1 Introductry Digital Concepts

Electronic circuits can be divided into two broad categories, digital and analog. Digital electronics involves quantities with discrete values, and analog electronics involves quantities with continuous values.

An analog quantities is one having continuous values. A digital quantity is one having a discrete set of values. Most things that can be measured quantitatively appear in nature in analog form. For example, the air temperature changes over a continuous range of values. During a given day, the temperature does not go from, say, 70° to 71 ° instantaneously; it takes on all the infinite values in between. If you graphed the temperature on a typical summer day, you would have a smooth, continuous curve similar to the curve in Fig.(1-1). Other examples of analog quantities are time, pressure, distance, and sound.

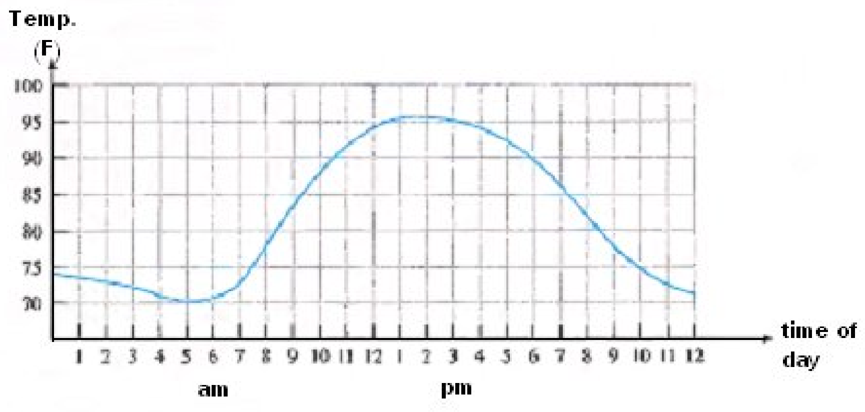


Fig.(1-1) Graph of an analog quantity.

Suppose, you just take a temperature reading every hour. Now, you have sampled values representing the temperature at discrete points in time (every hour) over a 24-hour period, as indicated in Fig.(1-2). You have effectively converted an analog quantity to a form that can now be digitized by representing each sample value by a digital code.

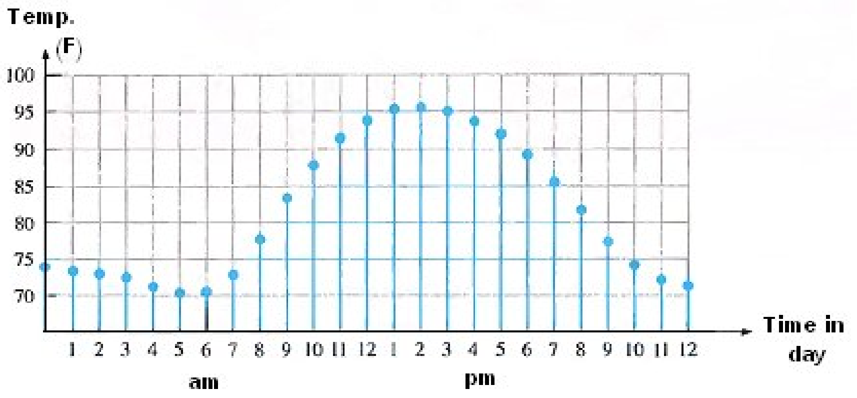


Fig.(1-2) Sampled-value representation of the analog quantity in Fig.(1-1).

<u>The Digital Advantages</u>

- Digital data can be processed and transmitted more efficiently and reliably than analog data.
- Digital data has a great advantage when storage is necessary. For example, music when converted to digital form can be stored more compactly and reproduced with greater accuracy and clarity than is possible when it is an analog form.
- Noise (unwanted voltage fluctuations) does not affect digital data nearly as much as it does analog signals.
- Digital systems are used in communication, business transaction, traffic control, space guidance, medical treatment, weather monitoring, the

internet, and many other commercial, industrial, and scientific enterprises.

We have digital telephones, digital TV’s, digital versatile discs, digital cameras……..

An Analog Electronic System

A pubic address system, used to amplify sound so that it can be heard by a large audience.

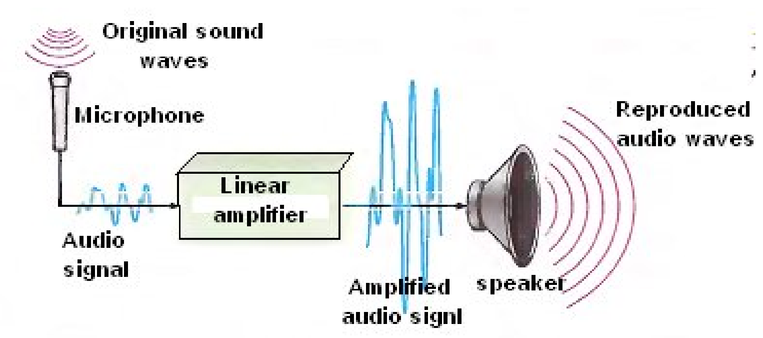


Fig.(1-3) A basic audio public address system.

Digital / Analog Electronic System

The compact disc (CD) player is an example of a system in which both digital and analog circuits are used. Music in digital form is stored on the compact disc. A laser diode optical system picks up the digital data from the rotating disc and transfers it to the (DAC) digital-to-analog converter. The output analog signal is amplified and sent to the speaker.

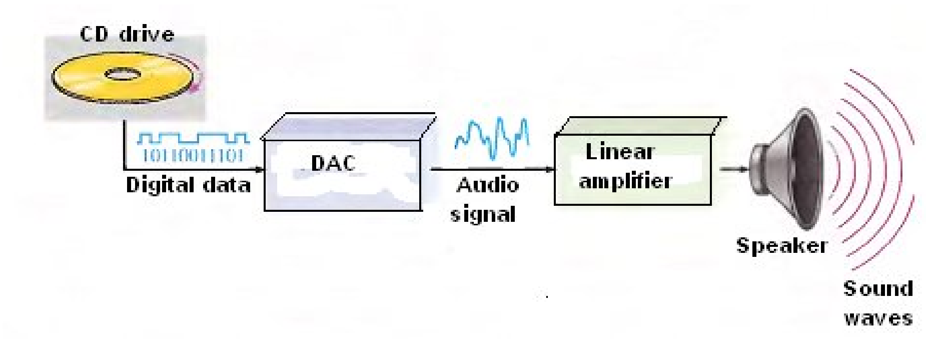


Fig.(1-4) Basic principle of a CD player.

Binary Digits

The two digits in the binary system, 1 & 0 are called bits, which is a contraction of the words binary digit.

In digital circuits, two different voltage levels are used to represent the two bits. A 1 is represented by the higher voltage (HIGH) and a 0 is represented by the lower voltage level (LOW). This is called positive logic.

HIGH = 1 : LOW = 0

Codes: groups of bits (combinations of 1s and 0s). Codes are used to represent numbers, letters, symbols, instructions and anything else required in a given application.

Logic Levels

The voltages used to represent a 1 and a 0 are called logic levels. Ideally, one voltage level represents a 1 and another voltage level represents a 0. In practical digital circuit, a HIGH can be any voltage between a specified minimum value and a specified maximum value. Likewise, a LOW can be any voltage between a specified minimum and a specified maximum. There can be no overlap between the accepted HIGH levels and the accepted LOW levels.

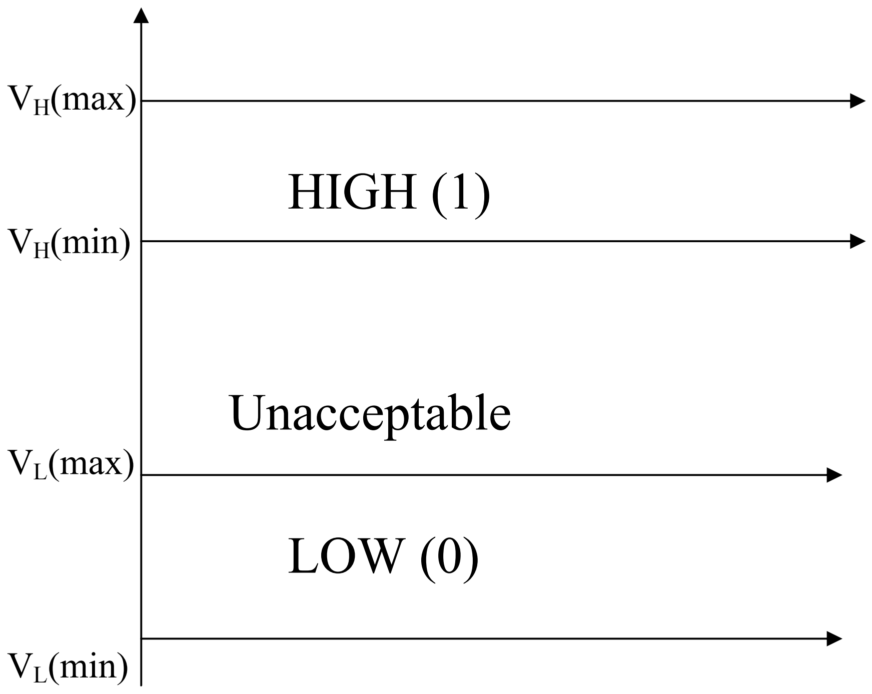


Fig.(1-5) Logic level ranges of voltage for a digital circuit.

Digital Waveforms

1. The positive-going pulse: is generated when the voltage (or current) goes from its normally LOW level to its HIGH level and then back to its LOW level.
2. The negative-going pulse is generated when the voltage goes from its normally HIGH level to its LOW level and back to its HIGH level.

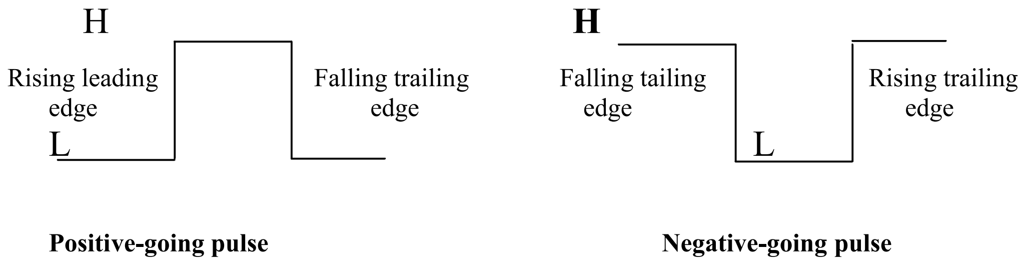


Fig.(1-6) Ideal pulses

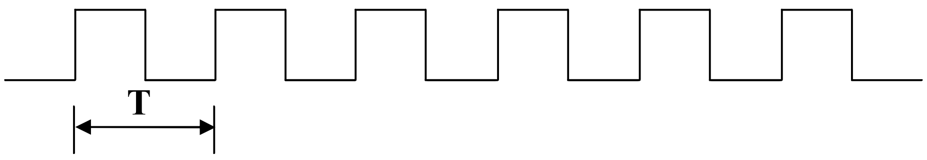
Q *What about nonideal pulse?*

Q *What is your information about rise time, fall time and pulse width?*

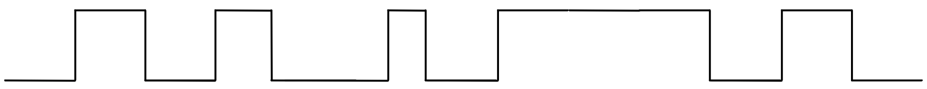
Pulse trains: series of pulses.

Periodic pulse waveform: is one that repeats itself at a fixed interval, called a period (T). Frequency (F) = 1/T , measured in (Hz), when (T) in (sec).

Nonperiodic pulse waveform does not repeat itself at fixed intervals and may be composed of pulses of randomly differing pulse width and/or randomly differing time intervals between the pulses.



(a)



(b)

Fig.(1-7) Pulse waveforms.

An important characteristic of a periodic digital waveform is its duty cycle. The duty cycle is defined as the ratio of the pulse width (t_w) to the period (T):

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

Timing Diagrams

A timing Diagram is a graph of digital waveforms showing the actual time relationship of two or more waveforms and how each waveforms and how each waveform changes in relation to the others.

The clock: In digital systems, all waveforms are synchronized with a base timing waveform called the clock. It is a periodic waveform in which each interval between pulses (the period) equals the time for one bit. The clock waveform itself does not carry information.

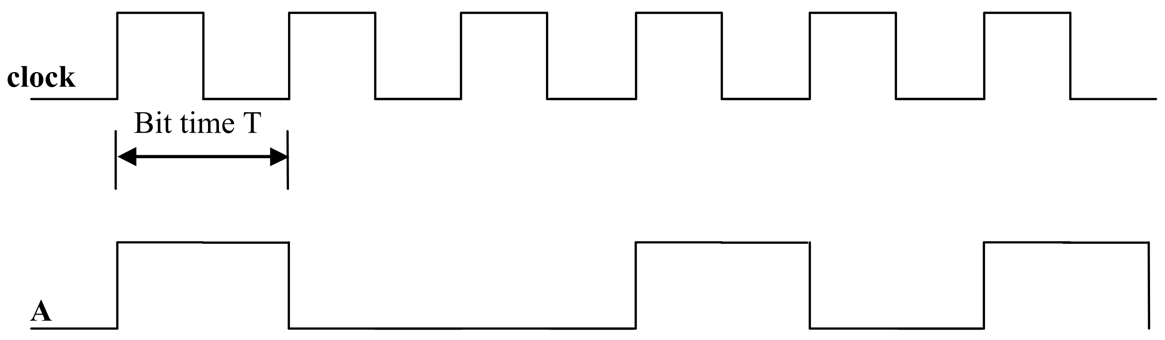


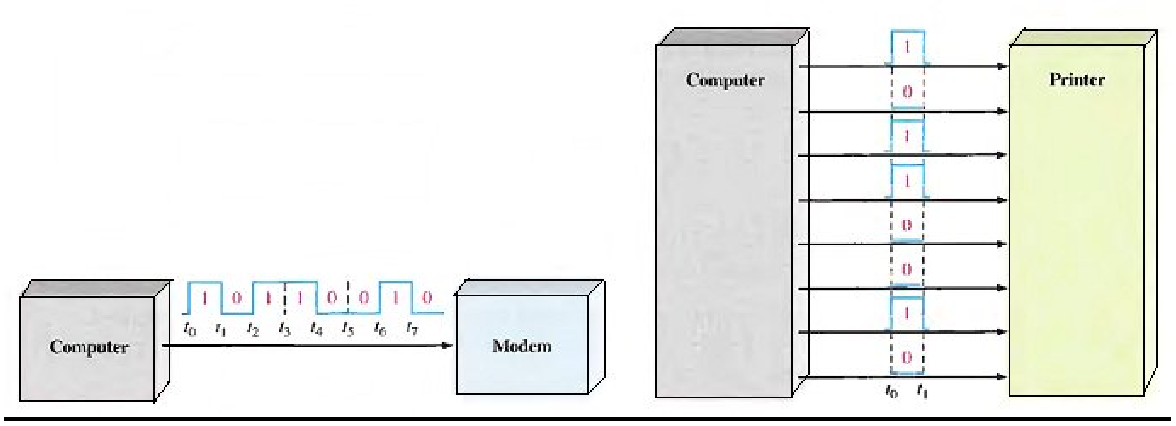
Fig.(1-8) A simple timing diagram.

Data Transfer

1. Serial.
2. Parallel.

When bits are transferred in serial from one point to another, they are sent one bit at a time along a single conductor. To transfer eight bits in series, it takes eight time intervals.

When bits are transferred in parallel form, all the bits in a group are sent out on separate lines at the same time. There is one line for each bit.



(a) Serial transfer. **(b) Parallel transfer**

Fig.(1-9) Illustration of serial and parallel transfer of binary data. Only the data lines are shown.

<u>Basic Logic Operations</u>

Gorge Boole (1850s)

Three basic logic operations are indicated by standard distinctive symbols in Fig.(1-6) below:

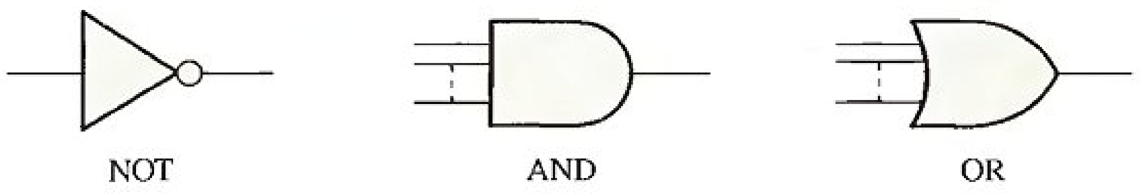


Fig.(1-10) Basic logic operations and symbols.

Logic gate is a circuit that performs a specified logic operation (AND, OR).

HIGH (true) : LOW (false)

NOT changes one logic level to the opposite level.

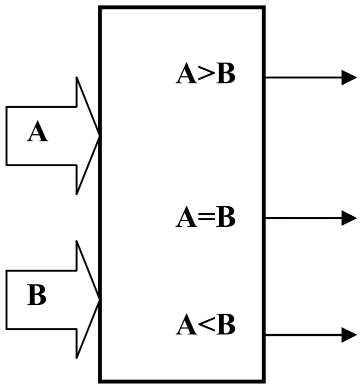
AND produces a HIGH output only if all the inputs are HIGH.

OR produces a HIGH output when any of the inputs is HIGH.

Basic Logic Functions

The three basic elements AND, OR, and NOT are combined to form more complex logic circuits that perform many useful operations and that are used to build complete digital systems.

1. The arithmetic functions:
    - Addition
    - Subtraction
    - Multiplication
    - Division
2. The comparison function:



3. The code conversion function: A code is a set of bits arranged in a unique pattern and used to represent specified information.

    $$\text{Binary} \longrightarrow \text{BCD}$$

    $$\text{Binary} \longrightarrow \text{Gray code}$$

4. The encoding function (Encoder): converts information e.g. decimal number into some coded form, like binary code.

5. The decoding function (Decoder): converts coded information into a noncoded form.

6. The data selection function:

   Multiplexer (MUX): is a logic circuit that switches digital data from several input lines onto a single output line in a specified time sequence.

   Demultiplexer (DEMUX): is a logic circuit that switches digital data from one input line to several output lines in a specified time sequence.

7. The storage function

   Flip-Flop, Registers, Semiconductor memories, Magnetic disks, ets…

8. The counting function.

Digital Integrated Circuits

A monolithic integrated circuit (IC) is an electronic circuit that is constructed entirely on a single small chip of silicon.

IC packages are classified according to the way they are mounted on printed circuit (PC) boards:

- The through-hole type packages have pins (leads) that are inserted through holes in the PC board and can be soldered to conductors on the opposite side. (DIP) dual-in-line package.
- The surface-mount technology (SMT) newer, space-saving, holes are unnecessary. (SOIC) small-outline integrated circuit.

ICs are classified according to their complexity:

1- Small-scale integration (SSI) < 12 gate/chip. They include basic gates and flip-flops.

2- Medium-scale integration (MSI) 12 – 99 gate/chip. They include encoders, decoders, counters, registers, multiplexers, small memories.
3- Large-scale integration (LSI) 100 – 9999 gate/chip, including memories.
4- Very large-scale integration (VLSI) 10,000 – 99,999 gate/chip, including (memories and microprocessors).
5- Ultra large-scale integration (ULSI) >100,000. It describes very large memories, larger microprocessors, and large single-chip computers.

ICs are classified according to their operation into Analog IC & Digital IC or both in one.

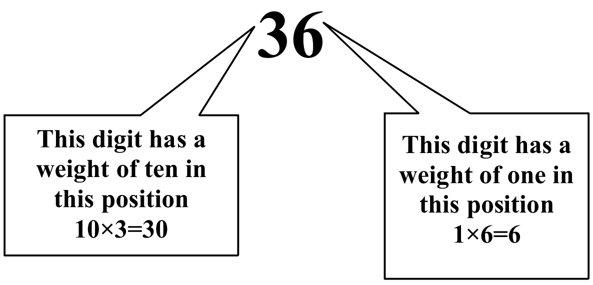
ICs are classified according to the type of transistors (BJT) bipolar junction transistor and (MOSFET) metal-oxide semiconductor field- effect transistors.

Finish

2 Number Systems

1- Decimal Numbers

In the decimal number system each of the ten digits 0 through 9, represents a certain quantity. These ten symbols (digits) don't limit you to expressing only ten different quantities, because you use the various digits in appropriate positions within a number to indicate the magnitude of the quantity.


36
This digit has a weight of ten in this position 10×3=30
This digit has a weight of one in this position 1×6=6

The position of each digit in a decimal number indicates the magnitude of the quantity represented, and can be assigned a weight. The decimal number system is said to be of base, or radix, 10 because it uses 10 digits and the coefficients are multiplied by powers of 10. In general, a number with a decimal point is represented by a series of coefficients:

$$\mathbf{a_4\ a_3\ a_2\ a_1\ a_0\ .\ a_{-1}\ a_{-2}\ a_{-3}}$$

The coefficients a_j are any of the 10 digits (0, 1, 2, ……,9), and the subscript value j gives the place value and, hence, the power of 10 by which the coefficient must be multiplied.

$$10^4\times\mathbf{a_4}+10^3\times\mathbf{a_3}+10^2\times\mathbf{a_2}+10^1\times\mathbf{a_1}+10^0\times\mathbf{a_0}.\ 10^{-1}\times\mathbf{a_{-1}}+10^{-2}\times\mathbf{a_{-2}}+10^{-3}\times\mathbf{a_{-3}}$$

2- Binary Numbers

This is another way to represent quantities. The binary system is less complicated than the decimal system because it has only two digits. It's a base-2 system.

A binary digit, called a bit, has two values 0 & 1. Each coefficient $\mathbf{a_j}$ **is multiplied by** $\mathbf{2^j}$**,** and the results are adde**d** to obtain the decimal equivalent of the number. For example,

11010.11 is equal to 26.75 as follows:

$$1\times\mathbf{2^4} + 1\times\mathbf{2^3} + 0\times\mathbf{2^2} + 1\times\mathbf{2^1} + 0\times\mathbf{2^0} + 1\times\mathbf{2^{-1}} + 1\times\mathbf{2^{-2}}$$

$$\mathbf{16 + 8 + 0 + 2 + 0 + 0.5 + 0.25 = (26.75)_{10}}$$

In general, a number expressed in a base-r system has coefficients multiplied by powers of r.

$$\boxed{r^n\times\mathbf{a_n} + r^{n-1}\times\mathbf{a_{n-1}} + \ldots + r^2\times\mathbf{a_2} + r^1\times\mathbf{a_1} + \mathbf{1}\times\mathbf{a_0} + \mathbf{r}^{-1}\times\mathbf{a_{-1}} + r^{-2}\times\mathbf{a_{-2}} + \ldots + r^{-m}\times\mathbf{a_{-m}}}$$

Now, let us begin to count in the binary system:

One bit	2 values	0, 1
Two bits	4 values	00, 01, 10, 11
Three bits	8 values	000, 001, 010, 011, 100, 101, 110, 111.
Four bits	16 values	0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.

And so on……..

Decimal number	Binary number 8421
0	0000 ← (LSB) least significant bit
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
	↑ (MSB) most significant bit

LSB (right-most bit) has a weight of $2^0 = 1$.

MSB (left- most bit) has a weight of $2^3 = 8$.

In general, with n – bit, you can count up to a number equal to:

Largest decimal number = $2^n - 1$

When n = 5, you can count from 0 – 31, (32 values). Max. number = $(31)_{10}$.

The weight structure of a fractional binary number is

2^{n-1}…..2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3}…… 2^{-n}

…... 8 4 2 1 . 0.5 0.25 0.125 ……..

3 – Hexadecimal Numbers

It has (16) digits and is used primarily as a compact way of displaying or writing binary numbers, it's very easy to convert between binary and hexadecimal numbers.

Decimal number	Binary number 8421	Hexadecimal number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

How do you count in hexadecimal once you get to F? Simply start over with another column and continue as follows:

10	11	12	13	14	15	16	17	18	19
1A	1B	1C	1D	1E	1F	20	21	22	23
24	25	26	27	28	29	2A	2B	2C	2D…….

With two hexadecimal digits, you can count up to FF which in decimal 255.

$(100)_{16} = (256)_{10}$

$(101)_{16} = (257)_{10}$

$(FFF)_{16} = (4095)_{10}$

$(FFFF)_{16} = (65535)_{10}$

4- Octal Numbers

This system provides a convenient way to express binary numbers and codes (like Hex. Number system), it is used less frequently than hexadecimal.

The octal number system is composed of eight digits, which are:

0	1	2	3	4	5	6	7

To count above 7, begin another column and start over

10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37……

$(15)_8 \quad = \quad (13)_{10} \quad = \quad (D)_{16}$

5- Binary Coded Decimal (BCD)

The 8421 code is a type of binary coded decimal. It means that each decimal digit 0 through 9 is represented by a binary code of four bits:

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

{1010, 1011, 1100, 1101, 1110, 1111} are invalid in 8421 BCD code.

10 = 0001 0000 : 32 = 0011 0010

6- The Gray Code

- is unweigthed.
- is not an arithmetic code.
- it exhibits only a single bit change from one code number to the next.

Decimal number	Binary number	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

7- ASCII

(The American Standard Code for Information Interchange) pronounced "askee".

- It has 128 characters and symbols represented by a 7-bit binary code.
- 32 ASCII characters are nongraphic never printed or displayed and used only for control purposes.
- Others are graphic symbols for printing and displaying.

8- Exess-3 Code

Is unweighted code in which each code is obtained from the corresponding binary value plus 3.

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.

Example

Convert the binary number 1101101 to decimal.

Solution

$$2^6 \times 1 + 2^5 \times 1 + 2^4 \times 0 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1$$

$$64 + 32 + 0 + 8 + 4 + 0 + 0 + 1 = (109)_{10}$$

Example

Convert 0.1011 to decimal.

Solution

$$2^0 \times 0 + 2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 1 + 2^{-4} \times 1$$

$$0 + 0.5 + 0.125 + 0 + 0.0625 = (0.6875)_{10}$$

Decimal – to – Binary Conversion

- Sum – of – Weights Method: determine the set of binary weights whose sum is equal to the decimal number.

Example: Convert 9 to binary.

Solution : 9= 8 + 1 "choosing numbers with power of 2 "

Binary weights are:	…..32	16	8	4	2	1
			1	0	0	1

So $(9)_{10} = (1001)_2$

Example

Convert the following

12 = 8 + 4 = 1100

25 = 16 + 8 + 1 = 11001

82 = 64 + 16 + 2 = 1010010.

Repeated Division -by- 2 Method:

12/2 = 6	reminder = 0	this is the LSB
6/2 = 3	reminder = 0	
3/2 = 1	reminder = 1	
1/2 = 0	reminder = 1	this is the MSB

Stop when the whole number quotient is 0.

Example: convert 45 to binary.

Solution:

45/2 = 22	reminder = 1
22/2 = 11	reminder = 0
11/2 = 5	reminder = 1
5/2 = 2	reminder = 1
2/2 = 1	reminder = 0
1/2 = 0	reminder = 1

$(45)_{10} = (101101)_2$

What about decimal numbers with fractions?

Binary weights . 0.5 0.25 0.125 0.0625

(0.625)10 = 0.5 + 0.125

= (0.101)2 (by using sum – of – weight method)

OR by using repeated MULTIPLECTION - by – 2 method

$0.625 \times 2 = 1.25$	1	this is the MSB
$0.25 \times 2 = 0.5$	0	
$0.5 \times 2 = 1.00$	1	this is the LSB

So $(0.625)_{10} = (0.101)_2$

MSB LSB

Binary to Hexadecimal Conversion

Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol.

Example

$$1100101001010111 = \underset{C}{1100}\ \underset{A}{1010}\ \underset{5}{0101}\ \underset{7}{0111}$$

Note: each group must be four bits.

Hexadecimal – to – Binary Conversion

Reverse the process and replace each hexadecimal symbol with the appropriate four bits.

Example

$$\begin{array}{ccccc} 10AF = & 1 & 0 & A & F \\ & \longleftrightarrow & \longleftrightarrow & \longleftrightarrow & \longleftrightarrow \\ & 0001 & 0000 & 1010 & 1111 \end{array}$$

So $(10AF)_{16} = (0001000010101111)_2$

Hexadecimal to Decimal Conversion

Repeated division of a decimal number by 16 will produce the equivalent hexadecimal number, formed by the reminders of the division. The firs reminder produced is the LSD. Each successive division by 16 yields a reminder that becomes a digit in the equivalent hexadecimal number.

Note : when a quotient has a fractional part, the fractional part is multiplied by the divisor to get the remainder.

Example

$(650)_{10} = (\ ?\ )_{16}$

Solution

650/16 = 40	reminder = 10	= A	this is the LSD
40/16 = 2	reminder = 8	= 8	
2/16 = 0	reminder = 2	= 2	this is the MSD

So $(650)_{10} = (28A)_{16}$

Octal – to – Decimal Conversion

$$\begin{aligned}(2374)_8 &= 2 \times 8^3 + 3 \times 8^2 + 7 \times 8^1 + 4 \times 8^0 \\ &= 1024 + 192 + 56 + 4 \\ &= (1276)_{10}\end{aligned}$$

Decimal – to – Octal Conversion

$(359)_{10} = (\ )_8$

359/8 = 44	reminder = 7	this is the LSD
44/8 = 5	reminder = 4	
5/8 = 0	reminder = 5	this is the MSD

So $(359)_{10} = (547)_8$

Octal – to – Binary Conversion

Replace each octal digit with the appropriate three bits.

Example

$(25)_8 = (010\ 101)_2$

$(7526)_8 =$ 7 5 2 6

111 101 010 110

$(7526)_8 = (111101010110)_2$

Binary – to – Octal Conversion

Start with the right – most group of three bits and moving from right to left, convert each 3-bit group to the equivalent octal digit.

Example

11010000100 = 011 010 000 100

3 2 0 4

$(11010000100)_2 = (3204)_8$

Binary – to – Gray Code Conversion

- The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
- Going from left to right, add each adjacent pair of binary code bits to get the next gray code bit.

For example, the conversion of the binary number 10110 to Gray code is as follows:

$$\begin{tikzcd}
1 \arrow[r, "+"] \arrow[d] & 0 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 0 \arrow[d] & \text{Binary} \\
1 & 1 & 1 & 0 & 1 & \text{Gray}
\end{tikzcd}$$

Gray – to – Binary Conversion

- The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
- Add each binary code bit generated to the Gray code bit in the next adjacent position.

For example, the conversion of the Gray code word 11011 to binary is as follows:

$$\begin{tikzcd}
1 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & \text{Gray} \\
1 \arrow[ru, "+"] & 0 \arrow[ru, "+"] & 0 \arrow[ru, "+"] & 1 \arrow[ru, "+"] & 0 & \text{Binary}
\end{tikzcd}$$

Binary Arithmatic

1- Binary Addition

0 + 0 = 0	with a carry = 0
0 + 1 = 1	with a carry = 0
1 + 0 = 1	with a carry = 0
1 + 1 = 0	with a carry = 1

Example

11 + 11 = 110 because

```
   11          3
+  11       +  3
  110          6


   110         6
+  100       + 4
  1010        10
```

2- Binary Subtraction

0 – 0 = 0	with a barrow = 0
0 – 1 = 1	with a barrow = 1
1 – 0 = 1	with a barrow = 0
1 – 1 = 0	with a barrow = 0

Example

```
   11          3
-  01        - 1
   10          2
```

3- Binary Multiplication

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

It involves forming partial products, shifting each successive partial product left one place and then adding all the partial products.

Examples

```
     11        3
    ×11       ×3
  _____________________
     11        9
   + 11
   ____
   1001


     111        7
    ×101       ×5
  _____________
     111       35
    000
  + 111
  ________
  100011
```

4-Binary Division

This operation follows the same procedure as division in decimal number system.

Example

110 ÷ 11 = 10   6 ÷ 3 = 2

```
        10
    ________
11 | 110
     -11
    ____
     000
```

```
      11
    _______
10 | 110
    - 10
      010
    -  10
       000
```

1’s and 2’s Complement of Binary Numbers

The l's complement and the 2's complement of a binary number are important because they permit the representation of negative numbers. The method of 2's complement arithmetic is commonly used in computers to handle negative numbers.

Finding the 1's Complement

The l's complement of a binary number is found by changing all 1s to 0s and all 0s to 1s, as illustrated below:

10110010	Binary number
↓↓↓↓↓↓↓↓	
01001101	1 's complement

The simplest way to obtain the l's complement of a binary number with a digital circuit is to use parallel inverters (NOT circuits), as shown in Fig.(2-1) for an 8-bit binary number.

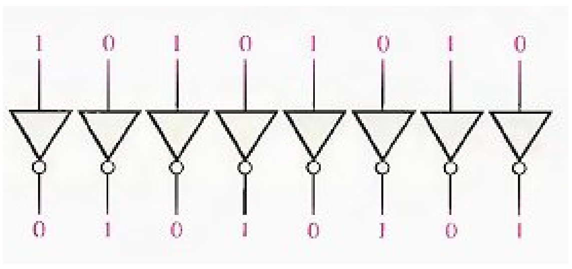


Fig.(2-1) Example of inverters used to obtain the 1 's complement of a binary number.

Finding the 2' s Complement

The 2's complement of a binary number is found by adding 1 to the LSB of the l's complement.

2's complement = (l's complement) + 1

Example

Find the 2's complement of 10110010.

Solution

```
    10110010        Binary number
    01001101        l's complement
 +         1        add 1
 -------------
    01001110        2's complement
```

An alternative method of finding the 2's complement of a binary number is as follows:

1. Start at the right with the LSB and write the bits as they are up to and including the first 1.
2. Take the 1's complements of the remaining bits.

Example:

Find the 2's complement of 10111000 using the alternative method.

Solution

```
    10111000        Binary number
    01001000        2's complement
```

The 2's complement of a negative binary number can be realized using inverters and an adder, as indicated in Fig.(2-2). This illustrates how an 8-bit number can be converted to its 2's complement by first inverting each bit (taking the l's complement) and then adding 1 to the l's complement with an adder circuit.

To convert from a l's or 2's complement back to the true (uncomplemented) binary form, use the same two procedures described previously. To go from the l's complement back to true binary, reverse all the bits. To go from the 2's complement form back to true binary, take the 1's complement of the 2's complement number and add 1 to the least significant bit.

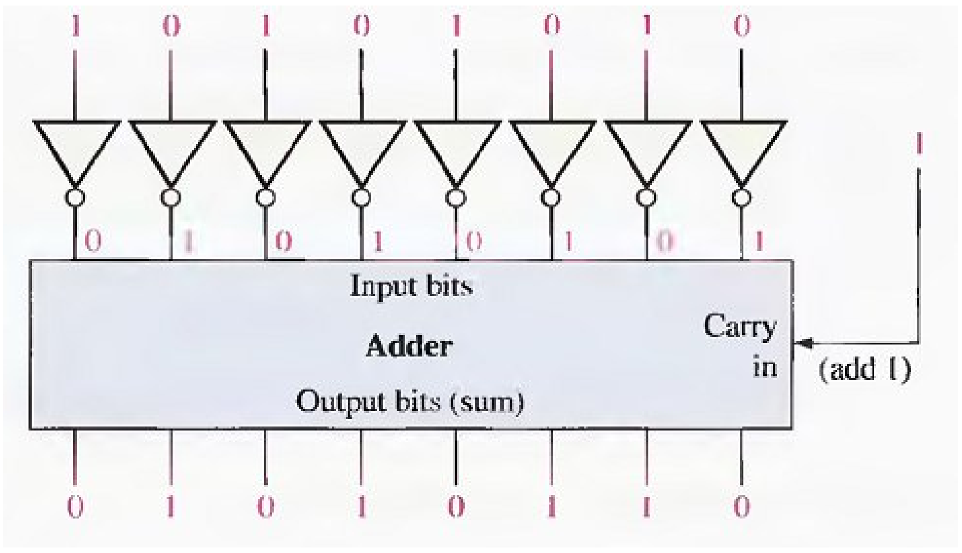


Fig.(2-2) Example of obtaining the 2's complement of a negative binary number.

Signed Numbers

Digital systems, such as the computer, must be able to handle both positive and negative numbers. A signed binary number consists of both sign and magnitude information. The sign indicates whether a number is positive or negative, and the magnitude is the value of the number. There are three forms in which signed integer (whole) numbers can be represented in binary: sign-magnitude, l's complement, and 2' complement. Of these, the 2's complement is the most important and the sign-magnitude is the least used.

The Sign Bit

The left-most bit in a signed binary number is the sign bit, which tells you whether the number is positive or negative. A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

Sign-Magnitude Form

When a signed binary number is represented in sign-magnitude, the left-most bit is the sign bit and the remaining bits are the magnitude bits. The magnitude bits are in true (uncomplemented) binary for both positive and negative numbers. For example, the decimal number + 25 is expressed as an 8-bit signed binary number using the sign-magnitude form as 00011001.
The decimal number - 25 is expressed as 1001100l. Notice that the only difference between + 25 and - 25 is the sign bit because the magnitude bits are in true binary for both positive and negative numbers.

In the sign-magnitude form, a negative number has the same magnitude bits as the corresponding positive number but the sign bit is a 1 rather than a zero.

l' s Complement Form

Positive numbers in 1's complement form are represented the same way as the positive sign-magnitude numbers. Negative numbers, however, are the l's complements of the corresponding positive numbers. For example. using eight bits. the decimal number -25 is expressed as the 1' s complement of + 25 (0001100 I) as 11100110.

In the l's complement form, a negative number is the l's complement of the corresponding positive number.

2 ' s Complement Form

Positive numbers in 2's complement form are represented the same way as in the sign-magnitude and l's complement forms. Negative numbers are the 2's complements of the corresponding positive numbers. Again, using eight bits, let's take decimal number -25 and express it as the 2's complement of +25 (00011001).

11100111

In the 2's complement form, a negative number is the 2's complement of the corresponding positive number.

Example:

Express the decimal number - 39 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

Solution

First, write the 8-bit number for + 39.

00100111

In the sign-magnitude form, - 39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the 1's complement form, -39 is produced by taking the l's complement of +39 (00100111).

11011000

In the 2's complement form, - 39 is produced by taking the 2's complement of +39 (00100111) as follows:

```
      11011000          1's complement
 +           1
 ---------------
      11011001          2's complement
```

The Decimal Value of Signed Numbers

Sign-magnitude: Decimal values of positive and negative numbers in the sign-magnitude form are determined by summing the weights in all the magnitude bit positions where there are 1s and ignoring those positions where there are zeros. The sign is determined by examination of the sign bit.

Example:

Determine the decimal value of this signed binary number expressed in sign-magnitude: 10010101

Solution

The seven magnitude bits and their powers-of-two weights are as follows:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

Summing the weights where there are 1s,

16 + 4 + 1 = 21

The sign bit is 1; therefore, the decimal number is - 21.

1's Complement: Decimal values of positive numbers in the l's complement form are determined by summing the weights in all bit positions where there are 1s and ignoring those positions where there are zeros. Decimal values of negative numbers are determined by assigning a negative value to the weight of the sign bit, summing all the weights where there are 1s, and adding 1 to the result.

Example:

Determine the decimal values of the signed binary numbers expressed in l's complement: (a) 00 010 111 (b) 11 101 000

Solution:

(a) The bits and their powers-of-two weights for the positive number are as follows:

$-2^7 \; 2^6 \; 2^5 \; 2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

0 0 0 1 0 1 1 1

Summing the weights where there are 1s, 16 + 4 + 2 + 1 = +23

(b) The bits and their powers-of-two weights for the negative number are as follows. $-2^7 \; 2^6 \; 2^5 \; 2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

1 1 1 0 1 0 0 0

Notice that the negative sign bit has a weight of -2^7 or -128.

Summing the weights where there are 1s, -128 + 64 + 32 + 8 = -24

Adding 1 to the result, the final decimal number is -24 + 1 = -23.

2's Complement: Decimal values of positive and negative numbers in the 2's complement form are determined by summing the weights in all bit positions where there are 1s and ignoring those positions where there are zeros. The weight of the sign bit in a negative number is given a negative value.

Example:

Determine the decimal values of the signed binary numbers expressed in 2's complement:

(a) 01010110 (b) 10101010

Solution:

(a) The bits and their powers-of-two weights for the positive number are as follows:

$^{-}2^7$	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

Summing the weights where there are 1s,

$64 + 16 + 4 + 2 = +86$

(b) The bits and their powers-of-two weights for the negative number are as follows. Notice that the negative sign bit has a weight of $-2^7 = -128$.

$^{-}2^7$	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

Summing the weights where there are 1s, $-128 + 32 + 8 + 2 = -86$

Range of Signed Integer Numbers That Can Be Represented

We have used 8-bit numbers for illustration because the 8-bit grouping is common in most computers and has been given the special name byte. With one byte or eight bits, you can represent 256 different numbers. With two bytes or sixteen bits, you can represent 65,536 different numbers. With four bytes or 32 bits, you can represent 4.295×10^9 different numbers. The formula for finding the number of different combinations of n bits is

$$\text{Total combinations} = 2^n$$

For 2's complement signed numbers, the range of values for n-bit numbers is

$$\text{Range} = -(2^{n-1}) \text{ to } + (2^{n-1} - 1)$$

where in each case there is one sign bit and (n – 1) magnitude bits. For example, with four bits you can represent numbers in 2's complement ranging from $-(2^3) = -8$ to $2^3 - 1 = +7$. Similarly, with eight bits you can go from -128 to + 127, with sixteen bits you can go from - 32,768 to + 32,767, and so on.

Finish

3 Logic Gates

THE INVERTER

The inverter (NOT circuit) performs the operation called inversion or complementation. The inverter changes one logic level to the opposite level. In terms of bits, it changes a 1 to a 0 and a 0 to a 1.
Standard logic symbols for the inverter are shown in Fig.(3-1), shows the distinctive shape symbols.

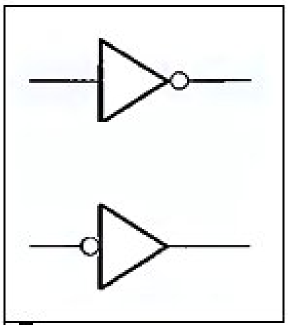


Fig.(3-1) Logic symbol for the inverter.

The negation indicator is a "bubble" (o) that indicates inversion or complementation when it appears on the input or output of any logic element. Generally, input is on the left of a logic symbol and the output is on the right. When appearing on the input, the bubble means that a 0 is the active, and the input is called an active-LOW input. When appearing on the output, the bubble means that a 0 is the active, and the output is called an active-LOW output.

When a HIGH level is applied to an inverter input, a LOW level will appear on its output. When a LOW level is applied to its input, a HIGH will appear on its output. This operation is summarized in Table 3-1, which shows the output for each possible input in terms of levels and corresponding bits. A table such as this is called a truth table.

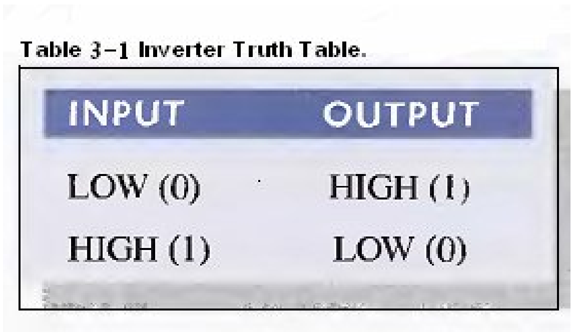


Table 3-1 Inverter Truth Table.

INPUT	OUTPUT
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

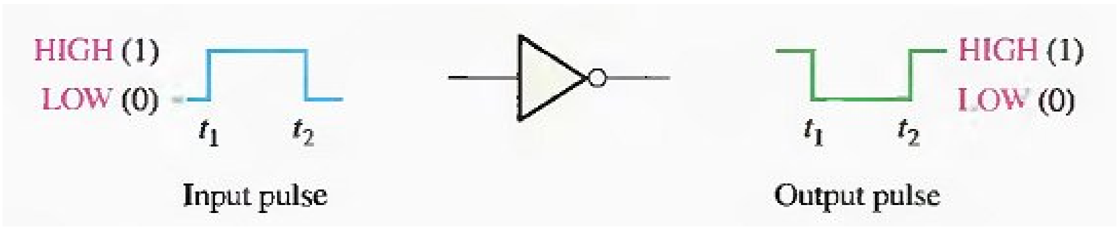


Fig.(3-2) Inverter operation.

The operation of an inverter (NOT circuit) can be expressed as follows: If the input variable is called A and the output variable is called X, then

$$X = \overline{A}$$

This expression states that the output is the complement of the input, so if A = 0, then X = 1, and if A = 1, then X = 0.

<u>The AND Gate</u>

The term gate is used to describe a circuit that performs a basic logic operation. The AND gate is composed of two or more inputs and a single output, as indicated by the standard logic symbols shown in Fig.(3-3). Inputs are on the left, and the output is on the right in each symbol. Gates with two inputs are shown; however, an AND gate can have any number of inputs greater than one.

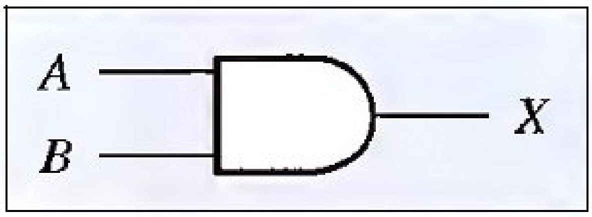


Fig.(3-3) AND gate symbol.

<u>Operation of an AND Gate</u>

An AND gate produces a HIGH output only when all of the inputs are HIGH. When any of the inputs is LOW, the output is LOW. Therefore, the basic purpose of an AND gate is to determine when certain conditions are simultaneously true, as indicated by HIGH levels on all of its inputs, and to produce a HIGH on its output to indicate that all these conditions are true. The inputs of the 2-input AND gate in Figure 3-8 are labeled A and B, and the output is labeled X. The gate operation can be stated as follows:

For a 2-input AND gate, output X is HIGH only when inputs A and B are HIGH; X is LOW when either A or B is LOW, or when both A and B are LOW.

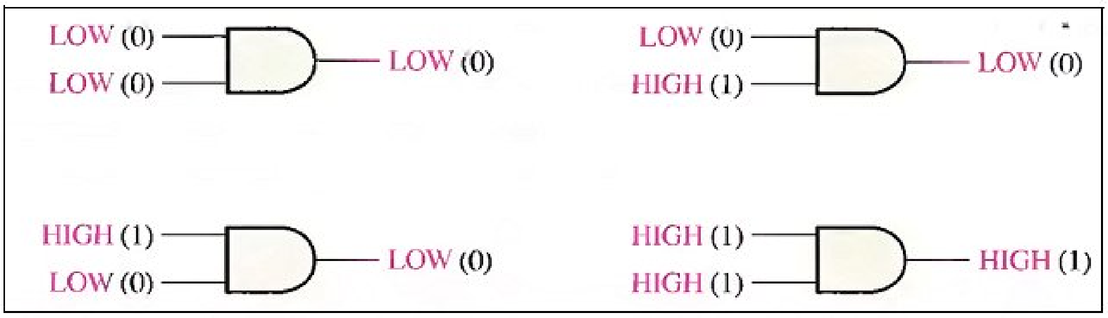
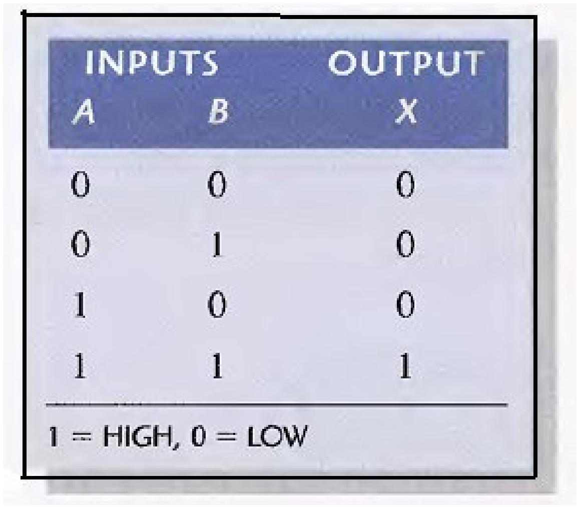


Fig.(3-4) All possible logic levels for a 2-input AND gate.

The logical operation of a gate can be expressed with a truth table that lists all input combinations with the corresponding outputs, as illustrated in Table 3-2 for a 2-input AND gate. The truth table can be expanded to any number of inputs. For any AND gate, regardless of the number of inputs, the output is HIGH only when all inputs are HIGH.

Table 3-2 The 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

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

$$\mathbf{N= 2^n}$$

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

For two input variables: N = 2^2 = 4 combinations.
For three input variables: N = 2^3 = 8 combinations.
For four input variables: N = 2^4 = 16 combinations.

Q/ Develop the truth table for a 3-input AND gate.

Let's examine the waveform operation of an AND gate by looking at the inputs with respect to each other in order to determine the output level at any given time. In Fig.(3-10), inputs A and B are both HIGH (1) during the time interval, t_1 making output X HIGH (1) during this interval. During time interval t_2 input A is LOW (0) and input B is HIGH (1), so the output is LOW (0). During time interval t_3 , both inputs are HIGH (1), and therefore the output is HIGH (1). During time interval t_4 , input A is HIGH 0) and input B is LOW (0), resulting in a LOW (0) output. Finally, during time interval t_5 , input A is LOW (0), input B is LOW (0), and the output is therefore LOW (0). As you know, a diagram of input and output waveforms showing time relationships is called a timing diagram.

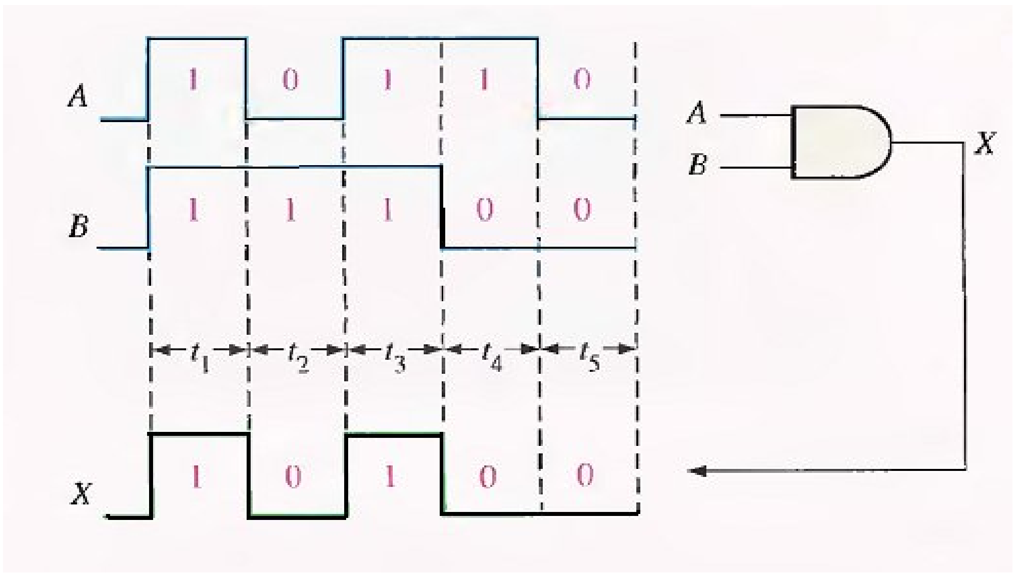


Fig.(3-5) Example of AND gate operation with a timing diagram showing input and output relationships.

Logic Expressions for an AND Gate

The logical AND function of two variables is represented mathematically either by placing a dot between the two variables, as A . B, or by simply writing the adjacent letters without the dot, as AB. We will normally use the latter notation because it is easier to write.

Boolean multiplication follows the same basic rules governing binary multiplication:

$$0 . 0 = 0$$
$$0 . 1 = 0$$
$$1 . 0 = 0$$
$$1 . 1 = 1$$

Boolean multiplication is the same as the AND function.

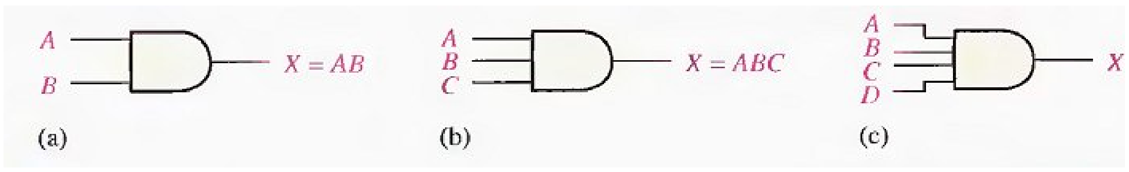


Fig.(3-6) Boolean expressions for AND gates with two, three, and four inputs.

The OR Gate

An OR gate can have more than two inputs. The OR gate is another of the basic gates from which all logic functions are constructed. An OR gate can have two or more inputs and performs what is known as logical addition.

An OR gate has two or more inputs and one output, as indicated by the standard logic symbol in Fig.(3-7), where OR gates with two inputs are illustrated. An OR gate can have any number of inputs greater than one.

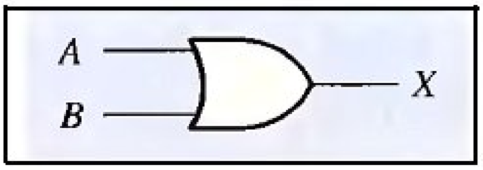


Fig.(3-7) Standard logic symbol for the OR gate.

Operation of an OR Gate

An OR gate produces a HIGH on the output when any of the inputs is HIGH. The output is LOW only when all of the inputs are LOW.

The inputs of the 2-input OR gate in Fig.(3-7) are labelled A and B. and the output is labelled X. The operation of the gate can be stated as follows:

For a 2-input OR gate, output X is HIGH when either input A or input B is HIGH, or when both A and B are HIGH; X is LOW only when both A and B are LOW.

The HIGH level is the active or asserted output level for the OR gate. Fig.(3-8) illustrates the operation for a 2-input OR gate for all four possible input combinations.

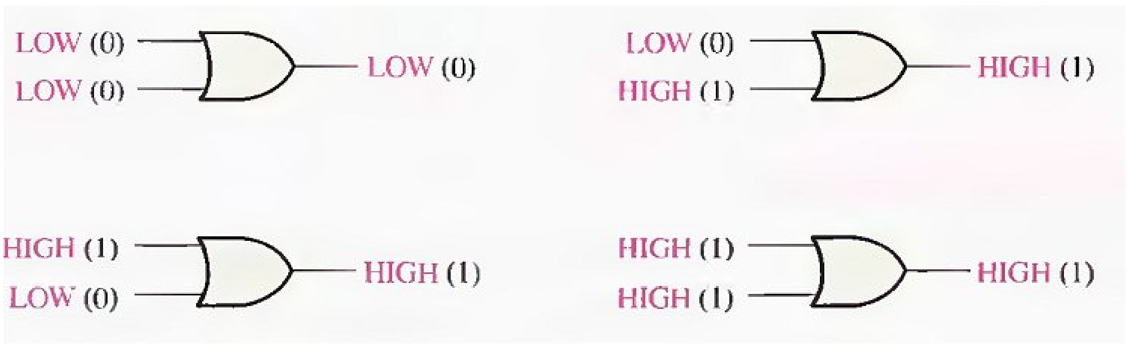
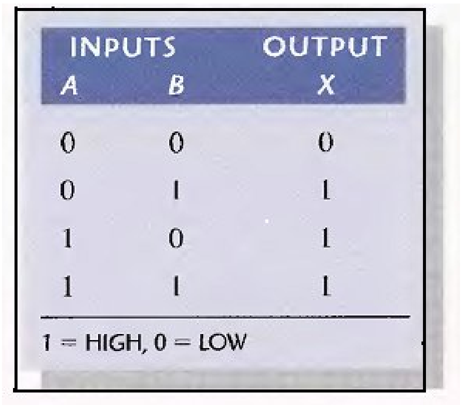


Fig.(3-8) All possible logic levels for a 2-input OR gate.

OR Gate Truth Table

The operation of a 2-input OR gate is described in Table 3-3. This truth table can be expanded for any number of inputs; but regardless of the number of inputs. the output is HIGH when one or more of the inputs are HIGH.

Table 3-3 The 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

Operation with Waveform Inputs

Now let's look at the operation of an OR gate with pulse waveform inputs, keeping in mind its logical operation. Again, the important thing in the analysis of gate operation with pulse waveforms is the time relationship of all the waveforms involved. For example, in Fig.(3-9), inputs A and B are both HIGH (1) during time interval t_1 making output X HIGH (1). During time interval t_2, input A is LOW (0), but because input B is HIGH (1), the output is HIGH (1). Both inputs are LOW (0) during time interval t_3 , so there is a LOW (0) output during this time. During time interval t_4 , the output is HIGH (1) because input A is HIGH (1).

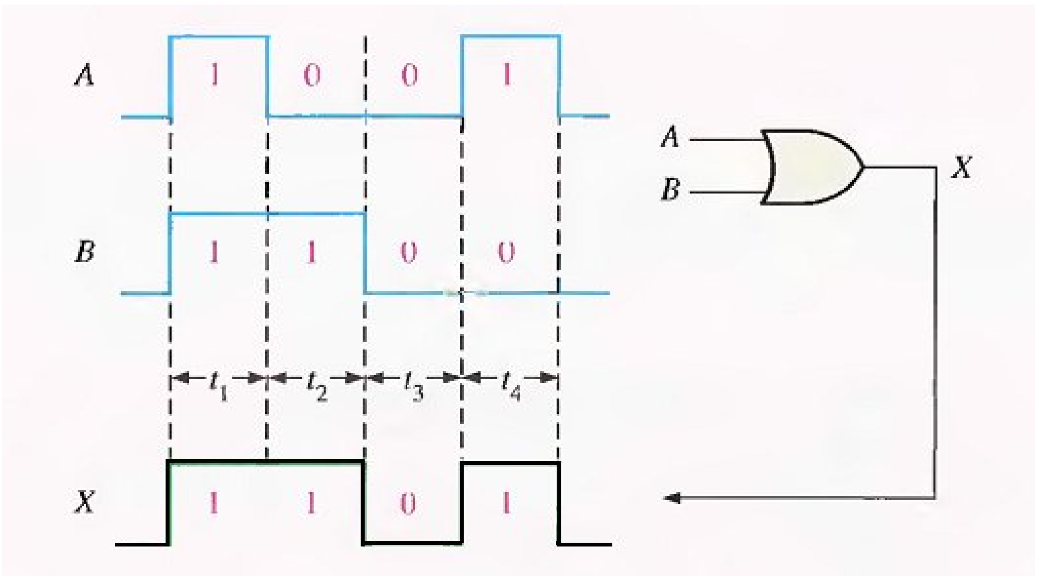


Fig.(3-9)) Example of OR gate operation with a timing diagram showing input and output relationships.

<u>Logic Expressions for an OR Gate</u>

The logical OR function of two variables is represented mathematically by a + between the two variables, for example, A + B. Addition in Boolean algebra involves variables whose values are either binary 1 or binary 0. The basic rules for Boolean addition are as follows:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

Boolean addition is the same as the OR function.

Notice that Boolean addition differs from binary addition in the case where two 1 s are added. There is no carry in Boolean addition.

The operation of a 2-input OR gate can be expressed as follows: If one input variable is A, if the other input variable is B, and if the output variable is X, then the Boolean expression is

$$X=A+B$$

Fig.(3-10)(a) shows the OR gate logic symbol with two input variables and the output variable labelled.

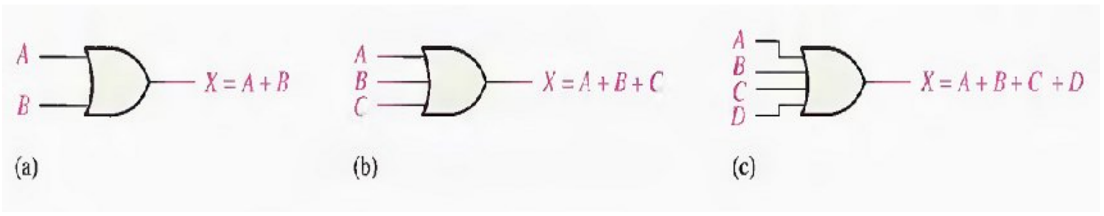


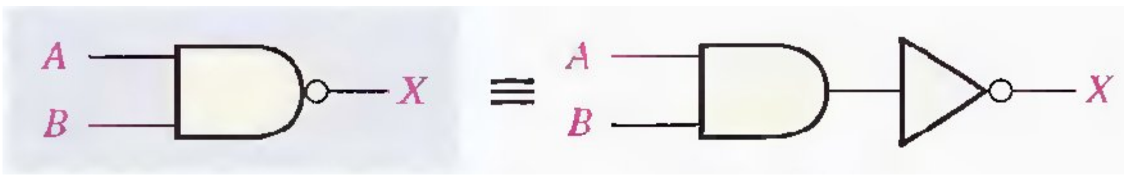
Fig.(3-10) Boolean expressions for AND gates with two, three, and four inputs.

To extend the OR expression to more than two input variables. a new letter is used for each additional variable. For instance, the function of a 3-input OR gate can be expressed as X = A + B + C. The expression for a 4-input OR gate can be written as X = A + B + C + D, and so on. Parts (b) and (c) of Fig.(3-10) how OR gates with three and four input variables, respectively.

THE NAND GATE

The NAND gate is a popular logic element because it can be used as a universal gate: that is, NAND gates can be used in combination to perform the AND, OR, and inverter operations.

The term NAND is a contraction of NOT-AND and implies an AND function with a complemented (inverted) output. The standard logic symbol for a 2-input NAND gate and its equivalency to an AND gate followed by an inverter are shown in Fig.(3-11)(a), where the symbol ≡ means equivalent to. A rectangular outline symbol is shown in part (b).



(a) (b)

Fig.(3-11) Standard NAND gate logic symbols.

<u>Operation of a NAND Gate</u>

A NAND gate produces a LOW output only when all the inputs are HIGH. When any of the inputs is LOW, the output will be HIGH. For the specific case of a 2-input NAND gate, as shown in Fig.(3-11) with the inputs labelled A and B and the output labelled X, the operation can be stated as follows:

For a 2-input NAND gate, output X is LOW only when inputs A and B are HIGH; X is HIGH when either A or B is LOW, or when both A and B are LOW.

Note that this operation is opposite that of the AND in terms of the output level. In a NAND gate, the LOW level (0) is the active or asserted output level, as indicated by the bubble on the output. Fig.(3-12) illustrates the operation of a 2-input NAND gate for all four input combinations, and Table 3-4 is the truth table summarizing the logical operation of the 2-input NAND gate.

The NAND is the same as the AND except the output is inverted.

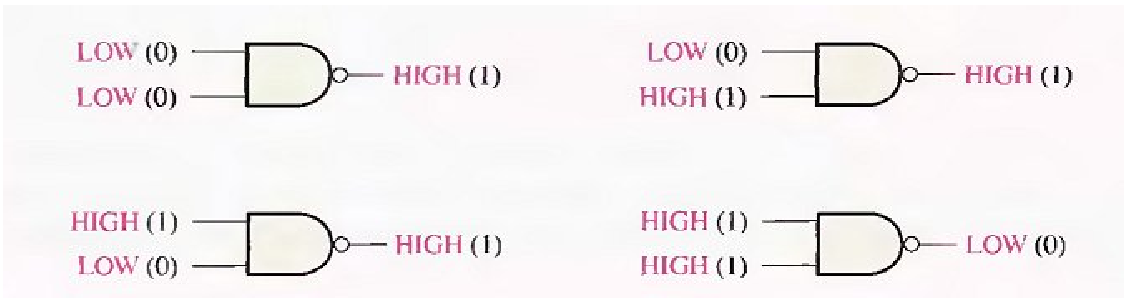
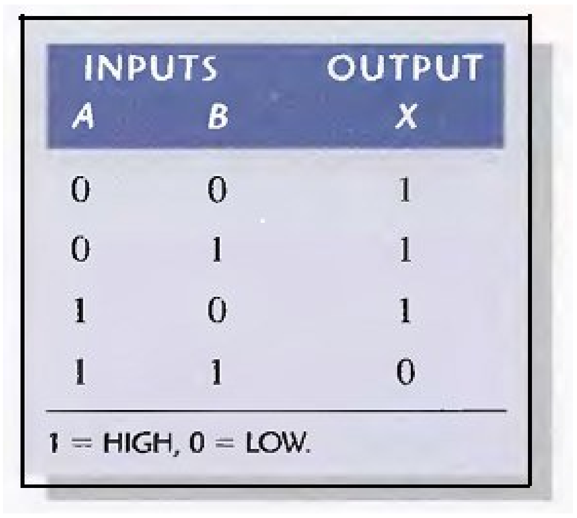


Fig.(3-12) Operation of a 2-input NAND gate.

Table 3-4 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.

<u>Negative-OR Equivalent Operation of a NAND Gate</u>

Inherent in a NAND gate's operation is the fact that one or more LOW inputs produce a HIGH output. Table 3-4 shows that output X is HIGH (1) when any of the inputs, A and B, is LOW (0). From this viewpoint, a NAND gate can be used for an OR operation that requires one or more LOW inputs to produce a HIGH output. This aspect of NAND operation is referred to as negative-OR. The term negative in this context mean that the inputs are defined to be in the active or asserted state when LOW.

For a 2-input NAND gate performing a negative-OR operation, output X is HIGH when either input A or input B is LOW, or when both A and B are LOW.

When a NAND gate is used to detect one or more LOWs on its inputs rather than all HIGHs, it is performing the negative-OR operation and is represented by the standard logic symbol shown in Fig.(3-13).

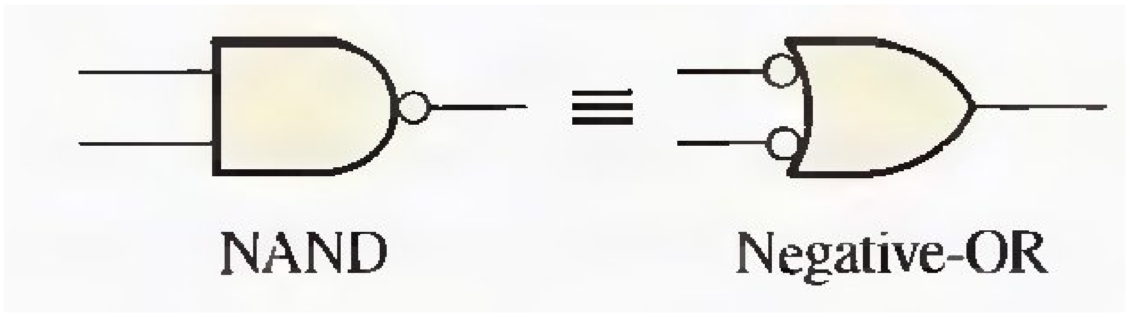


Fig.(3-13) Standard symbols representing the two equivalent operation of a NAND gate.

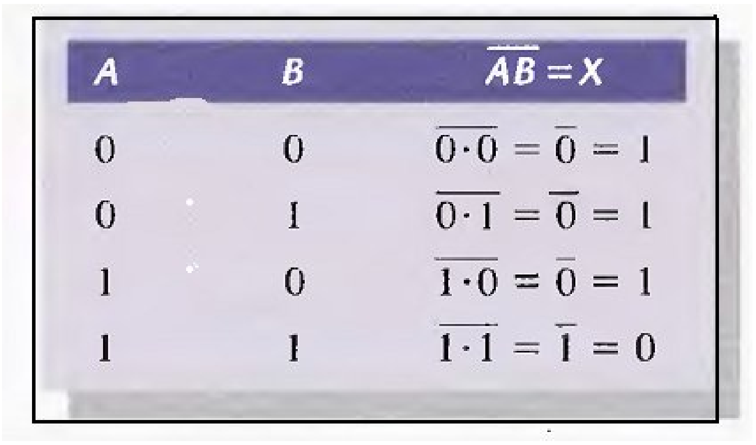
__Logic Expressions for a NAND Gate__

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

$$X = \overline{AB}$$

This expression says that the two input variables, A and B, are first ANDed and then complemented, as indicated by the bar over the AND expression. This is a description in equation form of the operation of a NAND gate with two inputs. Evaluating this expression for all possible values of the two input variables, you get the results shown in Table 3-5.

Table 3-5.

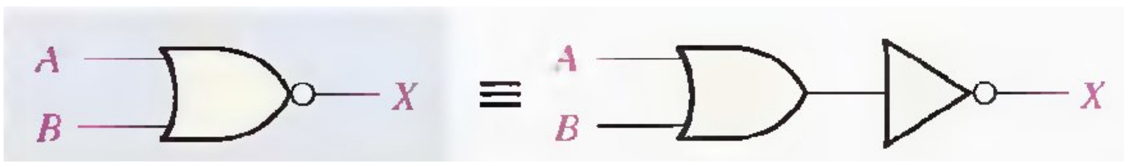


A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

The NOR Gate

The NOR gate, like the NAND gate, is a useful logic element because it can also be used as a universal gate; that is, NOR gates can be used in combination to perform the AND, OR, and inverter operations.

The term NOR is a contraction of NOT-OR and implies an OR function with an inverted (complemented) output. The standard logic symbol for a 2-input NOR gate and its equivalent OR gate followed by an inverter are shown in Fig.(3-14)(a).



(a) (b)

Fig.(3-14) Standard NOR gate logic symbols.

Operation of a NOR Gate

A NOR gate produces a LOW output when any of its inputs is HIGH. Only when all of its inputs are LOW is the output HIGH. For the specific case of a 2-input NOR gate, as shown in Fig.(3-14) with the inputs labelled A and B and the output labelled X, the operation can be stated as follows:

For a 2-input NOR gate, output X is LOW when either input A or input B is HIGH, or when both A and B are HIGH; X is HIGH only when both A and B are LOW.

This operation results in an output level opposite that of the OR gate. In a NOR gate, the LOW output is the active or asserted output level as indicated by the bubble on the output.

Fig.(3-15) illustrates the operation of a 2-input NOR gate for all four possible input combinations, and Table 3-6 is the truth table for a 2-input NOR gate.

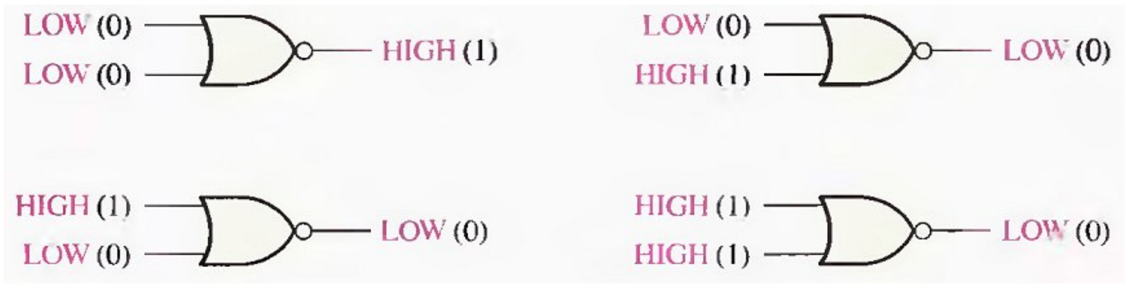
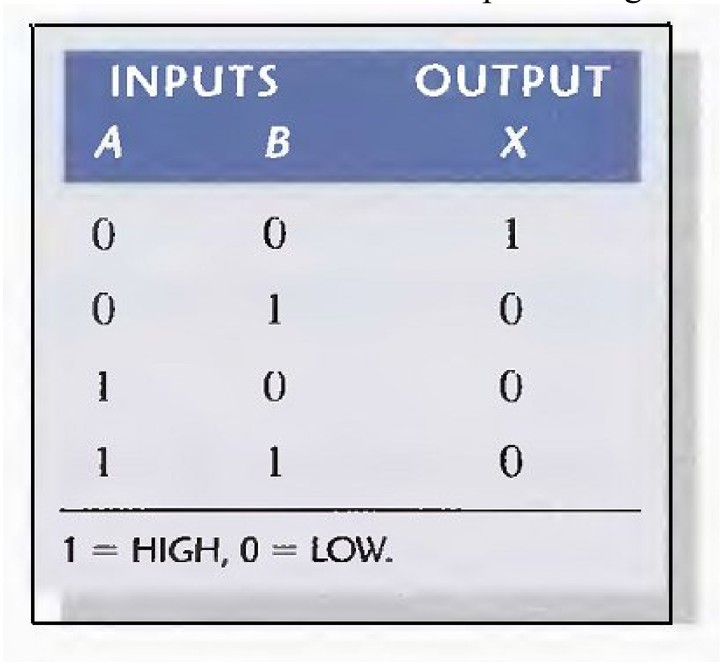


Fig.(3-15) Operation of a 2-input NOR gate.

Table 3-6 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.

Negative-AND Equivalent Operation of the NOR Gate

A NOR gate, like the NAND, has another aspect of its operation that is inherent in the way it logically functions. Table 3-6 shows that a HIGH is produced on the gate output only when all of the inputs are LOW. From this viewpoint, a NOR gate can be used for an AND operation that requires all LOW inputs to produce a HIGH output. This aspect of NOR operation is called negative-AND.

The term negative in this context means that the inputs are defined to be in the active or asserted state when LOW.

For a 2-input NOR gate performing a negative-AND operation, output X is HIGH only when both inputs A and B are LOW.

When a NOR gate is used to detect all LOWs on its inputs rather than one or more HIGHs, it is performing the negative-AND operation and is represented by the standard symbol in Fig.(3-16). It is important to remember that the two symbols in Fig.(3-16), represent the same physical gate and serve only to distinguish between the two modes of its operation.

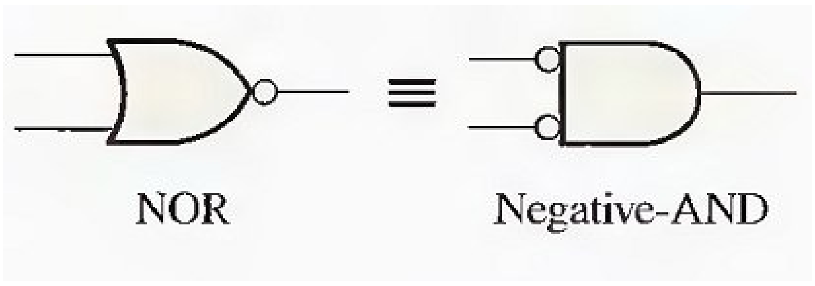


Fig.(3-16) Standard symbols representing the two equivalent operations of a NOR gate.

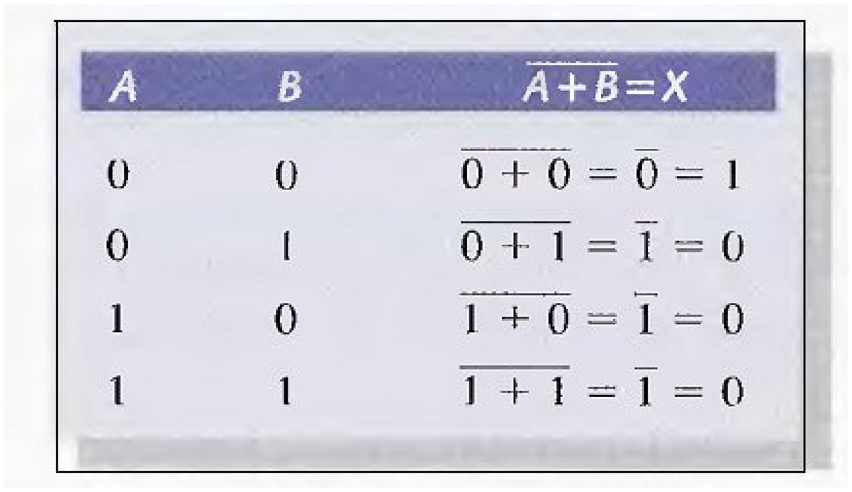
<u>Logic Expressions for a NOR Gate</u>

The Boolean expression for the output of a 2-input NOR gate can be written as

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

This equation says that the two input variables are first ORed and then complemented, as indicated by the bar over the OR expression. Evaluating this expression, you get the results shown in Table 3-7. The NOR expression can be extended to more than two input variables by including additional letters to represent the other variables.

Table 3-7



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

THE EXCLUSIVE-OR AND EXCLUSIVE-NOR GATES

Exclusive-OR and exclusive-NOR gates are formed by a combination of other gates already discussed. However, because of their fundamental importance in many applications, these gates are often treated as basic logic elements with their own unique symbols.

The Exclusive-OR Gate

Standard symbol for an exclusive-OR (XOR for short) gate is shown in Fig.(3-17). The XOR gate has only two inputs.

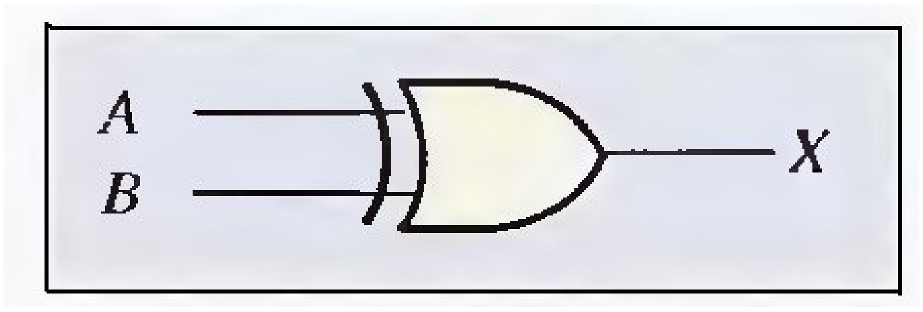


Fig.(3-17) Standard symbol for an exclusive-OR.

For an exclusive-OR gate, output X is HIGH when input A is LOW and input B is HIGH, or when input A is HIGH and input B is LOW: X is LOW when A and B are both HIGH or both LOW.

The four possible input combinations and the resulting outputs for an XOR gate are illustrated in Fig.(3-18). The HIGH level is the active or asserted output level and occurs only when the inputs are at opposite levels. The operation of an XOR gate is summarized in the table shown in Table 3-8.

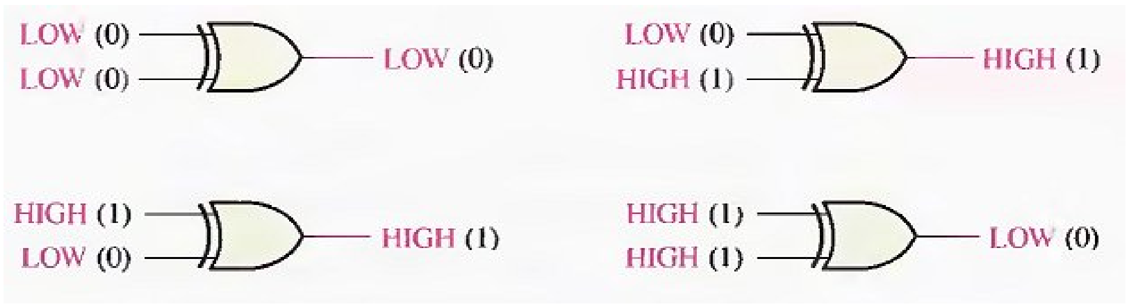
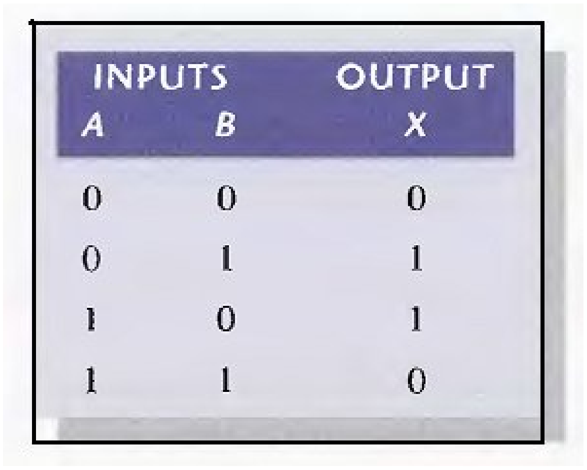


Fig.(3-18) All possible logic levels for an exclusive-OR gate.

Table 3-8 Truth table for an exclusive-OR gate.



INPUTS		OUTPUT
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

<u>The Exclusive-NOR Gate</u>

Standard symbols for an exclusive-NOR (XNOR) gate are shown in Fig.(3-19). Like the XOR gate, an XNOR has only two inputs. The bubble on the output of the XNOR symbol indicates that its output is opposite that of the XOR gate. When the two input logic levels are opposite, the output of the exclusive-NOR gate is LOW. The operation can be stated as follows (A and B are inputs, X is the output):

For an exclusive-NOR gate, output X is LOW when input A is LOW and input B is HIGH, or when A is HIGH and B is LOW; X is HIGH when A and B are both HIGH or both LOW.

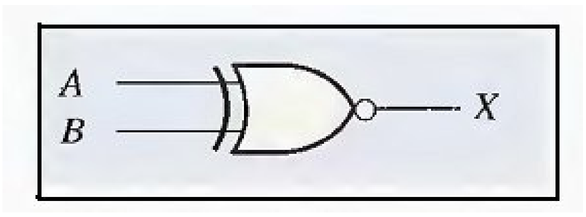


Fig.(3-19) Standard logic symbols for the exclusive-NOR gate.

The four possible input combinations and the resulting outputs for an XNOR gate are shown in Fig,(3-20). The operation of an XNOR gate is summarized in Table 3-9. Notice that the output is HIGH when the same level is on both inputs.

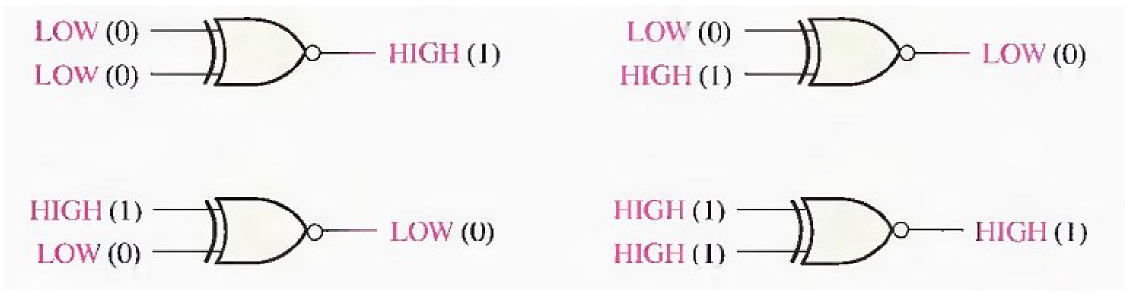
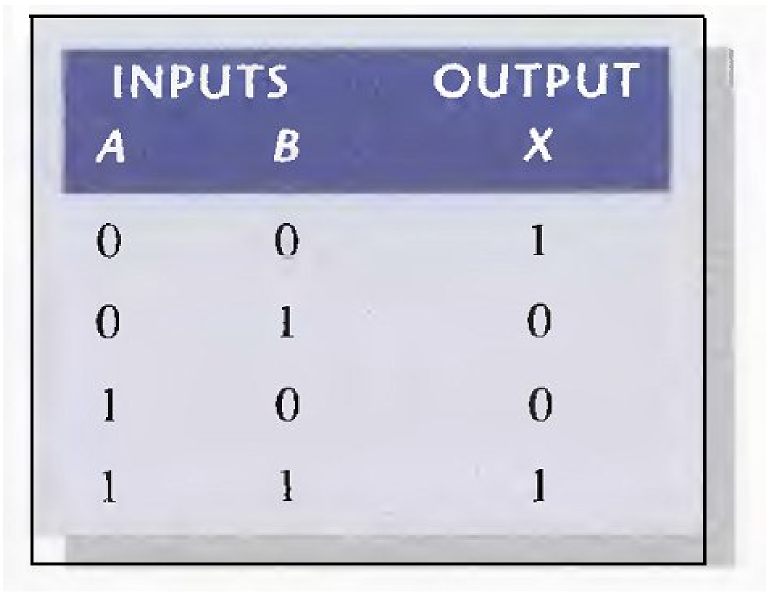


Fig.(3-20) All possible logic levels for an exclusive-NOR gate.

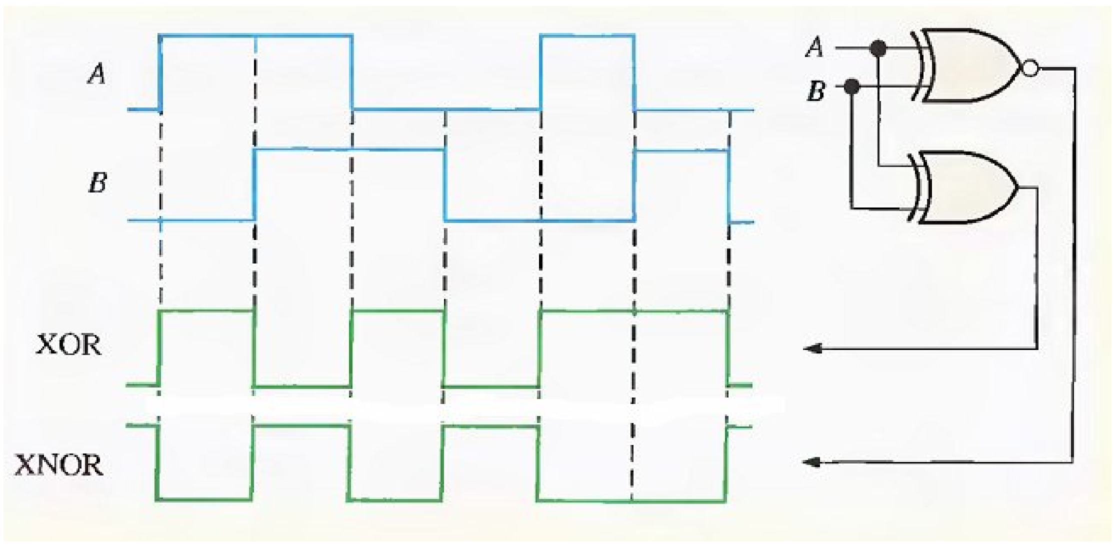
Table 3-9 Truth table for an exclusive-NOR gate.



INPUTS		OUTPUT
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Example

Determine the output waveforms for the XOR gate and for the XNOR gate, given the input waveforms, A and B, in Figure below:



Finish