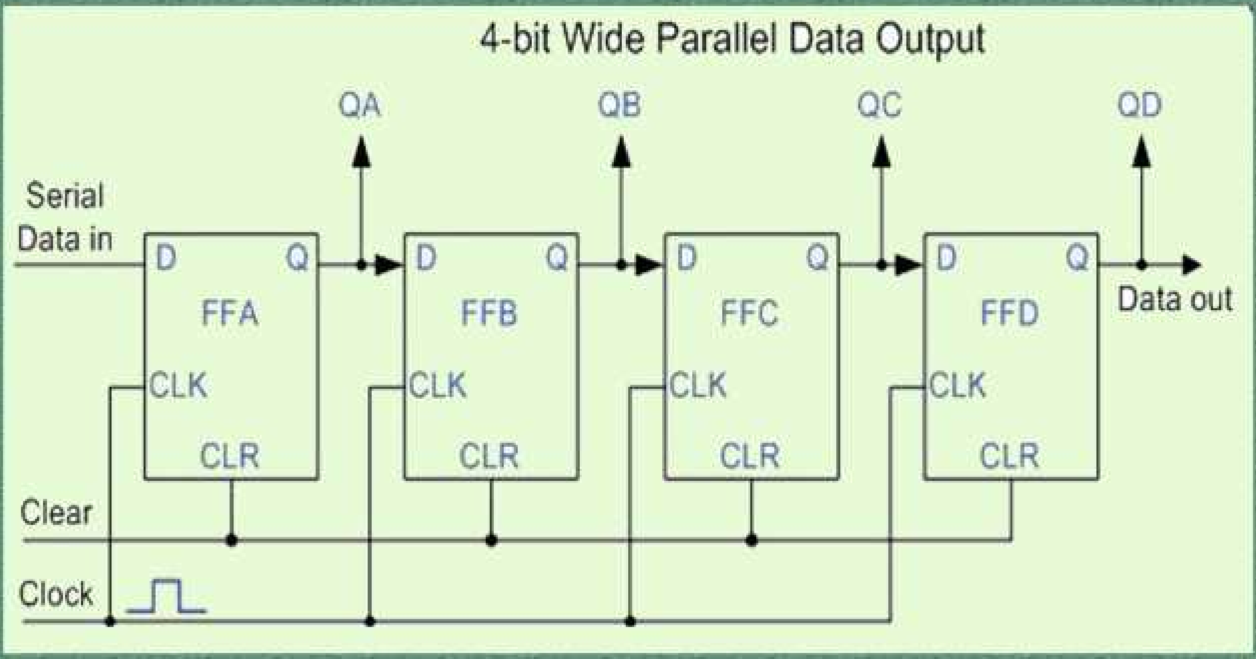
इरिसेट IRISET

TB3

DIGITAL ELECTRONICS



Indian Railways Institute of
Signal Engineering and Telecommunications

SECUNDERABAD - 500 017

TB3

DIGITAL ELECTRONICS

INDIAN RAILWAYS INSTITUTE OF SIGNAL ENGINEERING & TELECOMMUNICATIONS, SECUNDERABAD - 500 017

Issued in July 2014

TB3

DIGITAL ELECTRONICS

CONTENTS

Prepared by	M. V. Srikantha Raju, ICT-1
Approved by	C. K. Prasad, Professor - Tele
DTP and Drawings	K.Srinivas, JE (D)
No. of Pages	77
No.of Sheets	40

CHAPTER-1

DIGITAL ELECTRONICS

1.1. Introduction

In the present world digital electronics is the modern technology that is used most widely in almost every walk of human life. We find its applications in every field right from a simple calculator to the most intelligent space study satellites. Electronics has almost become omnipresent in this world. In electronics there are two main branches –

a) Analogue Electronics and
b) Digital Electronics

Let us first understand what is the basic difference between the terms analogue and digital.

1.2. What is Analogue?

The term analog refers to "the quantities which are continuous in nature and the changes in them occur in a gradual manner within a range of values". Another property of analog quantities is that they are real quantities.

In electronics, systems which are capable of processing electrical signals that change continuously over a range of values are called **Analog Systems**. An analog quantity may be represented by voltage or current that is proportional to the value of the quantity.

All natural quantities present in the real world are analogue in nature. Some examples of analogue quantities present in the nature are:

1. Day & night Temperature
2. Atmospheric Pressure
3. Flow of water in a canal
4. Change of Seasons in an year

We also come across analogue quantities in various devices and equipment used by us in day to day life, like

a) Accelerator of an automobile
b) Volume Control of a Radio or TV.
c) Train movement

1.3. What is Digital?

Unlike analogue *the quantities or systems that can assume only certain fixed values or states are called as **Digital***. This means the change in values of any digital quantity is not continuous but it is abrupt or in steps, among some fixed or discrete values.

Systems which process discrete values are called **digital systems.** In digital representation the quantities are represented not by proportional values but by symbols called **digits**.

For example, consider the digital watch, which provides the time of the day in the form of decimal digits representing hours and minutes. As we know, time of day changes continuously, but the digital watch reading does not change continuously; rather, it changes in steps of one per minute (or per second). In other words, '***time of day***' digital representation changes in *discrete steps*, as compared to the representation of time provided by an analog watch, where needles pointing time change continuously on the dial.

Some examples of devices or quantities which are digital in their behavior or construction and which we come across in our day to day life are given below.

S. No	Quantity	Fixed Levels/States Assumed
1	Toggle Switch	ON and OFF
2	Relay	Pick-up and Drop states
3	Lamp	Glowing or not glowing
4	Fan Regulator	1 to 5 steps of speed
5	Ladder/Staircase	Steps
6	Automobiles gear system	Speed gears

Table 1.1 Examples of Digital Quantities

1.4. Advantages of Digital Technology

The following are some of the main advantages of digital technology over the analog technology.

- Programming Facility is available in Digital systems.
- Through programming operations can be automated.
- Information can be stored and also processed as required.
- Easy and efficient storage of information
- Greater accuracy and precision is possible.
- Information transmission is easy and 100% accurate regeneration of it.
- Digital circuits are immune to noise, as long as the noise is not large enough to prevent them from distinguishing **High** and **Low** levels.
- Design of digital circuits is very easy as exact values of voltage and current are not required.

1.5. Limitations of Digital

Most physical quantities in real world are analog in nature and when these are to be processed by digital systems they must be converted to digital form and sometimes even re-conversion to analog form after processing is needed. For this purpose Analog to Digital Converters (ADC) and Digital to Analog Converters (DAC) or CODECs are used. Hence, there is possibility that the accuracy of conversion of these devices may not be 100% of the original signal.

1.6. Digital Systems

Now it should be clear that in electronics a Digital device or System is the one that operates on or represents only certain fixed voltage values or levels.

If a digital system has to process more number of such voltage levels then its circuit design becomes very difficult. Because the components like transistors, diodes, MOSFETs, LEDs, relays, switches etc, cannot differentiate between more numbers of voltage levels.

Hence, it is easy and practicable to operate digital circuits only at two fixed voltage levels - a low voltage level and a high voltage level. The voltage values representing these two levels are generally a 0V and Vcc (which may be 5V, 10V, 15V etc).

A digital signal representing only two voltage levels is also called as a **Binary Signal**. “Binary” means two-valued.

1.7. Concept of 0 and 1

Many of the digital devices involve mathematical processes like counting, arithmetic etc. In such cases representation of digital signals in terms of low and high voltages is not practicable. It leads to ambiguity and makes the procedure useless.

Hence an alternate way of representing these digital signals became a must. This alternate way of representation should be in numerals so as to enable mathematical operations.

Then came the representation of digital signals in the following manner.

Low voltage signal ----	‘0’
High voltage signal -----	‘1’

This notation of digital signals in terms of 0 and 1 has lead to the need of a special number system which contains only two digits (or symbols) 0 and 1.

1.8. Number Systems

In addition to the conventional **decimal number system** digital circuits have led to the use of the following two new number systems.

1) Binary Number system and
2) Hexadecimal Number system.

Before going into the details of these two, let us study the general concepts or rules applicable to every number system. In any number system there should be a set of symbols, which we call digits or numbers, to be used for counting quantities or in calculations. In decimal system these symbols are 0,1,2,3,4,5,6,7,8 and 9. These are 10 in total. This 10 is called the base of decimal number system.

We can represent any value using these 10 numerals, like:

25, 37, 105, 1145, etc.

The same number, when it changes position in a multi digit number, its value changes in accordance with the value of its positional multiplier. The order of positional multipliers for each digit in decimal number system is as below.

$$10^n, \ldots 10^4, 10^3, 10^2, 10^1, 10^0 \underset{\substack{\uparrow \\ \textbf{Decimal Point}}}{.} 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, \ldots 10^{-n}$$

In the above series, the values of positional multipliers change in steps of 10, which is the base for the decimal number system.

Thus the value 347.45 is equal to the sum of positional values of:

$3\times10^2 + 4\times10^1 + 7\times10^0 + 4\times10^{-1} + 5\times10^{-2}$. This gives

$300 + 40 + 7 + 0.4 + 0.05 = 347.45$

This same rule is applicable to every number system.

1.8.1. Binary Number System

In binary number system the digits available are only two, i.e. ‘0’ and ‘1’. That means its base is 2. The binary sequence of numbers equivalent to decimal values can be seen in the Table 1.2 given below.

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

Table 1.2 Decimal digits and their Binary equivalents

And so on. The values of positional multipliers in binary system are as below.

$$2^n, 2^4, 2^3, 2^2, 2^1, 2^0. 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, \dots 2^{-n}$$

To know the value of any binary number we have to multiply the respective digits with their positional multipliers.

For example: 1101 (in binary). To know its value assign positional multipliers as shown below

1	1	0	1	- digits in binary number
2^3	2^2	2^1	2^0	- positional digit multipliers

This is equal to: $1x2^3 + 1x2^2 + 0x2^1 + 1x2^0$ Gives, 1x8 + 1x4 + 0 + 1x1 = 13

Therefore, the value of binary 1101 = 13 in decimal.

Like this the value of any binary number can be known in decimal. This procedure gives conversion from binary to decimal.

Similarly, if a decimal number is given, its binary equivalent can be known by a method called “**double dabble**”. In this method the given decimal number is to be divided by 2, the base of binary.

Suppose we want 16 in binary then:

		Quotient	Reminder
16/2	=	8	0 ↑
8/2	=	4	0
4/2	=	2	0
2/2	=	1	0
1/2	=	0	1

Now, read the value in the direction shown by the arrow, i.e. 10000

1.8.2. Hexadecimal Number System

Hexadecimal number system base is 16. That means 16 digits are available for counting.

These are 0,1, 2, 3, 4, ------- 9, A, B, C, D, E, F

Positional multipliers are: 16^n, --------, 16^4, 16^3, 16^2, 16^1, 16^0, 16^{-1}, 16^{-2}, --------, 16^{-n}

The main use of this number system is to express very long binary values in shorter form with hexadecimal digits.

For example: Consider 110110110011

This binary number can be written as DB3 by taking 4 bit groups starting from least significant bit (right most digit) of binary and writing its value in equivalent hexadecimal digits as shown below

```
1101   1011   0011
  D      B      3
```

1.8.3. Denoting Hexadecimal Numbers

Hexadecimal values are represented using either of the following two conventions.

a) 2300h or 2300H - suffix h **(**or **H)** indicates that 2300 is a Hex value.

b) 0x2300 - prefix 0x is also used in lieu of suffix **h**.

1.8.4. Representation of Positive and Negative Numbers in Binary

The following three methods are used for representing positive and negative decimal values in binary form.

i) Sign-bit method
ii) 1's Complement method
iii) 2's complement method

1.8.4.1. Sign-bit Method

In this method the most significant bit (MSB) is used to denote **sign** (+ or -) of the number. MSB is 0 to represent positive (+) numbers and 1 to represent negative (-) numbers. The remaining bits representing the **magnitude** of the number in binary remain same for both positive and negative values.

Example: When 7 is to be represented in 8 bit binary with sign:

```
+7     00000111
-7     10000111
```

An n-bit binary number can represent decimal numbers in the range $-(2^{n-1} -1)$ to $+(2^{n-1} -1)$. Hence, in 8-bit signed representation decimal numbers in the range -127 to +127 are represented.

1.8.4.2. 1's Complement Method

In this method positive numbers representation is same as in sign-bit method but the negative number is represented as the 1's complement of its positive value. In this method also the MSB is used to differentiate between positive and negative values.

Example: The same 7 is represented in both positive and negative values in 8-bit as

+7	00000111
-7	11111000

In this method of representation the range of numbers represented by a n-bit binary is - (2^{n-1} -1) to +(2^{n-1} -1), the same as in sign-bit method. An 8-bit representation covers decimal values from -127 to +127.

1.8.4.3. 2's Complement Method

This method is same like 1's Complement method except for the 2's **Complement representation** of negative numbers. Again here also MSB is used as the sign bit.

Example: If you take the same 7 again, it is represented in both positive and negative values as

+7	00000111
-7	11111001

Let us work out the value of -7, which is given above as 11111001. Binary value of 7 is 0000111 (7 bits). Its 1's complement is 1111000. By adding +1 to this gives us its 2's complement which is 1111001. Add 1 as 8th bit (MSB) to indicate that it is a negative value. Then the final form of -7 in 2's complement method is 11111001.

The range of numbers represented in this method by a n-bit binary is -(2^{n-1}) to +(2^{n-1}-1). Hence, an 8-bit representation covers decimal values from -128 to +127. This inequality in the ranges of +ve and -ve values can be better understood from the Table 1.3 given below.

Sign Bit	Value Bits of Number							Decimal Value Represented	
0	1	1	1	1	1	1	1	=	127
0	1	1	1	1	1	1	0	=	126
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

Table 1.3 Representation of integers in two's-complement method in 8-bit form

In all the above three methods representation of positive values is same, there is no difference at all. But the main difference comes when negative values are represented. Only 2's complement method is used in all computers and microprocessor systems for representing negative values.

Important Binary Terms

Bit	- Each binary digit in a binary value is called as a **Bit**
Byte	- A group of 8 bits is called as a Byte
Word	- A group of bits that can be accessed at a time in parallel by a central processing unit (CPU).

Objective Questions

1. A digital quantity is one which represents only some ________ values
2. An analogue quantity is one which represents ________ values
3. Toggle switch is equivalent to ______________________
4. Staircase and Ramp are examples of ________ and ____________ quantities respectively.
5. ____________________________method is used for converting decimal values into binary.
6. The range of values that can be represented by 1’s complement method is ______________
7. In 2’s complement method the range of values covered is _______________________

Review Questions

1. Define Analogue and Digital quantities? Give some examples of both.
2. Why Binary numbers are used in digital circuits?
3. Convert the following values binary.

   25, 37, 48, 59
4. Convert the following binary values into Decimal.

   11011, 101010, 111000
5. What is Double Dabble method and where is its use?

CHAPTER-2

LOGIC GATES

2. Introduction

Logic gates are the elementary circuits in digital electronics. The term 'Logic' means '*a method of reasoning or argument that presents to show its conclusion to be true'*. True to this meaning each logic gate has its own method of working for its output value. There are total seven logic gates which have lead to the development of digital electronics as a separate branch of the field of electronics. These gates are listed below.

1) OR Gate
2) AND Gate
3) NOT Gate
4) NOR Gate
5) NAND Gate
6) Exclusive-OR Gate
7) Exclusive-NOR Gate

2.1. Basic Gates

All these logic gates are developed from just three ***fundamental*** or **basic** logic operations only.

These three basic logics are:

1) OR logic
2) AND logic
3) NOT or Inverse logic

The electronic circuits which implement these three basic logic operations are called basic logic gates and are as given below.

1) OR gate 2) AND gate and 3) NOT gate

2.1.1. OR Gate:

This gate (circuit) has a minimum of two inputs and one output terminal. It is common practice to represent an OR gate by the symbol which is given below. (All logic gates are represented by fixed graphical symbols called ANSI symbols).

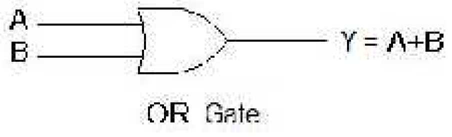


Fig.2.1 Symbol of OR gate

A & B are inputs and Y is output. Both output and input signals are only the binary 0 (Low) and 1(High) signals. Governed by rules of OR logic the output Y may become 0 or 1 depending on the values of both the inputs, A and B.

As per this logic the output ***Y becomes '1' only when either A or B is 1,*** *at the least*. This means at least one of the inputs should be 1 to get output as 1. This logic is expressed mathematically as Y = A + B. This equation is called Boolean logic equation for OR gate. The '+' sign is to be pronounced as 'OR'.

Logic Equation of OR gate Y= A+B

Truth Table

Table 2.1 called Truth Table is used to represent the input and output relation in a systematic way. All the possible input conditions are written in ascending binary order, along with their respective outputs.

Inputs		Output
A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1

Table 2.1 Truth Table of OR Gate

If an OR gate has more than 2 inputs, say 3 then its Boolean equation is written as Y = A + B + C and its symbol is shown as below.

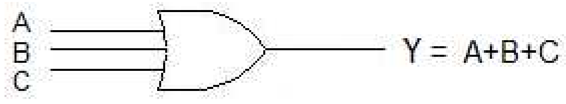


Fig.2.2 OR gate with 3-inputs

Similarly for any numbers of inputs an OR can be used.

2.1.2. AND Gate:

It also has a minimum of two inputs. Its graphic symbol and Boolean equation are as given below.

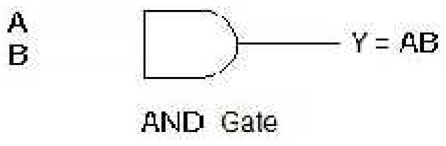


Fig.2.3 Symbol of AND gate

Logic Equation Y = A.B or simply, Y = AB

Truth Table

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

Table 2.2 Truth Table of AND Gate

The logic for functioning of this gate is "its ***output Y becomes '1' (High) only when A and B inputs are '1s' (High)***". That means all the inputs should have '1s' to give '1' at the output irrespective of number of inputs.

Its logic equation Y = A.B is to be pronounced as Y equals to A ***and B.***

2.1.3. NOT Gate:

Unlike the other two gates it has only one input. Its logic is also simple. It reverses the input signal and outputs its opposite value, i.e. ‘0’ is reversed to ‘1’ and ‘1’ is reversed to ‘0’. Its symbol and Boolean equation are as follows:

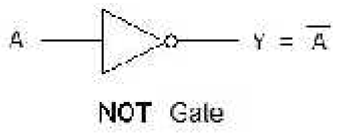


Fig.2.4 Symbol of NOT gate

Logic Equation of NOT gate $Y = \overline{A}$ ($\overline{A}$ is called the inverse of A.)

Truth Table of NOT Gate

A	Y
0	1
1	0

Table 2.3 Truth Table of NOT Gate

Number of other gates are derived from these **three fundamental** or **Basic** gates (logics) as can be seen below. Because of this these three gates are called as **Basic Logic Gates**.

2.2. Combination Gates

By the combination of these three basic logics four more logic gates, called as combination gates, are developed each of which has got its own importance.

2.2.1. NOR Gate: NOR gate is the combination of OR and NOT gate as can be seen below.

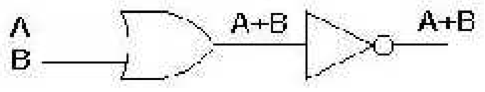


Fig.2.5 Logic of NOR

This combination is called as NOR gate and is given the following symbol.

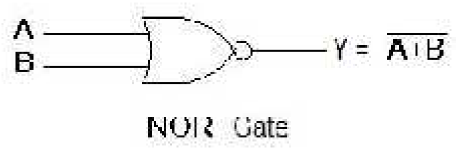


Fig.2.6 Symbol of NOR gate

Logic neither Equation of NOR Gate $Y = \overline{A+B}$

This means its output is always reverse to that of an OR gate for the same input values. The line above A+B in its Boolean equation is pronounced as ‘bar’.

Neither Truth Table of NOR Gate

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

Table neither 2.4 Truth Table of NOR Gate

2.2.2. NAND Gate: Functionally, it is the combination of AND and NOT gates.

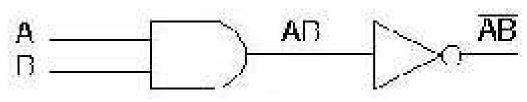


Fig.2.7 Logic of NAND

Its ANSI symbol is

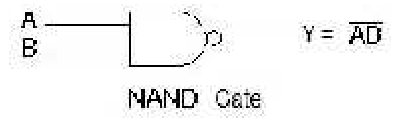


Fig.2.8 Symbol of NAND gate

Logic Equation $Y = \overline{AB}$

Truth Table

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

Table 2.5 Truth Table of NAND Gate

It should be noted that its output is always opposite or reverse to that of an AND gate for the same input values.

2.2.3. Exclusive OR Gate: It is also called in short as Ex-OR gate.

Logic of Ex-OR gate: Its logic is to check for the parity of number of High inputs (1s) to this gate. Parity means the value obtained when the number of 1s in the input to the gate are counted. If this parity comes in ODD value the gate output becomes High (1), otherwise it is Low (0).

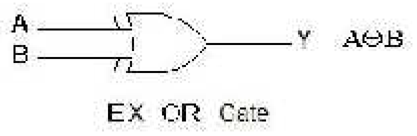


Fig.2.9 Symbol of EX-OR gate

Logic Equation

Its Logic or Boolean equation is written as $Y = A \oplus B$.

The symbol $\oplus$ just represents the Ex-OR operation. But this symbol has got no meaning mathematically. Hence, $Y = A \oplus B$ is represented by the following mathematically valid equation.

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

Truth Table

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

Table 2.6 Truth Table of Ex-OR Gate

Its logic of functioning is the count of number of ‘1’ input signals. If it is an odd value output Y goes to ‘1’, otherwise ‘0’.

2.2.4. Exclusive NOR Gate: Also known as Ex-NOR gate. It is combination of Ex-OR and NOT gates.

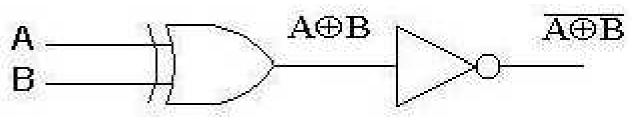


Fig. 2.10 Logic of Ex-NOR

Its symbol, logic equations and truth table are as below.

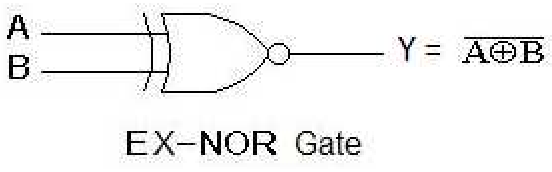


Fig.2.11 Symbol of EX - NOR gate

Logic Equation $Y = \overline{\overline{A}B + A\overline{B}}$ or $Y = \overline{A}\,\overline{B} + AB$

Truth table

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

Table 2.7 Truth Table of Ex-NOR Gate

Its logic is simply reverse to that of Ex-OR gate.

2.3. UNIVERSAL GATES

Among all the logic gates NOR gate and NAND gate are known as universal gates. The reason for this is all other gates or logic functions can be implemented by only NOR or only NAND gates. This is why they are called universal gates.

The following two theorems known as Demerger’s Theorems can explain better the functionality of universal gates.

Demerger’s Theorems

1) $\overline{A + B} = \overline{A}\;.\;\overline{B}$

2) $\overline{A\;.\;B} = \overline{A} + \overline{B}$

1. DeMorgan’s First Law (for NOR)

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

The left hand side term in this equation is the Boolean equation of NOR gate. According to this law the logic outputs of the following two gates are same for the same inputs. The logic equivalent neither of NOR as per this theorem is shown in the Fig. 2.12 given below.

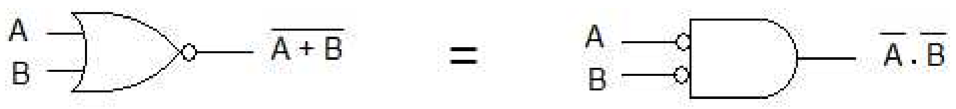


Fig. 2.12 DeMorgan's 1st Law

2. DeMorgan's Second Law (for NAND)

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

The left hand side term in this equation is the Boolean equation of NAND gate. According to this law the logic outputs of the following two gates are same for the same inputs. The logic equivalent of NAND as per this theorem is shown in the Fig. 2.13 given below.

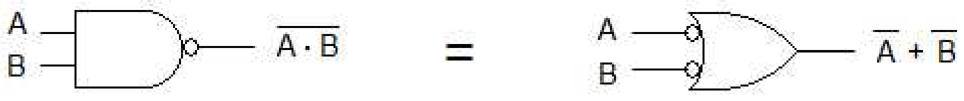


Fig.2.13 DeMorgan's 2nd Law

The following are some examples of using NOR and NAND gates to obtain other logic functions.

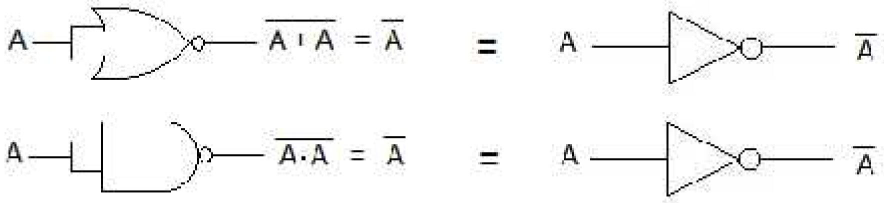

A
$\overline{A+A} = \overline{A}$
=
A
$\overline{A}$
A
$\overline{A \cdot A} = \overline{A}$
=
A
$\overline{A}$

Fig.2.14 Using Universal Gates for NOT Logic

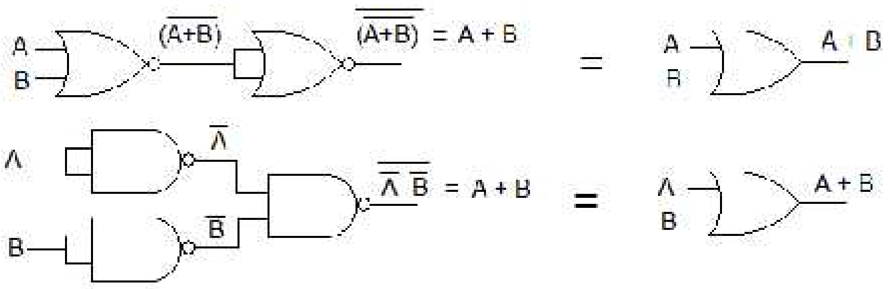

A
B
$(\overline{A+B})$
$(\overline{\overline{A+B}}) = A+B$
=
A
B
A+B
A
$\overline{A}$
B
$\overline{B}$
$\overline{\overline{A}\,\overline{B}} = A+B$
=
A
B
A+B

Fig.2.15 Using Universal Gates to Implement OR Logic

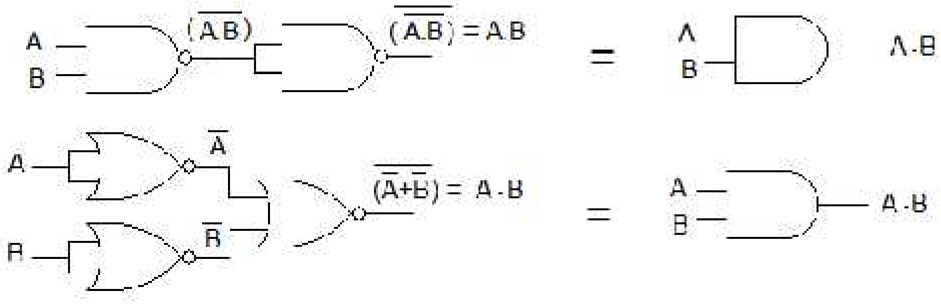

A
B
$(\overline{AB})$
$(\overline{\overline{AB}}) = AB$
=
A
B
A.B
A
$\overline{A}$
B
$\overline{B}$
$(\overline{\overline{A}+\overline{B}}) = A.B$
=
A
B
A.B

Fig.2.16 Using Universal Gates to Implement AND Logic

2.4 Multiple Input Gates

Logic gates can be developed with any number of inputs. The following are some examples of multiple input gates.

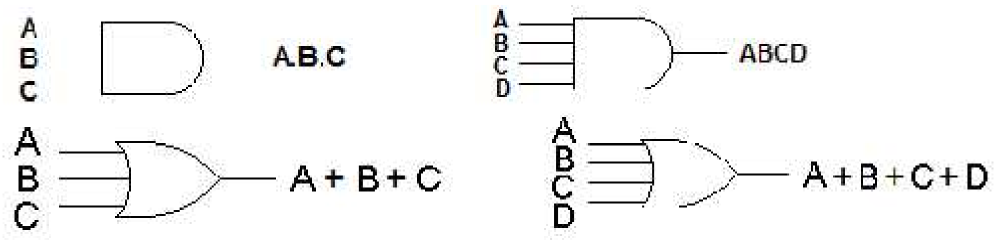


Fig. 2.17 - Multiple input Gates

2.5 Making Multi-Input Gates

Multi input gates can be made by joining gates of the same type but with less number of inputs than required. The diagrams below show how a three input AND gate and a four input AND gate can be made out of two input AND gates.

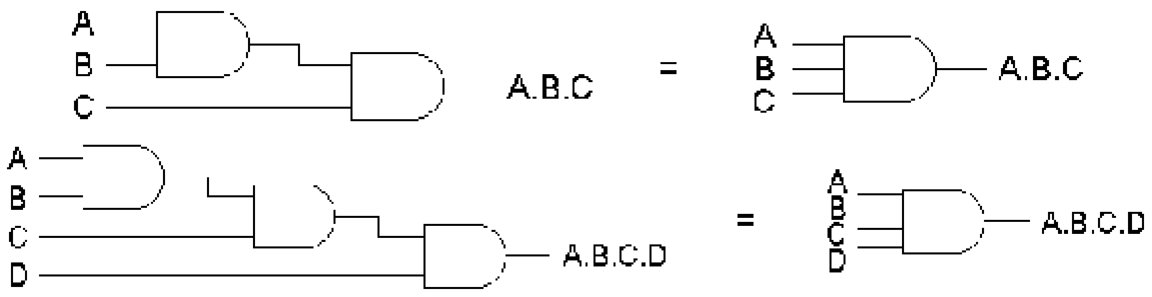


Fig.2.18 Making multi-input gate using 2-input gates

2.6 Laws of Boolean algebra

In addition to the Demerger's Theorems the following basic laws are used for solving and simplifying logic expressions and also for analyzing logic circuits. These are known as Laws of Boolean Algebra.

1. Commutative Law
2. Associative Law
3. Distributive Law
4. Identity Law
5. Duality Law
6. Precedence Law
7. Redundancy Law
8. De Morgan's Theorems

Note that every law has two expressions, (a) and (b). These are obtained by changing every AND(.) to OR(+), every OR(+) to AND(.) and all 1's to 0's and vice-versa. It has become conventional to drop the **.** (AND symbol) i.e. A**.**B is written as AB.

2.7 Boolean Laws

T1: Commutative Law

(a) $A + B = B + A$
(b) $A B = B A$

T2: Associate Law

(a) (A + B) + C = A + (B + C)
(b) (A B) C = A (B C)

T3: Distributive Law

(a) A (B + C) = A B + A C
(b) A + (B C) = (A + B) (A + C)

T4: Identity Law

(a) A + A = A
(b) A . A = A

T5: Duality Law

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

T6: Precedence Law

(a) A·B+C = (A·B) + C
(b) A+B·C = A + (B·C)

T7: Redundancy Laws

1) (a) A + A B = A
(b) A (A + B) = A
2) (a) 0 + A = A
(b) 0 . A = 0
3) (a) 1 + A = 1
(b) 1 . A = A
4) (a) $\overline{A} + A = 1$
(b) $A \,.\, \underline{A} = 0$
5) (a) $A + \overline{A}\,B = A + B$
(b) $A(\overline{A} + B) = A\,B$

De Morgan's Theorem

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

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

2.4. Boolean Postulates

A number of rules, called Boolean postulates can be derived from these relations as given below.

P1:	**X means it = 0 or 1**
P2:	**0. 0 = 0**
P3:	**1 + 1 = 1**
P4:	**0 + 0 = 0**
P5:	**1. 1 = 1**
P6:	**1. 0 = 0. 1 = 0**
P7:	**1 + 0 = 0 + 1 = 1**

Objective Questions

1. OR gate is one of the _______________ gates
2. NOR gate is called _____________________ gate
3. Logic of EX-OR gate is ________________ parity
4. The logic gate which inverts its input is ________________
5. NAND is equivalent to a __________ gate plus a _____________ gate

Review Questions

1. What is a Logic gate? Mention names of different Logic gates.
2. What do you understand by the term ‘Basic gates’? explain about them.
3. Explain about any two combinational gates.
4. What is a universal gate? Explain about them.
5. Write down DeMorgan theorems and explain about their usefulness along with some examples.

CHAPTER-3

COMBINATIONAL LOGIC CIRCUITS

3. Combinational Logic Devices

The circuits which are developed using logic gates are called as combinational logic circuits. The Most important of combinational logic circuits are:

1) Decoders
2) Adders and Subtractions.
3) Multiplexers & Demultiplexers

3.1. Decoder:

Decoder is a digital device which selects only one output at a time out of a number of outputs, as per the binary value applied on the input terminals.

Suppose, 'n' inputs are available, then the number of outputs is equal to 2^n. Each output is named serially starting from 0 and ending 2^{n-1}. That is 0,1,2,3, -------- 2^{n-1}.

The following diagram shows a 2 input and 4 output decoder.

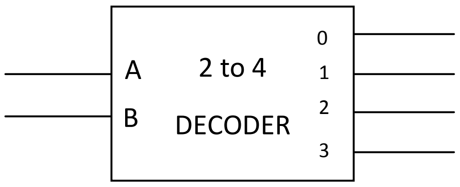


Fig. 3.1 Symbol of 2-4 Line Decoders

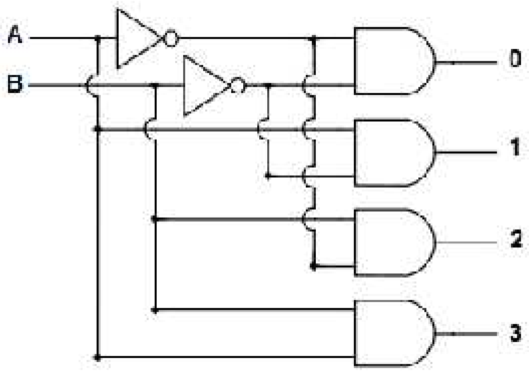
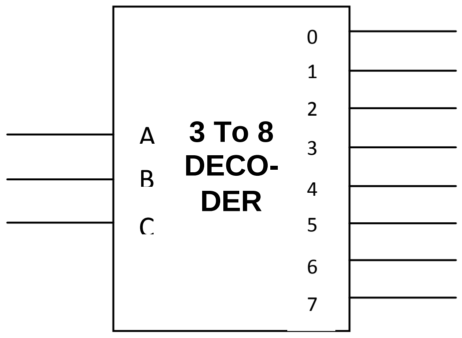


Fig.3. 2 2 - to - 4 Line Decoder

Inputs		Outputs			
A	**B**	**0**	**1**	**2**	**3**
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table 3.1- Truth Table of 2-to-4 Decoder

Similarly a 3 input decoder diagram is as below:



Inputs			Outputs							
A	**B**	**C**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Fig. 3.3 - 3 to 8 decoder

Table 3.2 – Truth Table of a 3 to 8 line Decoder

3.2. Applications of Decoder

Decoders are a must in all microprocessors, microprocessor based systems and in all memory chips for selecting the individual functional units or devices. The following figure 3.4 shows the scheme for selecting individual memory registers inside a memory chip.

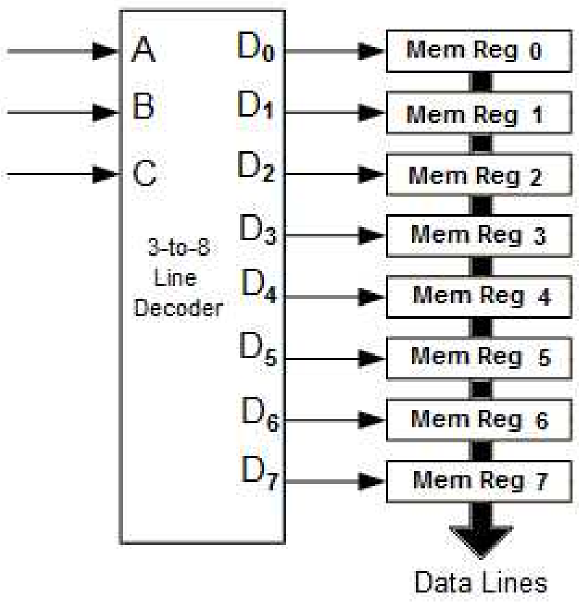


Fig. 3.4 - Memory Register Decoder of a Memory Chip

In the similar way decoders are used in microprocessor systems for selecting individual peripheral chips through chip select signals.

3.3. Adder:

It is an arithmetic circuit for performing additions of binary bits (numbers). Two types of adders are available.

1) Half Adder
2) Full Adder

3.3.1. Half Adder:

It adds any two binary bits at a time. X and Y are bits from two different binary numbers that are to be added. Its diagram is as below:

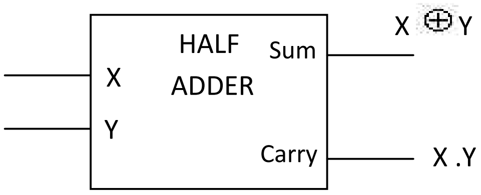


Fig. 3.5- Half Adder

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 3.3 -Truth Table of Half Adder

Example:

X	0	0	1	1
+Y	+ 0	+ 1	+ 0	+1
S	0	1	1	0
C	0	0	0	1

S = Sum Output

C = Carry Output

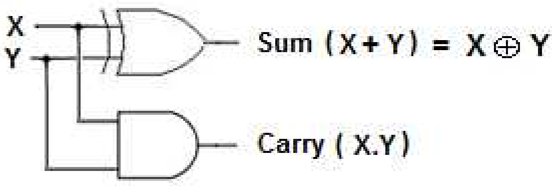


Fig.3.6- Logic Diagram of Half Adder

3.3.2. Full Adder:

Unlike Half Adder a Full adder can add 3 bits at a time. X, Y inputs are similar bits in the two numbers to be added and CIN input is carry bit produced in the addition of bits in immediate lower position in a multi-bit number.

X	Y	C_{IN}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

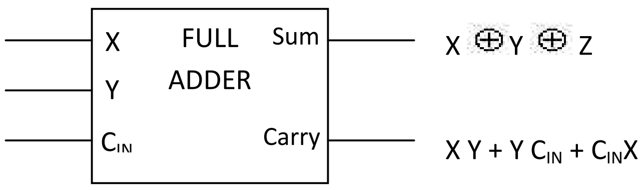


Fig.3.7- Full Adder

Table 3.4- Truth Table of Full Adder

3.4. Subtractor:

Like adders the Subtractions are also of two types.

1) Half Subtractor
2) Full Subtractor

3.4.1. Half Subtractor: It performs subtraction of two bits.

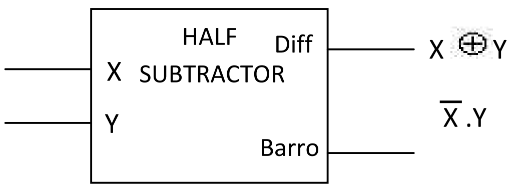


Fig.3.8- Half Subtractor

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 3.5- Truth Table of Half Sub tractor

Example:

X	0	0	1	1
-Y	-0	-1	-0	-1
D	0	1	1	0
B	0	1	0	0

D - Difference Output
B - Borrow Output

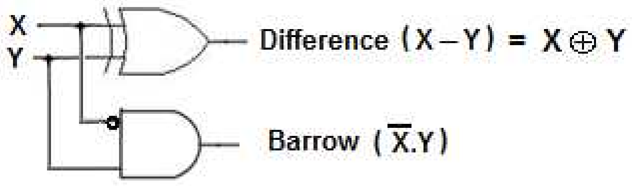


Fig.3.9- Logic Diagram of Half Sub tractor

3.4.2. Full Subtractor: A full sbtractor is designed for accommodating a 3rd input bit which is a barrow bit needed for subtraction in previous bit position. To account for such barrow bits a 3rd input (Bi) is provided in Full-subtractor. As the barrow bit is also a negative bit both Y and Bi are summed up and this sum is then subtracted from X to give the difference value, which evident from the equation given below in Fig.3.10.

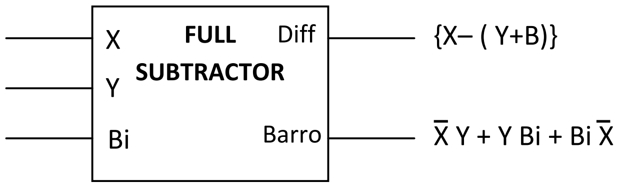


Fig.3.10- Full Subtractor

X	Y	Bi	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 3.6 - Truth Table of Full Subtractor

3.5. Multiplexer

*A digital **Multiplexer** is a device which facilitates connectivity between many inputs and a single output selectively.* That is, only anyone input line at a time can be connected to the output line, by input selection logic. The below given Fig.1 explains the logic of the multiplexer in a simple manner.

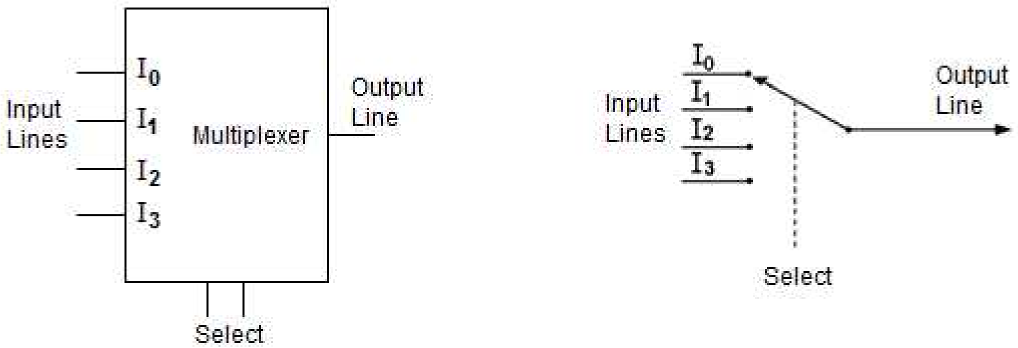


Fig.3.11 - Multiplexer Symbol and its Logic

3.6. A 4-bit Multiplexer

A 4-bit (input) multiplexer logic circuit using gates and a decoder is shown in the Figure 3.12 given below. The use of decoder for selecting input lines can be clearly seen in the diagram. The decoder is used to enable the AND gate which passes on a required input (among I_0 - I_3) to the output line. This is done by applying a binary value on select lines, which is equal to the serial number of the input line to be selected. For example signal on input I_2 is to be passed on to the output line. Then we need to apply binary 10 on the select lines. This enables the 3rd AND gate by placing a High on its 2nd input leg and thus makes it ready to pass on any digital signal that is applied to I_2 input line.

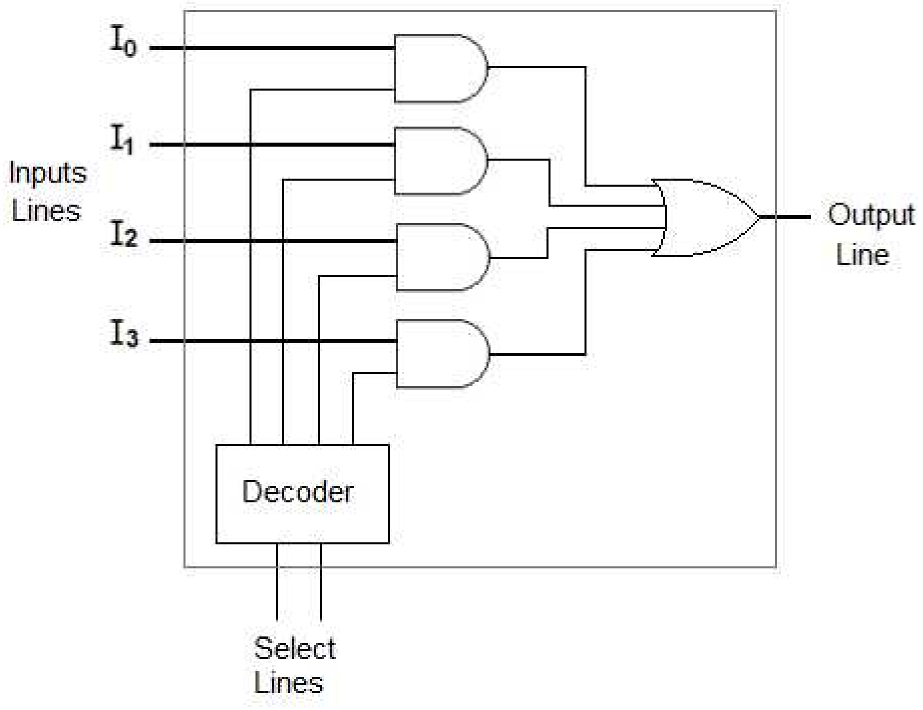


Fig.3.12 - Logic Diagram of a 4-input **Multiplexer**

Objective Questions

1. Decoder function is to select ______________ when a binary input is applied.
2. One application of Decoder is ____________________________.
3. A Full Adder adds _______________ bits at a time.
4. Multiplexer has ______________ inputs and ______________ output.
5. The selection logic in multiplexer is provided by a _________________.

Review Questions

1. Explain about Decoder with the help of a logic diagram.
2. Mention at least TWO the applications of decoder.
3. Draw the logic diagram of any one ADDER and explain.
4. What is the difference between Half-Adder and Full-Adder?
5. Explain with a diagram the working of a Multiplexer.

CHAPTER- 4

FLIP- FLOPS

4. Introduction

A Flip Flop is a fundamental digital device which has brought in revolutionary changes in the field of digital electronics by paving way for the development of a new branch of digital logic circuits called Sequential Logic Circuits.

A flip flop is a *basic memory or storage element*. The features of a flip-flop are:

1) It is a bi-stable device.
2) It has got one normal output ‘Q’ and its complementary ‘$\overline{Q}$’.
3) It can store a single bit of data

There are number of flip-flop types. Irrespective of their types all flip-flops have the above characteristics. The main types are:

4.1. SR- Flip Flop

It is the simplest form of flip-flop. It has got two inputs S and R. **S** stands for ‘**Set’** and R stands for ‘**Reset’**. When a high is applied to S input, the flip flop’s Q output changes to **high** (i.e., Q = 1) and then the flip flop is said to be in **Set** state. Likewise when high is applied to R input Q = 0 which is a Reset state. Symbol of SR Flip Flop is as given below:

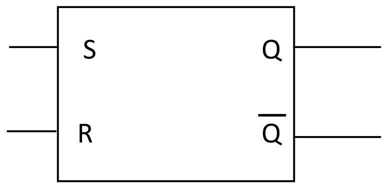


Fig.4.1- SR Flip Flop

S	R	Q	$\overline{Q}$	State
0	0	Q_n	$\overline{Q}_n$	No change
0	1	0	1	Reset
0	0	0	1	Previous
1	0	1	0	Set
0	0	1	0	Previous
1	1	?	?	Illegal State

Table 4.1- Truth Table of SR Flip Flop

Q_n, $\overline{Q}_n$ - Previous states and ? - Indeterminate state

The truth table showed above shows different output states for all input combinations of S and R. The first state when both S and R are ‘0’ the output state is ‘No Change’ state. That means the output remains in the previous state only.

The second input combination for S and R, i.e. S=0 and R=1 the output goes to ‘**Reset’** state. That is Q=0 and $\overline{Q}$=1 the state is known as Reset state.

The third combination of inputs, i.e. S=1 and R=0 changes the output to **Set** state. This is Q=1 and $\overline{Q}$=0.

The fourth and the final input combination for inputs S=1 and R=1, throws the flip-flop output to an illegal state. This means the state of both the outputs would become same that is either 0 or 1 at the same time. Moreover, when this input (S=1 and R=1) is withdrawn and 0 is applied on both inputs (S=0 and R=0) the next state to be obtained is unpredictable.

Hence this fourth combination S=1 and R=1 is called an "**illegal input**" combination and it should not be used with SR flip flop.

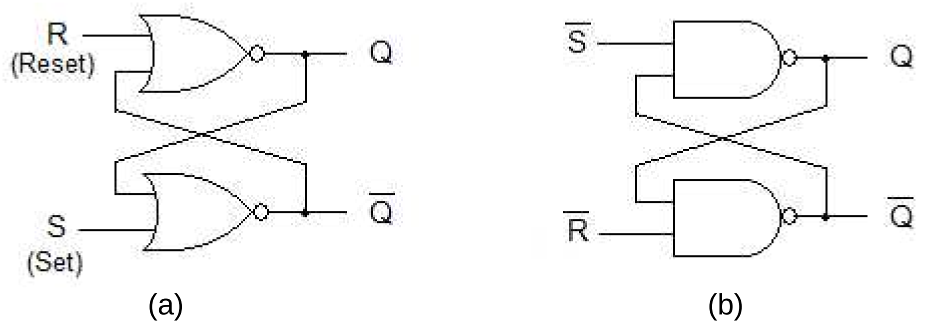


Fig.4.2- SR Flip Flop with (a) 2-NOR gates and (b) 2-NAND gates

Application of SR-Flip Flop: SR- Flip Flops are mainly used to activate or deactivate circuits. For example controlling timer 555.

4.2. D- Flip Flop

The main advantages of D flip-flop are:

1) The illegal input combination leading to indeterminate state in SR flip-flop is eliminated.
2) Unlike SR flip-flop it needs only one input signal 'D' to bring about output changes. It is called as D (data) flip flop because the value on D is reflected at Q.

Its symbol and truth table are:

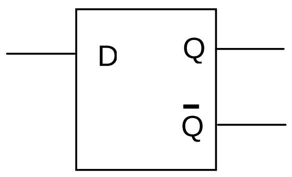


Fig. 4.3- D-Flip Flop

D	Q	$\overline{Q}$	State
0	0	1	Reset
1	1	0	Set

Table 4.2- Truth Table of D-Flip Flop

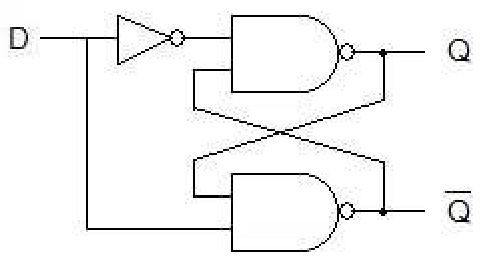


Fig. 4.4- Logic Circuit of D-Flip Flop

The versions of SR and D flip flops discussed so far are generally called as **SR Latch** and **D-Latch** respectively. These two are available in clocked version i.e. with an additional input called 'clock'. The logic symbols and their Truth tables are given below.

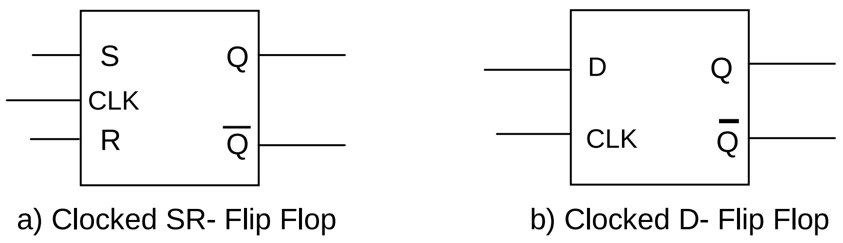


Fig. 4.5 - Clocked SR and D Flip Flops

CLK	S	R	Q	$\overline{Q}$	State
0	x	x	Q_n	$\overline{Q}_n$	No change
1	0	0	Q_n	$\overline{Q}_n$	No change
1	0	1	0	1	Set
1	1	0	1	0	Reset
1	1	1	?	?	Illegal State

Table 4.4- Truth Table of Clocked D Flip Flop

CLK	D	Q	$\overline{Q}$	State
0	x	Q_n	$\overline{Q}_n$	No change
1	1	1	0	Set
1	0	0	1	Reset

Table 4.3- Truth Table of Clocked SR Flip Flop

Application of D-Flip Flop: D- Flip Flops are mainly used for building Registers.

4.3. T- Flip Flop

This flip-flop like ‘D’ has got only one input called ‘T’. But the difference is that in case of D flip-flop whatever is given on D input the same thing is obtained on Q output. But with T flip-flop the output doesn’t change simply with a ‘0’ or a ‘1’ input. But it changes its output state with every clock pulse input. That too, on a particular edge of the clock pulse. The figure given below explains about edges in a clock pulse.

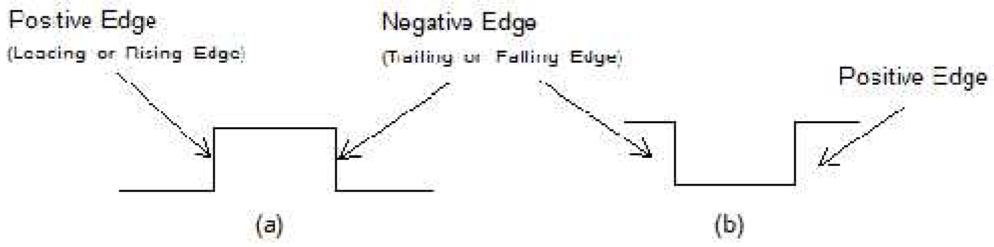


Fig.4.6 Positive and Negative Edges of a Pulse

The above figures represent two different pulses. ‘(a)’ represents a positive going pulse and ‘(b)’ represents its reverse, i.e. negative going pulse. But whatever may be the pulse shape, the main thing required to be noted is its edges. Every pulse has got essentially two edges - one positive edge and one negative edge.

A positive edge in the vertical line obtained while the signal changes from 0 to 1 value whereas the negative edge is its reverse, i.e. the vertical line obtained while the signal changes from 1 to 0. The positive and negative edges are marked by the upward and downward arrows respectively in any pulse as shown below.

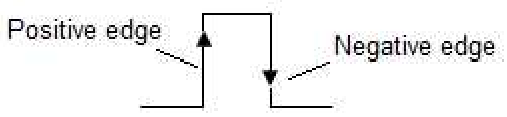
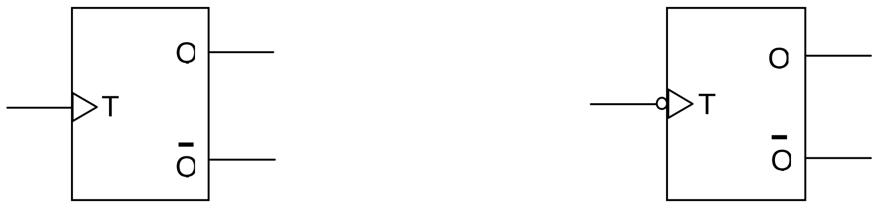


Fig.4.7 - Arrows showing +ve edge –ve edge in a pulse

The output of a T flip-flop changes state only during one of these two edges, but not during both the edges. Some T flip-flops are designed to change state during positive edge of a pulse. These are called as "*positive edge triggered*" T flip-flops. And some other flip-flops which are designed to change state during –ve edge of a pulse are called "*negative edge triggered*" T flip-flops. Both of these are shown by the following symbols.



a) Positive Edge Triggered b) Negative Edge Triggered

Fig.4.8 Symbols of Edge Triggered T Flip-Flops

A flip-flop changes state once with every clock pulse. This change is to the reverse state of the previous one. The Truth Tables 4.5 and 4.6 shown below explain this clearly. Initial or previous states of Q and $\underline{Q}$ are assumed as 1 and 0 respectively.

T	Q	$\overline{Q}$	State
--	1	0	Initial state
↑	0	1	Reset
↑	1	0	Set
↑	0	1	Reset
↑	1	0	Set

Table 4.5- Truth Table of Positive Edge Triggered T- Flip-Flop

T	Q	$\overline{Q}$	State
--	1	0	Initial state
↓	0	1	Reset
↓	1	0	Set
↓	0	1	Reset
↓	1	0	Set

Table 4.6- Truth Table of Negative Edge Triggered T- Flip-Flop

This nature of T flip-flop, reversing state with every clock pulse, is called as "**Toggling**".

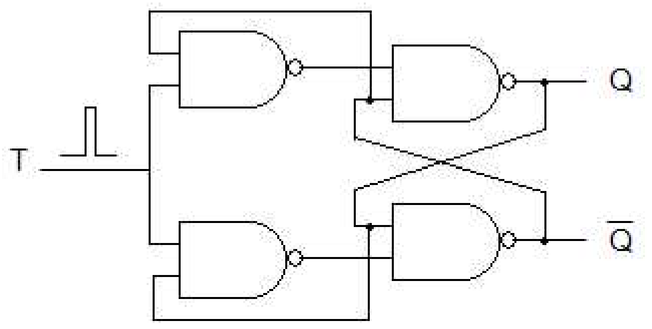


Fig.4.9 – Logic Diagram of T-Flip Flop

It can be seen from the diagram given in Fig.4.10 showing the timing and frequency relation between clock input & Q output of T flip flop that the output waveform has the same basic shape as that of input clock, but the only change is the output signal frequency becomes half of the input clock frequency. This means the T flip flop operates as a " Frequency Divider", that is, it divides clock frequency by 2.

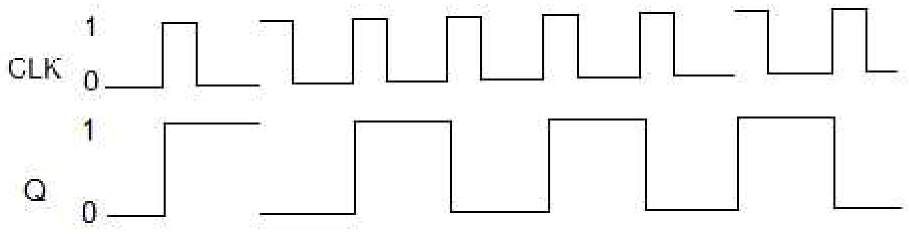
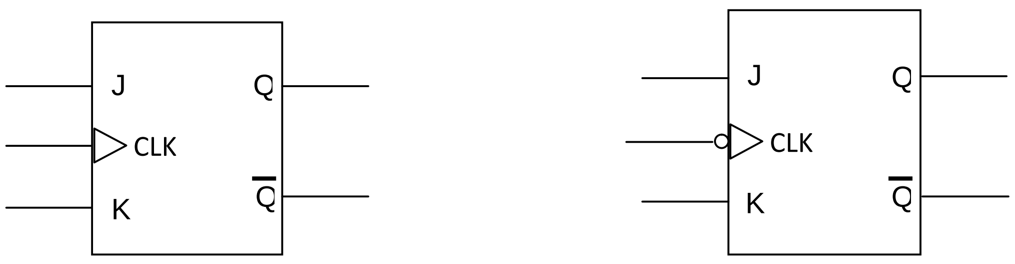


Fig.4.10- Frequency Division by T Flip Flop

4.4 JK FLIP-FLOP

It is versatile and most useful among all the flip-flops. Functionally, it is the combination of both SR and T flip-flops. A JK flip-flop is also an edge triggered one. Its diagrammatic representation is as given below in Fig. 4.11.



a) Positive Edge Triggered b) Negative Edge Triggered

Fig.4.11 - Symbols of Edge Triggered JK Flip Flops

The J-K flip-flop does exactly what an SR flip flop does, except for the special case when J = K = 1 (the "Toggle" state). In this case, the output of the circuit always switches to the opposite of its present state whenever a clock pulse edge occurs. This feature makes the JK flip flop as most useful among all the flip flops.

The following logic diagram in Fig. 4.12 shows the standard method of building JK flip flop. It uses four NAND gates. The input gates to which J and K are connected have a feedback with the Q and $\overline{Q}$ outputs so that at any time only one of the input NAND gate is enabled or will have favorable condition.

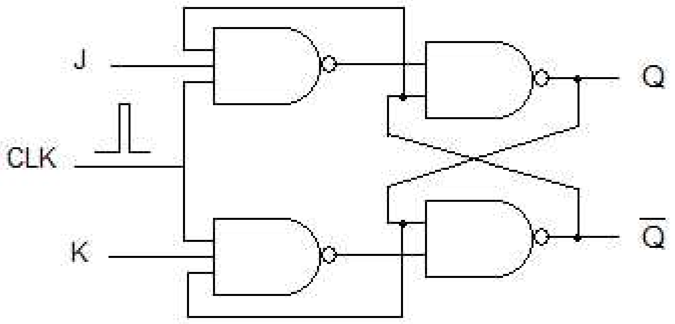


Fig. 4.12- JK Flip Flop Logic Diagram

The diagram given below shows how an S-R flip-flop can be used as J-K flip-flop using additionally two AND gates. The feedback to AND gates at the input of the S-R flip-flop prevent both S and R being 1 at the same time.

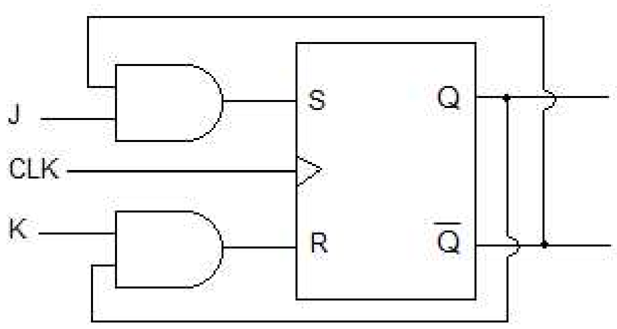


Fig. 4.13- Conversion of SR as JK Flip Flop

A JK Flip Flop is considered as the most versatile flip flop as it is possible to obtain the functions of all other flip flops either directly or with slight modification. It may be justified even if we call it as a **Universal flip flop**. The truth table for a positive edge triggered JK flip-flop is shown in Table 4.7 on the next page:

Application of T and JK Flip Flops: These two flip flops are used for constructing binary counters.

C	J	K	Q	$\overline{Q}$	State	Remark
0	X	X	Q_n	$\overline{Q_n}$	previous	No change
1	X	X	Q_n	$\overline{Q_n}$	previous	
↑	1	0	1	0	Set	SR Mode
↑	0	1	0	1	Reset	
↑	0	0	Q_n	$\overline{Q_n}$	previous	
↑	1	1	1	0	Set	Toggle mode
↑	1	1	0	1	Reset	
↑	1	1	1	0	Set	

Table 4.7- Truth Table of JK Flip Flop

Objective Questions

1. A Flip Flop is a ______________________________ device.
2. Meaning of S and R in SR Flip Flop is __________ and _________________.
3. Use of D flip flop is for ____.
4. T flip flop is mainly used for constructing _________________________________.
5. Which one of the flip flops can be called as a Universal flip flop. __________________.

Review Questions

1. What is a Flip Flop? Explain about its working.
2. Mention different types of Flip Flops and explain about JK Flip Flop.
3. Draw the logic diagram of SR Flip Flop and explain its working.
4. Mention at least THREE differences between SR and JK Flip Flops.
5. Why some Flip Flops need only edge-triggering? Explain with a reference.

CHAPTER-5

SEQUENTIAL LOGIC CIRCUITS

Using Flip Flop as the basic design element the following logic devices can be built. These devices are called as **Sequential** Logic Circuits.

1. Counters
2. Registers
3. Memory Devices

5.1. Counters :

A Counter is a device used to count the occurrence of an event, such as clock pulses. This counting is only in binary order. Because the counting is in binary order it is also called as a Binary counter. A binary counter is a sequential logic device because it is built using flip- flops as main construction element. Either T flip flops or JK flip-flops can be used for this purpose.

Basically Counters are of two types these are:

5.1.1. Asynchronous or Ripple counter

In this type of counters an external clock signal is applied to the first flip flop and its output is connected to the clock input of next flip flop as a clock signal.

5.1.2. Synchronous counter

In synchronous counter all flip flops receive the external clock pulse simultaneously. Ring counter and Johnson counter are the examples of synchronous counters.

5.2. Ripple Counters

In Ripple counters the flip flops used to build the counter do not change their output states at the same time but one after the other in a sequential order. Whereas in Synchronous counter all the flip flops change at the same instance.

A simple 3-bit ripple counter using T flip-flops is shown on the following page.

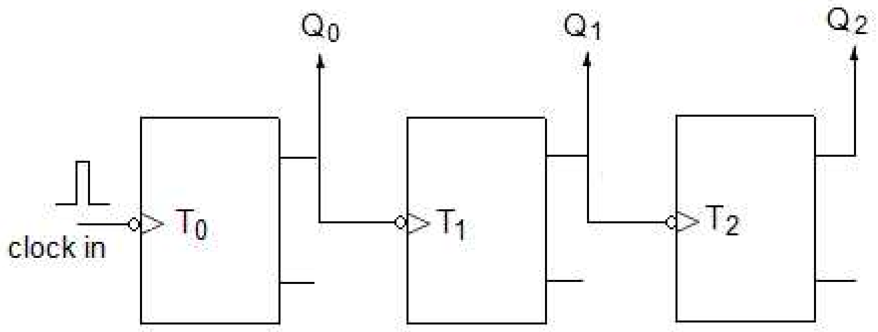


Fig.5.1- A 3-bit Ripple Counter

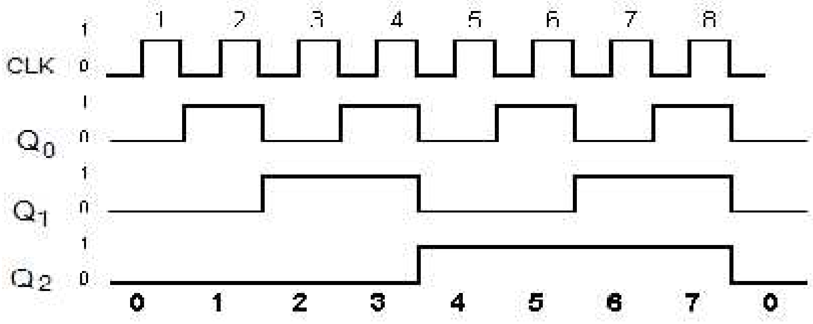


Fig.5.2- Clock and Output Waveforms relation in a Counter

Clock pulses are directly applied at T_0 of first flip-flop. The output Q_0 of this is connected to T_1 of second flip flop as a clock input. The output of second flip flop, Q_1 is connected to T_2 of third flip flop.

With negative edge of every clock pulse Q_0 changes between 0 and 1. These changes produce pulsing at T_1 for triggering it on the –ve edge. This results in change of Q1 and Q1 acts as clock to T_2. This results in a systematic change on the outputs Q_0, Q_1 and Q_2.

This change is in the following order. Let the initial status on Q outputs be 000. Then if we apply a stream of clock pulses at T_0, this gives,

Pulse		Q_2 Q_1 Q_0
1	-----	0 0 1
2	-----	0 1 0
3	-----	0 1 1
4	-----	1 0 0
5	-----	1 0 1
6	-----	1 1 0
7	-----	1 1 1
8	-----	0 0 0

From 1 to 7 the binary value held at Q0, Q_1 and Q_2 outputs is equivalent to the number of pulses applied. That means this arrangement of flip flops can count the applied pulses in binary order. With the 8th clock pulse it returns to the initial reading 000. If again clock pulses are applied the same counting order is repeated.

Modulus of Counter: Hence this circuit arrangement is capable of counting a maximum of 8 clock pulses. This is called the modulus (or mod) of the counter. The modulus of any binary counter can be known easily with,

Modulus = 2^n, N – is the number of bits (flip flops) in the counter.

5.3. Decade Counter:

A binary counter with **ten states** (Mod-10) in its counting sequence is called a *decade counter*. A Decade counter is designed using a 4-bit counter whose counting sequence is truncated at ten to make it a modulus-10 counter. Decade counter is an essential device for obtaining decimal readouts on various instruments and panel meters in our every day working. The circuit below is an implementation of a decade counter.

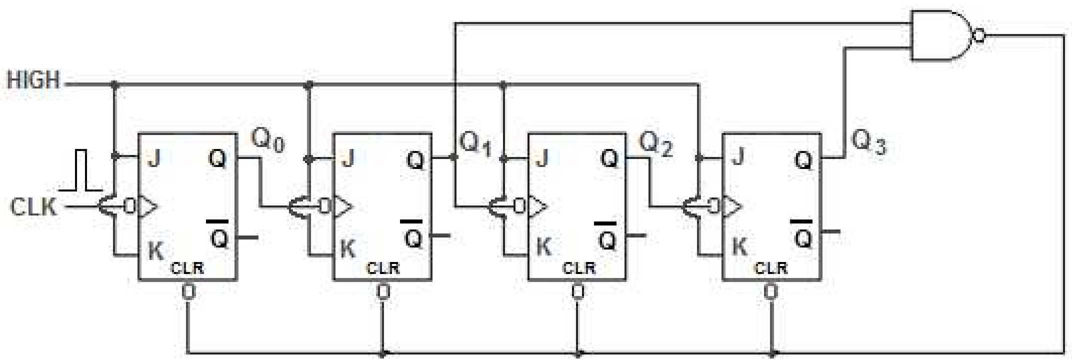


Fig.5.3- Decade counter

In the above circuit once the counter counts up to ten (1010), all the flip-flops are being cleared. Notice that only Q1 and Q3 are used to decode the count of ten. This is called partial decoding, as none of the other states (zero to nine) have both Q1 and Q3 in High state at the same time.

The counting sequence of decade counter is shown in the Table 5.1 below:

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Table 5.1- Counting sequence in a Decade Counter

5.4. UP and DOWN Counters :

The 3 bit binary counter which is discussed earlier is counting the pulses in ascending order that is upwards. This is called an UP counter. An UP counter uses all negative edge triggered flip flops.

In certain application counting in this reverse order to the above one that is descending or downward order is also needed. For example, for a foot ball match the clock is set for certain period say 50 min. When the match begins the clock is started to initiate down counting. Every minute the count decrements by 1, i.e., 49, 48, 47 etc. When the clock shows ‘00’ the match has to stop.

Similarly the well known countdown application is during Rocket launching.

Generally, all Up counters use -ve edge triggered flip flops whereas Down counters use +ve edge triggered flip flops. It is also possible to derive up and down counting from the same counter just by taking outputs from Qs for up counting and from $\overline{Q}$s for down counting. The Figures 5.4 and 5.5 on the next page show the logic diagrams of a 4-bit up counter and a 4-bit down counter respectively, whereas Fig. 5.6 show the diagram of an up/down counter.

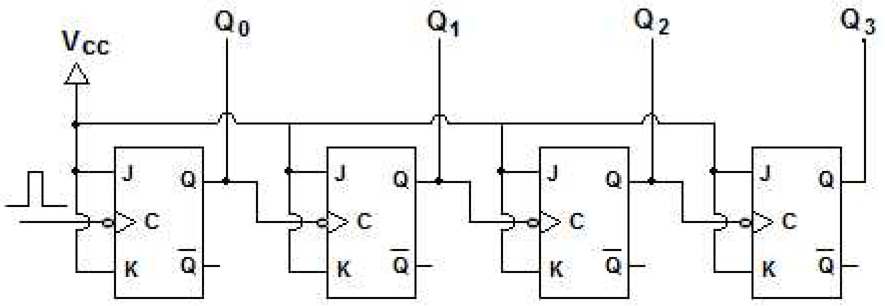


Fig.5.4 - 4-Bit Up Counter Using –ve Edge Triggered Flip Flops

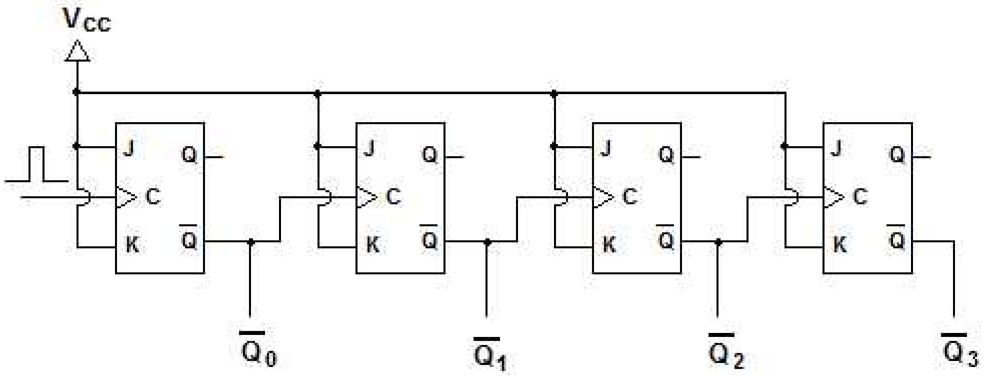


Fig.5.5 - 4-Bit Down Counter Using +ve Edge Triggered Flip Flops

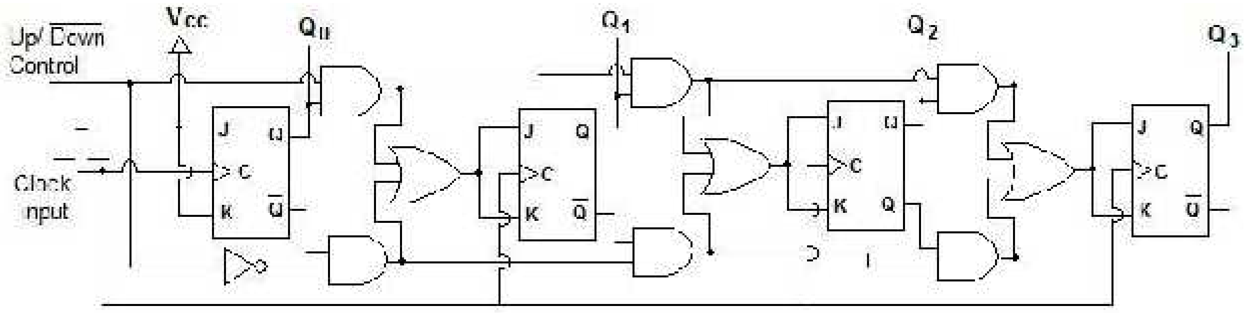


Fig.5.6 - A Synchronous Up/Down Counter.

5.5. Registers :

Like counter, register is also a sequential logic device. That means it is also built using D flip flops as main elements. As the name indicates a register is used to register or store binary information i.e., 0's and 1's.

Registers can be broadly divided into

a) Data / Memory Registers
b) Shift Registers

5.5.1. Memory Registers: Its application is in the memory devices. Each location in memory devices is a register made of D Flip Flops.

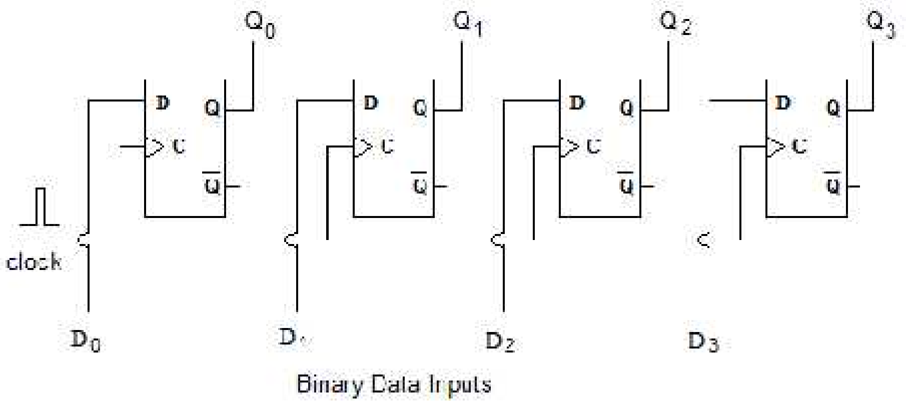


Fig.5.7- A Memory Register

The diagram in Fig.5.7 above shows a 4-bit register which can store any 4-binary bit value at a time. For example: 1101, 0101, and 1011 etc. The binary information to be stored (D0 – D_3) should be fed on to D inputs and it is stored at Q's after applying a clock pulse.

5.5.2. Shift Registers :

These are also like memory registers but the information stored into them can be shifted either from left to right or from right to left, by one bit position with the application of a clock pulse.

5.5.3. Types of Shift Registers

Depending upon data input (loading) and mode of output shift registers are classified into four types. This are-

a) Serial Input Serial Output (SISO) Shift Registers

Irrespective of bit length only a single input and a single output terminal is provided on this type of shift register for loading as well as retrieving data serially. As each new bit is loaded externally the data already loaded in SISO is shifted serially forward by one bit position. The diagram given below in Fig.5.7 shows SISO logic.

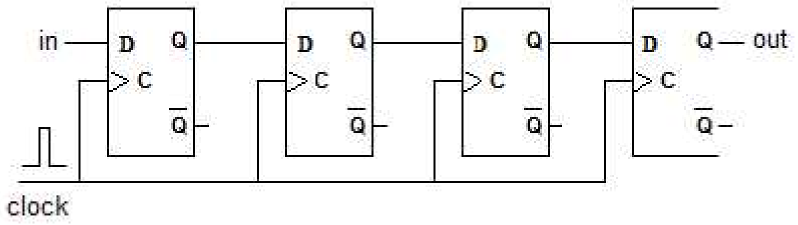


Fig.5.7- Construction of Serial in & Serial out Shift Register (SISO)

b) Serial Input Parallel Output (SIPO) Shift Registers

A SIPO has single input terminal to feed data serially and multiple output terminals to retrieve it. Symbol of SIPO is given below in Fig.5.8.

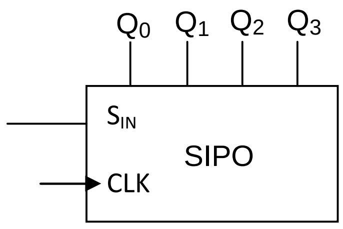


Fig.5.8- Symbol of a SIPO

c) Parallel Input Serial Output (PISO) Shift Registers

In this type of shift register data is loaded in parallel with a single clock pulse and thereafter the data can be shifted out serially one bit at a time. Symbol of PISO is given below in Fig.5.9.

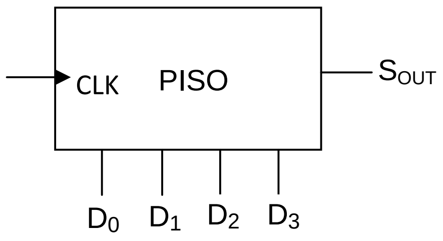


Fig.5.9- Symbol of a PISO

d) Parallel Input Parallel Output (PIPO) Shift Registers

A PIPO has provision for feeding as well as retrieving data in parallel. This means it has multiple and equal number of input/output terminals. Unlike the above three types actually there is no shifting of data inside it. Symbol of PIPO is given below in Fig.5.10. All data bits are loaded at once with a single clock pulse.

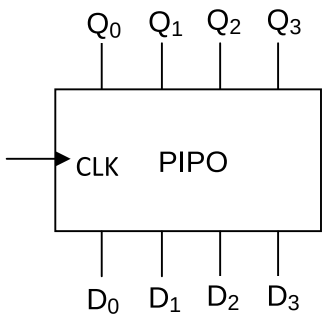


Fig.5.9- Symbol of a PIPO

Objective Questions

1. A counter is made up of ______________________ flip flops.
2. Generally, for constructing down counters _________________ triggered flip flops are used.
3. Registers are mainly of ______________ types.
4. Registers are constructed using ______________________ flip flops only.
5. Among Ripple and Synchronous counters which one is faster? ________________________

Review Questions

1. What is a sequential logic device? Explain about any one.
2. Draw the circuit of a simple 3-bit counter and explain.
3. Mention the difference between Ripple counter and a Synchronous counter.
4. Explain about Up & Down counters.
5. What is a Register? Draw the circuit of a 4-bit Register.

CHAPTER-6

MEMORY

6. What is memory?

Memory is a device that contains an array of memory registers in large number for storing information either temporarily or permanently. Memory is used to store data and programs in programmable systems like computers, Electronic Exchanges, Mobile phones, SSI, AWS etc. Memory is classified into two main groups. These are RAM and ROM.

6.1. RAM – Random Access Memory (or Read/Write Memory)

Both data storage (writing) and data retrieving (reading) operations are possible on this type of memory. That is why it is also called as Read/Write Memory.

RAM is **volatile** memory. It means that the information stored in the RAM will be lost once the power supply applied to it is removed. The working memory of computers is an example of RAM. Common types of RAM are –

1. Static RAM (SRAM) and
2. Dynamic RAM (DRAM)
3. Non- Volatile RAM (NVRAM)

6.1.1. Static RAM

SRAM is an array of flip flops of which the bit stored in the flip flop will remain until power is removed or another bit replaces it.

6.1.2. Dynamic RAM

DRAM stores a bit as the presence or absence of charge on MOSFET gate substrate capacitance. As the capacitance has leakage, it must be refreshed every few milliseconds. SRAM does not need to be refreshed. DRAM is usually up to 4 times denser than SRAM and hence cheaper. However, SRAM has faster access time than DRAM.

6.1.3. Non- Volatile RAM (NVRAM)

An NVRAM chip consists of both RAM and ROM. During power on reset, the contents of the ROM are copied to RAM. Before the power turns off, the system will copy the entire contents of the RAM into ROM for non volatile storage. The RAM in an NVRAM is called Shadow RAM. Thus NVRAM gives the benefits of both RAM and ROM.

6.2. Read Only Memory (ROM)

ROM is non volatile. ROM contents are not lost when its power is removed. Its name indicates only ‘Read’ operation, for retrieving stored data and programs, on ROM is possible. All ROMs can be programmed (storing information on it) at least once. ROM devices are classified based on the method of erasing or programming them. There are different types of ROM devices as given below.

6.2.1. Mask ROM

Mask-programmed Read only memory (ROM) has data stored on it during the manufacturing process. This data is permanent and cannot be altered or deleted.

6.2.2. PROM - Programmable ROM

The memory chip is manufactured blank, and the **programmer** stores the programs/data onto it. Once the data is transferred, it cannot be changed or erased. PROM can be **programmed** only **once** after manufacturing, by electrically burning specific transistors or diodes in the memory array. Because of this it is also called as **one time programmable (OTP)** memory.

6.2.3. EPROM - Erasable PROM

It is built using floating gate MOSFETs and hence offers high density of storage. Unlike PROM an EPROM can be erased by using ultraviolet (UV) light and can be reprogrammed again. It stores binary 1 and 0 in the form '***charge***' and '***no charge***' respectively. It is mainly used for storing permanent programs that are required in systems like computers, mobile phones etc. EPROM was a very common memory chip for storing BIOS (startup) programs in computers till recent times which is now replaced by Flash RAM.

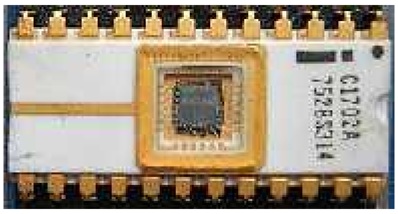


Fig.6. 1 - Photograph an EPROM chip

6.2.4. EEPROM- Electrically Erasable PROM

It is also called as E^{2}PROM. Unlike in EPROM one advantage of it is data on it can be erased electrically and hence no need of ultraviolet light. But its main disadvantage is it takes longer time for programming as compared to EPROM which is because it is erased and reprogrammed at the byte level.

6.2.5. Flash RAM

Flash memory is often used to hold control code such as the basic input/output system (BIOS) in a personal computer and makes its updating easy. Another advantage of Flash RAM is it does not need a high programming voltage(10 or 24 V DC) separately like in EPROM and EEPROM. The working and programming voltages are the same.

Advantages of Flash Memory

- Its main advantages are it combines the low cost and high density features of EPROM with the electrically erasable feature of EEPROM.
- Another advantage is it can be erased and reprogrammed without removing from circuit.
- But its one disadvantage is, flash RAM cannot be used in place of RAM because RAM needs to be addressable at the byte (not the block) level.

Symbols of Memory Devices

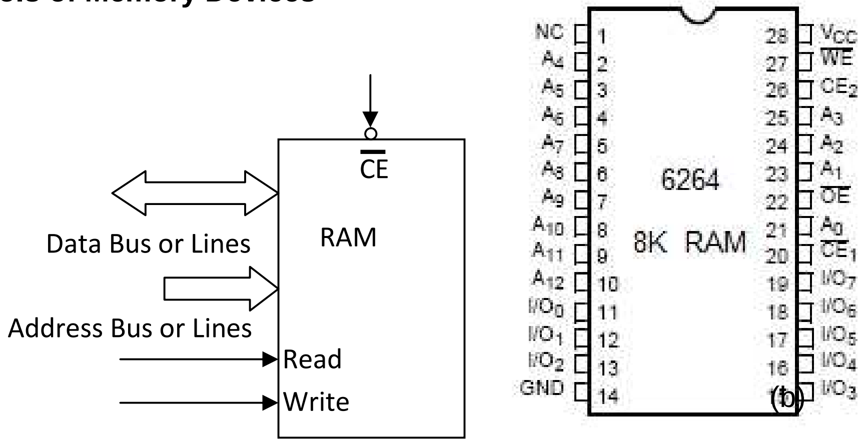


Fig.6.2 – **(a)** Symbol of RAM and **(b)** Pin Out of IC 6264 (8k RAM)

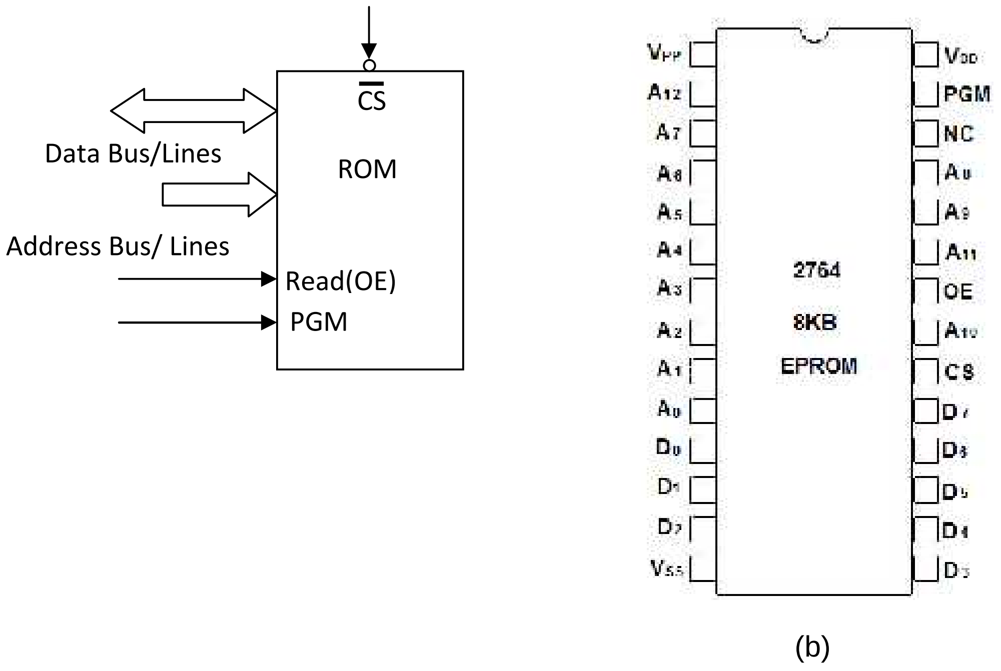


Fig.6.3 – **(a)** Symbol of ROM and **(b)** Pin Out of IC 2764 (8k EPROM)

The above two figures Fig.6.2 and Fig.6.3 show the symbols and IC pin out diagrams of RAM and ROM devices respectively.

Objective Questions

1. Memory is mainly divided into _____________ types.
2. ROM is called ____________________________________ memory.
3. Actually RAM should be called as ______________________________ memory.
4. Main disadvantage of EEPROM is ____.
5. Among all types of ROM which one you think as the best? _________________________.

Review Questions

1. What is memory? Mention its types.
2. Mention different types of RAM and explain them briefly.
3. Mention different types of ROM and explain them briefly.
4. What is the advantage of Flash RAM?
5. What type of memory you suggest for a system meant for a dedicated function and explains why.

CHAPTER-7

DIGITAL CODES

7. Digital Codes

Different types of codes are used in digital electronics for various purposes like

1. to represent numerals & characters
2. to represent alphabets of languages
3. to display numbers & text characters
4. to represent transmission data

Let us have a look at some of these important codes.

Numerical codes

These codes are used to represent decimal numerals in binary form.

1. BCD
2. Excess-3
3. Gray

Alphanumerical codes

The following are the names of binary codes used to represent alphanumeric characters.

1. ASCII
2. EBCDIC
3. Unicode

Transmission Data codes (Line Coding)

These codes are used for converting binary signal into a electrical digital signal to be transmitted on a communication line.

1. RZ - Return to zero
2. NRZ - Non- Return to zero
3. AMI - Alternate Mark Inversion
4. HDB3 - High Density Bipolar -3

7.1. Numerical codes

Let us study briefly the following numerical codes.

7.1.1. Binary Coded Decimal - BCD

It is used to represent the 10 numerals in decimal number system using 4-bit binary codes.

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

An Example of Usage: The following table 7.1 shows the BCD codes for some of the decimal numbers

S.No	Decimal Number	BCD code
1	**4**	0100
2	**25**	0010 0101
3	**347**	0011 0100 0111
4	**1060**	0001 0000 0110 0000

Table 7.1- BCD codes of Decimal Values

Disadvantage of BCD: In the addition of two BCD digits when sum exceeds 9 the binary value becomes invalid and ceases to be a BCD code. To correct this 0110 (6) to be added to the sum.

7.1.2. Excess-3 Code

It is also used for the same purpose like BCD but with different 4-bit binary codes. Each excess-3 binary code value is excess by 3 to the actual value of the digit being represented. Hence, like BCD it is not a weighted code. The decimal numerals representation in Excess-3 code can be seen in Table 7.2 given below.

Advantages of Excess-3 : It has the following advantages over BCD code.

- The bit inversion (1's complement) of an excess-3 value for decimal number gives directly its 9's compliment needed for subtractions.
- Performing addition and subtraction using excess-3 values is very easy. By simply adding/ subtracting 3 (0011) to/from the result the actual value is obtained.
- 3 (0011) to be added when the sum exceeds 1100(9) that is when there is a carry.

Examples of Addition: Let us perform the following additions

i) 52 + 36

```
     52    1000  0101    Excess-3 code for 43
   + 36 +  0110  1001    Excess-3 code for 36
     88    1110  1110    this sum is not an Excess-3 code of 88, but excess by 6

         - 0011 -0011    subtract 3 from each digit value to get value in excess-3
           1011  1011    this is correct Excess-3 code for 88
```

ii) 23 + 39

```
                        1
       23            0101    0110      Excess-3   code   for   23
      +39          + 0110    1100      Excess-3   code   for   39
       62            1100    0010
                                        Subtract 3 from 1st digit and add 3 to 2nd
                   - 0011  + 0011      (because there is a carry from 2nd to 1st digit)
                     1001    0101       Excess-3code for 62
```

Example of subtraction:

i) 83 - 34

```
                 -1
      83     1011   0110
   -  34   - 0110   0111
      49     0100   1111

                                  Add 3 to 1st digit as there is a barrow from
           +0011  - 0011          previous digit and  subtract  3 from 2nd  digit
             0111    1100          Excess-3 code for 62
```

If this same subtraction is performed by complements method

```
   -34     0110   0111    Excess-3 value of 34
    65     1001   1000     65 is 9's complement of 34.  In excess-3 65 is obtained
                           Simply by 1's complementing excess-3 value of 34
                           Now using this complement subtraction can be performed
                           As follows-

      83                          1011     0110
    + 65 (9's Complement        + 1001     1000
  (1) 48        of 34)      (1)   0100     1110
      +1                                     +1
      49                         0100     1111
                                +0011 -   0011    add 3 to 1st and subtract 3 from 2nd digits

                                 0111     1100     49 in excess-3
```

7.1.3. Gray Code

This was first used for indicating the positions of a shaft or rod of a machine while it is rotated for changing a parameter (ex: speed) of the machine. Later on this code is adopted for data transmission purposes also. The following Table 7.2 shows the values of decimal numbers in all the above three codes.

Numeral to be coded	BCD Code	Excess-3 Code	Gray Code
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0011
3	0011	0110	0010
4	0100	0111	0110
5	0101	1000	0111
6	0110	1001	0101
7	0111	1010	0100
8	1000	1011	1100
9	1001	1100	1101
10	0001 0000	0100 0011	1111
11	0001 0001	0100 0100	1110
12	0001 0010	0100 0101	1010
13	0001 0011	0100 0110	1011
14	0001 0100	0100 0111	1001
15	0001 0101	0100 1000	1000

Table 7.2- Numerals 1-15 represented in BCD, Excess-3 and Gray codes

Conversion from Binary to Gray Code

The below given are the three basic rules to be followed for converting a Binary value to its equivalent Gray Code –

(a) The first Gray digit is the same as the first binary digit.
(b) Add each pair of adjacent bits in binary to get the next Gray digit and
(c) Disregard any carries.

This same can be explained in detail as below:

- Retain MSB/ first bit (extreme left) of binary value as MSB in gray code also
- Add MSB and its adjacent bit to the right. Omit carry if any and take sum bit as the 2nd bit in gray code.
- Now take the bit next to MSB in binary and add it to its adjacent bit on the right. Take its sum as the next low order bit of gray code.
- Repeat the same procedure till the last two bits on extreme right of binary value are added to give the last bit (LSB) of gray code.

The following Fig. 7.1 shows the logic circuit using Ex-OR gates, which is used for converting a given binary value in to gray code automatically. For example let us take a binary number 11011 and convert it to gray. The above mentioned steps are automatically implemented in this circuit and the equivalent gray code is obtained as 10110.

Circuit for Conversion from Binary to Gray

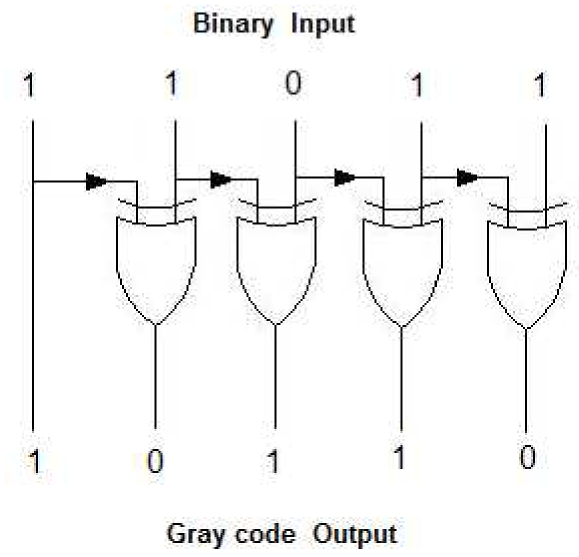


Fig. 7.1- Logic circuit for Binary to Gray conversion

Conversion from Gray to Binary

Similar to the above procedure there is another procedure for converting a gray value to a binary value. This procedure can be well understood by examining the following fig. 7.2 of a logic circuit which is used for converting a gray value in to a binary value.

Circuit for Conversion from Gray to Binary

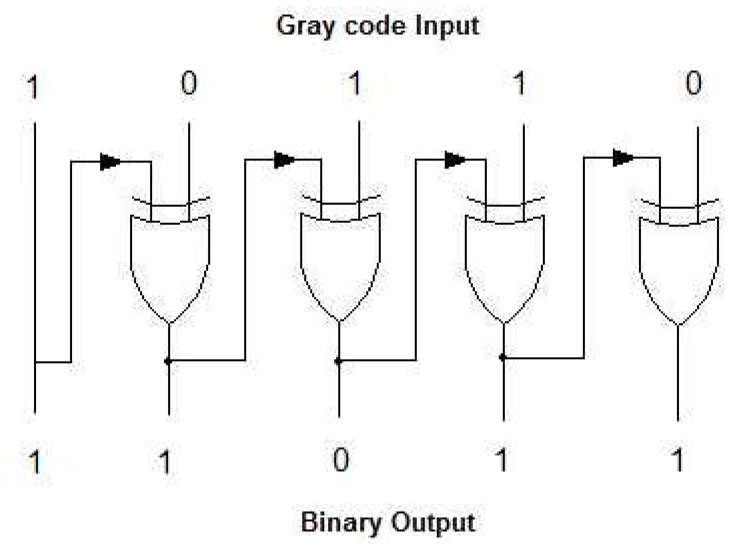


Fig. 7.2- Logic circuit for Gray to Binary conversion

7.2. Alphanumeric Codes

Digital systems and computers process information in digital (binary) form. Hence, alphanumerical characters, on keyboard for example, are required to be represented in binary form. For this purpose 7 or 8 bit binary codes are used to indicate different characters. Each of these 7 or 8 bit code is generally represented by its equivalent hexadecimal number.

7.2.1. **ASCII**- American Standard Code for Information Interchange

It is 7-bit binary code generally used for representing characters on a key board. In Table 7.2 we can see ASCII codes representing some of the keyboard characters.

7.2.2. **EBCDIC-** Extended BCD Interchange Code

It is an 8-bit (byte) binary code used for the same purpose as ASCII. EBCDIC is used mainly by IBM mainframes and compatibles. Each single byte **EBCDIC** is divided into two parts. The first four bits are called the **zone** and represent the category of the character, whereas the last four bits are the called the **digit** and identify the specific character.

7.2.3. **Unicode-** Universal Code

Unicode stands for universal code. It is developed to accommodate all the languages and symbols used in the world. It is a 16-bit code capable of representing 65536 characters which is more than enough for this purpose. It eliminates the complexity of multibyte character sets that are currently used on UNIX and Windows to support Asian languages. Unicode was created by a consortium of companies including Apple, Microsoft, HP, Digital and IBM and merged its efforts with the ISO-10646 standard to produce a single standard in 1993. Unicode is already the basis for latest operating systems like Windows NT, Windows 2000, Windows Vista etc and application packages like MS Office 2007. The following table gives the representation of key board characters in these three codes.

Character to be coded	ASCII	EBCDIC	Unicode
A	100 0001	1100 0001	0000 0000 0100 0001
B	100 0010	1100 0010	0000 0000 0100 0010
C	100 0011	1100 0011	0000 0000 0100 0011
a	110 0001	1000 0001	0000 0000 0110 0001
b	110 0010	1000 0010	0000 0000 0110 0010
c	110 0011	1000 0011	0000 0000 0110 0011
1	011 0000	1111 0001	0000 0000 0011 0001
2	011 0001	1111 0010	0000 0000 0011 0010
3	011 0010	1111 0011	0000 0000 0011 0011
+	010 1011	0100 1110	0000 0000 0010 1011
$	010 0100	0101 1011	0000 0000 0010 0100
&	010 0110	0101 0000	0000 0000 0010 0110
?	011 1111	0110 1111	0000 0000 0011 1111
क	---	---	0000 1001 0001 0101
त	---	---	0000 1001 0010 0100
म	---	---	0000 1001 0010 1110
स	---	---	0000 1001 0011 1000

Table 7.2- Alphanumerical characters Representation in ASCII, EBCDIC and Unicode

7.3. Data Transmission Codes (Line Coding)

Line coding used in data transmission is the process of converting binary data, a sequence of bits to a digital signal waveform. For example data, text, numbers, graphical images, audio and video that are stored in computer memory are all sequences of bits. Line coding converts a sequence of bits to a digital signal. The following figure shows waveforms to be transmitted on line in different types of line coding techniques.

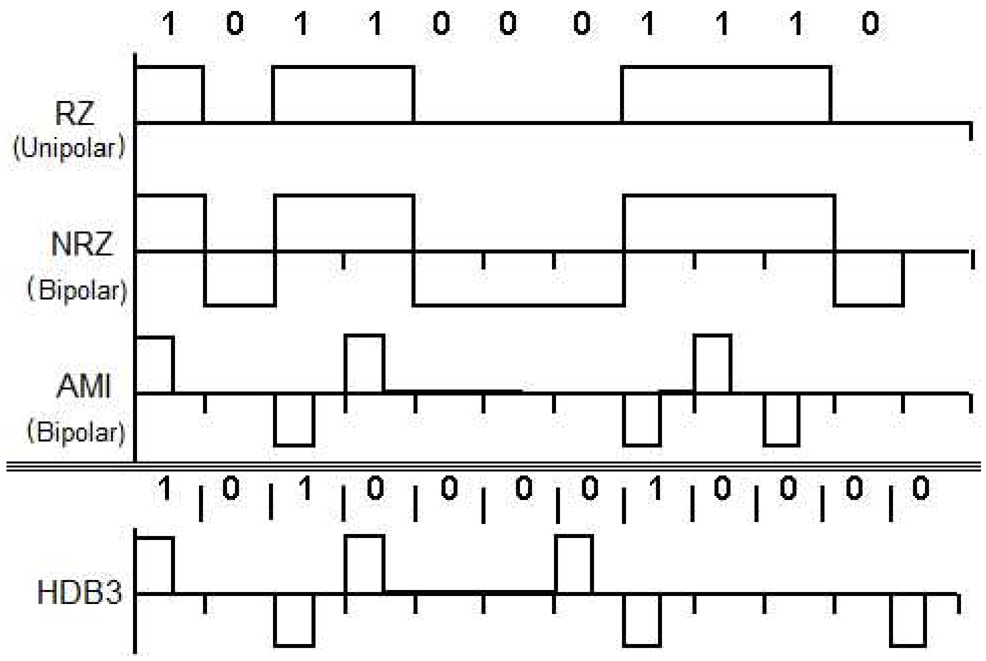


Fig.7.3- Waveforms of digital signal in different line codes

Objective Questions

1. One application of Codes in digital systems is to ___________________________________.
2. BCD is mainly used for ____.
3. Gray was first developed for using with ____.
4. ASCII expansion is ____.
5. Unicode is a _____________ bit code.

Review Questions

1. What is the purpose of using codes in digital systems?
2. Mention numerical codes and explain about BCD.
3. Mention the names of different alphanumeric codes.
4. Explain about ASCII code.
5. What is the difference between Unicode and other two alphanumeric codes

CHAPTER-8

DIGITAL LOGIC FAMILIES

8. Logic Circuit Technologies/Families

Logic circuits can be built using different components like diodes, transistors, MOSFETs, Resistors etc. Based on the type of components used logic circuits are classified into different *Logic Families*. The following are the names of the circuit technologies used from the initial days of building logic circuits.

- DL- Diode Logic
- RTL - Resistor Transistor Logic.
- DTL - Diode Transistor Logic.
- HTL - High Threshold Logic.
- TTL - Transistor Transistor Logic.
- IIL (I^2L) - Integrated Injection Logic.
- ECL - Emitter Coupled Logic.
- MOS - Metal Oxide Semiconductor Logic (PMOS and NMOS).
- CMOS - Complementary Metal Oxide Semiconductor Logic.

DL, RTL and DTL were the initial logic circuit design techniques. HTL is a variant of DTL which is used in environments where noise is very high. Now these are not in use. At present only TTL and CMOS are most widely used devices. The following five basic types of logic families of devices are available in the form of ICs (chips) . These are:

1. **TTL:** These devices use **Bi-Polar** transistors for their circuit design. **Bi-Polar** transistors are operated at cut off and saturation region only. TTL devices draw more amount of current compared to other logic families. These devices are available in **74, 74S, 74LS, 74AS, 74ALS, and 74F....** series. All digital functions are available in this logic family.
2. **MOS Logic:** It uses MOSFETs. It is again sub-classified as PMOS and NMOS logic families based on type of MOSFETs used. Memory chips and microprocessors are mostly from MOS logic family.
3. **CMOS Logic:** This logic uses complimentary pair of MOSFETs. **CMOS** devices come in 4000**/4500 series, 74C series** and also all other parts that are identified by the **'C'** in the part number.
4. **ECL:** These devices also use bipolar transistors, but they operate in the linear region. ECL devices are available in **10K, 100K, or 10E series**.
5. **Bi-CMOS:** Actually CMOS technology is used in analog, digital, and mixed-signal electronic circuit design. Now CMOS is further improved by combining bipolar transistors and MOSFETs in the circuits and it is called as Bi-CMOS technology.

8.1. Classification of Digital ICs

Depending on number of integrated components accommodated on integrated circuit the ICs are classified into four categories which are as shown below in Table 8.1.

S.No	IC Classification	Equivalent to no. of Gates	Number of Components
1	Small-scale integration (SSI)	Less than 12	Up to 99
2	Medium-scale integration (MSI)	12–99	100–999
3	Large-scale integration (LSI)	100–999	1,000–9,999
4	Very large-scale integration (VLSI)	Above 1,000	Above 10,000

Table 8.1- Classification of ICs

8.2. Important Characteristics of Logic Circuits

Logic circuits can be designed using different type of circuit technologies. The following parameters or characteristics are applicable to all logic families and are different for different technologies used for building circuits in each logic family.

- Fan-in.
- Fan-out.
- Propagation Delay.
- Noise Margin.
- Power Dissipation.
- Sink & Source Currents
- Logic Levels

8.2.1. Fan-in

Fan-in is the number of inputs a gate has, like a two input AND gate has fan-in of two, a three input NAND gate as a fan-in of three. So a NOT gate always has a fan-in of one. The figure below shows the effect of fan-in on the delay offered by a gate for a CMOS based gate. Normally delay increases when number of inputs (fan-in) increases.

8.2.2. Fan-out

The number of gates that each gate can drive, while providing voltage levels in the guaranteed range, is called the standard load or fan-out. The fan-out really depends on the amount of electric current a gate can **source** or **sink** while driving other gates. The effects of loading a logic gate output with more than its rated fan-out has the following effects.

- In the LOW state the output voltage V_{OL} may increase above V_{OLmax}.
- In the HIGH state the output voltage V_{OH} may decrease below V_{OHmin}.
- Output rise and fall times may increase beyond specifications
- The propagation delay may rise above the specified value.

8.2.3. Propagation Delay (T_{pd})

Propagation delay is the delay offered by a logic gate (or a logic circuit) while transferring the input signal on its input terminal to the output terminal. It can be stated as *"it is the time interval between the logic transition on an input and the resulting logic transition on the output of a logic gate/device"*. In following Fig. 8.1 propagation delay of a NOT gate is shown comparing the change in output signal against the change in its input signal.

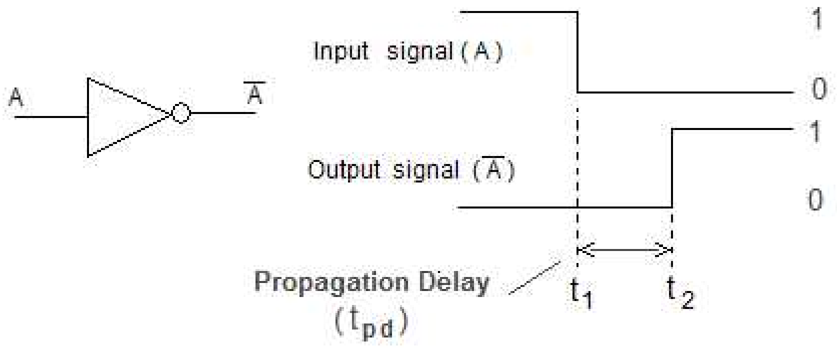


Fig.8.1- Propagation Delay of a NOT gate

For a gate propagation delay is not the same for both transitions, i.e. it will be different for low to high transition, compared to high to low transition. Low to high transition delay **(t_{PLH})** is called turn-on delay and High to low transition delay **(t_{PHL})** is called turn-off delay.

8.2.4. Noise Margin

Logic circuits are constructed to sustain variations in input and output voltage levels under the influence of external noise voltages. External noise voltages are superimposed on normal digital signal and result in the change of its logic state. The maximum noise voltage level that is tolerated by a gate or any other logic device is called **noise margin**.

8.2.5. Power Dissipation.

Each gate is connected to a power supply VCC (VDD in the case of CMOS). It draws a certain amount of current during its operation. Since each gate can be in a **High**, **Transition** or **Low** state, there are three different currents drawn from power supply.

- I_{CCH} : Current drawn during **High** state.
- I_{CCT} : Current drawn during High to Low or Low to High **transition**.
- I_{CCL} : Current drawn during **Low** state.

For TTL, I_{CCT} the transition current is negligible, in comparison to I_{CCH} and I_{CCL}. If we assume that I_{CCH} and I_{CCL} are equal then,

TTL Average Power Dissipation = Vic * (I_{CCH} + I_{CCL})/2

For CMOS, I_{CCH} and I_{CCL} current is negligible, in comparison to I_{CCT}. So the Average power dissipation is calculated as below.

CMOS Average Power Dissipation = Vcc * I_{CCT}.

From the above two equations it is evident that in TTL logic family, power dissipation does not depend on frequency of operation, and whereas in CMOS the power dissipation depends on the operation frequency.

Power dissipation is proportional to the heat generated by the chip or system. Excessive power dissipation may increase temperature and cause logic circuit to drift out of its normal operating range which results in malfunction of the logic circuit. Thus power dissipation of any logic device must be kept as low as possible. The power dissipation can be classified into Static power dissipation and Dynamic power dissipation.

- **Static Power Dissipation (P_S):** It is the power consumed when the output or input is not changing. Normally leakage current is the cause for static power dissipation.
- **Dynamic Power Dissipation (P_D):** It is the power consumed during output and input transitions. Pd can be stated as the actual power consumed during working of a logic circuit.

Thus

Total power dissipation = static power dissipation + dynamic power dissipation.

$$\mathbf{P_T = P_S + PD}$$

8.2.6. Sinking and Sourcing Currents

Digital IC outputs are often said to **'sink'** or **'source' current** depending on the logic level at the output. These two currents are in opposite directions. These current values are important for deciding the driving capacity of the devices.

i) Sinking Current: *It is the max. Current accepted (sinking) to flow into its output terminal by a logic device when its output is in low state.* If an IC is **sinking** a current means the current is flowing **into its output** terminal. This takes place when the output is at **low state**.

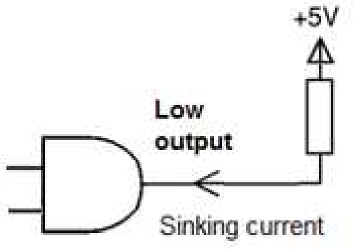


Fig. 8.2 - Sinking Current into a gate output

ii) Sourcing Current: *It is the current supplied (sourcing) by a logic device when its output is in high state.* If an IC is **sourcing a** current means the current is flowing **out of its output terminal**.

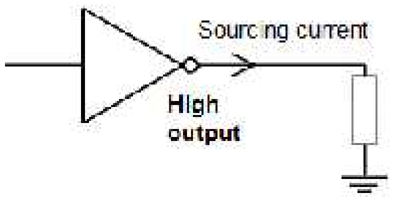


Fig. 8.3 - Sourcing Current from a Gate output

a) The maximum sinking and sourcing currents for TTL ICs are:

Sink Current (per output)	-	**16mA** max.
Source Current (per output)	-	**2mA.** Max.

b) But for CMOS devices the sink and source currents are same and are of the order of 1µA

Let us assume that an output of a TTL gate is connected to an input of another TTL gate. In TTL devices the sink and source currents from and to a single input lead are as given below:

Sink current	**-**	**1.6mA** (per input/device)
Source current	**-**	**200µA** (per input/device)

From these values it is possible to calculate the Fan-out of each TTL device. That is max no. of inputs that can be driven (Fan-out) in both states of a TTL output can be obtained by:

Fan-out = **Max current of output ÷ (divided by) current per input**

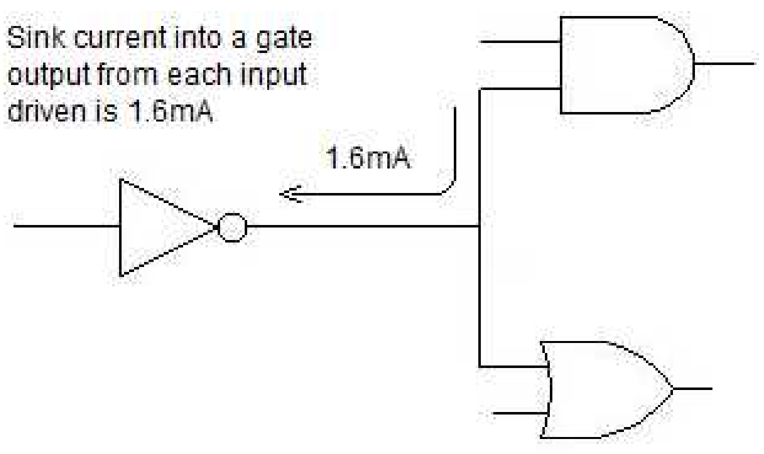


Fig.8.4 - Sink Current into a gate output terminal affected from a Single Input in TTL family

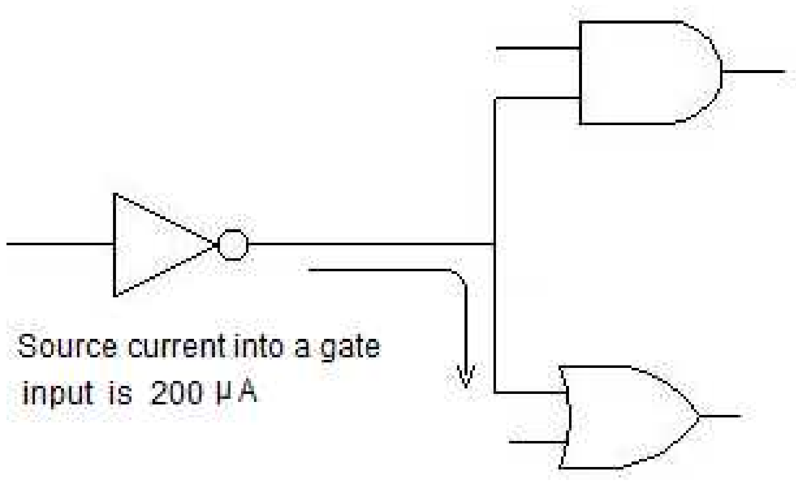


Fig.8.5 - Source Current into a Single TTL Input

8.2.7. Logic levels

Logic levels are the voltage levels required for logic high and logic low states of logic circuits. For example for TTL and CMOS (at V_{DD} of 5V) circuits the voltage ranges for logic states are as given in Table 8.2 below.

Device Type	Logic State	Input Voltage	Output Voltage
TTL	High	2.0 V to 5 V.	2.4 V to 5 V
	Low	0 V to 0.8 V	0 v to 0.4 V
CMOS	High	⅔ V_{DD} to V_{DD}	4.9 to V_{DD}
	Low	0 V to ⅓ V_{DD}	0 V to 0.1 V

Table 8.2- TTL & CMOS Logic Voltages

This same thing is represented graphically in the figure 8.6 given below.

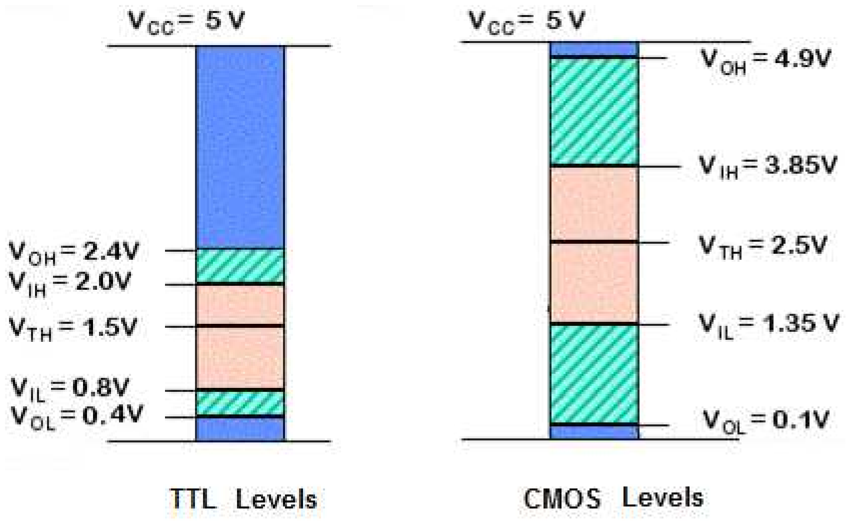


Fig.8.6 - Chart showing Logic Voltage Levels of TTL and CMOS Devices

8.2.8. Terms Related with Logic Levels:

- **V_{OHmin}:** It is the minimum output voltage in HIGH state (logic ‘1’). V_{Ohmin} is **2.4 V for TTL** and **4.9 V for CMOS.**
- **V_{OImax} :** It is the maximum output voltage in LOW state (logic ‘0’). V_{OImax} is **0.4 V for TTL** and **0.1 V for CMOS**.
- **V_{IHmin} :** It is the minimum input voltage guaranteed to be recognized as logic 1. V_{Ihmin} is **2 V for TTL** and **3.5 V for CMOS.**
- **V_{ILmax} :** It is the maximum input voltage guaranteed to be recognized as logic 0. V_{Ilmax} is **0.8 V for TTL** and **1.5 V for CMOS.**
- **V_T:** *[Threshold Voltage]* The voltage applied to a device which is "transition-Operated", which cause the device to switch.

Objective Questions

1. Mention the names of two popular logic families. _________________________________.
2. HTL is used in ________________________ environments.
3. In the design of TTL devices _________________ transistors are used.
4. Max. Sink current of TTL devices is _________________.
5. ________________ logic devices consume very low current.

Review Questions

1. Mention different types Logic Families.
2. Explain about TTL logic family characteristics
3. Write about the characteristics of logic devices.
4. Mention the advantages of CMOS logic family.
5. Write down the voltage values of logic levels in TTL and CMOS.

CHAPTER-9

MICROCONTROLLER – 8051

9.1. What is a Microcontroller?

After having studied about the microprocessor it is very easy to understand what a microcontroller is. Let us recall that a microprocessor is working as a CPU in any programmable system. To make it a fully fledged working system other peripheral devices like memory – both RAM and ROM, parallel I/O interfaces, timers, serial communication controllers etc., are very essential. Moreover, all these devices are to be properly connected with the CPU buses to make it a working system. This needs good designing skills, lot of efforts, and care. When a microcontroller is used most of these problems are eliminated.

A microcontroller is like a computer or a full-fledged programmable system on a single IC. It accommodates on a single semiconductor chip not only the CPU but also all the required peripheral devices with all interconnections already made between them to give us a **readymade programmable system**. By adding minimum external devices or components we can have a working system for any application.

Generally a microcontroller is used in dedicated applications using fixed programs. For example to control the operation of a machine, using a fixed program that is stored on the on-chip ROM called embedded system. Some other applications we come across outside are coin-inserting telephone boxes, washing machines, microwave ovens etc.

Figure 9.1 and 9.2 given below shows the main blocks of a typical microprocessor and microcontroller respectively. The design incorporates all of the features found in a microprocessor CPU: like ALU, PC, SP, and registers. It also has added other features needed to make it a complete programmable system: ROM, RAM, parallel I/O, serial I/O, counters, and a clock circuit.

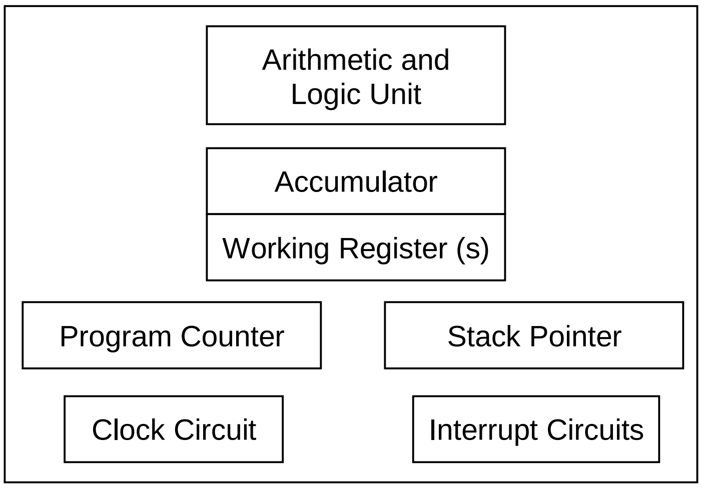


Fig. 9.1 Microprocessor

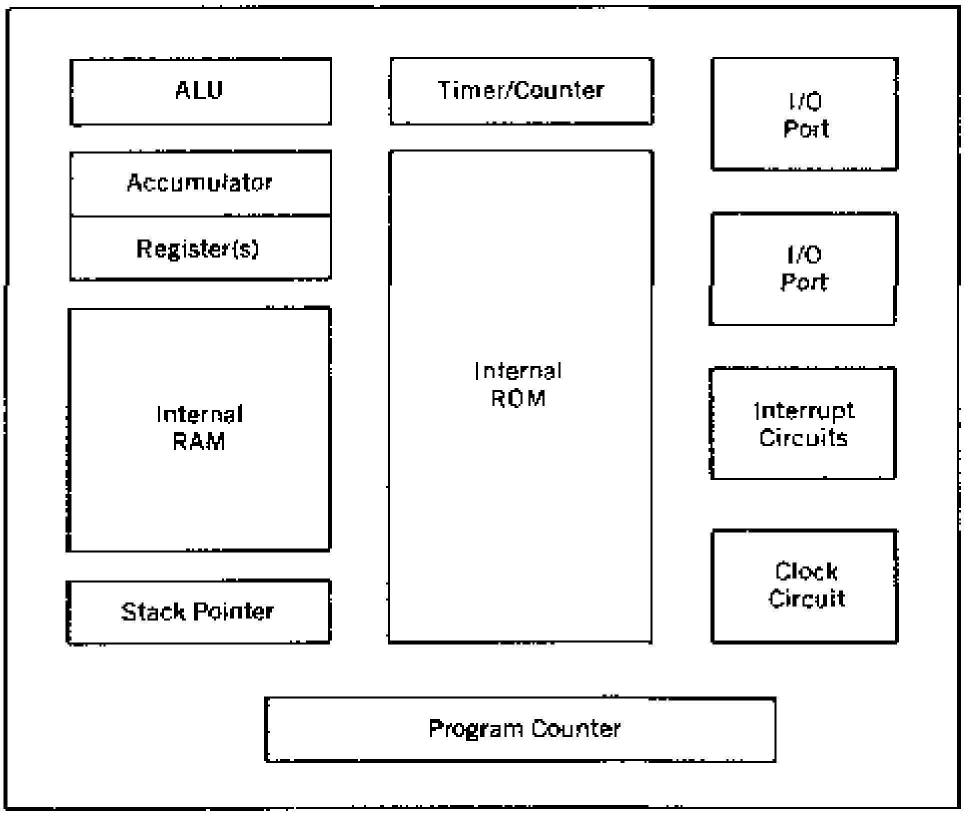


Fig. 9.2 Microcontroller

9.2. Special Features of Microcontrollers

Unlike microprocessor, a microcontroller has the following special features.

(a) The functional units inside the microcontroller are identified by addresses in addition to names like A, B, SP, T1, P2 etc.,

(b) There are number of SFRs performing the role of Control Registers in peripheral devices

(c) Individual bit addressing of registers is also possible

(d) The on-chip ROM or EPROM makes the microcontroller a perfect device for embedded system applications.

9.3. The vital differences in functioning of Microprocessor and Microcontroller are:

- The Microprocessor is concerned with rapid movement of code and data from external address to the chip but the Microcontroller is concerned with rapid movements of bits within the chip.
- It is a general purpose device.
- The microprocessor must have many additional parts to become operational but the microcontroller can function as a computer with no additional external digital parts.
- It is a single purpose device.

9.4. Intel 8051 Family of Microcontrollers:

Intel 8051 is a very popular microcontroller which is extensively used in number of applications. Because of its huge success, number of other companies produced their own version of microcontrollers based on 8051 basic design. The figure 9.3 given below shows the main blocks in the original design of 8051. The table 9.1 lists different makes of microcontrollers available in the market based on 8051 design.

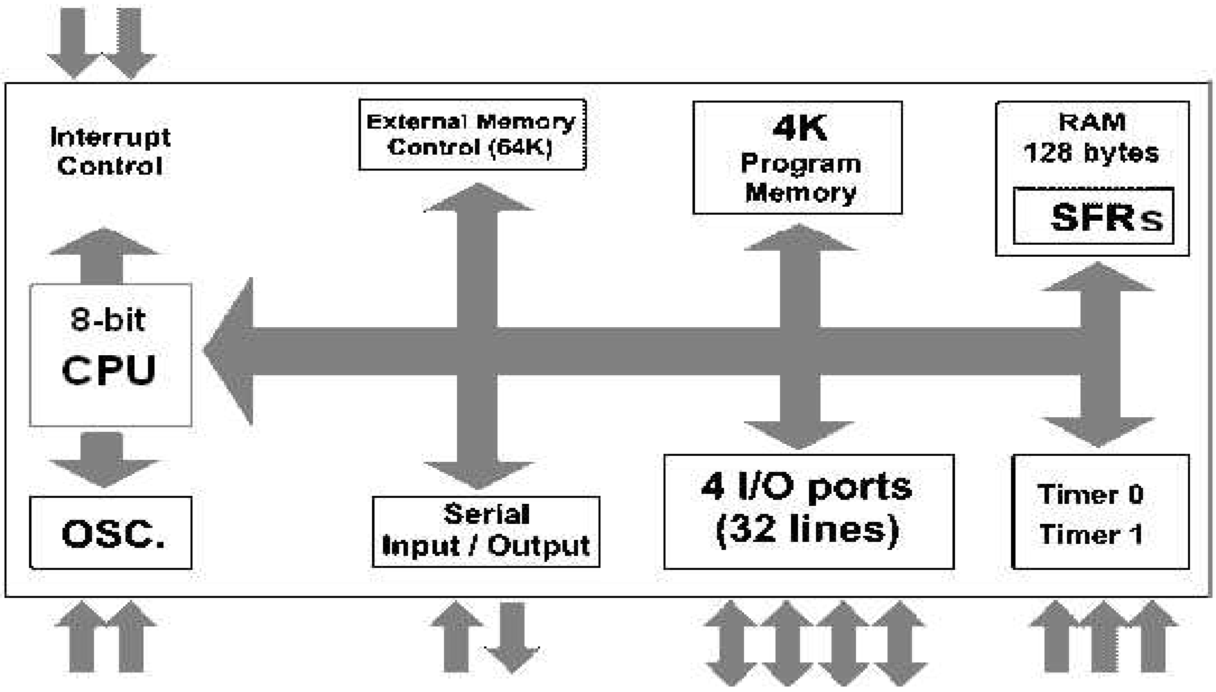


Fig. 9.3 Devices inside 8051

9.5. Different Makes of Microcontrollers:

The table 9.1 shown below gives several members of the 8051 family from different manufacturers.

Manufacturer/ Model	Pins: I/0	Counters	RAM (bytes)	ROM (bytes)	Other Features
Intel:8048	40:27	1	64	1K	External memory 8K
Intel:8051	40:32	2	128	4K	External memory 128K & serial port
Microchip:PICI6C56	18/12	0	25	1K	25/20 mA, sink/source, WDT, self reset, RC oscillator, low cost
National:COP820	28:24	1	64	1K	Serial bit I/O
Motorola:6805	28:20	1	64	1K	
Motorola:68HC11	52:40	2	256	8K	Serial ports; A/D; watch dog timer (WDT)
Rockwell:6500/1	40:32	1	64	2K	
Philips:87C552	68:48	3	256	8K	Serial port : A/D; WDT
TI:TMS 7500	40:32	1	128	2K	External memory up to 64K
TI: TMS370C050	68:55	2	256	4K	External memory up to 112K; A/D; serial ports; WDT
Zilog:Z8	40:32	2	128	2K	External memory up to 124K, serial port
ZilogZ86C83	28:22	2	256	4K	8 channel A/D; very low cost

Table 9.1

9.6. Architecture of 8051:

Intel 8051 is an 8-bit device. This generic part number actually includes a whole family of microcontrollers that have numbers ranging from 8031 to 8751 and are available in N-Channel Metal Oxide Silicon [NMOS] and Complementary Metal Oxide Silicon [CMOS] construction in a variety of package types. An enhanced version of the 8051, the 8052, also exists with its own family of variations and even includes one member that can be programmed in BASIC. In this chapter, we will study a generic 8051, housed in a 40-pin DIP.

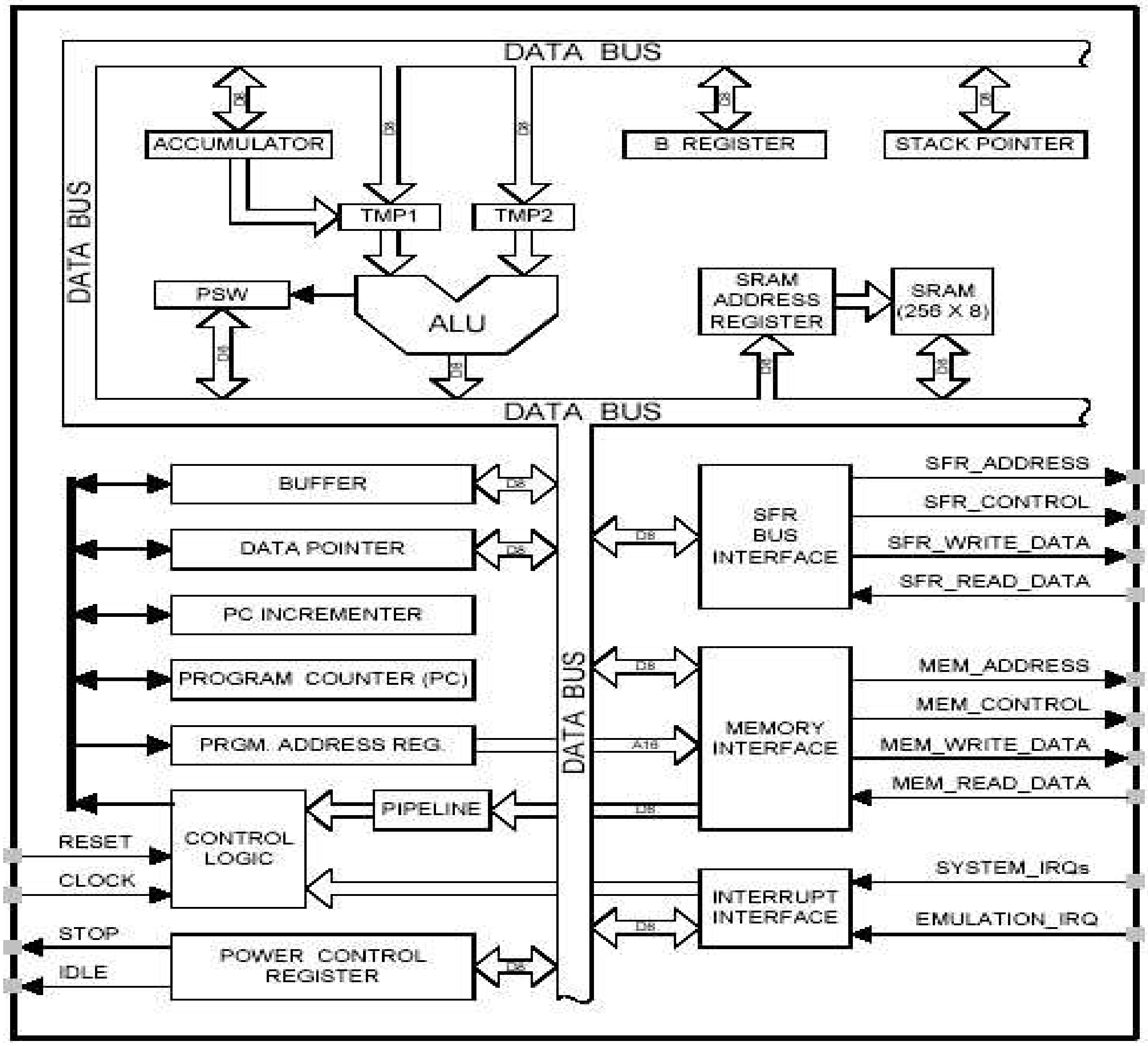


Fig. 9.4 Architecture of 8051

The architecture of 8051 consists of following specific devices and features:

(a) An 8-bit CPU with an accumulator [A] register and B register
(b) 16-bit program counter [PC] and data pointer [DPTR]
(c) 8-bit program status word [PSW] consisting flags
(d) 8-bit stack pointer [SP]
(e) Internal **ROM or EPROM** [0 bytes in 8031 to 4K bytes in 8051]
(f) Internal RAM of 128 bytes
(g) Thirty-two input/output lines arranged as four 8-bit ports: P0-P3

(h) Two 16-bit timer/counters: T0 and T1
(i) Full duplex serial data receiver/transmitter: SBUF
(j) Control registers: TCON, TMOD, SCON, PCON, IP, and IE
(k) Two external and three internal interrupt sources
(l) Oscillator and clock circuit

- 8051 has 34 general purpose or working registers.
- Two of these are A and B registers.
- The A register or Accumulator is used for many mathematical and logic operations.
- This is used for data transfers between 8051 and any external memory.
- B register is used with A register for multiplication and division operations.
- B register holds the second operand and will hold part of the result:
  - Upper 8 bits of the multiplication result.
  - Remainder in case of division.
- Other 32 are arranged as part of internal RAM in four banks, B0 to B3 of eight registers.
- Bank is chosen by setting RS1 and RS0 bits in PSW.
- Program Counter (PC) is a 16 bit register which tells the 8051 where the next instruction to execute is found in memory.
- PC is automatically incremented after every instruction byte is fetched.
- PC is the only register that does not have an internal address.
- DPTR is a 16 bit register made up of two 8 bit registers named DPH and DPL and is the only user accessible register.
- DPTR is used to point towards data and furnish memory addresses for external data access.
- Oscillator that generates clock pulses (1MHz to 16MHz) is the heart of the 8051 circuitry.
- The crystal frequency is the basic clock frequency of the microcontroller.
- Flags are one bit registers used to store the results of certain program instructions.
- Four flags are included in PSW.
- Program Status Word (PSW) is a special function register having 8 bits.
- 8051 has an 8 bit stack pointer.
- Stack refers to an area of internal RAM used to store and retrieve data quickly.
- The Stack Pointer is used to hold an internal RAM address called top of the stack.
- Special Function Registers used in 8051 may be addressed much like internal RAM to store data.
- Internal ROM occupies code address space 0000h to FFFFh.
- I/O ports connect the 8051 to the outside world.
- Two 16 bit timers/counters named T0 and T1 are provided for general use of the programmer.
- The counters are divided into two 8 bit registers called as timer low (TL0 & TL1) and timer high (TH0 & TH1).
- All counter actions are controlled by bit states in the TMOD and TCON.
- Timer Control (TCON) SFR is used to configure and modify the way of operation of two timers.
- SBUF, Serial Buffer SFR is used to send and receive data via the on-board serial port.

- In other words, the SBUF serves as the output port when written to and as the input port when read from.
- SCON (Serial Control) SFR is used to control the baud rate of the serial port.
- Interrupt Enable (IE) SFR is used to enable or disable specific interrupts.
- The lower bits of IE are used to enable or disable specific interrupts whereas higher bits are used to enable or disable all interrupts.
- Interrupt Priority (IP) SFR is used to specify the relative priority of each interrupt.
- Serial interrupt routine has the highest priority, so that no other interrupt will be able to interrupt while serial interrupt is executing.

9.7. Pin Configuration of 8051:

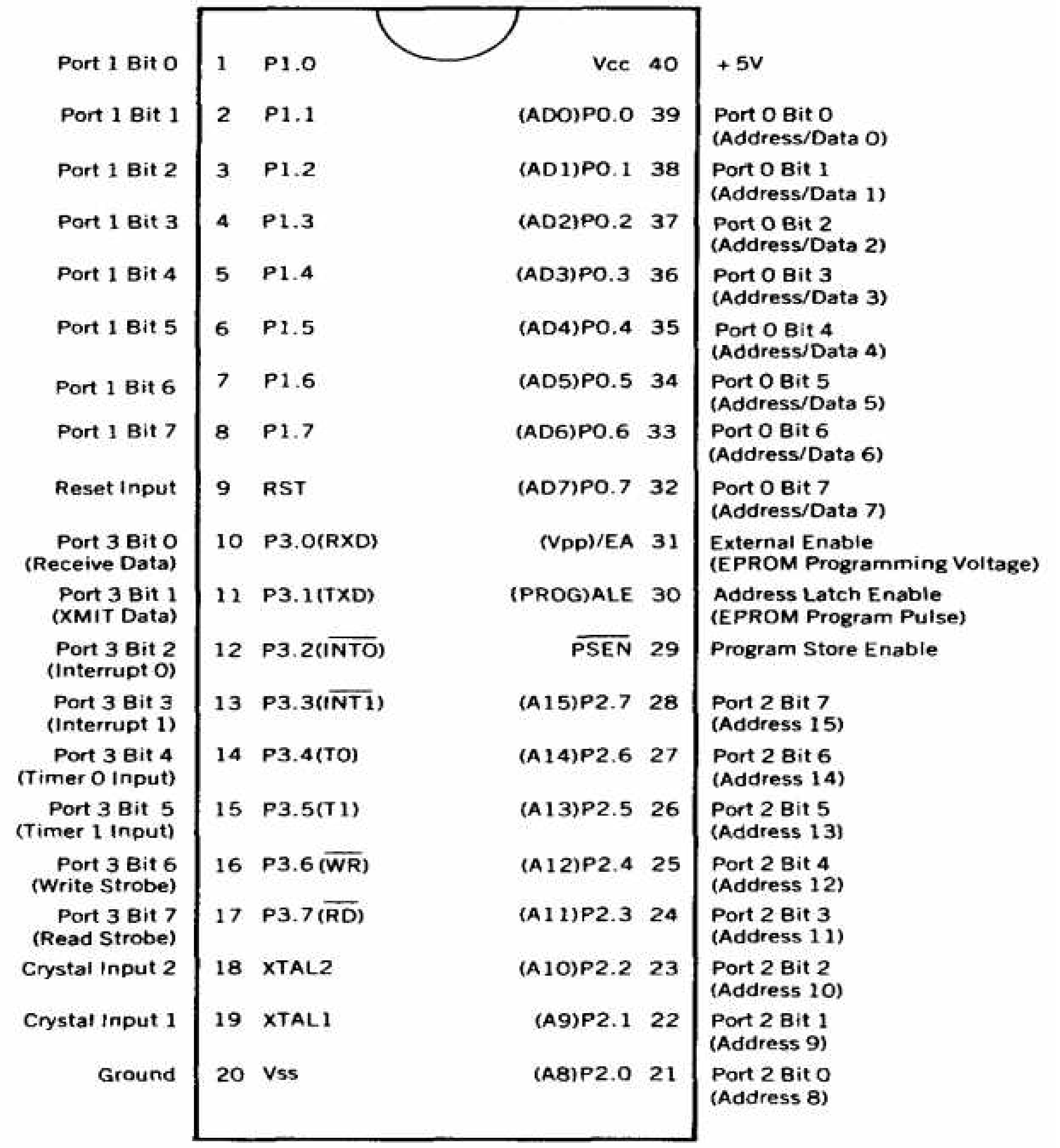

Port 1 Bit 0
1 P1.0
Vcc 40
+5V
Port 1 Bit 1
2 P1.1
(AD0)P0.0 39
Port 0 Bit 0 (Address/Data 0)
Port 1 Bit 2
3 P1.2
(AD1)P0.1 38
Port 0 Bit 1 (Address/Data 1)
Port 1 Bit 3
4 P1.3
(AD2)P0.2 37
Port 0 Bit 2 (Address/Data 2)
Port 1 Bit 4
5 P1.4
(AD3)P0.3 36
Port 0 Bit 3 (Address/Data 3)
Port 1 Bit 5
6 P1.5
(AD4)P0.4 35
Port 0 Bit 4 (Address/Data 4)
Port 1 Bit 6
7 P1.6
(AD5)P0.5 34
Port 0 Bit 5 (Address/Data 5)
Port 1 Bit 7
8 P1.7
(AD6)P0.6 33
Port 0 Bit 6 (Address/Data 6)
Reset Input
9 RST
(AD7)P0.7 32
Port 0 Bit 7 (Address/Data 7)
Port 3 Bit 0 (Receive Data)
10 P3.0(RXD)
(Vpp)/EA 31
External Enable (EPROM Programming Voltage)
Port 3 Bit 1 (XMIT Data)
11 P3.1(TXD)
(PROG)ALE 30
Address Latch Enable (EPROM Program Pulse)
Port 3 Bit 2 (Interrupt 0)
12 P3.2(INT0)
PSEN 29
Program Store Enable
Port 3 Bit 3 (Interrupt 1)
13 P3.3(INT1)
(A15)P2.7 28
Port 2 Bit 7 (Address 15)
Port 3 Bit 4 (Timer 0 Input)
14 P3.4(T0)
(A14)P2.6 27
Port 2 Bit 6 (Address 14)
Port 3 Bit 5 (Timer 1 Input)
15 P3.5(T1)
(A13)P2.5 26
Port 2 Bit 5 (Address 13)
Port 3 Bit 6 (Write Strobe)
16 P3.6(WR)
(A12)P2.4 25
Port 2 Bit 4 (Address 12)
Port 3 Bit 7 (Read Strobe)
17 P3.7(RD)
(A11)P2.3 24
Port 2 Bit 3 (Address 11)
Crystal Input 2
18 XTAL2
(A10)P2.2 23
Port 2 Bit 2 (Address 10)
Crystal Input 1
19 XTAL1
(A9)P2.1 22
Port 2 Bit 1 (Address 9)
Ground
20 Vss
(A8)P2.0 21
Port 2 Bit 0 (Address 8)

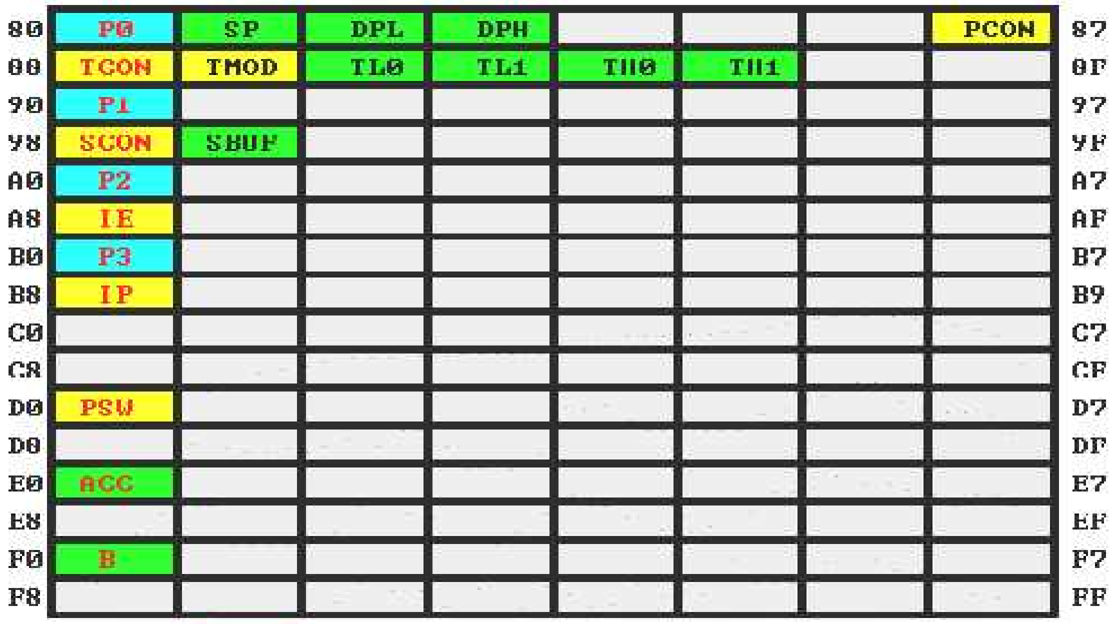
Fig. 9.5 Pin configuration

9.8. List of registers in 8051:

SPECIAL FUNCTION REGISTERS	
A	Accumulator
B	B Register
PSW	Program Status Word
P0	Port 0
P1	Port 1
P2	Port 2
P3	Port 3
IP	Interrupt Priority control
IE	Interrupt Enable control
TCON	Timer/Counter Control
OTHER REGISTERS	
SP	Stack Pointer
DPTR **DPL** **DPH**	Data Pointer
TMOD	Timer Mode
TH0	Timer/Counter 0 MSB
TL0	Timer/Counter 0 LSB
TH1	Timer/Counter 1 MSB
TL1	Timer/Counter 1 LSB

Table 9.3

9.9 SFR Locations in 8051:



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

🟦	Blue background are I/O port SFRs
🟨	Yellow background are control SFRs
🟩	Green blackground are other SFRs

9.9. Description of SFRs of 8051:

a) **P0 (Port 0, Address 80h, Bit-Addressable):** The P0 port is characterized by two functions. If external memory is used then the lower address byte (addresses A0-A7) is applied on it. Otherwise, all bits of this port are configured as inputs/outputs.

b) **P1 (Port 1, Address 90h, Bit-Addressable):** This is input/output port 1. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 0 is pin P0.0, bit 7 is pin P0.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

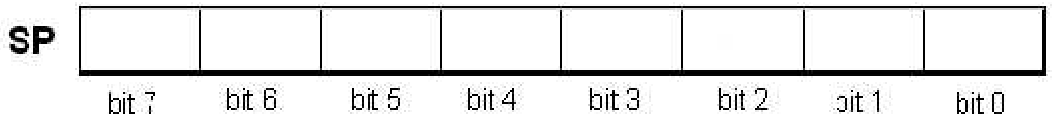
c) **P2 (Port 2, Address A0h, Bit-Addressable):** Port 2 acts similarly to P0 when external memory is used. Pins of this port occupy addresses intended for external memory chip. This time it is about the higher address byte with addresses A8-A15. When no memory is added, this port can be used as a general input/output port showing features similar to P1.

d) **P3 (Port 3, Address B0h, Bit-Addressable):** All port pins can be used as general I/O, but they also have an alternative function. In order to use these alternative functions, a logic one (1) must be applied to appropriate bit of the P3 register. In tems of hardware, this port is similar to P0, with the difference that its pins have a pull-up resistor built-in.

Alternate functions of P3:

Pin No.	Alternate use	SFR
P3.0 – RXD	Serial data input	SBUF
P3.1 – TXD	Serial data output	SBUF
P3.2 – INT0	External interrupt 0	TCON.1
P3.3 – INT1	External interrupt 1	TCON.3
P3.4 – T0	External timer 0 input	TMOD
P3.5 – T1	External timer 1 input	TMOD
P3.6 – WR	External memory write pulse	--
P3.7 - RD	External memory read pulse	--

e) **SP (Stack Pointer, Address 81h):** This is the stack pointer of the microcontroller. This SFR indicates where the next value to be taken from the stack of Internal RAM. This SFR is modified by all instructions which modify the stack, such as PUSH, POP, LCALL, RET, RETI, and whenever interrupts are provoked by the microcontroller.



A value stored in the Stack Pointer points to the first free stack address and permits stack availability. Stack PUSH increment the value in the Stack Pointer by 1. Likewise, stack POP decrement its value by 1. Upon any reset and power-on, it will load the address value of RAM reserved for the stack start location. If another value is written to this register, the entire Stack is moved to the new memory location.

f) **DPL/DPH (Data Pointer Low/High, Addresses 82h/83h):** The SFRs DPL and DPH work together to represent a 16-bit value called the Data Pointer. The data pointer is used in operations regarding external RAM and some instructions involving code memory. Their 16 bits are primarily used for external memory addressing. Besides, the DPTR Register is usually used for storing data and intermediate results. Since it is an unsigned two-byte integer value, it can represent values from 0000h to FFFFh.

g) **PCON (Power Control, Addresses 87h):** This SFR is used to control the 8051's power control modes. Certain operation modes of the 8051 allow it to go into a type of "sleep/idle" mode which requires much less power. These modes of operation are controlled through PCON. One of the bits in PCON is used to double the effective baud rate of the 8051's serial port.

PCON	SMOD				GF1	GF0	PD	IDL
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- When SMOD bit is high the Baud rate of serial port will become twice.
- GF1 General-purpose bit (available for use).
- GF1 General-purpose bit (available for use).
- GF0 General-purpose bit (available for use).
- By setting PD bit the microcontroller enters the *Power Down* mode.
- By setting IDL bit the microcontroller enters the *idle* mode.

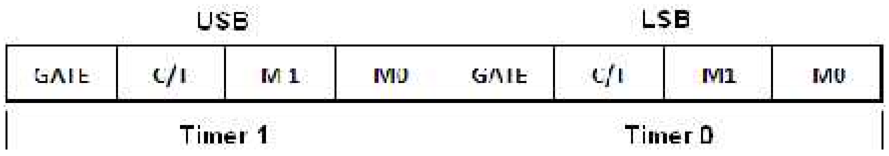
h) **TCON (Timer Control, Addresses 88h, Bit-Addressable):** The Timer Control SFR is used to configure and modify the way in which the 8051's two timers operate. Some non-timer related bits are located in the TCON SFR. These bits are used to configure the way in which the external interrupts are activated and also contain the external interrupt flags which are set when an external interrupt has occurred.

TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

TIMER: A timer always counts up. It doesn't matter whether the timer is being used as a timer, a counter, or a baud rate generator. A timer is always incremented by the microcontroller.

Timer SFRs: The 8051 has two timers, each function essentially the same way. One is Timer 0 and the other is Timer1. The two timers share two SFRs (TMOD and TCON) which control the timers, and each timer also has two SFRs dedicated solely to itself (TH0/TL0 and TH1/TL1).

i) **TMOD (Timer Mode, Addresses 89h):** The Timer Mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, an 8-bit auto reload timer, a 13-bit timer, or two separate timers.



- Both Timer 0 & Timer 1 use the same Mode register TMOD.
- It is an-8-bit register.
- The lower 4-bits are meant for Timer 0 & the upper 4-bits are meant for Timer 1. It is not bit addressable.

Functions of individual bits:

Bit	Name	Function	Timer
7	Gate-1	When this bit is set the timer will only run when INT1 (P3.3) is high.	1
6	C/T-1	When this bit is set the timer will count events on T1 (P3.5).	1
5	T1 M1	Timer Mode Bit	1
4	T1M0	Timer Mode Bit	1
3	Gate-0	When this bit is set the timer will only run when INT0 (P3.2) is high.	0
2	C/T-0	When this bit is set the timer will count events on T0 (P3.4).	0
1	T0 M1	Timer Mode Bit	0
0	T0M0	Timer Mode Bit	0

Timer Modes:

Tx M1	Tx M0	Timer Mode	Description of Mode
0	0	**0**	13 bit Timer
0	1	**1**	16 bit Timer
1	0	**2**	8 bit auto reload
1	1	**3**	Split Timer

1. **13-bit Timer Mode (mode 0):** Timer mode "0" is a 13-bit timer. Generally the 13-bit timer mode is not used in new development. When the timer is in this mode, TLx will count from 0 to 31. When TLx is incremented from 31, it will "reset" to 0 and increment THx. Thus, effectively, only 13 bits of the two timer bytes are being used: bits 0-4 of TLx and bits 0-7 of THx. This means, in essence, the timer can only contain 8192 values. If you set a 13-bit timer to 0, it will overflow back to zero after 8192 machine cycles.

2. **16-bit Timer Mode (mode 1):** Timer mode "1" is a 16-bit timer. This is a very commonly used mode. It functions just like 13-bit mode except that all 16 bits are used. TLx is incremented from 0 to 255. When TLx is incremented from 255, it resets to 0 and causes THx to be incremented by 1. Since this is a full 16-bit timer, the timer may contain up to 65536 distinct values. If you set a 16-bit timer to 0, it will overflow back to 0 after 65,536 machine cycles.

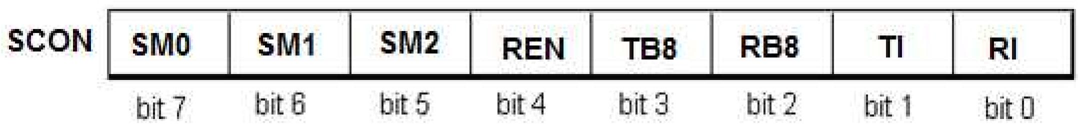
3. **8-bit auto reload Mode (mode 2):** Timer mode "2" is an 8-bit auto-reload mode. When a timer is in mode 2, THx holds the "reload value" and TLx is the timer itself. Thus, TLx starts counting up. When TLx reaches 255 and is subsequently incremented, instead of resetting to 0 (as in the case of modes 0 and 1), it will be reset to the value stored in THx. The auto-reload mode is very commonly used for establishing a baud rate.

4. **Split Timer Mode (mode 3):** Timer mode "3" is a split-timer mode. When Timer 0 is placed in mode 3, it essentially becomes two separate 8-bit timers. That is to say, Timer 0 is TL0 and Timer 1 is TH0. Both timers count from 0 to 255 and overflow back to 0. All the bits that are related to Timer 1 will now be tied to TH0. While Timer 0 is in split mode, the real Timer 1 (i.e. TH1 and TL1) can be put into modes 0, 1 or 2 normally; however, you may not start or stop the real timer 1 since the bits that are now linked to TH0. The real timer 1, in this case, will be incremented for every machine cycle.

j) **TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Bh):** These two SFRs, taken together, represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

k) **TL1/TH1 (Timer 1 Low/High, Addresses 8Ch/8Dh):** These two SFRs, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

l) **SCON (Serial Control, Addresses 98h, Bit-Addressable):** The Serial Control SFR is used to configure the behavior of the 8051's on-board serial port. This SFR controls the baud rate of the serial port, whether the serial port is activated to receive data, and also contain flags that are set when a byte is successfully sent or received.



SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

SM0	SM1	MODE	DESCRIPTION	BAUD RATE
0	0	0	8 bit shift register	1/12 quartz freq.
0	1	1	8 bit UART	Determined by timer 1
1	0	2	9 bit UART	1/32 quartz freq.
1	1	3	9 bit UART	Determined by timer 1

- **SM0** - Serial port mode bit 0 is used for serial port mode selection.
- **SM1** - Serial port mode bit 1.
- **SM2** -Serial port mode 2 bit, also known as multiprocessor communication enable bit.
- **REN** - Reception Enable bit enables serial reception when set. When cleared, serial reception is disabled.

- **TB8** - Transmitter bit 8. Since all registers are 8-bit wide, this bit solves the problem of transmitting the 9th bit in modes 2 and 3. It is set to transmit logic 1 in the 9th bit.
- **RB8** - Receiver bit 8 or the 9th bit received in modes 2 and 3. Cleared by hardware if 9th bit received is logic 0. Set by hardware if 9th bit received is logic 1.
- **TI** - Transmit Interrupt flag is automatically set at the moment the last bit of one byte is sent. It's a signal to the processor that the line is available for a new byte transmission. It must be cleared from within the software.
- **RI** - Receive Interrupt flag is automatically set upon one byte receive. It signals that byte is received and should be read quickly prior to being replaced by a new data. This bit is also cleared from within the software.

m) **SBUF (Serial Buffer, Addresses 99h):** This SFR is used to send and receive data via the on-board serial port. Any value written to SBUF will be sent out the serial port's TXD pin. Likewise, any value which the 8051 receives via the serial port's RXD pin will be delivered to the user program via SBUF.

Serial port must be configured prior to being used. In other words, it is necessary to determine how many bits is contained in one serial "word", baud rate and synchronization clock source. The whole process is in control of the bits of the SCON register (Serial Control).

n) **IE (Interrupt Enable, Addresses A8h):** The Interrupt Enable SFR is used to enable and disable specific interrupts. The low 7 bits of the SFR are used to enable/disable the specific interrupts, where as the highest bit is used to enable or disable ALL interrupts. Thus, if the high bit of IE is 0 all interrupts are disabled regardless of whether an individual interrupt is enabled by setting a lower bit.

IE	EA	--	ET2	ES	ET1	EX1	ET0	EX0
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- **EA** - Global interrupt enable/disable:

0 - Disables all interrupt requests.
1 - Enables all individual interrupt requests.

- **ES** - enables or disables serial interrupt:

0 - UART system cannot generate an interrupt.
1 - UART system enables an interrupt.

- **ET1** - bit enables or disables Timer 1 interrupt:

0 - Timer 1 cannot generate an interrupt.
1 - Timer 1 enables an interrupt.

- **EX1** - bit enables or disables external 1 interrupt:

0 - change of the pin INT0 logic state cannot generate an interrupt.
1 - enables an external interrupt on the pin INT0 state change.

- **ET0** - bit enables or disables timer 0 interrupt:

0 - Timer 0 cannot generate an interrupt.
1 - enables timer 0 interrupt.

- **EX0** - bit enables or disables external 0 interrupt:
  0 - change of the INT1 pin logic state cannot generate an interrupt.
  1 - enables an external interrupt on the pin INT1 state change.

o) IP (Interrupt Priority, Addresses B8h, Bit-Addressable): The Interrupt Priority SFR is used to specify the relative priority of each interrupt. On the 8051, an interrupt may either be of low (0) priority or high (1) priority.

For example, if we configure the 8051 so that all interrupts are of low priority except the serial interrupt, the serial interrupt will always be able to interrupt the system, even if another interrupt is currently executing. However, if a serial interrupt is executing no other interrupt will be able to interrupt the serial interrupt routine since the serial interrupt routine has the highest priority.

IP	--	--	PT2	PS	PT1	PX1	PT0	PX0
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- **PS** - Serial Port Interrupt priority bit
  Priority 0 & Priority 1
- **PT1** - Timer 1 interrupt priority
  Priority 0 & Priority 1
- **PX1** - External Interrupt INT1 priority
  Priority 0 & Priority 1
- **PT0** - Timer 0 Interrupt Priority
  Priority 0 & Priority 1
- **PX0** - External Interrupt INT0 Priority
  Priority 0 & Priority 1

p) PSW (Program Status Word, Address - D0h, Bit-Addressable): The PSW SFR contains the carry flag, the auxiliary carry flag, the overflow flag, and the parity flag. Additionally, the PSW register contains the register bank select flags which are used to select which of the "R" register banks are currently selected.

PSW	CY	AC	F0	RS1	RS0	OV	--	P
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- **P - Parity bit:** If a number stored in the accumulator is even then this bit will be automatically set (1), otherwise it will be cleared (0). It is mainly used during data transmit and receive via serial communication.
- **Bit 1.** This bit is intended to be used in the future versions of microcontrollers.
- **OV Overflow:** This occurs when the result of an arithmetical operation is larger than 255 and cannot be stored in one register. Overflow condition causes the OV bit to be set (1). Otherwise, it will be cleared (0).

- **F0 - Flag 0.** This is a general-purpose bit available for use.
- **AC - Auxiliary Carry Flag** is used for BCD operations only.
- **CY - Carry Flag** is the (ninth) auxiliary bit used for all arithmetical operations and shift instructions.
- **RS0, RS1 - Register bank select bits.** These two bits are used to select one of four register banks of RAM. By setting and clearing these bits, registers R0-R7 are stored in one of four banks of RAM.

RS1	RS0	Space in RAM
0	0	Bank 0 (00h -07h)
0	1	Bank 1 (08h – 0Fh)
1	0	Bank 2 (10h – 1fh)
1	1	Bank 3 (18h – 1Fh)

q) **A (Accumulator, Address E0, Bit addressable):** A register is a general-purpose register used for storing intermediate results obtained during operation. Prior to executing an instruction upon any number or operand it is necessary to store it in the accumulator first. All results obtained from arithmetical operations performed by the ALU are stored in the accumulator. Data to be moved from one register to another must go through the accumulator. In other words, the A register is the most commonly used register and it is impossible to imagine a microcontroller without it. More than half instructions used by the 8051 microcontroller use somehow the accumulator.

r) **B (B Register, Addresses F0h, Bit-Addressable):** The "B" register is used in two instructions; the multiply and divide operations.

9.11 INSTRUCTION SET:

The instruction set of 8051 comprises of the following 5 categories of instructions. They are as follows:

(a) Arithmetic instructions.
(b) Logic instructions.
(c) Data transfer instructions.
(d) Boolean variable manipulation instruction.
(e) Program and machine control instruction.

Mnemonic	Description	Byte	Cycle
Arithmetic Operations			
ADD A,Rn	Add register to accumulator	1	1
ADD A,direct	Add direct byte to accumulator	2	1
ADD A, @Ri	Add indirect RAM to accumulator	1	1
ADD A,#data	Add immediate data to accumulator	2	1
ADDC A,Rn	Add register to accumulator with carry flag	1	1
ADDC A,direct	Add direct byte to A with carry flag	2	1
ADDC A, @Ri	Add indirect RAM to A with carry flag	1	1
ADDC A, #data	Add immediate data to A with carry flag	2	1
SUBB A,Rn	Subtract register from A with borrow	1	1
SUBB A,direct	Subtract direct byte from A with borrow	2	1
SUBB A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB A,#data	Subtract immediate data from A with borrow	2	1
INC A	Increment accumulator	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	1
INC @Ri	Increment indirect RAM	1	1
DEC A	Decrement accumulator	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	1
DEC @Ri	Decrement indirect RAM	1	1
INC DPTR	Increment data pointer	1	2
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	4
DA A	Decimal adjust accumulator	1	1

Logic Operations

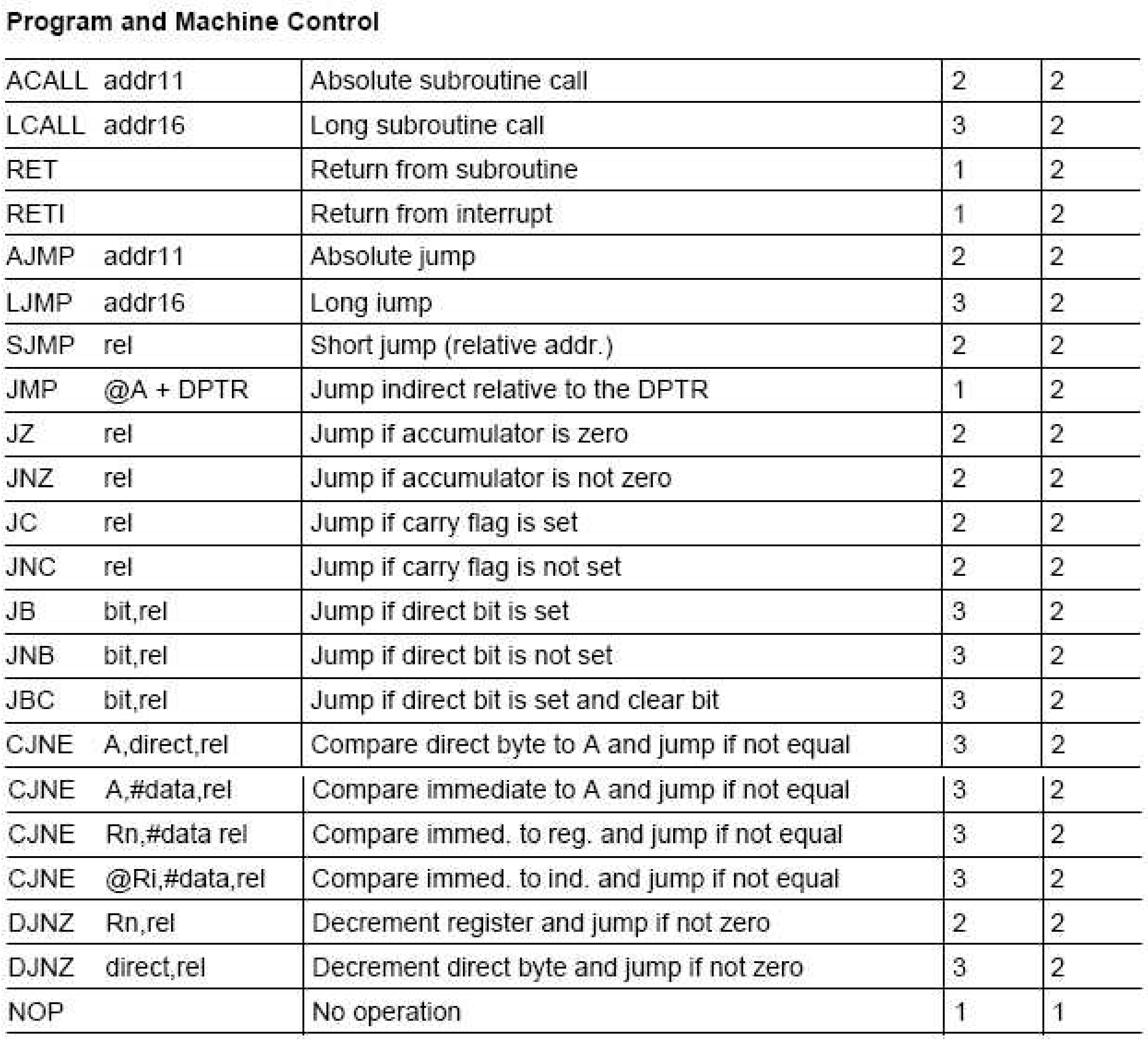
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

Data Transfer

MOV A,Rn	Move register to accumulator	1	1
MOV A,direct *)	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate data to indirect RAM	2	1
MOV DPTR, #data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A + DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A + PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A,Rn	Exchange register with accumulator	1	1
XCH A,direct	Exchange direct byte with accumulator	2	1
XCH A,@Ri	Exchange indirect RAM with accumulator	1	1
XCHD A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

Boolean Variable Manipulation

CLR C	Clear carry flag	1	1
CLR bit	Clear direct bit	2	1
SETB C	Set carry flag	1	1
SETB bit	Set direct bit	2	1
CPL C	Complement carry flag	1	1
CPL bit	Complement direct bit	2	1
ANL C,bit	AND direct bit to carry flag	2	2
ANL C,/bit	AND complement of direct bit to carry	2	2
ORL C,bit	OR direct bit to carry flag	2	2
ORL C,/bit	OR complement of direct bit to carry	2	2
MOV C,bit	Move direct bit to carry flag	2	1
MOV bit,C	Move carry flag to direct bit	2	2



Program and Machine Control

ACALL addr11	Absolute subroutine call	2	2
LCALL addr16	Long subroutine call	3	2
RET	Return from subroutine	1	2
RETI	Return from interrupt	1	2
AJMP addr11	Absolute jump	2	2
LJMP addr16	Long iump	3	2
SJMP rel	Short jump (relative addr.)	2	2
JMP @A + DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if accumulator is zero	2	2
JNZ rel	Jump if accumulator is not zero	2	2
JC rel	Jump if carry flag is set	2	2
JNC rel	Jump if carry flag is not set	2	2
JB bit,rel	Jump if direct bit is set	3	2
JNB bit,rel	Jump if direct bit is not set	3	2
JBC bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE Rn,#data rel	Compare immed. to reg. and jump if not equal	3	2
CJNE @Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ Rn,rel	Decrement register and jump if not zero	2	2
DJNZ direct,rel	Decrement direct byte and jump if not zero	3	2
NOP	No operation	1	1

9.12 DATA ADDRESSING MODES:

While operating, the processor processes data as per program instructions. Each instruction consists of two parts. One part describes WHAT should be done, while the other explains HOW to do it. The latter part can be a data (binary number) or the address at which the data is stored. Two ways of addressing are used for all 8051 microcontrollers depending on which part of memory should be accessed:

1. **Immediate addressing:** Immediate addressing is so-named because the value to be stored in memory immediately follows the op-code in memory, i.e. the instruction itself shows the value will be stored in memory.

   Ex.: MOV A, #20h

   Immediate addressing is very fast as the value to be loaded is included in the instruction.

2. **Direct addressing:** In this mode, the value to be stored in memory is directly retrieving from another memory location.

   Ex.: MOV A, 30h

   This instruction will fetch the data in internal RAM address 30 and load it into Accumulator. This addressing is generally fast as it fetches the data from the internal RAM of 8051.

3. **Indirect Addressing:** Indirect addressing is a very powerful addressing mode which in many cases provides an exceptional level of flexibility.

   Ex: MOV A, @R0

   The 8051 will load the accumulator with the value from Internal RAM which is found at the address indicated by R0.

   For example, let's say R0 holds the value 40h and Internal RAM address 40h holds the value 67h. When the above instruction is executed the 8051 will the Accumulator ends up holding 67h.

   Indirect addressing always refers to Internal RAM; it never refers to an SFR.

4. **External Direct Addressing:** External Memory is accessed using a suite of instructions which use "External Direct" addressing. It is because this appears to be direct addressing, but it is used to access external memory rather than internal memory.

   There are only two commands that use External Direct addressing mode:

   MOVX A,@DPTR

   MOVX @DPTR,A

   Once DPTR holds the correct external memory address, the first command will move the contents of that external memory address into the Accumulator and the second command will do the opposite.

5. **External Indirect Addressing:** This form of addressing is usually only used in relatively small projects that have a very small amount of external RAM. An example of this addressing mode is: MOVX @R0, A

   The value of R0 is first read and the value of the Accumulator is written to that address in External RAM.

9.13 8051 Microcontroller Interrupts

There are five interrupt sources for the 8051, which means that they can recognize 5 different events that can interrupt regular program execution.

- Timer 0 Overflow – TF0
- Timer 1 Overflow – TF1
- Reception/Transmission of Serial Character – TI / RI
- External Event 0 – INT0
- External Event 1 – INT1

Each interrupt can be enabled or disabled by setting bits of the IE register. Likewise, the whole interrupt system can be disabled by clearing the EA bit of the IE register.

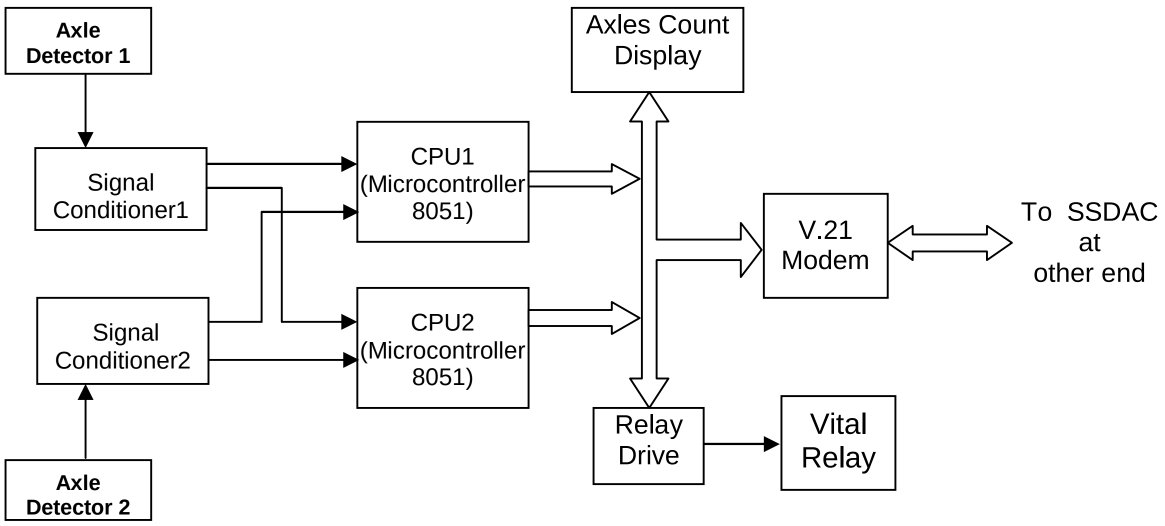


Fig.9.6 Block Diagram of SSDAC System

The Microcontroller Blocks are the heart of the unit. Two identical blocks are implemented to have 2 out-of 2 decision. These blocks basically perform all the functions related to Pulse Counting, Count Validation, Communication with adjacent Unit SSDAC and decision making to give output. If the results are not matching then the system goes into fail-safe operation.

9.14 DTMF Control Office Equipment

Another important application of 8051 is in the Control Office Equipment of Train Taffic Control Communication system. The DTMF headquarters Equipment initially supplied by M/s Tummala Electronics used 8749 microcontroller which is the predecessor of present 8051. Later on 8749 is replaced by 8051 itself, in all new supplies. Let us have a look into both these design types. The following block diagram in fig.6.2 shows the use of 8749 microcontroller in the first design of DTMF Headquarters equipment. The second block diagram in fig.6.3, shows the design of the same equipment using 8951 which is same as 8051 except for the on-chip Flash ROM. In both these designs you can notice the change in interfacing peripherals used.

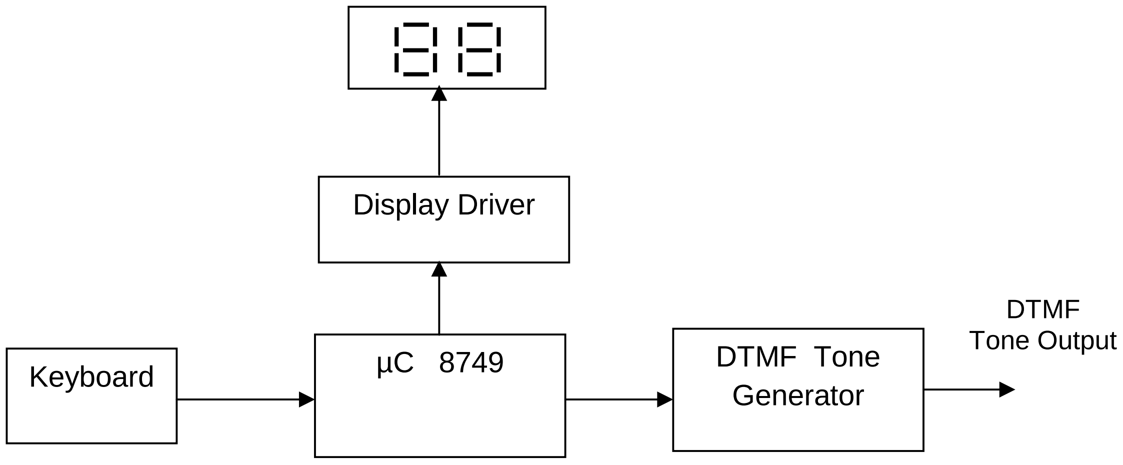


Fig. 9.7 First design of DTMF Headquarters Equipment

Objective:

1) Name some appliance found in daily life that uses 8051_________________.
2) No. of 8051 used in SSDAC__________________________.
3) The microcontroller used in the first design of DTMF HQ equipment.__________________
4) The microcontroller used in second design of DTMF HQ equipment.__________________
5) Mention any other application of 8051 which is seen in outside world._________________

Subjective:

1) Mention some applications of 8051 in S&T department with brief description about each.
2) Explain about 8051 role in digital axle counter with a simple block diagram.
3) Explain about any DTMF control office equipment using 8051 with a block diagram.