

Introduction

- We are in “Information age” since digital systems have such a prominent and growing role in modern society.
- They are involved in our business transactions, communications, transportation, medical treatment and entertainment.
- In industrial world they are heavily employed in design, manufacturing, distribution and sales.

Analog System

- Analog systems process analog signals (continuous time signals) which can take any value within a range
- for example the output from a speaker or a microphone.

Digital System

- Digital systems process digital signals which can take only a limited number of values (discrete steps), usually just two values are used: the positive supply voltage (+Vs) and zero volts (0V).
- Digital systems contain devices such as logic gates, flip-flops, shift registers and counters.
- The general purpose digital computer is a best known example of **digital system**.

Working principle of generic digital computer:

- Memory stores programs as well as input, output and intermediate data.
- The data path performs arithmetic and other data-processing operations as specified by the program.
- The control unit supervises the flow of information between the various units.
- A data path, when combined with the control unit, forms a component referred to as a *central processing unit*, or CPU.

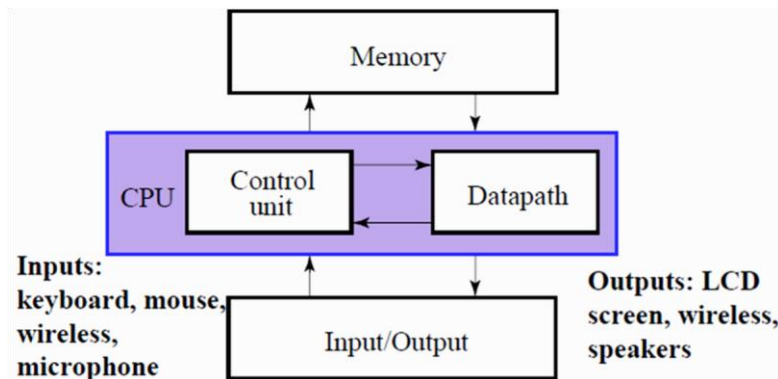


Fig: Block diagram of digital computer

- The program and data prepared by the user are transferred into memory by means of an input device such as a keyboard.
- An output device, such as a CRT (cathode-ray tube) monitor, displays the results of the computations and presents them to the user.

Advantages of digital system:

- Less expensive
- More reliable
- Easy to manipulate
- Flexibility and Compatibility
- Information storage can be easier in digital computer systems than in analog ones. New features can often be added to a digital system more easily too.

Disadvantages of digital system:

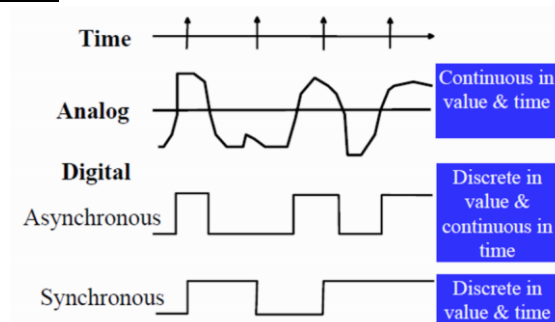
- Use more energy than analog circuits to accomplish the same tasks, thus producing more heat as well.
- Digital circuits are often fragile, in that if a single piece of digital data is lost or misinterpreted, the meaning of large blocks of related data can completely change.
- Digital computer manipulates discrete elements of information by means of a binary code.
- Quantization error during analog signal sampling.

Information Representation

Signals

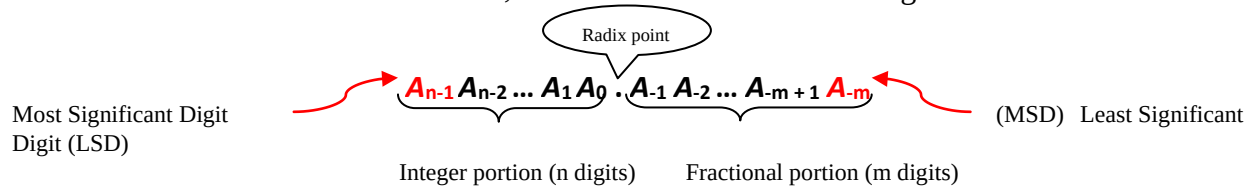
- Information variables represented by physical quantities.
- For digital systems, the variables take on discrete values.
- Two level or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
 - digits 0 and 1
 - words (symbols) False (F) and True (T) • words (symbols) Low (L) and High (H)
 - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities

Signal Examples over time



Number Systems

- A number with radix r is represented by a string of digits as below i.e. wherever you guys see numbers of whatever bases, all numbers can be written in general as:



in which $0 \leq A_i < r$ (since each being a symbol for particular base system viz. for $r = 10$ (decimal number system) A_i will be one of 0,1,2,...,8,9). Subscript i gives the position of the coefficient and, hence, the weight r^i by which the coefficient must be multiplied.

In general, a number in base r contains r digits, 0, 1, 2... $r-1$, and is expressed as a power series in r with the general form:

$$(\text{Number})_r = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + A_{-m+1}r^{-m+1} + A_{-m}r^{-m}$$

$$(\text{Number})_r = \left(\sum_{i=0}^{n-1} A_i \cdot r^i \right) + \left(\sum_{j=-m}^{-1} A_j \cdot r^j \right)$$

(Integer Portion) + (Fraction Portion)

Decimal system and other radix system

- Decimal system uses 10 digits for its numbering system. So, its base or radix is 10. In digital systems, binary numbers are used, i.e., with base or radix 2.

0	1	2	3	4	5	6	7	8	9	10	11	12			20			99	100			
0	1									10	11	100	101	110	111	1000	1001	1010	1011	1100		

- Consider the sets of numbers given above.
- The first one uses decimal systems, so all digits from 0 to 9 are used for representing COUNT. In the second set, only two digits 0 and 1 are used, so numbers having other digits 2 to 9 have been removed from the count.
- The resulting numbering system is BINARY.

- Hence the following table gives the binary equivalent of decimal numbers.

Table 1.1.1: Decimal and binary

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

- To understand the equivalence in more detail, let us first examine a decimal number, say 2053.
- This number essentially means

$$2 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 3 \times 10^0$$
- Similarly, a binary number, say 1001 means

$$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \text{ which is equal to 9 in decimal.}$$

Q. Find out the decimal equivalent of binary 1101.

A. 13

Q. Find out the decimal equivalent of binary 10101.

A. 21

Q. Find out the decimal equivalent of binary 101001.

A. 41

The base-2 systems are specifically represented as 1101_2 .

Other than binary system, the octal system (base-8) and hexadecimal system (base-16) are also used in digital electronics and computing system.

Base Conversion Methods

- For conversion of a decimal number to a binary number, the decimal number is divided by 2 successively.
- The quotient and remainder are noted down at each stage.
- The quotient of one stage is divided by 2 at the next stage.
- The process is repeated until the quotient becomes zero.
- The binary number equivalent to the decimal number is given by the following expression

$$\text{Binary Number} = R_N R_{N-1} R_{N-2} \dots R_2 R_1$$

- where $R_1, R_2, \dots, R_N$ are the remainders at first, second, $\dots$, and N^{th} stage respectively.

Example 1. Convert the decimal number 13 to binary.

<u>Quotient</u>	<u>Remainder</u>	<u>Remark</u>
$13 \div 2 = 6$	1 (LSB)	There are six 2s and one 1.
$6 \div 2 = 3$	0	There are three 4s and no 2s.
$3 \div 2 = 1$	1	There are one 8 and one 4.
$1 \div 2 = 0$	1 (MSB)	There is no 16 and one 8.

Therefore, $13_{10} = 1101_2$

Q. Convert the decimal numbers 25, 39, and 64 to binary.

A. 11001, 100111, 1000000

Q. Convert the binary numbers 1010, 0010 and 1101 to decimal.

A. 10, 2, 13

Fractions Conversion

- In the case of whole numbers, repeated division by 2 was used whereas in the case of fractions the repeated multiplication by 2 is used.

$$(0.8125)_{10} \quad \times 2 = 1.625 \quad \quad \quad 1 \quad \quad \text{MSB}$$

$(0.625)_{10}$	$\times 2 = 1.25$	1	
$(0.25)_{10}$	$\times 2 = 0.50$	0	
$(0.50)_{10}$	$\times 2 = 1.00$	1	LSB

$(0.8125)_{10} = (0.1101)_2$

Q. Convert 13.625 into binary real number.

A. 1 1 0 1.1 0 1

Q. Convert 12.25 into binary real number.

A. 1 1 0 1.1 0 1

Q. Convert 6.5 into binary real number.

A. 1 1 0 1.1 0 1

Q. Convert -12.25 into binary real numbers.

Complement Numbers

- Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation
- The r 's complement of a number N (the number N has the base r) is defined as: $r^n - 1$, where n is the number of digits (integer part) of the number.
- The complement of this is $r^n - (r^n - N) = N$.
- This proves that the two numbers N and $r^n - N$ are r 's complement of each other.
- For example, the 10's complement of the number 32 (with base 10) is given as $10^2 - 32 = 68$.
- The $(r-1)$'s complement of a number N is defined, similarly as $r^n - r^m - N$, where n is the number of digits in the integer part of the number N and m is the number of digits in the fraction part.
 - Again, r is the base of the number.

9's complement.

- To calculate 9's complement of decimal number, each digit of a decimal number is subtracted from 9.
- Example: the 9's complement of 26 is $99 - 26 = 73$
- Example: the 9's complement of 26.6 (using $(r-1)$'s complement with $r=10$)

$$\begin{aligned}
 &= 10^2 - 10^{-1} - 26.6 \\
 &= 100 - 0.1 - 26.6 \\
 &= 73.3
 \end{aligned}$$

10's complement.

- The 10's complement of a n-digit decimal number is calculated as 10^n minus given number.
- Example: the 10's complement of 26 is $10^2 - 26 = 74$
- Alternatively, the 10's complement can be calculated as the 9's complement of the number plus 1.
- Example: the 10's complement of 26 is $73 + 1 = 74$ 9's complement + 1 = $(99 - 26) + 1 = 74$.

Note that the sum of a n-digit number and its 10's complement is 10^n . Example: $26 + 10$'s complement of 26 = $26 + 74 = 100 = 10^2$

1's complement.

- The 1's complement in the binary number system is similar to 9's complement in the decimal system: each bit of the binary number is subtracted from 1.
- Example: the 1's complement of 10110 is $11111 - 10110 = 01001$

Note that 1's complement can be obtained by simply converting each 0 to 1, and each 1 to 0.

2's complement.

- As with the 10's complement of the decimal number, the 2's complement of a binary number is obtained by adding 1 to the 1's complement. (by definition of r's complement of a number N with base r, the 2's complement of a binary number (base 2) is $2^n - N$ and consequently the 1's complement is (r-1)'s complement with $r=2$. Verify it.)
- Example: the 2's complement of 10110 is $01001 + 1 = 01010$

Basic Arithmetic of Binary Numbers

- The following table shows a general rule for binary addition.

$$\begin{aligned}
 0 + 0 &= 0 \\
 0 + 1 &= 1
 \end{aligned}$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

↑

Carry

- That is, *a carry is generated every time we run out of a digit to represent the sum*, as is passed to the left. In decimal analogy, $0 + 0 = 0$, $1 + 0 = 1$, $1 + 1 = 2$, ..., $1 + 8 = 9$, $1 + 9 = 10$.
- When 1 is added to 1 in binary, the sum is 0 with a carry 1, and this carry is added to the sum of the adjacent bits. Example:

1 0 1 0 [10]	1 0 0 1 [9]	1 1 1 [7]	1 0 1 0 [10]
<u>+ 0 1 0 0</u> [+ 4]	<u>+ 0 1 0 1</u> [+ 5]	<u>+ 1 1</u> [+ 3]	<u>+ 1 1 0 1</u> [+ 13]
1 1 1 0 [14]	1 1 1 0 [14]	1 0 1 0 [10]	1 0 1 1 1 [23]
			↑
			Carry

Similarly, while subtracting an '1' from a '0', a borrow is required.

Example:

1 0 1 1 [11]	1 0 1 0 [10]	1 0 0 0 [8]
<u>- 1 0 0 1</u> [- 9]	<u>- 0 1 0 1</u> [- 5]	<u>+ 1 0 1</u> [- 5]
0 0 1 0 [2]	0 1 0 1 [5]	0 0 1 1 [3]

The results are different while a larger number is subtracted from a smaller number.

Example:

1 1 0 [6]	0 1 1 0 [6]
<u>- 1 1 1</u> [- 7]	<u>- 1 0 0 0</u> [- 8]
1 1 1 [- 1]	1 1 1 0 [- 2]

The results are not negative equivalent of the decimal, instead, their 2's complement.

Use of 2's complement in binary calculations

- When a binary number is added with its 2's complement, the result is zero, ignoring the carry.
- Example: the 2's complement of 1010 is 0101 + 1 = 0 1 1 0
- This applies for binary number of any length. If a n-bit binary number is added to its 2's complement, the result is n 'zeros', ignoring carry.
- Hence 2's complement of a binary number represents the negative of the same number.
- Example: add 9 and - 9.

$$1\ 0\ 0\ 1_2 = 9;$$

$$01\ 1\ 0 + 1 = 0\ 1\ 1\ 1 = -9 \text{ (i.e. 2's complement of 9)}$$

$$1\ 0\ 0\ 1\ [9] + 0\ 1\ 1\ 1\ [-9] = (1)\ 0\ 0\ 0\ 0\ [0]$$

- Thus the *addition of 2's complement of a number is equivalent to its subtraction.*
- Example:

$$\begin{array}{rcl} 1\ 1\ 1 & [7] & 1\ 0\ 0\ 1 & [9] \\ + 0\ 1\ 1 & [-5] & + 1\ 0\ 1\ 1 & [-5] \\ \hline (1)\ 0\ 1\ 0 & [2] & (1)\ 0\ 1\ 0\ 0 & [4] \end{array}$$

Representation of Signed and Unsigned Numbers

- In signed binary representation, the MSB represents the sign: a 0 means positive and an 1 means negative number.
- For one byte long binary - 9 can be represented in the following ways:

$$\text{Signed magnitude representation} \quad 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

$$\text{Signed 1's complement representation} \quad 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0$$

$$\text{Signed 2's complement representation} \quad 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1$$

- *Unsigned representation doesn't use MSB as sign bit.*
- Examples using signed bit:

$$\begin{array}{rcl} 0\ 1\ 0\ 0\ 1 & [9] & 0\ 1\ 0\ 0\ 1 & [9] & 0\ 0\ 0\ 1\ 0 & [2] \\ 0\ 0\ 1\ 0\ 1 & [+5] & 1\ 1\ 0\ 1\ 1 & [-5] & 1\ 1\ 0\ 1\ 1 & [-5] \\ 0\ 1\ 1\ 1\ 0 & [14] & (1)\ 0\ 0\ 1\ 0\ 0 & [4] & 1\ 1\ 1\ 0\ 1 & [-3] \\ & & \uparrow & & \uparrow & \end{array}$$

positive

negative

- The last result is negative and in 2's complement.
- To get the correct binary value, write 1's complement, add 1 and put a negative sign: 1's complement = 0 0 1 0, adding 1 gives 0 0 1 1; so the result is - 0 0 1 1

Ex. Add -9 and -5

1 0 1 1 1 1

1 0 1 1

(1) 1 0 0 1 0

↑ negative

To get the correct answer, perform the following operations:

0 1 1 0 1 [1's complement of the above result 10010]

 + 1 [Add 1 to the LSB]

0 1 1 1 0 = 14

[01110 is the 2's complement of the result 10010 and hence the correct answer]

Exercises:

Q. Add -2 and 14

Q. Add -10 and 9

Q. Subtract 12 from 9 using signed number format and also using 2's complement format.

Verify the results.

Octal, Hexadecimal and BCD

Octal

The base of the octal number system is 8, so it uses digits from 0 to 7 for its representation.

Table 1.7.1 shows octal and binary equivalents.

Table 1.7.1: Decimal, Binary and Octal numbers

Decimal	Octal	Binary coded octal
0	0	0

1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111
8	10	001 000
9	11	001 001
10	12	001 010
15	17	001 111
16	20	010 000
63	77	111 111
64	100	001 000 000
100	144	001 100 100

Conversion to decimal.

$$(56)_8 = 5 \times 8^1 + 6 \times 8^0 = (46)_{10}$$

$$(0.563)_8 = 5 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3} = (0.724...)_{10}$$

Conversion to octal

$$(62)_{10} \div 8 = 7 \quad 6 \text{ (remainder)} \quad \text{LSB}$$

$$(7)_{10} \div 8 = 0 \quad 7 \text{ (remainder)} \quad \text{MSB}$$

$$(62)_{10} = (76)_8$$

$$(0.96)_{10} \times 8 = 7.68 \quad 7 \quad \text{MSB}$$

$$(0.68)_{10} \times 8 = 5.44 \quad 5$$

$$(0.44)_{10} \times 8 = 3.52 \quad 3$$

$$(0.52)_{10} \times 8 = 4.16 \quad 4 \quad \text{LSB}$$

$$(0.96)_{10} = (0.7535)_8$$

Binary to octal

- For conversion from binary to octal, the binary word is divided into groups of 3 from right as $2^3 = 8$. Left zero padding (adding zeros to the left of the binary word) is used if the leftmost group falls short of required 3 bits.

$$\begin{aligned}(1101011)_2 &= 1\ 101\ 011 &= 153_8 \\(1101.11101)_2 &= 1\ 101.111\ 010 &= 15.72_8 \\(1100110001.010)_2 &= 1\ 100\ 110\ 001.010 &= 1461.2_8\end{aligned}$$

Octal to binary

$$\begin{aligned}56.34_8 &= 101\ 110.011\ 100_2 \\10234.32_8 &= 001\ 010\ 011\ 100.011\ 010_2 \\0.7664_8 &= 000.111\ 110\ 110\ 100\ 010_2\end{aligned}$$

Hexadecimal

- The radix of hexadecimal number is 16. So four binary bits represent a hexadecimal number. This number system is extensively used in computer industry. The following table shows hexadecimal numbers and their binary representations.

Table: 1.7.2: Decimal, hexadecimal and binary coded hexadecimal

Decimal	Hexadecimal	Binary Coded Hexadecimal
0	0	0000
1	1	0001
2	2	0010
9	9	1001
10	A	1010

11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	0001 0000
37	25	0010 0101
43	2B	0010 1011
255	FF	1111 1111

Binary Coded Decimal

- The BCD is the simplest binary code to represent a decimal number. In BCD code, a decimal number is represented by four binary bits.
- The table 1.7.3 shows the standard BCD codes for their decimal equivalents:

Table 1.7.3: Decimal and BCD

Decimal	BCD
0	0000
1	0001
9	1001
10	0001 0000
11	0001 0001
99	1001 1001
100	0001 0000 0000

- BCD codes are used where the decimal information is directly transferred into or out of a digital system.

- Electronic calculators, digital voltmeters, digital clocks etc. work with BCD numbers. BCD codes have also been used in early computers, but modern computers do not use BCD numbers, due to their complexity in forming complements and calculations.

- Example:

$$\begin{array}{r}
 8 \qquad \qquad \qquad 1\ 0\ 0\ 0 \\
 +5 \qquad \qquad \qquad \underline{0\ 1\ 0\ 1} \\
 13 \qquad \qquad \qquad 1\ 1\ 0\ 1 \rightarrow \text{incorrect sum} \\
 \qquad \qquad \qquad \underline{+ 0\ 1\ 1\ 0} \rightarrow \text{add 6} \\
 \qquad \qquad \qquad 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1 \rightarrow \text{correct sum}
 \end{array}$$

Thus the calculations in the case of binary coded decimal system are not so obvious and straightforward as in the case of simple binary system.

Gray code, Alphanumeric code

Gray code

- In gray code, the bits are arranged in such a way that it changes by only one bit as it sequences from one number to the next.
- So the transitional errors are minimized.
- A typical application of the Gray code occurs in a rotational or shaft encoder, which is used to convert some angular position of a shaft to a digital format.
- Binary to gray code and vice versa (do yourself)

Alphanumeric code

- Many applications of digital computers require handling of data that consist not only of numbers, but also of the letters of the alphabet and certain special characters.
- An alphanumeric character set is a set of elements that includes the 10 decimal digits, the 26 letters of the alphabet and a number of special characters such as \$, +, (, etc. Such a set contains between 32 to 64 elements if only uppercase letters are included, or between 64 to 128 if both upper case and lower case letters are included. In the first case, the binary code will require six bits and in the second case, seven bits.

- The ASCII (American Standard Code for Information Interchange) code is the standard alphanumeric binary code.
- In ASCII each character is represented by a 7-bit code and usually an eighth bit is inserted for parity.
- Another alphanumeric code used in IBM equipment is the EBCDIC (Extended BCD Interchange Code). It uses eight bits for each character (and a ninth bit for parity).
- Hollerith code is most commonly used when information is transferred using punched cards.
- It is a 12-bit code, so comparatively inefficient.
- Table 1.8.2: The alphanumeric character codes

TABLE 1-5 Alphanumeric character codes

Character	6-Bit internal code		7-Bit ASCII code		8-Bit EBCDIC code		12-Bit card code
A	010	001	100	0001	1100	0001	12,1
B	010	010	100	0010	1100	0010	12,2
C	010	011	100	0011	1100	0011	12,3
D	010	100	100	0100	1100	0100	12,4
E	010	101	100	0101	1100	0101	12,5
F	010	110	100	0110	1100	0110	12,6
G	010	111	100	0111	1100	0111	12,7
H	011	000	100	1000	1100	1000	12,8
I	011	001	100	1001	1100	1001	12,9
J	100	001	100	1010	1101	0001	11,1
K	100	010	100	1011	1101	0010	11,2
L	100	011	100	1100	1101	0011	11,3
M	100	100	100	1101	1101	0100	11,4
N	100	101	100	1110	1101	0101	11,5
O	100	110	100	1111	1101	0110	11,6
P	100	111	101	0000	1101	0111	11,7
Q	101	000	101	0001	1101	1000	11,8
R	101	001	101	0010	1101	1001	11,9
S	110	010	101	0011	1110	0010	0,2
T	110	011	101	0100	1110	0011	0,3
U	110	100	101	0101	1110	0100	0,4
V	110	101	101	0110	1110	0101	0,5
W	110	110	101	0111	1110	0110	0,6
X	110	111	101	1000	1110	0111	0,7
Y	111	000	101	1001	1110	1000	0,8
Z	111	001	101	1010	1110	1001	0,9
0	000	000	011	0000	1111	0000	0
1	000	001	011	0001	1111	0001	1
2	000	010	011	0010	1111	0010	2
3	000	011	011	0011	1111	0011	3
4	000	100	011	0100	1111	0100	4
5	000	101	011	0101	1111	0101	5
6	000	110	011	0110	1111	0110	6
7	000	111	011	0111	1111	0111	7
8	001	000	011	1000	1111	1000	8
9	001	001	011	1001	1111	1001	9
blank	110	000	010	0000	0100	0000	no punch
.	011	011	010	1110	0100	1011	12,8,3
(	111	100	010	1000	0100	1101	12,8,5
+	010	000	010	1011	0100	1110	12,8,6
\$	101	011	010	0100	0101	1011	11,8,3
*	101	100	010	1010	0101	1100	11,8,4
)	011	100	010	1001	0101	1101	11,8,5
—	100	000	010	1101	0110	0000	11
/	110	001	010	1111	0110	0001	0,1
,	111	011	010	1100	0110	1011	0,8,3
=	001	011	011	1101	0111	1110	8,6

Error Codes

- Binary information transmitted through some form of communication medium is subject to external noise that could change bits from 1 to 0 and vice versa.
- An error detection code is a binary code that detects digital errors during transmission.
- There are various algorithms for the generation of error detection and control codes.
- Some are capable of both detection and correction of error while other only detect the errors.

- The parity bits, hamming codes etc are the examples of error coding.

Parity Bits

- The most common error detection code used is the parity bit.
- It is a simple process of adding a special code bit to a data word that allows the detection of a single error in a given code word in which it is used.
- The encoding of a simple parity bit is done on an odd or even basis.
- An odd parity means the number of 1's in the data word including the parity bit is odd.
- For example, the A in ASCII is coded as

(A) $\rightarrow$ (100 0001)_{ASCII}

- An A in ASCII when coded with odd parity could be shown as

1100 0001

↑

Parity bit

- With the parity bit, the actual number of 1's in the complete word equals three, an odd number.
 - If any bit of the word is converted to 0 from 1, or 1 from 0, it is immediately detected.
 - Similarly the ASCII A with even parity could be coded as
- 0100 0001
- If two bits are transmitted incorrectly, however, the addition of a parity bit will not be of any use in detecting the error.
 - Also, the parity bit doesn't give us any information to correct the word when the error occurs.

Integrated Circuits (ICs)

- An Integrated circuit is an association (or connection) of various electronic devices such as resistors, capacitors and transistors etched (or fabricated) to a semiconductor material such as silicon or germanium.
- It is also called as a **chip** or **microchip**.
- An IC can function as an amplifier, rectifier, oscillator, counter, timer and memory. Sometime ICs are connected to various other systems to perform complex functions.

Types of ICs

- ICs can be categorized into two types
 - ✓ Analog or Linear ICs
 - ✓ Digital or logic ICs
- Further there are certain ICs which can perform as a combination of both analog and digital functions.

Analog or Linear ICs:

- They produce continuous output depending on the input signal.
- From the name of the IC we can deduce that the output is a linear function of the input signal.
- Op-amp (operational amplifier) is one of the types of linear ICs which are used in amplifiers, timers and counters, oscillators etc.

Digital or Logic ICs:

- Unlike Analog ICs, Digital ICs never give a continuous output signal.
- Instead it operates only during defined states.
- Digital ICs are used mostly in microprocessor and various memory applications.
- Logic gates are the building blocks of Digital ICs which operate either at 0 or 1.

Advantages of ICs

- In consumer electronics, ICs have made possible the development of many new products, including personal calculators and computers, digital watches, and video games.
- They have also been used to improve or lower the cost of many existing products, such as appliances, televisions, radios, and high-fidelity equipment.
- The logic and arithmetic functions of a small computer can now be performed on a single VLSI chip called a microprocessor.
- Complete logic, arithmetic, and memory functions of a small computer can be packaged on a single printed circuit board, or even on a single chip.

Levels of Integration

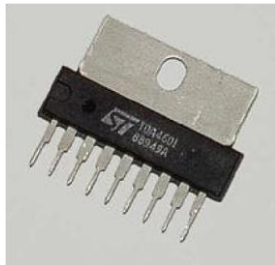
- During 1959 two different scientists invented IC's. Jack Kilby from Texas Instruments made his first germanium IC during 1959 and Robert Noyce made his first silicon IC during the same year.
- But ICs were not the same since the day of their invention; they have evolved a long way.
- Integrated circuits are often classified by the number of transistors and other electronic components they contain:
 - SSI (small-scale integration): Up to 100 electronic components per chip
 - MSI (medium-scale integration): From 100 to 3,000 electronic components per chip

- LSI (large-scale integration): From 3,000 to 100,000 electronic components per chip
- VLSI (very large-scale integration): From 100,000 to 1,000,000 electronic components per chip
- ULSI (ultra large-scale integration): More than 1 million electronic components per chip

SIP (Single In-line Package) and DIP (Dual In-line Package)

SIP

- A **single in-line package** is an electronic device package which has one row of connecting pins.
- It is not as popular as the dual in-line package (DIP) which contains two rows of pins, but has been used for packaging RAM chips and multiple resistors with a common pin. SIPs group RAM chips together on a small board.
- The board itself has a single row of pin-leads that resembles a comb extending from its bottom edge, which plug into a special socket on a system or system-expansion board.
- SIPs are commonly found in memory modules.
- SIP is not to be confused with SIPP which is an archaic term referring to Single In-line Pin Package which was a memory used in early computers.



DIP

- Dual in-line package (DIP) is a type of semiconductor component packaging.
- DIPs can be installed either in sockets or permanently soldered into holes extending into the surface of the printed circuit board.
- DIP is relatively broadly defined as any rectangular package with two uniformly spaced parallel rows of pins pointing downward, whether it contains an IC chip or some other device(s), and whether the pins emerge from the sides of the package and bend downwards.
- A DIP is usually referred to as a **DIP n** , where n is the total number of pins.
- For example, a microcircuit package with two rows of seven vertical leads would be a DIP14. The photograph below shows three DIP14 ICs.



Several DIP variants for ICs exist, mostly distinguished by packaging material:

- **Ceramic Dual In-line Package (CERDIP or CDIP)** ▪ **Plastic Dual In-line Package (PDIP)**
- **Shrink Plastic Dual In-line Package (SPDIP)** -A denser version of the PDIP with a 0.07 in. (1.778 mm) lead pitch.
- **Skinny Dual In-line Package (SDIP)** – Sometimes used to refer to a 0.3 in. wide DIP, normally when clarification is needed e.g. for a 24 or 28 pin DIP.



Fig: Several PDIPs and CERDIPs. The large CERDIP in the foreground is an Intel 8080 microprocessor.

SIMM (Single In-line Memory Module) and DIMM (Dual In-line Memory Module)

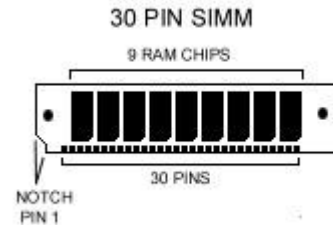
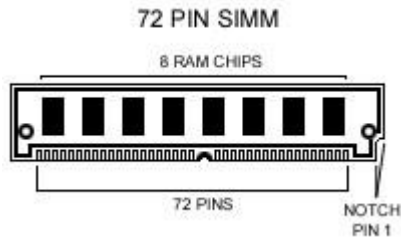
These two terms (SIMM and DIMM) refer to a way series of dynamic random access memory integrated circuits modules are mounted on a printed circuit board and designed for use in personal computers, workstations and servers.

SIMM

- Short for **Single In-line Memory Module**, **SIMM** is a circuit board that holds six to nine memory chips per board, the ninth chip usually an error checking chip (parity/non parity) and were commonly used with Intel Pentium or Pentium compatible motherboards.
- SIMMs are rarely used today and have been widely replaced by DIMMs.
- SIMMs are available in two flavors: 30 pin and 72 pin. 30-pin SIMMs are the older standard, and were popular on third and fourth generation motherboards.
- 72-pin SIMMs are used on fourth, fifth and sixth generation PCs.



4MB 72-PIN SIMM



DIMM

- Short for **Dual In-line Memory Module**, **DIMM** is a circuit board that holds memory chips. DIMMs have a 64-bit path because of the Pentium Processor requirements.
- Because of the new bit path, DIMMs can be installed one at a time, unlike SIMMs on a Pentium that would require two to be added. Below is an example image of a 512MB DIMM memory stick.



512MB DIMM

SO-DIMM is short for **Small Outline DIMM** and is available as a 72-pin and 144-pin configuration. SODIMMs are commonly utilized in laptop computers.

Some of the advantages DIMMs have over SIMMs:

- DIMMs have separate contacts on each side of the board, thereby providing twice as much data as a single SIMM.
- The command address and control signals are buffered on the DIMMs. With heavy memory requirements this will reduce the loading effort of the memory.

Characteristics of IC:

Refer to your class notes

Hamming Codes:(for BCA)

- Hamming codes are one of the very popular examples of linear block coding for error detection.

- It involves a (n,k) linear block codes pattern that have the following parameters:

$$\text{Block length:} \quad n = 2^m - 1$$

$$\text{Number of message bits} \quad k = 2^m - m - 1$$

$$\text{Number of parity bits:} \quad n - k = m$$

Where, $m \geq 3$ (usual case for Hamming codes)

- Considering $n = 7$ and $k = 4$, here is an example of the generator matrix:

$$\mathbf{G} = \begin{array}{cccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

- Similarly, the corresponding parity-check matrix is given by:

$$\mathbf{H} = \begin{array}{cccc|cccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

- Now, $k = 4$ means there are $2^k = 2^4 = 16$ distinct message words. For any given message work, the corresponding code words are defined by using the

$$x = mG$$

- One important fact about the Hamming codes is that this system is a *single error correcting* system.
- That is, the system can detect and correct only one bit error in the data word. If there are more than one errors, Hamming code system cannot correct them.
- Some other examples of error coding are Cyclic codes (examples: Golay code, ReedSolomon codes etc), Convolutional codes ...
- For the generator matrix specified above, the code words for all 16 possible message words are listed in table:

Table : Code word for the hamming code example

Message Word	Code Word	Weight of Code Word
0000	0000000	0
0001	1010001	3
0010	1110010	4
0011	0100011	3
0100	0110100	3
0101	1100101	4
0110	1000110	3
0111	0010111	4
1000	1101000	3
1001	0111001	4
1010	0011010	3
1011	1001011	4
1100	1011100	4
1101	0001101	3
1110	0101110	4
1111	1111111	7

The weight of the code word means the number of 1's in it.

Questions

- Q. What is the importance of number system in digital electronics?
- Q. Discuss the use of 2's complement method in arithmetic operations.
- Q. Differentiate between signed and unsigned numbers.
- Q. What are the advantages of gray codes?
- Q. Compare BCD, binary and Octal on the basis of their advantages and disadvantages.
- Q. What do you know about the alphanumeric character set?

- Q. What are error control codes?
- Q. Can you correct the errors by using the parity checking method? Explain with reason.
- Q. Design the truth table for a 4 bit even parity generator.
- Q. What do you mean by cyclic codes?
- Q. Obtain 9's and 10's complements of 98989, 23782, 1010119
- Q. Find the 10's complement of $(7678)_8$
- Q. Represent the decimal number 8987 in BCD and in octal.
- Q. Convert decimal number 365.5 into base 2, base 3, base 8 and base 16.

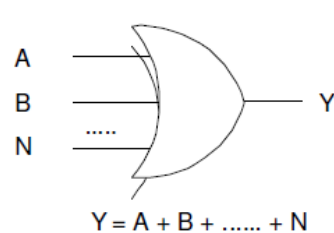
Digital Design Fundamentals

Logic Gates, Truth Table

- A digital computer uses binary number system for its operation.
- In the binary system there are only two digits 0 and 1.
- The computer receives, stores, understands and manipulates information composed of only 0s and 1s.
- The manipulation of binary information is done by electronic circuits known as logic gates. AND, OR, NOT, NAND, NOR and Exclusive-OR are examples of logic gates.
- A truth table is a tabular technique for listing all possible combinations of input variables and respective outputs in a vertical order, listing each input combination one row at a time.
- It is used in digital design to specify an INPUT/OUTPUT relation for a digital network.

The OR Gate

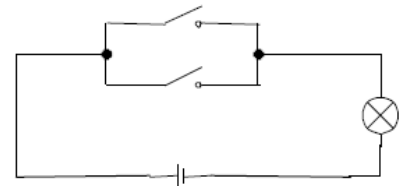
- An OR gate has two or more inputs and a single output.
- The output of the OR gate is in the state 1 if one or more inputs are in the state 1.
- The output will be 0 only when all inputs are 0.
- The n inputs to a logic circuit is designated by A, B, ..., N and the output by Y.
- A standard circuit symbol for the OR circuit is shown in fig 2.1.1 together with the Boolean expression for this gate.
- The equation is to be read “ Y equals A or B or or N.”
- Instead of defining a logical operation in words, an alternative method is to give a truth table which contains a tabulation of all possible input values and their corresponding output.



(a)

<u>Input</u>		<u>Output</u>
<u>A</u>	<u>B</u>	<u>Y</u>
0	0	0
0	1	1
1	0	1
1	1	1

(b)



(c)

Figure 2.1.1: The OR gate

- The following Boolean equations can be observed for an OR gate:

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

$$A + B = B + A$$

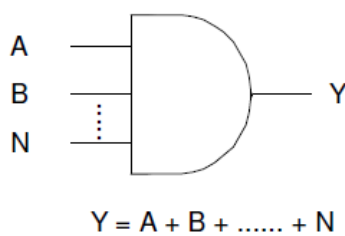
$$A + A = A$$

$$A + 1 = 1$$

$$A + 0 = A$$

The AND Gate s

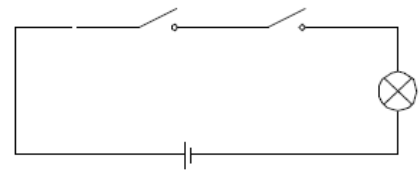
- An AND gate has two or more inputs and a single output.
- The output of the AND gate is in the state 1 if and only if all the inputs are in the state 1.
- The output will be 0 if any one or more inputs are 0.
- A circuit symbol for the AND circuit is shown in figure together with the Boolean expression for this gate.
- The truth table and an electrical analogy are also shown in figure.



(a)

<u>Input</u>		<u>Output</u>
<u>A</u>	<u>B</u>	<u>Y</u>
0	0	0
0	1	0
1	0	0
1	1	1

(b)



(c)

Figure 2.1.2 : The AND gate

a) The standard symbol for an AND gate and its Boolean expression

b) The truth table for a two input AND-gate

c) An electrical analogy explaining the operation of an AND-gate

The following Boolean equations can be observed for an AND gate:

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

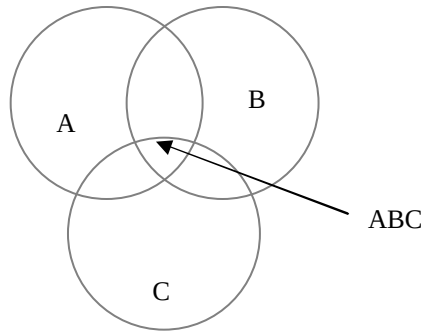
$$AB = BA$$

$$AA = A$$

$$A1 = A$$

$$A0 = 0$$

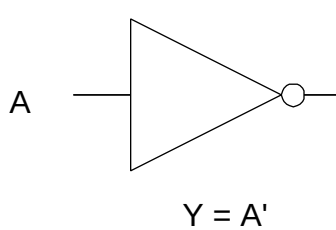
$$A(B + C) = AB + AC$$



The NOT Gate

- A NOT gate has a single input and a single output, and performs the operation of logic negation.
- The output of the NOT gate is in the state 1 if and only if the input is not in the state 1.
- Since the output of the NOT gate is the complement of the input signal, it is also known as an INVERTER.
- The symbol for a NOT gate, its Boolean expression and the truth table are shown in fig 2.1.3. The equation is to be read “Y equals NOT A” or “Y is complement of A”.

Sometimes a prime (') is used instead of the bar ($\bar{}$) to indicate the NOT operation.



(a)

<u>Input</u>	<u>Output</u>
<u>A</u>	<u>Y</u>
Y	1
0	0
1	

(b)

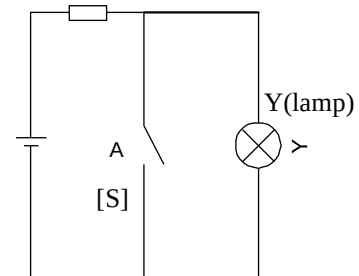


Figure 2.1.3: The NOT operation

a) The standard symbol for an NOT gate and its Boolean expression

b) The truth table for an INVERTER

c) An electrical analogy explaining the operation of a NOT-gate

The NOR and NAND Gates

- A NOR gate is equivalent to an inverter placed at the output of the OR gate and a NAND gate is an AND gate with an inverter at its output.

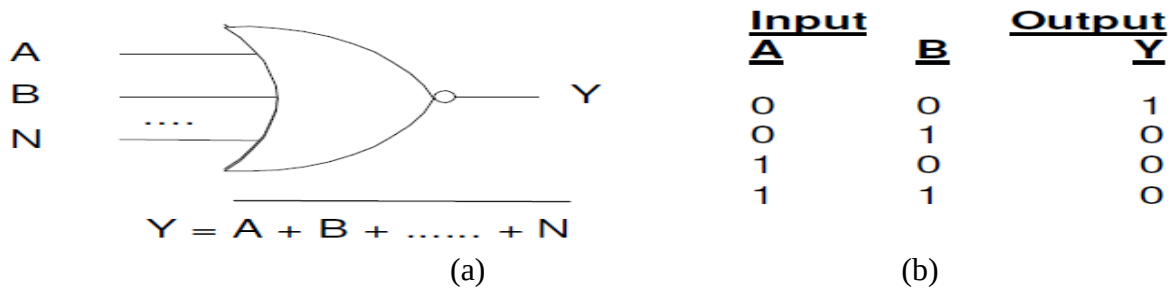


Figure 2.1.4: The NOR operation

a) The standard symbol for an NOR gate and its Boolean expression

b) The truth table for a two input NOR-gate

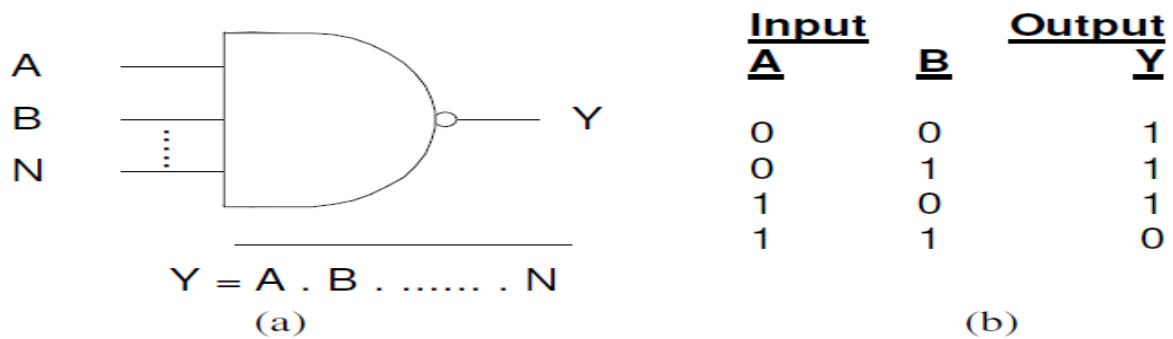


Figure 2.1.5: The NAND operation

a) The standard symbol for an NAND gate and its Boolean expression

b) The truth table for a two input NAND-gate

The Exclusive OR Gate

- The output of an Exclusive OR (XOR) gate with two inputs is in the state 1 if one and only one input is in the state 1.

- In other words, the output of a two-input XOR gate is 1 only when the two inputs are different. The output is 0 when both the inputs are same.
- The symbol, Boolean expression and the truth table of an XOR gate is shown in figure 2.1.6.

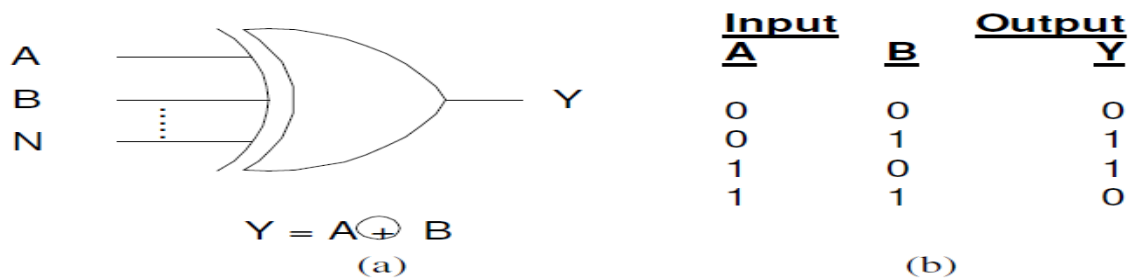


Figure 2.1.6: The XOR operation

a) The standard symbol for an XOR gate and its Boolean expression

b) The truth table for a two input XOR-gate

- The XOR statement can be redefined as “If $A = 1$ and $B = 0$, or if $A = 0$ and $B = 1$, then $Y = 1$. Thus

$$Y = A'B + B'A$$

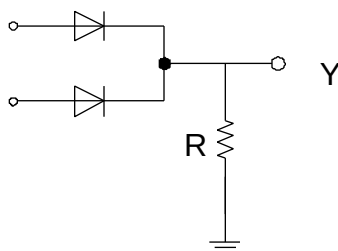
- Equivalently, “If $A = 1$ or $B = 1$ but not simultaneously, then $Y = 1$ ”

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

Realization of Logic Gates using Diodes, and using NAND/NOR

Diode

Realization of the OR Gate



Realization of the AND Gate

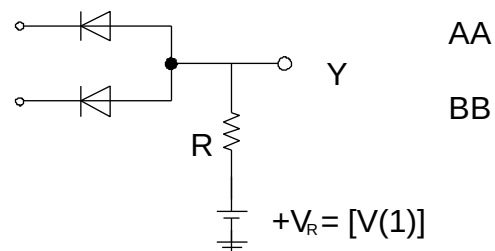
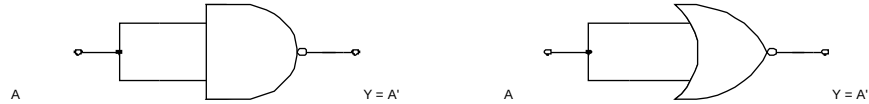


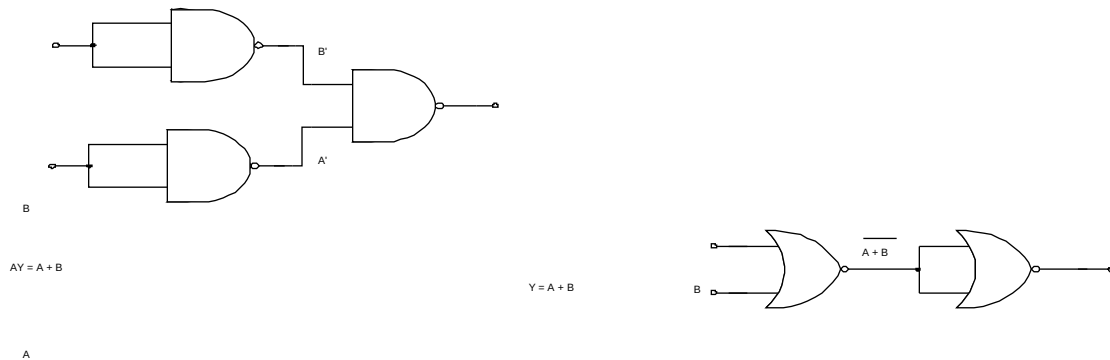
Figure 2.2.1 Gate realizations with diodes

NAND/NOR

Inverter



OR gate using NAND/NOR gates



AND gate using NAND/NOR gates

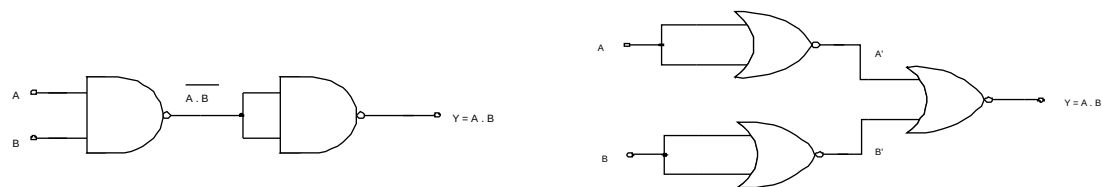


Figure 2.2.2: Gate realization using NAND/NOR gates

- By repeated use of the NAND circuit alone any logical function can be carried out. Similarly, all logic can be performed by using only the NOR circuit.

Logic Gates in the Form of Integrated Circuits

- The logic gates discussed in the previous sections are available in various different forms of integrated circuits or ICs.
- The classifications of these logic ICs is governed not only by the operations of the gates but also by the logic circuit family to which they belong.

- Each logic family has its own basic logic unit upon which all other logic functions and more complex digital circuits are based.
- The most common basic units used for such purposes are NAND and NOR.
- In general, the name of the basic electronic component used in fabrication of the logic circuit is used to derive the name of that particular logic family.
- The most common logic families and their basic electronic components are listed below:

TTL (Transistor-Transistor Logic)	Bipolar Junction Transistor
ECL (Emitter-Coupled Logic)	Bipolar Junction Transistor
MOS (Metal Oxide Semiconductor)	MOSFET
CMOS (Complementary MOS)	CMOS
I ² L (Integrated-Injection Logic)	Bipolar Junction Transistor

- Among these, TTL is the most extensively used family for the time being.
- This family has got an extensive list of digital function realizations and has got good noise performance.
- But, these circuits have higher power consumption rating compared to MOS or CMOS.
- Also, the fabrication area required by the TTL based circuits is larger than MOS or CMOS or I²L.
- So, TTL is not very useful for high density circuit design and fabrication.
- The ECL family of digital circuits is used in high speed applications.
- Still, the compactness of fabrication that can be reached with MOS or CMOS cannot be reached with the ECL.
- MOS and CMOS are useful for compact circuit fabrications.
- This is because the MOS and CMOS can be fabricated in very small areas compared to the TTL.
- This family of circuits is useful for the systems with low power requirements. CMOS is still more power efficient than MOS.
- I²L, though consists of BJTs as the basic elements, the realizable circuit density is high as a different scheme of circuit fabrication than with TTL is used.

- Figure 2.3.1 shows the pin layout of some typical SSI (Small Scale Integrated) ICs of gates.

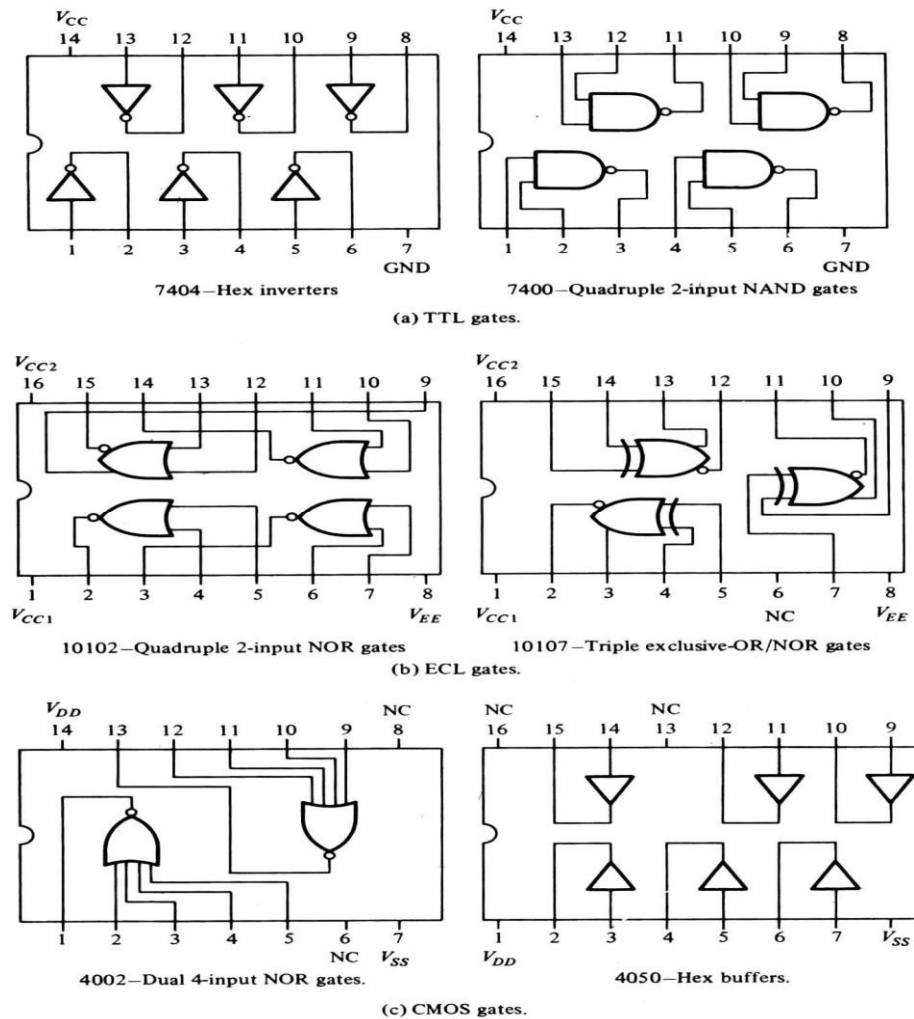


Figure 2.3.1: Pinout schemes of Some typical integrated circuit gates

Characteristics of digital logic families (Technology Parameters)

- For each specific implementation technology, there are details that differ in their electronic circuit design and circuit parameters.
- The most important parameters used to characterize an implementation technology are:

1. Fan-in

- For high-speed technologies, *fan-in*, the number of inputs to a gate, is often restricted on gate primitives to no more than four or five.
- This is primarily due to electronic considerations related to gate speed.
- To build gates with larger fan-in, interconnected gates with lower fan-in are used during technology mapping.
- A mapping for a 7-input NAND gate is made up of two 4-input NAOs and an inverter as shown in figure.

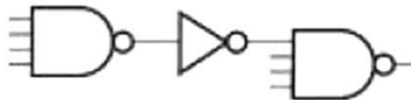


Fig: Implementation of a 7-input NAND Gate using NAND Gates with 4 or Fewer Inputs.

2. Propagation delay

- The signals through a gate take a certain amount of time to propagate from the inputs to the output.
- This interval of time is defined as the **propagation delay of the gate**.
- Propagation delay is measured in nanoseconds (ns). 1 ns is equal to 10^{-9} of a second.
- The signals that travel from the inputs of a digital circuit to its outputs pass through a series of gates.
- The sum of the propagation delays through the gates is the total delay of the circuit.
- The average propagation delay time of a gate is calculated from the input and output waveforms as:

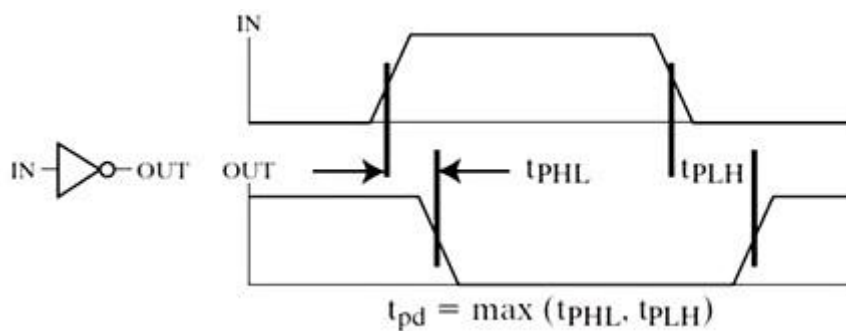


Fig: Measurement of propagation delay

- Delay is usually measured at the 50% point with respect to the H and L output voltage levels.
- High-to-low (t_{PHL}) and low-to-high (t_{PLH}) output signal changes may have different propagation delays.

- High-to-low (HL) and low-to-high (LH) transitions are defined with respect to the output, **not the input**.
- An HL input transition causes:
 - an LH output transition if the gate inverts and
 - An HL output transition if the gate does not invert.

3. Fan-out

- Fan-out specifies the number of standard loads driven by a gate output i.e. Fan-out is a measure of the ability of a logic gate output to drive a number of inputs of other logic gates of the same type.
- *Maximum Fan-out* for an output specifies the fan-out that the output can drive without exceeding its specified *maximum transition time*.
- Standard loads may be defined in a variety of ways depending upon the technology.
- For example: the input to a specific inverter can have load equal to 1.0 standard load.
- If a gate drives six such inverters, then the fan-out is equal to 6.0 standard loads.

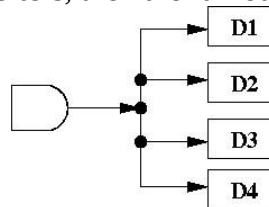


Fig: AND gate above is attached to the inputs of four other components so has a fan out of 4.

4. Power Dissipation

- Every electronic circuit requires a certain amount of power to operate.
- The power dissipation is a parameter expressed in milliwatts (mW) and represents the amount of power needed by the gate.
- The number that represents this parameter does not include the power delivered from another gate; rather, it represents the power delivered to the gate from the power supply.
- An IC with four gates will require, from its power supply, four times the power dissipated in each gate.
- The amount of power that is dissipated in a gate is calculated as:

$$P_D (\text{Power Dissipation}) = V_{CC} * I_{CC} \text{ Where } V_{CC} = \text{supply voltage and}$$

$$I_{CC} = \text{current drawn by the}$$

circuit

- The current drain from the power supply depends on the logic state of the gate.
- The current drawn from the power supply when the output of the gate is in the high-voltage level is termed I_{CH} .
- When the output is in the low-voltage level, the current is I_{CL} . The average current is

$$I_{CC(avg)} = \frac{I_{CCH} + I_{CCL}}{2}$$

And used to calculate the average power dissipation as,

$$P_D(avg) = I_{CC(avg)} \times V_{CC}$$

5. Noise Margin

- Undesirable or unwanted signals (e.g. voltages, currents etc) on the connecting wires between logic circuits are referred to as *noise*. There are two types of noise to be considered:
 - **DC noise** is caused by a drift in the voltage levels of a signal.
 - **AC noise** is a random pulse that may be created by other switching signals.
- Thus, noise is a term used to denote an undesirable signal that is superimposed upon the normal operating signal.
- **Noise margin** is the maximum noise voltage added to an input signal of a digital circuit that does not cause an undesirable change in the circuit output.
- The ability of circuits to operate reliably in a noise environment is important in many applications.
- Noise margin is expressed in volts and represents the maximum noise signal that can be tolerated by the gate.

Positive and Negative Logic and Logic Levels

- By definition of the digital logic, any digital signal has got only two levels:
- HIGH level and LOW level (except at the transitions).
- Out of the two levels, one signal value represents logic 1 whereas the other signal level represents logic 0.
- Each of the two signal values can be separately assigned to each of the logic level. Which is assigned to which may differ from product to product.
- The two signal levels should be distinct from each other for the representation of two different logic levels.
- That is, one of the two should be greater than the other.
- A positive logic system assigns the higher signal value to logic level 1 and the lower value to logic 0.
- The scheme is just reverse for the negative logic system.
- Thus the term positive or negative logic does not represent the polarity of the signal.

- It rather represents which logic level is greater in terms of signal amplitude.
- Figure 2.4.1 illustrates the notion graphically:

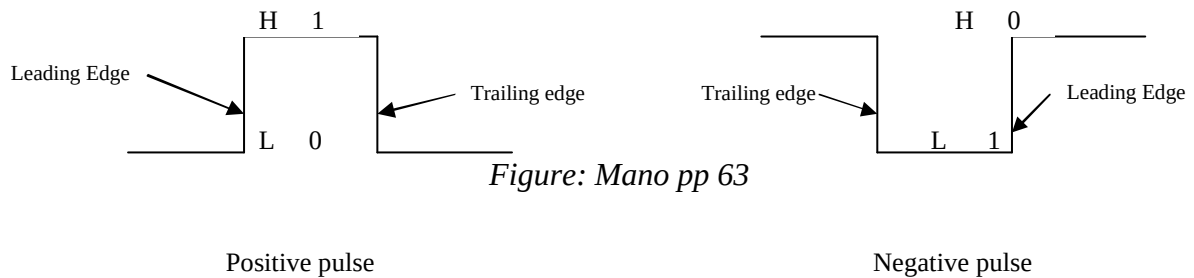


Figure 2.4.1: Signal amplitude assignment and type of logic.

- Table 2.4.1 below lists the logic levels and corresponding signal amplitudes for different families of logic circuits.

Table 2.4.1: High and Low levels in IC logic families

IC family type	Voltage supply (V)	High-level voltage (V)		Low-level voltage (V)	
		Range	Typical	Range	Typical
TTL	VCC = 5	2.4 – 5	3.5	0 – 0.4	0.2
ECL	VEE = -5.2	-0.95 - -0.7	-0.8	-1.9 - -1.6	-1.8
CMOS	VDD = 3 – 10	VDD	VDD	0 – 0.5	0
Positive logic			Logic 1		Logic 0
Negative logic			Logic 0		Logic 1

Basic theorems and Properties of Boolean algebra

Duality

- Postulates of Boolean algebra are found in pairs; one part may be obtained from the other if the binary operators and the identity elements are interchanged.
- This important property of Boolean algebra is called the **duality principle**.
- It states that “**Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged**”.
- In a two valued Boolean algebra, the identity elements and the elements of the set B are the same: 1 and 0.
- If the **dual** of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

Basic theorems

- The theorems, like the postulates, are listed in pairs; each relation is the dual of the one paired with it.

- The postulates are basic axioms of the algebraic structure and need no proof. The theorems must be proven from the postulates. six theorems of Boolean algebra are given below:

Theorem1:	Idempotence	(a) $x + x = x$	(b) $x \cdot x = x$	} theorems
Theorem2:	Existence: 0&1	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$ One variable	
Theorem3:	Involution	$(x')' = x$		}
Theorem4:	Associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$	
Theorem5:	Demorgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$	} 2 or 3 variable theorems
Theorem6:	Absorption	(a) $x + xy = x$	(b) $x(x + y) = x$	

Proofs:

(a) The proofs of the theorems with one variable are presented below:

THEOREM 1(a): $x + x = x$

$$\begin{aligned}
 x + x &= (x + x) \cdot 1 && \text{(P4: Identity element)} \\
 &= (x + x)(x + x') && \text{(P5: Existence of inverse)} \\
 &= x + xx' && \text{(P3: Distribution)} \\
 &= x + 0 && \text{(P5: Existence of inverse)} \\
 &= x && \text{(P4: Identity element)}
 \end{aligned}$$

THEOREM 1(b): $x \cdot x = x$

$$\begin{aligned}
 x \cdot x &= xx + 0 && \text{(P4: Identity element)} \\
 &= xx + xx' && \text{(P5: Existence of inverse)} \\
 &= x(x + x') && \text{(P3: Distribution)} \\
 &= x \cdot 1 && \text{(P5: Existence of inverse)} \\
 &= x && \text{(P4: Identity element)}
 \end{aligned}$$

THEOREM 2(a): $x + 1 = 1$

$$\begin{aligned}
 x + 1 &= 1 \cdot (x + 1) && \text{(P4: Identity element)} \\
 &= (x + x')(x + 1) && \text{(P5: Existence of inverse)} \\
 &= x + x' \cdot 1 && \text{(P3: Distribution)} \\
 &= x + x' && \text{(P4: Identity element)} \\
 &= 1 && \text{(P5: Existence of inverse)}
 \end{aligned}$$

THEOREM 2(b): $x \cdot 0 = 0$ by duality. inverse)

THEOREM 3: $(x')' = x$. From P5, we have $x + x' = 1$ and $x \cdot x' = 0$, which defines the complement of x . The complement of x' is x and is also $(x')'$. Therefore, since the complement is unique, we have that $(x')' = x$.

(b) The theorems involving two or three variables may be proven algebraically from the postulates and the theorems that have already been proven. For example, let's prove Demorgan's theorem:

THEOREM 5(a): $(x + y)' = x'y'$

From postulate P5 (Existence of inverse), for every x in a Boolean algebra there is a unique x' such that

$$x + x' = 1 \text{ and } x \cdot x' = 0$$

So it is sufficient to show that $x'y'$ is the complement of $x + y$. We'll do this by showing that $(x + y) + (x'y') = 1$ and $(x + y) \cdot (x'y') = 0$.

$(x + y) + (x'y') = [(x + y) + x'] [(x + y) + y']$	[OR distributes over AND (P3)]
$= [(y + x) + x'] [(x + y) + y']$	[OR is commutative (P2)]
$= [y + (x + x')] [x + (y + y')]$	[OR is associative (Theorem 3(a)), used twice]
$= (y + 1)(x + 1)$	[Complement, $x + x' = 1$ (P5), twice]
$= 1 \cdot 1$	[$x + 1 = 1$, (Theorem 2), twice]
$= 1$	[Idempotent, $x \cdot x = x$ (Theorem 1)]

Also,

$(x + y)(x'y') = (x'y')(x + y)$	[AND is commutative (P2)]
$= [(x'y')x] + [(x'y')y]$	[AND distributes over OR (P3)]
$= [(y'x')x] + [(x'y')y]$	[AND is commutative (P2)]
$= [y'(x'x)] + [x'(y'y)]$	[AND is associative (Theorem 3(b)), twice]
$= [y'(xx')] + [x'(yy')]$	[AND is commutative, twice]
$= *y' \cdot 0 + *x' \cdot 0 +$	[Complement, $x \cdot x' = 0$, twice]
$= 0 + 0$	[$x \cdot 0 = 0$, twice]
$= 0$	[Idempotent, $x + x = x$]

THEOREM 5(a): $(xy)' = x' + y'$ $\leftarrow$ can be proved similarly as in Theorem 5(a). Each step in the proof of 5(b) is the dual of its 5(a) counterparts.

x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Operator Precedence

The operator precedence for evaluating Boolean expressions is

1. Parentheses $\rightarrow ()$
2. NOT $\rightarrow '$
3. AND $\rightarrow \cdot$
4. OR $\rightarrow +$

Boolean Functions

- A Boolean function is an expression formed with binary variables (variables that takes the value of 0 or 1), the two binary operators OR and AND, and unary operator NOT, parentheses, and an equal sign. For given value of the variables, the function can be either 0 or 1.
- **Boolean function represented as an algebraic expression:** Consider Boolean function $F_1 = xyz'$. Function F is equal to 1 if $x=1$, $y=1$ and $z=0$; otherwise $F_1 = 0$. Other examples are: $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, $F_4 = xy' + x'z$ etc.
- **Boolean function represented in a truth table:**
The number of rows in the truth table is 2^n , where n is the number of binary variables in the function, The 1's and 0's combinations for each row is easily obtained from the binary numbers by counting from 0 to $2^n - 1$.

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

Boolean Algebra, De Morgan's Laws.

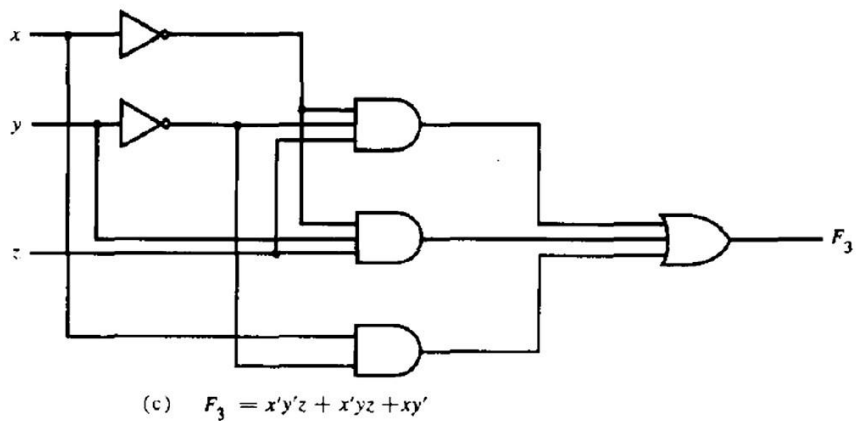
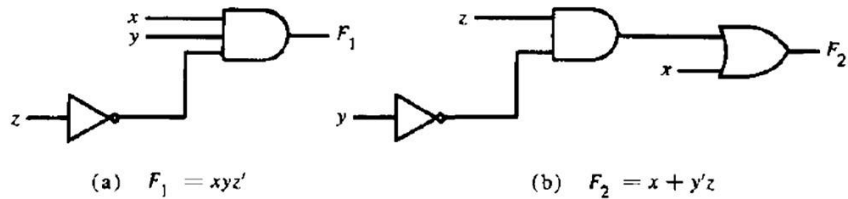
- The statement “If and only if all inputs are true (1), then the output is true (1)” is logically equivalent to the statement “if at least one input is false (0), then the output is false (0).”
- In Boolean notation, this equivalence is written as

$$ABC\dots\dots = A' + B' + C' + \dots\dots\dots [1]$$

Similarly, “If at least one input is true (1), then the output is true (1)” is logically equivalent to “If and only if all the inputs are false, then the output is false.” So,

$$A + B + C + \dots\dots = A' . B' . C' . \dots\dots$$

or $A + B + C + \dots\dots = A' . B' . C' . \dots\dots\dots [2]$

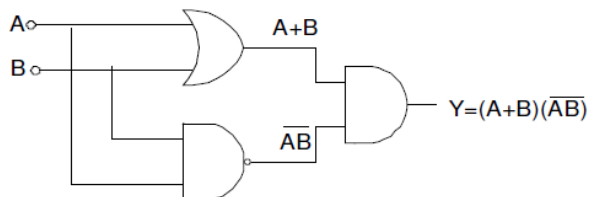
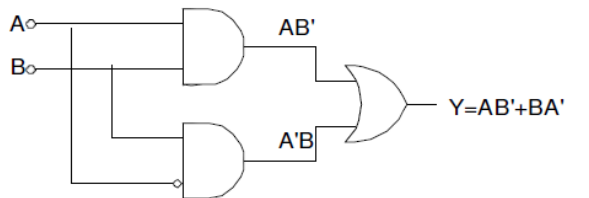


Q. Using De Morgan's laws, verify the XOR equivalence

$$Y = A \oplus B = A'B + B'A = (A + B)(AB)$$

A. Proof: $(A + B)(AB) = (A' + B)(A + B') = A'A + AB' + A'B + B'B = A'B + AB'$

Q. Represent $A\bar{B} + \bar{A}B$ and $(A + B)(\overline{AB})$ in logic block diagrams



You can swap shapes if at the same time you invert all inputs and outputs.

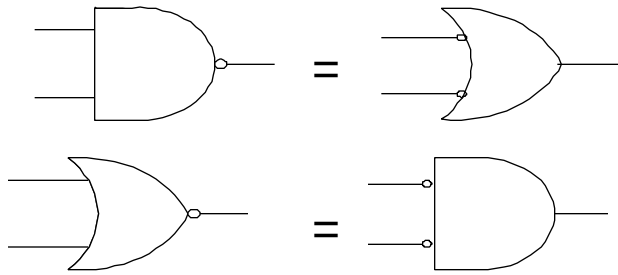


Figure 2.5.1: De Morgan's theorem in graphical form

Q. Show that $A + AB = A$

A. $A + AB = A(1 + B) = A \cdot 1 = A$

Q. Show that $(A + B)(A + C) = A + BC$

A. $(A + B)(A + C) = AA + AB + AC + BC = A + AB + AC + BC = A + BC$

Q. Show that $A + \overline{A}B = A + B$

=====

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

Summary of Basic Boolean Identities:

Fundamental Laws

OR

$$A + 0 = A$$

$$A + 1 = 1$$

$$A + A = A$$

$$A + A' = 1$$

AND

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

$$A \cdot A' = 0$$

NOT

$$A + A' = 1$$

$$A A' = 0$$

$$A = A$$

Associative Laws

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

$$(A.B) C = A (B.C)$$

Commutative Law

$$A+B=B+A$$

$$AB=BA$$

Distributive Law

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

De Morgan's Laws

$$ABC..... = A' + B' + C' + ...$$

$$A + B + C + ... = A' . B' . C'$$

Auxiliary Identities

$$A + AB = A$$

$$A + A' B = A + B$$

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

Application

Consider a case where a door can be locked or unlocked from inside and has to be opened. If y represents the “lock” condition, x, an attempt to open from inside, and z, an attempt to open from outside, then the situation can be represented by the truth table shown. The door can be opened from inside at any time, but to open it from outside, the door should not be locked. That is, F = 1 (F is the event of the door getting opened) when x = 1, but F = 1 only when z = 1, but y = 0.

Table 2.5.1 Truth table for the door example

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The truth table can be converted into a Boolean function as

$$\begin{aligned}
 F &= \bar{x} \bar{y} z + x \bar{y} \bar{z} + x \bar{y} z + xy \bar{z} + xyz \\
 &= (x \bar{y} z + x \bar{y} \bar{z}) + (xyz + xy \bar{z}) + \bar{x} \bar{y} z \\
 &= x \bar{y} (z + \bar{z}) + xy (z + \bar{z}) + \bar{x} \bar{y} z \\
 &= x \bar{y} + xy + \bar{x} \bar{y} z \\
 &= x (y + \bar{y}) + \bar{x} \bar{y} z \\
 &= x + \bar{x} \bar{y} z \\
 F &= x + \bar{y} z
 \end{aligned}$$

This Boolean function can be transformed from an algebraic expression into a logic diagram composed of AND, OR and inverter gates, as shown in figure.

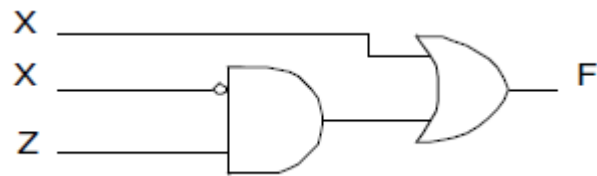


Figure 2.5.2 : Logic realization of the door example

- The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits. It provides a convenient tool to
 - express in algebraic form a truth table relationship between variables 2.
 - express in algebraic form the input-output relationship of logic diagrams
 - find simpler circuits for the same function.