The Public Authority for
Applied Education and Training

College of Technological Studies
Electronics Department

Lecture Notes for

Digital Circuits 1

محاضرات دوائر رقمية 1

By

Raed Ghaddar

Chapter 1
Introductory Concepts

Today, digital systems are widely used in all areas of life such as computers, automotive, toys, medical field, transportation,....etc.

Numerical Representation:

Two ways we can represent numerical values:

a) Analog (عددي) : They can vary over a continuous range of values from 0 to ∞ .

b) Digital (رقمي) : They change in steps

Examples:

- In a digital watch the time is displayed as defined numbers and changes in steps and not continuously. That's why we call it a digital watch.
- In a mercury thermometer, the temperature reading changes continuously and does not show the reading as a defined number. That's why it is an analog device

Digital Number System:

The digital world uses different number systems:

1) Decimal (from 0 to 9)
2) Binary (from 0 to 1)
3) Octal (from 0 to 7)
4) Hexadecimal (from 0 to 15)

Note: In digital systems the numbers are called digits

1) Decimal System

- This number system uses the digits 0,1,2,3,4,5,6,7,8,9
- It's called base-10 number system because it uses 10 different digits
- Each decimal number consists of digits and each digit in the number has a position and a weight.
- The digit to the left of the decimal point (الفاصلة) takes position 0 then the digit left to that is position 1 and then left to that is position 2 and so on. The digit to the right of the decimal point takes position -1 then the digit further right takes position -2 and so on.
- The weight of each digit is equal to 10^p, where (p) is the position of the digit.

Check the example below for the number **7586.491**

position	3	2	1	0		-1	-2	-3
	7	**5**	**8**	**6**	**.**	**4**	**9**	**1**
weight	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}

In this example, the digit 6 takes position 0 and weighs 10^0 or 1 (because $10^0 = 1$). The digit 8 is in position 1 and weighs 10^1 (or 10). The same thing applies to digits 5 and 7. The digit 5 is in position 2 and weighs 10^2 (or 100), while the digit 7 is in position 3 and weighs 10^3(or 1000).
Looking at the right of the decimal point we find the digit 4, which holds position -1 and weighs 10^{-1} (or 0.1). The digit 9 is in position -2 and weighs 10^{-2} (or 0.01) and the digit 1 is in position -3 and weighs 10^{-3} (or 0.001)

Now, let's calculate the value of the number in the example above based on the weights of the digits:

$$\begin{aligned}\mathbf{7586.491} &= 7\text{x}10^3 + 5\text{x}\ 10^2 + 8\text{x}10^1 + 6\text{x}10^0 + 4\text{x}10^{-1} + 9\text{x}10^{-2} + 1\text{x}10^{-3} \\ &= 7000 + 500 + 80 + 6 + 0.4 + 0.09 + 0.001\end{aligned}$$

Note: The first digit from the left in a decimal number is called the Most Significant Digit **(MSD)**. On the other hand, the first digit from the right in a decimal number is called the Least Significant Digit **(LSD)**.

So, in the previous example:

$$\underset{\textbf{MSD}}{\underset{\uparrow}{\mathbf{7}}}\ \mathbf{5\ 8\ 6\ .\ 4\ 9}\ \underset{\textbf{LSD}}{\underset{\uparrow}{\mathbf{1}}}$$

Decimal Counting:

- When we count using decimal numbers the digit in position 0 changes to the next digit every time we count. The maximum value it will take is 9 then on the next count it will go back to zero
- The digit in position 1 changes to the next digit every time the digit in position 0 changes from 9 to 0
- The digit in position 2 changes to the next digit every time the digit in position 1 changes from 9 to 0
- Similarly, for positions 3 and up, the digits in those positions will change only if the digits in the positions prior to them change from 9 to 0.

Example counting :
0000→ 0001→ 0002→0003→. . . . →0009→0010→0011→0012→. . . . →0019→0020→ 0021→. . . →0099 →0100→0101→. . . →0999→1000→1001 →. . .

2) Binary System

- This number system uses the digits 0, and 1 only
- It's called base-2 number system because it uses 2 different digits
- In binary systems we use the term **bit** instead of digit.
- Each binary number consists of bits and each bit in the number has a position and a weight.
  The bit to the left of the decimal point takes position 0 then the bit left to that is position 1 then position 2 and so on. The bit to the right of the decimal point takes position -1 then the bit further right takes position -2 and so on.
- The weight of each bit is equal to 2^p where (p) is the position of the bit
- The first bit from the left is called the Most Significant Bit **(MSB)** and the first bit from the right is called the Least Significant Bit **(LSB)**
- Check the example below for the number 1011.101

position	3	2	1	0		-1	-2	-3
	1	**0**	**1**	**1**	**.**	**1**	**0**	**1**
weight	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
	↑ **MSB**							↑ **LSB**

- The weight of the bits is important in finding the value of the binary number. So to convert the binary number to decimal we need to calculate the value of the number. Let's calculate the value of the binary number in the example above:

$$1\ 0\ 1\ 1\ .\ 1\ 0\ 1$$

$$= 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$$

$$= 1x\,8 + 0x4 + 1x2 + 1x1 + 1x\frac{1}{2} + 0x\frac{1}{4} + 1x\frac{1}{8}$$

$$= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$$

= <u>**11.625**</u> or **$(11.625)_{10}$**← الرقم 10 هو مؤشر لنوع النظام الذي ينتمي له العدد الذي بين قوسين

Binary Counting:

- When we count using binary numbers the bit in position 0 changes to the next bit every time we count. The maximum value it will take is 1 then on the next count it will go back to zero
- The bit in position 1 changes to the next bit every time the bit in position 0 changes from 1 to 0
- The bit in position 2 changes to the next bit every time the bit in position 1 changes from 1 to 0
- Similarly, for positions 3 and up, the bits in those positions will change every time the bits in the position prior to them change from 1 to 0.

- **Example counting :**

0000 → 0001 → 0010→ 0011→0100 →0101→0110→0111→1000→1001 → 1010→1011 →1100 → 1101→1110→1111→10000 →10001 →. . .

- **Notes:**

1) If a binary number has N bits, then we can make 2^N counts.

   Example 1: if a binary number has 2 bits (N=2), then we can make $2^2 = 4$ counts (00→01→10→11)

2) The largest decimal number which we can get from a binary number that has N bits is equal to $(2^N - 1)$.

   Example 2: if N=2, then the maximum binary number is 11.
   The decimal value for this number is: $2^2 - 1 = 4 - 1 = 3$
   This is the largest decimal value for a 2-bit binary number
   We can prove this by converting $(11)_2$ to decimal: $(11)_2 = 1x2^1+1x2^0=2+1=3$

   Example 3: what is the largest binary and decimal number you can get if N=8.

   Answer: maximum binary number= $(11111111)_2$

   maximum decimal value = $2^8 - 1 = (255)_{10}$

Exercises

1- What is the decimal value of $(110101.1)_2$?

2- What are the three numbers that come after $(10110)_2$

10110 → __________ → __________ → __________

3- What is the largest binary number that you can get if the binary number has 10 bits? And what is the value of this number in decimal system?

Max binary number =

Decimal value =

4- Which is bigger $(111)_2$ or $(11)_{10}$?

5- What is the weight of the MSB in $(100011011011)_2$

6- What is the weight of the MSB in a 16-bit binary number

7- Using the weights of the binary bits, what is the value of $(13)_{10}$ in binary

Chapter 2

Number Systems and Codes

Binary to Decimal Conversion:

3) The binary system is the most important one in digital systems
4) The decimal system is universally used
5) We already showed how to convert from binary to decimal

   Example $(10111)_2 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4 = 1 + 2 + 4 + 0 + 16 = (23)_{10}$

6) Shortcut method to convert from binary to decimal:

10	9	8	7	6	5	4	3	2	1	0	Position of bit
1024	512	256	128	64	32	16	8	4	2	1	Weight $=(2)^{position}$

To find the decimal value of a binary number just add the weights of the 1's in the Number.

Example1: $(\ 1\ 0\ 1\ 1\ 1\ )_2$

$16 + 0 + 4 + 2 + 1 = (\ 23\ )_{10}$

Example 2: $(1110)_2 = 8+4+2 = (\ 14\ )_{10}$

Example 3: from decimal to binary: $(5)_{10} = (101)_2$

(في هذا المثال وضعنا "1" في المواقع التي اذا جمعنا اوزانها نحصل على 5 و وضعنا "0" في الموقع الذي لا نحتاجه . لذلك اخترنا
الموقع 0 و الموقع 2 لوضع الرقم "1")

Example 4: $(15)_{10} = (1111)_2$..... (اختر المواقع التي اذا جمعت اوزانها تحصل على 15)

Decimal to Binary Conversion:

Another way to convert from decimal to binary is by doing repeated division over 2. We will explain this method using the following example.

Example 5: **convert $(5)_{10}$ to binary:**

$\frac{5}{2} = 2 \quad remainder(\text{الباقي}) = 1$ **(LSB)**

$\frac{2}{2} = 1 \quad remainder(\text{الباقي}) = 0$

$\frac{1}{2} = 0 \quad remainder(\text{الباقي}) = 1$ **(MSB)**

The binary number consists of all the bits in the remainder: $(101)_2$

Example 6: **convert $(25)_{10}$ to binary:**

$\frac{25}{2} = 12 \quad remainder = 1$ **(LSB)**

$\frac{12}{2} = 6 \quad remainder = 0$

$\frac{6}{2} = 3 \quad remainder = 0$

$\frac{3}{2} = 1 \quad remainder = 1$

$\frac{1}{2} = 0 \quad remainder = 1$ **(MSB)**

The answer is $(11001)_2$

Exercise 1: **Show that $(37)_{10} = (100101)_2$**

Octal Number System

- This number system uses the digits 0, 1,2,3,4,5,6,7
- It's called base-8 system because it uses 8 different digits
- Each octal number consists of digits and each digit in the number has a position and a weight just like decimal and binary number systems.
- The first digit from the left is called the Most Significant Digit **(MSD)** and the first digit from the right is called the Least Significant Digit **(LSD)**
- Check the example below for the number $(2376.451)_8$

position	3	2	1	0		-1	-2	-3
	2	**3**	**7**	**6**	**.**	**4**	**5**	**1**
weight	8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
	↑ **MSD**							↑ **LSD**

Octal to decimal conversion:

Example 7: Convert the number $(2367.451)_8$ to decimal:

$$2\ 3\ 7\ 6\ .\ 4\ 5\ 1$$

$$= 2x8^3 + 3x8^2 + 7x8^1 + 6x8^0 + 4x8^{-1} + 5x\,8^{-2} + 1x8^{-3}$$

$$= 2x\,512 + 3x64 + 7x8 + 6x1 + 4x\frac{1}{8} + 5x\frac{1}{8^2} + 1x\frac{1}{8^3}$$

$$= 1024 + 192 + 56 + 6 + 0.5 + 0.078125 + 0.001953125$$

$$= \mathbf{(1278.580078125)_{10}}$$

Example 8: Convert the number $(372)_8$ to decimal:

$\mathbf{(372)_8} = 2x8^0 + 7x8^1 + 3x8^2 = 2 + 56 + 192$

$= \mathbf{(250)_{10}}$

Decimal to Octal conversion:

Use repeated division by 8. The following example explains how.

Example 9: convert $(266)_{10}$ to octal:

$$\frac{266}{8} = 33 \qquad remainder \ = \ 2 \quad \text{(LSD)}$$

$$\frac{33}{8} = 4 \qquad remainder \ = \ 1$$

$$\frac{4}{8} = 0 \qquad remainder \ = \ 4 \quad \text{(MSD)}$$

The octal number consists of all the digits in the remainder: **$(412)_8$**

Octal to Binary conversion:

Each octal digit is represented by 3 bits binary as shown in the table below.

Octal digit	0	1	2	3	4	5	6	7
Binary value	000	001	010	011	100	101	110	111

To convert from octal to binary, just replace each octal digit with its binary value.

Example 10: Convert $(472)_8$ to binary

472

100 111 010

So $(472)_8 = (100111010)_2$

Example 11: Convert $(6431)_8$ to binary

6431

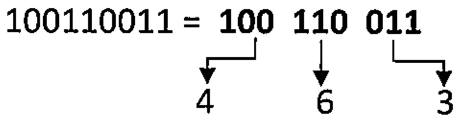
110 100 011 001

So $(6431)_8 = (110100011001)_2$

Binary to Octal conversion:

This is the reverse process of octal to binary conversion. The bits in the binary number are grouped into 3-bit groups starting from the LSB. Then each group of 3 bits is converted to octal.

Example 12: Convert $(100110011)_2$ to octal

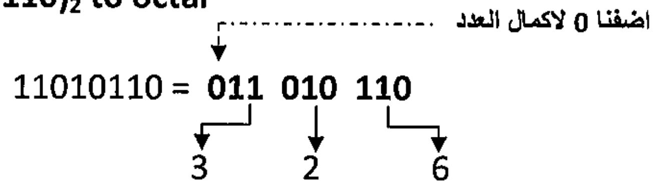


100110011 = **100 110 011**

4 6 3

So $(100110011)_2 = (463)_8$

Example 13: Convert $(11010110)_2$ to octal



اضفنا 0 لاكمال العدد

11010110 = **011 010 110**

3 2 6

So $(11010110)_2 = (326)_8$

Octal Counting:

- When we count using octal numbers the digit in position 0 changes to the next digit every time we count. The maximum value it will take is 7 then on the next count it will go back to zero
- The digit in position 1 changes to the next digit every time the digit in position 0 changes from 7 to 0
- The digit in position 2 changes to the next digit every time the digit in position 1 changes from 7 to 0
- Similarly, for positions 3 and up, the digits in those positions will change every time the digits in the position prior to them change from 7to 0.
- **Example counting :**
  0000 → 0001 → 0002 → 0003 →. . . . → 0007→ 0010 → 0011 → 0012 →. . . .
  →0017→ 0020→ 0021→. . . →0077→0100 → 0101→. . . → 0777→1000 →1001 →. .

Hexadecimal Number System

- This number system uses 16 digits: 0, 1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Hex digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary value	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

- It's called base-16 system because it uses 16 different digits
- Each hex number consists of digits and each digit in the number has a position and a weight just like the other number systems.
- The first digit from the left is called the Most Significant Digit **(MSD)** and the first digit from the right is called the Least Significant Digit **(LSD)**
- The weight of each digit is based on the number 16. The weight is $(16)^p$, where p is the position of the digit. Check the example below for the number $(237A.1D1)_{16}$

position	3	2	1	0		-1	-2	-3
	2	**3**	**7**	**A**	**.**	**1**	**D**	**1**
weight	16^3	16^2	16^1	16^0		16^{-1}	16^{-2}	16^{-3}
	↑ **MSD**							↑ **LSD**

Hex to decimal conversion:

Example 14: Convert the number $(237A.1D1)_{16}$ to decimal:

$$2\ 3\ 7\ A\ .\ 1\ D\ 1$$

$$= 2x16^3 + 3x16^2 + 7x16^1 + 10x16^0 + 1x16^{-1} + 13x\,16^{-2} + 1x16^{-3}$$

$$= 2x\,4096 + 3x256 + 7x16 + 10x1 + 1x\frac{1}{16} + 13x\frac{1}{256} + 1x\frac{1}{4096}$$

$$= 8192 + 768 + 112 + 10 + 0.0625 + 0.05078 + 0.\,000244$$

$$= \underline{\mathbf{(9082.113524)}}_{10}$$

Example 15: **Convert the number $(1BC2)_{16}$ to decimal:**

$(1BC2)_{16}$ $= 2\text{x}16^0 + 12\text{x}16^1 + 11\text{x}16^2 + 1\text{x}16^3 = 2 + 192 + 2816 + 4096$

$= (7106)_{10}$

Decimal to Hex conversion:

Use repeated division by 16

Example 16: **Convert the number $(423)_{10}$ to Hex**

$$\frac{423}{16} = 26 \quad \textit{remainder} = 7$$ **(LSD)**

$\swarrow$

$$\frac{26}{16} = 1 \quad \textit{remainder} = A$$ (10 = A تذكر أن)

$\swarrow$

$$\frac{1}{16} = 0 \quad \textit{remainder} = 1$$ **(MSD)**

So $(423)_{10}$ = $(1A7)_{16}$

Example 17: **Convert the number $(214)_{10}$ to Hex**

$$\frac{214}{16} = 13 \quad \textit{remainder} = 6$$ **(LSD)**

$\swarrow$

$$\frac{13}{16} = 0 \quad \textit{remainder} = D$$ **(MSD)** (13 = D تذكر أن)

So $(214)_{10}$ = $(D6)_{16}$

Hex to Binary conversion:

Each hex digit is represented by 4 bits binary as shown in the table previously shown. Just replace each hex digit with the binary value and put them together.

Example 18: Convert the number $(9F2)_{16}$tobinary

9 F 2

1001 1111 0010

So $(9F2)_{16} = (100111110010)_2$

Example 19: Convert the number $(BA6)_{16}$tobinary

B A 6

1011 1010 0110

so $(BA6)_{16} = (101110100110)_2$

Binary to Hex conversion:

Use the reverse process of the hex to binary conversion. The bits of the binary number are grouped into groups of 4 bits starting from LSB. Convert each group of 4 bits binary to its hex value.

Example 20: Convert $(111010010110)_2$ to hex

111010010110 = **1110 1001 0110**

E 9 6

So $(111010010110)_2 = (E96)_{16}$

Example 21: Convert $(1010111101)_2$ to hex

اضفنا 00لاكمال العدد

1010111101 = 00**10 1011 1101**

2 B D

So $(1010111101)_2 = (2BD)_{16}$

Hex Counting:

Use the same technique used in binary and octal.

Example 22:Count from $(18)_{16}$ to $(22)_{16}$

Answer: 18 →19→1A→1B→1C→1D→1E→1F→20→21→22

BCD Code:

- BCD = Binary Coded Decimal.
- This is a system used to make the conversion between binary and decimal easier.
- The BCD number looks like a binary number but it's different
- In BCD system each decimal digit is represented by 4 bits binary just like hex numbers. The table below shows the relation between decimal digits and BCD.

Decimal digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

BCD to Decimal and Decimal to BCD:

Example 23: Convert $(874)_{10}$ to BCD also convert to binary

874

1000 0111 0100

So, $(874)_{10} = (100001110100)_{BCD}$

To convert $(874)_{10}$ to binary we need to do repeated division by 2:

$\frac{874}{2} = 437 \quad remainder = 0$ **(LSB)**

$\frac{437}{2} = 218 \quad remainder = 1$

$\frac{218}{2} = 109 \quad remainder = 0$

$\frac{109}{2} = 54 \quad remainder = 1$

$\frac{54}{2} = 27 \quad remainder = 0$

$\frac{27}{2} = 13 \quad remainder = 1$

$\frac{13}{2} = 6 \quad remainder = 1$

$\frac{6}{2} = 3 \quad remainder = 0$

$\frac{3}{2} = 1 \quad remainder = 1$

$\frac{1}{2} = 0 \quad remainder = 1$ **(MSB)**

So, $(874)_{10} = (1101101010)_2$ (different from BCD because different system)

Example 24: Convert $(9060)_{10}$ to BCD

$(9060)_{10} = (1001000001100000)_{BCD}$

Example 25: Convert $(1001101000010)_{BCD}$ to decimal

$(1\text{-}0011\text{-}0100\text{-}0010)_{BCD} = (1342)_{10}$

Notes:

-Computers and microcomputers deal with data as bytes
-The **Byte** =8 bits binary number
-The smallest data in a computer is a byte

Example 26: how many bytes are there in 32 bits of data
Answer: Bytes = 32 bits ÷ 8 bits/byte = 4 bytes

Example 27: how many bytes are needed to represent ($846569)_{10}$ in BCD?
Answer: $(846569)_{10} = (\underbrace{1000\ 0100}_{\text{byte}}\ \ \underbrace{0110\ 0101}_{\text{byte}}\ \ \underbrace{0110\ 1001}_{\text{byte}})_{BCD}$ = 3 bytes

Alphanumeric Codes:

- Computers must be able to handle letters, numbers , and special characters in binary system
- It uses a special code system for any character we type.
- The most widely used alphanumeric code (alphabet and numbers code) is called **ASCII.**
- **ASCII** = **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- ASCII is a 7-bit code. This means that any number or letter or character the computer processes consists of 7 binary bits.
- For example, the letter ' Y ' is represented by the ASCII code 1011001

Exercises

1- Convert the following numbers to Decimal:

a) $(01110111)_2$

b) $(165.1)_8$

c) $(37FD)_{16}$

d) $(0010010001101000)_{BCD}$

2- Convert $(405)_{10}$ to binary

3- Convert $(177)_{10}$ to octal and then from octal to binary

4- Convert $(2AF)_{16}$ to octal

5- Convert $(01110000)_{BCD}$ to binary

6- Which is bigger $(00100111)_{BCD}$ or this $(11110)_2$? (وضح كيف)

7- Which is bigger $(30)_{16}$ or this $(11110)_2$? (وضح كيف)

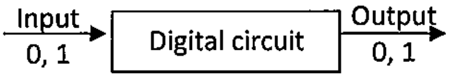
8- Count from $(75)_8$ to $(102)_8$

9- Count from $(F8)_{16}$ to $(102)_{16}$

Chapter 3
Logic Gates and Boolean Algebra

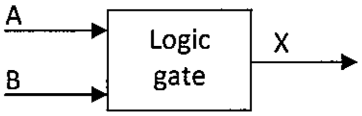
About Digital Circuits:

- The inputs and outputs of digital circuits use two types of voltage values: **0** volt and **5** volts
- Those two voltages are considered binary values, where 0 volt is the digit 0 and 5 volts is the digit 1. Therefore, digital circuits operate in the binary mode.

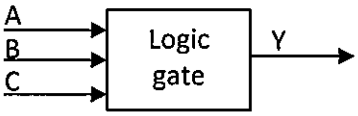


- To design digital circuits we use mathematical operations and rules called **Boolean algebra**, which defines the relation between the input and output in a digital circuit.
- To build a digital circuit we use components (parts) called logic gates.
- Boolean algebra uses three basic operations :
  1) OR
  2) AND
  3) NOT

- In logic gates we use the letters A, B, C, D as inputs and the letters X, Y, Z as outputs:



2-input logic gate



3-input logic gate

A Truth Table:

It is a table that shows the values of both inputs and outputs

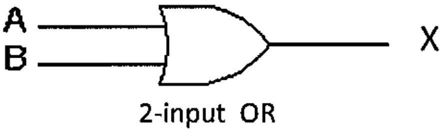
Example:

Inputs		output
A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	1

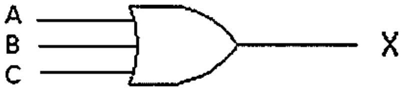
Inputs			output
A	**B**	**C**	**X**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Basic Logic Gates:

1- **OR gate:** This gate has at least two inputs but one output.



2-input OR



3-input OR

We use the operator '+' to express the action of the OR gate. The above circuits can be expressed in terms of equations as follows:

X = A + B (2 inputs)
X = A + B + C (3 inputs)

We call these equations Boolean expressions. A Boolean expression is the equation that explains how the gate works.

The truth table for the OR gate is as follows:

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

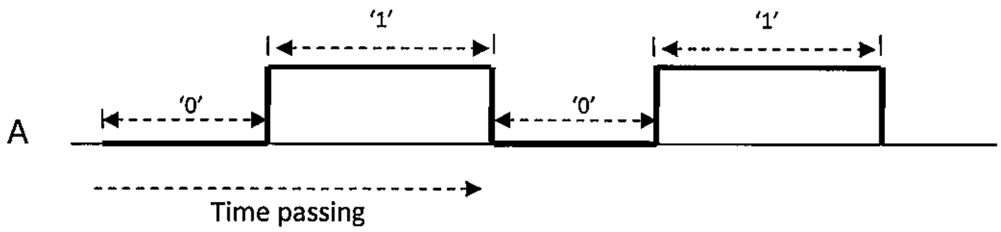
(2-inputs)

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

(3-inputs)

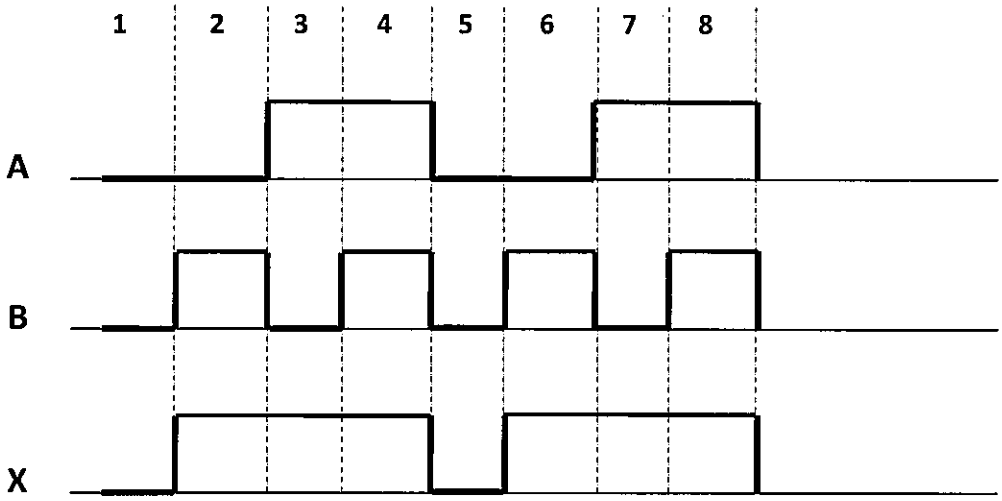
(Note that the output is always '1' except when all the inputs are all '0')

Timing Diagrams: A timing diagram shows the state of the inputs and output as they change with time. This is another way of explaining what is happening in the circuit.
A timing diagram uses apulse to indicate that the value of the input or output is logic '1'. A flat line on the base line indicates a logic '0' as shown in the following diagram:



Timing Diagram for the input A

Now let's try to draw the timing diagram for the 2-input OR gate. First, we give the shape of the 2 inputs A and B then we draw the shape of the output X based on the values of A and B



Timing Diagram for the 2-input OR gate
X = A + B

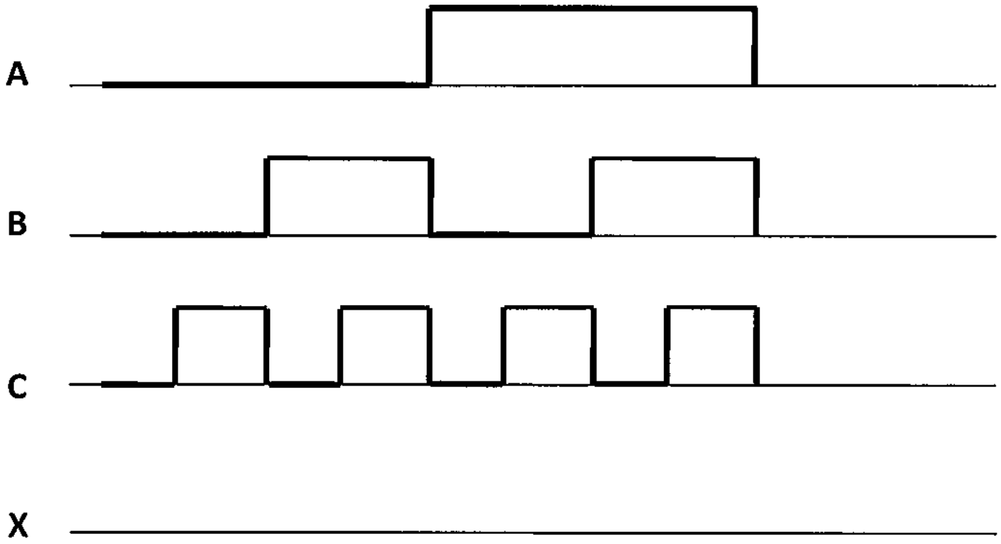
In the timing diagram above we notice that the inputs A and B change values with time, therefore we divided the diagram into periods (from 1 to 8) to calculate the output X as A and B change.

For example, in period 2 in the diagram above, we can see that A=0 and B=1. Since the two inputs are going into an OR gate then from the truth table we know that if A=0 and B=1 then X=1.

Another example, in period 5, we can see that A=0 and B=0, from the truth table of the OR gate we know that if A=0 and B=0 then X=0.

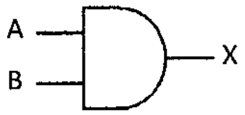
This is how we draw the output X based on the values of the inputs in the different periods.

<u>Exercise1</u>: Draw the waveform of the output X for a 3-input OR gate with A, B and C as inputs.

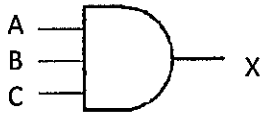


Timing Diagram for the 3-input OR gate
X = A + B + C

2- **AND Gate:** This gate has at least two inputs but one output.



2-input AND



3-input AND

We use the operator '·' to express the action of the AND gate. The Boolean expressions for the AND gate are as follows:

2 inputs: **X = A · B** or **X = AB**
3 inputs: **X = A · B · C** or **X = ABC**

The truth table for the AND gate is:

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

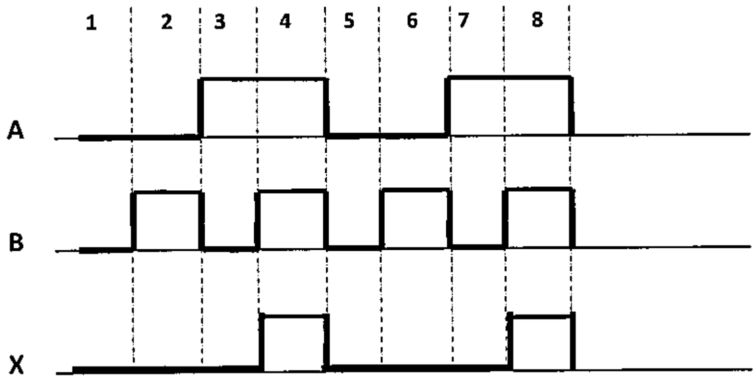
(2-inputs)

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

(3-inputs)

(Note that the output is always '0' except when all the inputs are all '1')

Timing Diagram:

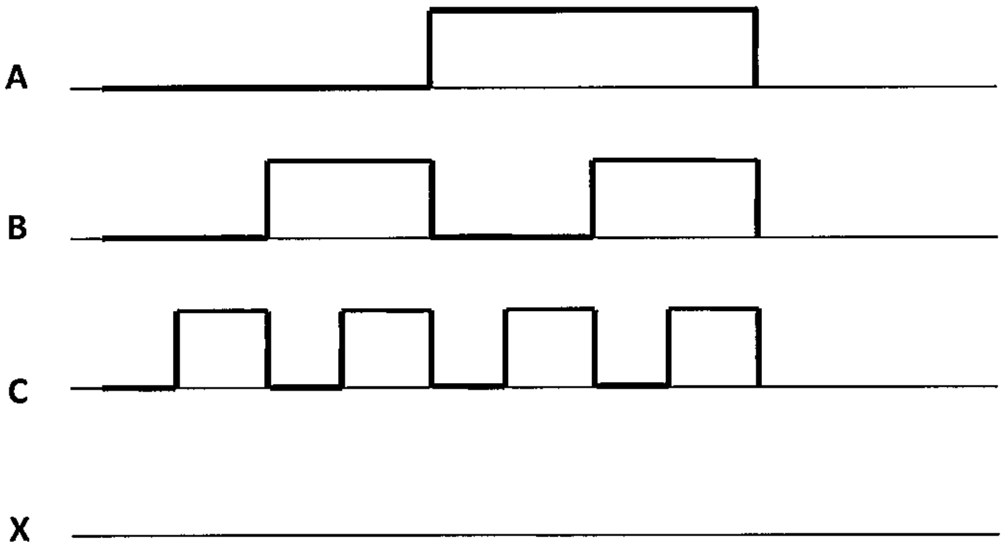


Timing Diagram for the 2-input AND gate

Notes:

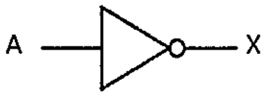
- In period 1 in the diagram above, we can see that A=0 and B=0. Since the two inputs are going into an AND gate then from the truth table we know that if A=0 and B=0 then X=0.
- In period 2, we see that A=0 and B=1, from the truth table we know that if A=0 and B=1 then X=0.
- In period 4, we see that A=1 and B=1, from the truth table we know that if A=1 and B=1 then X=1.

<u>Exercise2</u>: Draw the waveform of the output X for a 3-input AND gate with A, B and C as inputs.



Timing Diagram for the 3-input AND gate
X = A · B ·C

3- **<u>NOT Gate (inverter):</u>** This gate has only one input and one output.



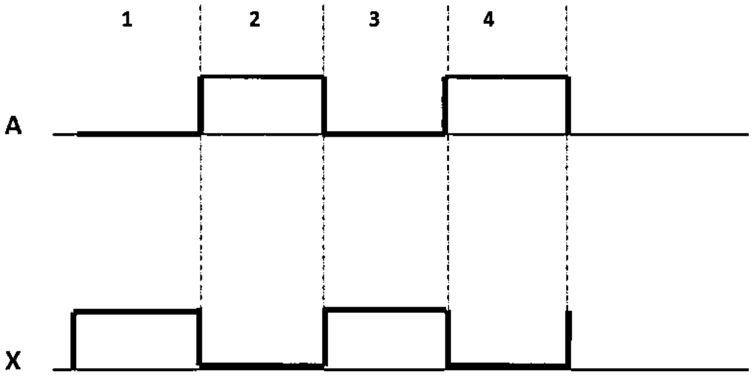
We use a bar over the letter (¯) to express the action of the inverter. The Boolean expression for the inverter is as follows:

$$X = \bar{A}$$

The truth table for the inverter is:

A	X
0	1
1	0

Timing Diagram:

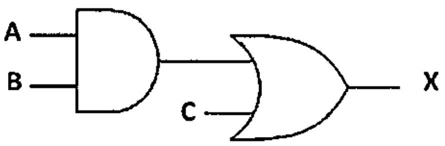


Timing Diagram for the inverter
X = $\bar{A}$

In the diagram above, we can see that when A=0 then X=1.
Also, note that when A=1, then X = 0.

Describing Logic Circuits Using Boolean Expressions:

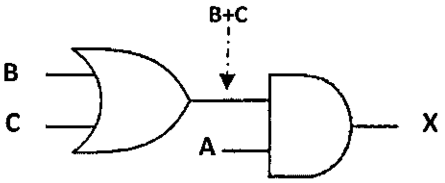
Any logic circuit can be described in terms of a Boolean expression. Let's show how we can do this using the following circuit:



We need to find the expression for the output X. The first gate on the left is the AND gate. The output of this gate is A·B or AB. This output goes to one of the inputs of the OR gate where C goes to the other input. Therefore, the expression for the OR gate gives the expression for the output X:

$$X = AB + C$$

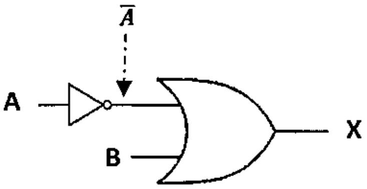
Example1: write the expression for the output X in the following circuit:



$$X = (B + C) \cdot A$$

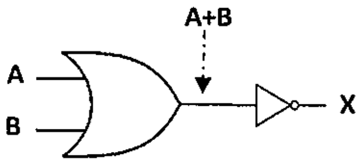
Example 2: write the expression for the output X for the following circuits:

a)



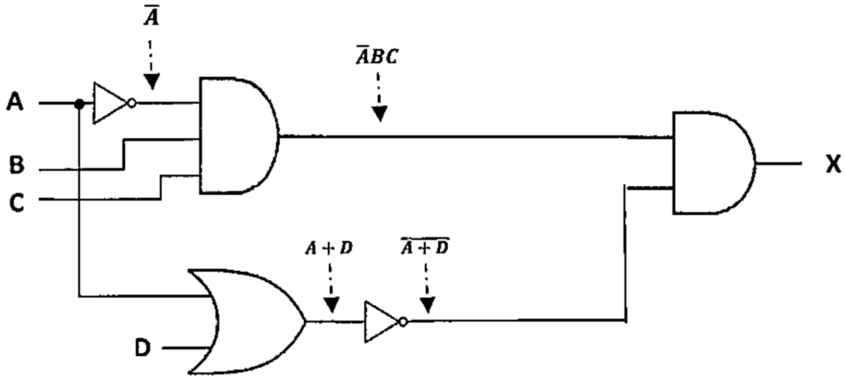
$$X = \bar{A} + B$$

b)



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

c)



$$X = (\bar{A}BC) \cdot (\overline{A+D})$$

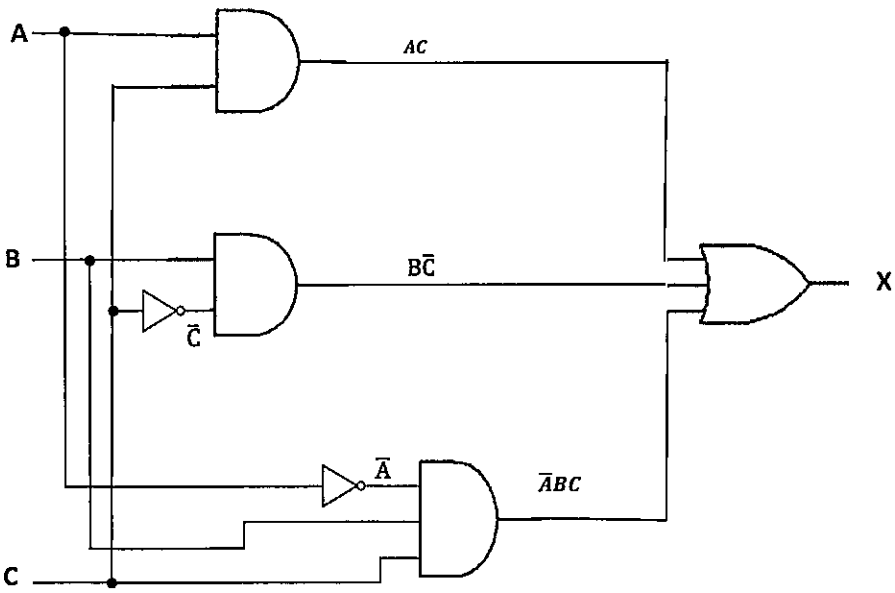
Drawing Logic Circuits from Boolean Expressions:

Here we learn how to draw the logic circuit if we have the Boolean expression.

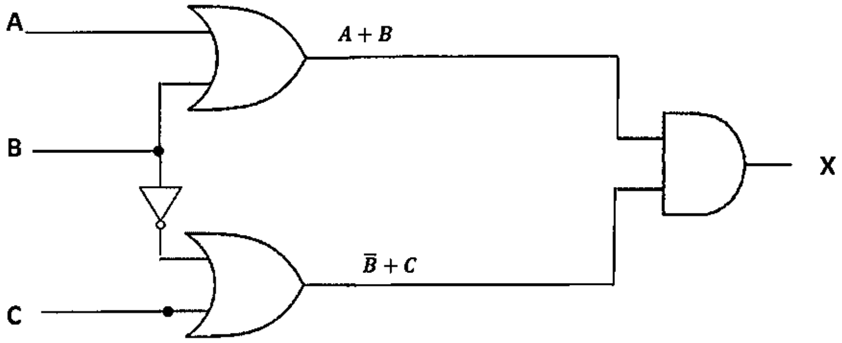
Example 3: We will use the following expression as an example:

$$X = AC + B\bar{C} + \bar{A}BC$$

Now draw the circuit diagram



Example 4: **Draw the logic circuit for** $X = (A + B)(\overline{B} + C)$



Evaluating Outputs:(كيفية حساب قيمة X)

Example 5: if $X = \bar{A}BC(\overline{A + D})$

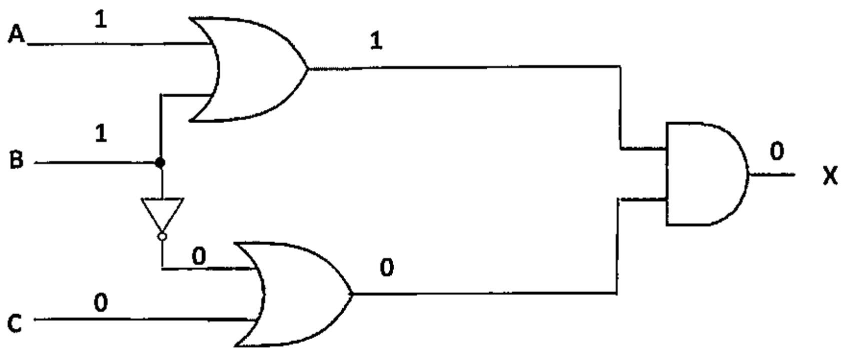
What is the value of the output X if:
1) A=0, B=0, C=1, D=0
2) A=0, B=1, C=1, D=0

Answer:

1) $X = \bar{0}\cdot 0\cdot 1\cdot(\overline{0 + 0})$
$= 1\cdot 0\cdot 1\cdot(\bar{0})$
$= 0\cdot(1)$
$= 0$

2) $X = \bar{0}\cdot 1\cdot 1\cdot(\overline{0 + 0})$
$= 1\cdot 1\cdot 1\cdot(\bar{0})$
$= 1\cdot(1)$
$= 1$

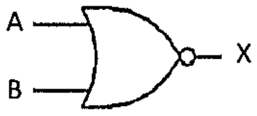
Example 6: Find the value of X for the following logic circuit if A=1, B=1, C=0:



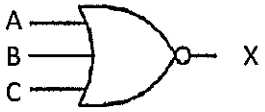
Answer: X = 0

NOR and NAND Gates:

1- **NOR gate:** This gate has at least two inputs but one output.



2-input NOR



3-input NOR

The Boolean expressions for the NOR gate are as follows:

2 inputs: $X = \overline{A + B}$

3 inputs: $X = \overline{A + B + C}$

The truth table for the NOR gate is:

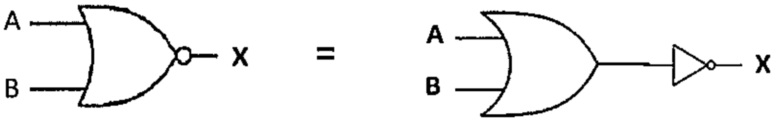
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(2-inputs)

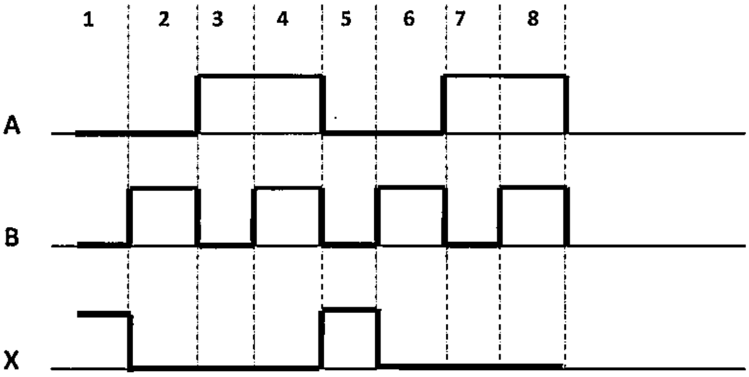
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(3-inputs)

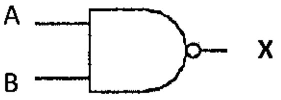
NOR = OR + NOT



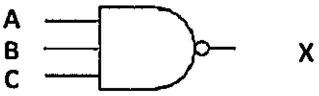
Timing Diagram:



2- **<u>NAND gate:</u>** This gate has at least two inputs but one output.



2-input NAND



3-input NAND

The Boolean expressions for the NAND gate is as follows:

2 inputs: $X = \overline{A \cdot B}$ *or* $X = \overline{AB}$

3 inputs: $X = \overline{A \cdot B \cdot C}$ *or* $X = \overline{ABC}$

The truth table for the NAND gate is:

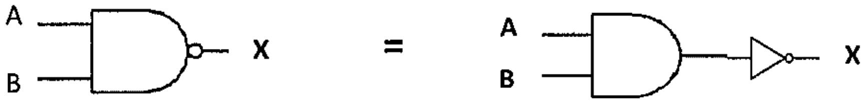
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(2-inputs)

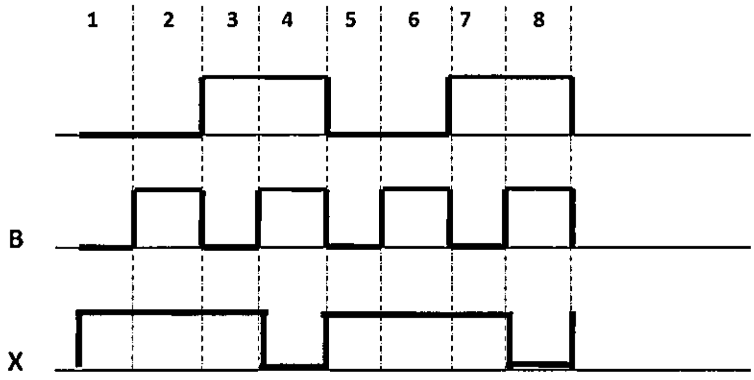
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(3 inputs)

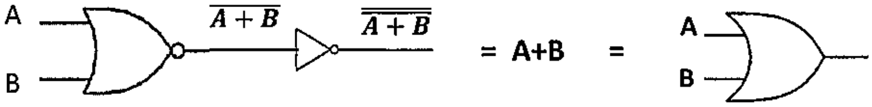
NAND = AND + NOT

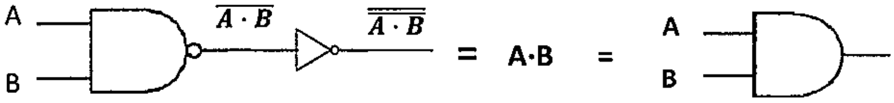


Timing Diagram:



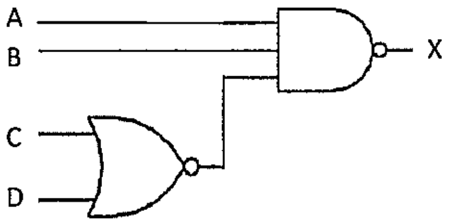
Note: if two NOT gates come after each other, then, both NOT gates are cancelled. So if you add an inverter to a NOR gate it will become an OR gate. Similarly, if you add an inverter to a NAND gate it will become an AND gate.





Example 7: Draw the logic circuit using only NOR and NAND gates to express the following Boolean expression: $X = \overline{A \cdot B \cdot (\overline{C + D})}$

Answer:



Example 8: What is the value of X in example 7 if A=B=C=1 and D=0?

Answer:

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

$$X = \overline{1 \cdot 1 \cdot (\overline{1 + 0})}$$

$$X = \overline{1 \cdot 1 \cdot (\overline{1})}$$

$$X = \overline{1 \cdot 1 \cdot 0}$$

$$X = \overline{0} = 1$$

Boolean Theorems (نظريات بوليان):

There are 15 Boolean theorems that help simplify logic expressions. The theorems are as follows:

1	$A \cdot 0 = 0$	5	$A + 0 = A$	9	$A + B = B + A$	13	A(B+C) = AB+AC (A+B)(C+D)=AC+AD+BC+BD
2	$A \cdot 1 = A$	6	$A + 1 = 1$	10	AB = BA	14	A+AB = A
3	$A \cdot A = A$	7	$A + A = A$	11	A+(B+C) = (A+B)+C = A+B+C	15	$A+\overline{A}B=A+B$ $\overline{A}+AB = \overline{A}+B$
4	$A \cdot \overline{A} = 0$	8	$A + \overline{A} = 1$	12	A(BC)=(AB)C=ABC		

Note: Theorem 13 can be used in the reverse way by taking a common factor. For example, AB + AC = A (B + C).
Another example, ABC + ABD = AB (C + D)

DeMorgan's Theorem (نظريات ديمورغان):
DeMorgan added two more theorems to Boolean's theorems. They are as follows:

1) $\overline{AB} = \bar{A} + \bar{B}$

2) $\overline{A + B} = \bar{A} \cdot \bar{B}$

Example 9: Simplify the following equations using Boolean's theorems:

1) $X = A\bar{B}D + A\bar{B}\bar{D}$
2) $X = (\bar{A} + B)(A + B)$
3) $X = ACD + \bar{A}BCD$

Answer:

1) $X = A\bar{B}D + A\bar{B}\bar{D}$
$= A\bar{B}(D + \bar{D})$ (theorem #13)
$= A\bar{B}(1)$ (#8)
$= A\bar{B}$ (#2)

2) $X = (\bar{A} + B)(A + B)$
$= \bar{A}A + \bar{A}B + BA + BB$ (#13)
$= 0 + B(\bar{A}+A) + B$ (# 4, #13, #3)
$= B(\bar{A}+A) + B$ (#5)
$= B(1) + B$ (#8)
$= B + B$ (#2)
$= B$ (#7)

3) $X = ACD + \bar{A}BCD$
$= CD\,(A + \bar{A}B)$ (#13)
$= CD\,(A + B)$ (#15)

Example 10: Simplify the expression $Z = \overline{(\bar{A} + C)(B + \bar{D})}$

Answer:

$z = \overline{(\bar{A} + C)} + \overline{(B + \bar{D})}$ (DeMorgan's #1)

$Z = \bar{\bar{A}}\,\bar{C} \quad + \quad \bar{B}\bar{\bar{D}}$ (DeMorgan's #2)

$Z = A\bar{C} \quad + \quad \bar{B}D$ (double inverters are cancelled)

Example 11: Simplify the expression $Z = \overline{A + \bar{B}C}$

Answer:

$Z = \bar{A} \cdot \overline{\bar{B}C}$ (DeMorgan's #2)

$Z = \bar{A} \cdot (\bar{\bar{B}} + \bar{C})$ (DeMorgan's #1)

$Z = \bar{A} \cdot (B + \quad \bar{C})$ (double inverters are cancelled)

Note: DeMorgan's theorems can be applied to more than just 2 terms such as the following:

$\overline{XYZ} = \bar{X} + \bar{Y} + \bar{Z}$

$\overline{ABCD} = \bar{A} + \bar{B} + \bar{C} + \bar{D}$

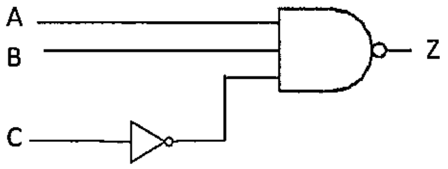
$\overline{X + Y + Z} = \bar{X}\,\bar{Y}\,\bar{Z}$

$\overline{A \ + \ B \ + \ C \ + \ D} = \bar{A}\,\bar{B}\,\bar{C}\bar{D}$

Example 12: For the circuit shown below answer the following

1) Determine the expression of the output Z

2) Simplify the expression



Answer:

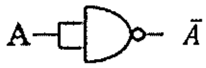
1) Z = $\overline{\mathrm{AB}\bar{\mathrm{C}}}$

2) Z = $\bar{A} + \bar{B} + \bar{\bar{C}}$ (DeMorgan's #1)

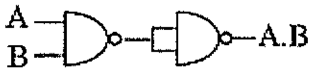
$= \bar{A} + \bar{B} + C$

Building The Basic Gates (OR, AND,NOT) Using Only NAND Gates:

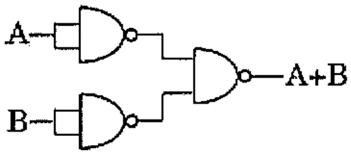
1) NOT gate:



2) AND gate:



3) OR gate:



Exercises

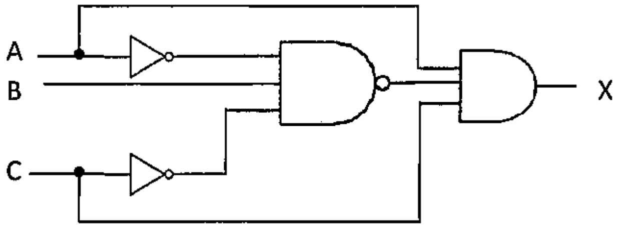
1) Simplify the following expressions:

a) $X = \overline{\bar{A}B\bar{C}}$

b) $Y = \bar{A}B\bar{C} + AB\bar{C} + B\bar{C}D$

c) $Z = (A + B)(\bar{A} + C)(\bar{B} + \bar{C})$

2) For the circuit shown below answer the following



a) Determine the expression of the output X

b) Simplify the expression

3) Build the basic gates (OR, AND, NOT) using NOR gates only

a) NOT gate

b) OR gate

c) AND gate

Chapter 4
Combinational Logic Circuits

Introduction: In chapter 3 we studied basic logic gates and used Boolean algebra to describe the operation of the gates. We also covered circuits that have more than one logic gate. We call these circuits Combinational logic gates.
In this chapter we continue working on methods to simplify combinational logic gates.

Definitions:

A product term: is a group of inputs "AND"ed together. For example, **AB** is called a product term. **A$\bar{B}$C is** also considered a product term. In a product term the 'NOT' bar we use over the inputs cannot extend to cover two or more inputs at the same time. For example $\overline{AB}$ is not a product term but $\bar{A}\bar{B}$ is a product term.

Sum Of Product (SOP): is "OR"ing two or more product terms.
For example **AB + ABC + C** is called SOP term.

Example 1: which of the following forms are considered SOP?

a) AB + CD + E (SOP)
b) AB(C + D) (not SOP)
c) AB + $\overline{CD}$ (not SOP)
d) (A+B)(C+D) (not SOP)
e) AB + $\bar{C}$ (SOP)
f) ABBC$\bar{D}$ (SOP)

Simplifying Expressions:

When you simplify expressions, you need to do the following

a) Put the expression in an SOP form
b) Look for common factors among terms
c) Apply Boolean and DeMorgan's theorems

Example 2: Simplify $Z = \bar{A}C\overline{(\bar{A}BD)} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$

Answer:

$$Z = \bar{A}C(\bar{\bar{A}} + \bar{B} + \bar{D}) + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$$
$$Z = \bar{A}C(A + \bar{B} + \bar{D}) + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$$
$$Z = \bar{A}CA + \bar{A}C\bar{B} + \bar{A}C\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$$
$$Z = 0 + \bar{A}C\bar{B} + \bar{A}C\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$$
$$Z = 0 + \bar{A}C\bar{B} + \bar{A}C\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$$

$Z = \bar{A}C\bar{B} + A\bar{B}C + \bar{A}C\bar{D} + \bar{A}B\bar{C}\bar{D}$ (اعادة ترتيب)

$Z = C\bar{B}(\bar{A} + A) + \bar{A}\bar{D}(C + B\bar{C})$ (اخذ معامل مشترك)

$Z = C\bar{B}(1) + \bar{A}\bar{D}(C + B)$ (Boolean #8 and #15)

$Z = C\bar{B} + \bar{A}\bar{D}(C + B)$ (Boolean #2 and #13)

Example 3: Simplify $Z = (\bar{A} + B)(A + B + D)\bar{D}$

Answer:

$$Z = (\bar{A}A + \bar{A}B + \bar{A}D + BA + BB + BD)\bar{D}$$
$$Z = (0 + \bar{A}B + \bar{A}D + BA + B + BD)\bar{D}$$
$$Z = 0 + \bar{A}B\bar{D} + \bar{A}D\bar{D} + BA\bar{D} + B\bar{D} + BD\bar{D}$$
$$Z = \bar{A}B\bar{D} + 0 + BA\bar{D} + B\bar{D} + 0$$
$$Z = B\bar{D}(\bar{A} + A) + B\bar{D}$$
$$Z = B\bar{D}(1) + B\bar{D}$$
$$Z = B\bar{D} + B\bar{D}$$
$$Z = B\bar{D}$$

Designing Logic Circuits from Truth Tables:

So far, we have been designing logic circuits using Boolean expressions. Now we want to show how we can design logic circuits using truth tables.

Let's take for example the following truth table:

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

First, we look at the times when X=1. We can see that X=1 at two different times.

One time is when A=0 and B=1. Now, for these values of A and B we want to write a product term that contains A and B, which will generate an output equals to 1. The answer is: $\bar{A}$B . Since $\bar{A}$=1 and B=1 then $\bar{A}$B =1.

We also notice that X= 1 when A=1 and B=0. So the product term containing A and B that will produce a '1' in this case is: A$\bar{B}$

So, the expression which will give the result in the table above consists of the two product terms added together:

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

Example 4: a) Write the Boolean expression for the circuit described by the following truth table.
b) Simplify the expression

Answer:

a)

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

1) when A=0, B=1, C=0 → Z=1
product term: $\bar{A}B\bar{C}$

2) when A=0, B=1, C=1 → Z=1
product term: $\bar{A}$BC

3) when A=1, b=1, C=1 → Z=1
product term: ABC

The Boolean expression for Z will be the sum of all the product terms:
Z=$\bar{A}B\bar{C}$ + $\bar{A}$BC+ ABC

b) Now let's simplify the expression using Boolean rules

Z=$\bar{A}$B($\bar{C}$ + C) + ABC
Z=$\bar{A}$B(1) + ABC
Z= $\bar{A}$B + ABC
Z= B($\bar{A}$ + AC)
Z= B($\bar{A}$+C)

Note: make sure that the expression satisfies the truth table

Example 5: a) Write the Boolean expression for the following truth table.
b) Simplify the expression.
c) Use the simplified expression to draw the logic circuit

A	B	C	Z	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	→ $\bar{A}BC$
1	0	0	0	
1	0	1	1	→ $A\bar{B}C$
1	1	0	1	→ $AB\bar{C}$
1	1	1	1	→ ABC

Answer:

a) $Z = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

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

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

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

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

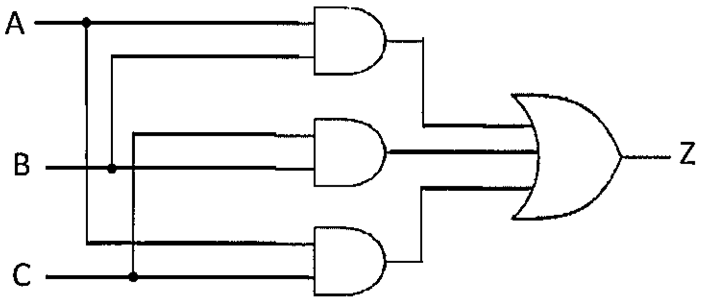
$Z = \bar{A}BC + AC + AB$

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

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

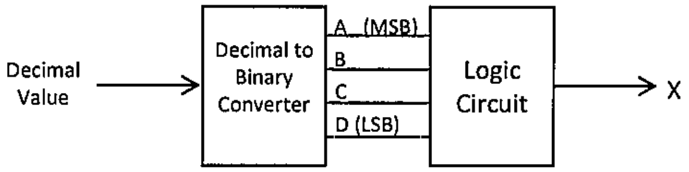
$Z = BC + AC + AB$

c)



Example 6: **Design a logic circuit that will read the binary value from the decimal to binary converter and produce an output of 1 when the binary value is greater than 0110. You need to do the following:**

1) **Write the truth table using A, B, C, and D as inputs and X as an output. Remember, X =1 only when ABCD is greater than 0110**
2) **Write the product terms on the truth table**
3) **Write the Boolean expression for X**
4) **Simplify the expression**
5) **Draw the circuit diagram**



Answer:

1) &2)

A	B	C	D	Z	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	1	→ $\bar{A}BCD$
1	0	0	0	1	→ $A\bar{B}\bar{C}\bar{D}$
1	0	0	1	1	→ $A\bar{B}\bar{C}D$
1	0	1	0	1	→ $A\bar{B}C\bar{D}$
1	0	1	1	1	→ $A\bar{B}CD$
1	1	0	0	1	→ $AB\bar{C}\bar{D}$
1	1	0	1	1	→ $AB\bar{C}D$
1	1	1	0	1	→ $ABC\bar{D}$
1	1	1	1	1	→ $ABCD$

3) $X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}\overline{\mathrm{C}}\overline{\mathrm{D}} + \mathrm{A}\overline{\mathrm{B}}\overline{\mathrm{C}}\mathrm{D} + \mathrm{A}\overline{\mathrm{B}}\mathrm{C}\overline{\mathrm{D}} + \mathrm{A}\overline{\mathrm{B}}\mathrm{CD} + \mathrm{AB}\overline{\mathrm{C}}\overline{\mathrm{D}} + \mathrm{AB}\overline{\mathrm{C}}\mathrm{D} + \mathrm{ABC}\overline{\mathrm{D}} + \mathrm{ABCD}$

4) $X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}\overline{\mathrm{C}}(\overline{\mathrm{D}} + \mathrm{D}) + \mathrm{A}\overline{\mathrm{B}}\mathrm{C}(\overline{\mathrm{D}} + \mathrm{D}) + \mathrm{AB}\overline{\mathrm{C}}(\overline{\mathrm{D}} + \mathrm{D}) + \mathrm{ABC}(\overline{\mathrm{D}} + \mathrm{D})$

$X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}\overline{\mathrm{C}}(1) + \mathrm{A}\overline{\mathrm{B}}\mathrm{C}(1) + \mathrm{AB}\overline{\mathrm{C}}(1) + \mathrm{ABC}(1)$

$X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}\overline{\mathrm{C}} + \mathrm{A}\overline{\mathrm{B}}\mathrm{C} + \mathrm{AB}\overline{\mathrm{C}} + \mathrm{ABC}$

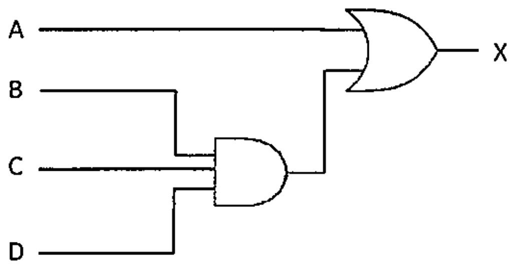
$X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}(\overline{\mathrm{C}} + \mathrm{C}) + \mathrm{AB}(\overline{\mathrm{C}} + \mathrm{C})$

$X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}}(1) \qquad + \mathrm{AB}(1)$

$X = \overline{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\overline{\mathrm{B}} + \mathrm{AB}$

$X = \overline{A}BCD + A(\overline{B} + B)$
$X = \overline{A}BCD + A(1)$
$X = \overline{A}BCD + A$
$X = BCD + A$ .... (Boolean #15)

5)



Karnaugh Map Method (K-MAP):

K-Map is a method that uses special tables to find a simplified expression for the output.

From Truth table to K-Map:

a) 2-variable system

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

Truth table

------------->

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

K-Map

Notes:

1) In the K-map, we draw a table that has cells equal to the number of values that the output takes. In this example, the truth table shows 4 values for the output X, so the K-map has to have 4 cells.

2) We label the top side of the map with one of the 2 input variables (here we use B), and the left side of the map is labeled with the other input (in this case we use A).

3) The labels A and B are shown with the two possibilities as inverted and not inverted. A non-inverted variable means that this variable is '1' and the inverted variable means that the variable is '0'. For example, B in the map, means that B=1, and $\bar{B}$ means that B=0. The same goes for A.

4) Now the values in the cells of the K-map are the values of the output X in the truth table. Note in the K-map, we put '1' in the cell which is labeled$\bar{A}$ and $\bar{B}$. This is because in the truth table when A=0 and B=0 , then X=1 (remember that $\bar{A}$ in the map means that A=0 and $\bar{B}$ in the map means B=0). Also note that in the cell labeled A and $\bar{B}$ the value of X equals to 0 (because in the truth table, when A=1 and B=0, then X=0).

b) 3-variable system

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

Truth table

------------>

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

K-Map

Notes:

1) In the truth table the output X has 8 values, so the K-map will have 8 cells.

2) Since the system has three inputs and our K-map uses two sides to label the inputs, then two of the input variables will be used on one side of the map (in this case we used the top side) and the remaining input variable will label the other side (the left side).

3) Note the order we put AB with. You need to memorize the order.

b) 4-variable system

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

Truth table

------------->

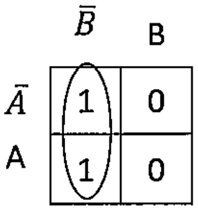
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	0	1
$\bar{C}D$	1	0	0	1
CD	0	0	0	0
$C\bar{D}$	1	1	1	1

K-Map

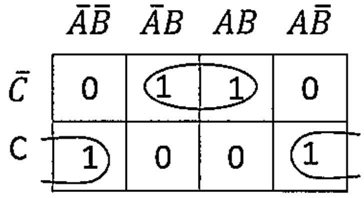
Looping:

To get simplified expressions from K-maps we need to do a process called looping. There are three types of loops: a pair (2), a quad (4) , and an octet (8).

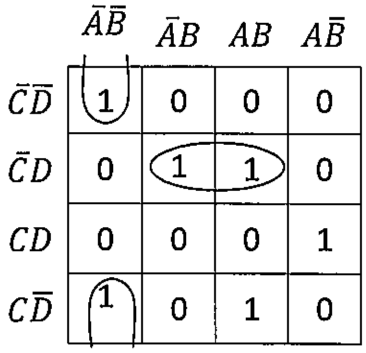
1) Pair loops: making loops (circles) around two adjacent 1's. Examples:



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

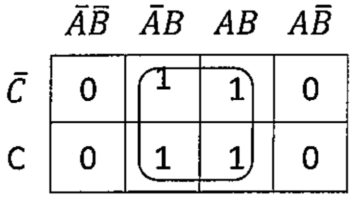


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	1	1	0
C	1	0	0	1

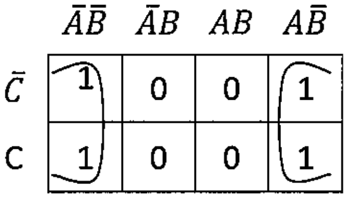


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	0	0
$\bar{C}D$	0	1	1	0
CD	0	0	0	1
$C\bar{D}$	1	0	1	0

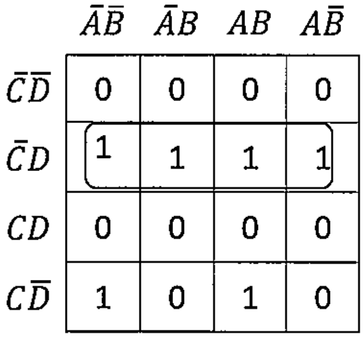
2) Quad loops: making loops around four adjacent 1's. Examples:



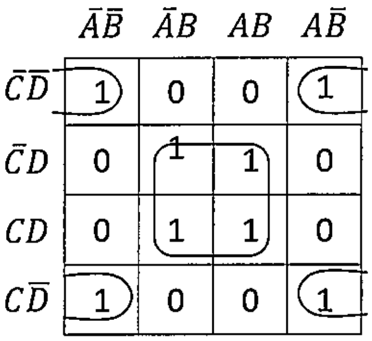
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	1	1	0
C	0	1	1	0



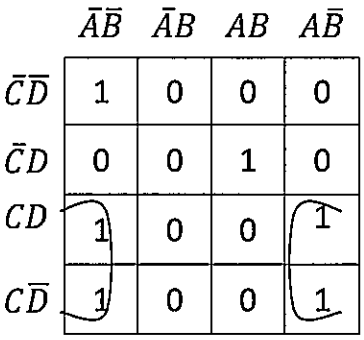
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	1	0	0	1
C	1	0	0	1



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	1	1	1
CD	0	0	0	0
$C\bar{D}$	1	0	1	0

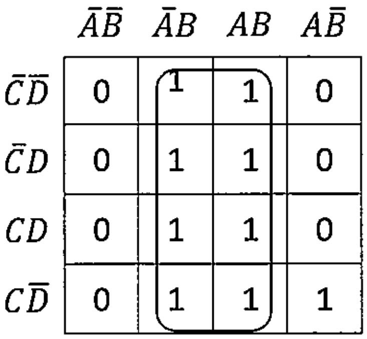


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	0	1
$\bar{C}D$	0	1	1	0
CD	0	1	1	0
$C\bar{D}$	1	0	0	1

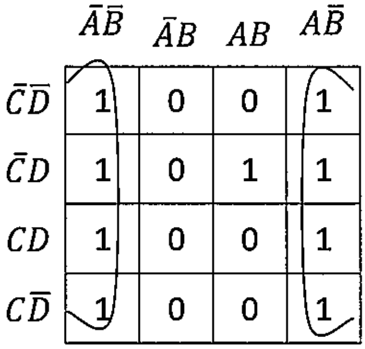


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	0	0
$\bar{C}D$	0	0	1	0
CD	1	0	0	1
$C\bar{D}$	1	0	0	1

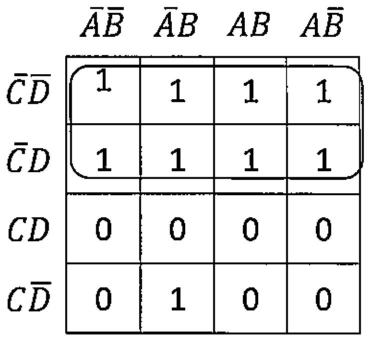
3) Octet loops: making loops (circles) around eight adjacent 1's. Examples:



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	1	1	0
$\bar{C}D$	0	1	1	0
CD	0	1	1	0
$C\bar{D}$	0	1	1	1



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	0	1
$\bar{C}D$	1	0	1	1
CD	1	0	0	1
$C\bar{D}$	1	0	0	1

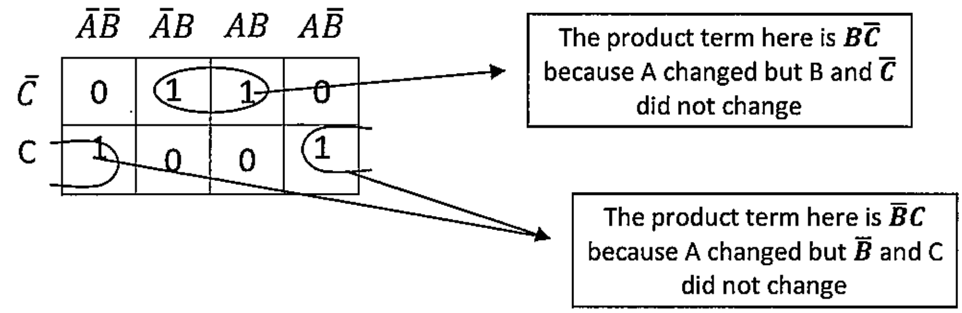


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	1	1
$\bar{C}D$	1	1	1	1
CD	0	0	0	0
$C\bar{D}$	0	1	0	0

Converting Loops into product terms:

Each loop you make is a product term. Remember that a product term is the multiplication of input terms together. To find the product term of any loop, look at the labels at the top and the left side of the k-map that cover the loop. If you see a change in condition of any input in this loop (like if you see A and $\bar{A}$ covering this loop) then don't include it in the product term. Otherwise, this input will be included in the product term.

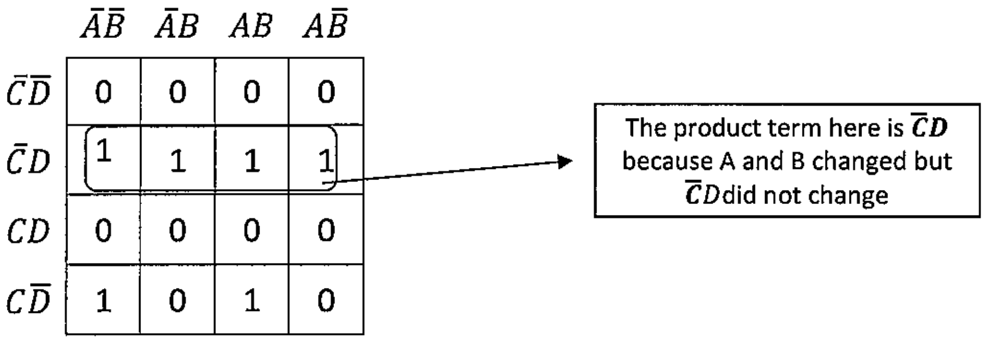
Check the following examples:



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

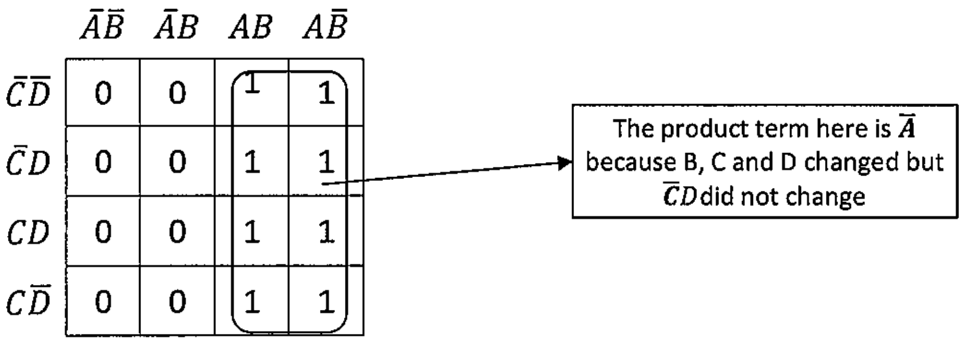
The product term here is $B\bar{C}$ because A changed but B and $\bar{C}$ did not change

The product term here is $\bar{B}C$ because A changed but $\bar{B}$ and C did not change



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	1	1	1
CD	0	0	0	0
$C\bar{D}$	1	0	1	0

The product term here is $\bar{C}D$ because A and B changed but $\bar{C}D$did not change



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	1
$\bar{C}D$	0	0	1	1
CD	0	0	1	1
$C\bar{D}$	0	0	1	1

The product term here is $\bar{A}$ because B, C and D changed but $\bar{C}D$did not change

Making Simplified Boolean Expressions Using K-Maps:

1- Draw the k-map using the truth table or a given expression.

2- Loop any isolated '1' you find in the k-map (any '1' that is not next to any 1)

3- Look for '1's that are next to only one '1' and loop them. Examples:

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	1
$\bar{C}D$	0	1	0	1
CD	0	1	0	1
$C\bar{D}$	0	0	0	1

Only two loops

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	1	0	0
$\bar{C}D$	0	1	1	1
CD	0	0	0	1
$C\bar{D}$	1	1	0	1

nothing applies here

4- Look for octets and loop them
5- Look for quads and loop them
6- Look for pairs and loop them
7- Give each loop you make a product term
8- The simplified expression is the sum of all the product terms you have

Notes:

- Apply the k-map rules in the same order shown above
- a '1' can be included in more than one loop
- Stop making loops when all the '1's in the k-map are covered
- There may be more than one solution

Example 7: Use k-map method to find the simplified expression for the system shown in the truth table below:

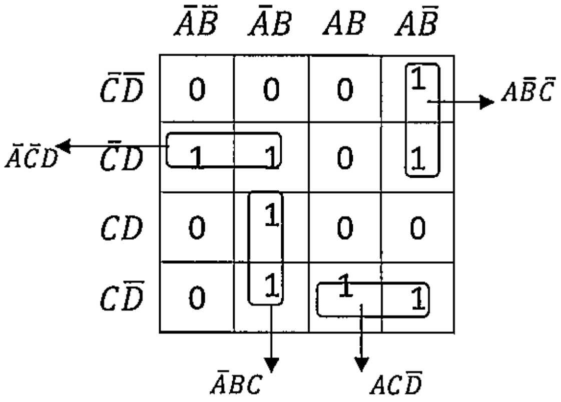
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Answer: Solution number 1

k-map →

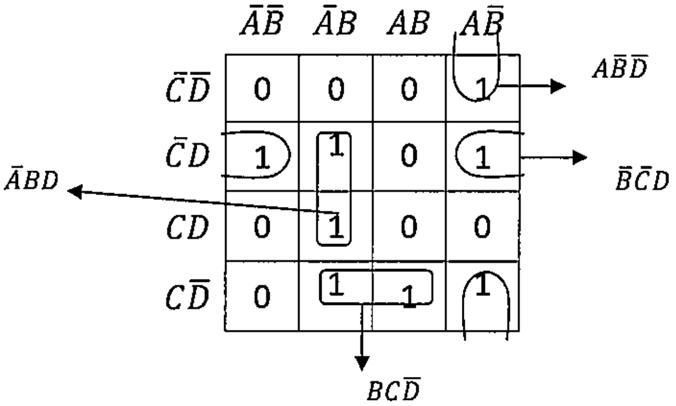
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	1
$\bar{C}D$	1	1	0	1
CD	0	1	0	0
$C\bar{D}$	0	1	1	1

→ loops



Expression: $X = \bar{A}\bar{C}D + \bar{A}BC + AC\bar{D} + A\bar{B}\bar{C}$

Solution number 2:



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	1
$\bar{C}D$	1	1	0	1
CD	0	1	0	0
$C\bar{D}$	0	1	1	1

Expression: $X = \bar{A}BD + BC\bar{D} + \bar{B}\bar{C}D + A\bar{B}\bar{D}$

Example 8: Use k-map method to simplify the following expression:

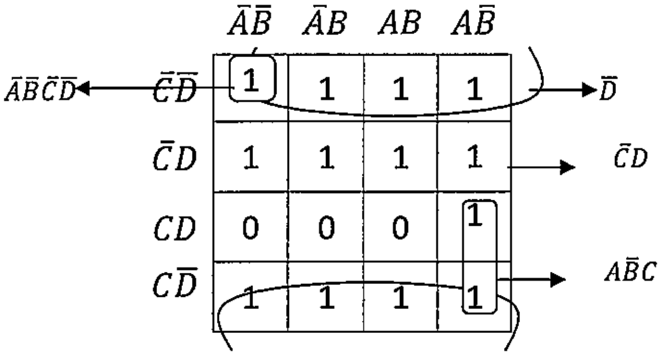
$$\boldsymbol{Y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}}$$

Answer:

First put the expression in an SOP form:

$$\boldsymbol{Y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}}$$

Then fill the k-map with the 1's corresponding to the product terms

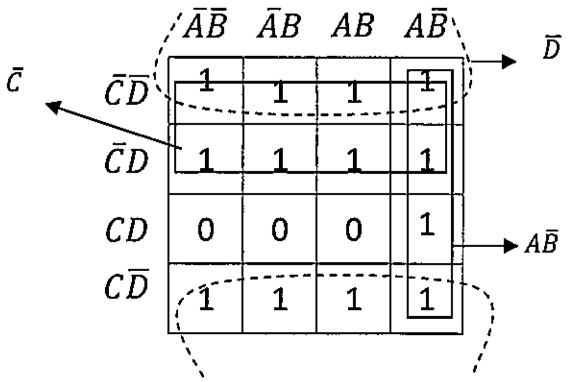


	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	1	1
$\bar{C}D$	1	1	1	1
CD	0	0	0	1
$C\bar{D}$	1	1	1	1

Now apply k-map rules:

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	1	1
$\bar{C}D$	1	1	1	1
CD	0	0	0	1
$C\bar{D}$	1	1	1	1

→ looping



Simplified expression: $Y = A\bar{B} + \bar{C} + \bar{D}$

<u>'Don't care' Condition:</u>

Sometimes the design of a logic circuit doesn't care if the output is a logic 1 or 0 at certain input values. This condition is called a "don't care" condition, and you will see an x for the output in the truth table instead of 1 or 0.

For example, consider the following 3-input system:

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

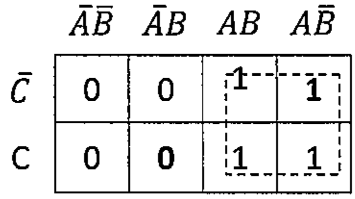
Don't care

When you use k-maps with systems having don't care conditions, change the x's in the map to either 1 or 0 to help you get less terms.

First draw the k=map, then decide the best values for the x's, and finally make your loops and write the expression for the output X:

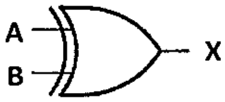
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	0	1	x
C	0	x	1	1

Best solution ⇨



Expression ⇨ X = A

Exclusive – OR (XOR)

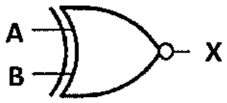


Boolean Expression : $X = A \oplus B$
$= A\bar{B} + \bar{A}B$

Truth table:

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

Exclusive – NOR (XNOR)



Boolean Expression : $X = \overline{A \oplus B}$
$= AB + \overline{A}\,\overline{B}$

Truth table:

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

Example 9: Simplify the following expression using XNOR: $Z = ABCD + A\bar{B}\bar{C}D + \bar{A}\bar{D}$

Answer:

$Z = AD\ (\ BC + \overline{B}\overline{C}\ ) + \overline{A}\overline{D} = AD(\ \overline{B \oplus C}) + \overline{A}\overline{D}$

Example 10: Simplify the following expression: $X = A + (A \oplus B)$

Answer:

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

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

$= A + B$

Exercises

1) Simplify the following equations:

a) $X = \mathrm{ABC} + \overline{\mathrm{A}}\mathrm{C}$

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

c) $Z = ABC + A\bar{B}C + \bar{A}$

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

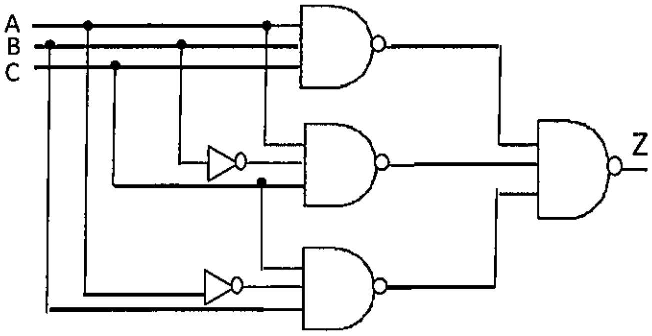
e) $Y = \bar{A}\bar{B}\bar{C} + \bar{A}BC + ABC + A\bar{B}\bar{C} + A\bar{B}C$

f) $Z = AB\left(\overline{\bar{C}D}\right) + \bar{A}BD + \bar{B}\bar{C}\bar{D}$

2) For the circuit shown:

a) Write the Boolean expression for the circuit shown

b) Simplify the expression



3) For the truth table shown below:

a) Write the expression of the output X using SOP terms

b) Simplify the expression

c) Draw the circuit

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

4) Determine the expression for each K map shown below:

a)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	1	1
$\bar{C}D$	1	1	0	0
CD	0	0	0	1
$C\bar{D}$	0	0	1	1

b)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	0	1	1
$\bar{C}D$	1	0	0	1
CD	0	0	0	0
$C\bar{D}$	1	0	1	1

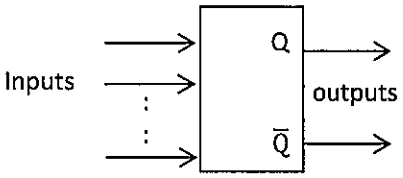
c)

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

Chapter 5
Flip Flops

Introduction:

1. A flip flop (FF) is a logic component that can have one or more inputs but has two outputs. One of the outputs is the inverse of the other output (Q and $\overline{Q}$).
2. When we talk about the state of the flip flop, we are normally referring to the output Q.For example, when we say the flip flop is in the high state, we mean that Q=1.
3. The output of the flip flop depends on previous input values



4. The flip flop has the following output states:
   a) Q=1, $\overline{Q}$=0 : here, we call the flip flop is "set"
   b) Q=0, $\overline{Q}$=1 : here, we call the flip flop is "Reset" or "clear"

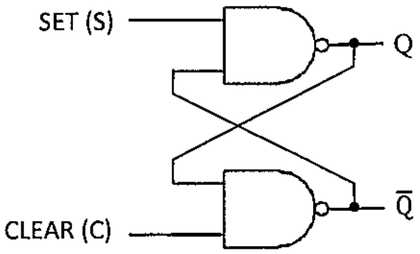
In this chapter we will discuss three types of flip flops:

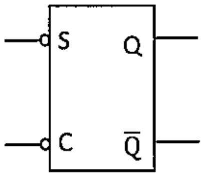
1. S-C flip flops
2. J-K flip flops
3. D flip flops

1- S-C flip flop (latch):

We can build an S-C flip flop using one of two methods. We can build it using NAND gates or by using NOR gates

a) Using NAND gates:





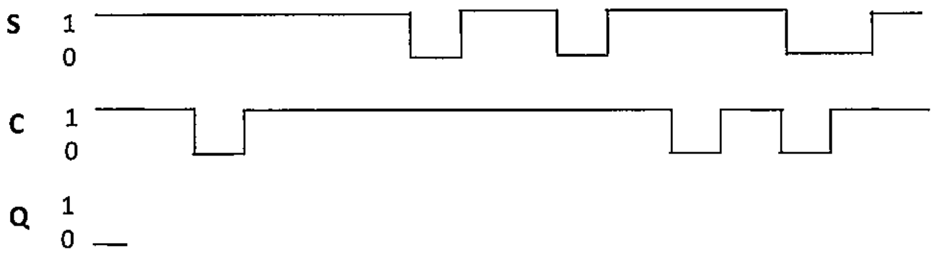
SET means it sets Q to high (1), and CLEAR means it clears Q to low (0).

Operation:

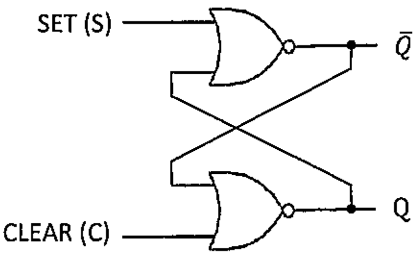
- Normally S&C inputs are both high.
- Q will respond to the input that goes low. If S goes low, then Q will be set tohigh, but when C goes low then Q goes to low.
- It's not allowed to have both S & C go low at the same time. We call this state invalid.
- The truth table of the NAND SC latch is as follows:

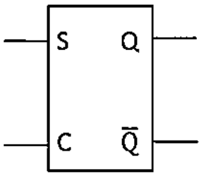
S	C	Q
1	1	no change
0	1	1 "set"
1	0	0 "clear"
0	0	invalid

Exercise 1: If the inputs of an S-C NAND FF are shown in the timing diagram below, draw the expected Q output.



b) Using NOR gates:



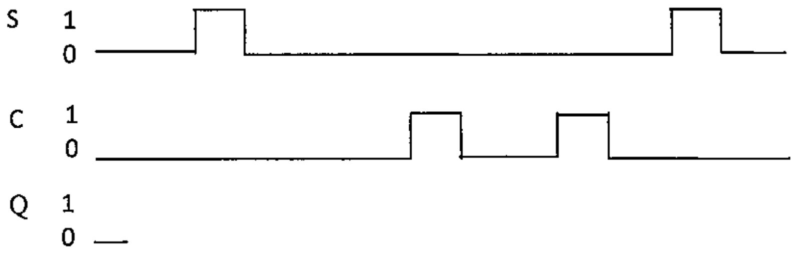


Operation:

- Normally S&C inputs are both low.
- Q will respond to the input that goes high. If S goes high, then Q will be set to high, but when C goes high then Q goes to low.
- It's not allowed to have both S & C go high at the same time. This is an invalid state.
- The truth table of the NOR SC latch is as follows:

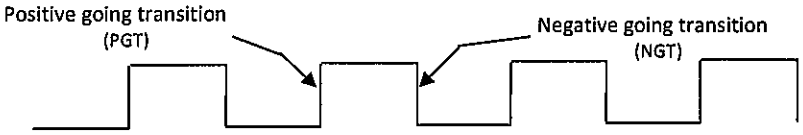
S	C	Q
0	0	no change
1	0	1 "set"
0	1	0 "clear"
1	1	invalid

Exercise 2: If the inputs of an S-C NOR FF are shown in the timing diagram below, draw the expected Q output.

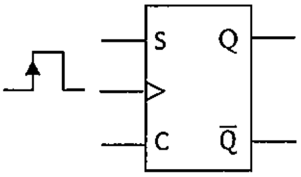


Clocked S-C Flip Flops:

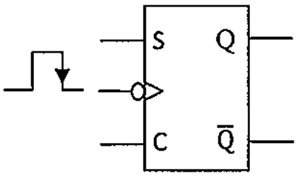
- A clocked flip flop has an additional input called a clock.
- The clock signal takes the shape of a square wave.
- The purpose of the clock is to control the action of a flip flop.
- This type of flip flops are called synchronous systems



Clock signal



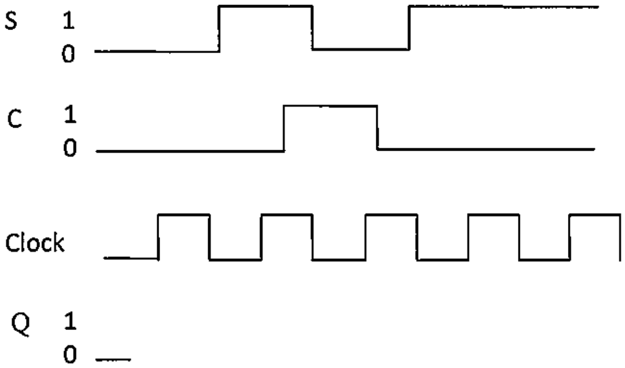
flip flop activated
by a PGT

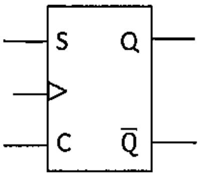


flip flop activated
by an NGT

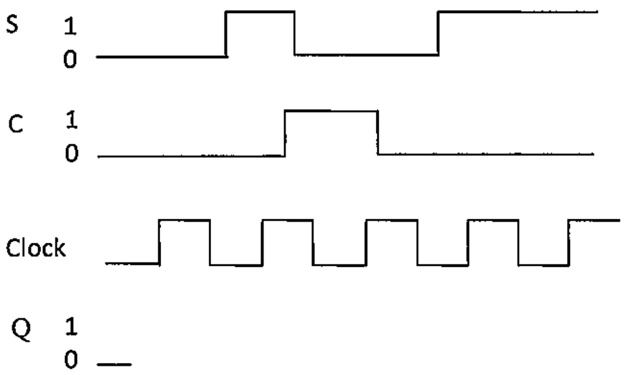
- A clocked flip flop will not read the input values S & C until the clock gives a PGT or an NGT signals. Otherwise, Q doesn't change values.
- A clocked SC flip flop triggered by a PGT will change states only when the clock signal goes from 0 to 1
- A clocked SC flip flop triggered by an NGT will change states only when the clock signal goes from 1 to 0

Exercise 3: **Draw the waveform of Q for a NOR S-C flip flop with PGT clock**



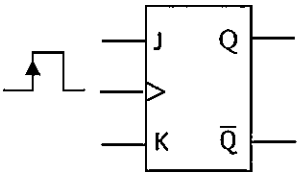


Exercise 4: **Draw the waveform of Q for a NOR S-C flip flop with NGT clock**

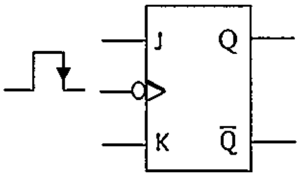


2- Clocked J-K flip flop:

- J-K flip flop is like a clocked NOR S-C flip flop where J=S and K=C.
- J-K flip flop allows both J and K inputs to go high at the same time.
- When J=K= 1, the output Q changes from its current state to the opposite state. That is, if Q=1 then it will become 0 and if Q=0 it will become 1. We call this state "toggle".
- The output Q responds to the inputs J&K only when there is a PGT or an NGT clock signals.



J-K flip flop
activated by a PGT

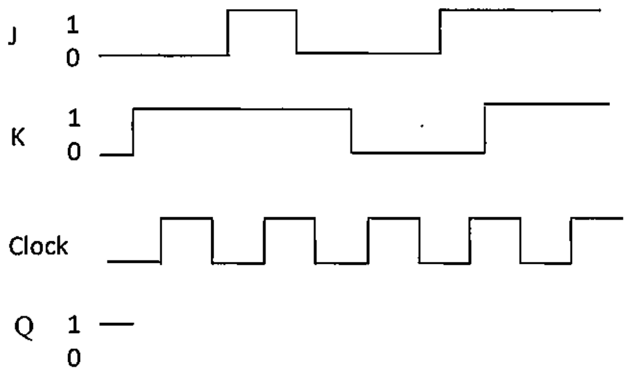


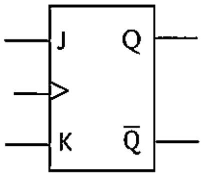
J-K flip flop
activated by an NGT

- The truth table for a J-K flip flop is as follows:

J	K	Q
0	0	no change
1	0	1 "set"
0	1	0 "clear"
1	1	Toggle

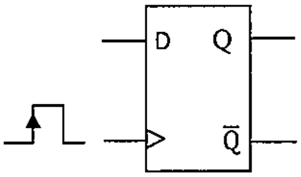
Exercise 5: **Draw the waveform of Q for a J-K flip flop with PGT clock**



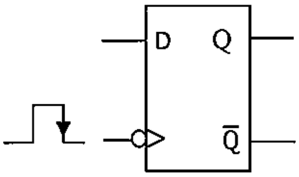


3- Clocked D flip flop:

- D flip flop stands for data flip flop
- It has one input: D. The output is triggered (controlled) by either a PGT or an NGT clock depending on the D flip flop
- The output Q will equal the input D in value every time the flip flop gets a clock signal. When there is no clock signal, the output does not change values.



D flip flop
activated by a PGT



D flip flop
activated by an NGT

- The truth table for a clocked D flip flop is as follows:

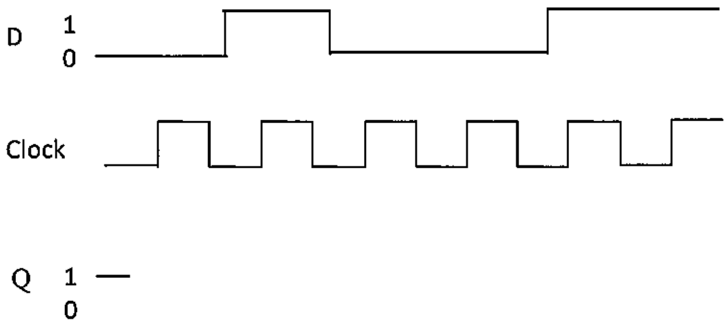
PGT clock

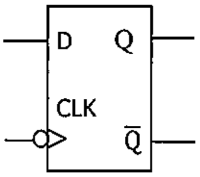
D	clk	Q
0	↑	0
1	↑	1
0	no PGT	no change
1	no PGT	no change

NGT clock

D	clk	Q
0	↓	0
1	↓	1
0	no NGT	no change
1	no NGT	no change

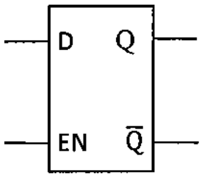
Exercise 6: **Draw the waveform of Q for the flip flop shown**





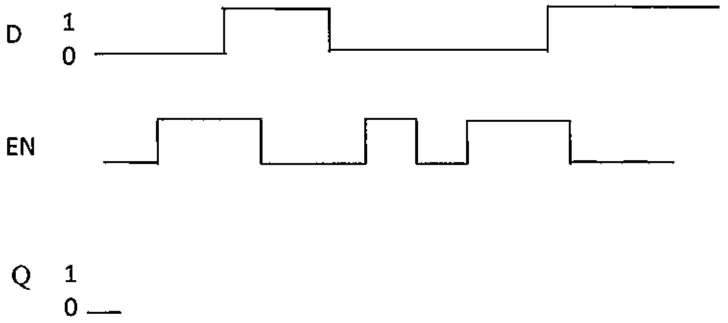
4- D-Latch:

- D-latch has two inputs: D and Enable (EN). The output is controlled by the Enable input.
- As long as the Enable input is high, the output Q will follow the input D in value. When the Enable goes low, the output freezes and does not change values.
- The symbol and truth table of a D-latch are as follows:



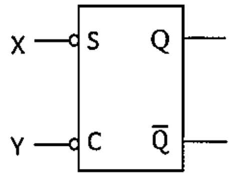
D	EN	Q
0	1	0
1	1	1
0	0	no change
1	0	no change

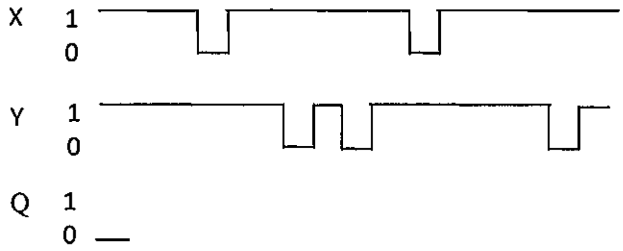
Exercise 7: **Draw the waveform of Q for the D-latch flip flop**



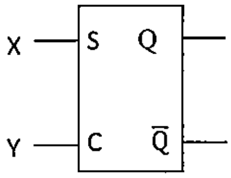
Exercises

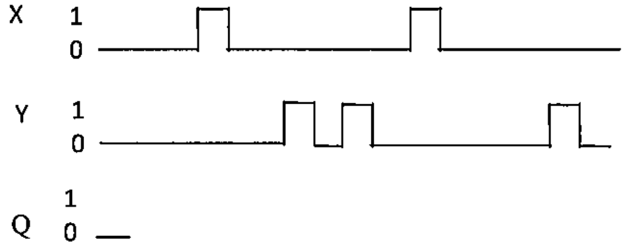
1) For the flip flop shown determine the Q waveform.



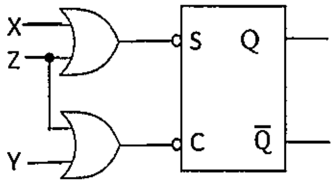


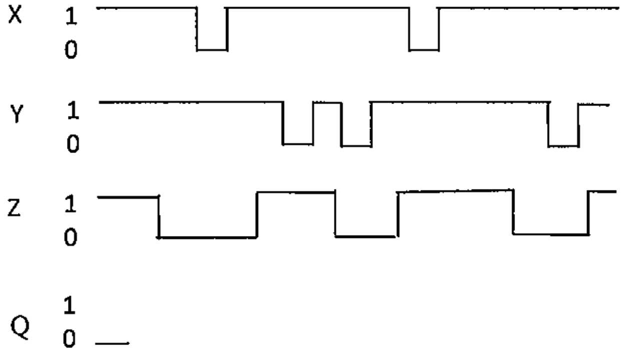
2) For the flip flop shown determine the Q waveform.



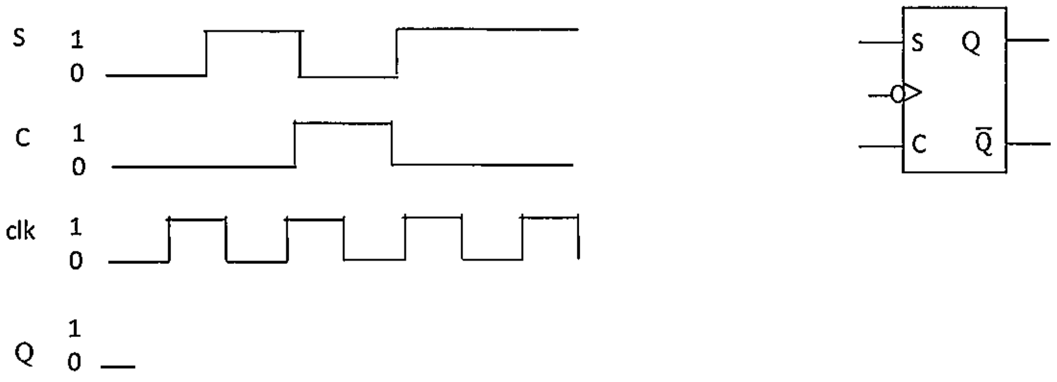


3) For the circuit shown determine the Q waveform.

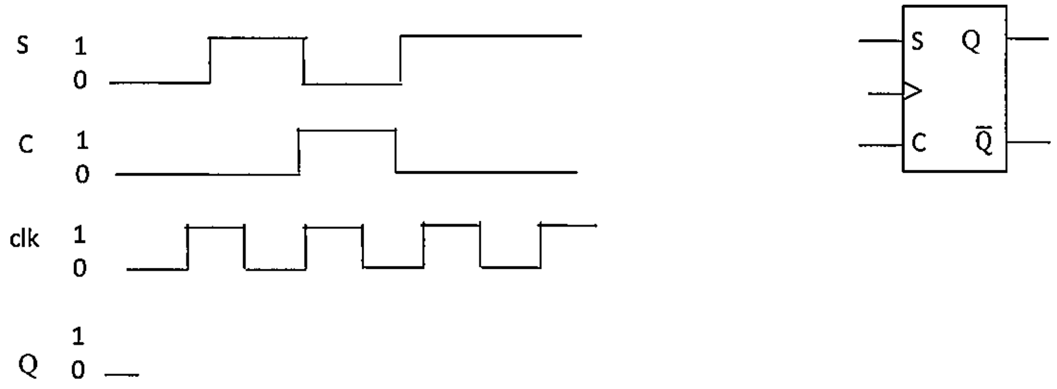




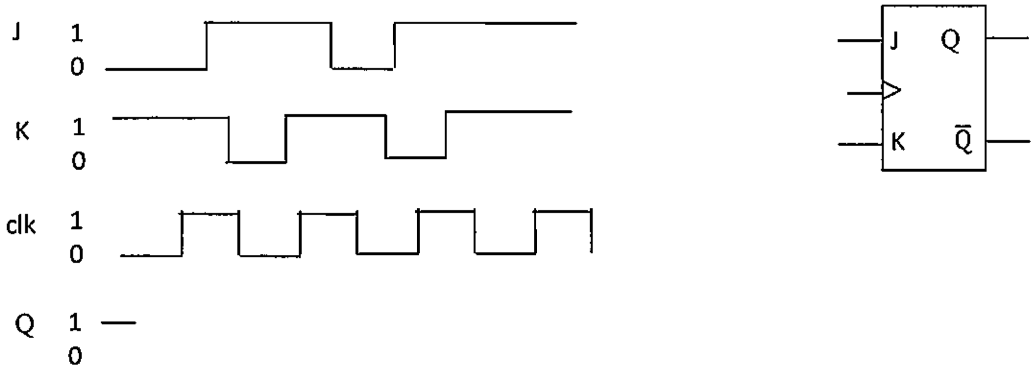
4) Determine the waveform of the output Q using the circuit shown



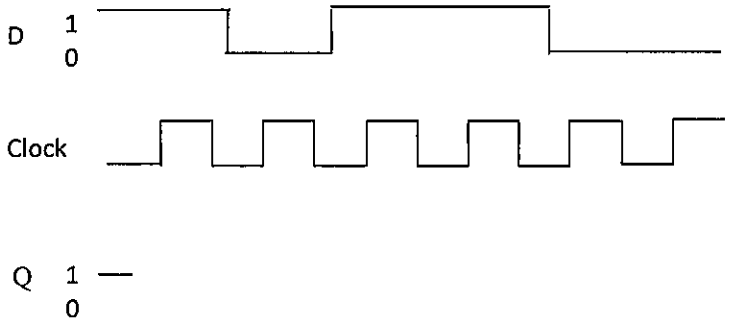
5) Determine the waveform of the output Q using the circuit shown

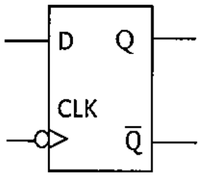


6) Determine the waveform of the output Q using the circuit shown

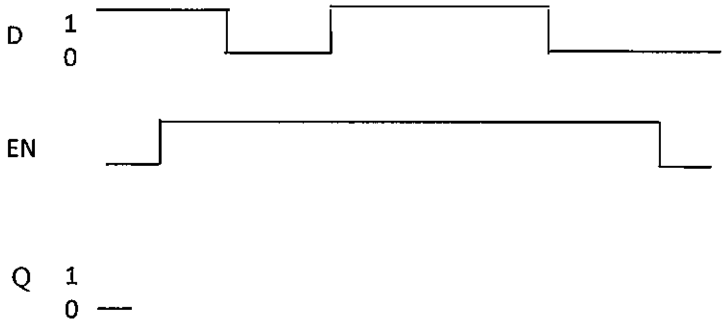


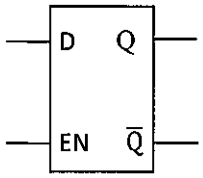
7) Determine the waveform of the output Q using the circuit shown





8) Determine the waveform of the output Q using the circuit shown





Chapter 6
Digital Arithmetic

Binary Addition:

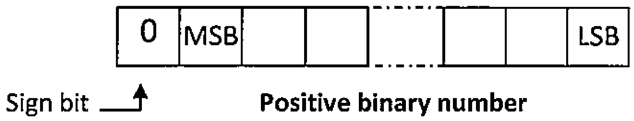
0 + 0 = 0
0 + 1 = 1
1 + 1 = 0 + a carry of 1 into the next position
1 + 1 + 1 = 1 + a carry of 1 into the next position

Example 1: Perform the following additions

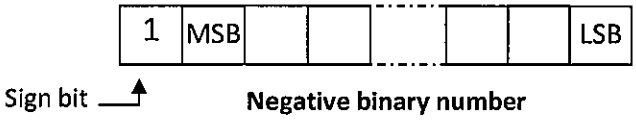
```
A)                      B)                        C)

 1 1     <--carry        1 1 1 1   <--carry        1 1 1 1       <-- carry
  011                     1001                      11.011
+ 110                   + 1111                    + 10.110
 1001                    11000                     110.001
```

Signed Binary Numbers:

A signed binary number has an additional bit to indicate the sign. The sign bit is located right before the MSB.



Positive binary number



Negative binary number

If the sign bit is 0 then the binary number is positive

If the sign bit is 1 then the binary number is negative

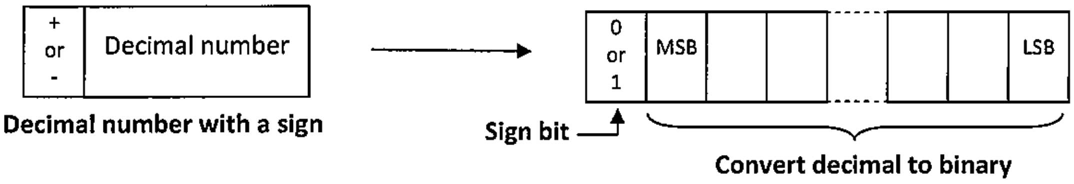
There are two methods used for representing binary numbers with a sign. These are:

1) Sign-magnitude

2) 2's complement

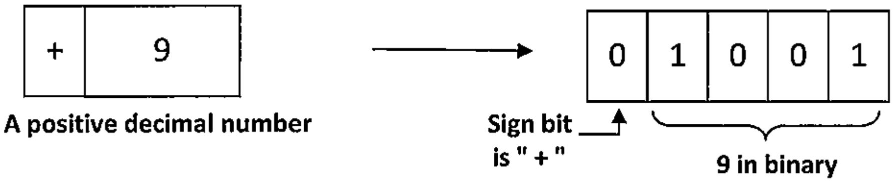
1) Sign-Magnitude System:

To convert from a signed decimal number to a sign-magnitude number, do the following:



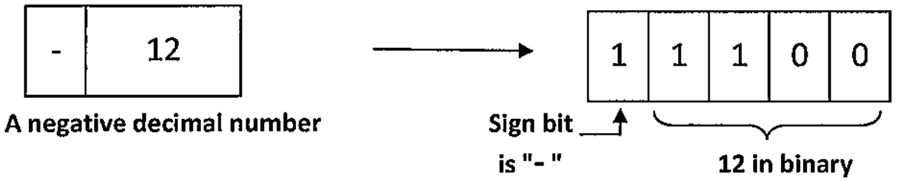
Example 2: Convert $(+9)_{10}$ to binary using the sign-magnitude system

Answer:



Example 3: Convert $(-12)_{10}$ to binary using the sign-magnitude system

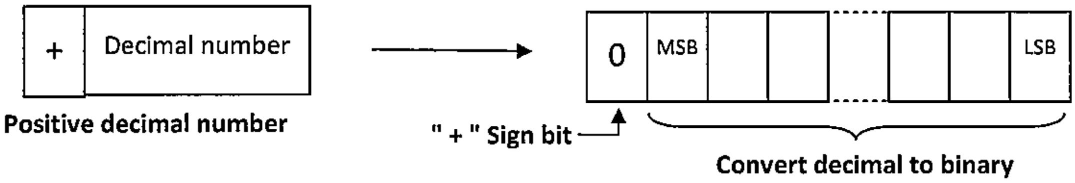
Answer:



2) 2's Complement System:

The conversion to binary in this system depends on the sign of the decimal number.

a) If the decimal number is positive, then use the same method as in the sign-magnitude system:



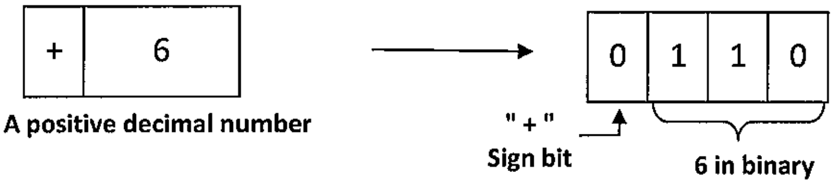
b) If the decimal number is negative, then do the following steps:

1. Convert the decimal number to binary and add the sign bit ' 1 '.
2. Perform the 2's complement transformation:
   a) Change all the bits (without the sign bit) to their opposite values (the 0 becomes 1 and the 1 becomes 0). This process is called the <u>1's complement</u>.
   b) Perform the addition of 1 to the 1's complement number.

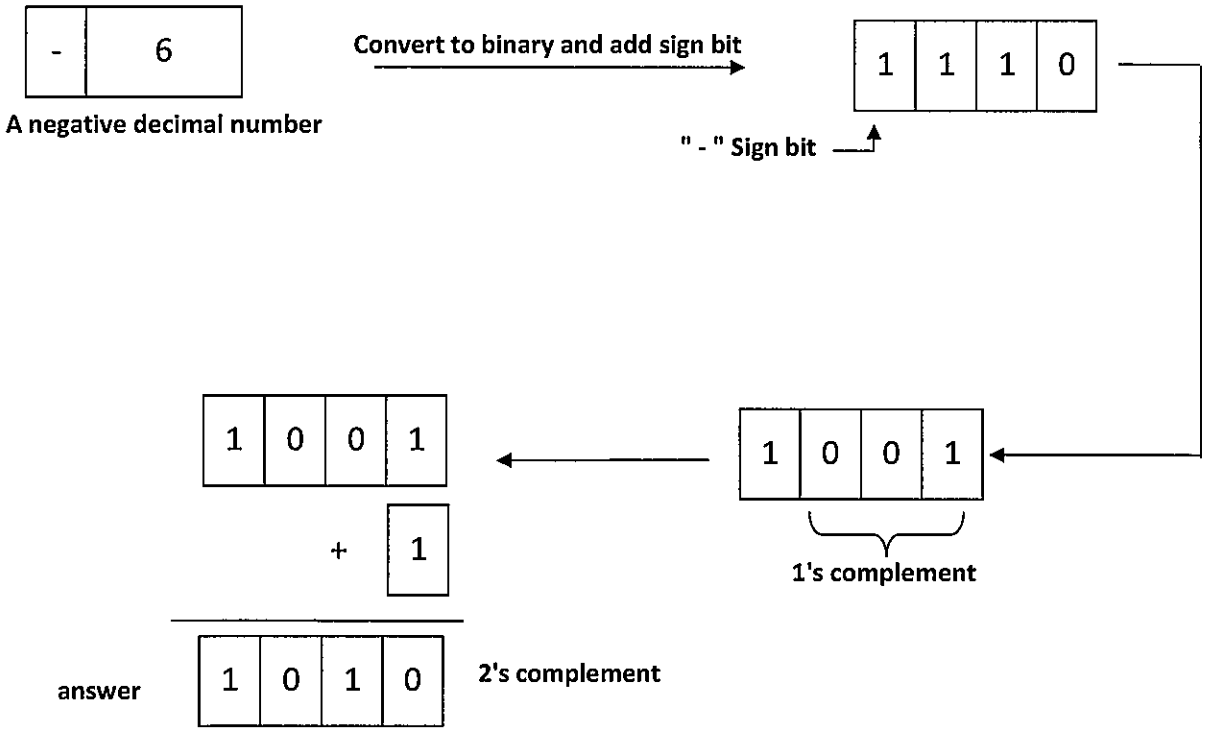
Example 4: Use 4 bits to convert $(+6)_{10}$ and $(-6)_{10}$ to signed binary using the 2's complement system

Answer:

a) $(+6)_{10}$

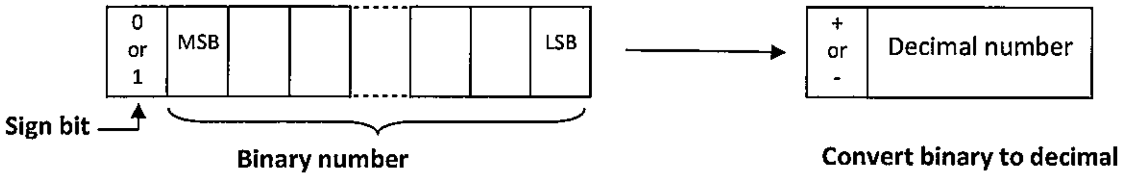


b) $(-6)_{10}$



How to convert a signed binary to a signed decimal:

1) Using Sign-Magnitude System:



Example 5: Convert the following sign-magnitude numbers to decimal

a- $(1\ 1\ 1\ 0\ 0\ 1)_2$

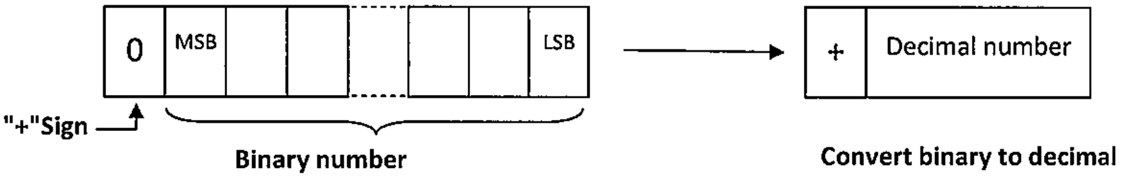
b- $(0\ 0\ 1\ 1\ 1\ 0)_2$

Answer:

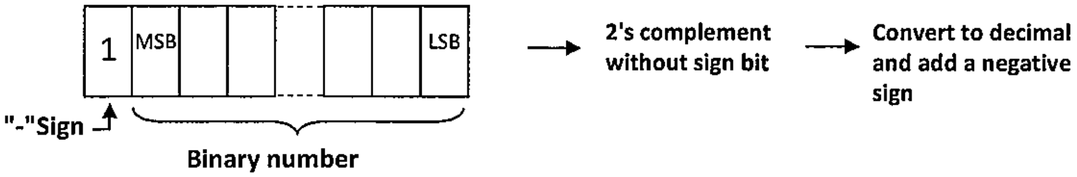
a- $(\underset{-}{\underset{\downarrow}{1}}\ \underbrace{1\,1\,0\,0\,1}_{25})_2$ b- $(\underset{+}{\underset{\downarrow}{0}}\ \underbrace{0\,1\,1\,1\,0}_{14})_2$

2) Using 2's Complement System:

a) If the binary number is positive, do the following:



b) If a negative binary, do the following:

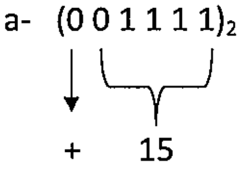

1
MSB
LSB
"-"Sign
Binary number
2's complement without sign bit
Convert to decimal and add a negative sign

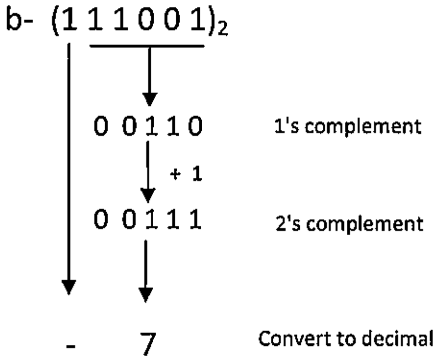
Example 6: Convert the following 2's complement numbers to decimal

a- $(0\ 0\ 1\ 1\ 1\ 1)_2$

b- $(1\ 1\ 1\ 0\ 0\ 1)_2$

Answer:





Exercise 1: Convert each of the following decimal numbers to a signed binary number in the 2's complement system (use a total of 5 bits including the sign bit):

a) +13 b) -9 c) +3 d) -2

Exercise 2: Determine the decimal values of the following signed binary numbers in the 2's complement system:

a) 01100 b) 11010 c) 10001

<u>Negation:</u> is the process of converting a signed binary number to the opposite sign (from positive to negative or from negative to positive). To do so, just perform the 2's complement to the whole number including the sign bit.

Example 7: Negate the number +9 (use 5 bits).

Answer

```
Convert to binary and add sign ------>  01001
 1's complement                -----> 10110
 2's complement                -----> 10111    (-9)
```

<u>Addition of signed binary numbers:</u>

In the 2's complement system, when you add two numbers, the sign bit is part of the number and is added to the sign bit of the other number. Discard any carry beyond the sign bit.

Examples:

a) Adding 2 positive numbers

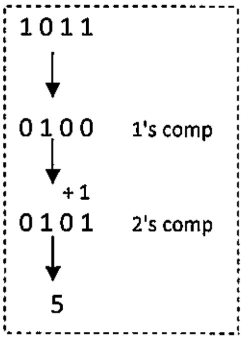
```
+9          0 1 0 0 1
+4          0 0 1 0 0
            ----------
+13         0 1 1 0 1
```

b) Adding positive and a smaller negative

```
+9          0 1 0 0 1
-4          1 1 1 0 0     (2's complement of 4)
            ----------
+5        1 0 0 1 0 1
         /
    discard
```

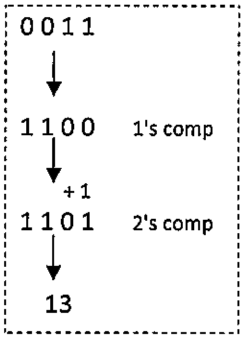
c) Adding positive and a larger negative

```
-9          1 0 1 1 1     (2's complement of 9)
+4          0 0 1 0 0
            ----------
-5          1 1 0 1 1  ------------------------->
```



d) Adding 2 negative numbers

```
-9            10111    (2's complement of 9)
-4            11100    (2's complement of 4)
              -----
-13        1  10011
discard
```



e) Adding equal and opposite numbers

```
+9            01001
-9            10111    (2's complement of 9)
              -----
0          1  00000
discard
```

Overflow: Occurs when two positive or two negative numbers are added together, where the resultant sign bit is incorrect.

Example 8: add +9 to +8 (use 5 bits).

Answer

```
+9            01001
+8            01000
              -----
wrong         10001    overflow
sign
```

The problem is that there are not enough bits in the number. To correct the error, just add one bit to each signed binary (total 6 bits).

```
+9            001001
+8            001000
              ------
+17           010001     correct answer
```

Subtraction of Signed binary numbers:

To subtract a number B from a number A, then negate B and add it to A.

Example 9: Subtract the number +4 from the number +9 in binary (use 5 bits)

Answer:

+9 = 0 1 0 0 1

+4 = 0 0 1 0 0

Now, negate +4 : $0\,0\,1\,0\,0 \xrightarrow[\text{1's comp}]{} 1\,1\,0\,1\,1 \xrightarrow[\text{2's comp}]{} 1\,1\,1\,0\,0$ (-4)

Next add +9 to -4:

```
+9          0 1 0 0 1
-4          1 1 1 0 0
            ----------
 +5        1  0 0 1 0 1
         ↗
   discard
```

Example 10: Subtract the number +9 from the number -4 in binary (use 5 bits)

Answer:

+9 = 0 1 0 0 1

-4 = 1 1 1 0 0 (2's complement of 4)

Now, negate +9 : $0\,1\,0\,0\,1 \xrightarrow[\text{1's comp}]{} 1\,0\,1\,1\,0 \xrightarrow[\text{2's comp}]{} 1\,0\,1\,1\,1$ (-9)

Next add -4 to -9:

```
- 4          1 1 1 0 0
- 9          1 0 1 1 1
             ----------
- 13        1 1 0 0 1 1 ----------------> 0 0 1 1
          ↗                                  ↓
    discard                                1 1 0 0   1's comp
                                             ↓ + 1
                                           1 1 0 1   2's comp
                                             ↓
                                            13
```

BCD Addition:

- Each decimal digit = 4 bits BCD code.
- The largest BCD code is 1001, which is 9 in decimal.
- When you add two BCD numbers, take them 4 bits at a time. Start with the first 4 bits from the left from both numbers to be added.
- If the sum of two 4 bits is greater than 1001 or results in a carry to the next position, then correct the answer by adding 0110 to it. Any carry is moved to the next position.

Example 11: Add $(45)_{10}$ to $(33)_{10}$ in BCD

Answer:

$$\begin{array}{rl} 45_{10} & 0100\ 0101 \\ + & \\ 33_{10} & 0011\ 0011 \\ \hline 78_{10} & 0111\ 1000 \end{array}$$

Example 12: Add $(47)_{10}$ to $(35)_{10}$ in BCD

Answer:

$$\begin{array}{rrl} 47_{10} & 0100\ 0111 & \\ + & & \\ 35_{10} & 0011\ 0101 & \\ \hline & 0111\ 1100 & \text{More than 1001} \\ & +0110 & \\ \hline \mathbf{82_{10}} & \mathbf{1000\ 0010} & \end{array}$$

Hex Addition:

- Add hex numbers just like you add decimal numbers starting from the LSD
- If the sum of two hex digits is greater than F, then subtract 16 from the sum and record it and carry 1 to the next position
- Discard any overflow

Example 13: Add $(58)_{16}$ to $(24)_{16}$

Answer:

```
      5 8
   +
      2 4
-----------------
      7 C
```

Example 14: Add $(58)_{16}$ to $(4B)_{16}$

Answer:

```
      5 8
   +
      4 C
-----------------
      A 4  →  Because 8 + C = 20 > 16
              Subtract 16 from 20 = 4
              And add a carry to the 5 and
              add it to the 4 = A
```

Hex Subtraction:

If X and Y are Hex numbers and you need to calculate X – Y, then do the following:

1. Convert Y to binary
2. Find the 2's complement of Y, we will call the answer Z
3. Convert Z back to Hex
4. Now perform X + Z (follow the rules of Hex addition). Discard any overflow.

Example 15: subtract $(3A5)_{16}$ from $(592)_{16}$

Answer:

```
                          2's complement
      3A5 = 0011  1010  0101 -----------> 1100  0101  1011  = C5B
```

Now do the addition:

```
          5 9 2
      +
          C 5B
      -----------------
Discard  → 1   1ED
overflow
```

(you can prove the answer by adding 1ED to 3A5)

Exercises

1) Add the following numbers in binary:

   a) 1010 + 1011
   b) 1111 + 0011
   c) 1011.1101 + 11.1
   d) 0.1011 + 0.1111

2) Represent each of the following signed decimal numbers in the 2's complement system (use 8 bits total including sign):

   a) +32
   b) -14
   c) +63
   d) +127
   e) -55
   f) -3

3) Find the decimal values of the following signed 2's complement numbers:

   a) 01101
   b) 11101
   c) 01111011
   d) 10011001
   e) 01111111
   f) 10000001
   g) 11111111

4) Find the 2's complement value of each of the following decimal numbers then negate it (use 8 bits total):

   a) +73
   b) -12
   c) +15
   d) -1
   e) -128

5) Perform the following operations in the 2's complement system (use 8 bits):

   a) Add +9 to +6
   b) Add +14 to -17
   c) Add -48 to -80

d) Subtract +21 from +31
e) Subtract +21 from -13
f) Subtract +47 from +47

6) Convert the following decimal numbers to BCD and then do addition:

a) 74 + 23
b) 147 + 380
c) 623 + 599

7) Add the following Hex numbers:

a) 3E91 + 2F93
b) ABC + DEF
c) 2FFE + 0002
d) D191 + AAAB

8) Subtract the following Hex numbers:

a) 3E91 – 2F93
b) 91B – 6F2
c) 0300 – 005A
d) F000 – EFFF