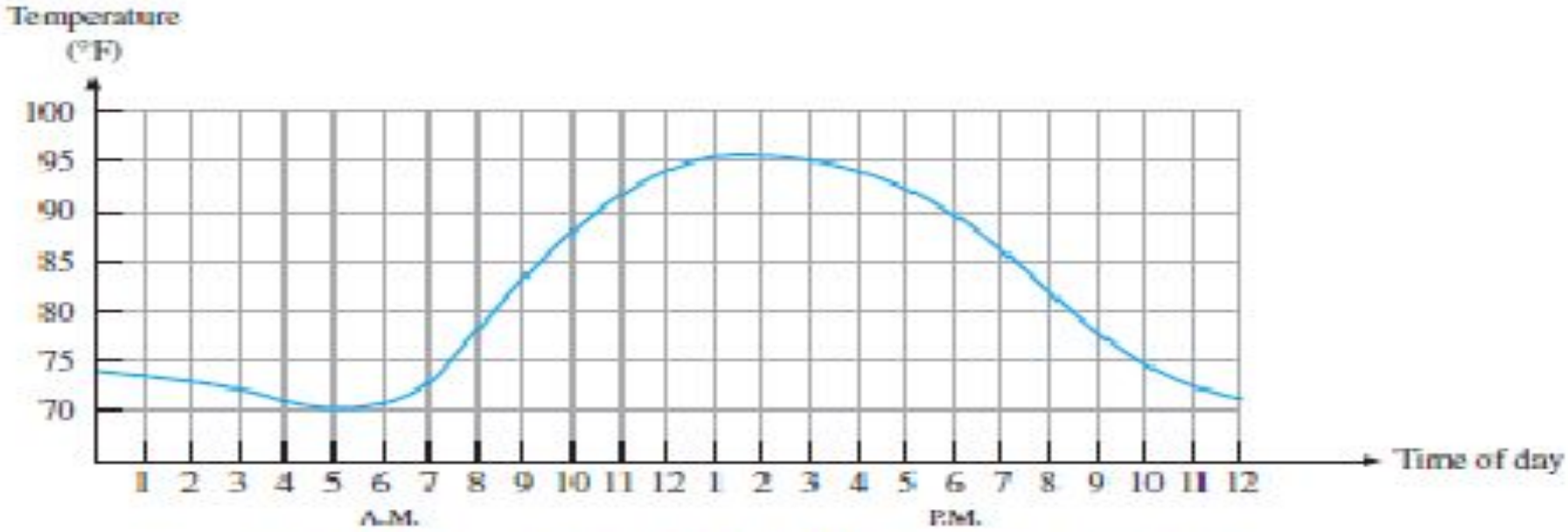


Digital and Analog Quantities

- For many years, applications of digital electronics were limited to computer systems.
- Digital Technology have wide range of applications
  - ✔ Television
  - ✔ Communication systems
  - ✔ Radar
  - ✔ Navigation
  - ✔ Military Systems
  - ✔ Medical Instrumentations
  - ✔ Industrial Process Control
  - ✔ Consumer electronics etc.
- Electronics circuits can be divided into two broad categories
  - ✔ Digital
  - ✔ Analog

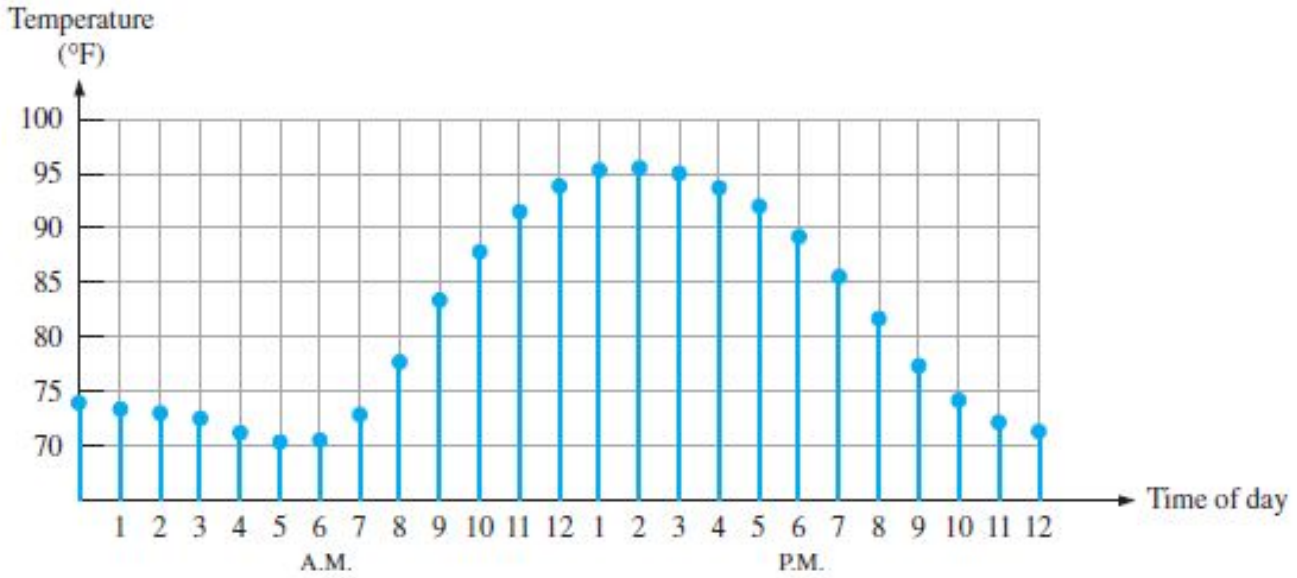
Digital and Analog Quantities

- Analog: is one having continuous values
- Digital: is one having discrete set of values
- Example: Air temperature changes over continuous range of values
- **Analog Signal**



Graph of an analog quantity (temperature versus time).

Digital code

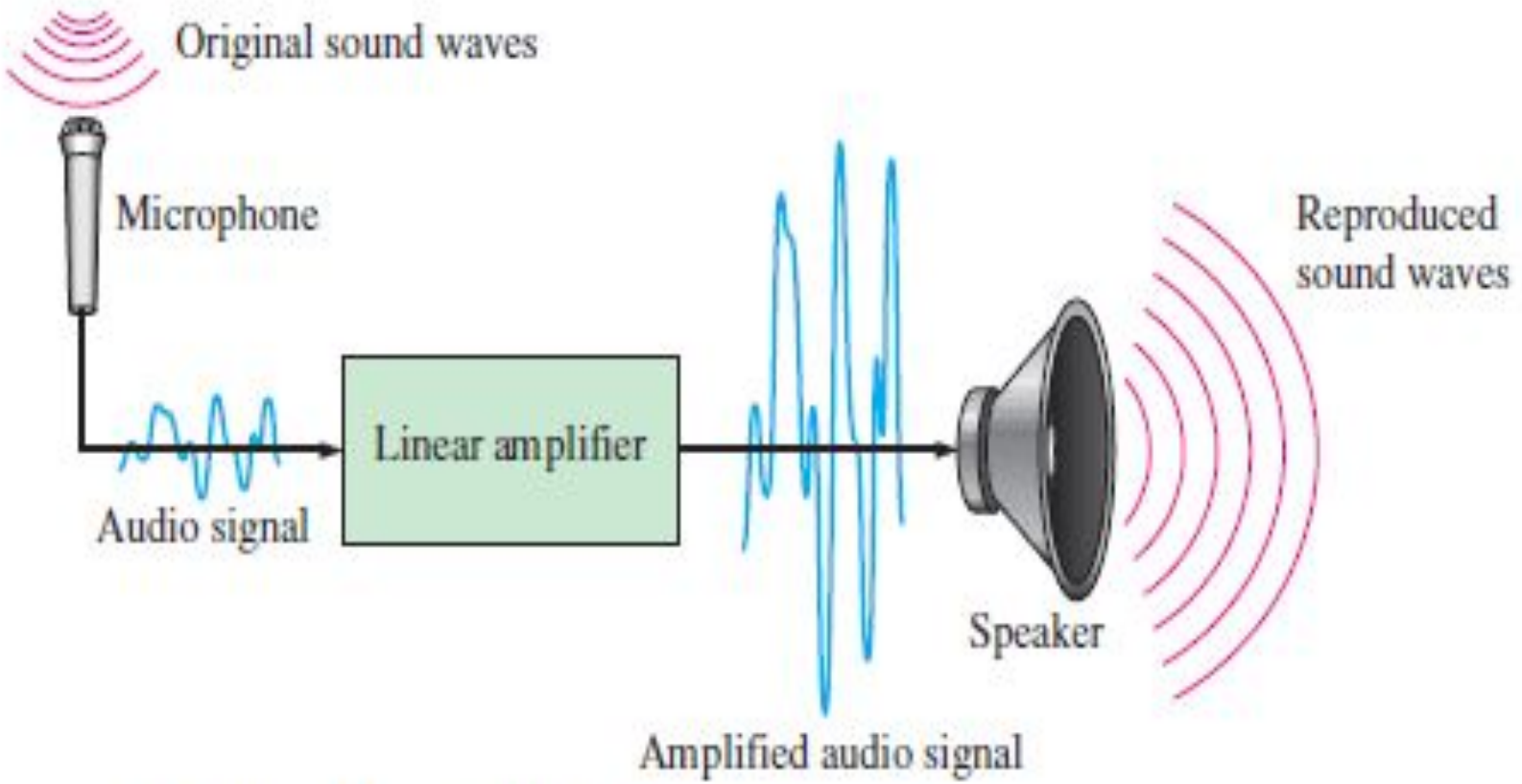


Sampled-value representation (quantization) of the analog quantity in
Each value represented by a dot can be digitized by representing it as a digital code that consists of a series of 1s and 0s.

Advantages of Digital signals

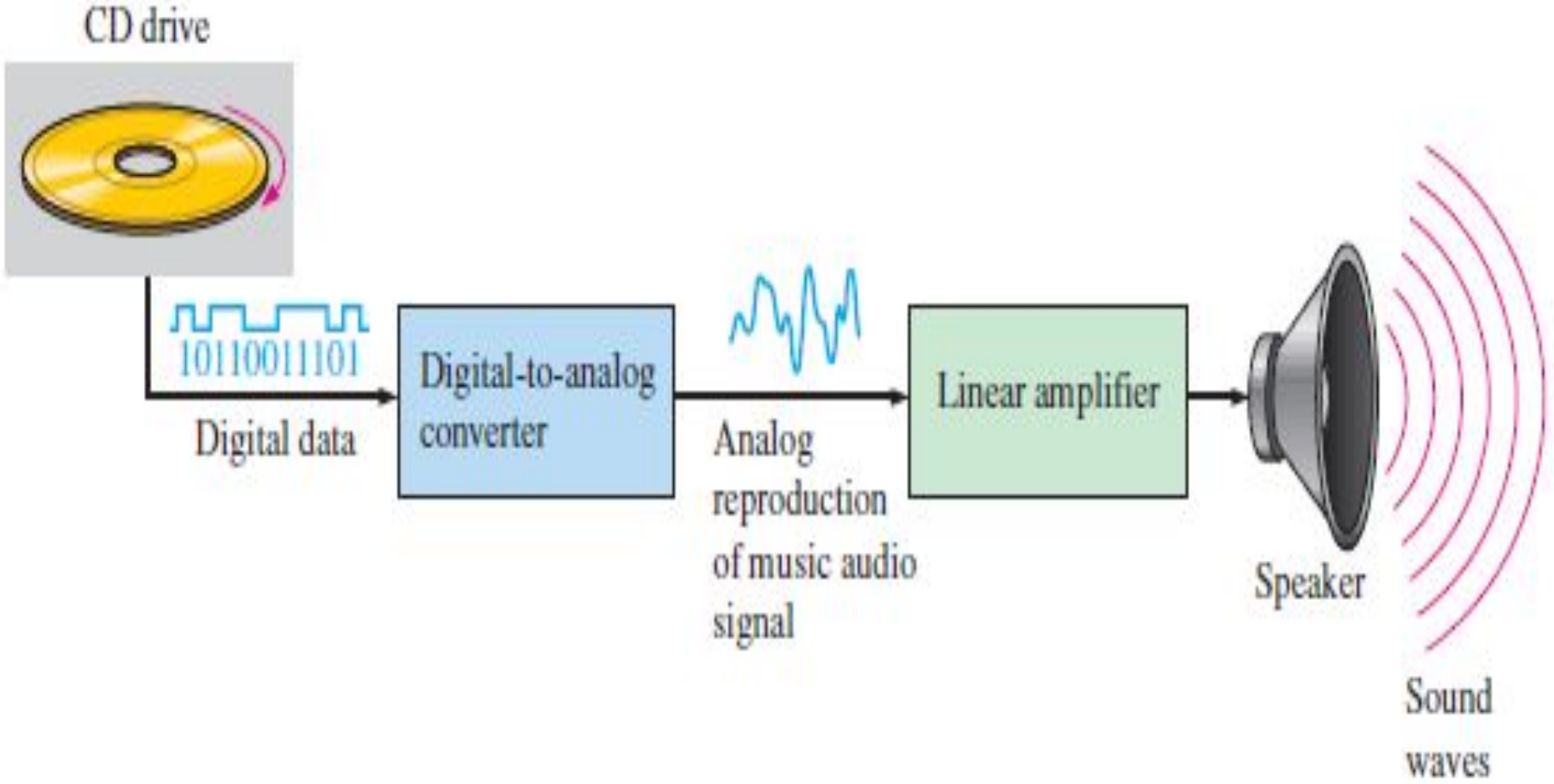
- Data transmission is more effective and reliable
- Digital data has a great advantage when storage is necessary
- Digital data is more accurate and precise
- Its ripple free or noise free

Analog system



A basic audio public address system.

A System Using analog and Digital Methods

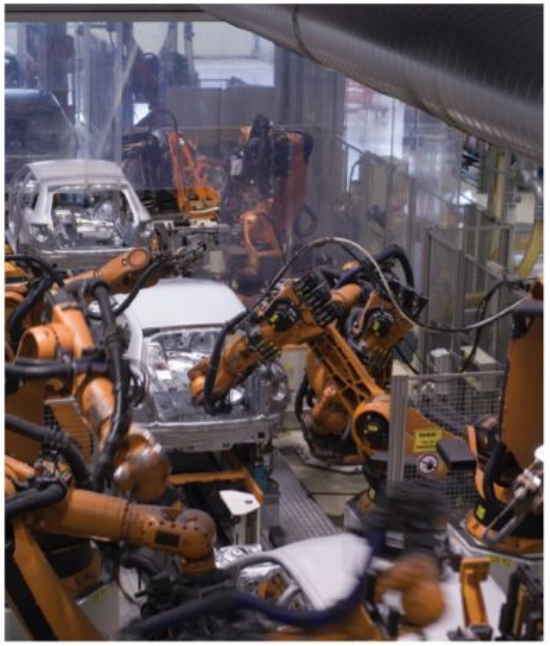


Basic block diagram of a CD player. Only one channel is shown.

Mechatronics

- The interdisciplinary field that comprises both mechanical and electronic components is known as mechatronics
- Both digital and analog electronics are used in the control of various mechanical systems.

(b) Robotic arm



(c) Automotive assembly line

Digital Electronics

- It uses Binary Digits 0 and 1
- Each of the two digits in the binary system, 1 and 0, is called a bit, Two different voltage levels are used to represent the two bits.
- 1 - is represented by the higher voltage, which will be referred as a HIGH,
- 0 - is represented by the lower voltage level, which will be referred as a LOW.
- This is called positive logic and will be used throughout the topic.

HIGH = 1 and LOW = 0

Logic Level ranges or Voltage for a digital circuits

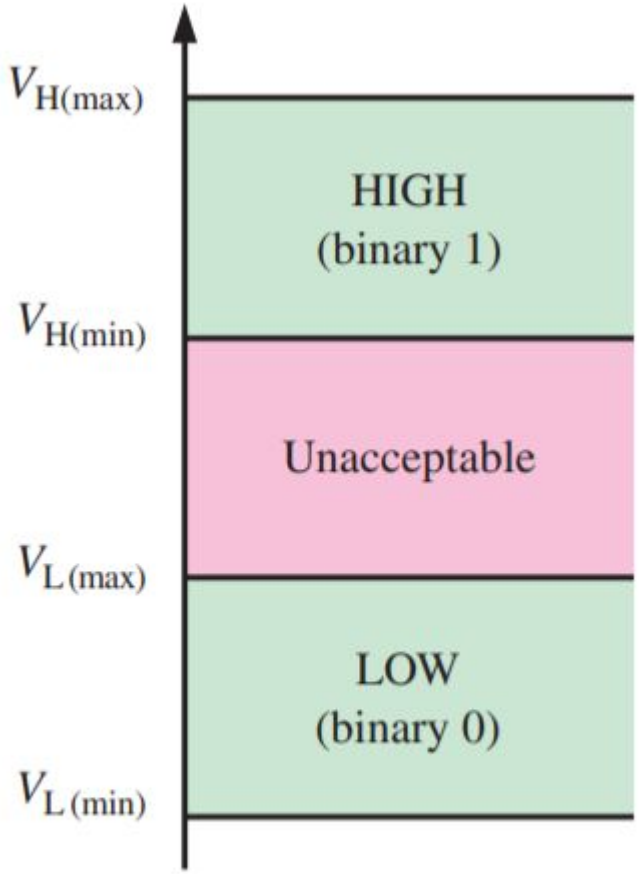
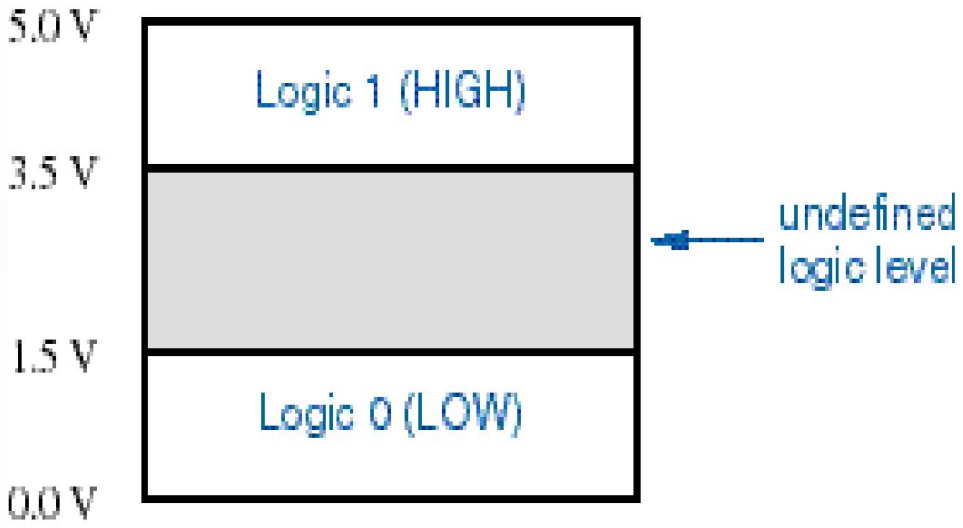


FIGURE 1–6 Logic level ranges of voltage for a digital circuit.

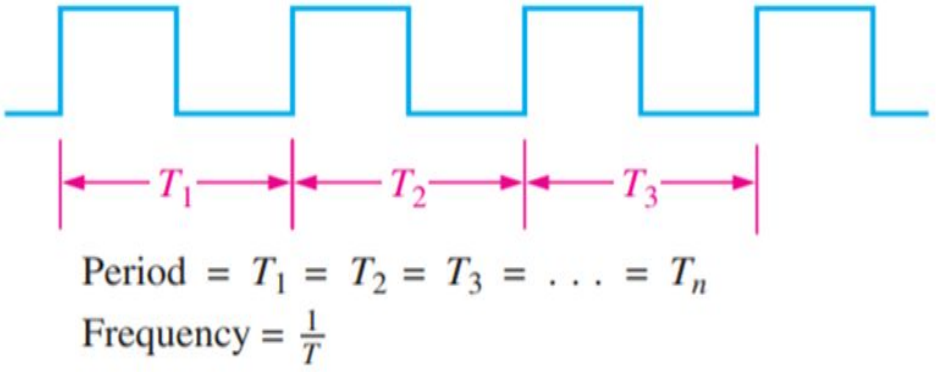
Logic Levels

- **Positive Logic levels**



Examples of digital waveforms

(a) Periodic (square wave)



(b) Non-periodic

- An important characteristic of a periodic digital waveform is its duty cycle, which is the Ratio of the pulse width (t_W) to the period (T). It can be expressed as a percentage

$$\text{Duty cycle} = \left(\frac{t_W}{T}\right)100\%$$

Digital waveforms

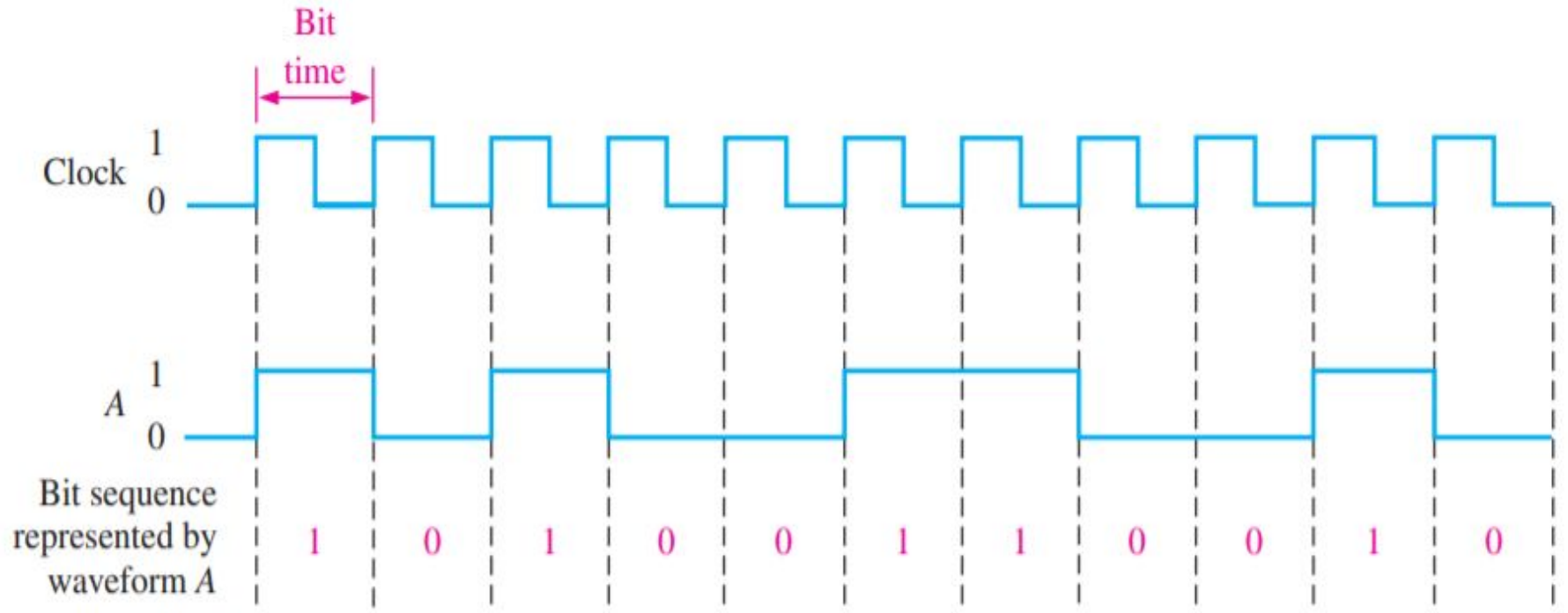
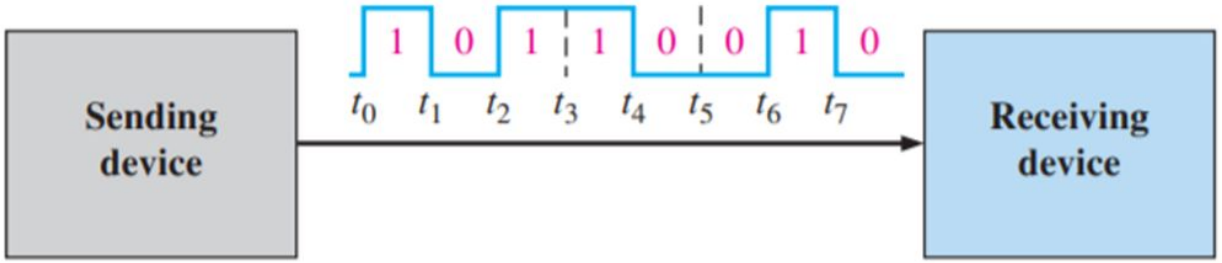
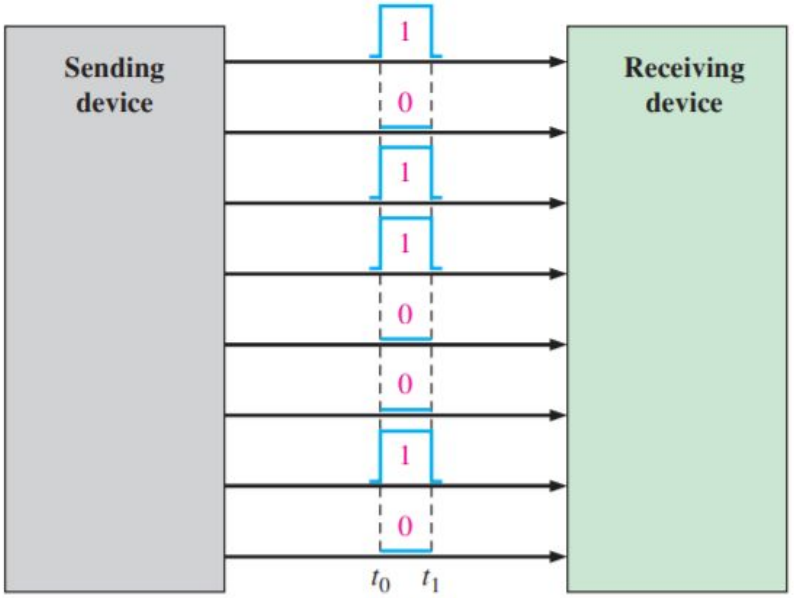


FIGURE 1–11 Example of a clock waveform synchronized with a waveform representation of a sequence of bits.

Digital Data Transfer

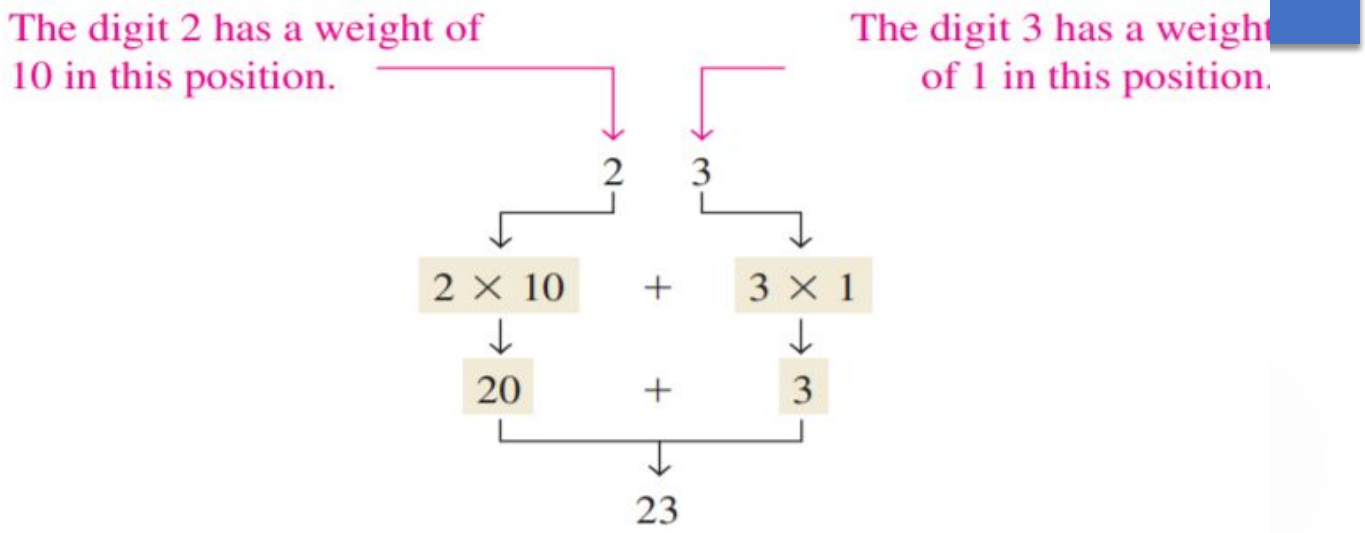


(a) Serial transfer of 8 bits of binary data. Interval t_0 to t_1 is first.



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

Decimal Number System



- The decimal number system has a base of 10.
- The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a weight.
- The weights for whole numbers are positive powers of ten that increase from right to left, beginning with $10^0 = 1$.

$$\ldots\ 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

Binary Numbers

- Binary system has only two digits.
- Binary system with its two digits is a base-two system.
- The two binary digits (bits) are 1 and 0.
- The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit.
- The weights in a binary number are based on powers of two.

	MSB			LSB
Decimal No.	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Binary Numbers

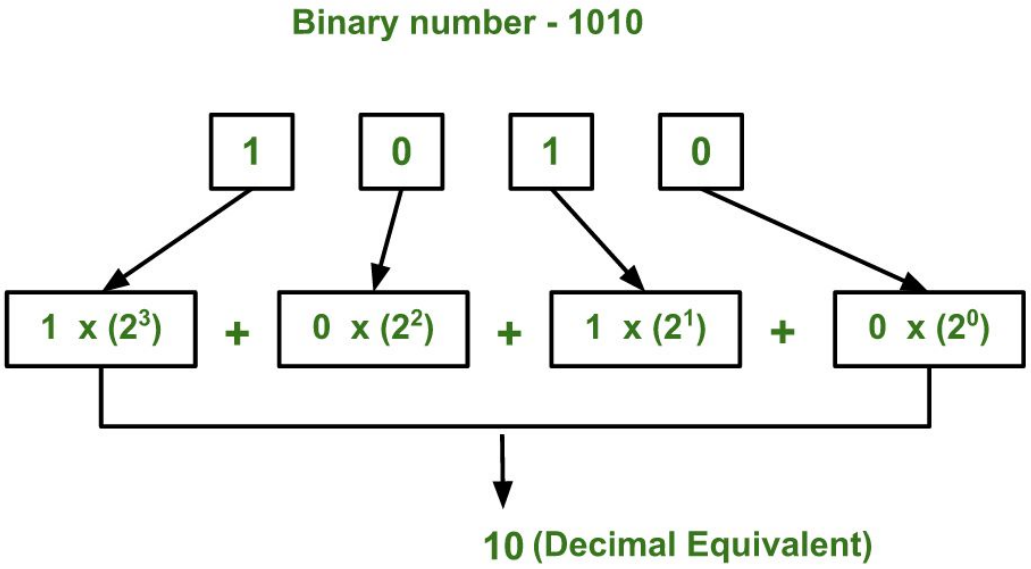
Binary weights.

Positive Powers of Two (Whole Numbers)								
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Negative Powers of Two (Fractional Number)					
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

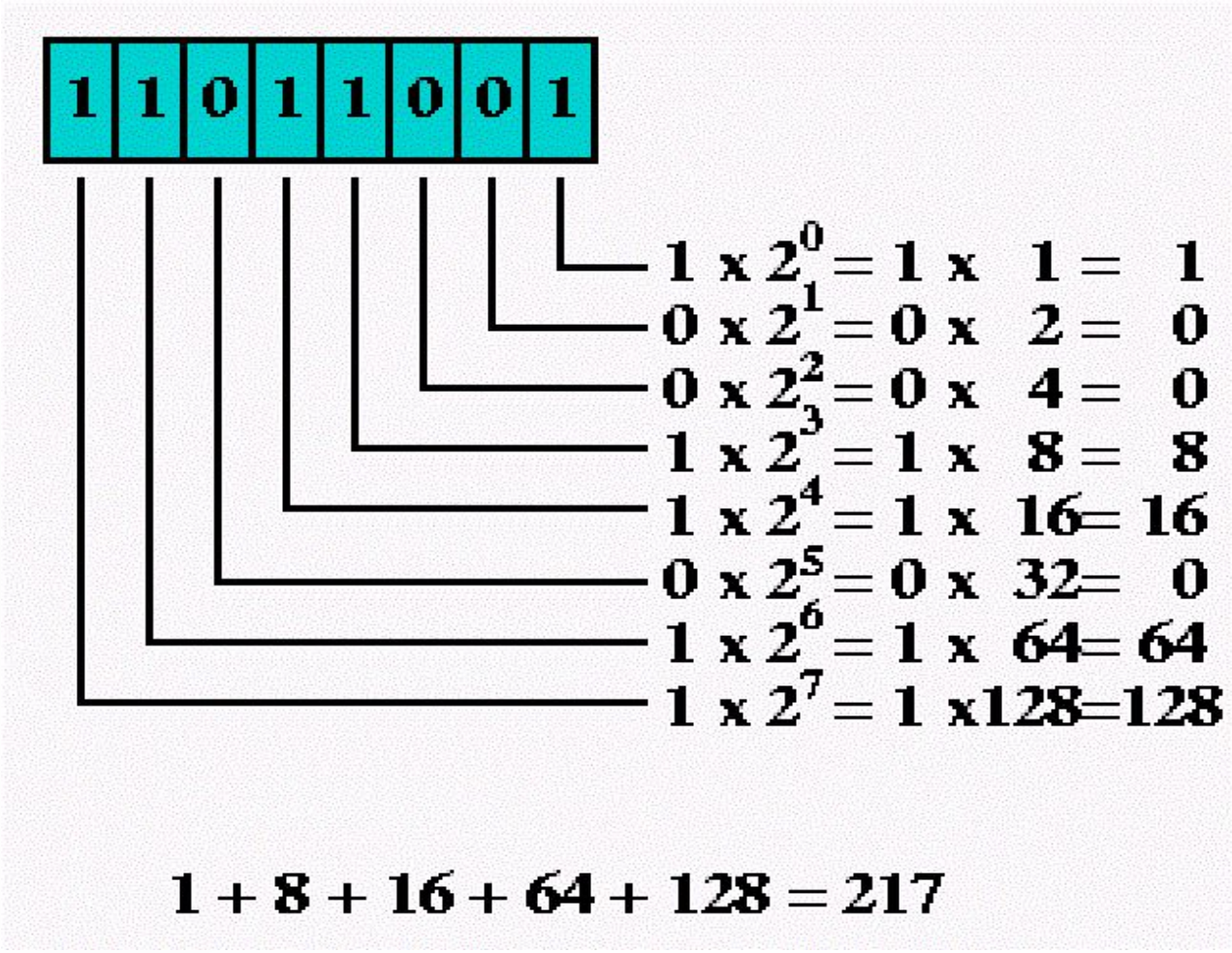
Binary-to-Decimal Conversion

- The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.



- Convert the binary whole number 11011001 to decimal.
- Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

Binary-to-Decimal Conversion



Binary-to-Decimal Conversion

Steps for Decimal to Binary Conversion

Step – 1 Divide the decimal number which is to be converted by two

Step – 2 The remainder which is obtained from step 1 is the least significant bit of the binary number.

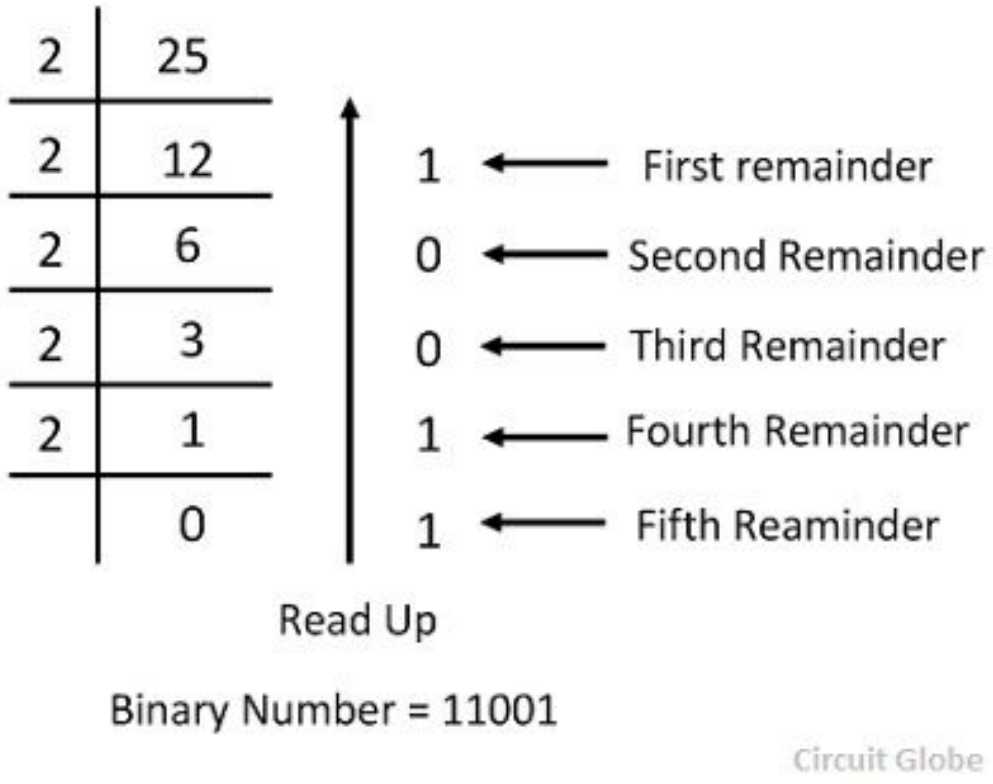
Step – 3 Divide the quotient which is obtained from the step 2 and the remainder obtained from this is the second least significant bit of the binary number.

Step – 4 Repeat the process until the quotient remains zero.

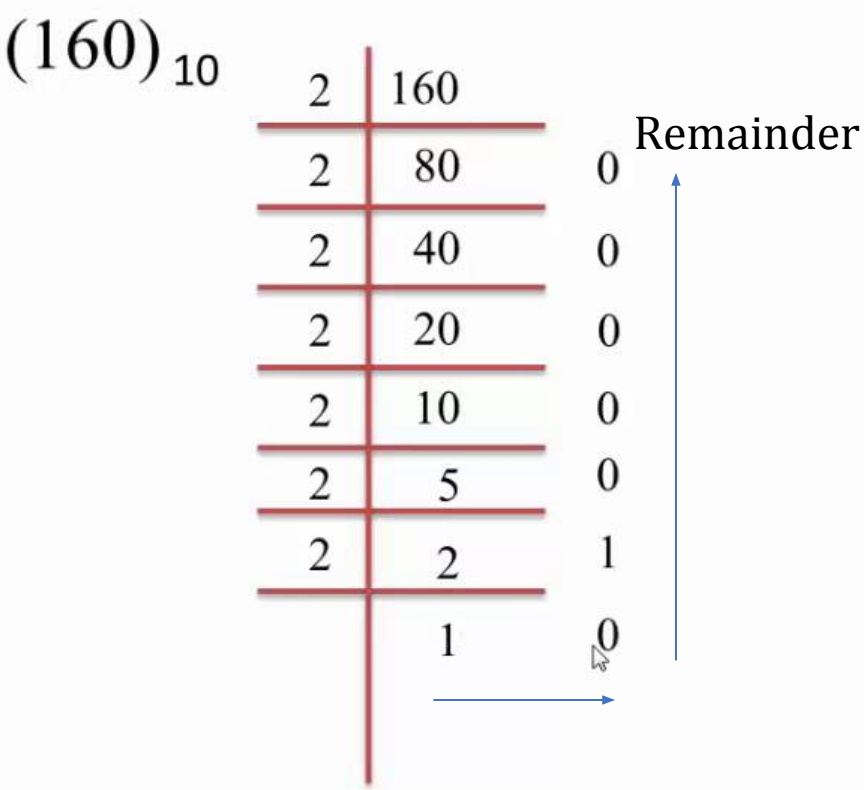
Step – 5 The last remainder obtained from the division is the most significant bit of the binary number. Hence arrange the number from most significant bit to the least significant bit (i.e., from bottom to top).

Binary-to-Decimal Conversion

Convert decimal 25 to binary



Binary-to-Decimal Conversion



Binary Number = 10100000

Binary Addition

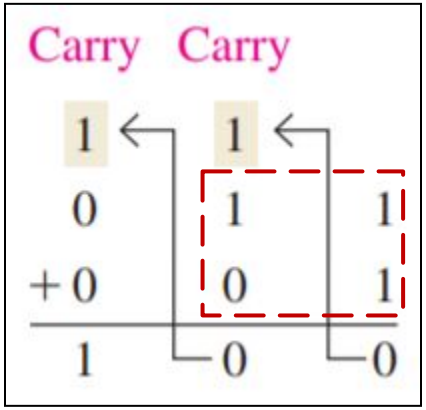
- The four basic rules for adding binary digits (bits) are as follows:

$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1

In binary 1 + 1 = 10, not 2.

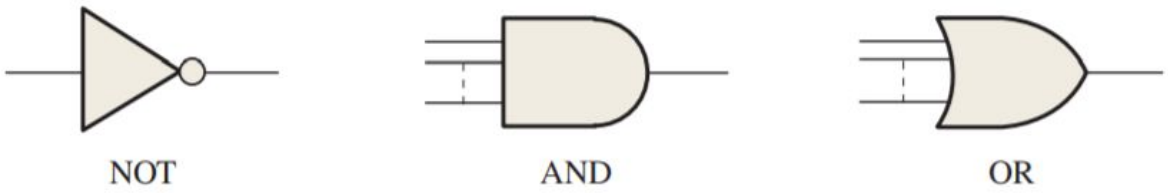
Ex. Addition of 11 + 1:

Answer: 100



Basic Logic Functions

- In the 1850s, the Irish logician and mathematician George Boole developed a mathematical system for formulating logic statements with symbols so that problems can be written and solved in a manner similar to ordinary algebra
- The term logic is applied to digital circuits used to implement logic functions.
- Three basic logic functions - NOT, AND, and OR
- Logic functions are indicated by standard symbols shown below
- The inputs are on the left of each symbol and the output is on the right.
- A circuit that performs a specified logic function (AND, OR) is called a logic gate.
- AND and OR gates can have any number of inputs.



The basic logic functions and symbols

Logic Functions

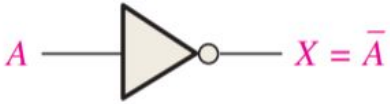
- In logic functions, the true/false conditions are represented by a HIGH (true) and a LOW (false).

1. NOT function/NOT Gate

The NOT function is implemented by a logic circuit known as an inverter

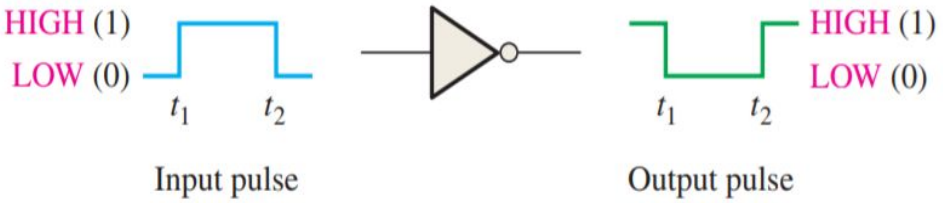
FIGURE 1–17 The NOT function.

Logic Expression for an Inverter :



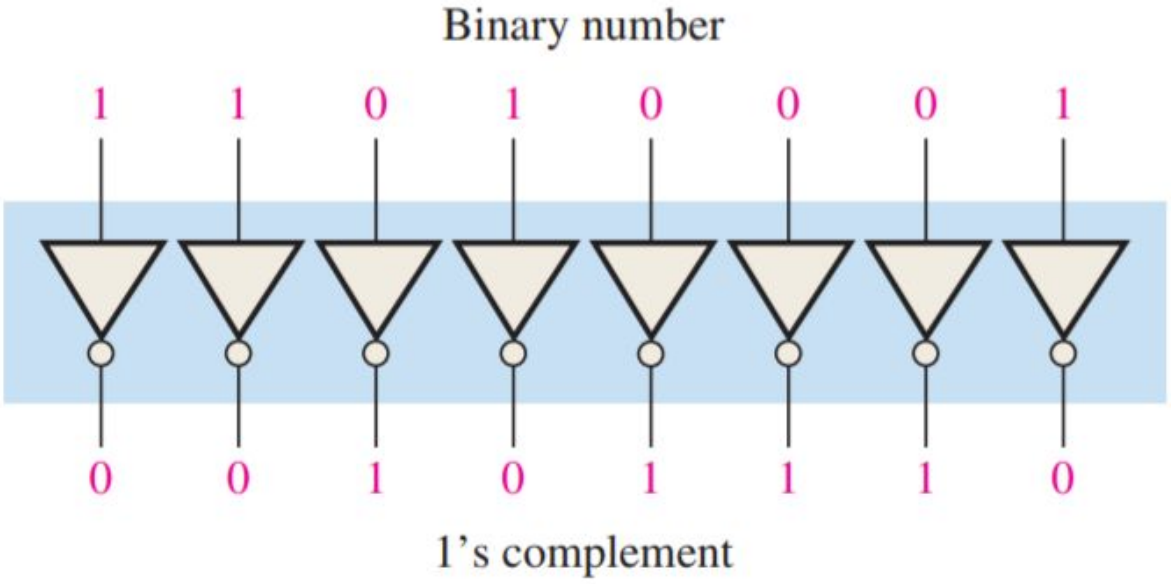
Inverter truth table.

Input	Output
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)



An Application of Inverter

- Figure shows a circuit for producing the 1's complement of an 8-bit binary number.

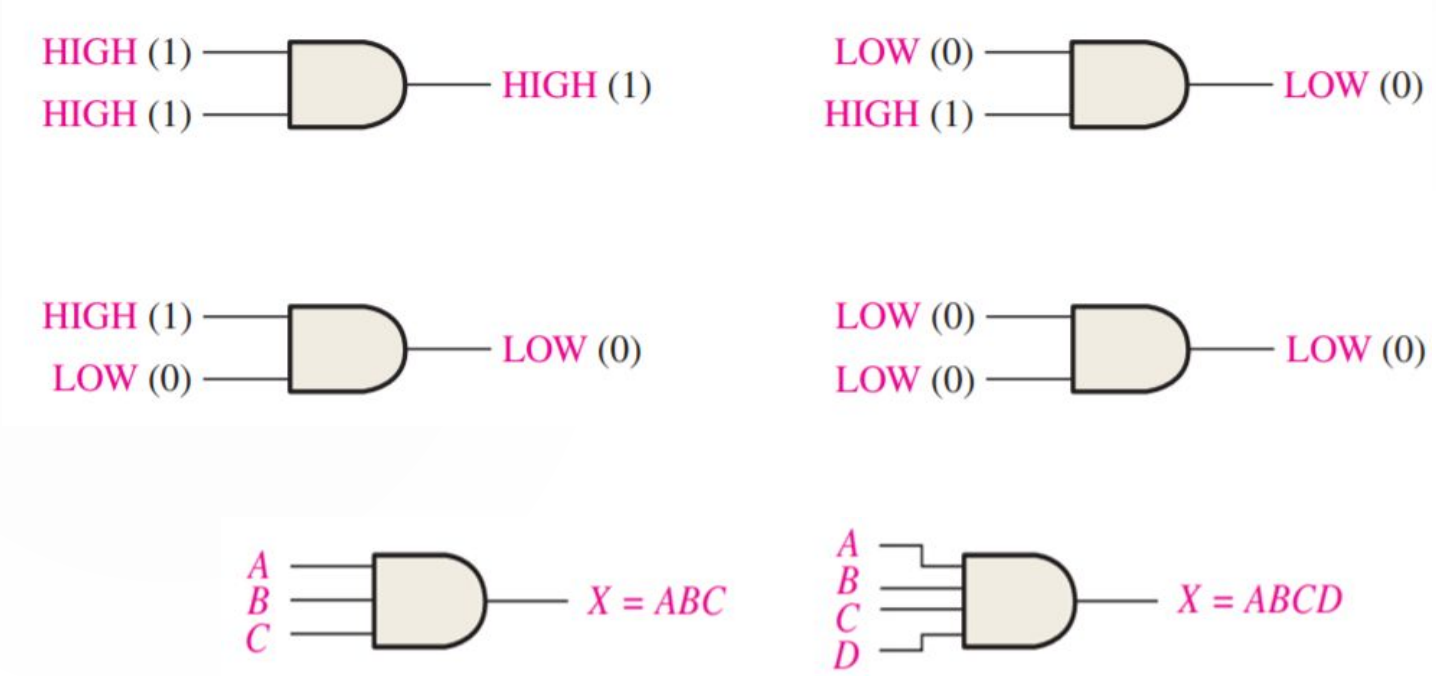


Logic Functions (AND)

2. <u>AND function/ AND Gate</u>

The AND function produces a HIGH output only when all the inputs are HIGH, as indicated in Figure for the case of two inputs.

- When one input is HIGH and the other input is HIGH, the output is HIGH.
- When any or all inputs are LOW, the output is LOW.
- The AND function is implemented by a logic circuit known as an AND gate.



AND Gate

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

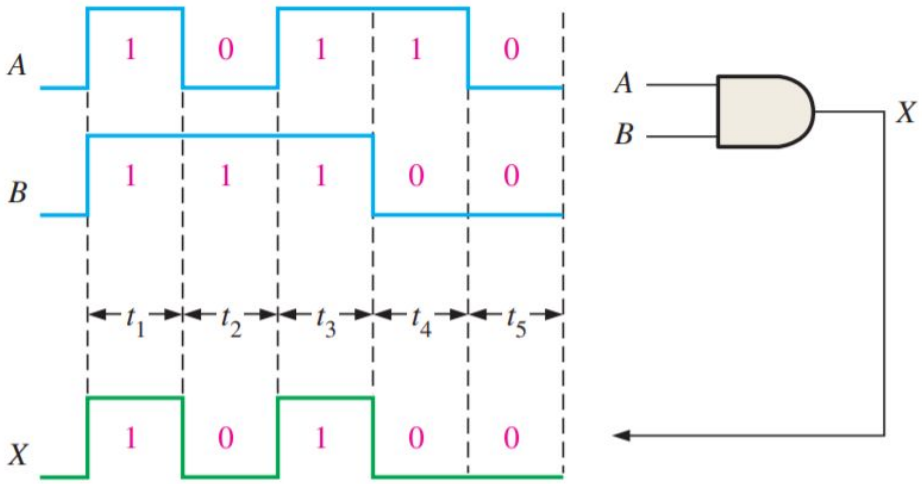
$$N = 2^n$$

Logic Expression for AND gate : X = A • B = AB

Truth table for a 2-input AND gate.

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

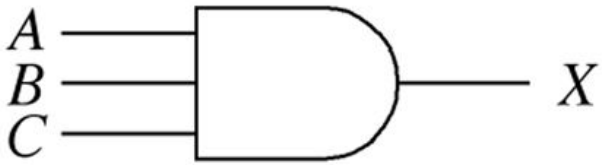
1 = HIGH, 0 = LOW

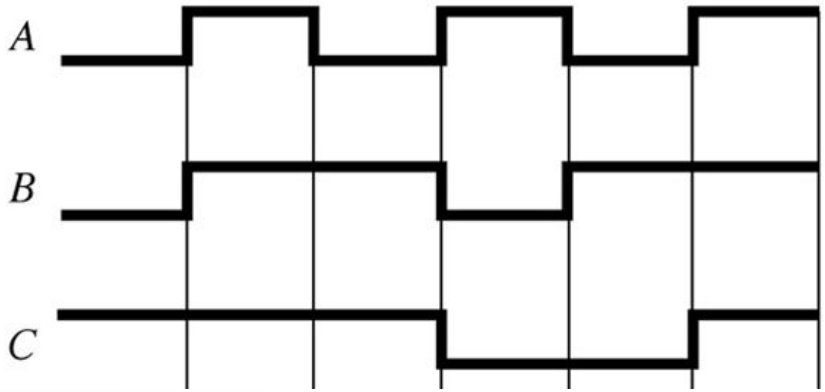


Example of AND gate operation with a timing diagram showing input and output relationships.

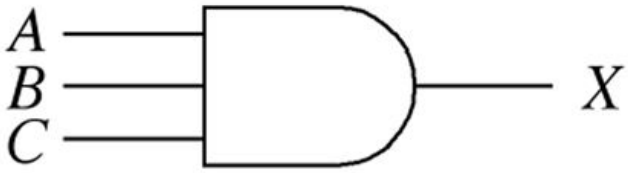
Example 1

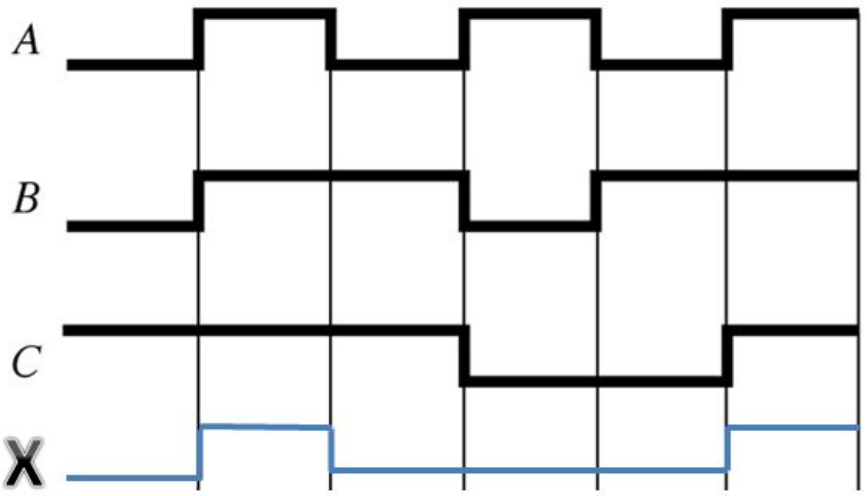
Determine output X: Write logical expression and output waveform





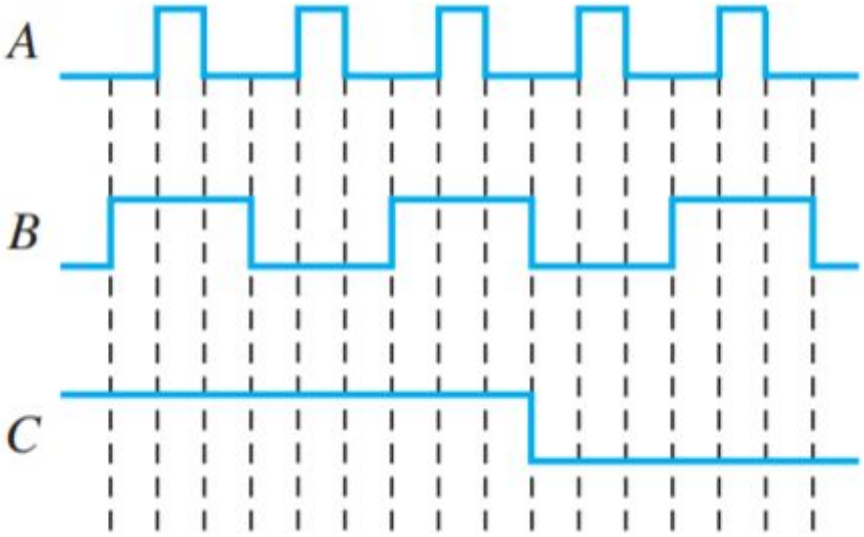
Example 1 Solution

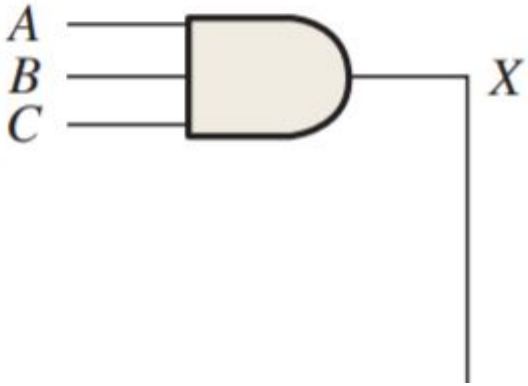




Example 2

Determine output X: Write logical expression and output waveform

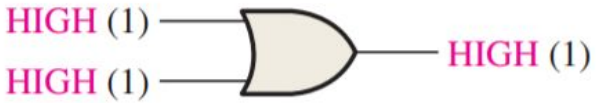


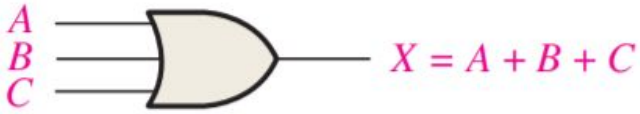


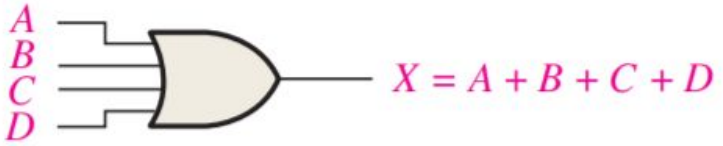
Logic Functions (OR)

3. OR function/ OR Gate

- The OR function produces a HIGH output when one or more inputs are HIGH, as indicated in Figure for the case of two inputs.
- When one input is HIGH or the other input is HIGH or both inputs are HIGH, the output is HIGH.
- When both inputs are LOW, the output is LOW.
- The OR function is implemented by a logic circuit known as an OR gate.







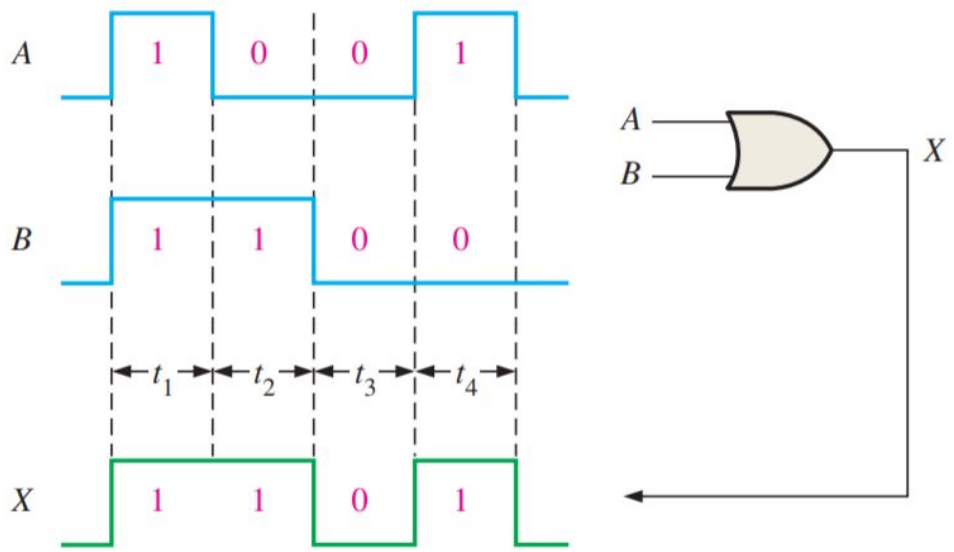
OR Gate

Logic Expression OR gate X = A + B

Truth table for a 2-input OR gate.

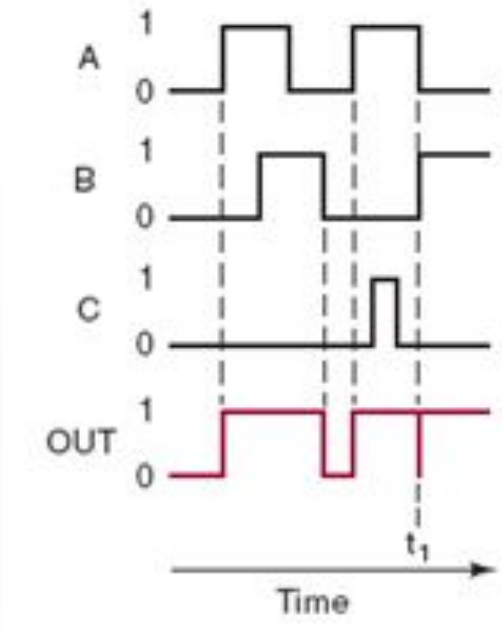
Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

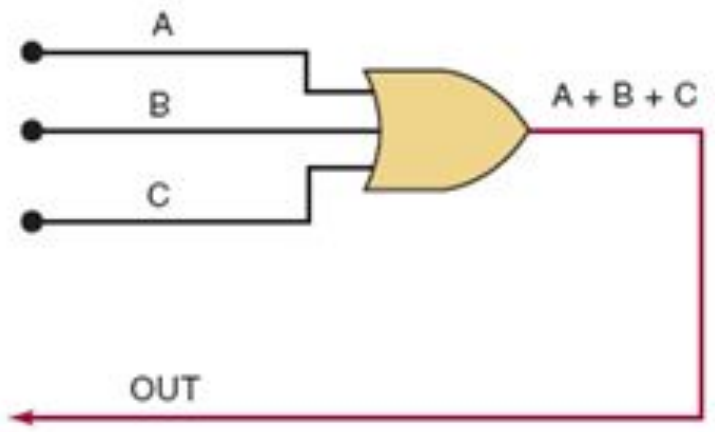
1 = HIGH, 0 = LOW



Example of OR gate operation with a timing diagram showing input and output relationships.

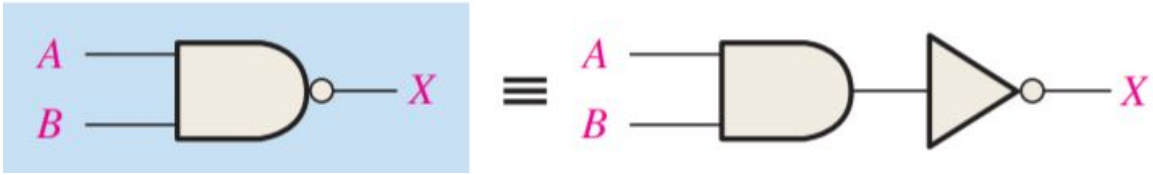
OR Gate





Logic Function NAND

The NAND gate is the same as the AND gate with the inverted output.



The Boolean expression for the output of a 2-input NAND gate is

$$X = \overline{AB}$$

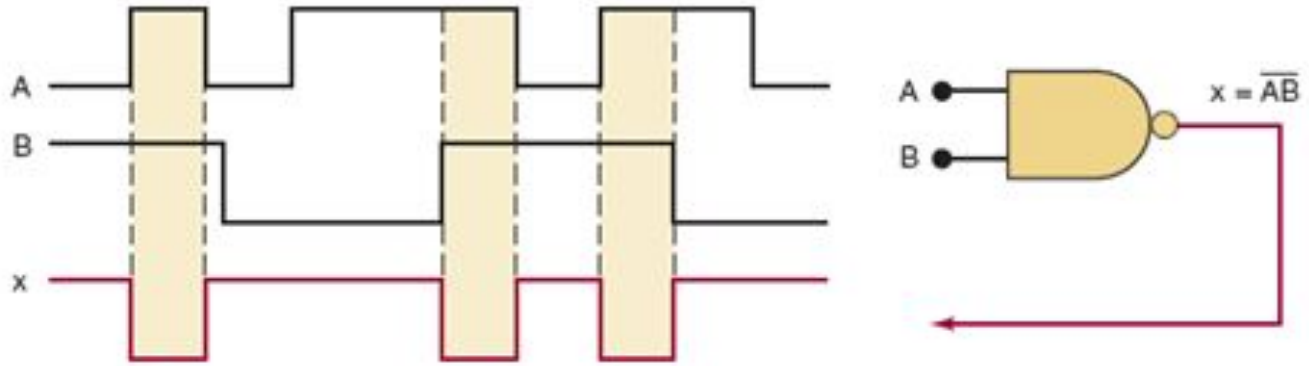
A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

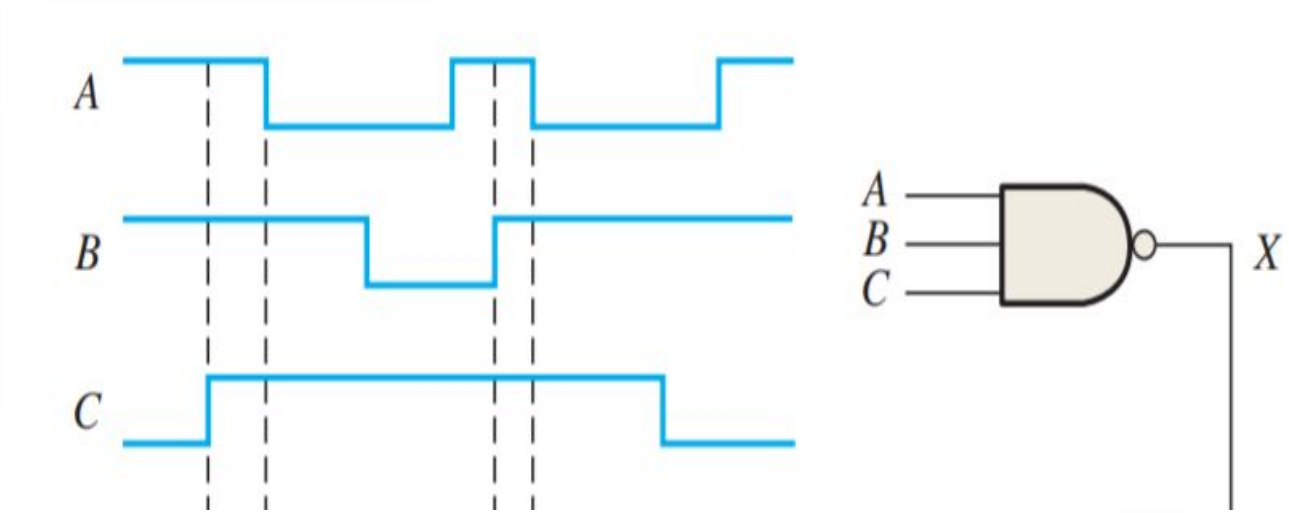
Truth table for a 2-input NAND gate.

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

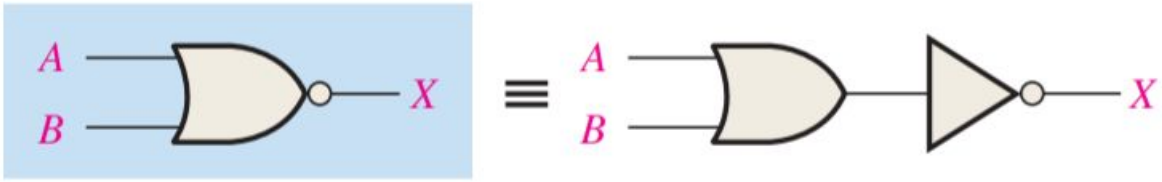
1 = HIGH, 0 = LOW.

Example NAND Gate





NOR Gate



Logic Expressions for a NOR Gate: -> $X = \overline{A + B}$

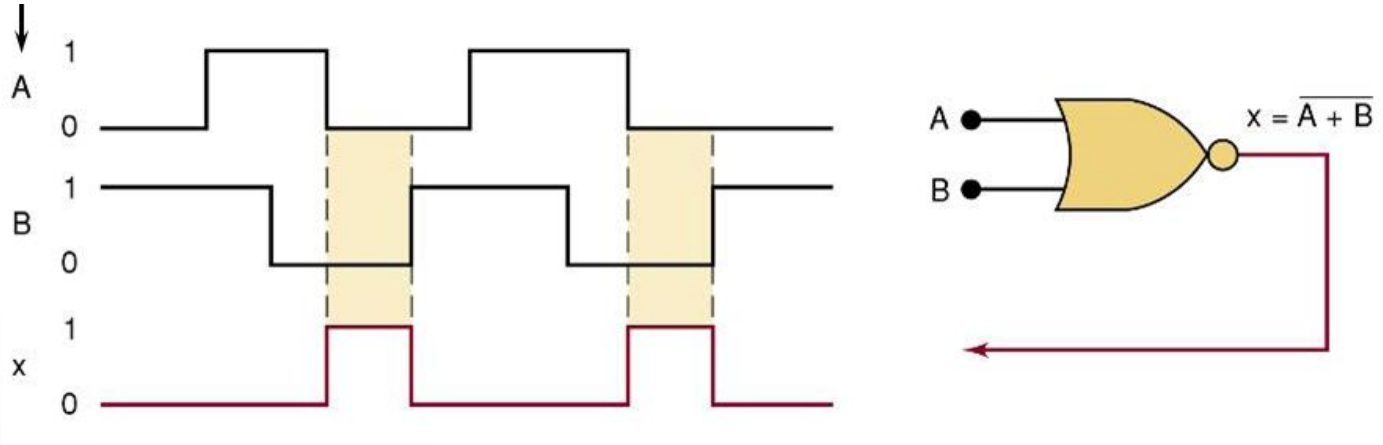
Truth table for a 2-input NOR gate.

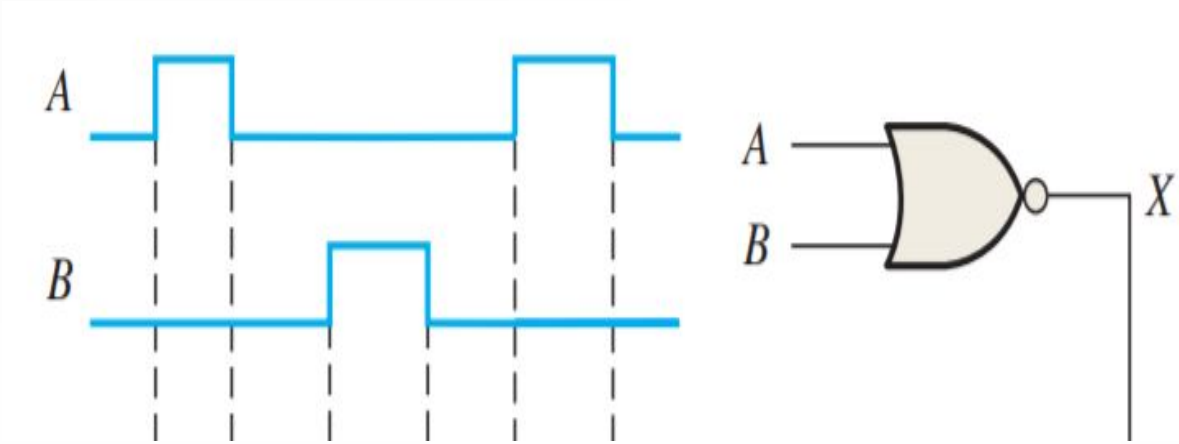
Inputs		Output
A	**B**	**X**
0	0	1
0	1	0
1	0	0
1	1	0

1 = HIGH, 0 = LOW.

A	**B**	$\overline{A + B} = X$
0	0	$\overline{0 + 0} = \overline{0} = 1$
0	1	$\overline{0 + 1} = \overline{1} = 0$
1	0	$\overline{1 + 0} = \overline{1} = 0$
1	1	$\overline{1 + 1} = \overline{1} = 0$

Example NOR gate

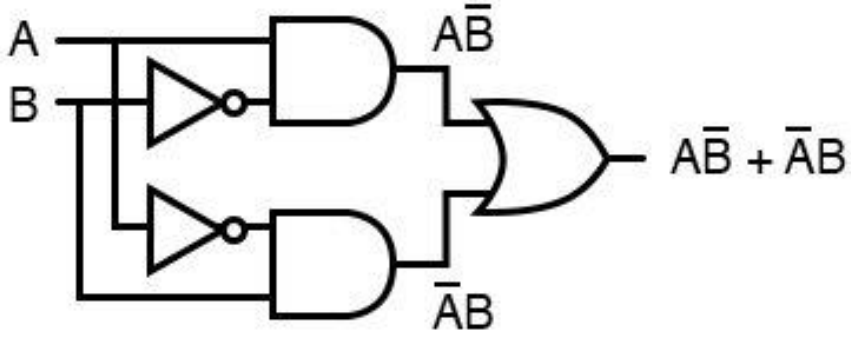




EX-OR Gate

There are two special gates, i.e., Ex-OR and Ex-NOR. These gates are not basic gates in their own and are constructed by combining with other logic gates.

. . . is equivalent to . .



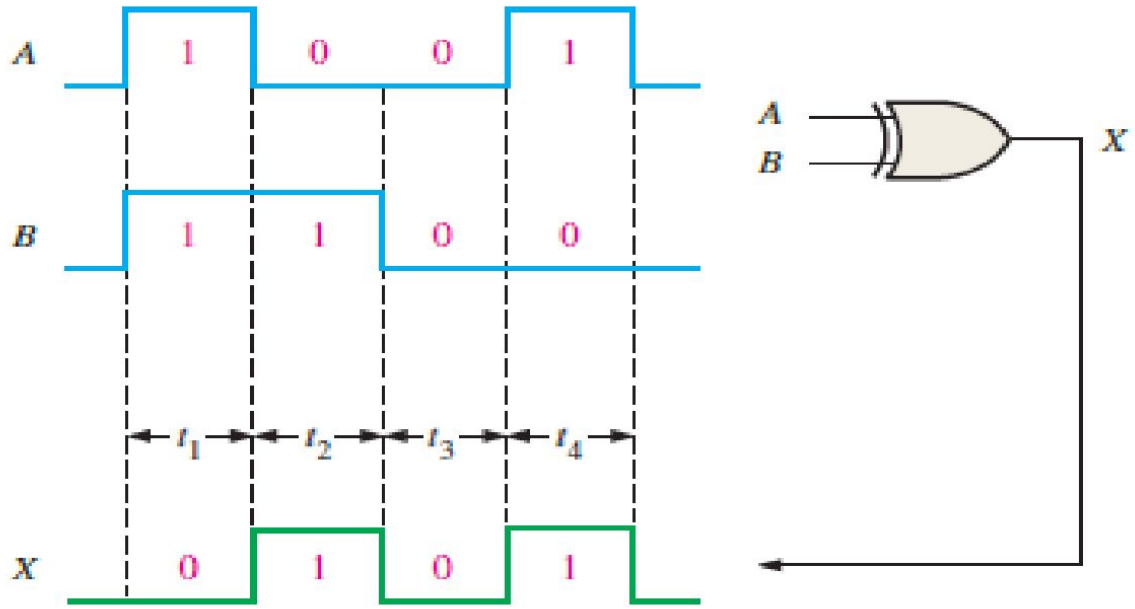
Truth Table

INPUTS		OUTPUTS
A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Logic Expressions for a XOR Gate:

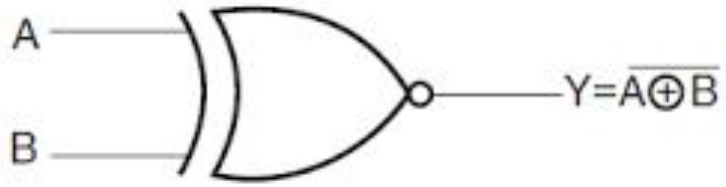
$$A \oplus B = A\bar{B} + \bar{A}B$$

EX-OR Gate



Example of exclusive-OR gate operation with pulse waveform inputs.

EX-NOR Gate

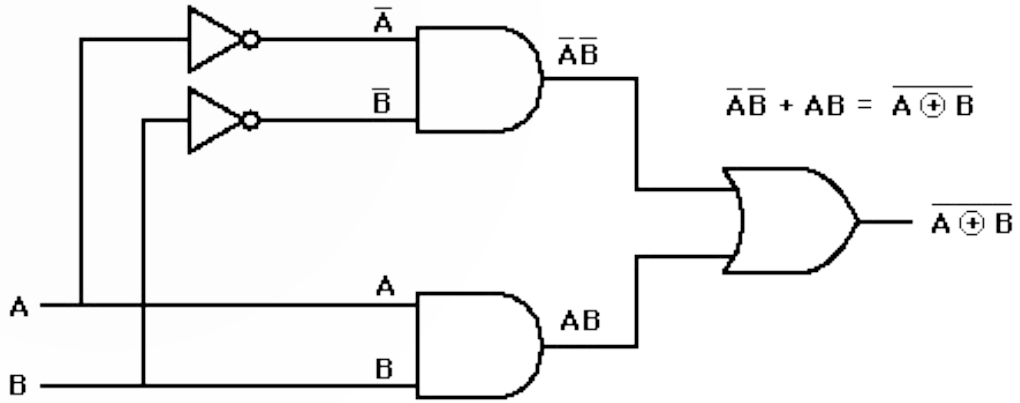


$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B})$$

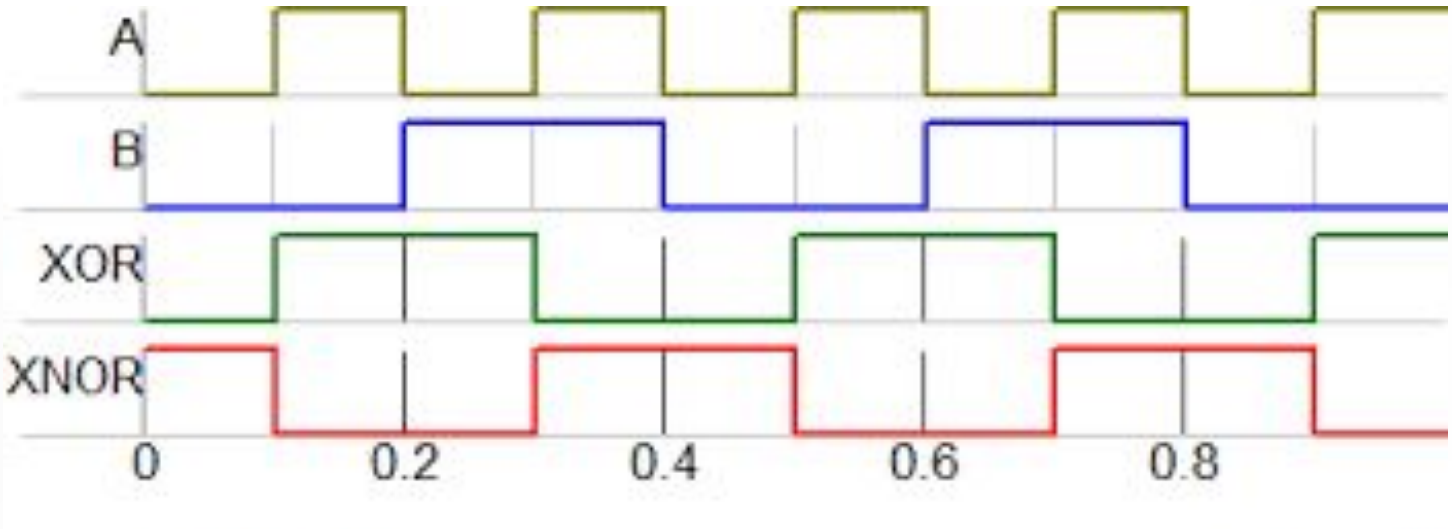
Truth

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

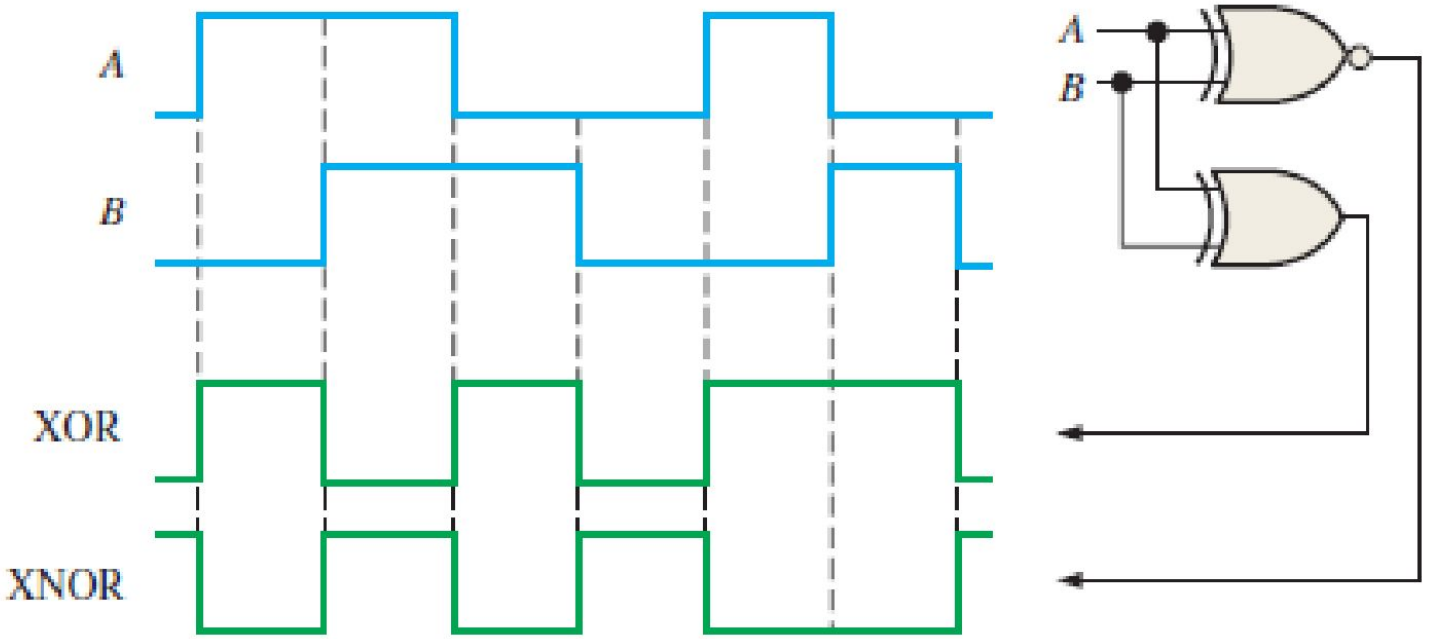
Logic Expressions for a XOR Gate:



EX-NOR Gate

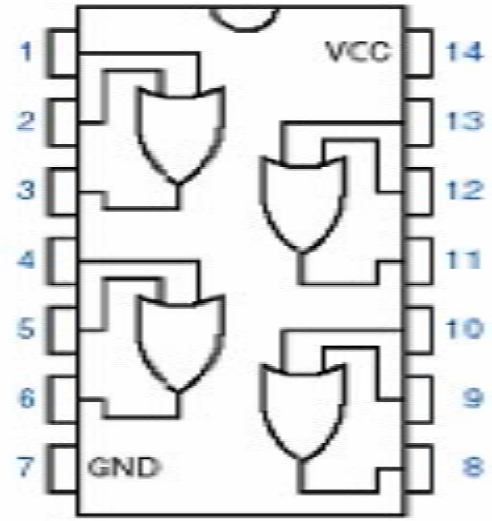


Determine the output waveform for XOR and XNOR gate for given input waveforms A and B



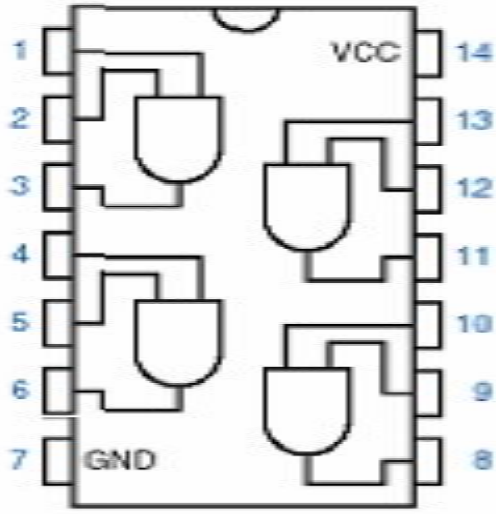
Pin Diagram

7432

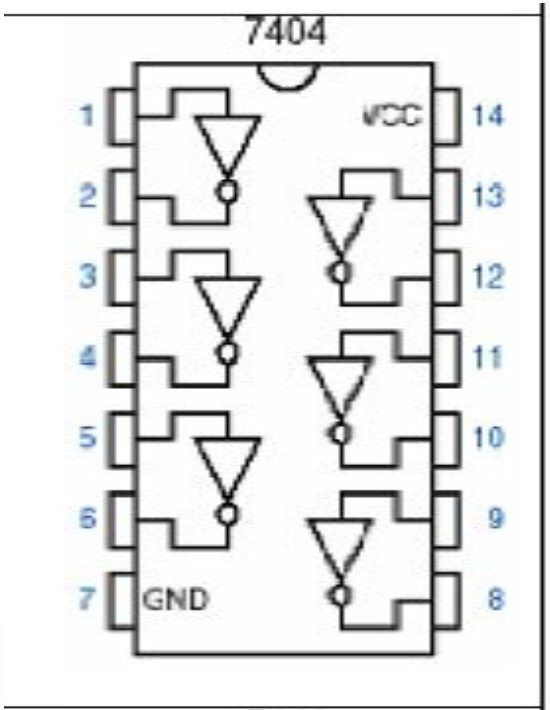


OR Gate

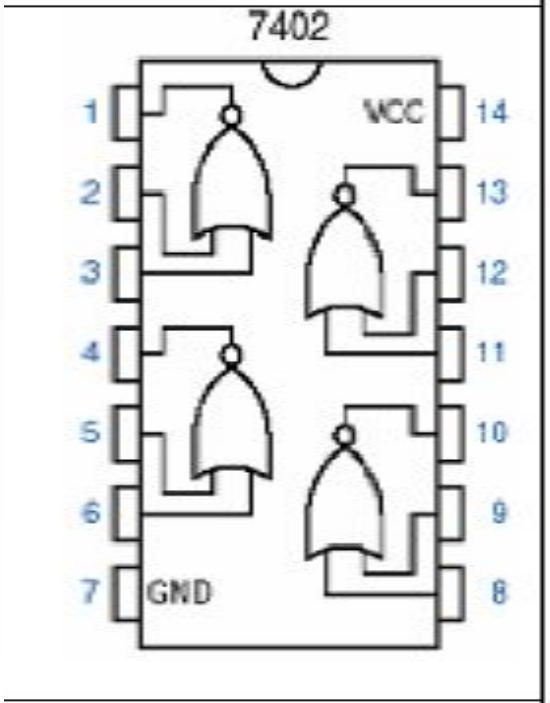
7408



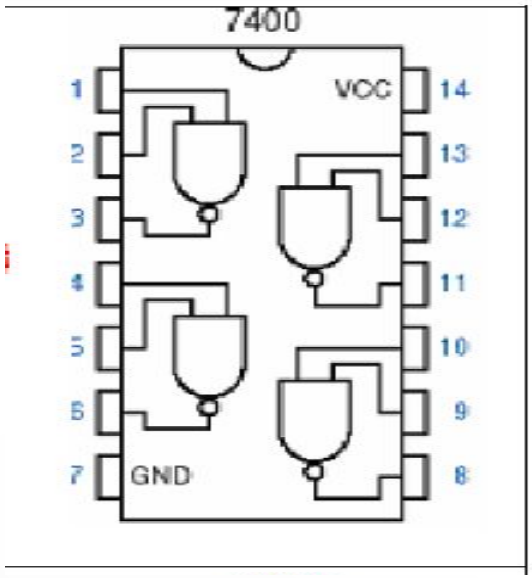
AND Gate



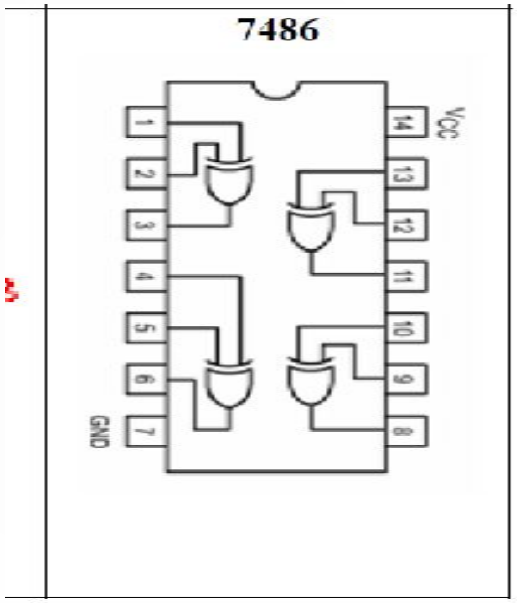
NOT Gate



NOR Gate



NAND Gate



XOR Gate

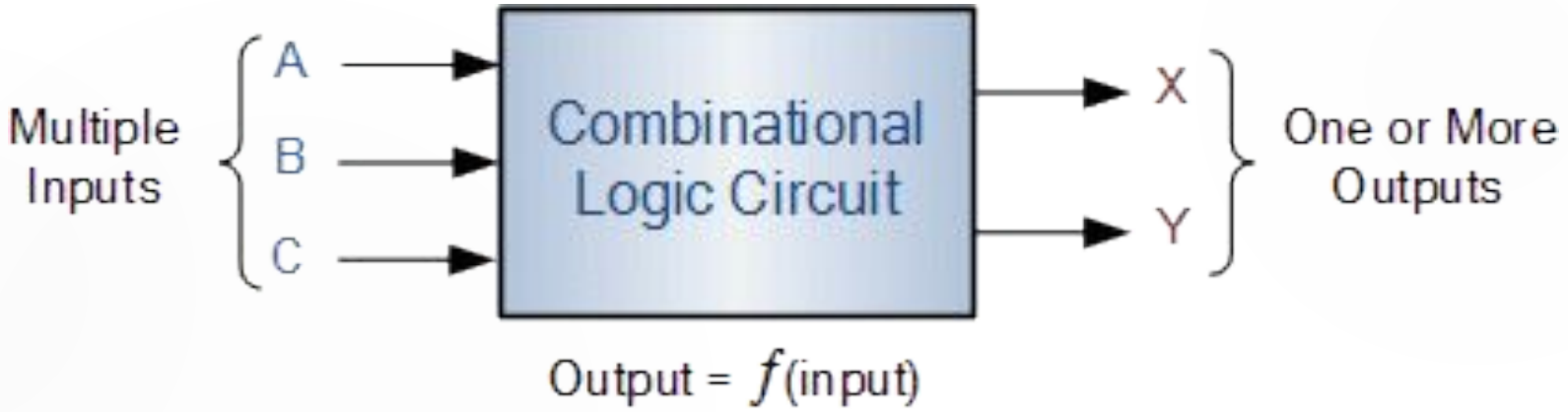
Digital Circuits

Basically, Digital Circuits are divided into two broad categories

- ✔ **Combinational circuits**
  - **Combinational Circuit** is the type of **circuit** in which output depend upon the input present at that particular instant.

- ✔ **Sequential circuits**
  - **Sequential circuit** is the type of **circuit** where output at any instant of time depend upon the current input as well as on the previous input/output.
  - It consists of memory element

Combinational Logic Circuit Representation



Combinational Logic Circuits

- Combinational logic circuits have no feedback, and any changes to the signals being applied to their inputs will immediately have an effect at the output.
- It has no “memory”, “timing” or “feedback loops”.
- The three main ways of specifying the function of a combinational logic circuit are:
  - **Boolean Expression** – This forms the algebraic expression showing the operation of the logic circuit
  - **Truth Table** – Shows all the output states in tabular form for each possible combination of input variable
  - **Logic Diagram** – This is a graphical representation of a logic circuit

Combinational Logic Circuit

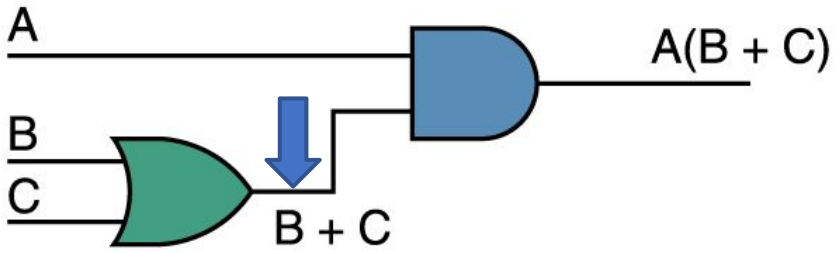
Boolean expression

$$A(B + C)$$

Truth Table

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

Logic Diagram



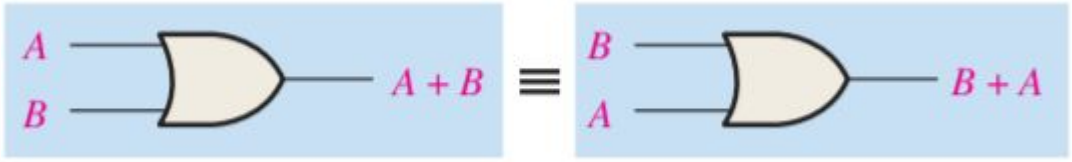
Boolean Algebra

Boolean algebra is the mathematics of digital logic

Commutative Laws :

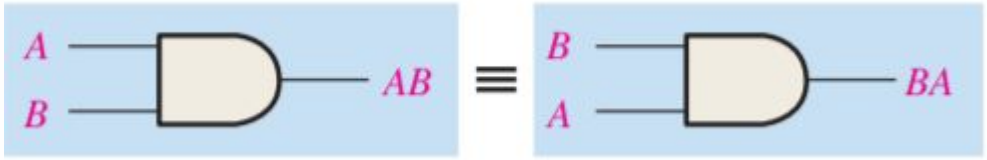
- The commutative law of addition for two variables is written as

$$A + B = B + A$$



- The commutative law of multiplication for two variables is

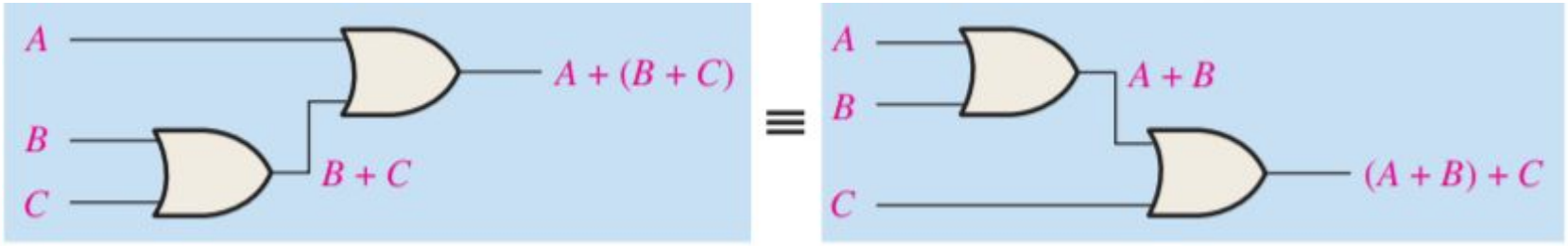
$$AB = BA$$



Associative Laws

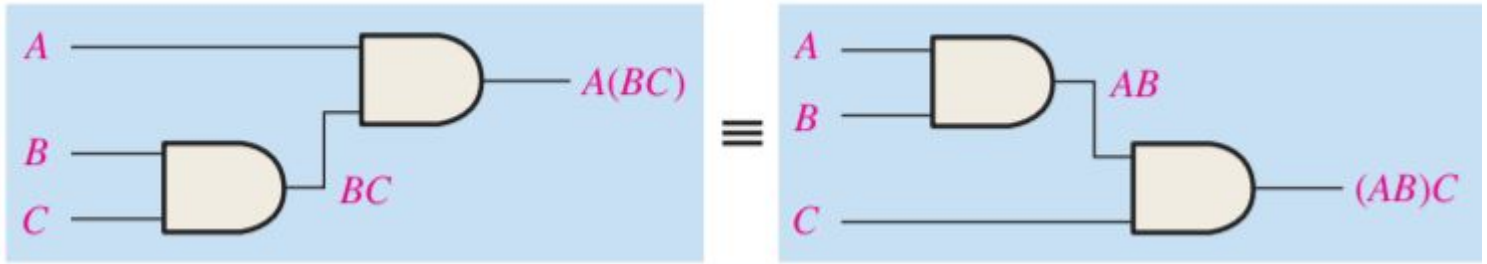
- The associative law of logical addition is written as follows for three variables:

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



- The associative law of logical multiplication is written as follows for three variables

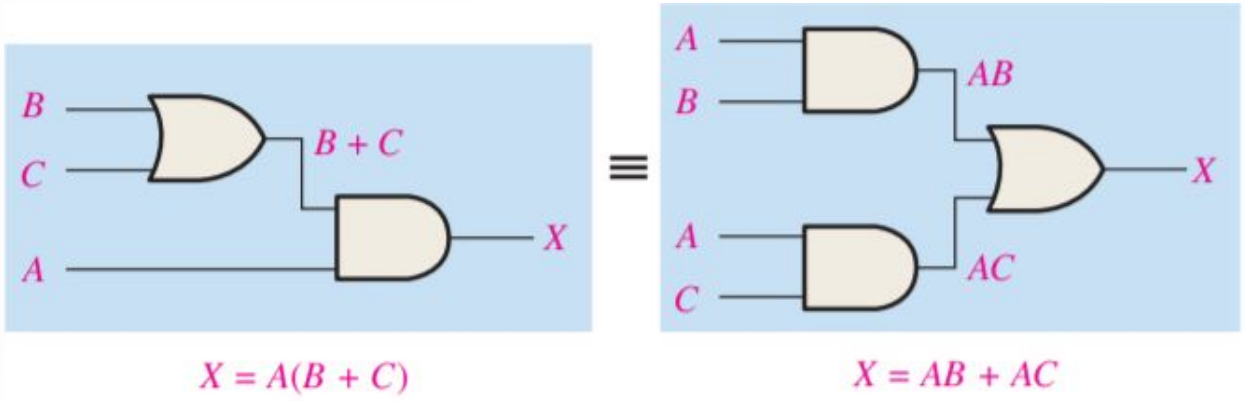
$$A(BC) = (AB)C$$



Distributive Law

- The distributive law is written for three variables as follows:

$$A(B + C) = AB + AC$$



A(B + C) = A.B + A.C (OR Distributive Law)

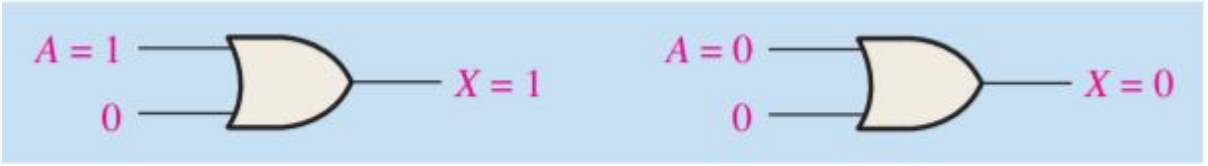
A + (B.C) = (A + B).(A + C) (AND Distributive Law)

Basic rules of Boolean algebra

Basic rules of Boolean algebra.

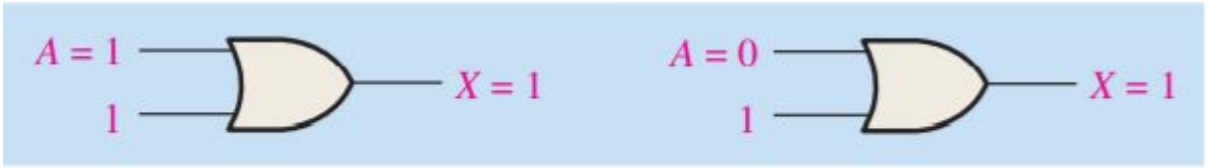
1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, or C can represent a single variable or a combination of variables.



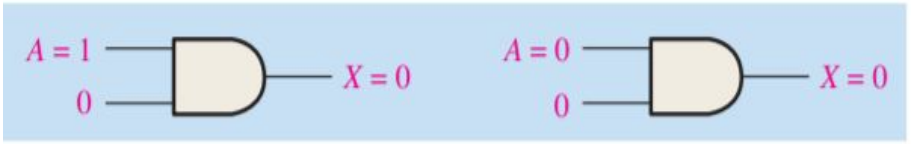
$X = A + 0 = A$

Rule 1



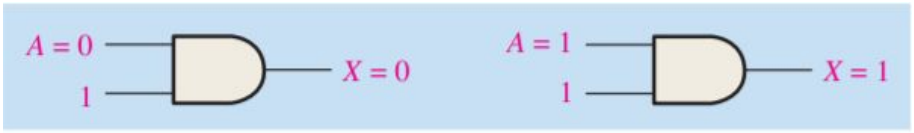
$X = A + 1 = 1$

Rule 2



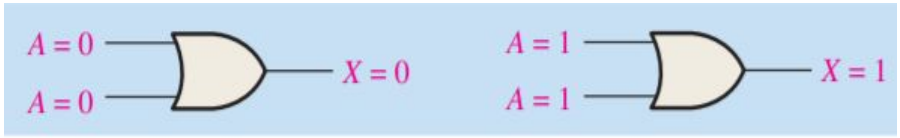
$X = A \cdot 0 = 0$

Rule 3



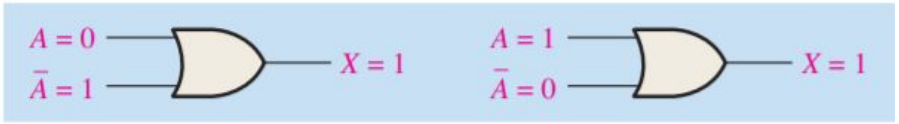
$X = A \cdot 1 = A$

Rule 4



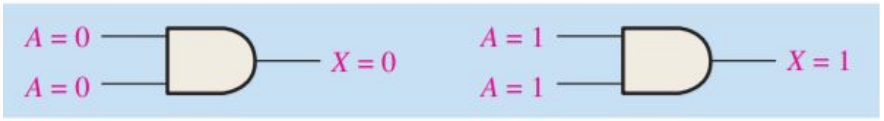
$X = A + A = A$

Rule 5



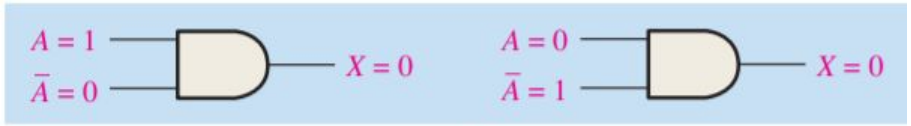
$X = A + \bar{A} = 1$

Rule 6



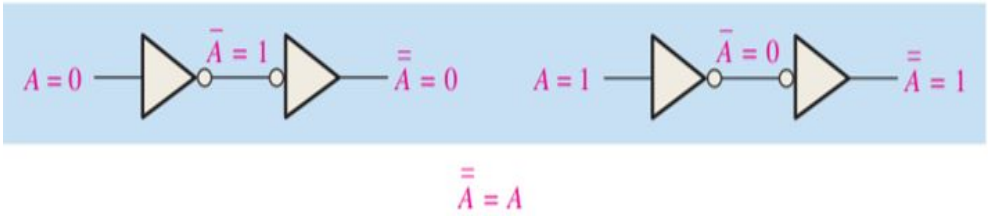
$X = A \cdot A = A$

Rule 7



$X = A \cdot \bar{A} = 0$

Rule 8



Rule 9

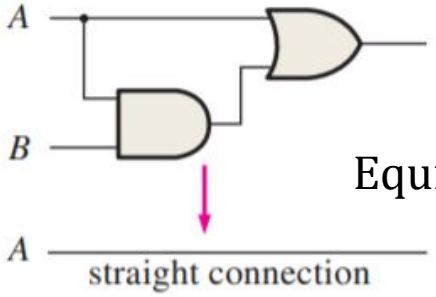
Rule 10: $A + AB = A$ This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

$$A + AB = A \cdot 1 + AB = A(1 + B) \quad \text{Factoring (distributive law)}$$
$$= A \cdot 1 \quad \text{Rule 2: } (1 + B) = 1$$
$$= A \quad \text{Rule 4: } A \cdot 1 = A$$

Rule 10: $A + AB = A$. Open file T04-02 to verify.

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

equal



Equivalence

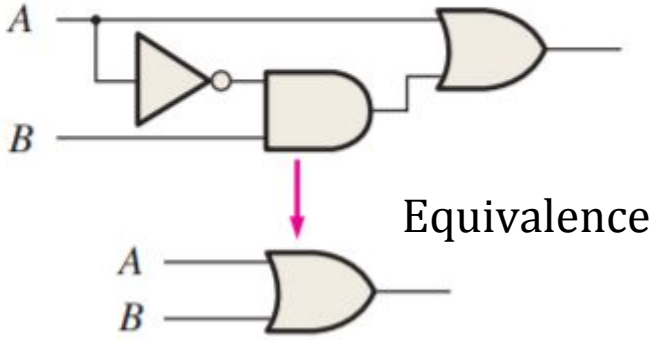
Rule 11: $A + \bar{A}B = A + B$ This rule can be proved as follows:

$$
\begin{aligned}
A + \bar{A}B &= (A + \bar{A})(A + B) && \text{Factoring} \\
&= 1 \cdot (A + B) && \text{Rule 6: } A + \bar{A} = 1 \\
&= A + B && \text{Rule 4: drop the 1}
\end{aligned}
$$

Rule 11: $A + \bar{A}B = A + B$. Open file T04-03 to verify.

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

equal



Equivalence

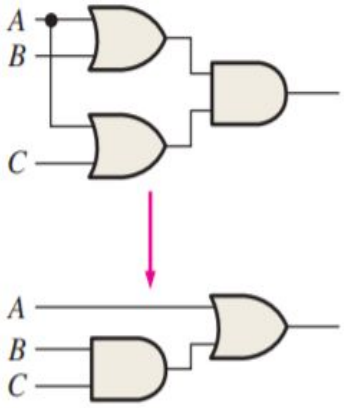
Rule 12: $(A + B)(A + C) = A + BC$ This rule can be proved as follows:

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC && \text{Distributive law} \\
&= A + AC + AB + BC && \text{Rule 7: } AA = A \\
&= A(1 + C) + AB + BC && \text{Factoring (distributive law)} \\
&= A \cdot 1 + AB + BC && \text{Rule 2: } 1 + C = 1 \\
&= A(1 + B) + BC && \text{Factoring (distributive law)} \\
&= A \cdot 1 + BC && \text{Rule 2: } 1 + B = 1 \\
&= A + BC && \text{Rule 4: } A \cdot 1 = A
\end{aligned}
$$

Rule 12: $(A + B)(A + C) = A + BC$. Open file T04-04 to verify.

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



De Morgan's theorem

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

OR

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

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

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

OR

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

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

De Morgan's Theorem 1

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

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

EQUAL

NAND = Bubbled OR

De Morgan's Theorem 2

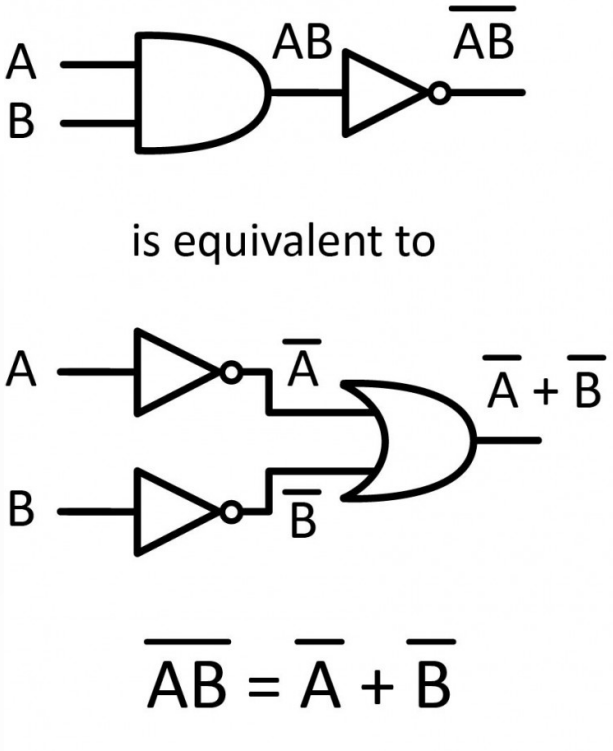
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

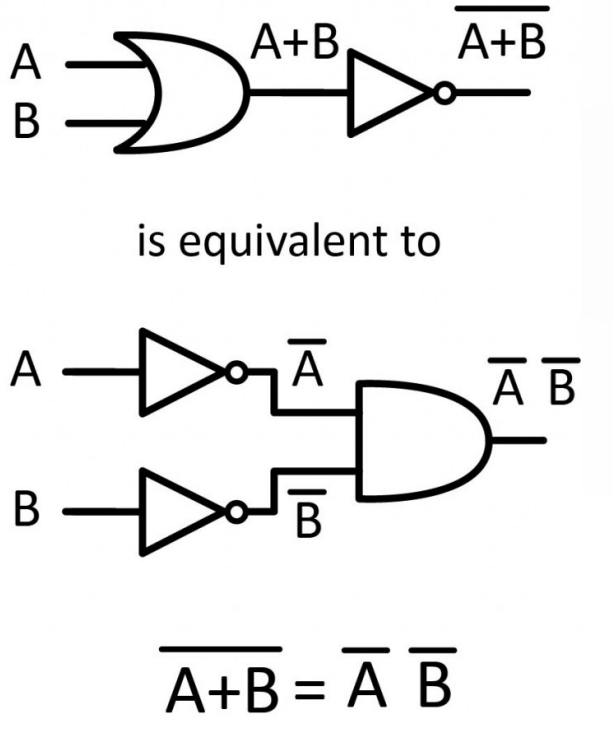
A	B	A + B	$\overline{A + B}$		$\overline{A}$	$\overline{B}$	$\overline{A} \times \overline{B}$
0	0	0	1		1	1	1
0	1	1	0		1	0	0
1	0	1	0		0	1	0
1	1	1	0		0	0	0

EQUAL

NOR = Bubbled AND

De Morgan's Law



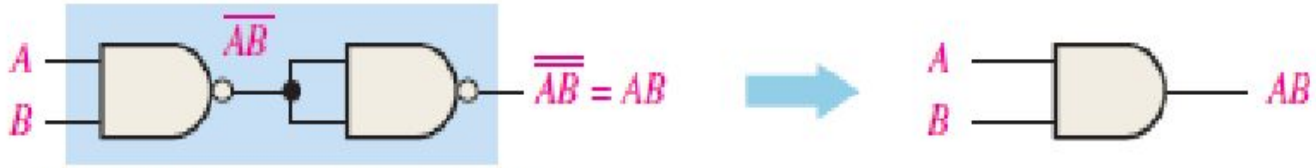


Summary

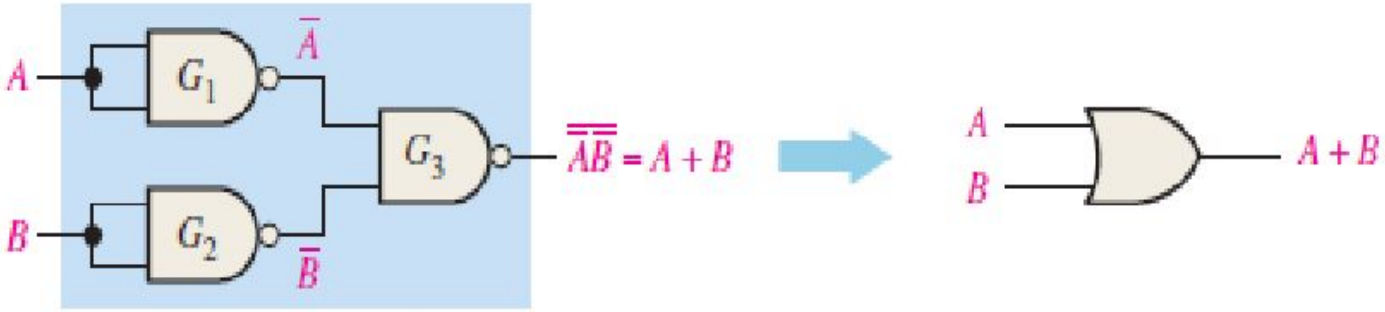
- NAND is the same as OR with complemented inputs
- NOR is the same as AND with complemented inputs

NAND gate as universal logic gate

(a) One NAND gate used as an inverter



(b) Two NAND gates used as an AND gate



(c) Three NAND gates used as an OR gate

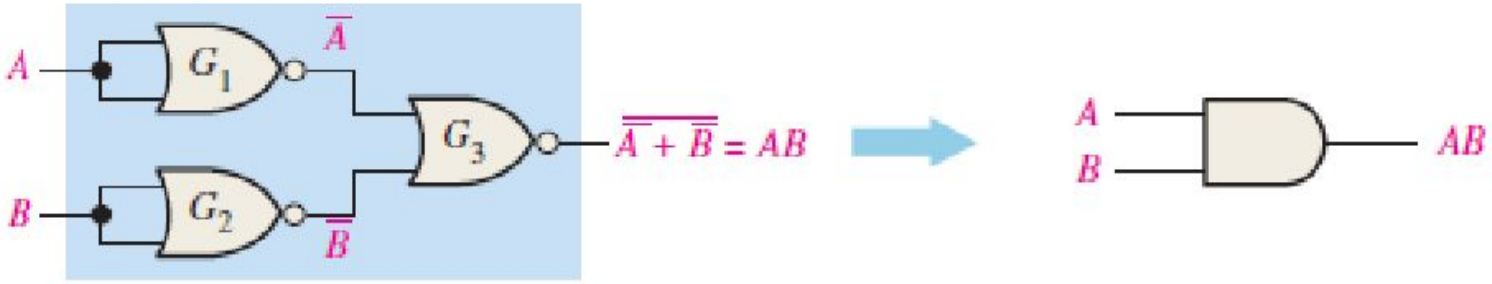
OR gate using NAND gate

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

NOR gate as universal logic gate

(a) One NOR gate used as an inverter

(b) Two NOR gates used as an OR gate



(c) Three NOR gates used as an AND gate

Exercise

Apply DeMorgan's theorem to the expression $\overline{\overline{X} + \overline{Y} + \overline{Z}}$.

Apply DeMorgan's theorem to the expression $\overline{\overline{W}\overline{X}\overline{Y}\overline{Z}}$.

Apply DeMorgan's theorems to each of the following expressions:

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

(b) $\overline{ABC + DEF}$

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

Exercise- Solution

Apply DeMorgan's theorem to the expression $\overline{\overline{X} + \overline{Y} + \overline{Z}}$.

= X .Y . Z

Apply DeMorgan's theorem to the expression $\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$.

= W + X + Y + Z

Apply DeMorgan's theorems to each of the following expressions:

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

(b) $\overline{ABC + DEF}$ $= \overline{A\ B\ C} \cdot \overline{D\ E\ F} = (\overline{A} + \overline{B} + \overline{C}) \cdot (\overline{D} + \overline{E} + \overline{F})$

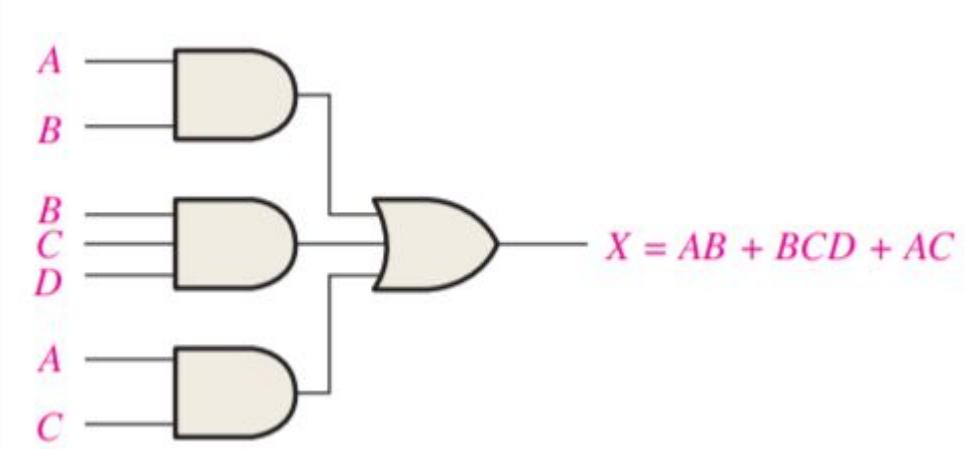
(c) $\overline{A\overline{B} + \overline{C}D + EF}$

Standard Forms of Boolean Expressions

1. Sum-of-products form (SOP)
2. Product-of-sums form (POS)

The Sum-of-Products (SOP) Form:

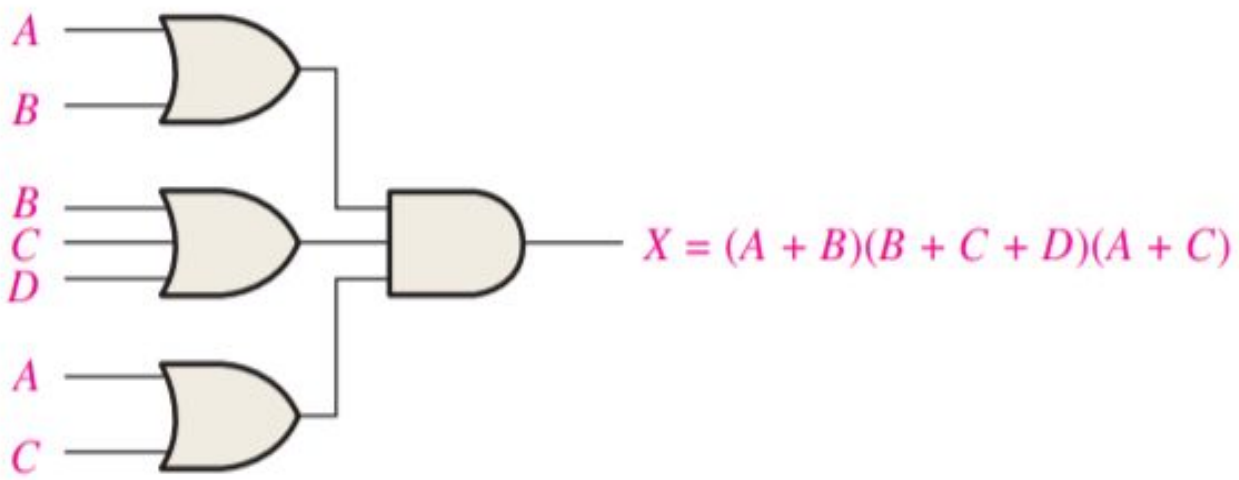
- An SOP expression can be implemented with one OR gate and two or more AND gates.



Implementation of the SOP expression $AB + BCD + AC$.

The Product-of-Sums (POS) Form

- When two or more sum terms are multiplied, the resulting expression is a product-of-sums (POS).
- Implementing the **POS** expression simply requires **AND** ing the outputs of two or more **OR** gates

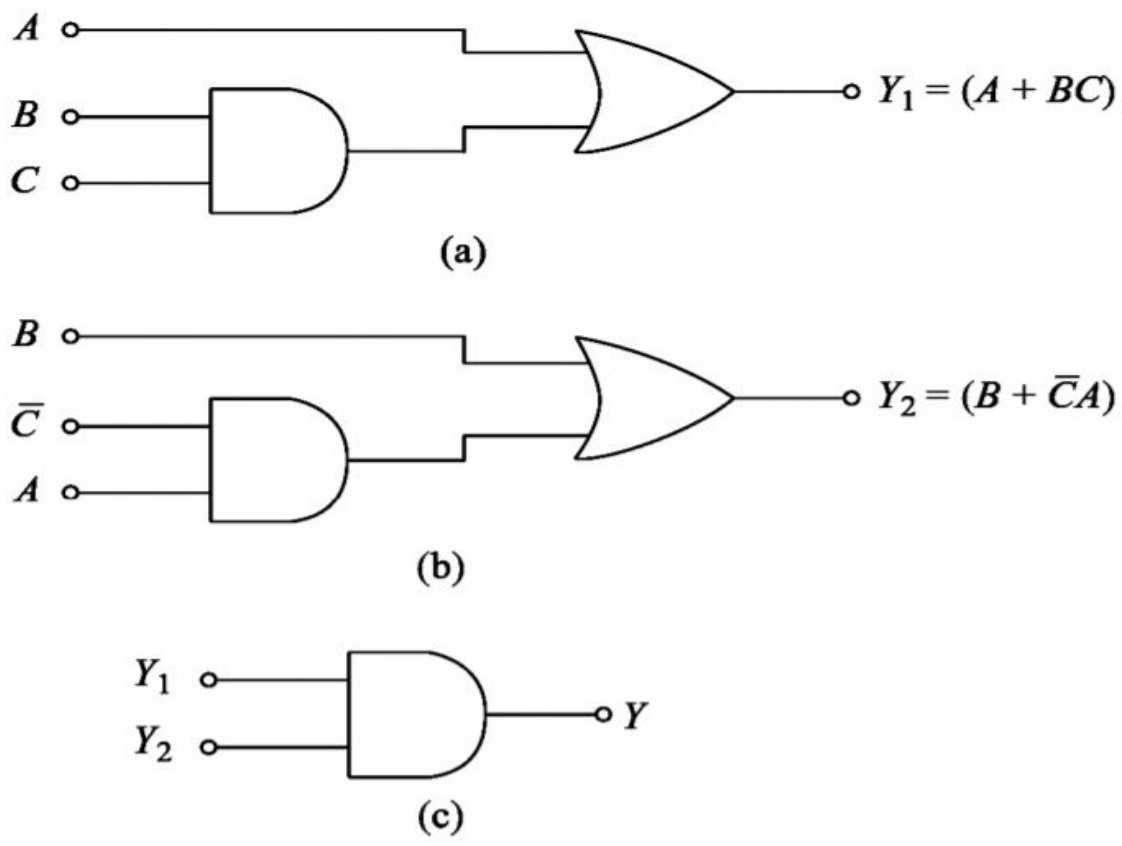


Implementation of the POS expression $(A + B)(B + C + D)(A + C)$.

Combinational Circuit Realization

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

Realization of logic expression (3 level):



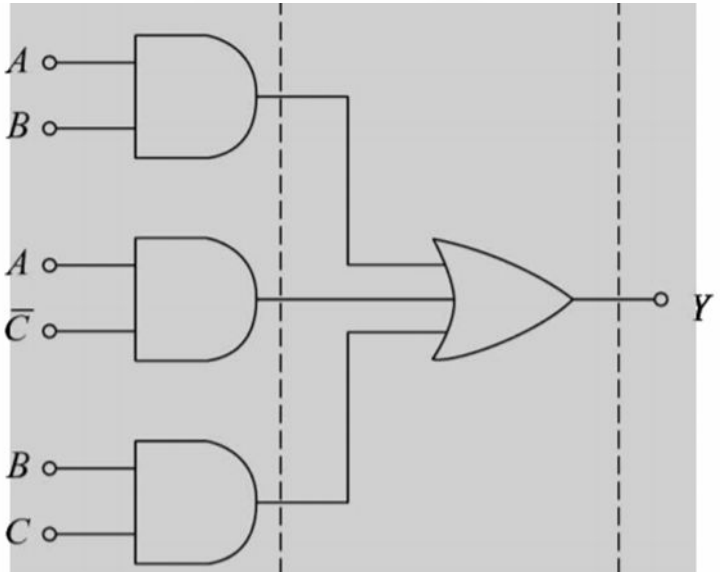
Combinational Circuit realization (SOP)

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

SOP form:

$$\begin{aligned} Y &= A(B + \overline{C}A) + (BC)(B + \overline{C}A) \\ &= AB + A\overline{C}A + BCB + BC\overline{C}A \\ &= AB + A\overline{C} + BC \end{aligned}$$

$$\begin{aligned} A\overline{C}A &= (A \cdot A) \cdot \overline{C} = A\overline{C} && \text{(Theorem 1.6)} \\ BCB &= (B \cdot B) \cdot C = BC && \text{(Theorem 1.6)} \\ BC\overline{C}A &= B \cdot (C \cdot \overline{C}) \cdot A = B \cdot 0 \cdot A && \text{(Theorem 1.8)} \\ &= 0 && \text{(Theorem 1.4)} \end{aligned}$$



2 level realization

SOP realization using only one type of gate(NAND)

$$\overline{\overline{Y}} = \overline{\overline{AB + A\overline{C} + BC}}$$

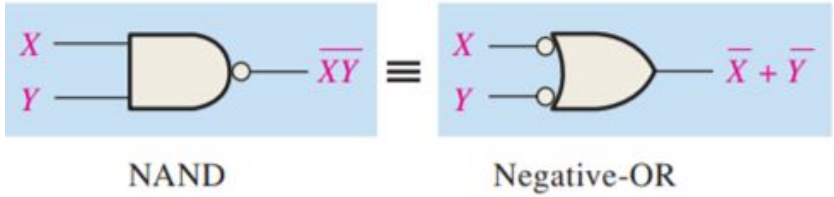
$$= \overline{\overline{AB} \cdot \overline{A\overline{C}} \cdot \overline{BC}}$$

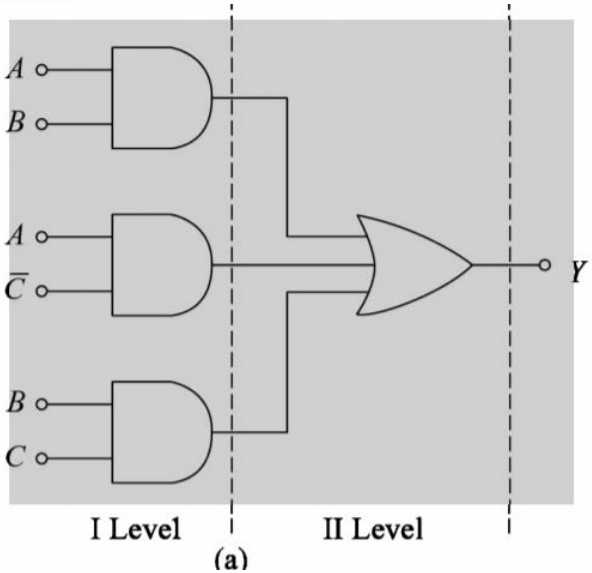
$$Y = \overline{Y_1 \cdot Y_2 \cdot Y_3}$$

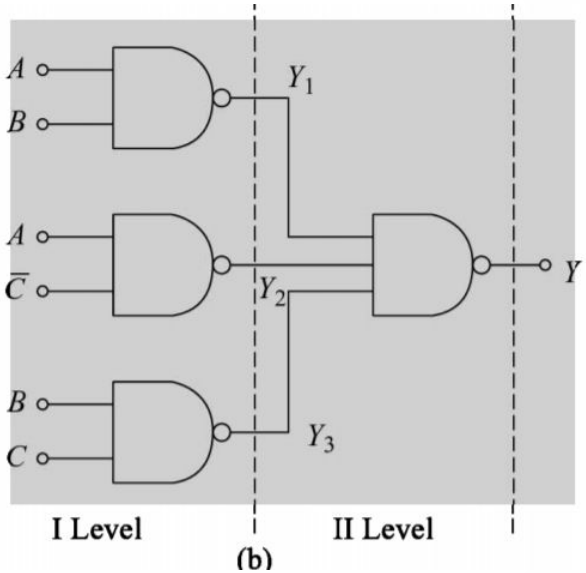
$$Y_1 = \overline{AB}$$

$$Y_2 = \overline{A\overline{C}}$$

$$Y_3 = \overline{BC}$$







SOP realized using NAND gates only

Boolean Algebra Rules

Basic rules of Boolean algebra.

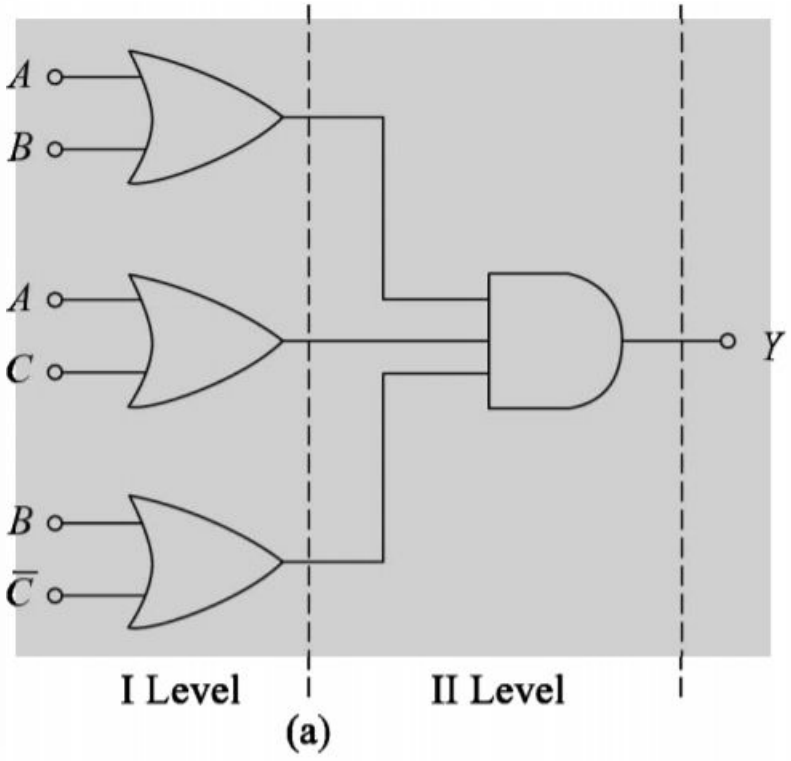
1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, or C can represent a single variable or a combination of variables.

Combinational Circuit Realization(POS)

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

$$Y = (A+B)(A+C)(B+\overline{C})(B+A) \qquad \text{(Theorem 1.10)}$$
$$= (A+B)(A+C)(B+\overline{C}) \qquad \text{(Theorem 1.6)}$$



(a)

POS realization using only one type of gate(NOR)

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

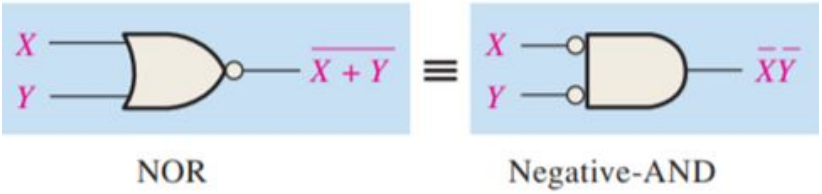
$$= \overline{\overline{(A+B)} + \overline{(A+C)} + \overline{(B+\overline{C})}}$$

$$Y = \overline{Y_A + Y_B + Y_C}$$

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

$$Y_B = \overline{A+C}$$

$$Y_C = \overline{B+\overline{C}}$$



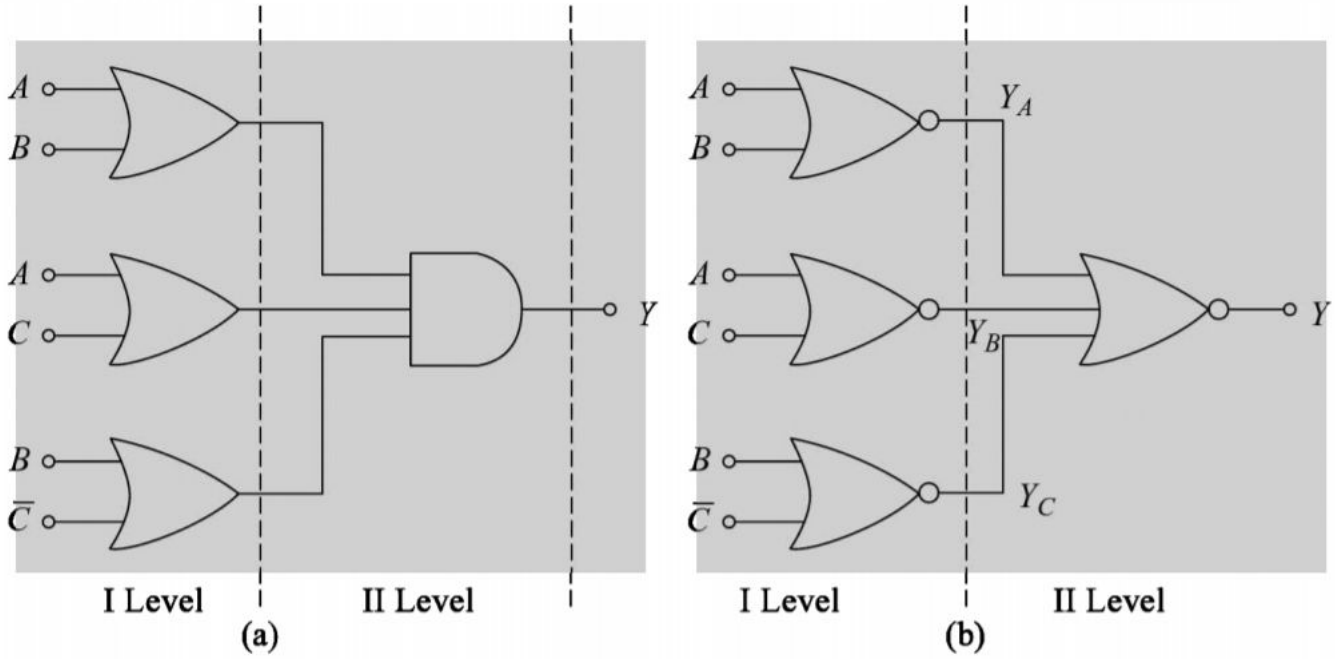
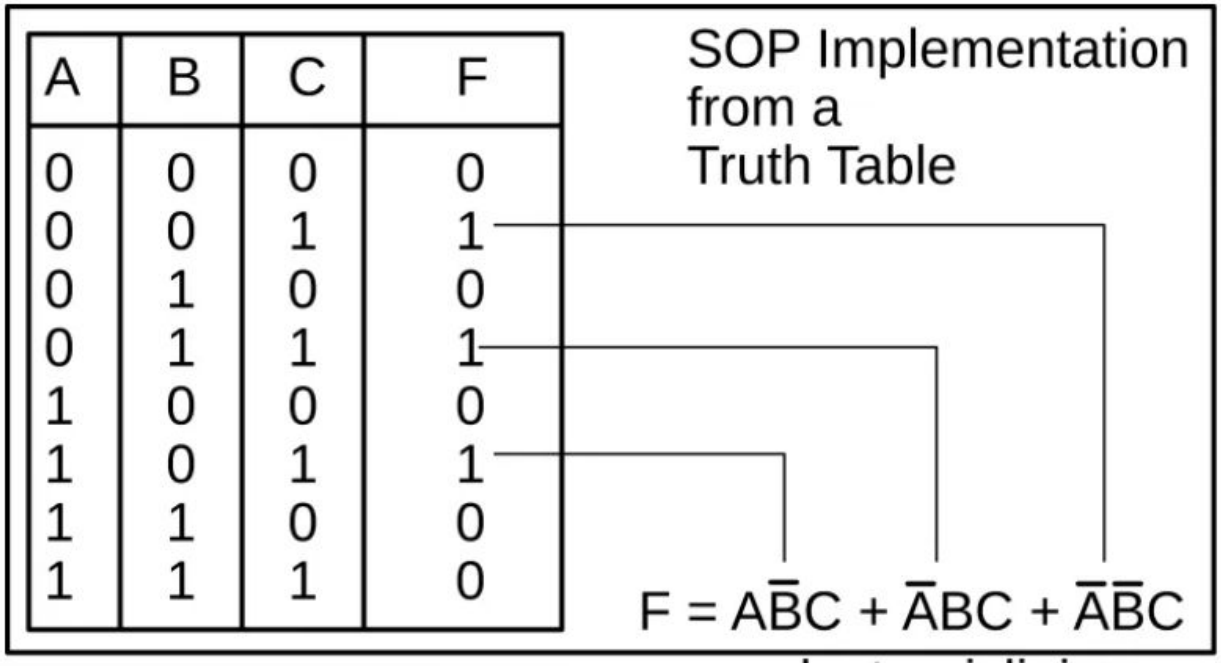
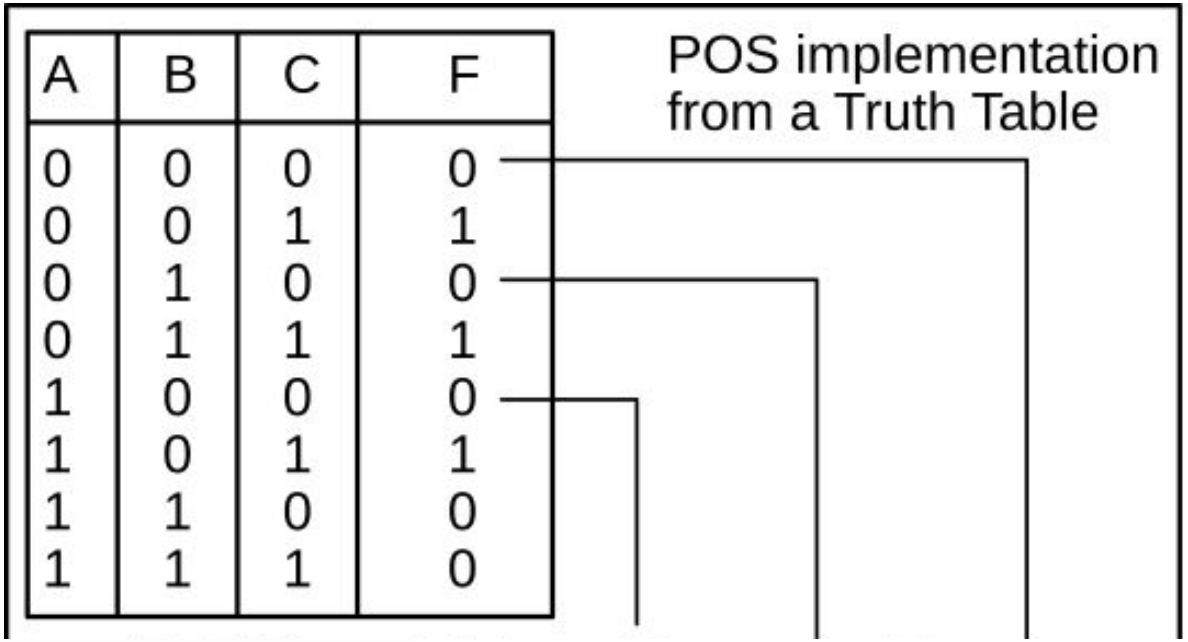


Fig. 5.3 ***Realisation of Eq. (5.5) Using (a)*** OR–AND ***(b)*** NOR–NOR

Construct SOP form expression from a Truth Table



Construct POS form expression from a Truth Table



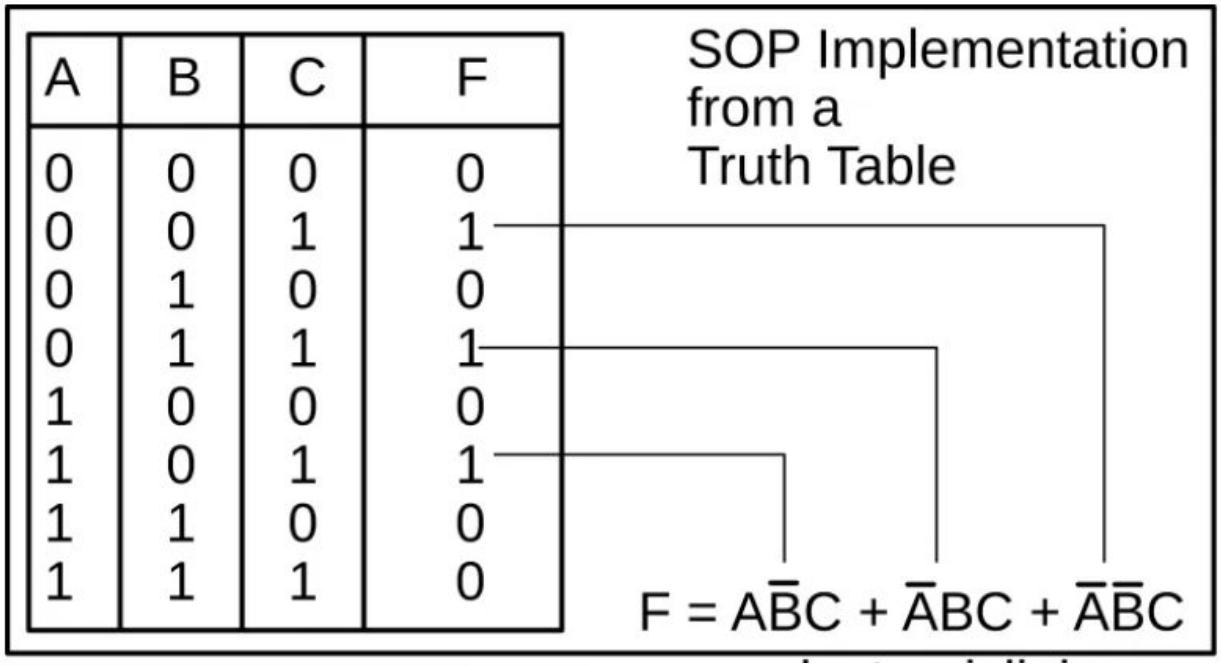
$$F = (\overline{A} + \overline{B} + \overline{C})\,(\overline{A} + \overline{B} + C)\,(\overline{A} + B + C)\,(A + \overline{B} + C)\,(A + B + C)$$

Canonical SOP form and POS form

- Each individual term in canonical SOP form is called as **minterm** and in canonical POS form as **maxterm**

			Minterms		Maxterms	
X	Y	Z	Term	Designation	Term	Designation
0	0	0	$\bar{X}\bar{Y}\bar{Z}$	mo	$X+Y+Z$	Mo
0	0	1	$\bar{X}\bar{Y}Z$	m1	$X+Y+\bar{Z}$	M1
0	1	0	$\bar{X}Y\bar{Z}$	m2	$X+\bar{Y}+Z$	M2
0	1	1	$\bar{X}YZ$	m3	$X+\bar{Y}+\bar{Z}$	M3
1	0	0	$X\bar{Y}\bar{Z}$	m4	$\bar{X}+Y+Z$	M4
1	0	1	$X\bar{Y}Z$	m5	$\bar{X}+Y+\bar{Z}$	M5
1	1	0	$XY\bar{Z}$	m6	$\bar{X}+\bar{Y}+Z$	M6
1	1	1	XYZ	m7	$\bar{X}+\bar{Y}+\bar{Z}$	M7

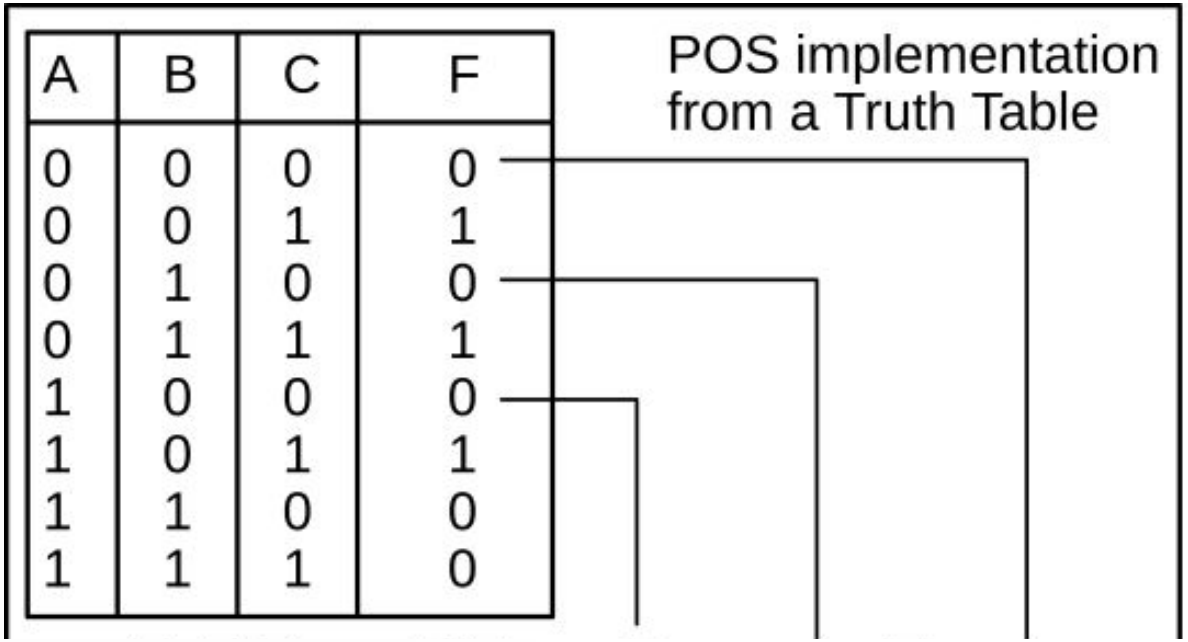
Shorthand form of canonical SOP using minterms



$F = m1 + m3 + m5$

$F = \Sigma m\ (1, 3, 5)$

Shorthand form of canonical POS using maxterms



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

$F = M0 \bullet M2 \bullet M4 \bullet M6 \bullet M7$

$F = \Pi\, M\,(0, 2, 4, 6, 7)$

Four Variables

A	B	C	D	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	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	15

Complementary Property of minterms and maxterms

$$Y = \Sigma m(0, 3, 6, 7, 10, 12, 15)$$

then $$Y = \Pi M(1, 2, 4, 5, 8, 9, 11, 13, 14)\,.$$

Simplification of Boolean expression

- The required Boolean results are transferred from a truth table onto a two-dimensional grid where, in Karnaugh maps, the cells are ordered in Gray code,[6][4] and each cell position represents one combination of input conditions, while each cell value represents the corresponding output value. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table.[7] These terms can be used to write a minimal Boolean expression representing the required logic.

The Karnaugh Map(K map) Technique

- A Karnaugh map technique provides a **graphical,** systematic method for simplifying Boolean/ logic expressions
- Karnaugh map is an array of cells in which each cell represents a binary value of the input variables
- K-map produce the simplest SOP or POS expression possible, known as the minimum expression
- The information contained in the truth table or available in SOP/ POS form is represented on K map
- Karnaugh maps can be used easily used as a tool to simplify expressions with two, three, four , five variables

K-maps

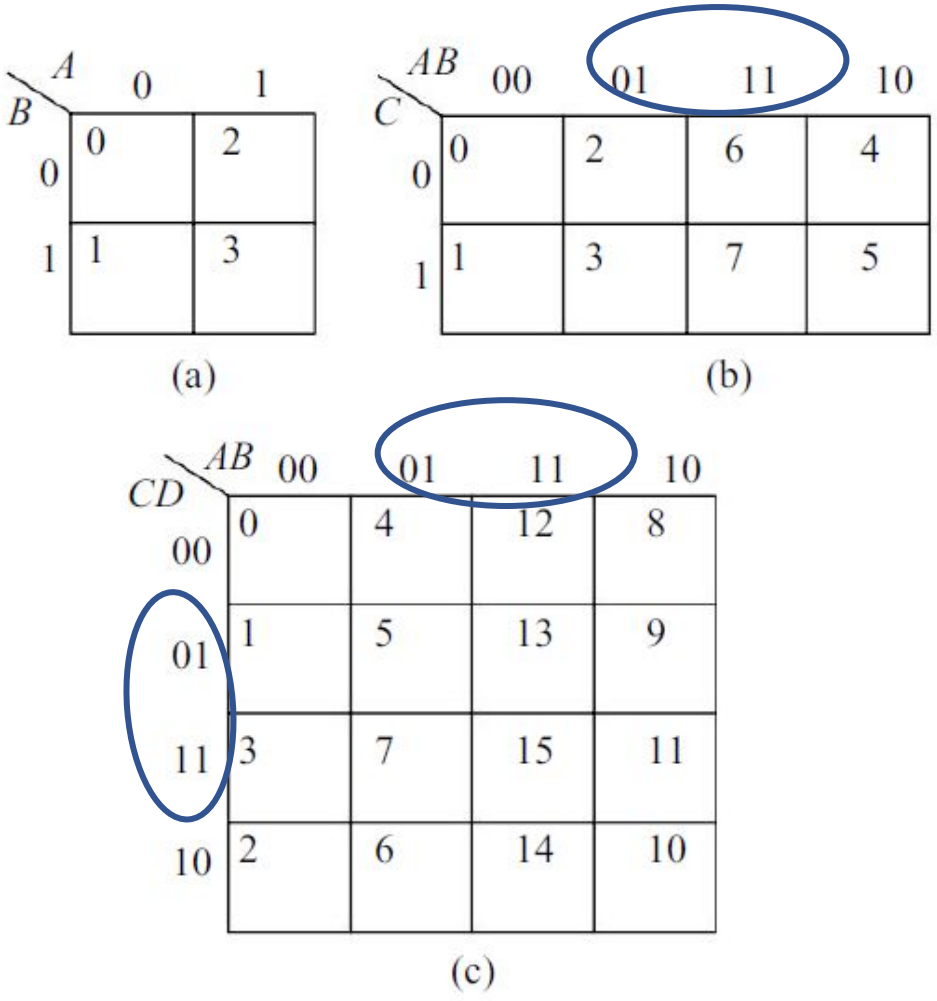


Fig. 5.5 *Karnaugh-maps (a) Two-variable (b) Three-variable (c) Four-variable*

K-maps adjacencies wrap around edges

- Wrap from first to last column
- Wrap top row to bottom row

- Numbering scheme is based on **Gray–code** (only a single bit changes in code for adjacent map cells

- K-maps are hard to draw and visualize for more than 4 dimensions, and virtually impossible for more than 6 dimensions

minterms/ maxterms written inside the corresponding cell in the K-map

A \ B	0	1
0	$\bar{A}\bar{B}$	$A\bar{B}$
1	$\bar{A}B$	AB

minterms (a)

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

(b) Maxterms

Two variable K-map

AB \ C	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

(c)

AB \ C	00	01	11	10
0	$A+B+C$	$A+\bar{B}+C$	$\bar{A}+\bar{B}+C$	$\bar{A}+B+C$
1	$A+B+\bar{C}$	$A+\bar{B}+\bar{C}$	$\bar{A}+\bar{B}+\bar{C}$	$\bar{A}+B+\bar{C}$

Three variable K-map

Minterms and maxterms for each cell in Four Variable K-map

CD \ AB	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}B\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$	$A\bar{B}\bar{C}\bar{D}$
01	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}B\bar{C}D$	$AB\bar{C}D$	$A\bar{B}\bar{C}D$
11	$\bar{A}\bar{B}CD$	$\bar{A}BCD$	$ABCD$	$A\bar{B}CD$
10	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}BC\bar{D}$	$ABC\bar{D}$	$A\bar{B}C\bar{D}$

(e)

CD \ AB	00	01	11	10
00	$A+B+C+D$	$A+\bar{B}+C+D$	$\bar{A}+\bar{B}+C+D$	$\bar{A}+B+C+D$
01	$A+B+C+\bar{D}$	$A+\bar{B}+C+\bar{D}$	$\bar{A}+\bar{B}+C+\bar{D}$	$\bar{A}+B+C+\bar{D}$
11	$A+B+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	$\bar{A}+B+\bar{C}+\bar{D}$
10	$A+B+\bar{C}+D$	$A+\bar{B}+\bar{C}+D$	$\bar{A}+\bar{B}+\bar{C}+D$	$\bar{A}+B+\bar{C}+D$

(f)

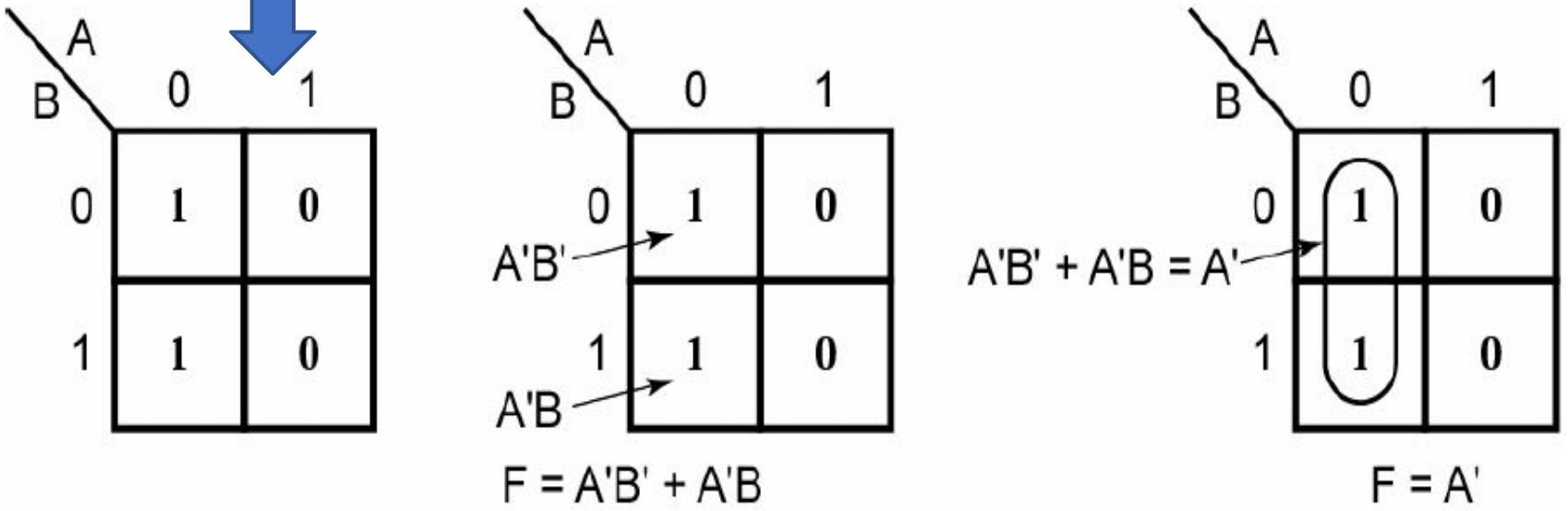
Procedure for Obtaining Simplified Boolean equation from K-map

1. Identify adjacent ones for grouping (Group size 2, 4, 8)
2. See the values of the variables associated with these cells
3. **Only one variable will be different** and it gets eliminated
4. Other variables will appear in ANDed form in the term, it will be in the uncomplemented form if it is 1 and in complemented form if it is 0.
5. Determine the term corresponding to each group of adjacent ones.
6. These terms are ORed to get the simplified equation in SOP form.

C \ AB	00	01	11	10
0	0 0	2 1	6 0	4 1
1	1 1	3 0	7 1	5 0

K-map examples (2 variable)

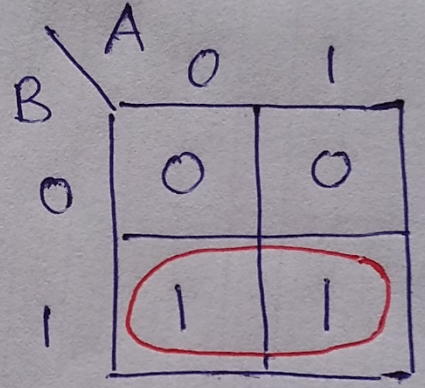
A	B	F
0	0	1
0	1	1
1	0	0
1	1	0



A' is A bar and B' is B bar

K Map – 2 variables examples

A	B	F
0	0	0
0	1	1
1	0	0
1	1	1



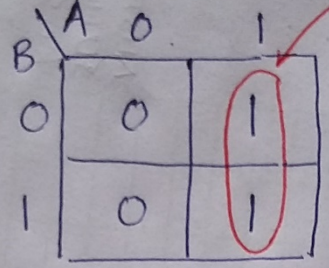
Adjacent 1's grouped.

F = B.

Variable A eliminated

K Map – 2 variables examples

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1



Adjacent 1's grouped.

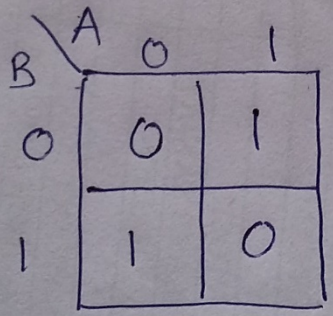
$F = AB' + AB$

$= A(B + B')$

$= A$.

Variable B eliminated.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



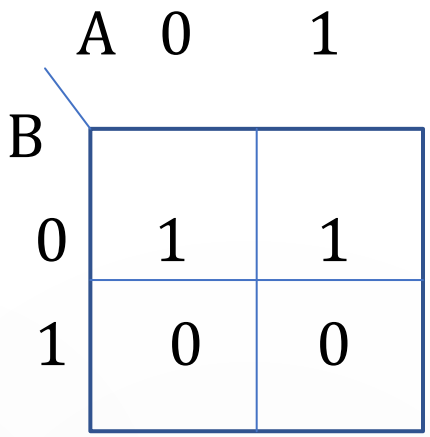
No adjacent 1's

Diagonal grouping

Not allowed.

$F = AB' + A'B$.

K Map – 2 variables examples

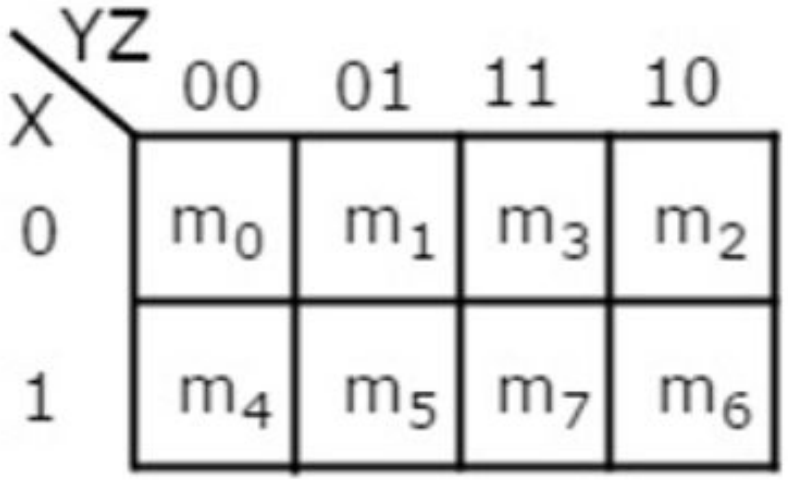


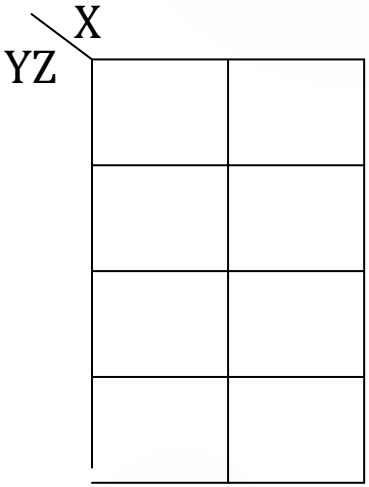
Find the Boolean expression

K-map examples (3 variable)

x	y	z	F
0	0	0	m0
0	0	1	m1
0	1	0	m2
0	1	1	m3
1	0	0	m4
1	0	1	m5
1	1	0	m6
1	1	1	m7

3-variable Truth table of F





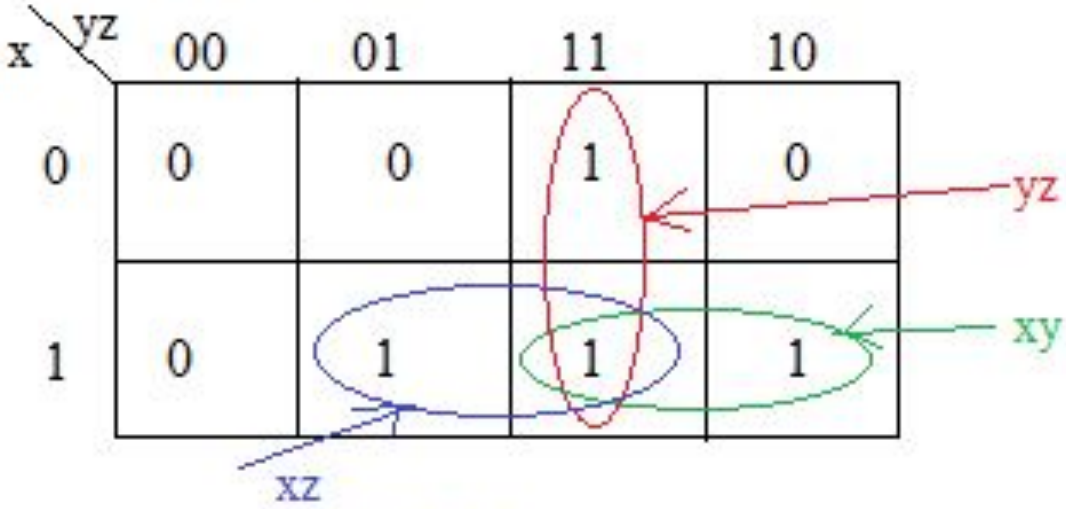
K-map examples (3 variable)

x	y	z	F	minterm
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	m3
1	0	0	0	
1	0	1	1	m5
1	1	0	1	m6
1	1	1	1	m7

Truth Table of 3-variable Majority Function

x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

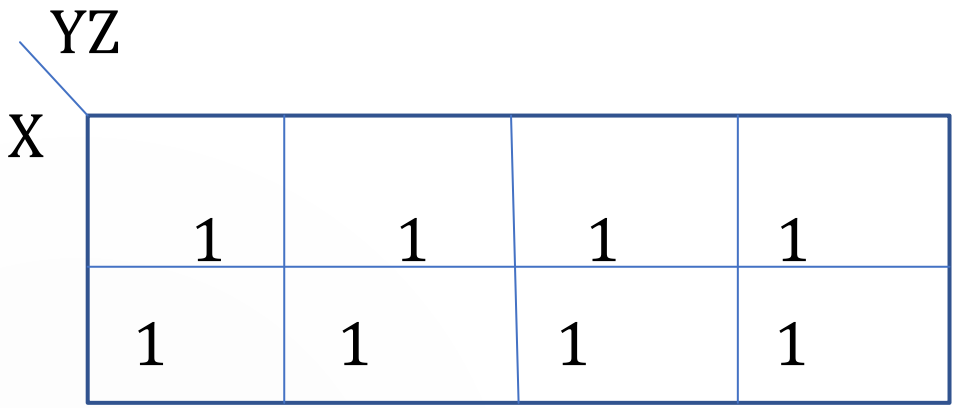
K-Map of Majority Function



Pair in K-Map

Solution is : F = yz + xz + xy

-
-
-



F=1

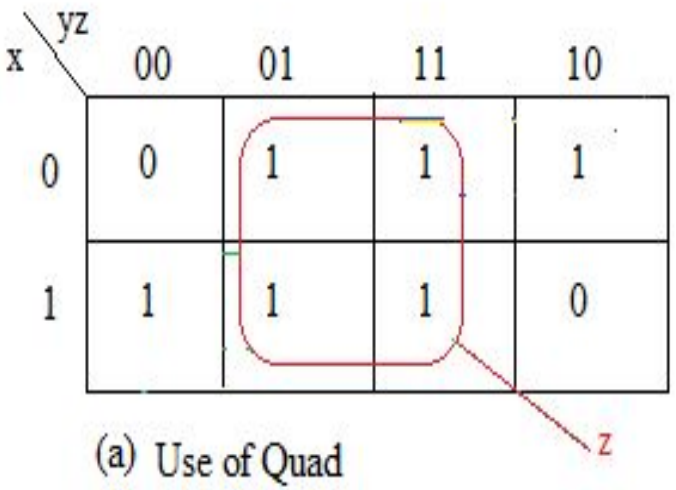
K-map examples (3 variable)

Example:

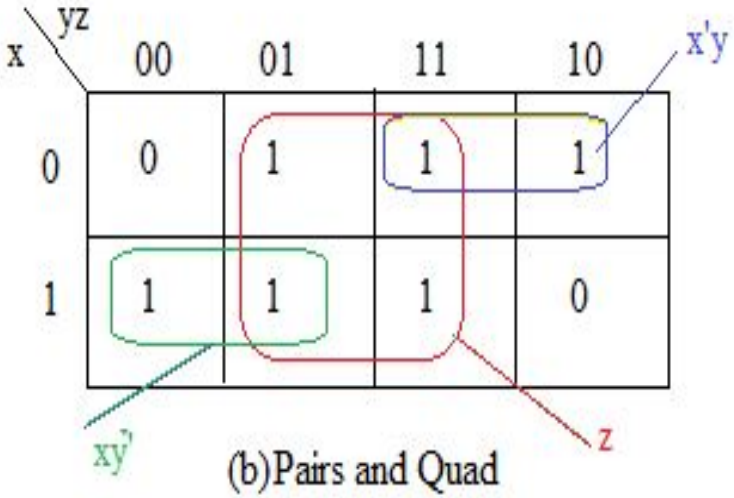
$$F = x'y'z + x'yz + x'yz' + xy'z' + xy'z + xyz$$

Solution:

The equation has six minterms. So, enter 1's at appropriate positions in the K-Map



(a) Use of Quad



(b)Pairs and Quad

Solution is: $F = z + x'y + xy'$

K-map examples (3 variable)

Simplify the K-map of Fig. 5.9.

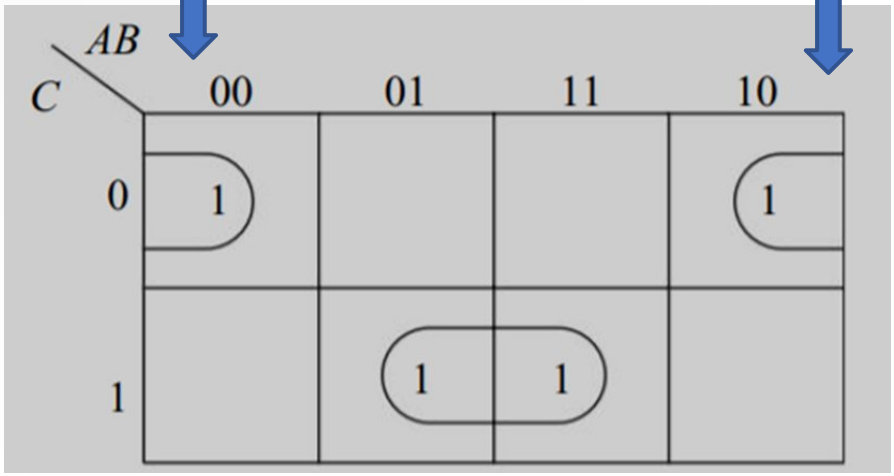
The canonical SOP form of equation can be written by inspection as

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

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

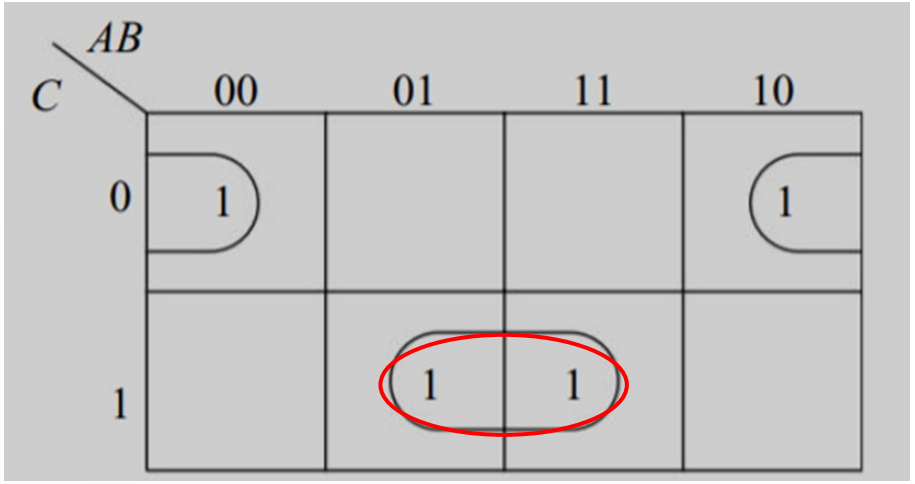
$$= \bar{B}\bar{C} + BC \text{ (Theorems 1.7 and 1.2)}$$

Adjacent cells for grouping



A	B	C	Y	Y
0	0	0	1	$\bar{A}\bar{B}\bar{C}$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

K-map examples (3 variable)

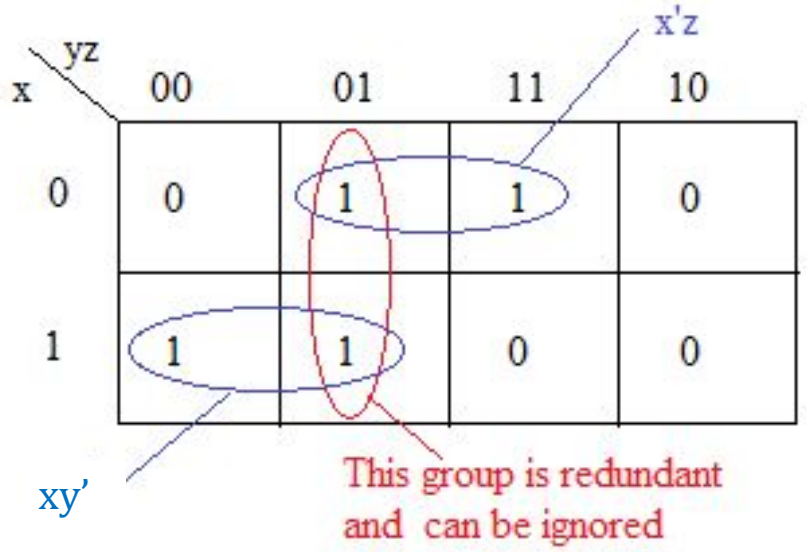


A	B	C	Y	Y
0	0	0	1	$\bar{A}\,\bar{B}\,\bar{C}$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}\,B\,C$
1	0	0	1	$A\,\bar{B}\,\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	$A\,B\,C$

$$= \bar{B}\bar{C} + BC$$

Redundant Groups

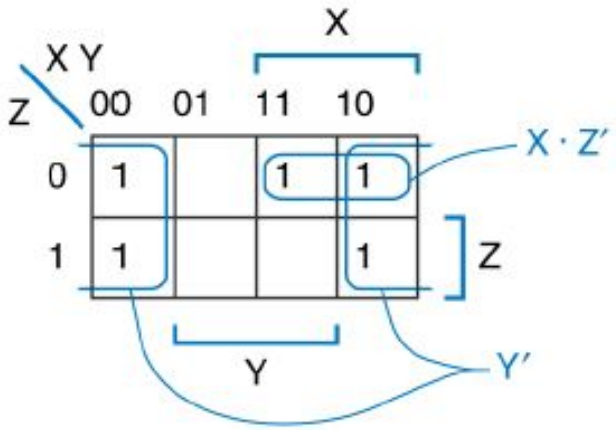
- A redundant group is one whose all the 1's have been consumed by other groups. So, there is no need to form such group. Whenever you see that all 1's of a group have been exhausted, simply ignore that group.



$$y = x'z + xy'$$

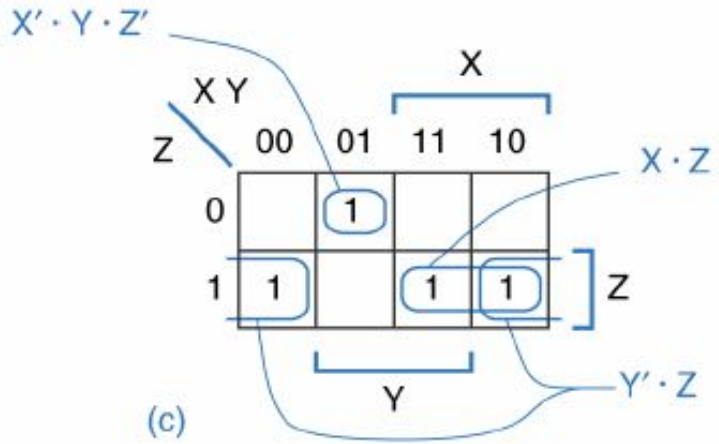
K-map examples (3 variable)

1. F = Σm(0, 1, 4, 5, 6)



F = Y' + X Z'

2. F = Σm(1, 2, 5, 7)

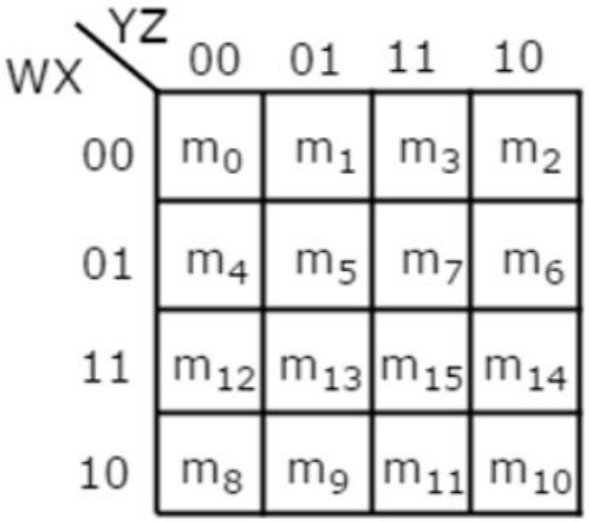


F = X' Y Z' + X Z + Y' Z

K-map examples (4 variable)

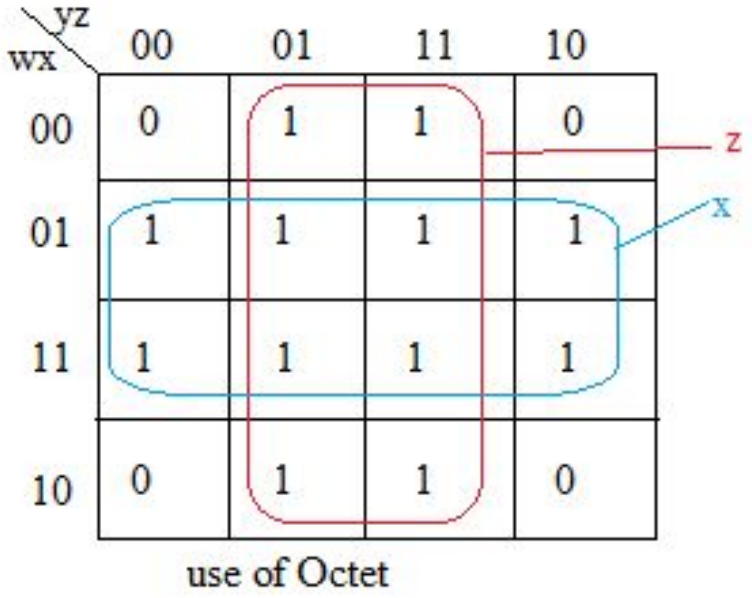
w	x	y	z	F
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15

Truth Table 4-variable function



K-map examples (4 variable)

$F(W,X,Y,Z) = \Sigma m(1,3,4,5,6,7,9,11,12,13,14,15)$



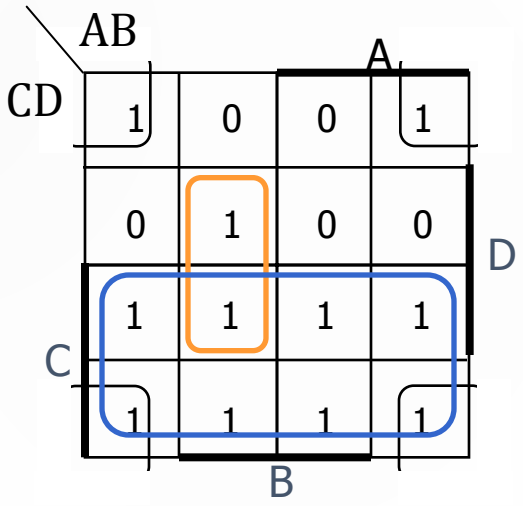
Grouping 8 adjacent binary ones

Simplified expression is $F = x + z$

K-map examples (4 variable)

$F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$

F= C + A'BD + B'D'



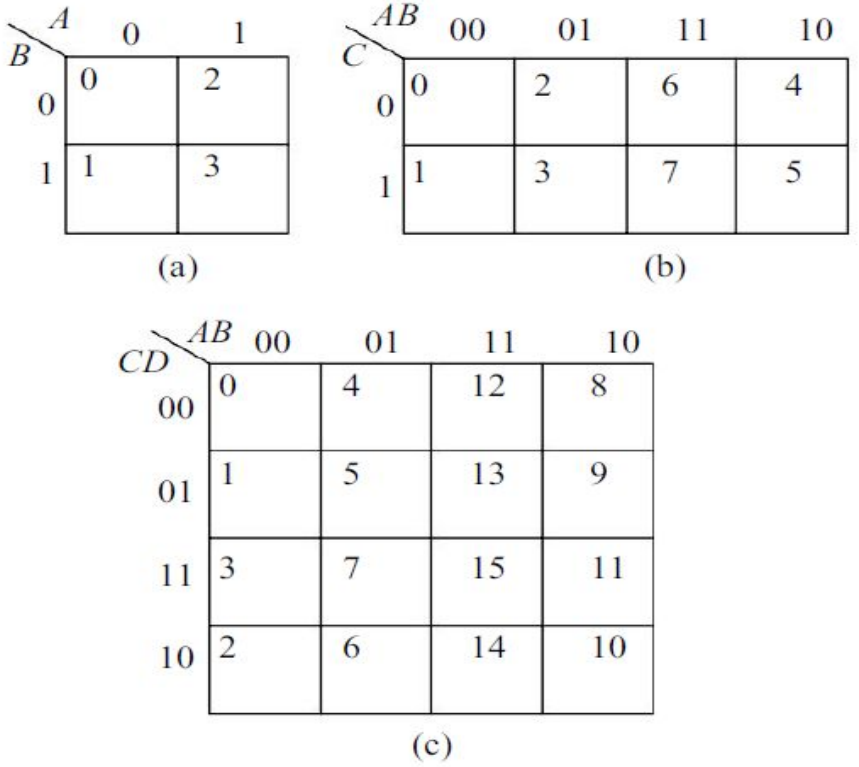
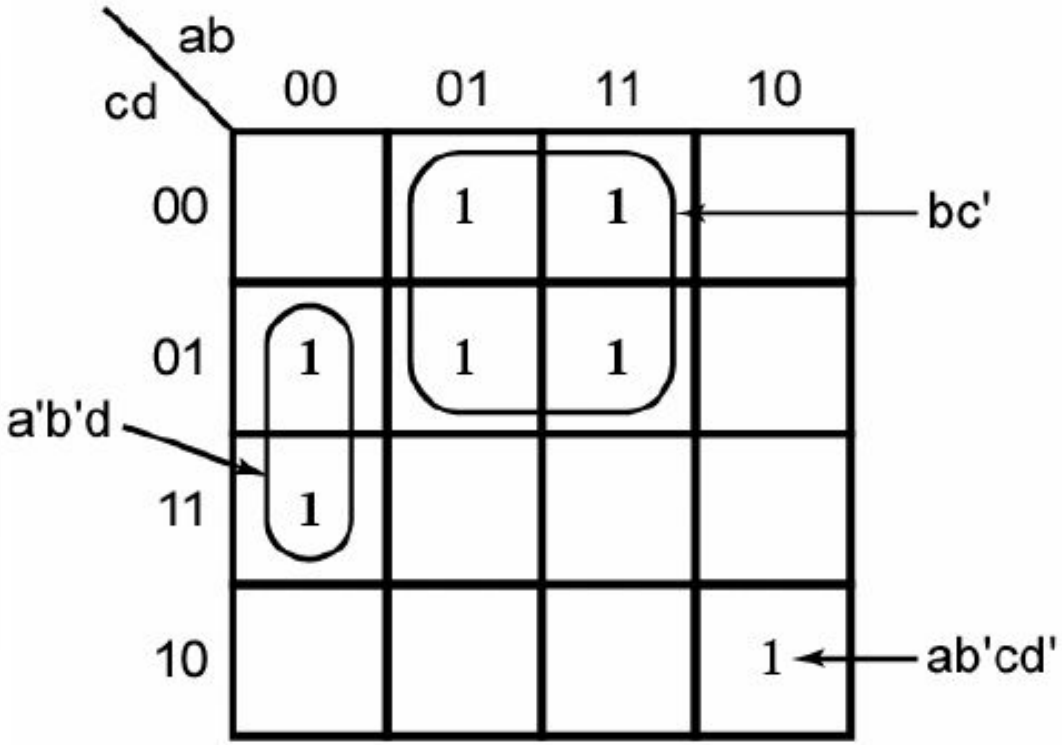


Fig. 5.5 *Karnaugh-maps (a) Two-variable (b) Three-variable (c) Four-variable*

K-map examples (4 variable)
Simplify the expression

$$f_1 = \Sigma m(1, 3, 4, 5, 10, 12, 13)$$

$$= bc' + a'b'd + ab'cd'$$



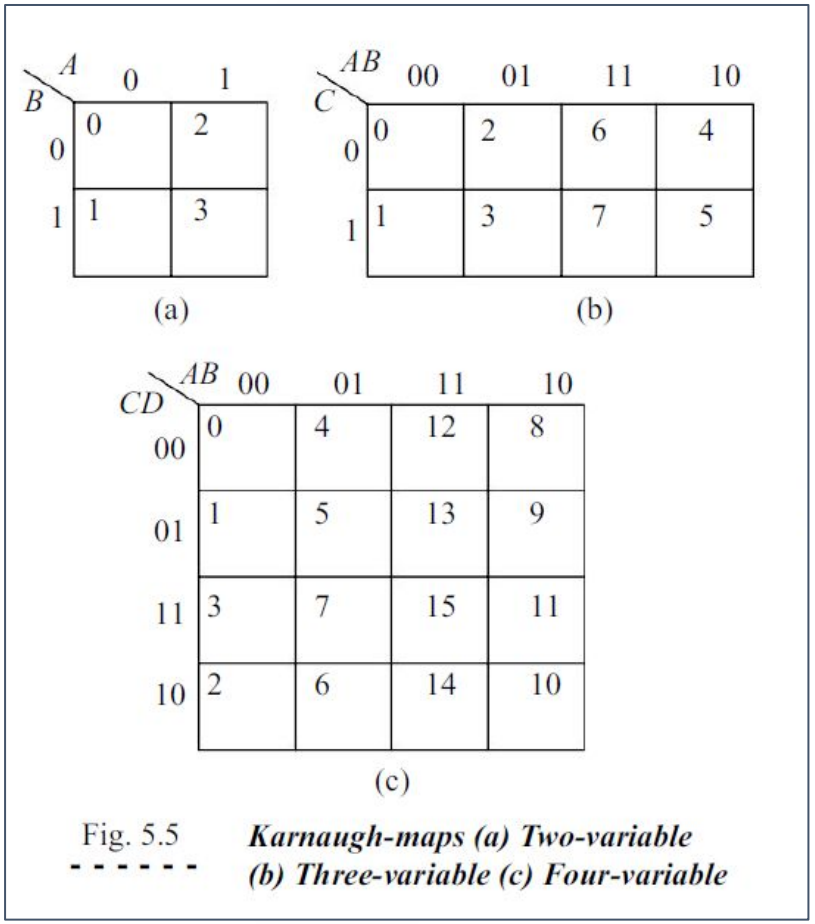


Fig. 5.5 *Karnaugh-maps (a) Two-variable (b) Three-variable (c) Four-variable*

K map is given, Find the truth table, Boolean expression in SOP and POS forms

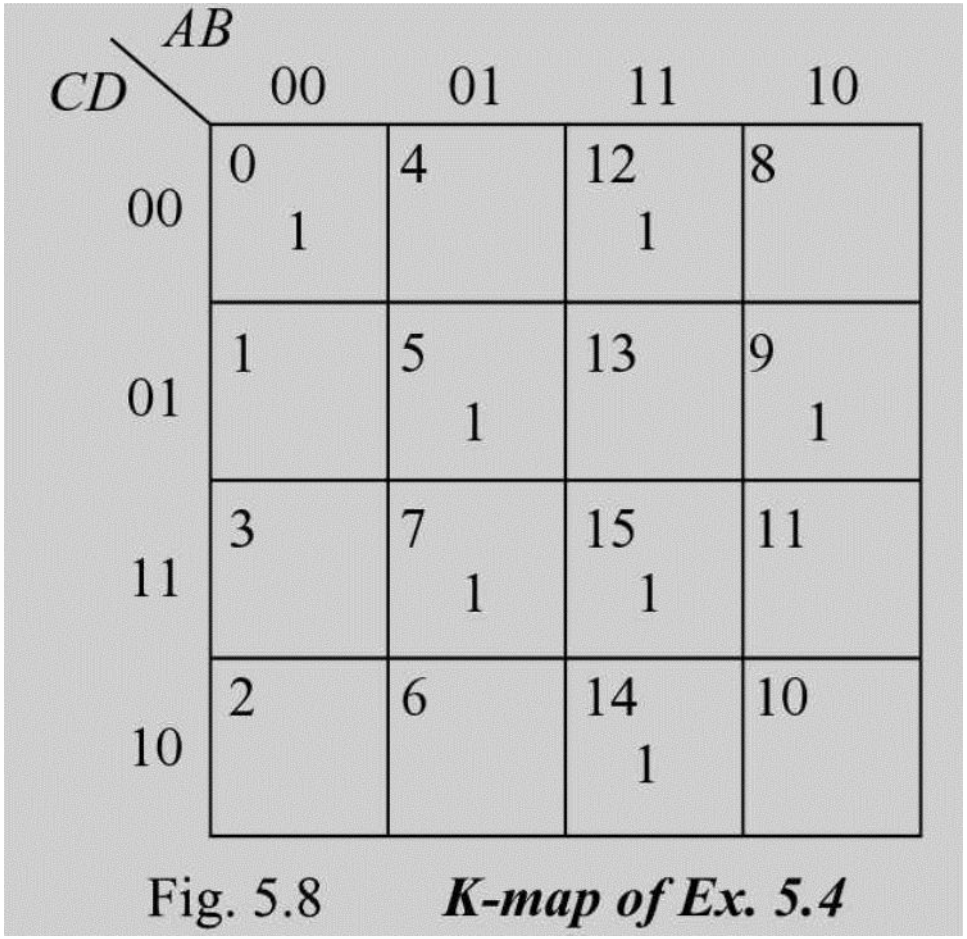


Fig. 5.8 *K-map of Ex. 5.4*

How many variables in the expression?

How many SOP terms in the expression?

How many POS terms in the expression?

How many groupings for simplification?

How many terms in the simplified expression?

Truth Table

Row No.	Inputs				Output
	A	*B*	*C*	*D*	*Y*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Logical expression in canonical SOP and POS forms

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + \bar{A}BCD + ABCD + ABC\bar{D}$$
$$= \Sigma m(0, 5, 7, 9, 12, 14, 15)$$

$$Y = (A + B + C + \bar{D})(A + B + \bar{C} + D)\,(A + B + \bar{C} + \bar{D})$$
$$(A + \bar{B} + C + D)(A + \bar{B} + \bar{C} + D)\,(\bar{A} + B + C + D)$$
$$(\bar{A} + B + \bar{C} + D)(\bar{A} + B + \bar{C} + \bar{D})(\bar{A} + \bar{B} + C + \bar{D})$$
$$= \Pi M(1, 2, 3, 4, 6, 8, 10, 11, 13)$$

Simplified Boolean Expression will have five terms

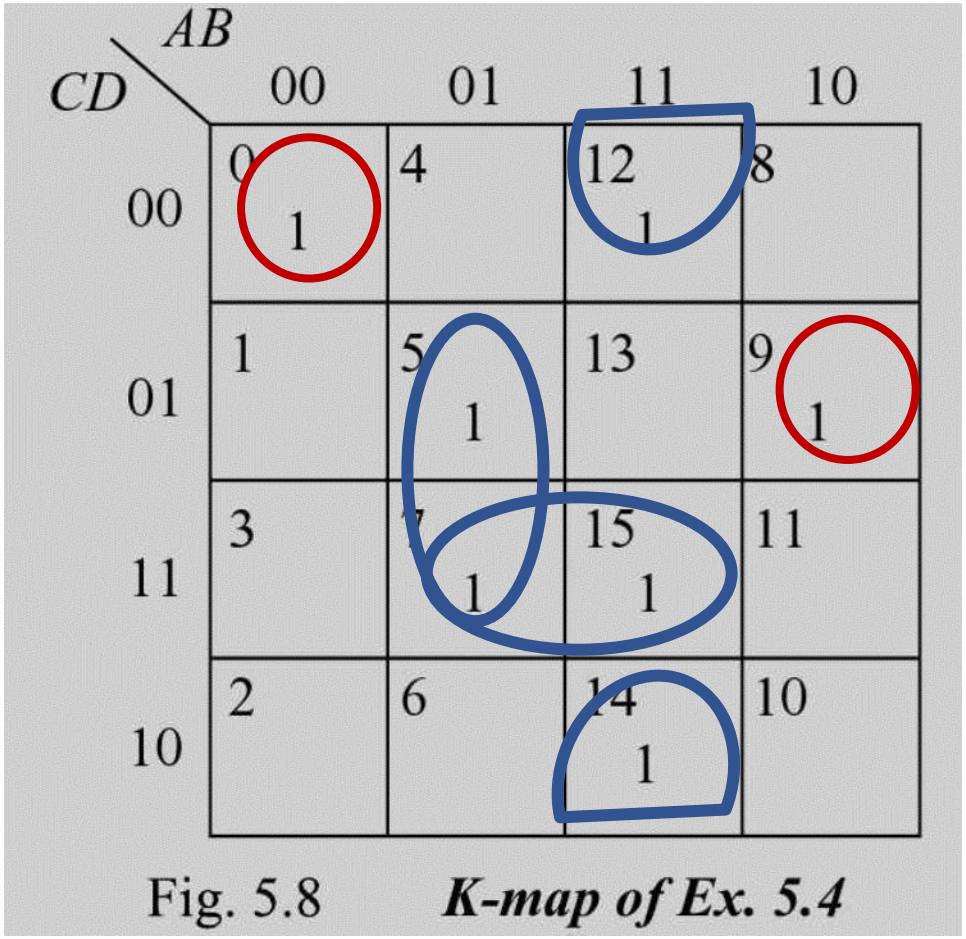


Fig. 5.8 *K-map of Ex. 5.4*

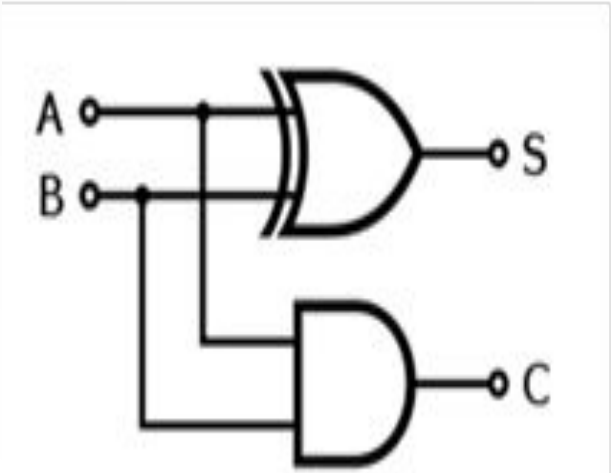
F =
A'B'C'D'
+
A'BD
+
BCD
+
ABD'
+
AB'CD'

Arithmetic Circuits: Half Adder

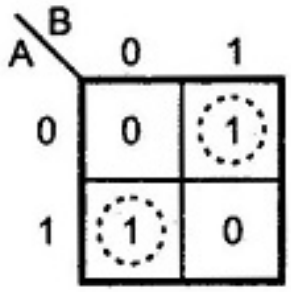
- A *half-adder is an arithmetic circuit block that can be used to add two 1 bit numbers. Such a circuit thus has two* inputs that represent the two bits to be added and two outputs, with one producing the SUM output and the other producing the CARRY.
- Possible input combinations and the corresponding outputs are as given in the truth table.
- The Boolean expressions for the SUM and CARRY outputs are given by the following equations.

Half Adder Truth Table , Kmap, Realization

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

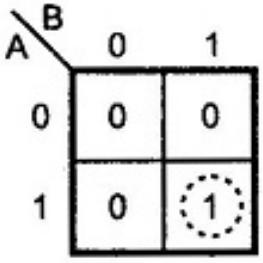


Sum = S



Sum = $A\bar{B} + \bar{A}B$
= $A \oplus B$

Carry = C



Carry = AB

Full Adder

- A *full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a* SUM and a CARRY output.
- Such a building block is needed in order to add binary numbers with a large number of bits.
- The full adder circuit overcomes the limitation of the half-adder, which can be used to add two bits only

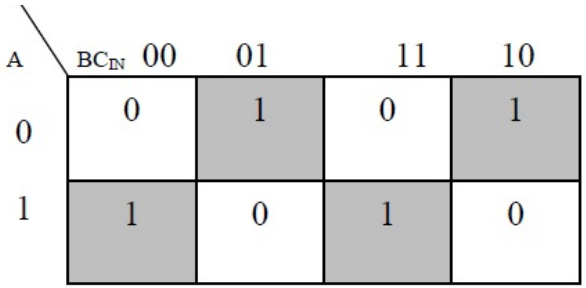
Full Adder

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \sum m(1,2,4,7)$

$C = \sum m(3,5,6,7)$

Sum = S



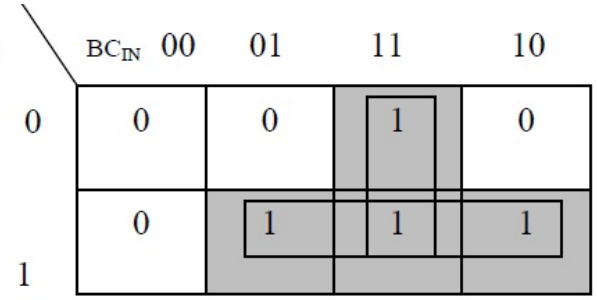
S = A'B'C + A'BC'+AB'C'+ABC

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

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

= A ⊕ B ⊕ C

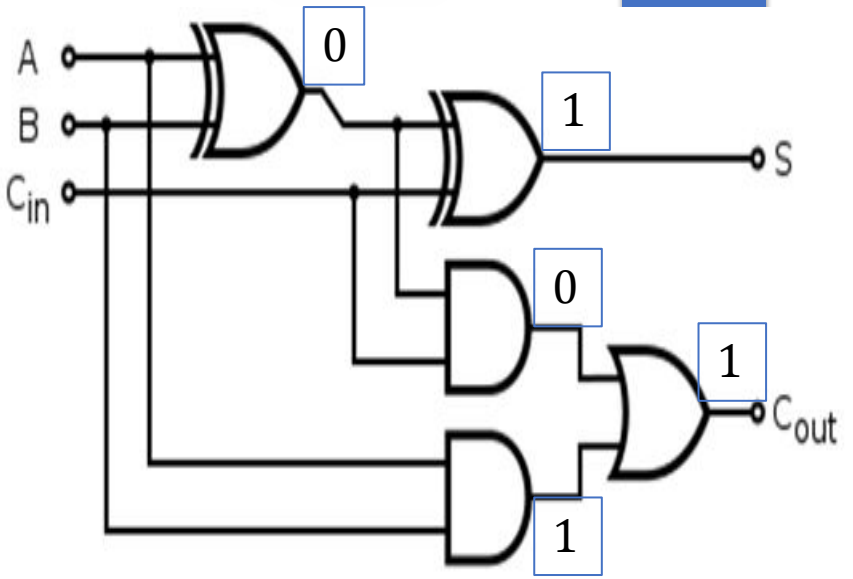
Carry = CR

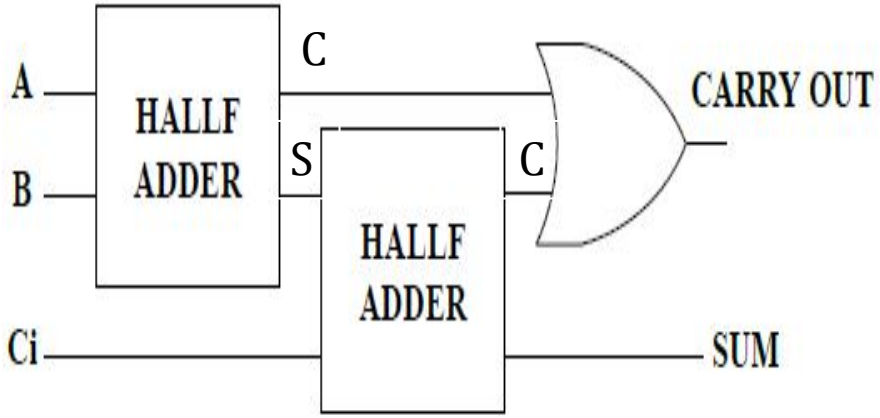


CR = AB + BC + AC

Full Adder

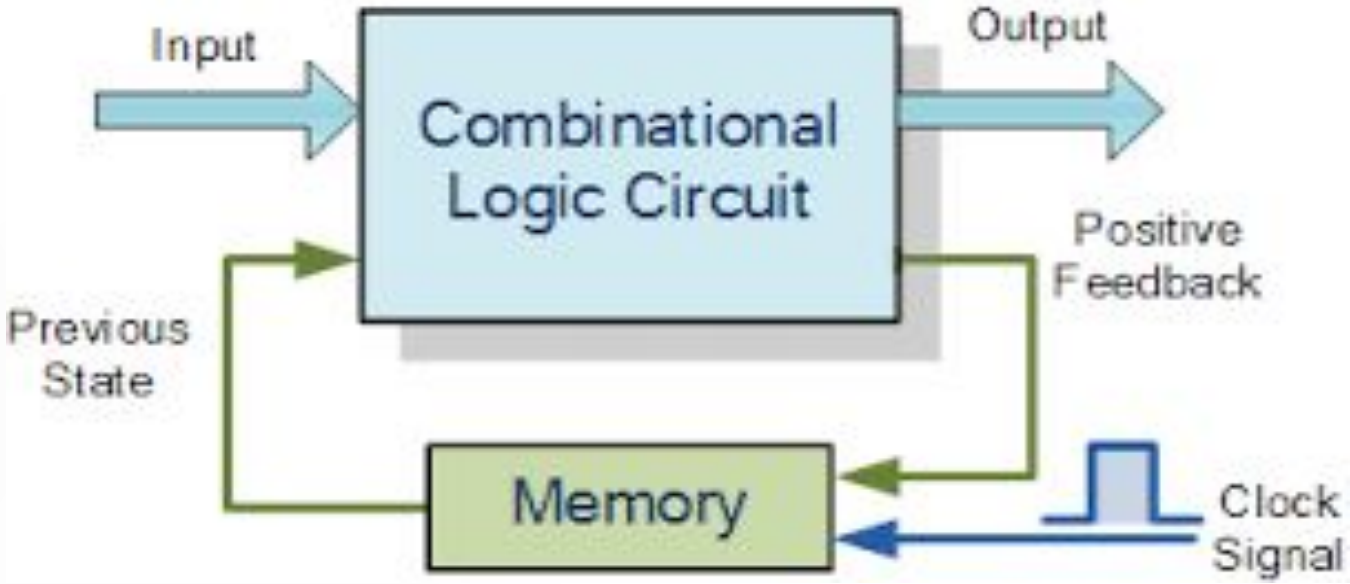
Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





Sequential Logic Circuit

- **Sequential logic** is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs, the input history.
- Sequential logic is combinational logic with memory.



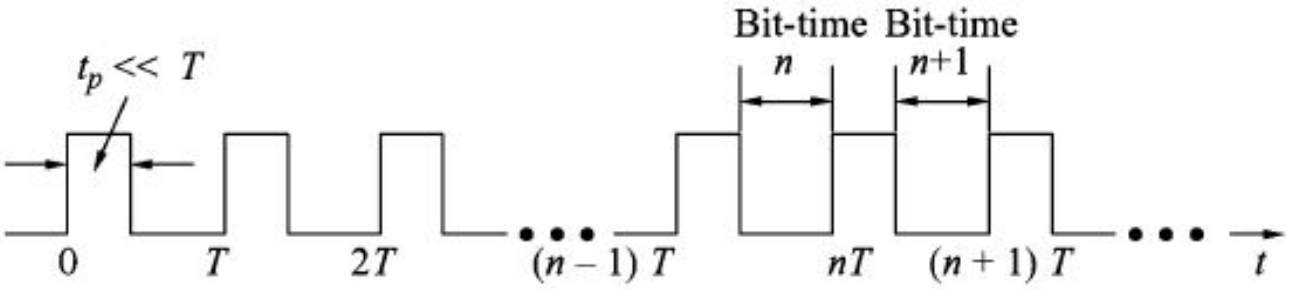
- Sequential logic circuits are classified in 2 categories

✔Synchronous

✔Asynchronous

Clock Pulse-
CK =0 or CK=1

- A Synchronization is achieved by the timing device known as system clock which generates a periodic train of clock pulses shown in figure.
- The outputs are affected with the application of clock pulse, CK.



A Train of Pulses

SR latch, SR flipflop resources

- https://youtu.be/kt8d3CYWGH4
- https://youtu.be/HZg7fNu-l24

SR Latch (1-Bit Memory Cell)

1. The outputs Q and $\bar{Q}$ are always complementary.
2. The circuit has two stable states; in one of the stable state $Q = 1$ which is referred to as the 1 state (*or set state*) whereas in the other stable state $Q = 0$ which is referred to as the 0 state (*or reset state*).
3. If the circuit is in 1 state, it continues to remain in this state and similarly if it is in 0 state, it continues to remain in this state. This property of the circuit is referred to as *memory*, i.e. it can store 1-bit of digital information.

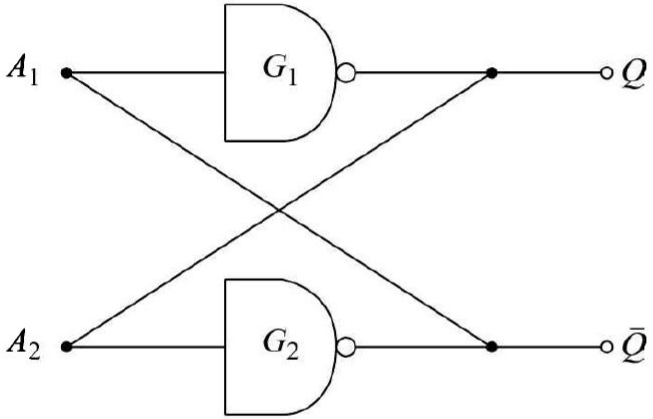


Fig. 7.3 ***Cross-coupled Inverters as a Memory Element***

Basic SR Flip-flop

- The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown
- There is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the Set, S and the other called the Reset, R with two corresponding outputs Q and its inverse or complement Q (not-Q) as shown below.

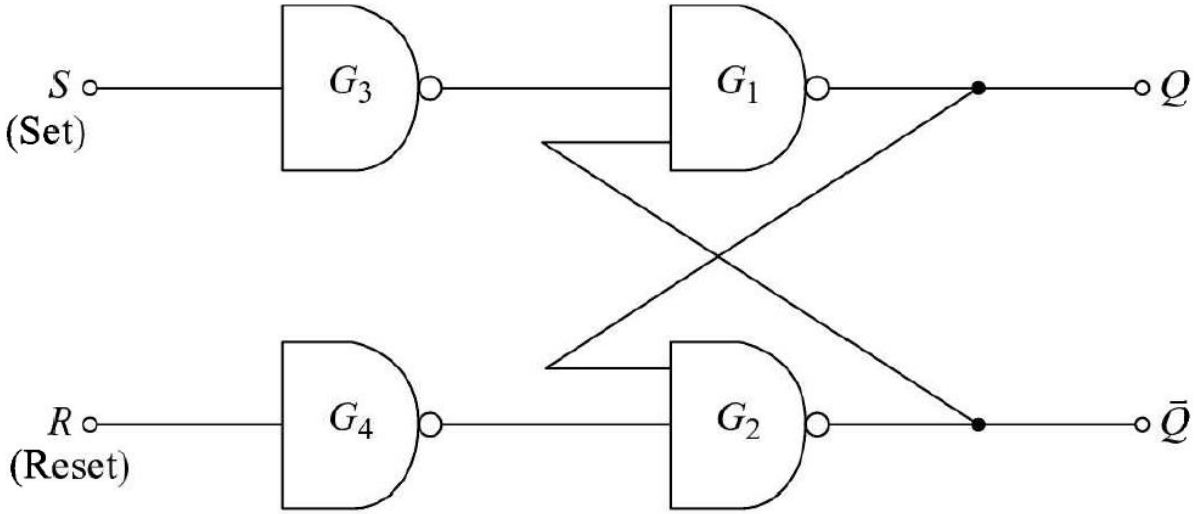


Fig. 7.4 ***The Memory Cell with Provision for Entering Data***

Clocked SR Flip-Flop

In this circuit, if a clock pulse is present ($CK = 1$), its operation is exactly the same as that of Fig. 7.4. On the other hand, when the clock pulse is not present ($CK = 0$), the gates G_3 and G_4 are inhibited, i.e. their outputs are 1 irrespective of the values of S or R. In other words, the circuit responds to the inputs S and R only when the clock is present.

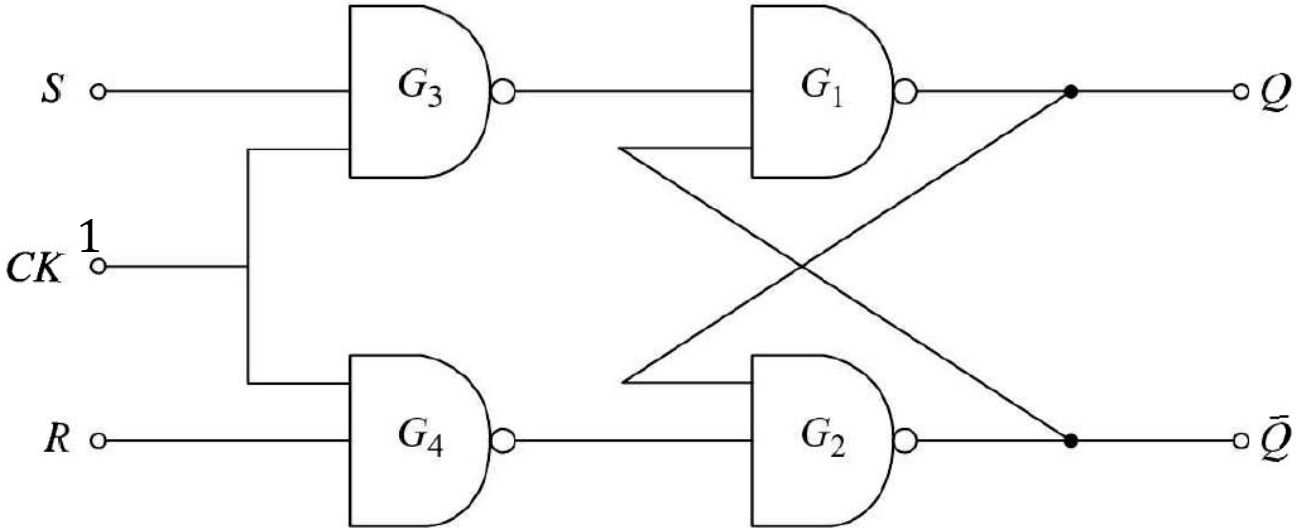


Fig. 7.5 ***A Clocked S–R*** FLIP-FLOP

SR Flip-Flop

Block diagram

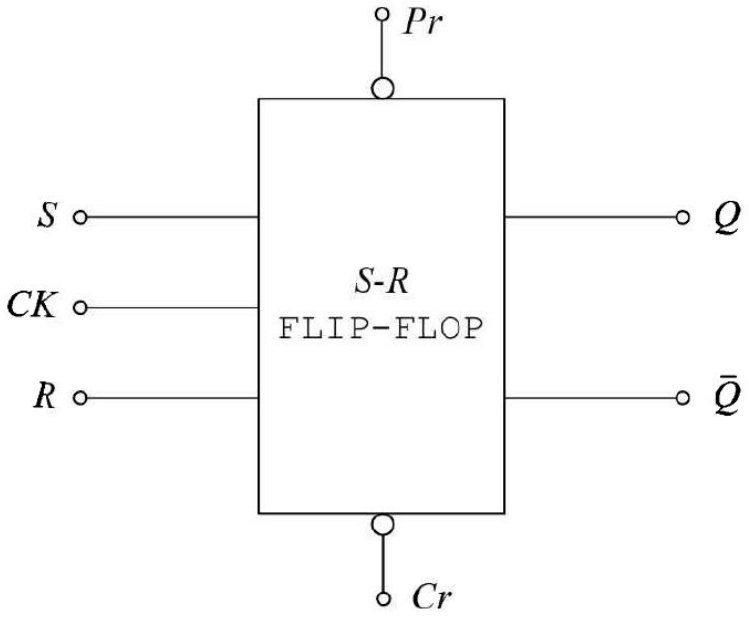


Table 7.1 ***Truth Table of S–R*** FLIP-FLOP

Inputs		Output
S_n	R_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	?

Race Around

SR Flip flop Operation

S.N.	Condition	Operation
1	**S = R = 0 , CK=1**	If S = R = 0 then output of NAND gates 3 and 4 are forced to become 1. Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs.
2	**S = 0, R = 1, CK = 1**	Since S = 0, output of NAND-3 i.e. R' = 1 and CK = 1 the output of NAND-4 i.e. S' = 0. Hence Q_{n+1} = 0 and Q_{n+1} bar = 1. This is reset condition.
3	**S = 1, R = 0, CK = 1**	Output of NAND-3 i.e. R' = 0 and output of NAND-4 i.e. S' = 1. Hence output of S-R NAND latch is Q_{n+1} = 1 and Q_{n+1} bar = 0. This is the set condition.
4	**S = 1, R = 1, CK = 1**	As S = 1, R = 1 and E = 1, the output of NAND gates 3 and 4 both are 0 i.e. S' = R' = 0. Hence the **Race** condition will occur in the basic NAND latch.

J-K Flip-Flop

Uncertainty in the state of SR flip-flop when S=R=1 can be eliminated by converting it into JK flip-flop

$$S = J \cdot \bar{Q}$$

$$R = K \cdot Q$$

Asynchronous inputs- Active low
Pr= Preset
Cr= Clear

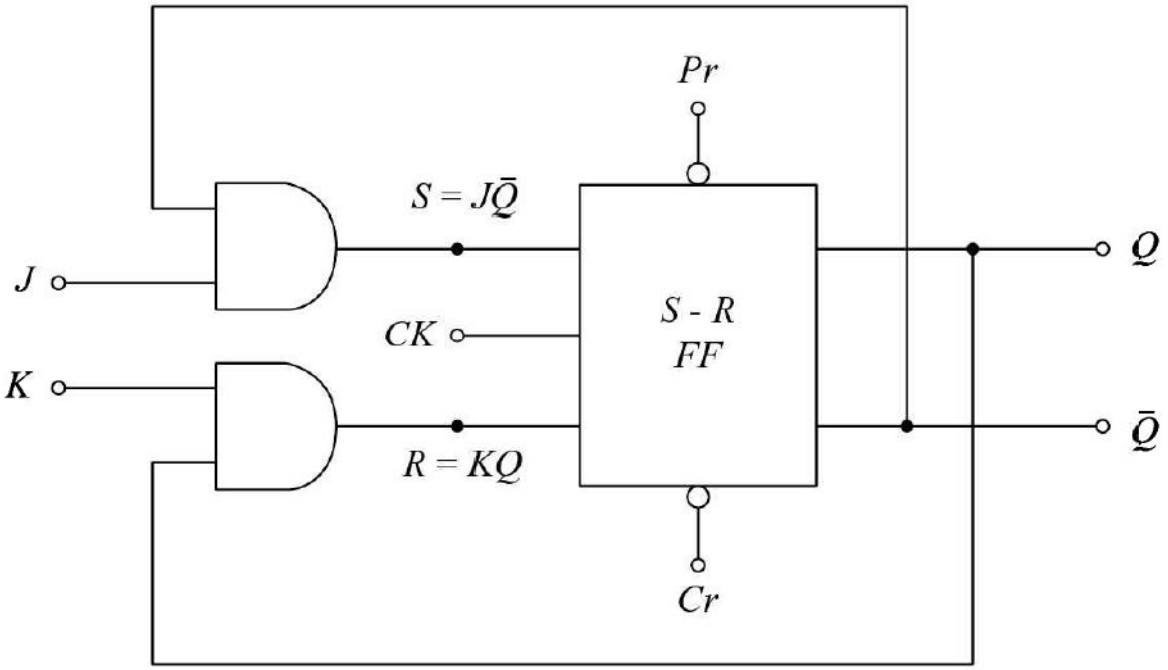


Fig. 7.8 *An S–R* FLIP-FLOP *Converted into J–K* FLIP-FLOP

Truth Table of JK flip flop

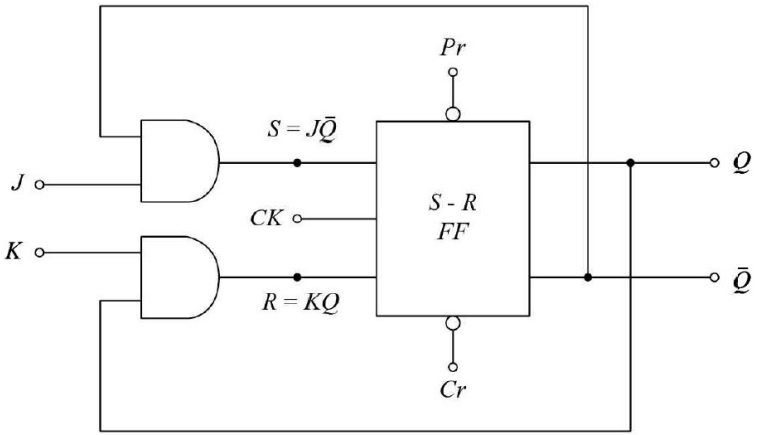


Fig. 7.8 *An S–R* FLIP-FLOP *Converted into J–K* FLIP-FLOP

Data inputs		Outputs		Inputs to S–R FF		Output
J_n	K_n	Q_n	$\bar{Q}_n$	S_n	R_n	Q_{n+1}
0	0	0	1	0	0	$\begin{matrix}0\\1\end{matrix} = Q_n$
0	0	1	0	0	0	
1	0	0	1	1	0	$\begin{matrix}1\\1\end{matrix} = 1$
1	0	1	0	0	0	
0	1	0	1	0	0	$\begin{matrix}0\\0\end{matrix} = 0$
0	1	1	0	0	1	
1	1	0	1	1	0	$\begin{matrix}1\\0\end{matrix} = \bar{Q}_n$
1	1	1	0	0	1	

Truth Table of J-K Flip-Flop

Fig: Block Diagram

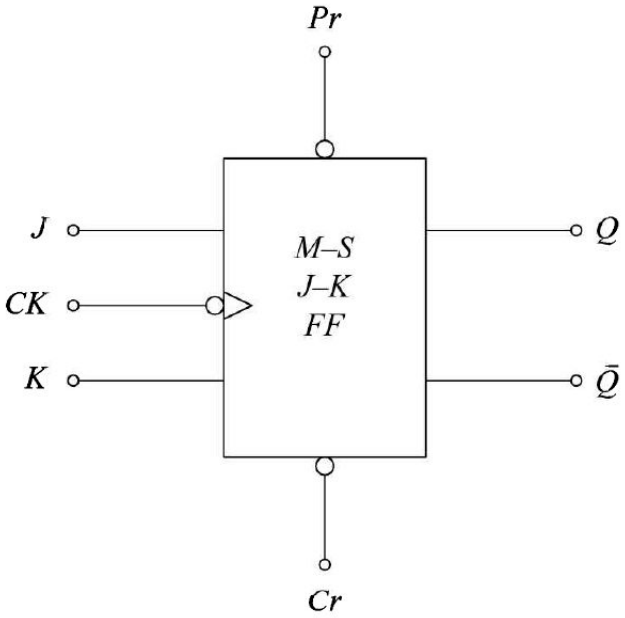


Table 7.3b ***Truth Table of J–K*** FLIP-FLOP

Inputs		Output
J_n	K_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$

D Flip-Flop

- If we use only middle two rows of SR or JK flip-flop, We obtain D Flip-flop.

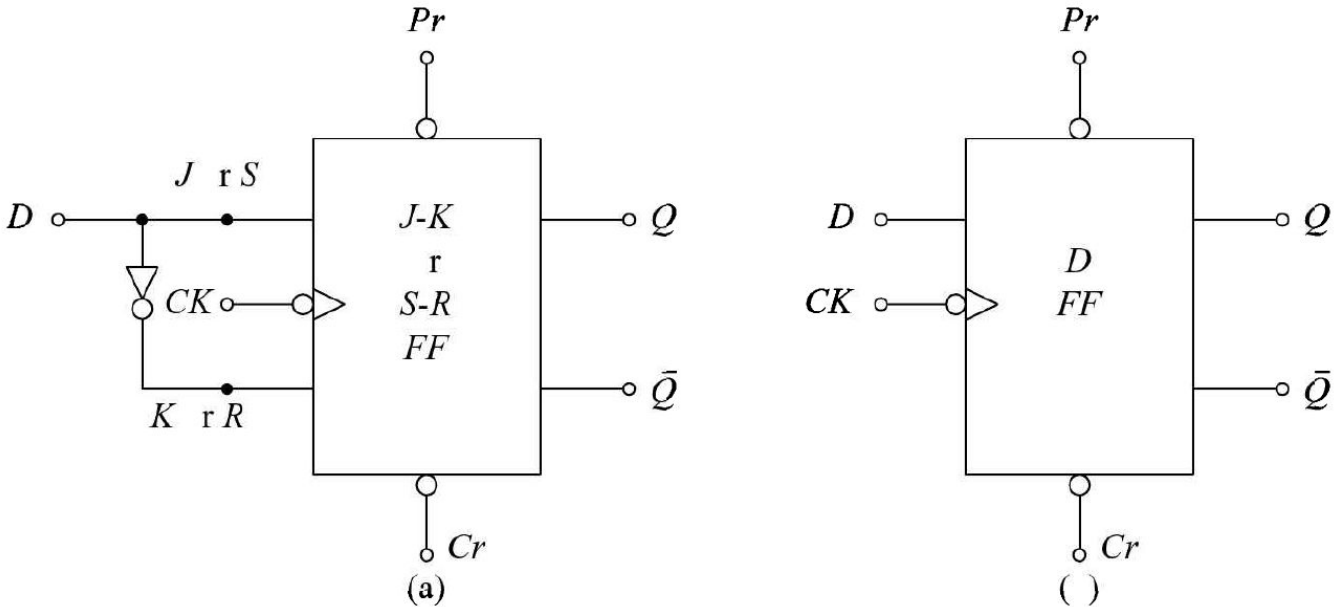


Table 7.4 ***Truth Table of a D-type*** `FLIP-FLOP`

Input D_n	Output Q_{n+1}
0	0
1	1

Operation of D Flip-Flop

S. No.	Condition	Operation
1	**E = 0**	Latch is disabled. Hence no change in output.
2	**E = 1 and D = 0**	If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state, the next state is Q_{n+1} = 0 and Q_{n+1} bar = 1. This is the reset condition.
3	**E = 1 and D = 1**	If E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and Q_{n+1} = 1 and Q_{n+1} bar = 0 irrespective of the present state.

T Flip-Flop

- In JK flip-flop, if J=K, the resulting Flip-flop is called as T Flip-flop.

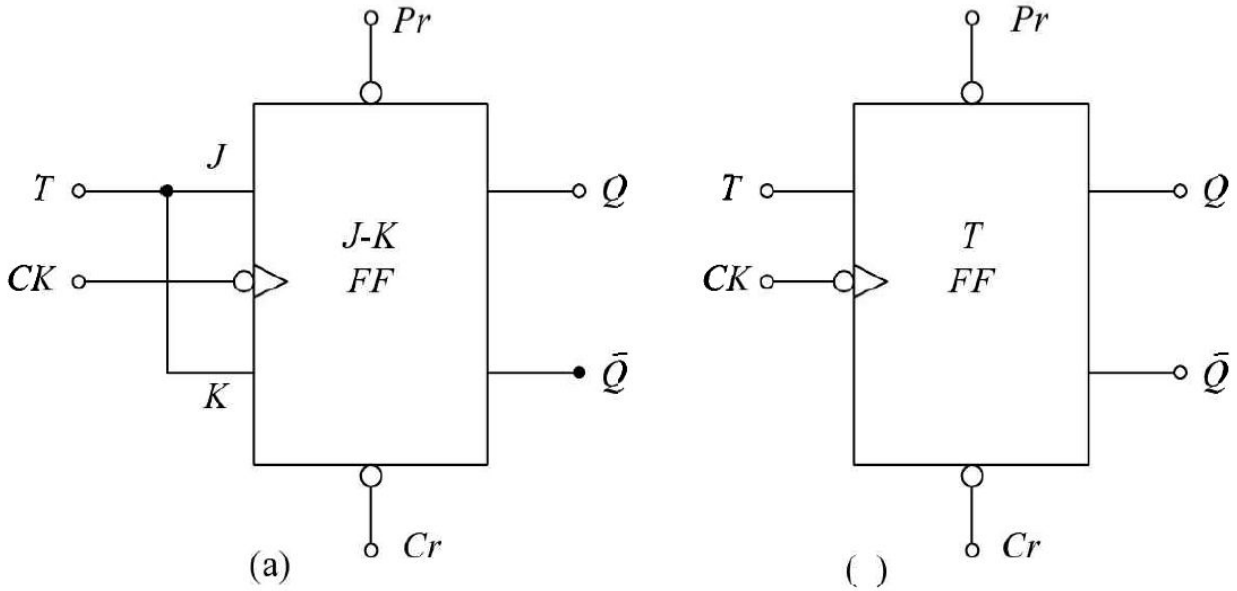


Fig. 7.15 ***(a) A J–K*** FLIP-FLOP ***Converted into a T-type*** FLIP-FLOP ***(b) its Logic Symbol***

Table 7.5 ***Truth Table of T-type*** FLIP-FLOP

Input T_n	Output Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

Operation of T Flip-Flop

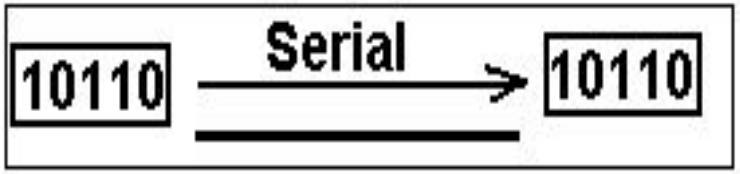
S.N.	Condition	Operation
1	**T = 0, J = K = 0**	The output Q and Q bar won't change
2	**T = 1, J = K = 1**	Output will toggle corresponding to every leading edge of clock signal.

Shift Register

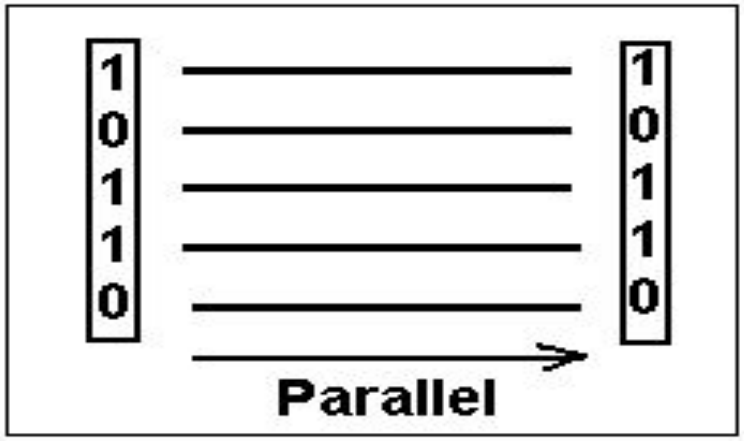
- Register is a digital circuit with two basic functions:
  1. Data storage
  2. Data Movement
- A register can consist of one or more flip flops used to store and shift data
- Shift Registers are an important Flip-Flop configuration with a wide range of applications, including:
  - Computer and Data Communications
  - Serial and Parallel Communications
  - Multi-bit number storage
  - Sequencing
  - Basic arithmetic such as scaling (a serial shift to the left or right will change the value of a binary number a power of 2)
  - Logical operations

Parallel versus Serial Communication

- Serial communications: provides a binary number as a sequence of binary digits, one after another, through one data line.

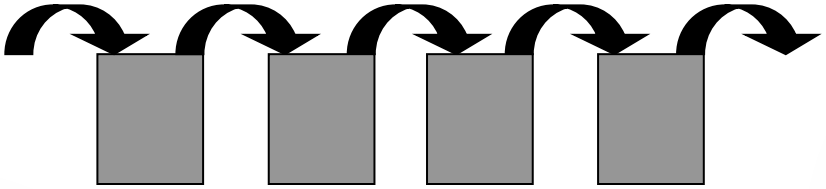


- Parallel communications: provides a binary number as binary digits through multiple data lines at the same time.

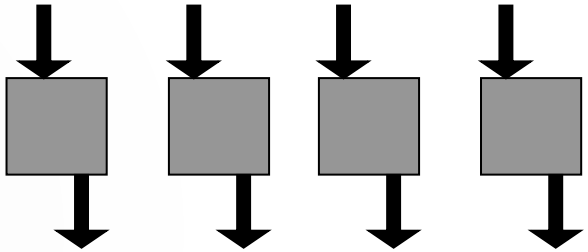


Application of Flip flop- Shift Registers

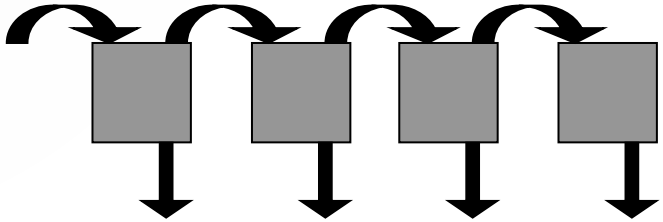
- Shift Registers are devices that store and move data bits in serial fashion (to the left or the right),



- In parallel fashion



- A combination of serial and parallel fashion

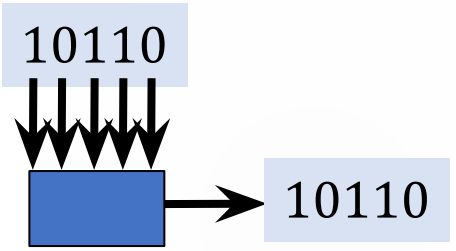


Data Transfer Methods/ Modes

1. SISO: Serial In, Serial Out

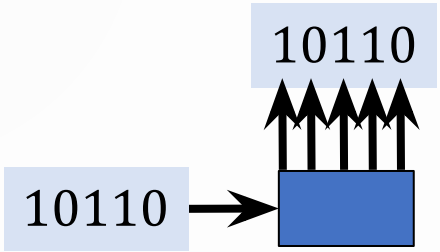
2n-1 clock cycle

3. PISO: Parallel In, Serial Out



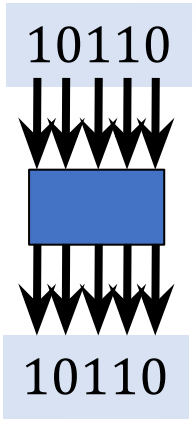
n clock cycle

2. SIPO: Serial In, Parallel Out



n clock cycle

4. PIPO: Parallel In, Parallel Out

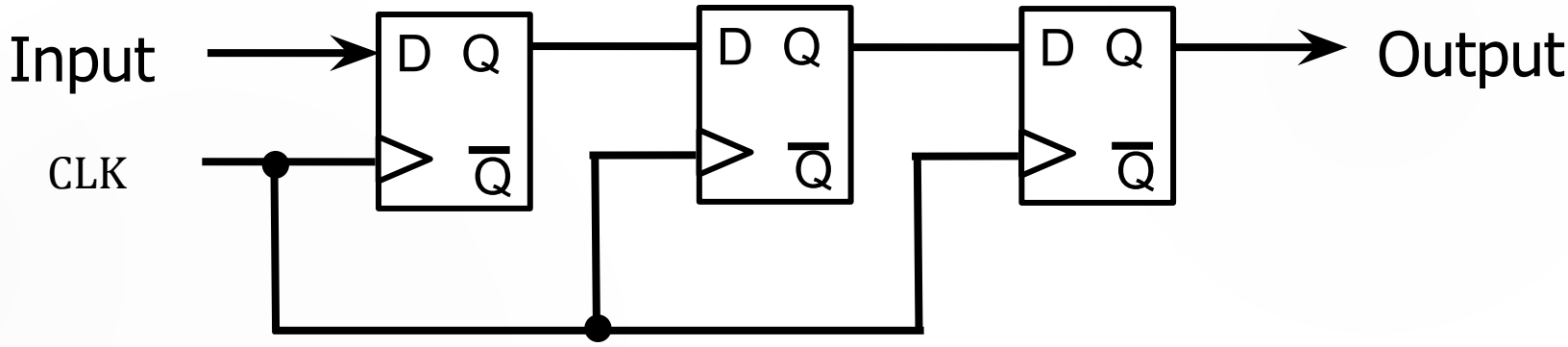


1 clock cycle

How many clock edges are required for each operation?

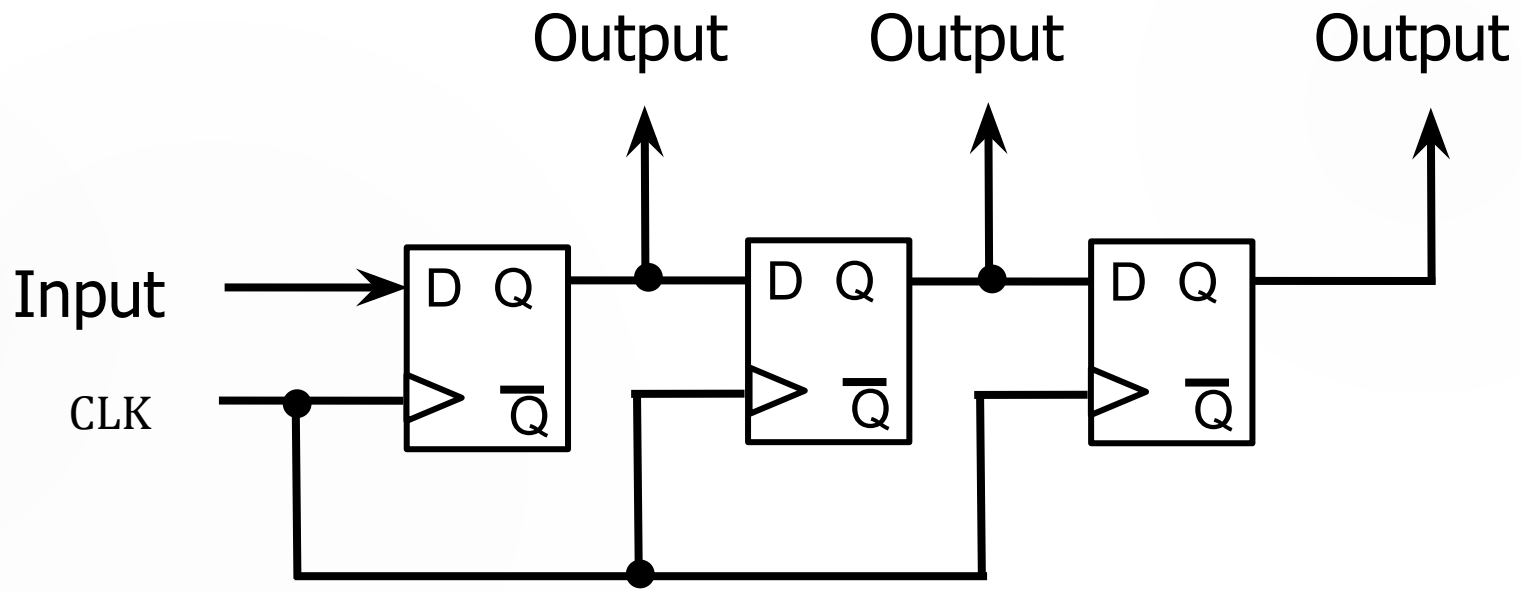
SISO Flip-Flop Shift Register

- A **Serial In Serial Out** shift register has a single input and a single output



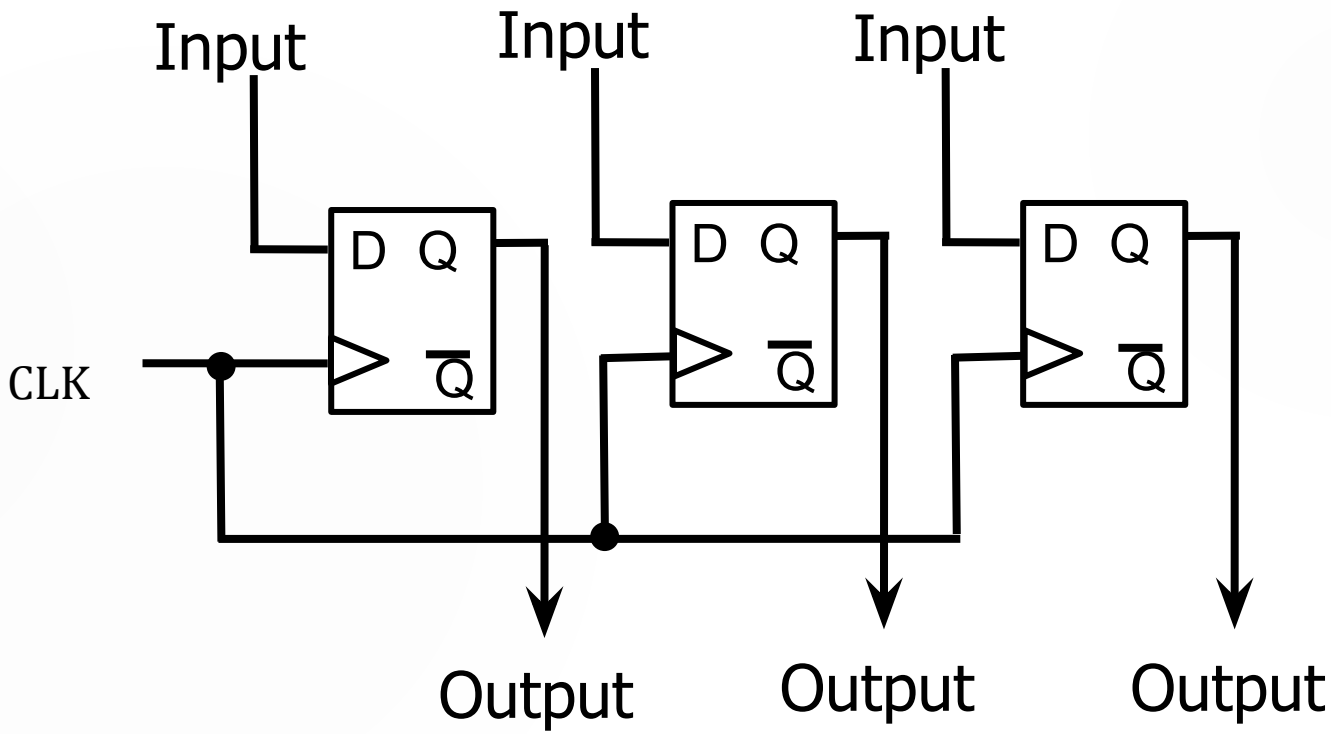
SIPO Flip-Flop Shift Register

- **Serial In Parallel Out** shift register has a single input and access to all outputs



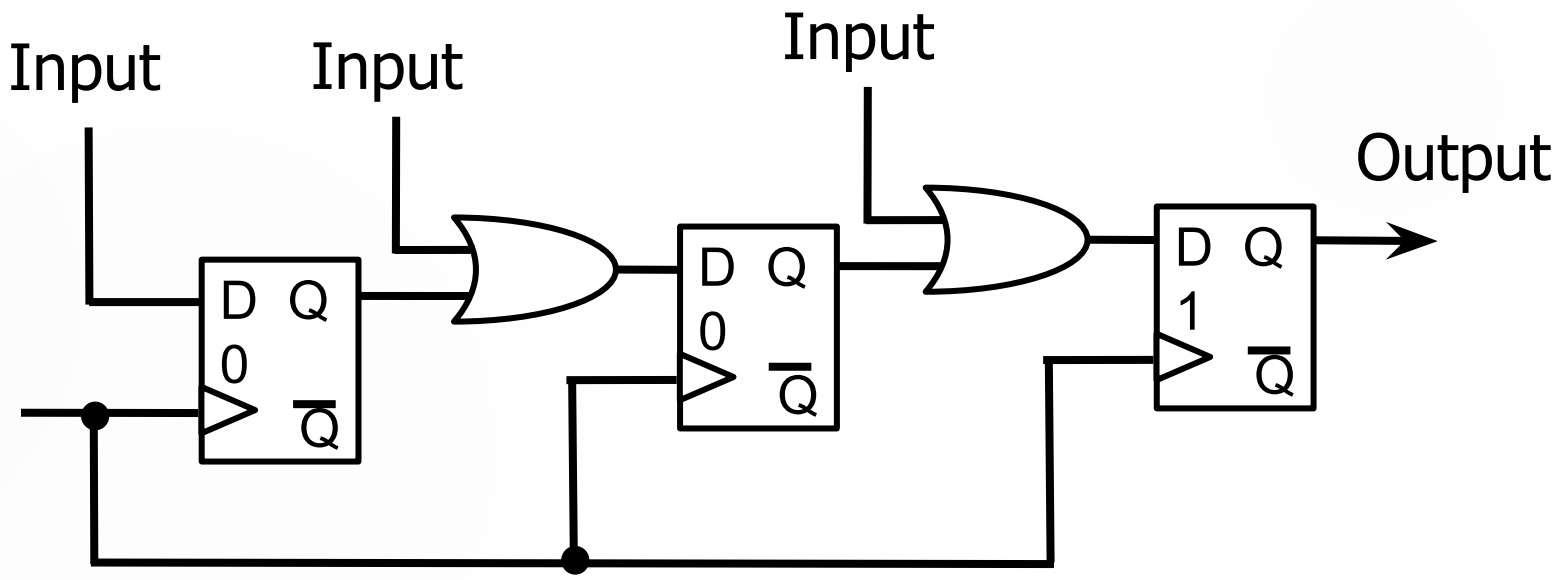
PIPO Flip-Flop Shift Register

- **Parallel In Parallel Out** register has the simplest configuration. It represents a memory device.



PISO Flip-Flop Shift Register

- **Parallel In Serial Out** shift register requires additional gates, and the parallel input must revert to logic low.



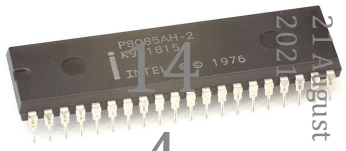
Universal Shift Registers

- Universal Shift Registers can be configured to operate in a variety of modes. For instance, they can be configured to have either Serial or Parallel Input/Output.

- Mode of Operations
  - SISO
  - SIPO
  - PISO
  - PIPO

Contributions of IC technology

- Timers
- Memories
- Microprocessors and micro-controllers
- Embedded Systems etc.



Introduction to Microprocessor

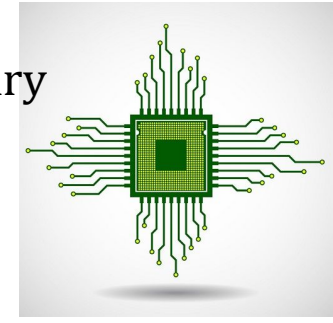
A Microprocessor is a computer processor which incorporates the functions of a computer's Central Processing Unit (CPU) on a single Integrated Circuit (IC), or at most a few integrated circuits.

The microprocessor is a
- multipurpose,
- clock driven,
- register based,
- digital-integrated circuit,

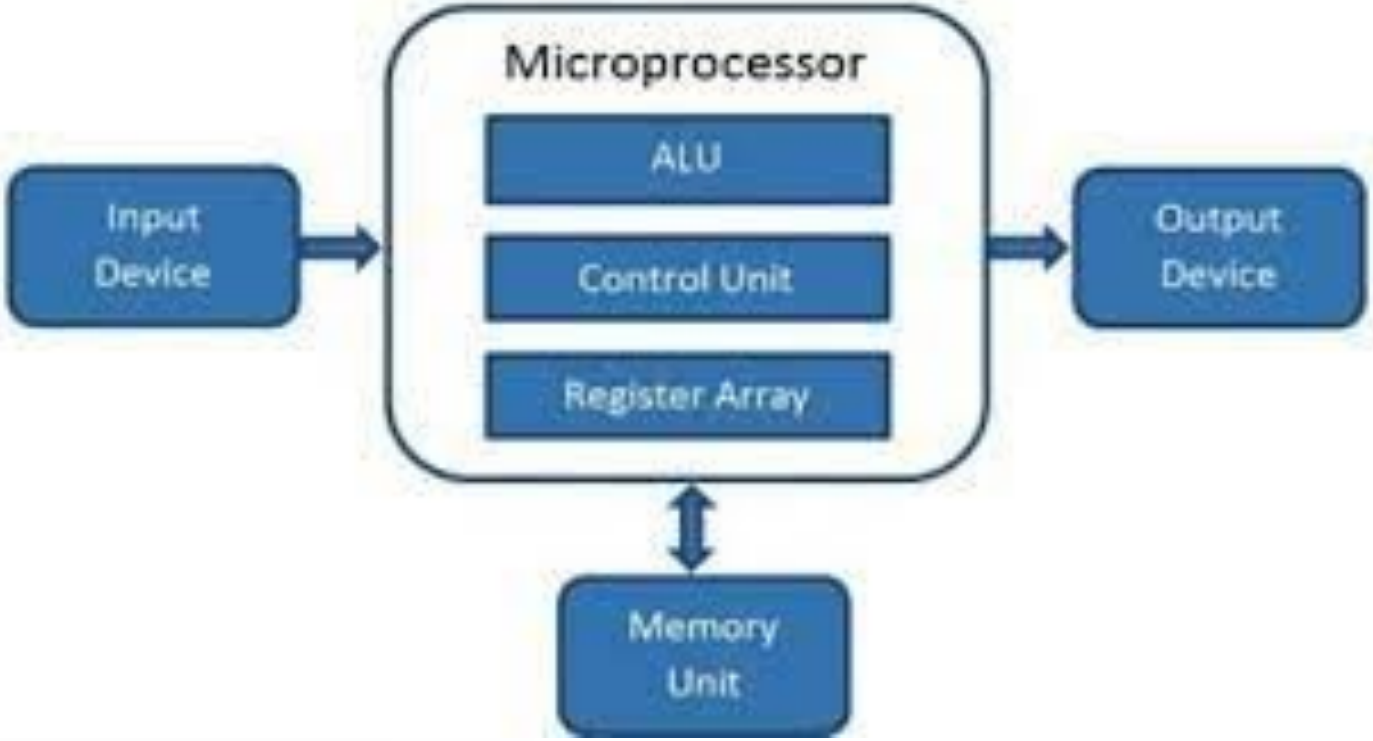
which accepts binary data (0,1) as input, processes it according to instructions stored in its memory, and provides results as output.

Microprocessors contain both combinational logic and sequential digital logic.

Microprocessors operate on numbers and symbols represented in the binary numeral system.



Building Blocks of Processor Systems



CPU

- CPU is fabricated as a very large scale integrated circuit (VLSI) and has these parts –
- Instruction register – It holds the instruction to be executed.
- Decoder – It decodes (converts to machine level language) the instruction and sends to the ALU (Arithmetic Logic Unit).
- ALU – It has necessary circuits to perform arithmetic, logical, memory, register and program sequencing operations.
- Register – It holds intermediate results obtained during program processing. Registers are used for holding such results rather than RAM because accessing registers is almost 10 times faster than accessing RAM.

Memory

- Microprocessor has two types of memory

- **RAM –** Random Access Memory is volatile memory that gets erased when power is switched off. All data and instructions are stored in RAM.

- ROM – Read Only Memory is non-volatile memory whose data remains intact even after power is switched off. Microprocessor can read from it any time it wants but cannot write to it. It is preprogrammed with most essential data like booting sequence by the manufacturer.

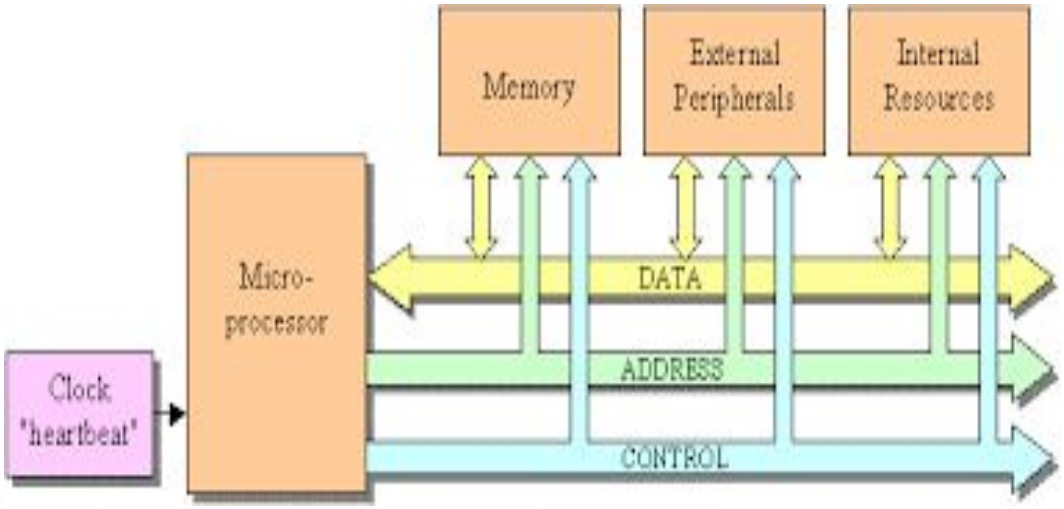
IO Interfacing

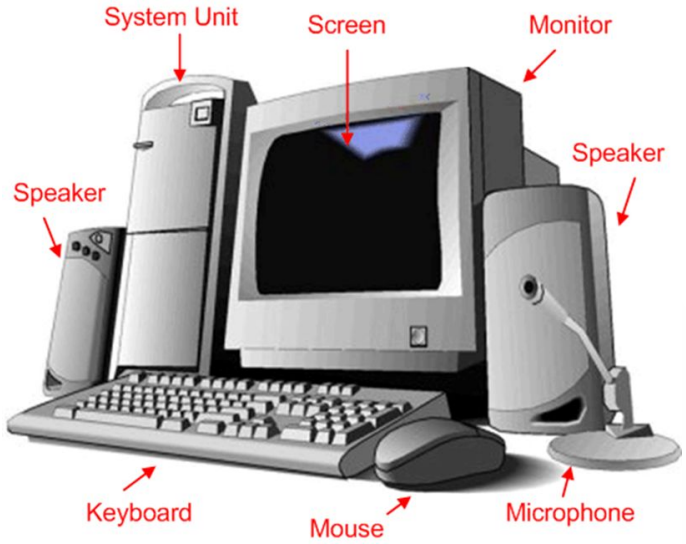
- There are various communication devices like the keyboard, mouse, printer, etc.

- So, we need to interface the keyboard and other devices with the microprocessor by using latches and buffers.

- This type of interfacing is known as Input-Output interfacing.

DIAGRAM OF A COMPUTER SYSTEM



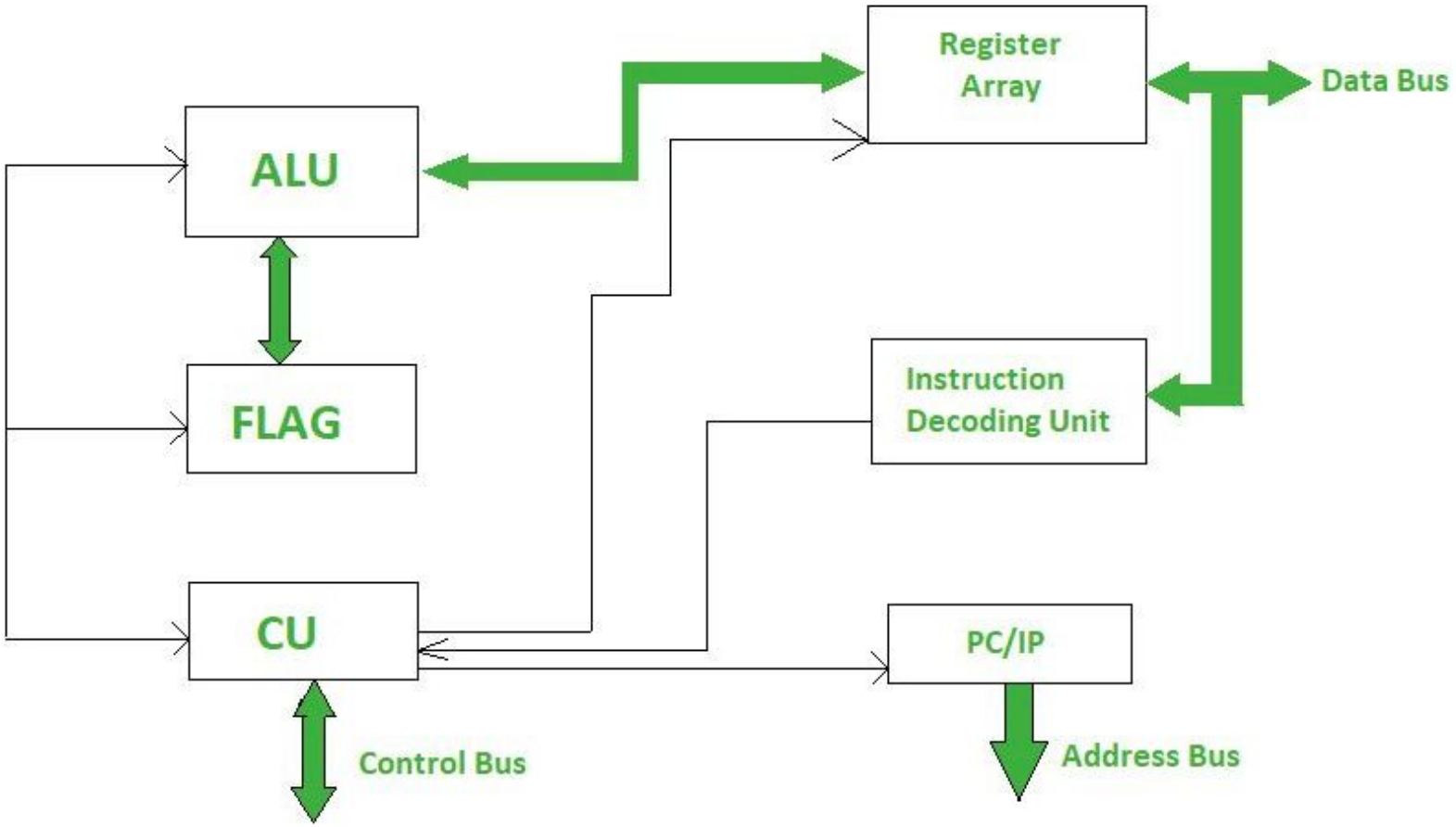


- **Data Bus** − Lines that carry data to and from memory are called data bus. It is a bidirectional bus with width equal to word length of the microprocessor.

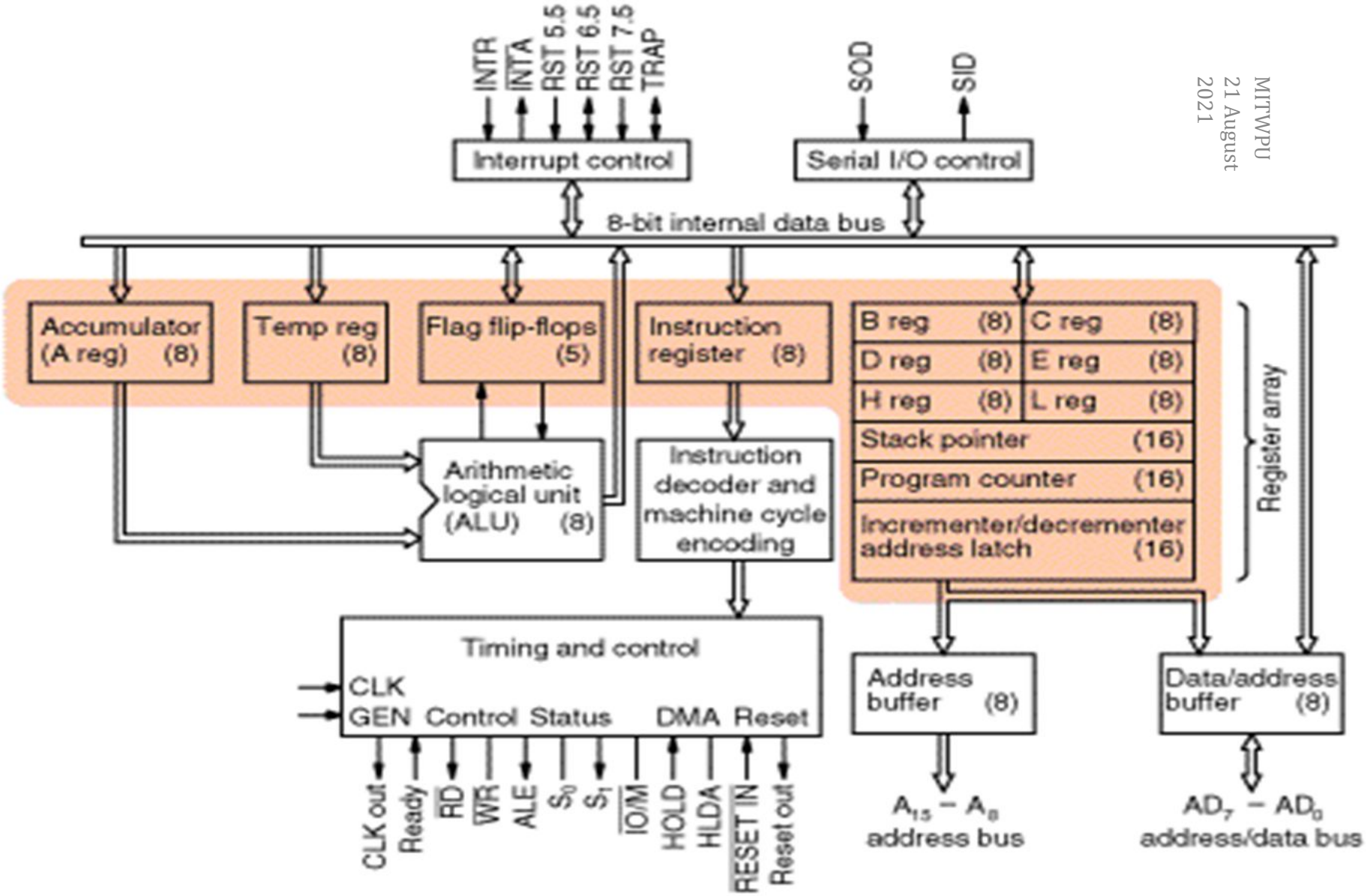
- **Address Bus** − It is a unidirectional responsible for carrying address of a memory location or I/O port from CPU to memory or I/O port.

- **Control Bus** − Lines that carry control signals like **clock signals, interrupt signal** or **ready signal** are called control bus. They are bidirectional. Signal that denotes that a device is ready for processing is called **ready signal**. Signal that indicates to a device to interrupt its process is called an **interrupt signal**.

A typical Microprocessor structure



Architecture of Intel 8085 Microprocessor



Intel 8085 Microprocessor

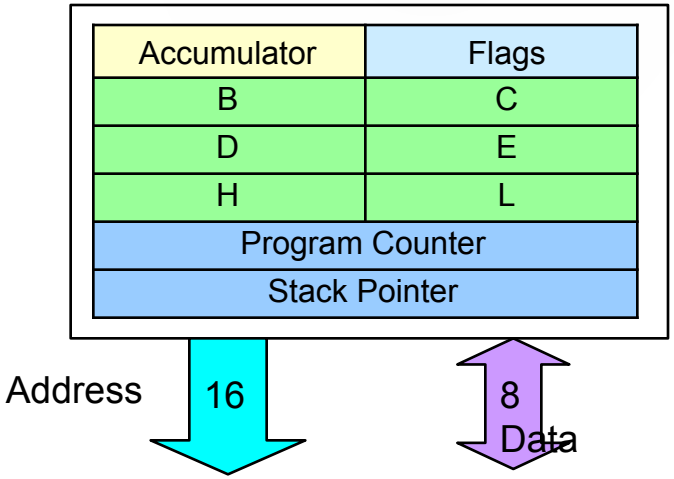
- Microprocessor consists of:
  - **Control unit**: Controls microprocessor operations.
  - **ALU**: Performs data processing function.
  - **Registers**: Provide storage, internal to CPU.
  - **Interrupts**
  - **Internal data bus**

The ALU

- In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.

- Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.

- Registers
  - General Purpose Registers
    - B, C, D, E, H & L (8 bit registers)
    - Can be used singly
    - Or can be used as 16 bit register pairs
      - BC, DE, HL
    - H & L can be used as a data pointer (holds memory address)
  - Special Purpose Registers
    - Accumulator (8 bit register)
      - Store 8 bit data
      - Store the result of an operation
      - Store 8 bit data during I/O transfer



- Flag Register
  - 8 bit register – shows the status of the microprocessor before/after an operation
  - S (sign flag), Z (zero flag), AC (auxillary carry flag), P (parity flag) & CY (carry flag)

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

  - Sign Flag
    - Used for indicating the sign of the data in the accumulator
    - The sign flag is set if negative (1 – negative)
    - The sign flag is reset if positive (0 –positive)

- Zero Flag
  - Is set if result obtained after an operation is 0
  - Is set following an increment or decrement operation of that register

```
  10110011
  --------
1
00000000
```

```
  01001101
+
```

- Carry Flag
  - Is set if there is a carry or borrow from arithmetic operation

```
1011 0101
     +  0110
     1100
Carry
```

```
1011 0101
-  1100 1100
---------
Borrow 1  1110
          1001
```

- Auxillary Carry Flag
  - Is set if there is a carry out of bit 3
- Parity Flag
  - Is set if parity is even
  - Is cleared if parity is odd

Registers….

- The Program Counter (PC)
  - This is a register that is used to control the sequencing of the execution of instructions.
  - This register always holds the address of the next instruction.
  - Since it holds an address, it must be 16 bits wide.
- The Stack pointer
  - The stack pointer is also a 16-bit register that is used to point into memory.
  - The memory this register points to is a special area called the stack.
  - The stack is an area of memory used to hold data that will be retreived soon.
  - The stack is usually accessed in a Last In First Out (LIFO) fashion.

Non Programmable Registers

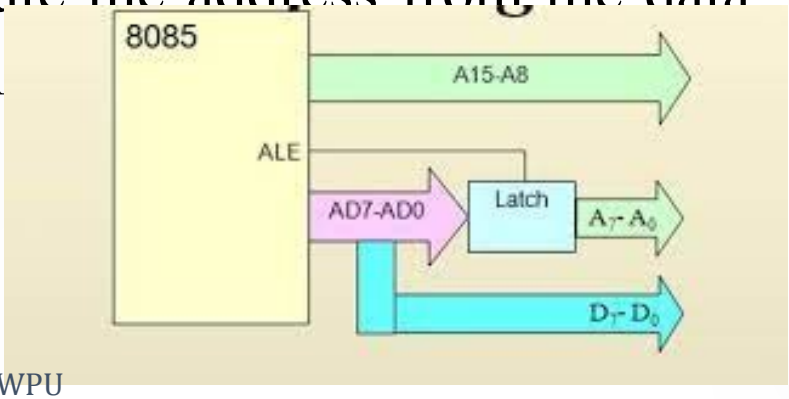
- Instruction Register & Decoder
  - Instruction is stored in IR after fetched by processor
  - Decoder decodes instruction in IR

Internal Clock generator

- 3.125 MHz internally
- 6.25 MHz externally

The Address and Data Busses

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.
  - So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
  - In order to separate the address from the data we can use a latch to save th of the bits changes.



Interrupts

- Interrupt is a process where an external device can get the attention of the microprocessor.
  - The process starts from the I/O device
  - The process is asynchronous.
- Interrupts can be classified into two types:
  - Maskable (can be delayed)
  - Non-Maskable (can not be delayed)
- Interrupts can also be classified into:
  - Vectored (the address of the service routine is hard-wired)
  - Non-vectored (the address of the service routine needs to be supplied externally)

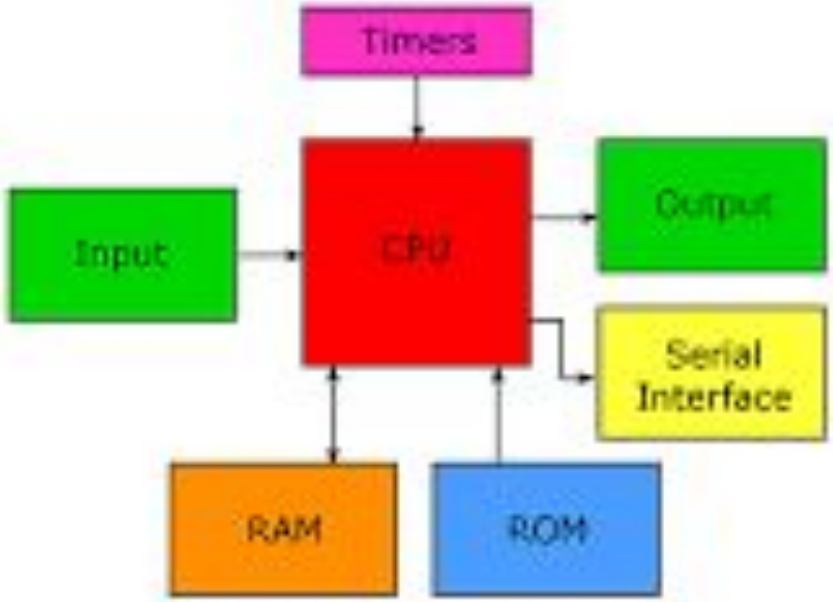
Interrupts…

- An interrupt is considered to be an emergency signal.
  - The Microprocessor should respond to it as soon as possible.
- When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an Interrupt Service Routine (ISR) to respond to the incoming interrupt.
  - Each interrupt will mos

Interrupt name	Maskable	Vectored
INTR	Yes	No
RST 5.5	Yes	Yes
RST 6.5	Yes	Yes
RST 7.5	Yes	Yes
TRAP	No	Yes

Microprocessor Microcontroller

Microprocesser: CPU
and several supporting chips.



Microcontroller: CPU
on a single chip.

Additional links for more information:

- http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/
- https://www.electronicshub.org/half-adder-and-full-adder-circuits/

Thank You