RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL
New Scheme Based On AICTE Flexible Curricula
Information Technology, III-Semester
IT304 Digital Circuits and Systems

Course Objectives
1 Understand working of logic gates.
2 To design and implement combinational and sequential logic circuits
3 Understand the process of analog to digital and digital to analog conversion
4 To understand various logic families

Unit I- Number systems and logic gates: Decimal, Binary, Octal, Hexadecimal number systems and radix conversion. Codes- BCD, excess 3, gray, ASCII. Boolean algebra-Theorems and properties, Boolean functions, canonical and standard forms, De Morgans theorem, digital logic gates, Karnaugh maps.

Unit II- Combinational circuits: Introduction to combinational circuits, multilevel NAND, NOR implementation. Designing binary Adders and Subtractors. Decoder, Encoder, Multiplexer, Demultiplexer circuits.

Unit III- Sequential circuits: Introduction to Sequential circuits, flip-flops, RS, D, T, JK, M/S JK-flipflops, truth tables, excitation tables and characteristic equations, clocked and edge triggered flipflops, Registers- Definition, serial, parallel, shift left/right registers, Johnson counter, asynchronous and synchronous counters.

Unit IV- Digital logic families: Bipolar and unipolar logic families, Digital IC specifications, RTL, DTL, All types of TTL circuits, ECL, IIL, PMOS, NMOS & CMOS Logic.

Unit V- Clocks and timing circuits: Bistable, Monostable & Astable multivibrator, Schmitt trigger circuit, Introduction of Analog to Digital & Digital to Analog converters, Display devices, 7 and 16 segment LED display, LCD.

Course Outcomes
On the completion of this course
1 Students will be able to perform number base conversions, use Boolean logic to create digital circuits.
2. Student can understand use of encoders, decoders, multiplexers and demultiplexers in communication systems.
3 By learning design of combinational and sequential circuits student can understand its use in digital systems such as computers, communication systems and other modern technologies.
4 Study of ADC and DAC along with display devices will enable students to understand signal conversion and its display and their applications in digital devices.

Reference Books:
1. M. Morris Mono, “Digital logic design”, Pearson Education Pvt. Ltd.
2. A Anand Kumar, “Fundamentals of digital circuits“, PHI Learning Pvt Ltd.
3. A K Maini, “Digital Electronics Principles and Integrated Circuits, Wiley India Pvt Ltd.
4. R P Jain, “Modern Digital Electronics”, Tata McGraw-Hill publishing company Ltd.
5. D P Kothari and J S Dhillon, "Digital Circuits and Design", Pearson Education Pvt. Ltd.

List of Experiments:

1. Study and verify the operation of AND, OR, NOT, NOR and NAND logic gates.

2. Design all basic logic gates using NOR universal gate.

3. Design all basic logic gates using NAND universal gate.

4. Verification of Demorgan’s theorem.

5. Construction and verification of half adder andfull adder circuits.

6. Construction and verification of half subtractor and full subtractor circuits.

7. Design of Binary to Grey & Grey to Binary code Converters .

8. Design of BCD to excess-3 code converter.

9. Design and verification of Multiplexer circuit

10. Design and verification of De-multiplexer circuit.

Subject Name: Digital Circuit And Systems **Subject Code: IT 304**

Subject Notes

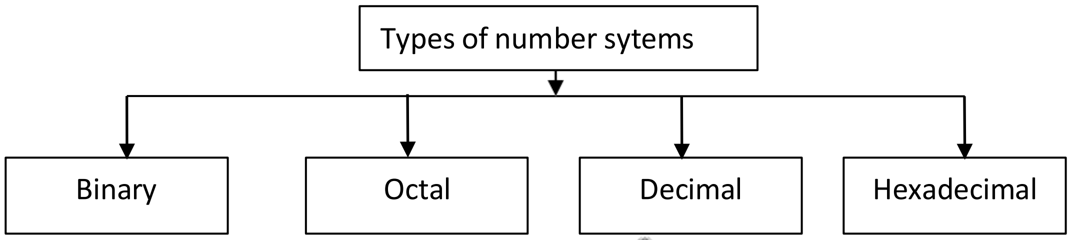
UNIT-I

Number systems and logic gates: Decimal, Binary, Octal, Hexadecimal number systems and radix conversion. Codes- BCD, excess 3, gray, ASCII. Boolean algebra- Theorems and properties, Boolean functions, canonical and standard forms, De Morgans theorem, digital logic gates, Karnaugh maps

---

1.1 NUMBER SYSTEMS:

Number systems are the presentation of information which is used in all operations in information processing systems. The information is divided into a group of symbols in number system; like 26 English letters and 10 decimal digits.



Binary Number System : - This number system has a base or radix of 2. The symbols or digits used to represent the any number in this system are 0 & 1.

Octal Number System : - This number system has a base or radix of 8. The symbols or digits used to represent the any number in this system are 0 through 7 i.e.(0, 1, 2, 3, 4, 5, 6, 7)

Decimal Number System :- This number system has a base or radix of 10. The symbols or digits used to represent the any number in this system are 0 through 9 i.e. (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Hexadecimal Number System :- This number system has a base or radix of 16. The symbols or digits used to represent the any number in this system are 0 through F i.e. (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

1.1.1 CONVERSION OF NUMBER SYSTEM

➢ **BINARY TO DECIMAL CONVERSION:-**

$(1110 . 11010)_2 = (\ ?\ )_{10}$

Integeral part : $(1110)_2 = (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$= 8 + 4 + 2 + 0$

$= 14$

ie. $(1111)_2 = (14)_{10}$

Fractional Part : $(0.11010)_2 = (1\times 1/2) + (1\times 1/4) + (0\times 1/8) + (1 \times 1/16) + (0\times 1/32)$

$= 0.5 + 0.25 + 0 + 0.0625 + 0$

$= 0.8125$

ie. $(0.11010)_2 = (0.8125)_{10}$

Thus $(1111.11010)_2 = (14 . 8125)_{10}$

Note: To convert any binary number into decimal number system following points should be consider

1. In conversion of integral part binary digit is multiplied by power of 2 in increasing order from right to left.
2. In conversion of fractional part binary digit is divided by power of 2 in increasing order from left to right.

➢ **DECIMAL TO BINARY CONVERSION :-**

$(14 . 812)_{10} = (?)_2$

<u>Integeral part</u> :

2	14	0
2	7	1
2	3	1
	1	

$\boxed{\text{ie.} \quad (14)_{10} = (1110)_2}$

<u>Fractional Part</u> :

$(0.812 \times 2) = 1.624 \rightarrow 1$
$(0.624 \times 2) = 1.248 \rightarrow 1$
$(0.248 \times 2) = 0.496 \rightarrow 0$
$(0.496 \times 2) = 0.992 \rightarrow 0$
$(0.992 \times 2) = 1.984 \rightarrow 1$

$\boxed{\text{ie. } (0 . 812)_{10} = (0.11001)_2}$

Thus $(15 . 812)_{10} = (1111 . 11001)_2$

Note: To convert any decimal number into binary number system following points should be consider

1. In conversion of integral part decimal number is divided by 2 and remainder is noted in side. After division the remainders are arranged from bottom to top.
2. In conversion of fractional part decimal number is multiplied by 2 and integer part of number is noted in side. After multiplication the integers are arranged from top to bottom.

➢ **BINARY TO OCTAL CONVERSION:**

$(10001110 . 01101)_2 = (?)_8$

Integeral part :

$(10001110)_2 = \{ \ 010, \ 001, \ 110 \ \}$
$= (\underbrace{010,}_{2} \ \underbrace{001,}_{1} \ \underbrace{110}_{6})_2$

ie. $(10001110)_2 = (216)_8$

Fractional Part :

$(0. 01101)_2 = \{ \ \underbrace{011,}_{3} \ \underbrace{010}_{2} \}$

ie. $(0. 00101)_2 = (0.12)_8$

Thus $(10011110 . 00101)_2 = (216 . 32)_8$

Note: To convert any binary number into octal number system following points should be consider

1. In conversion of integral part binary digits are grouped in pair of three digits from left to right and if the digit in right is less then three put zero as required. After making the pair equivalent decimal number is obtained.

2. In conversion of fractional part binary digit is divided by power of base i.e. 2 in increasing order from left to right.

- **OCTAL TO BINARY CONVERSION:**

  $(404.245)_8 = (?)_2$

Integeral part :

$$(404)_8 = \{\overbrace{100}^{4}\quad\overbrace{000}^{0}\quad\overbrace{100}^{4}\} = (100000100)_2$$

ie. $(404)_8 = (100000100)_2$

Fractional Part :

$$(0.245)_8 = \{\overbrace{010}^{2}\quad\overbrace{100}^{4}\quad\overbrace{101}^{5}\} = (0.010100101)_2$$

Thus $(404.245)_8 = (100000100.010100101)_2$

Note: To convert any octal number into binary number system following points should be consider

1. In conversion of integral part octal digits are converted into equivelant binary digits in group in pair of three digits.

2. In conversion of fractional part same prosess will be followed as for intregal part.

- **BINARY TO HEXADECIMAL CONVERSION:**

$(1001111010100110.001011111010)_2 = (?)_{16}$

Integeral part :

$$(1001111010100110)_2 \xleftarrow{\qquad} = \{1001,\quad 1110,\quad 1010,\quad 0110\}$$

$$= (\underbrace{1001}_{9},\quad \underbrace{1110}_{E},\quad \underbrace{1010}_{A},\quad \underbrace{0110}_{6})_2$$

i.e. $(1001111010100110)_2 = (9EA6)_{16}$

Fractional Part:

$$(0.001011111010)_2 = \xrightarrow{\qquad} \{\underbrace{0010}_{2},\quad \underbrace{1111}_{F},\quad \underbrace{1010}_{A},\}$$

i.e. $(0.001011111010)_2 = (0.2FA)_{16}$

Thus $(1001111010100110.001011111010)_2 = (9EA6.2FA)_{16}$

Note: To convert any binary number into hexadecimal number system following points should be consider

1. In conversion of integral part binary digits are grouped in pair of four digits from left to right and if the digit in right is less then four put zero as required. After making the pair equivalent hexadecimal number is obtained.

2. In conversion of fractional part binary digit is divided by power of base i.e. 2 in increasing order from left to right.

➢ **HEXADECIMAL TO BINARY CONVERSION:**

$(99E\ .\ 2FA)_{16} = (\ ?\ )_2$

Integeral part :

$(99E)_{16} = \{\overbrace{1001}^{9}\quad \overbrace{1001}^{9}\quad \overbrace{1110}^{E}\} = (10011001110)_2$

ie. $(99E)_{16} = (10011001110)_2$

Fractional Part :

$(0\ .\ 2FA)_{16} = \{\overbrace{0010}^{2}\quad \overbrace{1111}^{F}\quad \overbrace{1010}^{A}\} = (0.\ 001011111010)_2$

Thus $(99E\ .\ 2FA)_{16} = (1001111010100110\ .\ 001011111010)_2$

Note: To convert any hexadecimal number into binary number system following points should be consider
1. In conversion of integral part hexadecimal digits are converted into equivelant binary digits in pair of four binary digits.
2. In conversion of fractional part same prosess will be followed as for intregal part.

➢ **OCTAL TO DECIMAL CONVERSION:-**

$(57\ .\ 245)_8 = (\ ?\ )_{10}$

Integeral part :

$(57)_8 = (5 \times 8^1) + (7 \times 8^0)$
$= 40 + 7$
$= 47$

ie. $(57)_8 = (47)_{10}$

Fractional Part :

$(0.245)_8 = (2\times 1/8) + (4\times 1/64) + (5\times 1/512)$
$= 0.25 + 0.0625 + 0.0097$
$= 0.3222$

ie. $(0.245)_8 = (0.3222)_{10}$

Thus $(57.245)_8 = (47\ .\ 3222)_{10}$

Note: To convert any octal number into decimal number system following points should be consider
1. In conversion of integral part octal digit is multiplied by power of 8 in increasing order from right to left.
2. In conversion of fractional part octal digit is divided by power of 8 in increasing order from left to right.

➢ **DECIMAL TO OCTAL CONVERSION :-**

$(303\ .\ 322)_{10} = (\ ?\ )_8$

Integeral part :

8	303	7 ↑
8	37	5
	4	

ie. $(303)_{10} = (457)_8$

Fractional Part :

$(0.322 \times 8) = 2.576 \rightarrow 2$
$(0.576 \times 8) = 4.608 \rightarrow 4$
$(0.608 \times 8) = 4.864 \rightarrow 4$
$(0.864 \times 8) = 6.912 \rightarrow 6$
$(0.912 \times 8) = 7.296 \rightarrow 7 \downarrow$

$\boxed{\text{ie. } (0 . 322)_{10} = (0 . 24467)_8}$

Thus $(303 . 322)_{10} = (457 . 24467)_8$

Note: To convert any decimal number into octal number system following points should be consider

1. In conversion of integral part decimal number is divided by 8 and remainder is noted in side. After division the remainders are arranged from bottom to top.
2. In conversion of fractional part decimal number is multiplied by 8 and integer part of number is noted in side. After multiplication the integers are arranged from top to bottom.

➢ **OCTAL TO HEXADECIMAL CONVERSION:**
$(174654 . 273054)_8 = (?)_{16}$

Integeral part :

$(174654)_8 = \{ \underbrace{1}_{001} \quad \underbrace{7}_{111} \quad \underbrace{4}_{100} \quad \underbrace{6}_{110} \quad \underbrace{5}_{101} \quad \underbrace{4}_{100} \}$

$= (\underbrace{0000}_{0}, \underbrace{1111}_{F}, \underbrace{1001}_{9}, \underbrace{1010}_{A}, \underbrace{1100}_{C})_2$

ie. $(174654)_8 = (F9AC)_{16}$

Fractional Part :
$(0.273054)_8 = \underbrace{2}_{010} \quad \underbrace{7}_{111} \quad \underbrace{3}_{011} \quad \underbrace{0}_{000} \quad \underbrace{5}_{101} \quad \underbrace{4}_{100}$

$\longrightarrow$

$= (0 . \underbrace{0101}_{5}, \underbrace{1101}_{D}, \underbrace{1000}_{8}, \underbrace{1011}_{B}, \underbrace{0000}_{0})$

ie. $(0.273054)_8 = (0 . 5D8B0)_{16}$
Thus $(174654 . 273054)_8 = (F9AC . 5D8B)_{16}$

Note: To convert any octal number into hexadecimal number system following points should be consider

1. In conversion of integral part octal digits are converted in binary and grouped in pair of four binary digits. After grouping the digit equivalent hexadecimal number is obtainted.
2. In conversion of fractional part same prosess will be followed as for intregal part.

➢ **HEXADECIMAL TO OCTAL CONVERSION:**
$(F9AC . 5D8B)_{16} = (?)_8$

Integeral part :

$(F9AC)_{16} = \{ \underbrace{F}_{1111} \quad \underbrace{9}_{1001} \quad \underbrace{A}_{1010} \quad \underbrace{C}_{1100} \} = (1111100110101100)_2$

$\longleftarrow$

ie. $(F9AC)_{16} = (1, \quad 111, \quad 100, \quad 110, \quad 101, \quad 100)_2$

$= (001, \quad 111, 100, \quad 110, \quad 101, \quad 100)_2 \quad = \quad 174654$

ie. $(F9AC)_{16} = (174654)_8$

Fractional Part :

$(0.5D8B)_{16} = \underbrace{5}\ \underbrace{D}\ \underbrace{8}\ \underbrace{B}$

$\{0101 \quad 1101 \quad 1000 \quad 1011\} = (0.010, 111, 011, 000, 101, 100)_2$

$\longrightarrow$

$= (\underbrace{010,}_{2} \quad \underbrace{111,}_{7} \quad \underbrace{011,}_{3} \quad \underbrace{000,}_{0} \quad \underbrace{101,}_{5} \underbrace{100}_{4})_2$

$= (0.273054)_8$

ie. $(0.5D8B) = \ = (0.273054)_8$

Thus $(F9AC.5D8B)_{16} = (174654 \quad .273054)_8$

- **DECIMAL TO HEXADECIMAL CONVERSION:**

$(3750.365)_{10} = (?)_{16}$

Integeral part :

16	3750	6	$\longrightarrow$ 6
16	234	10	$\longrightarrow$ A
	14		$\longrightarrow$ E

$\boxed{\text{ie.} \quad (3750)_{10} = (EA6)_{16}}$

Fractional Part :

$(0.365 \times 16) = 5.84 \rightarrow 5 \rightarrow 5$

$(0.84 \times 16) = 13.44 \rightarrow 13 \rightarrow D$

$(0.44 \times 16) = 7.04 \rightarrow 7 \rightarrow 7$

$(0.04 \times 16) = 0.64 \rightarrow 0 \rightarrow 0$

$(0.64 \times 16) = 10.24 \rightarrow 10 \rightarrow A$

$\boxed{\text{ie. } (0.365)_{10} = (0.5D70A)_{16}}$

Thus $(3750.365)_{10} = (EA6.5D70A)_{16}$

- **HEXADECIMAL TO DECIMAL CONVERSION:-**

$(EA6.2FA)_{16} = (?)_{10}$

Integeral part : $(EA6)_{16} = (E \times 16^2) + (A \times 16^1) + (6 \times 16^0)$

$= (14 \times 16^2) + (10 \times 16^1) + (6 \times 1)$

$= 3584 + 160 + 6$

$= 3750$

ie. $(EA6)_{16} = (3750)_{10}$

Fractional Part : $(0.2FA)_{16} = (2 \times 1/16) + (F \times 1/16^2) + (A \times 1/16^3)$

$= (2 \times 1/16) + (15 \times 1/256) + (10 \times 1/4096)$

$= 0.125 + 0.0586 + 0.00244$

$= 0.18604$

ie. $(0.2FA)_{16} = (0.18604)_{10}$

Thus $(EA6.2FA)_8 = (3750.18604)_{10}$

1.2 CODES

Numbers, letters or words are represented by a specific group of symbols, called code.

1.2.1 Weighted code: Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Example: Straight bit binary code, BCD code.

➢ **Straight bit binary code:**

Decimal Number	2			
Positional weights	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1
Binary Code	0	0	1	0

➢ **Binary Coded Decimal (BCD) code:**

In this code each decimal digit (0 to 9) is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the Binary, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

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

Advantages of BCD Codes

1. It is very similar to decimal system.
2. We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes

1. The addition and subtraction of BCD have different rules.
2. The BCD arithmetic is little more complicated.
3. BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

1.2.2 Non-weighted code: In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

➢ **Excess-3 code**

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding $(0011)_2$ or $(3)_{10}$ to each code word in 8421. The excess-3 codes is obtained, as follows

Decimal Number $\longrightarrow$ (8421) BCD $\longrightarrow$ ADD 3 i.e.(+0011) $\longrightarrow$ Excess-3

Example:

Decimal	BCD				Excess-3			
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1

➢ **Gray Code**

It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that has only one bit will change each time the decimal number is incremented . As only one bit changes at a time, the gray code is called as a unit distance code. Gray code cannot be used for arithmetic operation.

3 Alphanumeric codes

The alphanumeric codes are the codes that represent numbers and alphabetic characters. Mostly such codes also represent other characters such as symbol and various instructions necessary for conveying information. An alphanumeric code should at least represent 10 digits and 26 letters of alphabet i.e. total 36 items. The following three alphanumeric codes are very commonly used for the data representation.

1. American Standard Code for Information Interchange (ASCII).
2. Extended Binary Coded Decimal Interchange Code (EBCDIC).
3. Five bit Baudot Code.

ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code. ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.

CODE CONVERSION:

1. (0010)$_{BCD}$ to gray code

Steps: 1) Write the MSB bit as it is i.e. 0

2) Start EXORing the consecutive bits from LHS i.e. 0 (EXOR) 0 = 0

3) 0 (EXOR) 1 = 1, 1 (EXOR) 0 = 1.

4) Final Result. (0011) $_{gray}$

BCD code				Gray code			
0	0	1	0	0	0	1	1

2. (0010)$_{gray}$ to BCD code

Steps: 1) Write the MSB bit as it is i.e. 0

2) Starting from LHS, EXORing the result obtained in step 1 with 2nd bits i.e. 0 (EXOR) 0 = 0

3) Follow step 2.

4) Final Result. (0010) $_{BCD}$

Gray code				BCD code			
0	0	1	1	0	0	1	0

<u>1.3 BINARY ARITHMETIC:</u>

Unsigned Binary Numbers: In unsigned binary numbers all the bits are used for representing only the magnitude of the corresponding decimal number. For example, the smallest 8 bit binary number is 0000 0000 and the largest 8 bit binary number is 1111 1111. Hence the total range of unsigned 8 bit binary number is from $(00)_H$ to $(FF)_H$ or from $(00)_{10}$ to $(255)_{10}$. With 16- bit binary numbers, the total range is from $(0000)_H$ to $(FFFF)_H$

1	0	1	1	0	1	0	0

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{All the bits are used for representing only the magnitude}}$$

Sign- Magnitude Numbers: Binary numbers which contains a sign bit followed by magnitude bits are called Sign- Magnitude Numbers. 0 is used to represent the (+) sign and 1 is used to represent a (-) sign. The MSB of binary number is used to represent the sign and remaining bit is used to represent the magnitude. MSB represents the sign and rest of bits represent the magnitude.

1 / 0	0	1	1	0	1	0	0
MSB	**MAGNITUDE**						

For an 8-bit sign-magnitude number, the largest negative number is $(-127)_{10}$ and positive number is $(+127)_{10}$.

BINARY ADDITION:-

1. 0 + 0 = Sum 0 with carry of 0.
2. 0 + 1 = Sum 1 with carry of 0.
3. 1 + 0 = Sum 1 with carry of 0.
4. 1 + 1 = Sum 0 with a carry of 1.
5. 1 + 1 + 1 = Sum 1 with carry of 1.

Example: Add $(111011.1101)_2$ with $(011111.0110)_2$

	1	1	1	1	1	1		1				**carry**
	1	1	1	0	1	1	.	1	1	0	1	**Augend**
+	0	1	1	1	1	1	.	0	1	1	0	**Addend**
1	0	1	1	0	1	1	.	0	0	1	1	**sum**

BINARY SUBTRACTION:

The basic principles of binary subtraction include the following:

A) 0 – 0 = 0. **B)** 1 – 0 = 1. **C)** 1 – 1 = 0. **D)** 0 – 1 = 1 with a borrow of 1 from the next more significant bit.

Example: Subtract $(11111.011)_2$ from $(111011.1101)_2$

	1	1	1	0	1	1	.	1	1	0	1	**Minuend**
–	0	1	1	1	1	1	.	0	1	1	0	**Subtraend**
	0	1	1	1	0	0	.	0	1	1	1	**Difference**

BINARY SUBTRACTION USING 1'S COMPLEMENT METHOD:

1's complement of any binary number is obtained by subtracting each binary bit by 1.

For example: $(1110011)_2 \xrightarrow{\text{1's complement}}$ $(1111111 – 1110011) = (0001100)_2$

Example(a) Subtract (5 – 3) using 1's complement method.

$(5)_{10} = (0\ 1\ 0\ 1)_2$ and $(3)_{10} = (0\ 0\ 1\ 1)_2$

First obtained 1's complement of negative number i.e.1's complement of 3 = 1 1 0 0

Second add the result with positive number i.e

	0	1	0	1
+	1	1	0	0
End around carry 1	0	0	0	1

Now in the result we can see that there is an overflowing bit/end around carry which we have to add with the remaining result.

	0	0	0	1
+				1
	0	0	1	0

$(0010)_2=(2)_{10}$ is the desired result.

(b) Subtract (3 – 5) using 1's complement method.

$(5)_{10} = (0\ 1\ 0\ 1)_2$ and $(3)_{10} = (0\ 0\ 1\ 1)_2$

First obtained 1's complement of negative number i.e.1's complement of 5 = 1 0 1 0
Second add the result with positive number i.e

	0	0	1	1
+	1	0	1	0
No carry bit generated	1	1	0	1

From the result, we can see that no carry bit/end around carry bit is generated. So to get the desired result, take the 1's complement of the result and attach a negative sign i.e.
1's complement of $(1101)_2$ is $-(0010)_2$
$-(0010)_2 = -(2)_{10}$ is the desired result.

BINARY SUBTRACTION USING 2'S COMPLEMENT METHOD:

2's complement is obtained by adding 1 to the 1's complement.
For example, we have to find out 2's complement of 0100.
we have to subtract 0100 from 1111 since it is the highest four digit number to find out 1's complement i.e. 1111 – 0100 = 1011. Hence, 2's complement will be 1011 + 1 = 1100.

Example (a) Subtract (65 – 63) using 2's complement method.

1. Find 2's complement of the negative number.i.e.
   2's complement of 63 = 1000001
2. Add the positive number with 2's complement of the negative number.i.e.

	1	0	0	0	0	0	1
+	1	0	0	0	0	0	1
1 Carry generated	0	0	0	0	0	1	0

3. Discard the carry generated.
4. After discarding the carry, keep the result which is the result of subtraction.
   i.e. Final answer is (0000010)

(b) Subtract (63 – 65) using 2's complement method.

1. Find 2's complement of the negative number.i.e.2's complement of 65 = 0111111
2. Add the positive number and 2's complement of the negative number.i.e.

	0	1	1	1	1	1	1
+	0	1	1	1	1	1	1
No Carry generated	1	1	1	1	1	1	0

From the result, we can see that no carry bit/end around carry bit is generated. So to get the desired result, take the 2's complement of the result and attach a negative sign i.e.
2's complement of $(1111110)_2$ is $-(0000010)_2$ **i.e. Final answer is –(0000010).**

BINARY MULTIPLICATION:

The basic rules of multiplication are listed as follows:
1. 0 × 0 = 0.
2. 0 × 1 = 0.
3. 1 × 0 = 0.
4. 1 × 1 = 1.

Example: Multiply $(10.11)_2$ by $(11)_2$

		1	0	.	1	1	**Multiplicand**
			X		1	1	**Multiplier**
		1	0	.	1	1	
+	1	0	1	.	1	0	
1	0	0	0	.	0	1	**Product**

BINARY DIVISION:

Example: Divide $(110001)_2$ by $(111)_2$

Divisor 111	110001-Divident -0111	111 Quotient
	01010 -111	
	0111 -111	
	0000-Remainder	

BOOLEAN ALGEBRA:

Boolean algebra or switching algebra is a system of mathematical logic to perform different mathematical operations in binary system. There is only three basis binary operations, AND, OR and NOT by which all simple as well as complex binary mathematical operations are to be done. There are many rules in Boolean algebra by which those mathematical operations are done. In Boolean algebra, the variables are represented by Capital Letter like A, B, C etc and the value of each variable can be either 1 or 0, nothing else. In Boolean algebra an expression given can also be converted into a logic diagram using different logic gates like AND gate, OR gate and NOT gate, NOR gates, NAND gates, XOR gates, XNOR gates etc.

Some basic logical Boolean operations,

AND operation	**OR operation**	**NOT operation**
0.0 = 0	0 + 0 = 0	(1)’ = 0
1.0 = 0	0 + 1 = 1	(0)’ = 1
0.1 = 0	1 + 0 = 1	
1.1 = 1	1 + 1 = 1	

Some basic laws and theorems for Boolean Algebra:

1] Idempotent Law : (i) A + A = A (ii) A . A = A

2] Commutative Law : (i) A+ B = B+ A (ii) A. B = B . A

3] Associative Law : (i) (A + B) + C = A + (B+ C) (ii) (A . B) . C = A . (B . C)

4] Distributive Law : (i) A + (B . C) = (A + B) . (A+ C) (ii) A . (B+ C) = (A . B)+ (A . C)

5] Complement Law : (i) A + A’ = 1 (ii) A . A’ = 0

6] Double Complement Law : (A’)’ = A

7] Identity Law : (i) A + 0 = A (ii) A . 1 = A

8] Null (Dominance) Law : (i) A + 1 = 1 (ii) A . 0 = 0

9] Absorption Law : (i) A. (A + B) = A (ii) A + (A . B) = A

(iii) A . (A’+B) = (A . B) (iv) A + A’. B = (A + B)

De Morgan’s Theorem:

I Theorem : Complement of sum is equal to the product of complements.

(A+B)’ = A’.B’

II Theorem : Complement of product is equal to the sum of complements.

(A.B)′ = A′ + B′

Proof :

Inputs		Outputs			
A	B	(A+B)′	A′.B′	(A.B)′	A′+B′
0	0	1	1	1	1
0	1	0	0	1	1
1	0	0	0	1	1
1	1	0	0	0	0

Column for (A+B)′ and A′.B′ are same. Column for (A.B)′ and A′ + B′ are same. Hence proved.

DUALITY THEOREM :

The Duality theorem states that the Dual of a Boolean expression can be obtained by :
(i) Replacing OR operator by AND operator. (iii) Replacing AND operator by OR operator.
(ii) Replacing 0s by 1s and 1s by 0s.
(iv) Complementing the variables ie. A is replaced by A′ and A′ is replaced by A.
Example of Duality principle are stated below :
(i) A + A = A (ii) A . A = A BY DUALITY :- (i) A′ . A′ = A′ (ii) A′ + A′ = A′

<u>APPLICATION OF BOOLEAN ALGEBRA</u> :

Simplification of Boolean Functions:-

(1) Y = A.B.C + A.B′.C + A.B.C′

Y= A.C.(B+B′) + A.B.C′ {B + B′ = 1 Complementation law}
Y= A.C.(1) + A.B.C′ {A.C .1 = A.C – Identity law}
Y= A.C + A.B.C′
Y= A.[C + C′.B] {C + C′.B = C + B –Absorption law}
Y= A.[C+B]

(2) Y = A.B.C′ + A.B′.C′ + A.B.C + A.B′.C + A′.B.C

Y = A.B.(C+C′) + A.B′. (C+C′) + A′.B.C {C + C′ = 1 - Complementation law}
Y = A.B.(1) + A.B′.(1) + A′.B.C {A.1 = A - Identity law}
Y = A.B + A.B′ + A′.B.C
Y = B.[A + A′.C] + A.B′ {A+A′.C = A+C - Absorption law}
Y= B.[A+C] + A.B′ {Using Distributive law}
Y= A.B + B.C + A.B′
Y = A.[B+B′] + B.C {B + B′ = 1 - Complementation law}
Y = A.(1) + B.C {A.1 = A - Identity law}
Y = A + B.C

(3)Y = A.B.C + A′.B.C + A.B′.C + A.B.C′ + A′.B′.C′

Y = A.B.(C+C′) + A′.B.C + A.B′.C + A′.B′.C′ {C + C′ = 1 - Complementation law}
Y = A.B.(1) + A′.B.C + A.B′.C + A′.B′.C′
Y = A.(B+B′.C) + A′.B.C + A′.B′.C′ {(B + B′.C) = B+C - Absorption law}
Y = A.(B+C) + A′.B.C + A′.B′.C′ {Using Distributive law}
Y = A.B + A.C + A′.B.C + A′.B′.C′ {Using Associative law}
Y = A.B + A′.B.C + A.C + A′.B′.C′ {A+A′.C = A+C -Absorption law}
Y = B.(A + A′.C) + A.C + A′.B′.C′

Y = B.(A+C) + A.C + A'.B'.C'

1.4 TECHNIQUES TO MINIMIZE THE BOOLEAN FUCTION:

Two types of minimization techniques: 1) Karnaugh-Map Method 2) Quines McCluskey Method

1.4.1 The Karnaugh map method (K-Map)

A Karnaugh map is a graphical representation of the logic system. It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions.

Sum of Product(Minterm): Each minterm is obtained from an ANDterm of the 'n' variables, with each variable being primed if the corresponding bit of the binary numbers is a '0' and unprimed if a '1'. (**m_j**) is the symbol of minterm where subscript 'j' is the decimal equivalent of binary number of minterm designated. Table of minterms as follows:

Decimal number	**Variables**	**Term**	**Designation**
	X Y Z		
0	0 0 0	X'.Y'.)'	m_0
1	0 0 1	X'.Y'.)	m_1
2	0 1 0	X'.Y.)'	m_2
3	0 1 1	X'.Y.)	m_3
4	1 0 0	X.Y'.)'	m_4
5	1 0 1	X.Y'.)	m_5
6	1 1 0	X.Y.)'	m_6
7	1 1 1	X.Y.Z	m_7

Example: Boolean expression in SOP form:

F = X'.Y'.)' + X'.Y'.) + X'.Y.) + X.Y'.) + X.Y.Z

F(A,B,C,D) = ∑m (0, 1, 3, 5, 7)

Product of Sum(Maxterm): Each maxterm is obtained from an O'term of the 'n' variables, with each variable being primed if the corresponding bit of the binary numbers is a '1' and unprimed if a '0'. (M_j) is the symbol of maxterm where subscript 'j' is the decimal equivalent of binary number of maxterm designated. Table of maxterms as follows:

Decimal number	**Variables**	**Term**	**Designation**
	X Y Z		
0	0 0 0	X+Y+Z	M_0
1	0 0 1	X+Y+)'	M_1
2	0 1 0	X+Y'+)	M_2
3	0 1 1	X+Y'+)'	M_3
4	1 0 0	X'+Y+)	M_4
5	1 0 1	X'+Y+)'	M_5
6	1 1 0	X'+Y'+)	M_6
7	1 1 1	X'+Y'+)'	M_7

Example: Boolean expression in POS form:

F = (X'+Y+)') (X'+Y'+Z) (X'+Y+Z) (X+Y'+Z) (X+Y+Z)

F(A,B,C,D) = ∏M (5,6,4,2,0)

K-MAP Representation:

2-Variable K-Map

	B'	B
A'	m0	m1
A	m2	m3

3-Variable K-Map

	B'C'	B'C	BC	BC'
A'	m0	m1	m3	m2
A	m4	m5	m7	m6

4-Variable K-Map

	C'D'	C'D	CD	CD'
A'B'	m0	m1	m3	m2
A'B	m4	m5	m7	m6
AB	m12	m13	m15	m14
AB'	m8	m9	m11	m10

5-Variable K-Map

	C'D'E'	C'D'E	C'DE	C'DE'	CDE'	CDE	CD'E	CD'E'
A'B'	m0	m1	m3	m2	m6	m7	m5	m4
A'B	m8	m9	m11	m10	m14	m15	m13	m12
AB	m24	m25	m27	m26	m30	m31	m29	m28
AB'	m16	m17	m19	m18	m22	m23	m21	m20

Question: Simplify the Boolean function: F = ∑m (1, 3, 5, 9, 11, 13). Using K-Map Method.
Solution: F = ∑m (1, 3, 5, 9, 11, 13)

	C'.D'	C'.D	C.D	C.D'
A'.B'	0	1	1	0
A'.B	0	1	0	0
A.B	0	1	0	0
A.B'	0	1	1	0

The minimized Boolean function is F = C'D + B'D

Question : Using Karnaugh maps, write the minimized Boolean expressions for the output functions : Y1 = A'.B.C' + A.B'.C' + A.B.C + A'.B'.C' and Y2 = A'.B'.C + A.B.C' + A'.B'.C' + A.B'.C + A.B.C + A.B'.C'
Solution: Y1 = ∑m (0, 2, 4, 7) and Y2 = ∑m (0, 1, 4, 5, 6, 7)

K-Map for Y1:

	B'C'	B'C	BC	BC'
A'	1	0	0	1
A	1	0	1	0

K-Map for Y2:

	B'C'	B'C	BC	BC'
A'	1	1	0	0
A	1	1	1	1

The minimized expressions are as follows: Y1 = B'.C' + A'.C' + A.B.C and Y2 = A + B'

Question: Simplify the Boolean function: F = YM (1,4,5,6,11,12,13,14,15) in POS form. Using K-Map Method.

Solution: F = YM (1,4,5,6,11,12,13,14,15)

	C'.D'	**C'.D**	**C.D**	**C.D'**
A'.B'	1	**0**	1	1
A'.B	**0**	**0**	1	**0**
A.B	**0**	**0**	**0**	**0**
A.B'	1	1	**0**	1

The minimized expression in POS form is F = (B'+C) (B'+D) (A+C+D') (A"+C'+D')

1.4.2 Quine–McCluskey or Tabular Method

Tabulation method of simplification consists of two parts: i) Determination of prime implicants(Find all terms include in the simplified function) and ii) Determination of essential prime implicants (choose among the prime implicats, that give an expression with the least number of terms).

Determination of prime implicants:

Simplify the Boolean function by using tabulation method: F = ∑m (0, 1, 2, 8, 10, 11, 14, 15)

Step-01: Group binary representation of the minterms according to the number of 1's contained.

Group	**minterm**	**Variables**				
		w	**x**	**y**	**z**	
0	0	0	0	0	0	√
1	1	0	0	0	1	√
	2	0	0	1	0	√
	8	1	0	0	0	√
2	10	1	0	1	0	√
3	11	1	0	1	1	√
	14	1	1	1	0	√
4	15	1	1	1	1	√

Step-02: Any two minterms which differ from each other by only one variable can be combined, and the unmatched variable removed and a check (√) is placed to the right of both the minterms to show that they have been used . The minterms in one group i.e group 0, are compared with next down group i.e group1 only.

minterm	**Variables**				
	w	**x**	**Y**	**z**	
0,1	0	0	0	-	uncheck
0,2	0	0	-	0	√
0,8	-	0	0	0	√
2,10	-	0	1	0	√
8,10	1	0	-	0	√
10,11	1	0	1	-	√
10,14	1	-	1	0	√

11,15	1	-	1	1	√
14,15	1	1	1	-	√

Step-03: Comparing process is repeated.

minterm	Variables				
	w	x	Y	z	
0,2,8,10	-	0	-	0	uncheck
0,8,2,10	-	0	-	0	uncheck
10,11,14,15	1	-	1	-	uncheck
10,14,11,15	1	-	1	-	uncheck

Step-04: The uncheck terms form the prime implicants(P.I).

F = w’x’y’ + x’z’ + wy

Selection of Essential prime implicants:

P.I	Minterms	0	1	2	8	10	11	14	15
w’x’y’	0,1	x	x						
x’z’	0,2,8,10	x		x	x	x			
wy	10,11,14,15					x	x	x	x
			√	√	√		√	√	√

Prime Implicants that cover minterms with a single cross in their column are called essential prime implicants.

So,Simplified Boolean function is: F = w’x’y’ + x’z’ + wy

Subject Name: **Digital Circuits & Systems** Subject Code: **IT 305**

Subject Notes

UNIT-II

Combinational Circuits: Introduction to Combinational Circuits, Multilevel NAND,NOR Implementation. Designing Binary Adders and Subtractors. Decoder, Encoder, Multiplexer, Demultiplexer circuits.

2.1 COMBINATIONAL CIRCUITS:

INTRODUCTION TO COMBINATIONAL CIRCUITS:

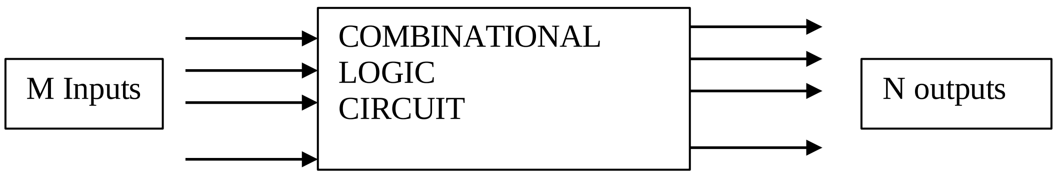


Fig.2.1 Block Diagram of a Combinational Circuit

- A **Combinational Circuit** consists of logic gates whose **outputs at any time** are determined directly from the **present combination of inputs** without regard to previous inputs. It consists of input variables, logic gates and output variables.
- A Combinational circuit does not consists of memory element as it depends only on present input.
- The design procedure of any combinational circuit start from the verbal outline of the problem and ends in a logic circuit diagram, or asset of Boolean functions from which the logic diagram can be easily obtained.
- The design steps are:

a) The problem is stated or Design specifications are stated.
b) The number of input and required output variables is determined.
c) Truth table is derived.Truth Table expresses the complete relation between output and input variables.
d) Simplified Boolean function for each output is obtained.
e) Logic diagram is drawn.

Some examples of combinational circuits are: Half adder, full adder, half subtractor, full subtractor,Encoders,Decoders,Multiplexers,Demultiplexers…etc.

2.2 MULTI LEVEL IMPLEMENTATION NAND,NOR GATES

Multi Level Implementation means that any path from input to output path contains maximum two gates or more hence the name Multilevels for various levels of gates.
Implementing two level logic using NOR gate requires the Boolean expression to be in Product of Sum (POS) Form.
In POS form the first level of gate is an OR gate and the second level is an AND gate.
To Implement Boolean Using NOR Gate, there are basically three steps:
1) At first we need to have a simplified POS expression for the function you need to implement.Simplified POS expression can be made using Karnaugh map by combining the ‘0’ and then inverting the output function .

Suppose we have a simplified POS Expression: F=(A+B).(C+D)

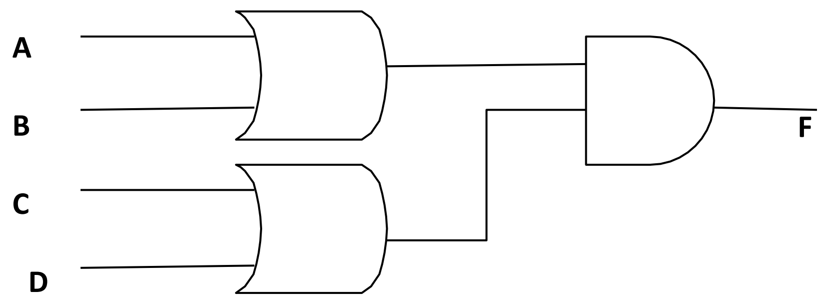


Fig 2.2.1 : Logical diagram of F=(A+B).(C+D)

2) Next Step is to draw above mentioned schematic using OR –INVERT and INVERT-AND gates.OR INVERT should replace OR gates and INVERT AND gates replaces AND gates.

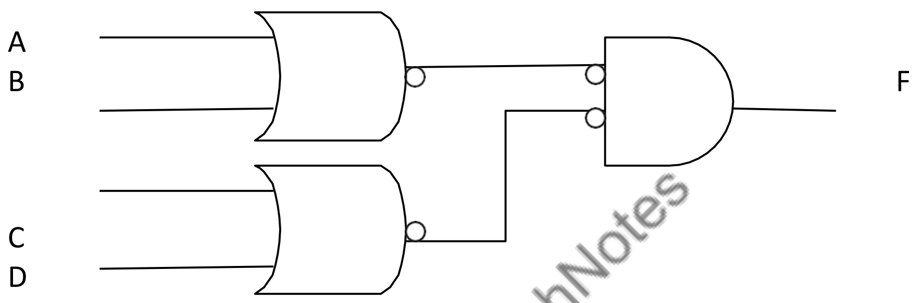


Fig 2.2.2 Converting AOI to NOR

3) The last Step is to redraw the whole Schematic Replacing OR -INVERT and INVERT -AND Gates by NOR Symbol.

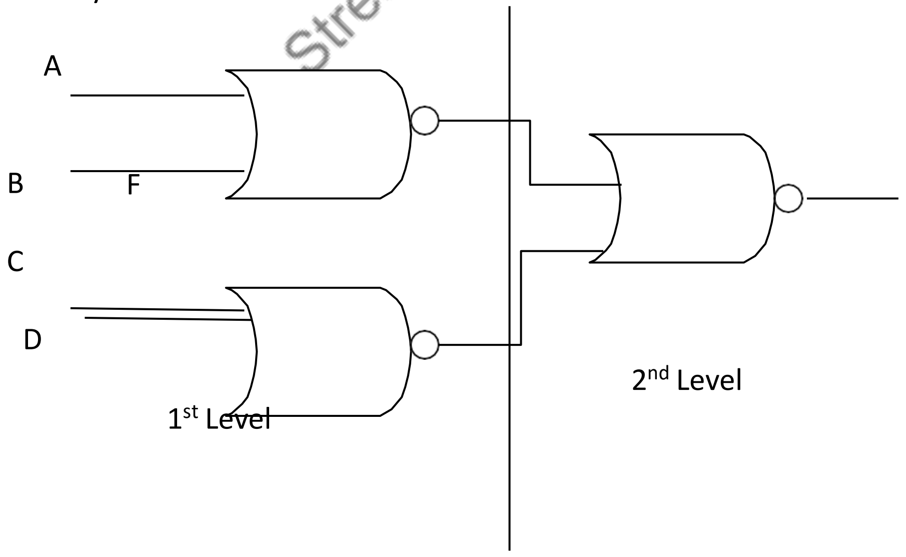


Fig 2.2.3 Multi level NOR Implementation

So, like wise we can draw for many levels of NOR gates .

MULTI LEVEL IMPLEMENTATION USING NAND:

Multi Level Implementation means that any path from input to output path contains maximum two or more gates hence the name Multilevel for various levels of gates.

Implementing Multi level logic using NAND gate requires the Boolean expression to be in Sum

of Product (SOP) Form.
In SOP form the first level of gate is an AND gate and the second level is an OR gate.
To Implement Boolean Using NAND Gate, there are basically three steps:
1) At first we need to have a simplified SOP expression for the function you need to implement. Simplified SOP expression can be made using Karnaugh map by combining the '1' and then inverting the output function .
Suppose we have a simplified SOP Expression: F=AB+CD

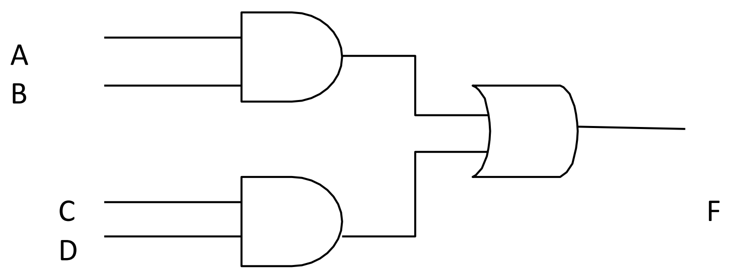


Fig 2.2.4 : Logical diagram of F=AB+CD

2) Next Step is to draw above mentioned schematic using AND –INVERT and INVERT-OR gates. AND -INVERT should replace AND gates and INVERT OR gates replaces OR gates.

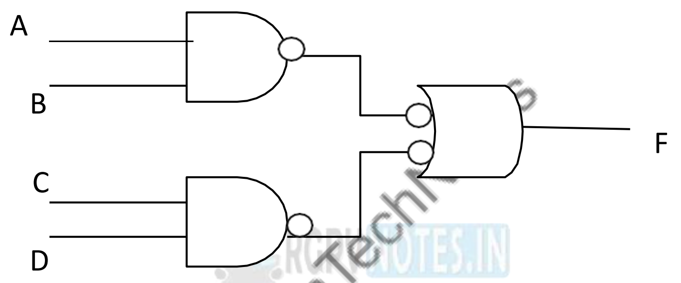


Fig 2.2.5 Converting AOI to NAND

3) The last Step is to redraw the whole Schematic Replacing AND -INVERT and INVERT -OR Gates by NAND Symbol.

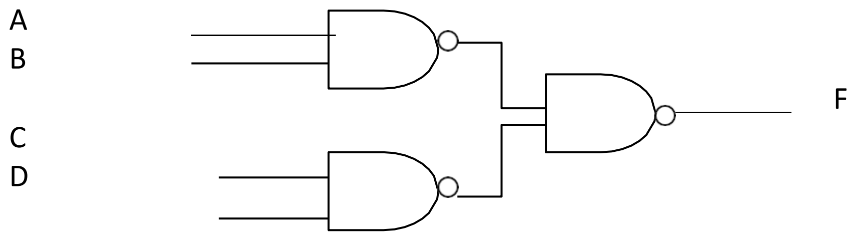


Fig 2.2.6 Multi level NAND Implementation

So, like wise we can draw for many levels of NAND gates .

2.3 DESIGNING BINARY ADDERS AND SUBTRACTORS

2.3.1 HALF ADDER:

A combinational circuit that perform the addition of two bits is called half adder. This circuit needs two binary inputs and two binary outputs. The input variables, augend (X) and addend (Y) bits; the output variables SUM (S) and CARRY (C).

Truth Table:

INPUTS		OUTPUTS	
X	Y	SUM (S)	CARRY(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The simplified output Boolean function : **SUM (S) = X'.Y + X.Y' and CA‘‘Y (C) = X.Y**
The logic diagram of Half Adder :

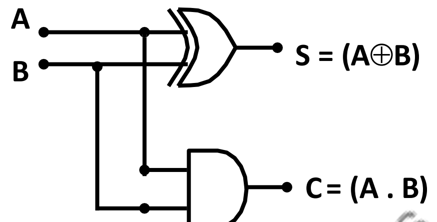


Figure 2.3.1 Half Adder

2.3.2 FULL ADDER:

When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits. The combinational circuit that performs the addition of three bits (two significant bits and a previous carry) is a full adder. It consists of three inputs (X and Y are actual 2-inputs and third input represents the CARRY$_{IN}$ (C$_{IN}$) generated from the previous lower significant bit position) and two outputs, SUM (S) and CARRY$_{OUT}$ (C$_{OUT}$).

Truth Table:

INPUTS			OUTPUTS	
X	Y	C_{IN}	SUM (S)	CARRY(C_{OUT})
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

Boolean expression shown from the truth table which is shown:

SUM = $X'. Y'. C_{IN} + X'. Y. C_{IN}' + X. Y'. C_{IN}' + X. Y. C_{IN}$

SUM = $(X \oplus Y \oplus C_{IN})$

CA‘‘Y = $X'. Y. C_{IN} + X. Y. C_{IN}' + X. Y'. C_{IN} + X. Y. C_{IN}$

$$= Y.C_{IN}(X + X') + X.\,Y.\,C_{IN}' + X.\,Y'.\,C_{IN}$$
$$= Y.C_{IN} + X.\,Y.\,C_{IN}' + X.\,Y'.\,C_{IN}$$
$$= Y(C_{IN} + C_{IN}'.X) + X.\,Y'.\,C_{IN}$$
$$= Y(C_{IN} + X) + X.\,Y'.\,C_{IN}$$
$$= Y.C_{IN} + Y.X + X.\,Y'.\,C_{IN}$$
$$= C_{IN}\,(Y + X.\,Y') + Y.X$$
$$= C_{IN}\,(Y + X) + Y.X$$
$$\text{CARRY} = C_{IN}.Y + C_{IN}.X + Y.X$$

Using K-Map method, simplify the output expression for SUM(S) and CARRY (C_{OUT}):

K-Map for SUM:

	$Y'.C_{IN}'$	$Y'.C_{IN}$	$Y.C_{IN}$	$Y.C_{IN}'$
X'	0	1	0	1
X	1	0	1	0

$$S = X'.Y'.C_{IN} + X'.Y.C_{IN}' + X.Y'.C_{IN}' + X.Y.C_{IN}$$

K-Map for CARRY:

	$Y'.C_{IN}'$	$Y'.C_{IN}$	$Y.C_{IN}$	$Y.C_{IN}'$
X'	0	0	1	0
X	0	1	1	1

$$C_{OUT} = X.Y + Y.C_{IN} + X.C_{IN}$$

Logic Diagram of Full Adder using 2-half adder and OR gate:

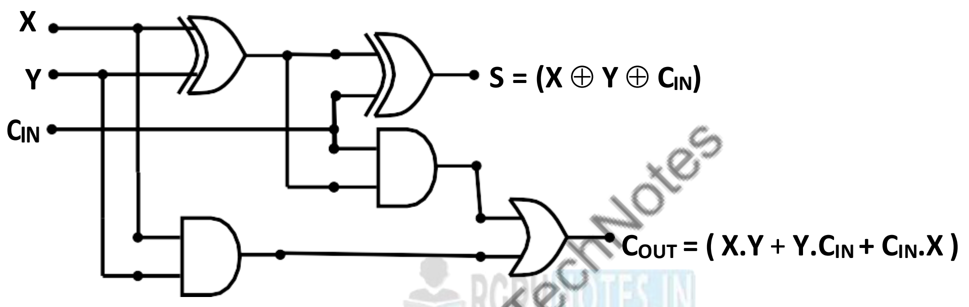


Figure 2.3.2 Full Adder

2.3.3 HALF SUBTRACTOR:

The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow). The logic symbol and truth table are shown below.

TRUTH TABLE FOR HALF SUBTRACTOR:

INPUT		OUTPUT	
A	B	DIFFERENCE(D)	BORROW (BOR_{OUT})
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

	B'	B
A'	0	1
A	1	0

D = A’.B + A.B’

	B'	B
A'	0	1
A	0	0

BOR_{OUT} = A’.B

LOGIC DIAGRAM OF HALF SUBTRACTOR:

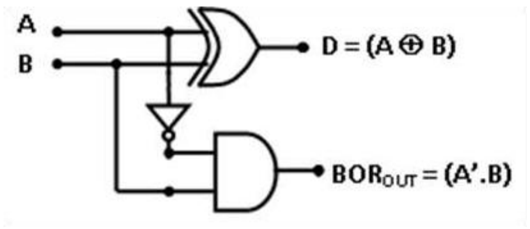


Figure 2.3.3 Half Subtractor

2.3.4 FULL SUBTRACTOR

The half-subtractor can be used for LSB(Least Significant Bit) subtraction. If there is a borrow during the subtraction of the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that coloumn used for the subtraction in the preceding column. Such a subtraction is performed by a full-subtractor.

The Full-subtractor is a combinational circuit which is used to performs subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. It has three inputs, A(minuend) and B (subtrahend) and BORROW IN (BOR_{IN}) and two outputs D (difference) and BOR_{OUT} (borrow out).

TRUTH TABLE FOR FULL SUBTRACTOR:

INPUT			OUTPUT	
A	B	BOR_{IN}	DIFFERENCE(D)	BORROW (BOR_{OUT})
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

	B'.BOR$_{IN}$'	B'.BOR$_{IN}$	B.BOR$_{IN}$	B.BOR$_{IN}$'
A'	0	1	0	1
A	1	0	1	0

$D = A'.B'.BO'_{IN} + A'.B.BO'_{IN}' + A.B'.BO'_{IN}' + A.B.BO'_{IN}$

	B'.BOR$_{IN}$'	B'.BOR$_{IN}$	B.BOR$_{IN}$	B.BOR$_{IN}$'
A'	0	1	1	1
A	0	0	1	0

$BOR_{OUT} = A'.B + B.BO'_{IN} + A'.BO'_{IN}$

LOGIC DIAGRAM OF FULL SUBTRACTOR:

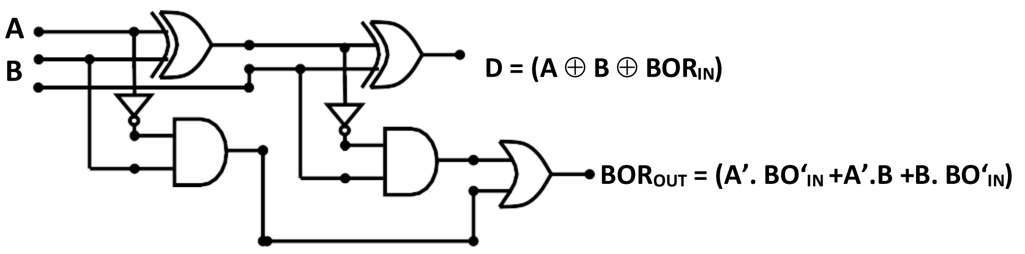


Figure 2.3.4 Full Subtractor

2.3.5 LOOK AHEAD CARRY GENERATOR:

Consider the full adder circuit,

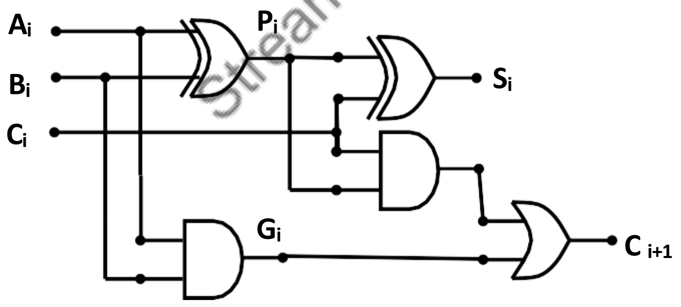


Figure 2.3.5 Full Adder

Where, G_i is carry generate and produces the carry when Ai and Bi are 1, regardless of input carry. P_i is carry propagate, term associated with propagation of carry from C_i to C_{i+1}.
From above circuit, we define two new binary variables:

$$P_i = A_i \oplus B_i \quad \text{and} \quad G_i = A_i . B_i$$

Output sum and carry is expressed as:

$$S_i = P_i \oplus C_i \quad \text{and} \quad C_{i+1} = G_i + P_i . C_i$$

Now writing the boolean function for the carry output of each stage and substituting for each C_i its value from the previous equations , we get:

$C_1 = G_0 + P_0 . C_0$

$C_2 = G_1 + P_1 . C_1 \quad = \quad G_1 + P_1 . (G_0 + P_0 . C_0) \quad = \quad G_1 + P_1 .G_0 + P_1. P_0 . C_0$

$C_3 = G_2 + P_2 . C_2 \quad = \quad G_2 + P_2 . (G_1 + P_1 .G_0 + P_1. P_0 . C_0) \quad = \quad G_2 + P_2 . G_1 + P_2. P_1 .G_0 + P_2. P_1. P_0 . C_0$

From the above equation it is noted that C_3 does not have to wait for C_2 and C_1 to propagate; in fact C_3 is propagated at the same time as C_2 and C_1. Hence the carry's are propagated on the same time so no carry propagation delay occurs. Logic diagram of look ahead carry generator is shown below.

Logic diagram of Look Ahead Carry Generator:

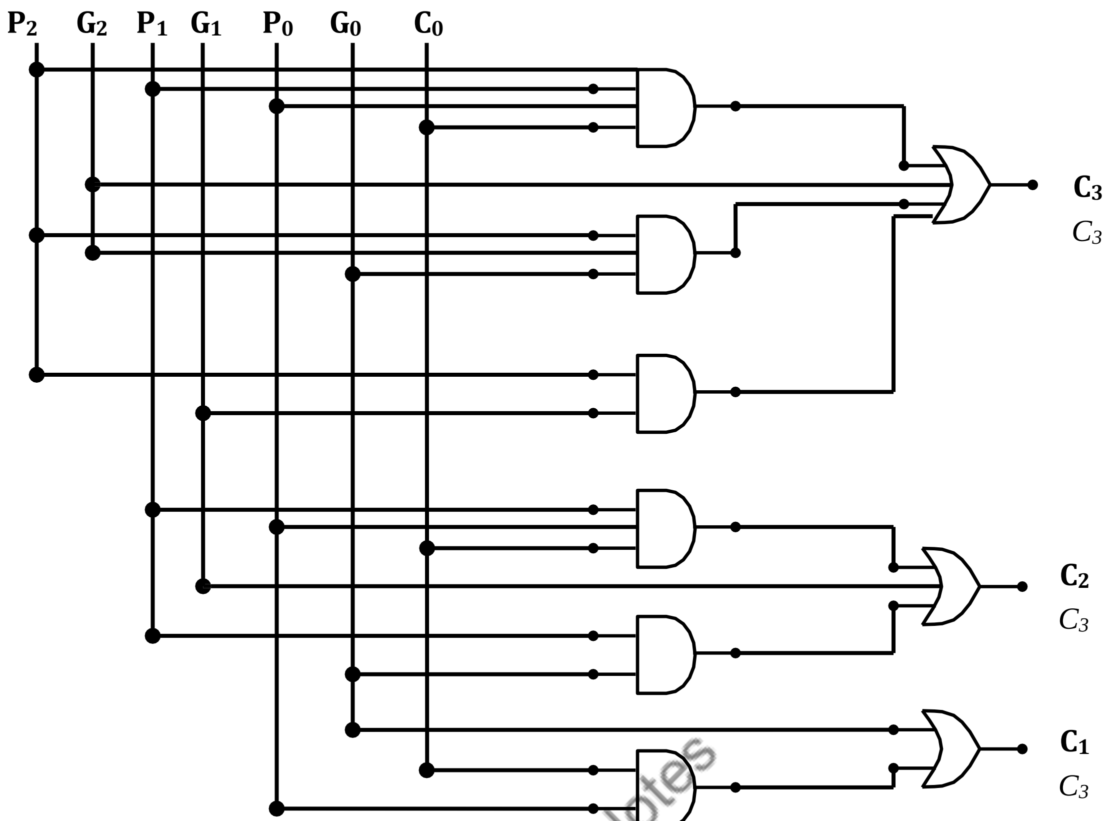


Figure 2.3.5 Look Ahead Carry Generator

2.3.6 SERIES AND PARALLEL ADDER (4-Bit Binary Parallel Adder) :

A binary parallel adder produces a sum of 2-binary numbers in parallel. It consists of full adders connected in cascade, with the output carry from one full adder connected to the input carry of the next full adder. An 4-bit parallel adder requires 4-full adders. So, to design an n-bit binary parallel adder, it requires n full adders. Below figure 01 shows the interconnection of 4-full adder circuits to provide a 4-bit binary parallel adder.

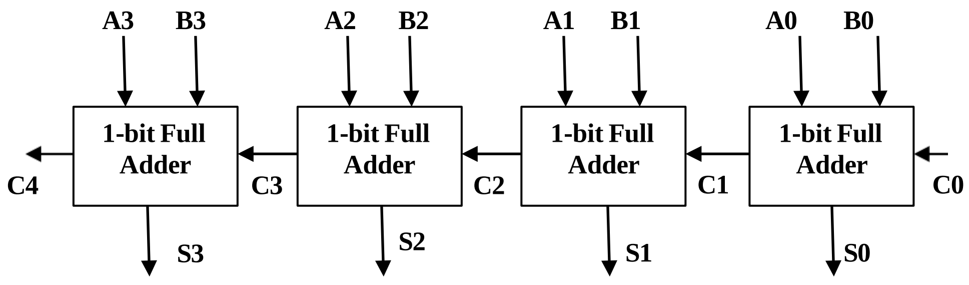


Figure 2.3.6 4-Bit Binary Parallel Adder

From figure 2.3.6, the augend bits of A (A_0,A_1,A_2,A_3) and the addend bits of B (B_0,B_1,B_2,B_3) are designated by subscript numbers from right to left, with subscript 1 denoting the lower order bit. The carries (C_0,C_1,C_2,C_3) are connected in chain through full adders. The input carry to the full adder is C_0 and output carry is C_4. The S (S_0,S_1,S_2,S_3) outputs generates the required sum bits.

Example: Consider **A = 1011** and **B = 0011**

Input carry	0 **(C_3)**	1 **(C_2)**	1 **(C_1)**	0 **(C_0)**	**C_i**
Augend	1 **(A_3)**	0 **(A_2)**	1 **(A_1)**	1 **(A_0)**	**A_i**
Addend	0 **(B_3)**	0 **(B_2)**	1 **(B_1)**	1 **(B_0)**	**B_i**
Sum	1 **(S_3)**	1 **(S_2)**	1 **(S_1)**	0 **(S_0)**	**S_i**
Output carry	0 **(C_4)**	0 **(C_3)**	1 **(C_2)**	1 **(C_1)**	**C_{i+1}**

Subscrip "i" represent i = 0,1,2,3.
In parallel adder, input carry C_0 in the least significant position must be 0.

Carry Propagation In Binary Parallel Adder: Consider the figure 2.3.6, a 4-bit binary parallel adder. The inputs A_3 and B_3 reach a steady state value as soon as inputs signals are applied to the adder. But input carry C_3 does not settle to is final steady state value until C_2 is in its steady state value. Simillarly, C_2 has to wait for C_1. Thus only after the carry propagates through all the stages will the last output S_3 and C_4 settle to its final steady state value. Thus to get the corrected output, carry has to propagate on time, if not the outputs will not be correct. So, the carry propagation time is the limiting factor.
The solution for reducing the carry propagation delay time is 1) to employ the faster gates with reduced delays. But physical circuits have a limit. 2) to increase the equipment complexity in such a way that the carry delay time is reduced. 3) the most widely used technique employs the principle of look ahead carry generator.

2.4 ENCODER:

A combinational logic circuit that produces the reverse operation of a decoder. Encoder has 2^n(or less) input lines and 'n' output lines. The output lines generate the binary code for 2^n input lines.

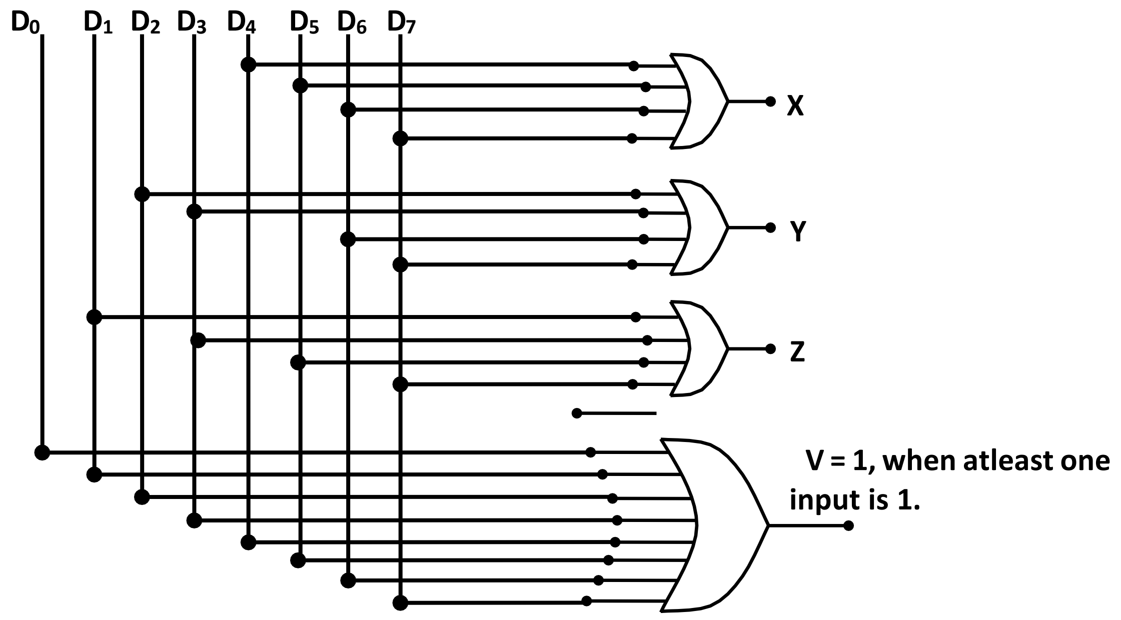
8 to 3 Line Encoder (Octal to Binary Encoder):
Octal to binary encoder consists of eight inputs, one for each of the eight digits, and 3-outputs that generate the corresponding binary number.

Truth table-

Inputs								**Outputs**		
D_0	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**	**X**	**Y**	**Z**
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Output "X"is 1 for octal digits: 4,5,6,7. So, **$X = D_4 + D_5 + D_6 + D_7$**
Output "Y"is 1 for octal digits: 2,3,6,7. So, **$Y = D_2 + D_3 + D_6 + D_7$**

Output “)”is 1 for octal digits: 1,3,5,7. So, **Z = D_1 +D_5 + D_3 + D_7**



D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7
X
Y
Z
V = 1, when atleast one input is 1.

Figure 2.4.1:Logic diagram of 8 to 3 line encoder

The encoder in figure2.4.1, assumes that only one input line can be equal to 1 at any time. Note that, circuit has 8-inputs i.e 2^8 = 256 possible input combinations and only 8 of these combinations have any meaning. The other input combinations are don’t care condition. The output V in figure is to indicate the fact that all inputs are not 0’s, as shown in figure.
Similarly we can design priority encoder, decimal to BCD encoder, hexadecimal to binary encoder.

PRIORITY ENCODER:
Priority encoder establish an input priority to ensure that only the highest priority input line is encoded.Thus, if priority is given to an input with a highest subscript number over one with a lower subscript number, then the encoded output will be of highest subscript number.
Truth Table of priority encoder:

Inputs				**Outputs**	
D_3	**D_2**	**D_1**	**D_0**	**A**	**B**
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

“ X” indicates don’t care condition. i.e. “X” can be 1 or 0.
From the truth table, input D_3 is with the highest priority than inputs D_2, D_1 and Do. If D_3 = 1, and D_2, D_1 and Do are don’t care , then output will be (11).
Simillarly, if D_3 = 0, D_2 = 1 and D_1 and Do are don’t care , then output will be (10).
If D_3 = 0, D_2 = 0, D_1 = 1 and Do is don’t care , then output will be (01).
If D_3 = 0, D_2 = 0, D_1 = 0 and Do = 1 , then output will be (00).
For example: Consider the truth table of 8 to 3 Line Encoder (Octal to Binary Encoder). Suppose, input D_5 has highest priority than D_2, then if both D_2 and D_5 are logic-1 simultaneously, the output will be 101 because D_5 has highest priorty over D_2 .

Design a 4 line to 2 line priority encoder. Include an output E to indicate that atleast one input is a 1.

Solution: Priority given to the input with highest subscript number i.e. input D_3

Truth table of 4 line to 2 line priority encoder:

Inputs				Outputs		
D_3	D_2	D_1	D_0	A	B	E
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Simplification of output expression using K-map, we get:

$A = D_3 + D_2$; $B = D_3 + D_2'.D_1$; $E = D_3 + D_2 + D_1 + D_0$

Logic Diagram of 4 line to 2 line priority encoder:

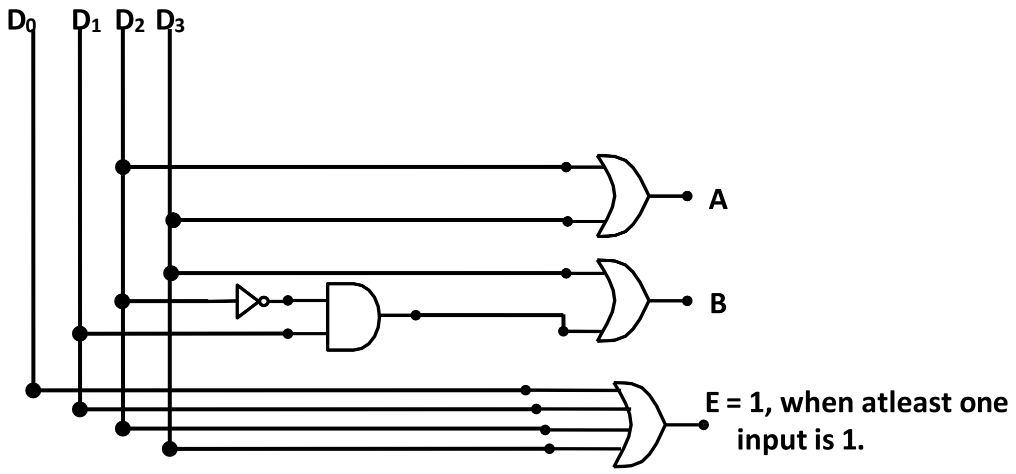


Figure 2.4.2: Logic diagram of 4 to 2 line priority encoder

2.5 DECODER:

A decoder is a combinational logic circuit that converts binary information from n-input lines to a maximum of 2^n unique output lines. If the n-bit decided information has unused or don't care combinations, the decoder output will have lass than 2^n outputs.

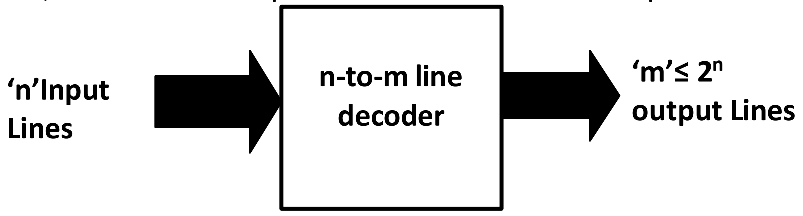


Figure 2.5.1 Block diagram of Decoder

3 to 8 Line Decoder:

3-inputs are decoded into eight output, each output representing one of the minterms of the 3-input variables.

Inputs			Outputs							
X	**Y**	**Z**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_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

Truth Table of 3 to 8 line decoder

The output line whose value is equal to 1 represents the minterm equivalent of the binary number available in the input lines.

Output expressions: $D_0 = X'. Y'.)'$; $D_1 = X'. Y'.)$; $D_2 = X'. Y.)'$; $D_3 = X'. Y.)$
$D_4 = X. Y'.)'$; $D_5 = X. Y'.)$; $D_6 = X. Y.)'$; $D_7 = X. Y. Z$

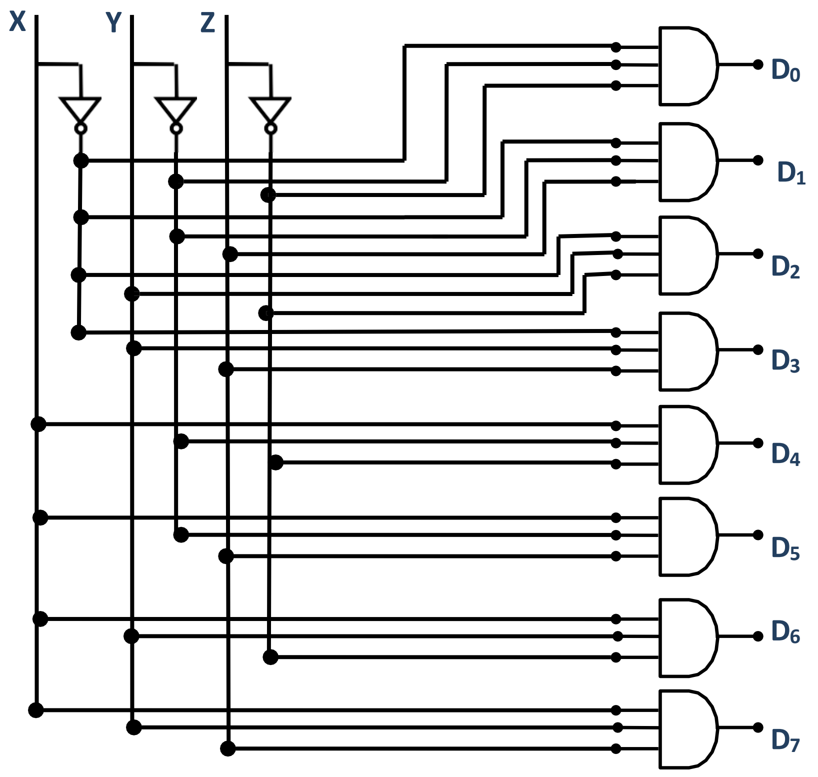


Figure 2.5.1: Logic diagram of 3 to 8 Line Decoder

Simillarly, 2 to 4 line decoder and 4 to 16 line decoder can be designed.

2.6 MULTIPLEXERS (MUX):

Multiplexing means transforming a large number of information over a smaller number of channels or lines. A multiplexer is a combinational logic circuit that selects binary information from one of the input lines and directs it to a single output line. That's why multiplexer is also called data selector. Selection of a particular line is controlled by a set of selection input lines. A Multiplexer has 2^n input lines and "n" selection lines, whose bit combination determine which input is selected.

Block Diagram of Multiplexer:

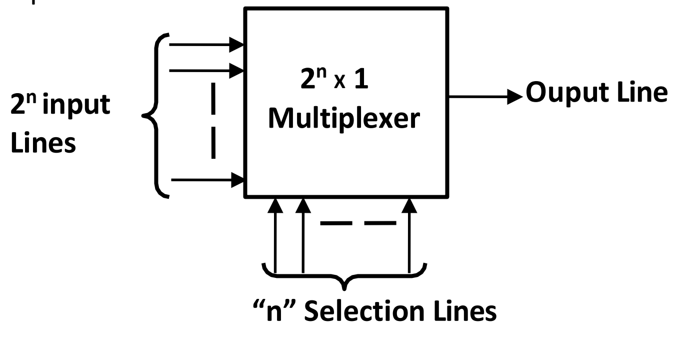


Figure 2.6.1 Multiplexer

4 x 1 Multiplexer:
A 4x 1 Multiplexer has 4-input lines ,1-output line and 2-selection lines.

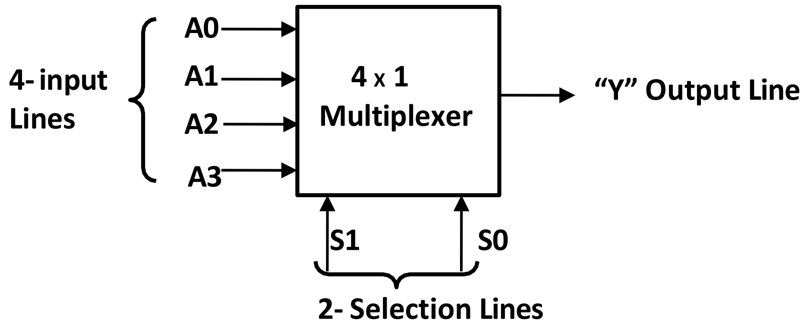


Figure 2.6.2 Block diagram of 4x1 Multiplexer

From the block diagram of 4x1 multiplexer, there are 4- input lines (A0,A1,A2,A3), 2-selection lines (S0 and S1) and 1-output line (Y).

Truth Table:

Input Selection Lines		Output Line
S1	**S0**	**Y**
0	0	A0
0	1	A1
1	0	A2
1	1	A3

According the truth table,

1. When selection line S0=S1=0; input line A0 is connected with output Y.
2. When selection line S0=1 and S1=0; input line A1 is connected with output Y.
3. When selection line S0= 0 and S1=1; input line A2 is connected with output Y.
4. When selection line S0=S1=1; input line A3 is connected with output Y.

Output expression: Y = A0.S0'.S1' + A1.S0.S1' + A2.S0'.S1 + A3.S0.S1

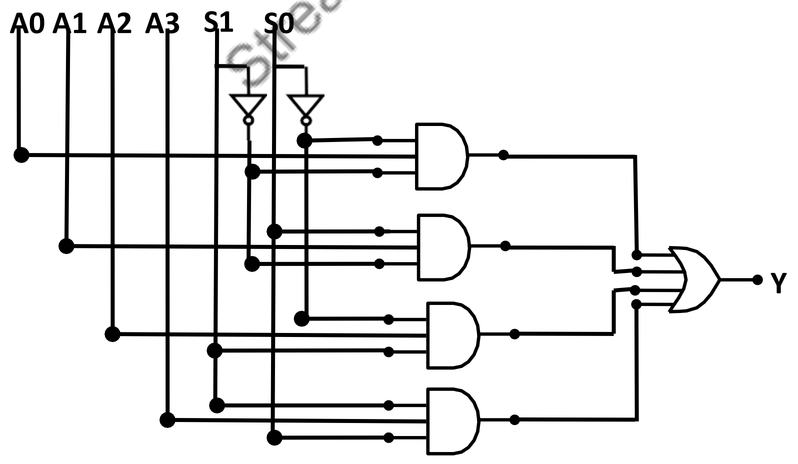


Figure 2.6.3 Logic diagram of 4x1 MUX

Simillarly, Multiplexer 2x1, 8x1, 16x1, 32x1 and so on can be designed.

2.7 DEMULTIPLEXER CIRCUITS (DEMUX):

A demultiplexer is a combinational logic circuit that receives information on single input line and transmits this information on one of 2^n possible output lines. The selection of a specific output line is controlled by the bit values of "n" selection lines.

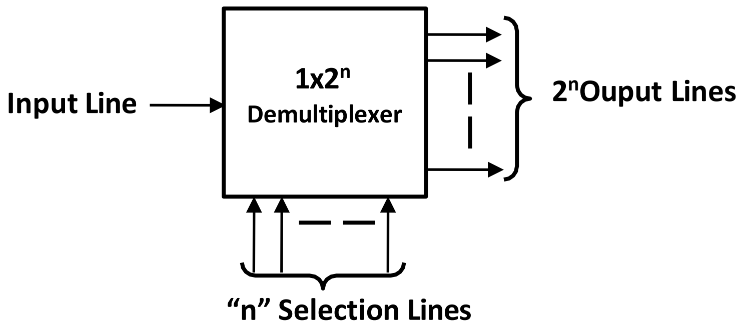


Figure 2.7.1 Block Diagram of Demultiplexer

1 x 4 Demultiplexer:

A 1 x 4 Demultiplexer has 1-input line ,4-output lines and 2-selection lines.

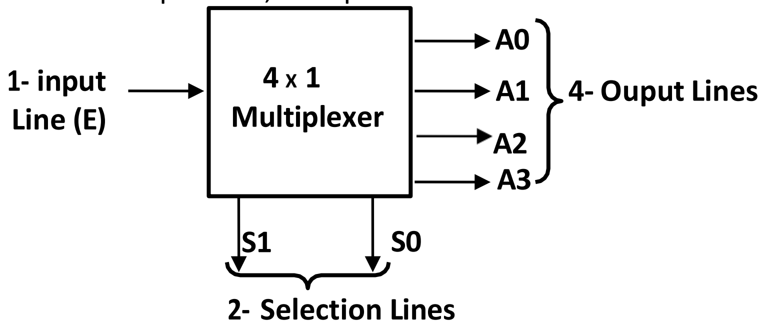


Figure 2.7.2 Block diagram of 1 x 4 Demultiplexer

From the block diagram of 1 x 4 Demultiplexer, there is 1- input line (E) ,2-selection lines (S0 and S1) and 4-output line (A0,A1,A2,A3).

Truth Table:

Input	Input Selection Lines		Output Line
	S1	S0	
E	0	0	A0 = E
E	0	1	A1 = E
E	1	0	A2 = E
E	1	1	A3 = E

According the truth table,

1. When selection line S0=S1=0; input line E is connected with output A0.
2. When selection line S0=1 and S1=0; input line E is connected with output A1.
3. When selection line S0= 0 and S1=1; input line E is connected with output A2.
4. When selection line S0=S1=1; input line E is connected with output A3.

Output expression: A0 =E.S0'.S1'; A1=E.S0.S1' ; A2=E.S0'.S1 ; A3=E.S0.S1

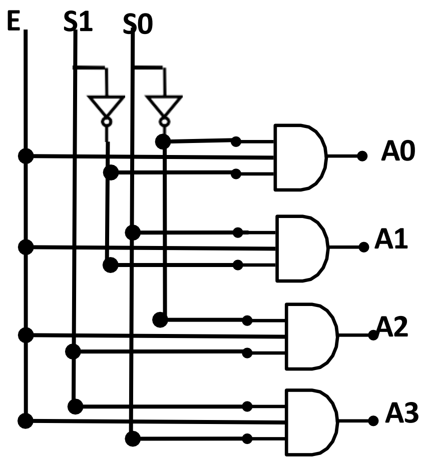


Figure 2.7.3 Logic diagram of 1 x 4 DEMUX

Subject Name: Digital Circuits & Systems **Subject Code**: IT 305

Subject Notes

UNIT-III

Sequential circuits: Introduction to Sequential circuits, flip-flops, RS, D, T, JK, M/S JK-flipflops, truth tables, excitation tables and characteristic equations, clocked and edge triggeredflipflops, Registers- Definition, serial, parallel, shift left/right registers, Johnson counter, asynchronous and synchronous counters.

3.1 INTRODUCTION TO SEQUENTIAL CIRCUITS:

Sequential logic circuits are those, whose output depends not only on the present value of the input but also on previous values of the input signal (history of values) which is in contrast to combinational circuits where output depends only on the present values of the input, at any instant of time. Sequential circuit can be considered as combinational circuit with feedback circuit. Sequential circuit uses a memory element like flip – flops as feedback circuit in order to store past values. The block diagram of a sequential logic is shown below.

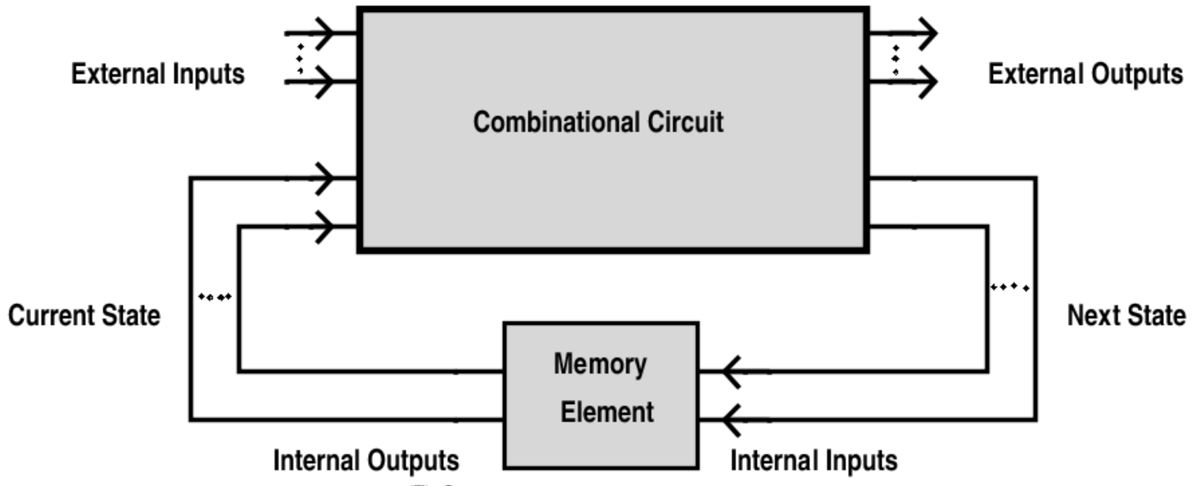


Fig 3.1 Block Diagram of Sequential Logic

3.2 FLIP-FLOPS:

A flip flop is a basic data storage element. A NAND or NOR gate individually act as a storage element when they are cross coupled with feedback. Such cross coupled NAND or NOR gates with feedback are known as flip flops. A flip flop is a bistable (output will remain permanently either 0 or 1 until it is forced to change the state by an external trigger) circuit. A flip flop have two outputs Q and Q', and are complement to each other.

(a) R-S (RESET-SET)FLIP FLOP USING NOR GATES:

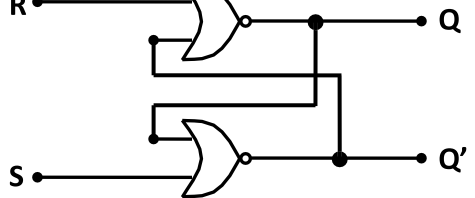


Figure 3.2.1 RS Flip Flop using NOR Gate

Truth Table:

S	R	Q (output)
0	0	No Change
1	0	1 (SET)
0	1	0 (RESET)
1	1	Invalid

The truth table shown for NOR gate flip flop is similar to that of transistor flip flop.

(b) R-S (RESET-SET) FLIP FLOP USING NAND GATES:

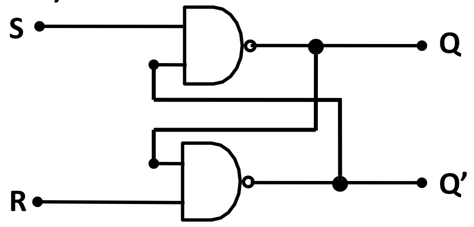


Figure 3.2.2 RS Flip Flop using NAND Gate

Truth Table:

S	R	Q (output)
0	0	Invalid
1	0	0 (RESET)
0	1	1 (SET)
1	1	No Change

The truth table shown for NAND gate flip flop is inverted to that of NOR gate flip flop, hence inverters gates are used to drive the inputs to the gates as shown:

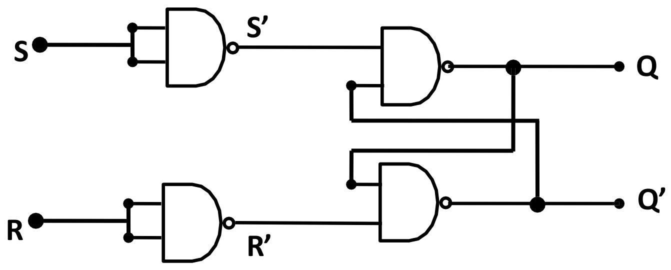


Figure 3.2.3 RS Flip Flop

Truth Table:

S	R	Q (output)
0	0	No Change
1	0	1 (SET)
0	1	0 (RESET)
1	1	Invalid

The truth table for NAND gate flip flop with inverters is similar to that of transistor flip flop, hence this flip flop is used to realize the desired flip flop.

(c) CLOCKED R-S FLIP FLOP:

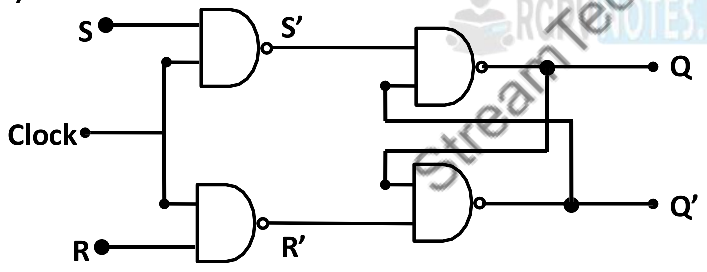


Figure 3.2.4Clocked RS Flip Flop

Truth Table:

Clock	S	R	Q (output)
1	0	0	No Change
1	1	0	1 (SET)
1	0	1	0 (RESET)
1	1	1	Invalid

The clock signal or the enabling signal which makes the circuit to perform the required operation. If clock = 0, the circuit output will remain unchanged. If clock = 1, the flip flop is enabled and respond to the applied input signal.

(d) J-K FLIP FLOP:

In order to overcome the invalid condition in R-S Flip Flop, the J-K Flip Flop is used.

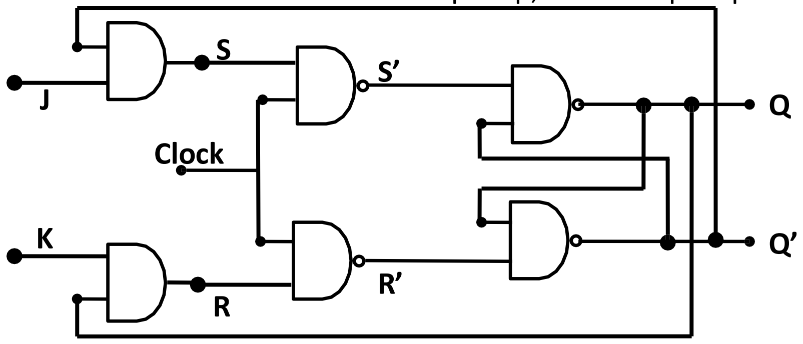


Figure 3.2.5JK Flip Flop

Truth Table of J-K Flip Flop:

CLK	J	K	Qn	S	R	Q_{n+1} (output)	Remarks
1	0	0	0	0	0	0	Qn (No change)
1	0	0	1	0	0	1	
1	0	1	0	0	0	0	0 (RESET)
1	0	1	1	0	1	0	
1	1	0	0	1	0	1	1 (SET)
1	1	0	1	0	0	1	
1	1	1	0	1	0	1	Toggle or complement
1	1	1	1	0	1	0	

Qn represent the past state; Qn+1 represent the present state i.e. the state of the output after the clock pulse is applied.

The J-input is analogous to the S input and K to the R input. So, when J=1 and K=0, the J-K flip flop is in SET state and when, J = 0 and K = 1, the flip flop is in RESET state. When J=K=1, the flip flop will complement its output condition, with high clock signal. This is the RACE around condition and it is aproblem in JK flip flop. To overcome the problem of race around condition, Master-Slave JK flip flop is used.

(e) Master-Slave JK Flip-flop

The master-slave flip-flop eliminates all the timing problems by using two SR flip-flops connected together in a series configuration. One flip-flop acts as the "Master" circuit, which triggers on the leading edge of the clock pulse while the other acts as the "Slave" circuit, which triggers on the falling edge of the clock pulse. This results in the two sections, the master section and the slave section being enabled during opposite half-cycles of the clock signal.

The TTL 74LS73 is a Dual JK flip-flop IC, which contains two individual JK type bistable's within a single chip enabling single or master-slave toggle flip-flops to be made. Other JK flip flop IC's include the 74LS107 Dual JK flip-flop with clear, the 74LS109 Dual positive-edge triggered JK flip flop and the 74LS112 Dual negative-edge triggered flip-flop with both preset and clear inputs.

Dual JK Flip-flop 74LS73

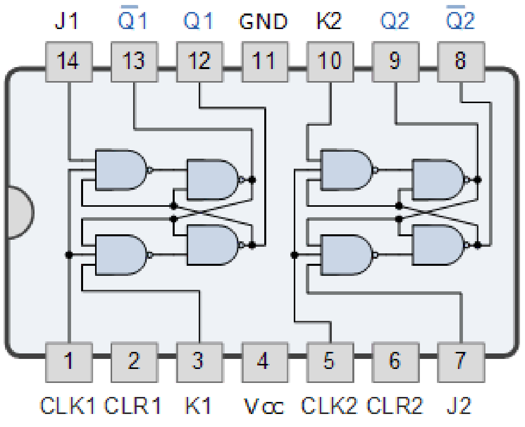


Figure 3.2.6 DualJK Flip Flop

The Master-Slave JK Flip-flop

The **Master-Slave Flip-Flop** is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse. The outputs from Q and Q from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip flop as shown below.

The Master-Slave JK Flip Flop

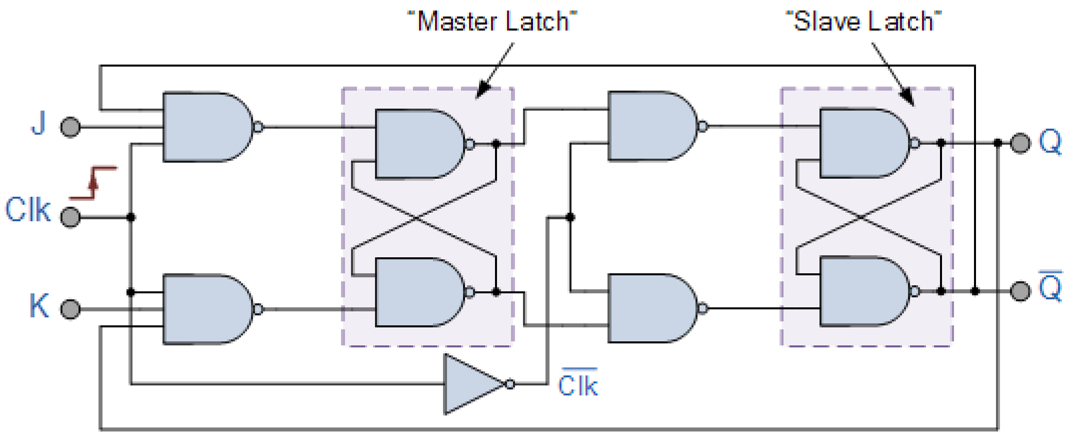


Figure 3.2.7 Dual JK Flip Flop

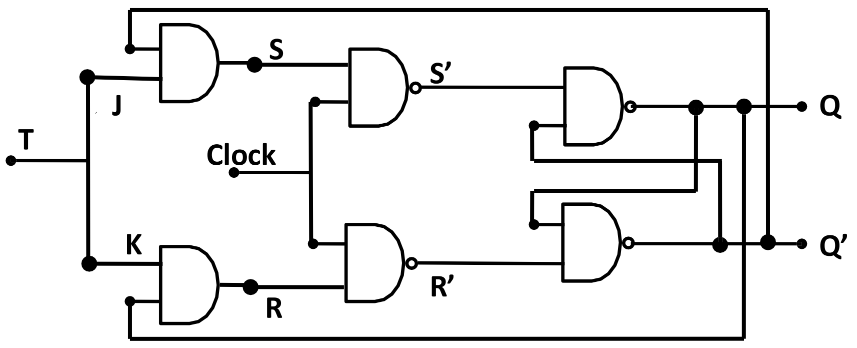
The input signals J and K are connected to the gated "master" SR flip flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1". As the clock input of the "slave" flip flop is the inverse (complement) of the "master" clock input, the "slave" SR flip flop does not toggle. The outputs from the "master" flip flop are only "seen" by the gated "slave" flip flop when the clock input goes "LOW" to logic level "0".

When the clock is "LOW", the outputs from the "master" flip flop are latched and any additional changes to its inputs are ignored. The gated "slave" flip flop now responds to the state of its inputs passed over by the "master" section.

Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip flop are fed through to the gated inputs of the "slave" flip flop and on the "High-to-Low" transition the same inputs are reflected on the output of the "slave" making this type of flip flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the **Master-Slave JK Flip flop** is a "Synchronous" device as it only passes data with the timing of the clock signal.

(f) T- Flip Flop (Toggle or change state):
T- flip flop is a single input version of the JK flip flop. T flip flop is obtained if both the inputs of JK flip flop are tied together.



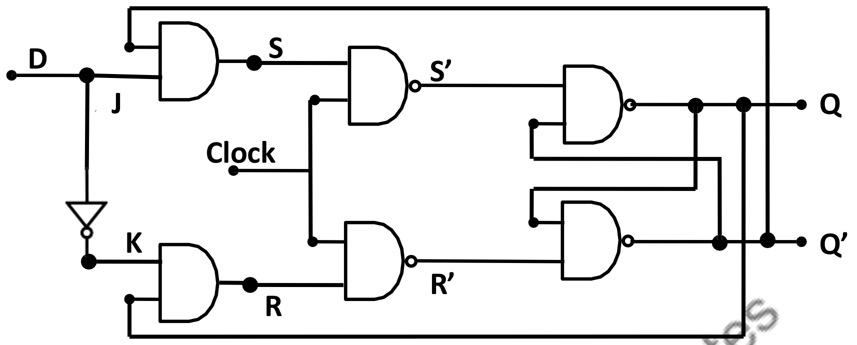
Truth Table:

Clock	T	Qn+1 (output)
1	0	Qn
1	1	Qn'

Figure 3.2.8JK Flip Flop

(g) D-Flip Flop:

The SR or JK flip flop can be converted into a delay (D) flip flop. The D flip flop receives the designation from its ability to transfer data into a flip flop i.e when clock is high the output Q follows the state of the input line D.



Truth Table:

Clock	D	Qn+1 (output)
1	0	0
1	1	1

Figure 3.2.9D Flip Flop

3.3 EDGE & LEVEL TRIGGERED CIRCUITS:

Triggering is very important in the sequential circuits. The circuits operates with the clock waveforms. The circuit can be made to work on either level or edge triggering.

Level Triggered Circuits:

We know that the clock pulse is having the two levels. Therefore if the output is changing during the positive or negative levels of the clock pulse, then the triggering is called the Level Triggered Circuits. In level triggering the circuit gets activated when the clock waveform reaches certain level i.e. either 1 or 0. Clock waveform for level triggering is shown in figure 3.3.1 (a).

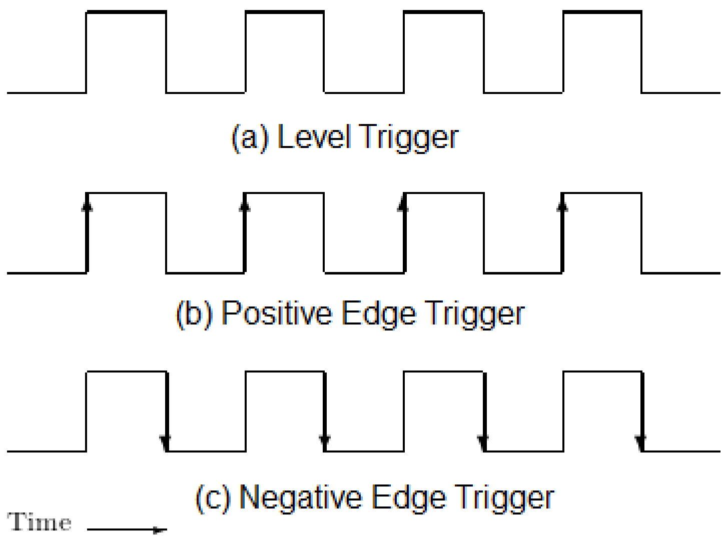


Figure 3.3.1 Level and Edge Triggering

Edge Triggered Circuits:
If the output of any sequential circuits is changing during the transition period of the clock waveform then the triggering is called the Edge Triggering. This can be seen in figure 3.3.1 (b) and (c).
Edge triggered circuits are those circuits which reads the data available at the input pins i.e. gets activated only on the edge of the clock cycle. The edge may be positive or negative depending on that the circuits are called Positive Edge Triggered Circuits and Negative Edge Triggered Circuits.
In positive edge triggered circuts the circuit gets activated when the clock pulse is moving from level 0 to level 1, whereas in negative edge triggered circuits the circuit gets activated when the clock amplitude is moving towards 0 from 1.

3.4 SHIFT REGISTERS:
The register capable of shifting itsbinary information either to the right or to the left is called shift register. Shift register consists of array of flio flops connected in cascade. All flip flops receive a common clock pulse which causes the shift from one stage to the next stage.
Shift registers are classified depending upon the way data are loaded and retrieved.
Classified as- 1.Serial Input Serial Output Shift Register 2.Serial Input Parallel Output Shift Register 3.Parallel Input Serial Output Shift Register 4.Parallel Input Parallel Output Shift Register.

1. Serial Input Serial Output Shift Register (SISO):

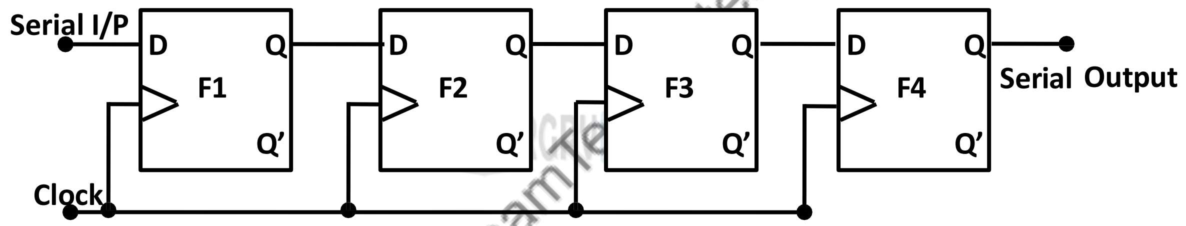


Figure 3.4.1Serial Input Serial Output Shift Register

From the above diagram of SISO shift register, output of eah flip flop is connected to the input of the next flip flop at its right. Each clock pulse shifts the contents of the register one bit to the right.

Operation:
Serial loading of Information or data:

Input Sequence	F1	F2	F3	F4	Clock Pulse
1 1 0 1	0	0	0	0	Initial state
	1	0	0	0	After 1 CP
	0	1	0	0	After 2 CP
	1	0	1	0	After 3 CP
	1	1	0	1	After 4 CP
Serial Loading of data					

Serial retriving of Information or data:

Clock Pulse	F1	F2	F3	F4	Output Sequence
After 4 CP	1	1	0	1	-----
After 5 CP	X	1	1	0	1
After 6 CP	X	X	1	1	01
After 7 CP	X	X	X	1	101
After 8 CP	X	X	X	X	1101
Serial retriving of data					

2. Serial Input Parallel Output Shift Register (SIPO):
In SIPO, data is loaded serially, one bit at a time but data can be read simultaneously.Clock pulse stop at the movement the binary information is stored. Each output is available on a separate line, and they may be read simultaneously.

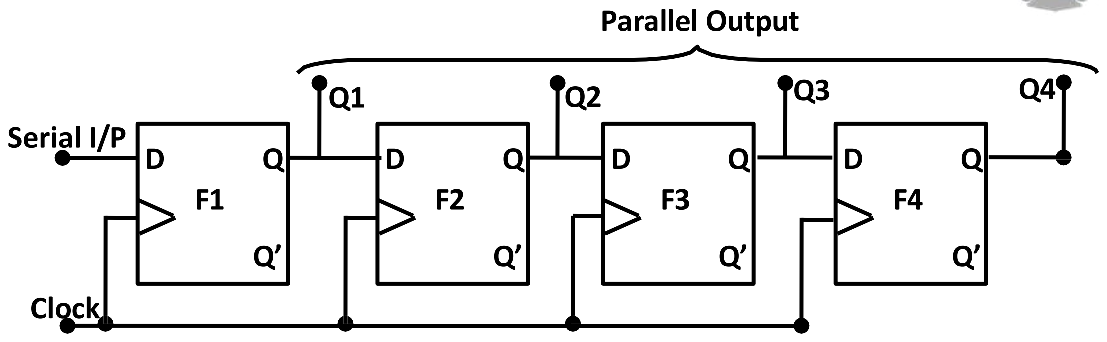


Figure 3.4.2 Serial Input Parallel Output Shift Register

3. Parallel Input Parallel Output Shift Register (PIPO):

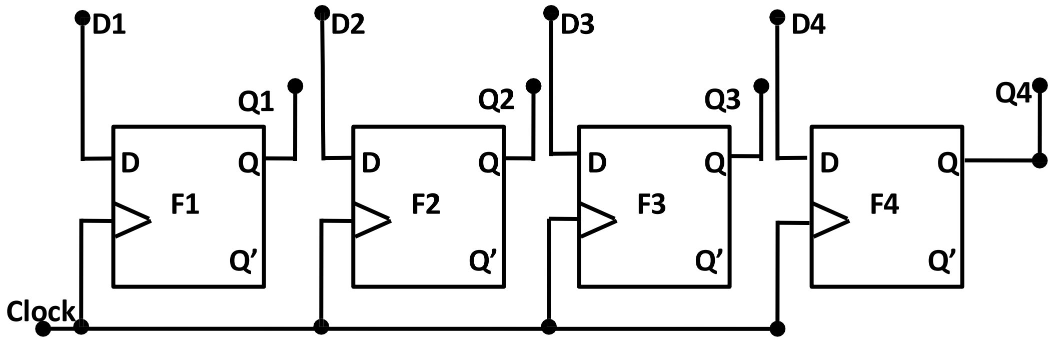


Figure 3.4.3 Parallel Input Parallel Output Shift Register

In PIPO shift register, input data D1,D2,D3 and D4 can be loaded into the flip flop simultaneously through enable pulse and can be retrived simultaneously also from outputs Q1,Q2,Q3 and Q4.

4. Parallel Input Serial Output Shift Register (PISO):

In PISO shift register, all flip flops are preset and the input binary information is written into the register, all bits in parallel, by the preset enable pulse. The stored information may be read serially by applying clock pulses. Since the data can be loaded into the flip flop simultaneously and can be read from the register one bit at a time by clock pulse, this shift register is a parallel to serial converter.

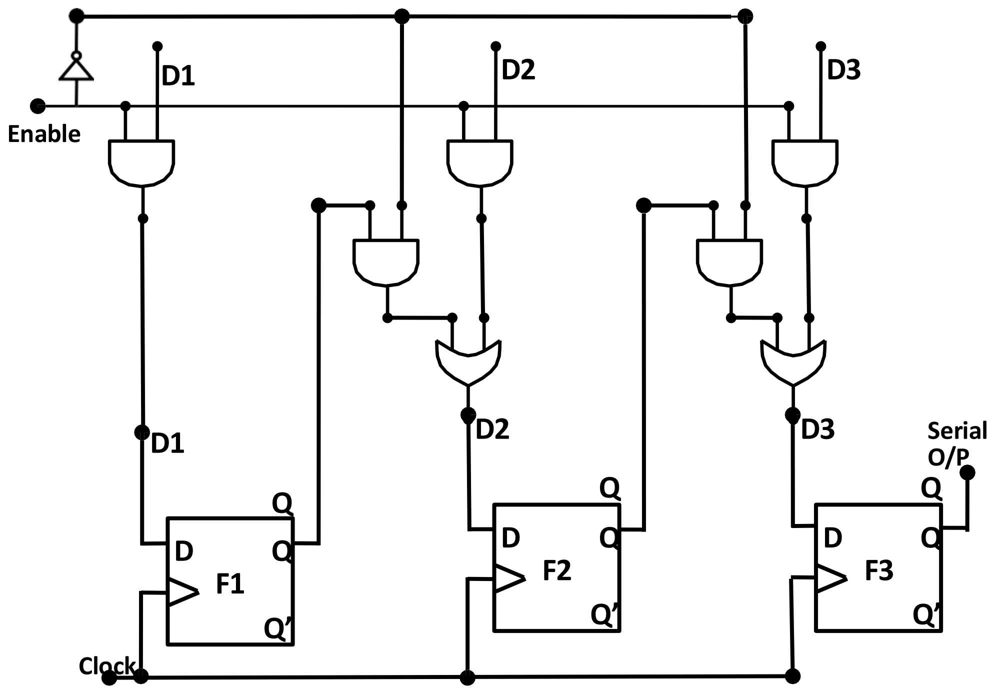


Figure 3.4.4 Parallel Input Serial Output Shift Register

From the above figure of PISO shift register, data input D1,D2 and D3 are loaded simultaneously, when Enable = 1. When Enable = 0, loaded data is taken out serially from Q3 output of F3 flip flop. On application of clock pulse.

APPLICATIONS OF SHIFT REGISTERS:

1.TheSISO shift register is used to provide a time delay from input to output. If N is the number of flip flop and fc is the clock frequency, then the time delay is-

$$T_d = N \times 1 / fc$$

2. RING COUNTER: Ring counter is a shift register, in which output of last stage is feedback to the input of first stage.

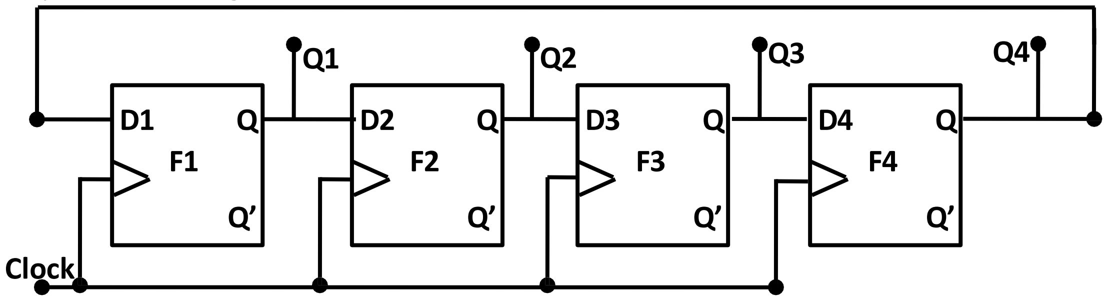


Figure3.4.5: 4-bit Ring Counter

	Q1	Q2	Q3	Q4	Clock pulse
Reference State	1	0	0	1	
	1	1	0	0	1
	0	1	1	0	2
	0	0	1	1	3
	1	0	0	1	4

Count Sequence

Above table shows the count sequence of 4-bit ring counter. Assuming the starting state as Q1=1,Q2=0,Q3=0 and Q4=1. After a clock pulse 1, Q1 bit is shifted to Q2, Q2 to Q3, Q3 to Q4 and from Q4 to Q1 and counter is in 1100 state. The second and third clock pulse produces 0110 and 0011 respectively. Further clock pulses causes the sequence to repeat. Since it has 4-different states before the sequence repeats, therefore it is also called MOD-4 counter.

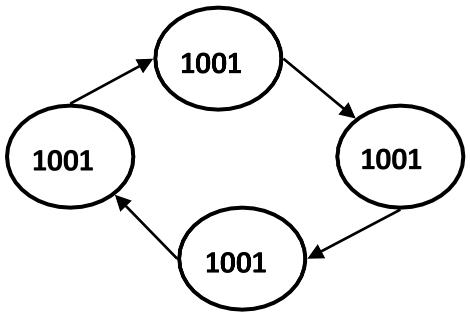


Figure 3.4.6: State Diagram of Ring Counter

3.5JOHNSON COUNTER (Twisted Ring Counter or Moebius Counter):

In johnson counter, the complement output of last stage flip flop is feedback to input of first stage.

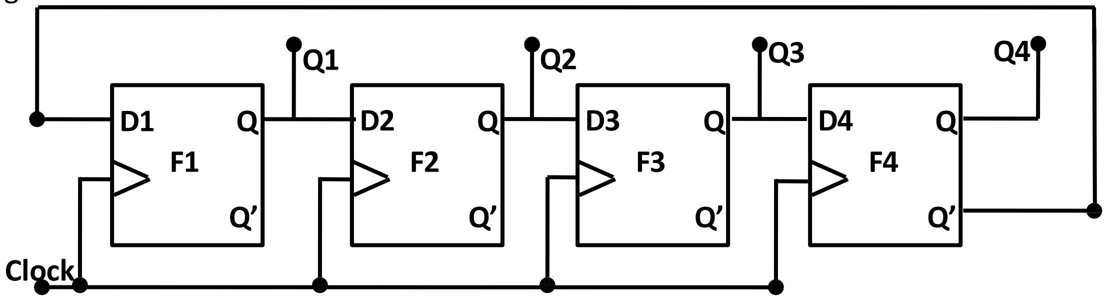


Figure3.5.1: 4-bit Johnson Counter

	Q1	Q2	Q3	Q4	Clock pulse
Reference State	0	0	0	0	
	1	0	0	0	1
	1	1	0	0	2
	1	1	1	0	3
	1	1	1	1	4
	0	1	1	1	5
	0	0	1	1	6
	0	0	0	1	7
	0	0	0	0	8

Table: Johnson Counter Count Sequence

3.6 ASYNCHRONOUS AND SYNCHRONOUS COUNTERS:

Counters are digital circuit which is used to count the clock pulses or to generate the sequence of states. Science clock pulses occur at known intervals, the counter can be used for measuring time and therefore period or frequency.

Asynchronous Counter or Serial Counter:

1. Each flip flop is triggered by the previous flip flop. Except the first flip flop.
2. Counter is simple and straight forward in operation.
3. Construction usually requires a minimum number of hardware.
4. Counters has a cumulative settling time.

Synchronous Counter or Parallel Counter:

1. Every flip flop is triggered by the clock (in synchronism).
2. Increase in speed of operation.
3. Construction- Increased in hardware.
4. Settling time is equal to the delay time of a single flip flop.

Modulus of Counter(MOD): Modulus of counter is the number of different states before the sequence repeats. Example- 3-bit binary counter has 8 different states, so MOD-8 counter.

ASYNCHRONOUS UP- COUNTER (BINARY RIPPLE COUNTER):

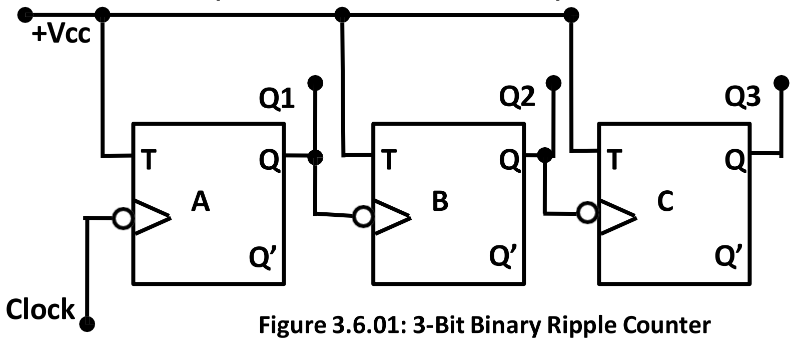


Figure 3.6.01: 3-Bit Binary Ripple Counter

3-bit binary ripple counter using T-Flip flop is shown above. From the figure, negative edge triggered clock drives the flip flop “A”. Output Q1 of flip flop “A” drives the flip flop “B” and output Q2 of flip flop “B” drives flip flop “C”. Flip flop “A” change state before it can trigger the flip flop “B” and flip flop “B” has to change state before it can trigger the flip flop “C”. The trigger moves through the flip flop like a ripple in water, hence the name Ripple Counter.

Operation: Let assume all flip flop are reset to produce “0”. Consider flip flop “A” as least significant bit (LSB) and flip flop “C” as most significant bit (MSB), so the content of counter is CBA = 0 0 0.

Cloc	Q	Q	Q	Coun
	0	0	0	**0**
a	0	0	1	**1**
b	0	1	0	**2**
c	0	1	1	**3**
d	1	0	0	**4**
e	1	0	1	**5**
f	1	1	0	**6**
g	1	1	1	**7**
h	0	0	0	**0**

TRUTH TABLE

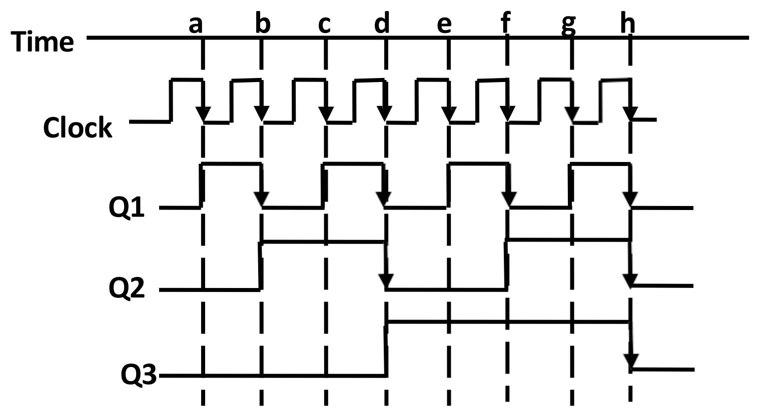


Figure 3.6.02: Waveforms

From the waveform, for every negative clock transition, output Q1 of flip flop A will change the state. Since flip flop A output Q1, act as a clock for flip flop B, each time the waveform at Q1 goes low, output Q2 of flip flop B will toggle (change the state). Each time the output Q2 of flip flop B goes low, output Q3 of flip flop C will toggle (change the state).

ASYNCHRONOUS DOWN COUNTER:

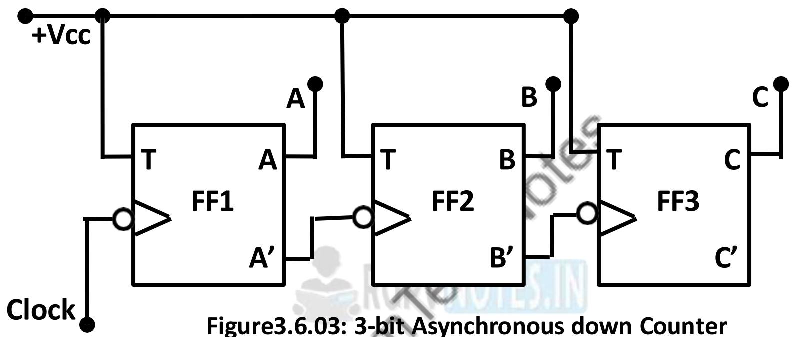


Figure3.6.03: 3-bit Asynchronous down Counter

System clock is used to drive flip flop 1, but the complement A' of flip flop-1, is used to drive the flip flop-2 and complement B' of flip flop-2, is used to drive the flip flop-3. Output A of flip flop-1 toggles with every negative clock transition. But flip flop 2 will toggle each time A goes high i.e. A' goes low and it is this negative transition that triggers flip flop-2. On the time line, flip flop-2 toggles at point a,c,e,g and i. similarly, flip flop-3 is triggered by B' and so flip flop-3 will toggle each time flip flop-2 goes high. Thus flip flop-3 toggles high at point a on time line and toggles back at point e and high at point i. Notice that the counter content are reduced by one count with each clock transition i.e in count down mode.

TRUTH TABLE

Clock	C	B	A	Count
	1	1	1	**7**
a	1	1	0	**6**
b	1	0	1	**5**
c	1	0	0	**4**
d	0	1	1	**3**
e	0	1	0	**2**
f	0	0	1	**1**
g	0	0	0	**0**
h	1	1	1	**7**

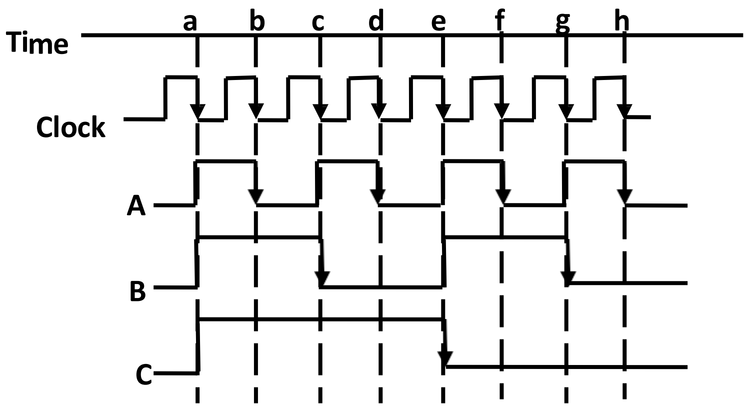


Figure 3.6.04: Waveforms

SYNCHRONOUS COUNTER OR PARALLEL COUNTER:

In a ripple counter (asynchronous counter) flip flop delay times are additive and the total settling time for the counter is approximate the delay times the total number of flip flops.

These problem is overcome by the use of a synchronous counter or parallel counter. Here every flip flop is triggered in synchronism with the clock.

SYNCHRONOUS 3-BIT BINARY UP COUNTER OR MOD-8 PARALLEL COUNTER:

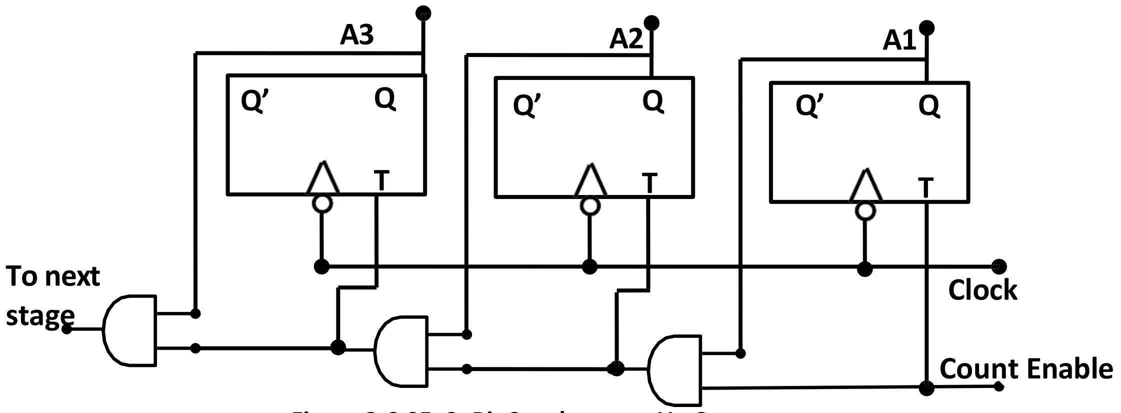


Figure 3.6.05: 3- Bit Synchronous Up Counter

Truth Table:

Count Enable	Clock pulse	A3	A2	A1	count
		0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	2
1	1	0	1	1	3
1	1	1	0	0	4
1	1	1	0	1	5
1	1	1	1	0	6
1	1	1	1	1	7
1	1	0	0	0	0

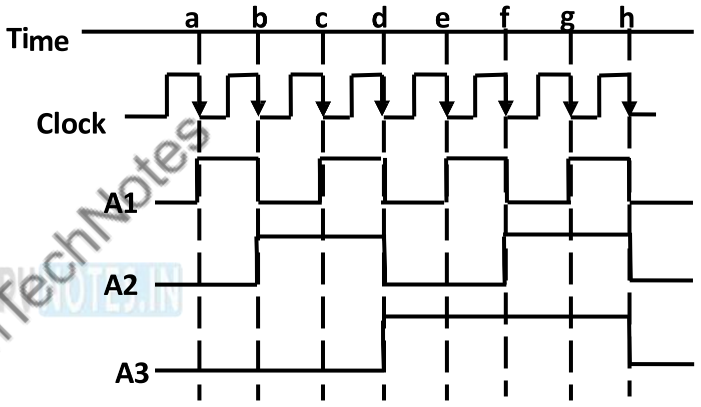


Figure 3.6.06: Waveforms

The truth table and waveforms of 3-bit synchronous up counter is shown above.

SYNCHRONOUS 3-BIT BINARY DOWN COUNTER :

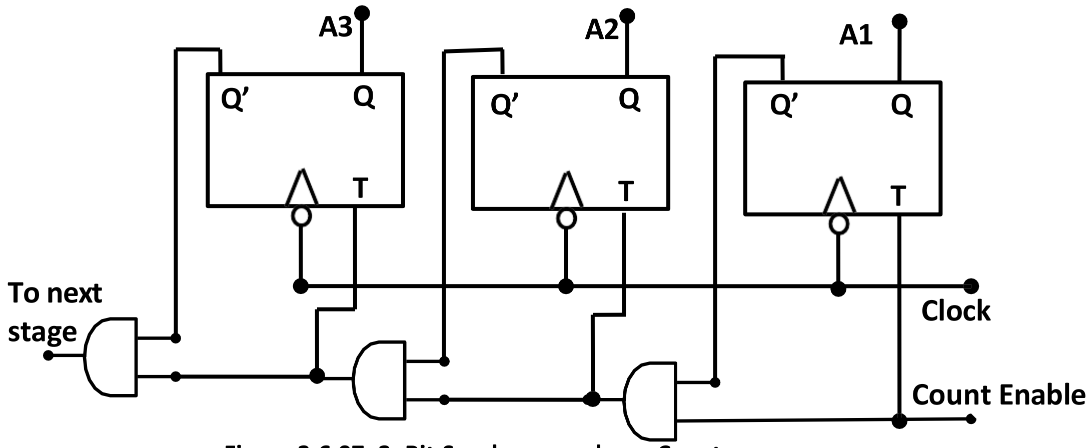
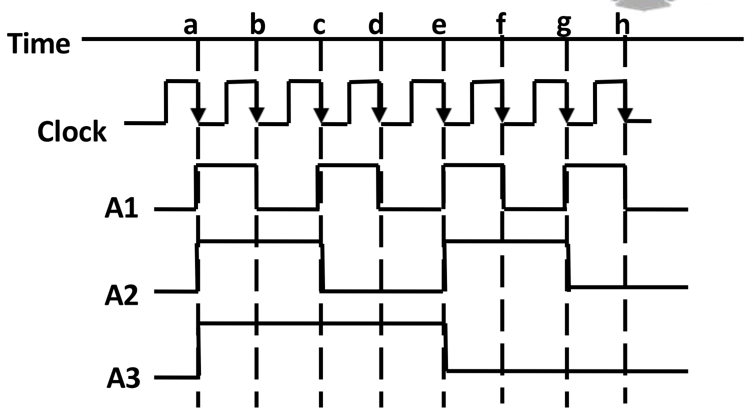


Figure 3.6.07: 3- Bit Synchronous down Counter

TRUTH TABLE

Clock	A3	A2	A1	Count
a	1	1	1	**7**
b	1	1	0	**6**
c	1	0	1	**5**
d	1	0	0	**4**
e	0	1	1	**3**
f	0	1	0	**2**
g	0	0	1	**1**
h	0	0	0	**0**
	1	1	1	**7**



Waveforms

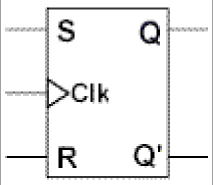
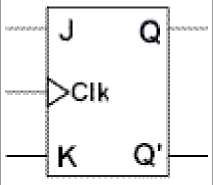
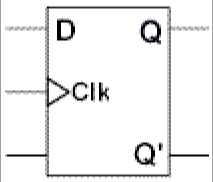
Figure 3.6.08: Waveform and truth table of 3-bit binary down counter

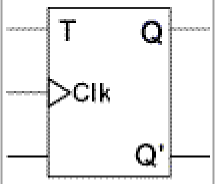
Since memory elements in sequential circuits are usually flip-flops, it is worth summarizing the behavior of various flip-flop types before proceeding further.

Summary of Characteristics Table & Excitation table

All flip-flops can be divided into four basic types: **SR**, **JK**, **D** and **T**. They differ in the number of inputs and in the response invoked by different value of input signals. The four types of flip-flops are defined in Table 1.

Table 1. Flip-flop Types

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		**S** | **R** | **Q(next)** 0 | 0 | Q 0 | 1 | 0 1 | 0 | 1 1 | 1 | ?	**Q(next) = S + R'Q** **SR = 0**	**Q** | **Q(next)** | **S** | **R** 0 | 0 | 0 | X 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 1 | X | 0
JK		**J** | **K** | **Q(next)** 0 | 0 | Q 0 | 1 | 0 1 | 0 | 1 1 | 1 | Q'	**Q(next) = JQ' + K'Q**	**Q** | **Q(next)** | **J** | **K** 0 | 0 | 0 | X 0 | 1 | 1 | X 1 | 0 | X | 1 1 | 1 | X | 0
D		**D** | **Q(next)** 0 | 0 1 | 1	**Q(next) = D**	**Q** | **Q(next)** | **D** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 0 1 | 1 | 1

T		**T** | **Q(next)** 0 | Q 1 | Q'	**Q(next) = TQ' + T'Q**	**Q** | **Q(next)** | **T** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0

Each of these flip-flops can be uniquely described by its graphical symbol, its characteristic table, its characteristic equation or excitation table. All flip-flops have output signals Q and Q'.

The ***characteristic table*** in the third column of Table 1 defines the state of each flip-flop as a function of its inputs and previous state. **Q** refers to the present state and **Q(next)** refers to the next state after the occurrence of the clock pulse. The characteristic table for the RS flip-flop shows that the next state is equal to the present state when both inputs S and R are equal to 0. When R=1, the next clock pulse clears the flip-flop. When S=1, the flip-flop output Q is set to 1. The equation mark (?) for the next state when S and R are both equal to 1 designates an indeterminate next state.

The characteristic table for the JK flip-flop is the same as that of the RS when J and K are replaced by S and R respectively, except for the indeterminate case. When both J and K are equal to 1, the next state is equal to the complement of the present state, that is, Q(next) = Q'.

The next state of the D flip-flop is completely dependent on the input D and independent of the present state.

The next state for the T flip-flop is the same as the present state Q if T=0 and complemented if T=1.

The characteristic table is useful during the analysis of sequential circuits when the value of flip-flop inputs are known and we want to find the value of the flip-flop output Q after the rising edge of the clock signal. As with any other truth table, we can use the map method to derive the characteristic equation for each flip-flop, which are shown in the third column of Table 1.

During the design process we usually know the transition from present state to the next state and wish to find the flip-flop input conditions that will cause the required transition. For this reason we will need a table that lists the required inputs for a given change of state. Such a list is called the ***excitation table***, which is shown in the fourth column of Table 1. There are four possible transitions from present state to the next state. The required input conditions are derived from the information available in the characteristic table. The symbol X in the table represents a "don't care" condition, that is, it does not matter whether the input is 1 or 0.

---------END----------

Subject Name: Digital Circuits & Systems **Subject Code**: IT 305

Subject Notes

UNIT-IV

Digital logic families: Bipolar and unipolar logic families, Digital IC specifications, RTL, DTL, All types of TTL circuits, ECL, IIL, PMOS, NMOS & CMOS Logic.

4.1 Digital Logic families:

It is a collection of different IC chips that have similar input, output and internal circuit characteristics i.e. group of compatible ICs with same logic levels and supply voltages but perform different logic functions.

NOTE:

1) Chips from same family can be interconnected.

2) Chips from different family may not be compatible, means they may use different power supply voltages and input, output conditions.

4.2 Bipolar and unipolar logic families

Logic families are mainly classified as two types as

Bipolar Logic Families: It mainly uses bipolar devices like diodes, transistors in addition to passive elements like resistors and capacitors.

These are sub classified as Saturated bipolar logic family and Unsaturated bipolar logic family.

1) Saturated Bipolar Logic Family: In this family the transistors used in ICs are driven into saturation.

Examples:

a) Transistor-Transistor Logic (TTL)

b) Resistor-Transistor Logic (RTL)

c) Direct Coupled Transistor Logic (DCTL)

d) Diode Transistor Logic (DTL)

e) High Threshold Logic(HTL)

f) Integrated Injection Logic (IIL or I2L)

2) Unsaturated Bipolar logic family: In this family the transistors used in ICs are not driven into saturation.

Examples:

a) Schottky TTL

b) Emitter Coupled Logic(ECL)

Unipolar Logic Families: It mainly uses Unipolar devices like MOSFETs in addition to passive elements like resistors and capacitors. These logic families have the advantages of high speed and lower power consumption than Bipolar families.

These are classified as

PMOS or P-Channel MOS Logic Family

NMOS or N-Channel MOS Logic Family

CMOS Logic Family

4.3 Digital IC Specifications:

Miniature, low-cost electronics circuits whose components are fabricated on a single, continuous piece of semiconductor material to perform a high-level function. This IC is usually referred to as a monolithic IC first introduced in 1958. The digital ICs are categorized as,

1. Small scale integration SSI <12 no of gates
2. Medium scale integration MSI 12 to 99 no of gates
3. Large scale integration LSI 100 to 9999 no of gates
4. Very large scale integration VLSI 10,000 or more

In this section, we will be concern only with the digital IC. Digital IC can be further categorized into bipolar or unipolar IC.

Bipolar ICs are devices whose active components are current controlled while unipolar ICs are devices whose active components are voltage controlled.

IC Packaging

1. IC packaging Protect the chip from mechanical damage and chemical contamination.
2. Provides a completed unit large enough to handle.
3. So that it is large enough for electrical connections to be made.
4. Material is molded plastic, epoxy, resin, or silicone. Ceramic used if higher thermal dissipation capabilities required. Metal/glass used in special cases.

Three most common packages for ICs are

a) Dual-in-line (DIPS) (most common)

b) Flat pack

c) Axial lead (TO5)

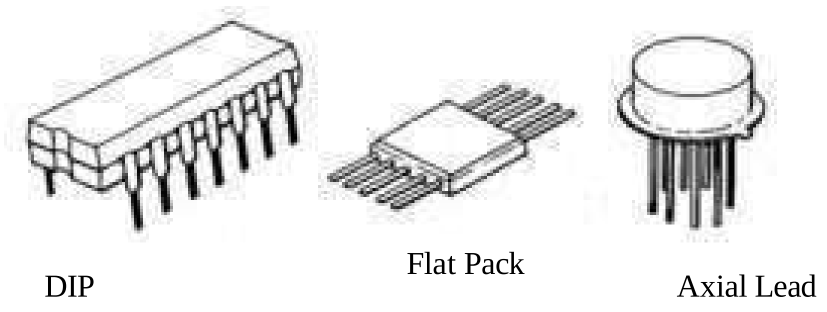


Fig. 4.3.1 Various IC Packages

Characteristics of Digital Logic ICs

Input /Output voltage level:

The following currents and voltages are specified which are very useful in the design of digital systems.

High-level input voltage, V_{IH} : This is the minimum input voltage which is recognized by the gate as logic 1.

Low-level input voltage, V_{IL}: This is the maximum input voltage which is recognized by the gate as logic 0.

High-level output voltage, V_{OH}: This is the minimum voltage available at the output correspondingto logic 1.

Low-level output voltage, V_{OL}: This is the maximum voltage available at the output corresponding to logic 0.

High-level input current, I_{IH} : This is the minimum current which must be supplied by a driving source corresponding to 1 level voltage.

Low-level input current, I_{IL}: This is the minimum current which must be supplied by a driving source corresponding to 0 level voltage.

High-level output current, I_{OH}: This is the maximum current which the gate can sink in 1 level.

Low-level output current, I_{OL}: This is the maximum current which the gate can sink in 0 level.

High-level supply current, I_{CCh} (1): This is the supply current when the output of the gate is at logic 1.

Low-level supply current, I_{CCL} (0): This is the supply current when the output of the gate is at logic (0).

Propagation Delay:

Definition: The time required for the output of a digital circuit to change states after a change at one or more of its inputs. The speed of a digital circuit is specified in terms of the propagation delay time. The delay times are measured between the 50 percent voltage levels of input and output waveforms. There are two delay times, tpHL: when the output goes from the HIGH state to the LOW state and tpLH, corresponding to the output making a transition from the LOW state to the HIGH state. The propagation delay time of the logic gate is taken as the average of these two delay times.

Fan-in:

Definition: Fan-in (input load factor is the number of input signals that can be connected to a gate without causing it to operate outside its intended operating range. expressed in terms of standard inputs or units loads (ULs).

Fan-out:

Definition:Fan-out (output load factor) is the maximum number of inputs that can be driven by a logic gate. A fanout of 10 means that 10 unit loads can be driven by the gate while still maintaining the output voltage within specifications for logic levels 0 and 1.

4.4 RTL:

RTL is a type of digital circuits built using resistors as the input network and bipolar junction transistors (BJTs) as switching devices. RTL is the earliest class of transistorized digital logic circuit used commercially.

Two input RTL NOR Gate: A simple two input RTL NOR gate is shown in the Fig.4.4.1

The circuit consists of four resistors and one transistor(T).Here the transistor acts as an invertor.

Working : When both the inputs are zero(A=0,B=0), the input to the transistor is zero.So the output of the transistor is HIGH or Logic 1.Similarly,when one of the inputs is HIGH (logic1),the input to the transistor is HIGH. So the output Q is LOW or logic 0

On the same lines when both the inputs are HIGH the input to the transistor is HIGH(logic1) So the output of the transistor is LOW or logic 0.This is the working of two input NOR gate which is also shown in the truth table 4.4.1.

A	B	$Y = \overline{A+B}$
0	0	1
1	0	0
0	1	0
1	1	0

Table.4.4.1 Truth table for two input RTL NOR gate

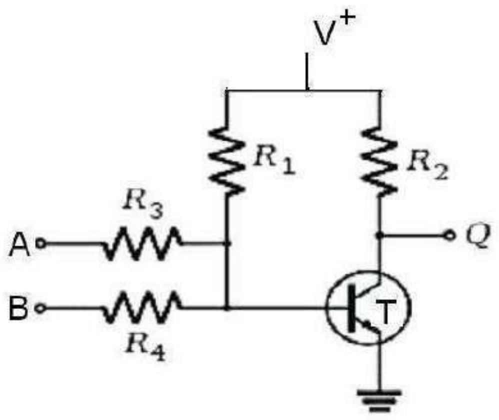


Fig.4.4.1 Two Input RTL NOR gate circuit

The basic advantage of RTL technology is that it uses a minimum number of transistors, which was an important consideration before integrated circuit technology ,as transistors were the most expensive component to produce. (Early IC logic production such as Fairchild's in 1961, used the same RTL approach briefly, but quickly transitioned to higher-performance circuits such as diode–transistor logic and then transistor–transistor logic , since diodes and transistors were no more expensive than resistors).

The obvious disadvantage of RTL is its high current dissipation when the transistor conducts to overdrive the output biasing resistor. Also another limitation is its limited Fan-In, 3 inputs being the limit for many circuit designs.

Note: Now a days the RTL circuits are no more in use.They are obsolete

4.5 DTL: It is a class of digital circuits built from bipolar junction transistors (BJT), diodes and resistors. It is called **Diode–Transistor logic** because the logic gating function (e.g., AND) is performed by a diode network and the amplifying function is performed by a transistor (contrast this with RTL and TTL).

TWO Input DTL NAND gate:

A simple two input DTL NAND gate circuit is shown in the Fig.4.5.1.It consists of two Diodes,four resistors and one Transistor and a power supply V^+.Here the Transistor acts as an invertor.

Working :When the inputs A & B are at logic 0(LOW) , the current due to V^+ flows through the Diodes to ground and the current through the resistors R3 is Zero.i.e the input to the Transistor T is zero and ,the Transistor is in off state and the output at Q is High (Logic1).

Similarly when one of the inputs is HIGH (Logic1) ,the other will be at LOW.So the current due to V^+ will pass through one diode and the other diode is in reverse bias..i.e the input to the transistor is zero or the transistor is in cut-off state. So the output at Q is HIGH or at logic1.

Suppose both the inputs are HIGH (logic 1).The current due to V^+ do not pass through the diodes.The input of the Transistor will be at Logic 1.i.e the transistor is in conducting state.Hence the output of the Transistor is LOW or at logic 0. This working of NAND gate is shown in the truth table 4.5.1

A	B	$Y=\overline{A.B}$
0	0	1
1	0	1
0	1	1
1	1	0

Table 4.5.1 The truth table for two input DTL NAND gate

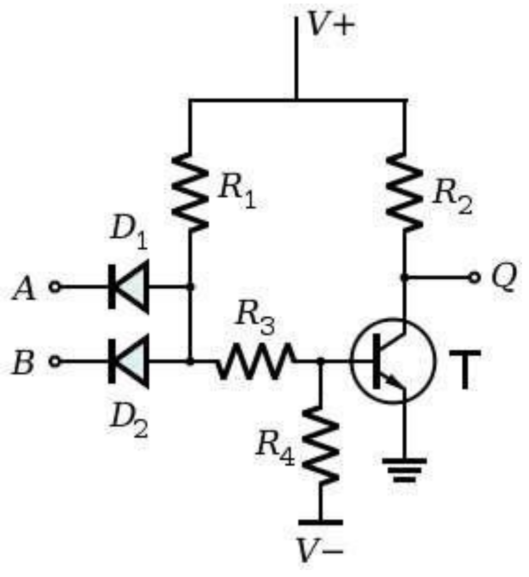


Fig.4.5.1Two input DTL NAND gate circuit.

One problem that DTL suffers is its low speed, especially when the transistor is being turned off. Turning off a saturated transistor in a DTL gate requires it to first pass through the active region before going into cut-off. Cut-off, however, will not be reached until the stored charge in its base has been removed. The dissipation of the base charge takes time if there is no available path from the base to ground. This is why in DTL circuits a base resistor is connected to ground so that it provides a path for discharge as shown in the figure. Another problem with turning off the DTL output transistor is the fact that the effective capacitance of the output needs to charge up through R_2 before the output voltage rises to the final logic '1' level, which also consumes a relatively large amount of time.

But, one advantage of DTL over RTL is its better noise margin and greater fan-outs than RTL, but it suffers from low speed, especially in comparison to TTL.

4.6 All types of TTL circuits:

It is a logic family consisting completely of transistors. It employs transistor with multiple emitters. Commercially it starts with the 74 series like the 7404, 74S86 etc. It was build in 1961 by James L Bui and commercially used in logic design in 1963.

Classification of TTL:
TTLs are classified based on the output.

1)Open Collector Output: The main feature is that its output is 0 when low and floating when high. Usually an external Vcc may be applied.

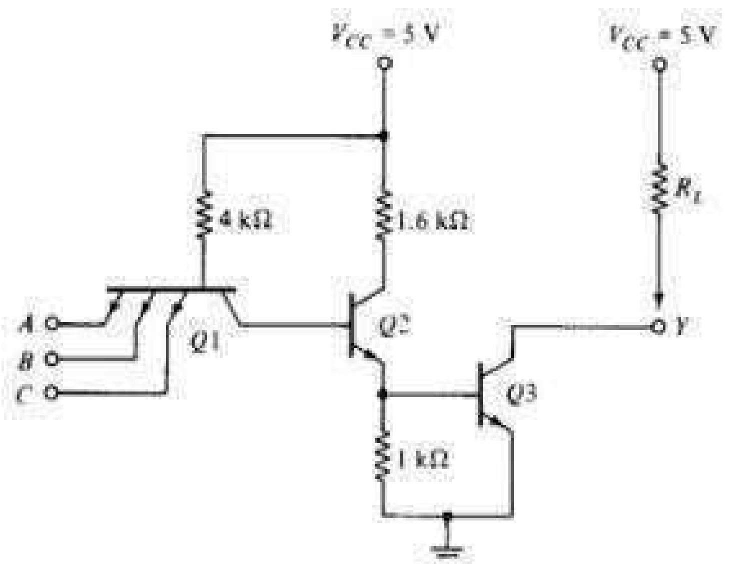


Fig.4.6.1 Open collector TTL Configuration

Transistor Q1 actually behaves as cluster of diodes placed back to back. With any of the input at logic low, corresponding emitter base junction is forward biased and the voltage drop across the base of Q1 is around 0.9V, not enough for the transistors Q2 and Q3 to conduct. Thus output is either floating or Vcc, i.e. High level.

Similarly when all inputs are high, all base emitter junctions of Q1 are reverse biased and transistor Q2 and Q3 get enough base current and are in saturation mode. Clearly output is at logic low. (For a transistor to go to saturation, collector current should be greater than β times the base current).

Applications of open-collector output:

It is used in 3 major applications:

1. In driving lamps or relays
2. In performing wired logic
3. In construction of a common bus system

2. Totem Pole Output:

Totem Pole means addition of an active pull up circuit in the output of the Gate which results in reduction of propagation delay.

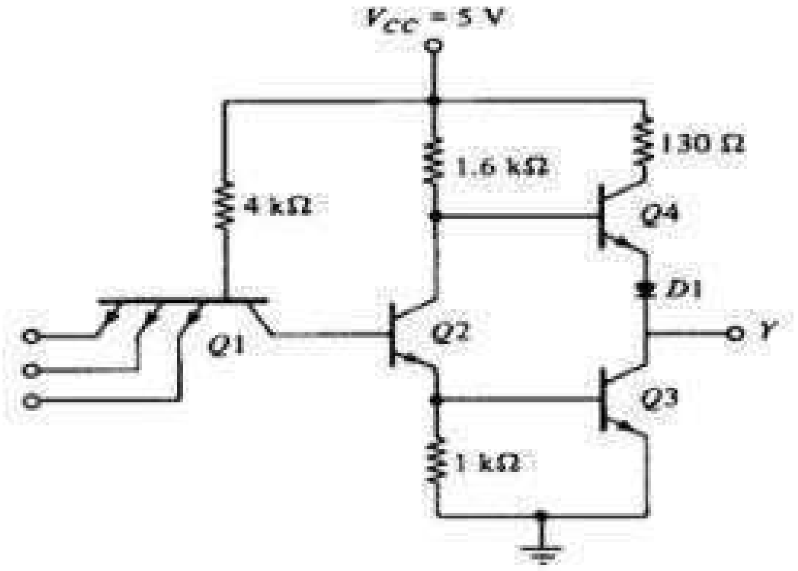


Fig 4.6.2 Totem pole Configuration

Logic operation is same as the open collector output. Use of transistors Q4 and diode is to provide quick charging and discharging of parasitic capacitance across Q3. Resistor is used to keep the output current to a safe value.

3. Three state Gate:

It provides 3 state output.

1. Low level state when lower transistor is ON and upper transistor is OFF.
2. High level state when lower transistor is OFF and upper transistor is ON.

3. Third state when both transistors are OFF. It allows a direct wire connection of many outputs.

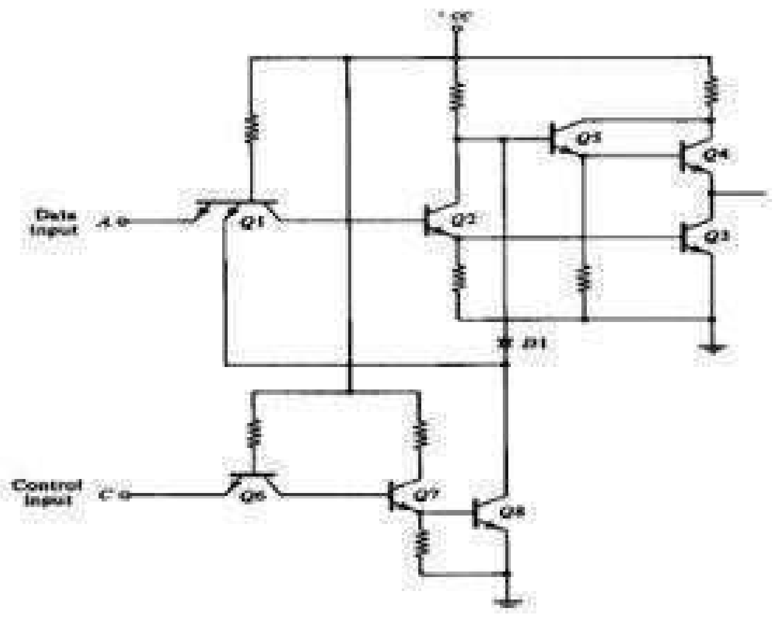


Fig 4.6.3 Totem pole Configuration

Features of TTL Family:

1. Logic low level is at 0 or 0.2V.
2. Logic high level is at 5V.
3. Typical fan out of 10. It means it can support at most 10 gates at its output.
4. A basic TTL device draws a power of almost 10mW, which reduces with use of schottky devices.
5. Average propagation delay is about 9ns.
6. The noise margin is about 0.4V.

Series of TTL IC:

TTL ICs mostly start with the 7 series. It has basically 6 subfamilies given as:

1. Low Power device with propagation delay of 35 ns and power dissipation of 1mW.
2. Low power Schottky device with delay of 9ns
3. Advanced Schottky device with delay of 1.5ns.
4. Advanced low power Schottky device with delay of 4 ns and power dissipation of 1mW.

In any TTL device nomenclature, first two names indicate the name of the subfamily the device belongs to. The first two digits indicate the temperature range of operation. The next two alphabets indicate the subfamily the device belongs to. The last two digits indicate the logic function performed by the chip.

TTL Applications:

1. Used in controller application for providing 0 to 5Vs
2. Used as switching device in driving lamps and relays

3. Used in processors of mini computers like DEC VAX
4. Used in printers and video display terminals

4.7 ECL:

ECL was invented in August 1956 at IBM by Hannon S. Yourke. Originally called current-steering logic, it was used in the Stretch, IBM 7090, and IBM 7094 computers.

Emitter coupled logic (ECL) is the fastest of all logic families and therefore used in applications where very high speed is needed. The ECL is fastest because ,the transistors are used in difference amplifier configuration in which they are never driven into saturation .The transistor is driven between On and OFF states(cut-off & active) only.Hence propagation delay of less than 1ns per gate has become possible.(In other transistor based logic circuits ,switching is between cut-off and saturation states.Hence delay is more).Basically ECL is realized using difference amplifier in which the emitters of the two transistors are connected and hence it is referred to as emitter-coupled logic.

The ECL circuits usually operate with negative power supplies (positive end of the supply is connected to ground) in contrast to other logic families in which negative end of the supply is grounded. This is done mainly to minimize the influence of the power supply variations on the logic levels as ECL is more sensitive to noise on the V_{CC} and relatively immune to noise on V_{EE}. Because ground should be the most stable voltage in a system, ECL is specified with a positive ground.

Basic ECL Circuit : The basic circuit configuration consists of a pair of NPN transistors with their emitters connected together and fed by a current source I_E as show in the Fig.4.7.1.

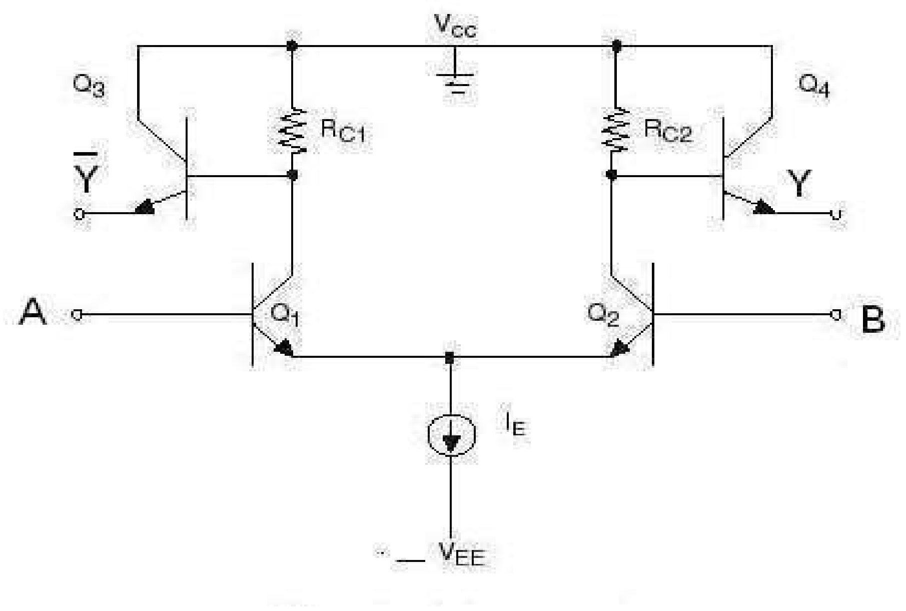


Fig.4.7.1 Basic ECL Configuration

Emitter coupled logic is basically a combination of differential amplifier with Q1 &Q2 and Emitter followers with Q3 & Q4. This emitter-follower provides low output impedance typically of the order of 6-8 Ω.In the steady state, either Q1 or Q2 is ON but not both, and the output logic state is determined by the voltage difference between the bases of Q1 and Q2. If $Vb_1 - Vb_2 > 200$ mV, Q1 will be turned ON and Q2 turned OFF, and vice versa.

The inputs can be driven either differentially or single-ended. In the single-ended mode, the non-driven base must be connected to a suitable bias voltage, V_{BB}, which is either supplied internally by the device, or externally. The voltages developed at the collectors of Q1 and Q2 are connected to a pair of emitter followers, Q3 and Q4. The outputs are taken at the emitters of Q3 and Q4.The two outputs are complements of each other.So if one is OR the other will be the NOR output. Note that the output emitters are open, and, unlike TTL/CMOS circuits, there will be no output until a pull-down resistor is connected to the open emitter. This pull-down resistor plays a very important role in determining the performance of the ECL circuit.

The important specifications of ECL are : The power dissipation is of the order of 50mW and Fan-out is typically around 15 to 20.Due to the disadvantage of high power dissipation and relatively low noise margins ECL family has limited applications ,except in high frequency applications.

4.8 IIL(Integrated Injection Logic:I^2L):

Integrated Injection Logic eliminates the need for any resistors, capacitors or transistor isolation. Thisenables an extremely compact logic circuit to be formed which has low power consumption while maintaining the normal speed of transistor-transistor logic. I^2L is built with multiple collector bipolar junction transistors. Although the logic levels are very close (High: 0.7V, Low: 0.2V), I^2L has high noise immunity because it operates by current instead of voltage. It is also known as merged-transistor logic.

Operation :IIL circuit The heart of an I2L circuit is the common emitter open collector inverter. Typically, an inverter consists of an NPN transistor with the emitter connected to ground and the base biased with a forward current. The input is supplied to the base as either a current sink (low logic level) or as a high-z floating condition (high logic level). The output of an inverter is at the collector. Likewise, it is either a current sink (low logic level) or a high-z floating condition (high logic level). Like direct-coupled transistor logic, there is no resistor between the output (collector) of one NPN transistor and the input (base) of the following transistor. To understand how the inverter operates, it is necessary to understand the current flow. If the bias current is shunted to ground (low logic level), the transistor turns off and the collector floats (high logic level). If the bias current is not shunted to ground because the input is high-z (high logic level), the bias current flows through the transistor to the emitter, switching on the transistor, and allowing the collector to sink current (low logic level). Because the output of the inverter can sink current but cannot source current, it is safe to connect the outputs of multiple inverters together to form a wired AND gate. When the outputs of two inverters are wired together, the result is a two-input NOR gate because the configuration (NOT A) AND (NOT B) is equivalent to NOT (A OR B). This logical relationship is known as De Morgan's Theorem.

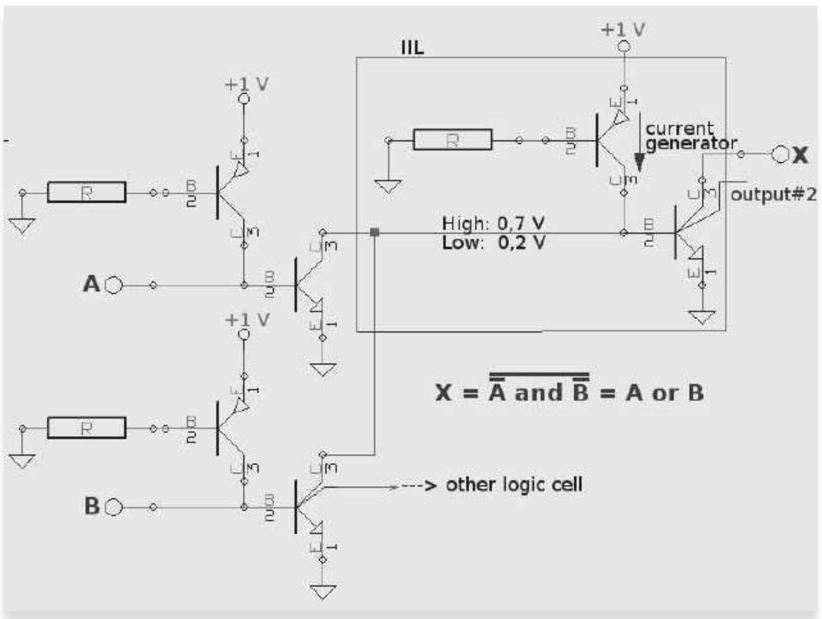


Fig.4.8.1 IIL Configuration Logic

4.9 PMOS, NMOS & CMOS Logic:

PMOS:

P-type metal-oxide-semiconductor logic uses p-channel metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits. PMOS transistors operate by creating an inversion layer in an n-type transistor body. This inversion layer, called p-channel, can conduct holes between p-type "source" and "drain" terminals. The p-channel is created by applying voltage to the third terminal called gate. Like other MOSFETs, PMOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation. The p-type MOSFETs are arranged in a so-called "pull-up network" (PUN) between the logic gate output and positive supply voltage, while a resistor is placed between the logic gate output and the negative supply voltage. The circuit is designed such that if the desired output is high, then the PUN will be active, creating a current path between the positive supply and the output.

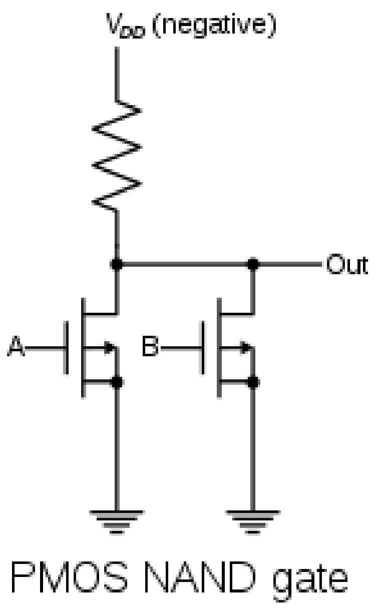


Fig 4.9.1 Pmos Logic for NAND Logic

NMOS:

N-type metal-oxide-semiconductor logic uses n-type metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits. NMOS transistors have four modes of operation: cut-off (or sub-threshold), triode, saturation (sometimes called active), and velocity saturation. The n-type MOSFETs are arranged in a so-called "pull-down network" (PDN) between the logic gate output and negative supply voltage, while a resistor is placed between the logic gate output and the positive supply voltage. The circuit is designed such that if the desired output is low, then the PDN will be active, creating a current path between the negative supply and the output.

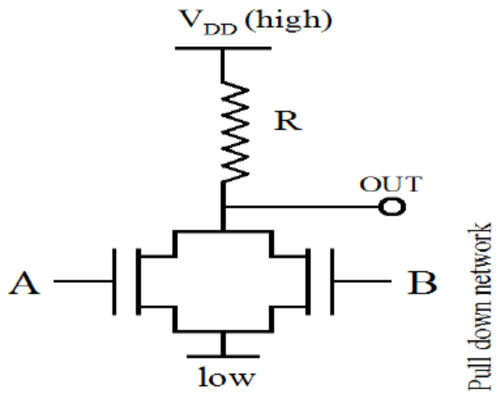


Fig 4.9.2 Nmos Logic for NOR Logic

CMOS:

Complementary metal–oxide–semiconductor (CMOS) /'si:mɒs/ is a technology for constructing integrated circuits. CMOS technology is used in microprocessors, microcontrollers, static RAM, and other digital logic circuits. CMOS technology is also used for several analog circuits such as image sensors (CMOS sensor), data converters, and highly integrated transceivers for many types of communication. Frank Wanlass patented CMOS in 1963 (US patent 3,356,858). CMOS is also sometimes referred to as complementary-symmetry metal–oxide–semiconductor (or COS-MOS).The words "complementary-symmetry" refer to the fact that the typical digital design style with CMOS uses complementary and symmetrical pairs of p-type and n-type metal oxide semiconductor field effect transistors (MOSFETs) for logic functions. Two important characteristics of CMOS devices are high noise immunity and low static power consumption. Since one transistor of the pair is always off, the series combination draws significant power only momentarily during switching between on and off states. Consequently, CMOS devices do not produce as much waste heat as other forms of logic, for example transistor–transistor logic (TTL) or NMOS logic, which normally have some standing current even when not changing state. CMOS also allows a high density of logic functions on a chip. It was primarily for this reason that CMOS became the most used technology to be implemented in VLSI chips.

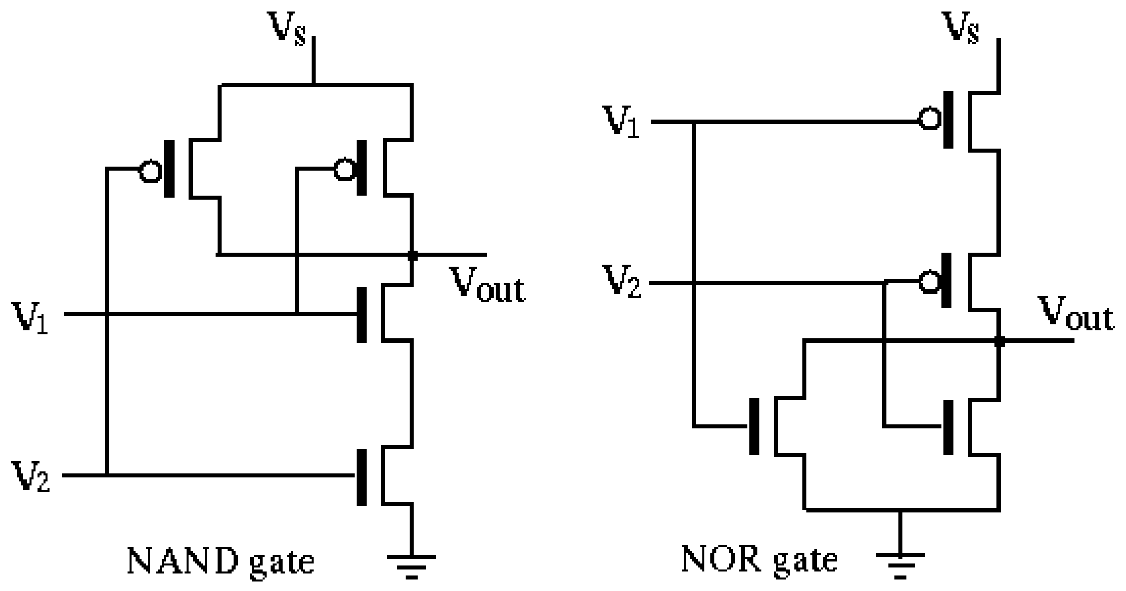
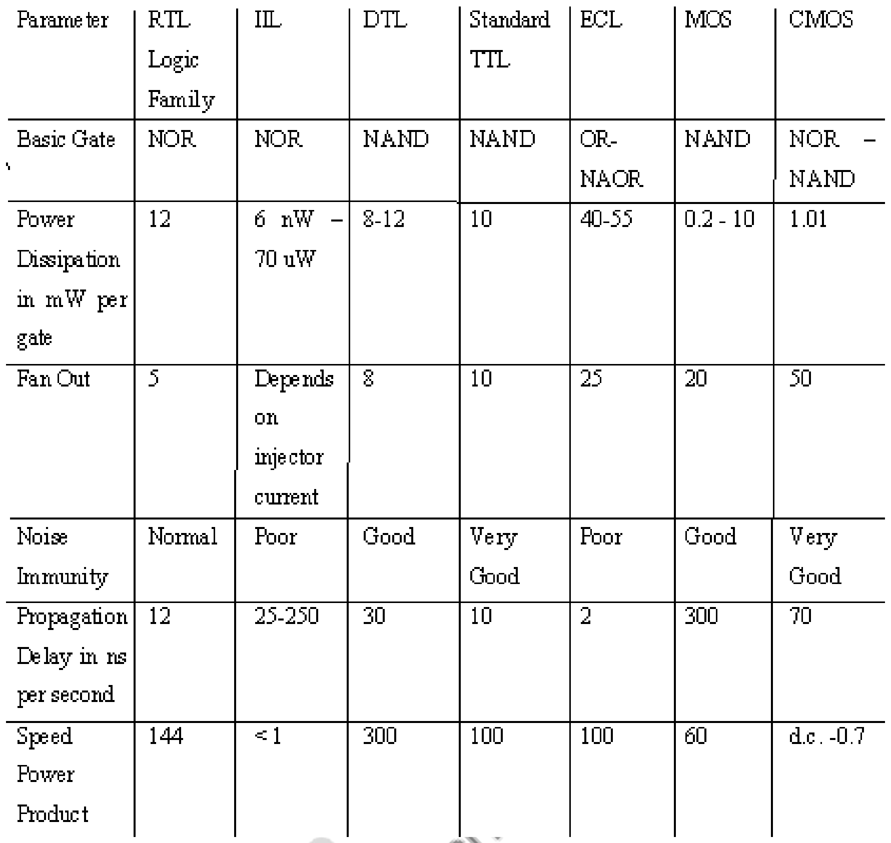


Fig 4.9.3 Cmos Logic for NAND& NOR Logic

Summary of Comparison of Various Characteristics of Logic Families:



Parameter	RTL Logic Family	IIL	DTL	Standard TTL	ECL	MOS	CMOS
Basic Gate	NOR	NOR	NAND	NAND	OR-NAOR	NAND	NOR – NAND
Power Dissipation in mW per gate	12	6 nW – 70 uW	8-12	10	40-55	0.2 - 10	1.01
Fan Out	5	Depends on injector current	8	10	25	20	50
Noise Immunity	Normal	Poor	Good	Very Good	Poor	Good	Very Good
Propagation Delay in ns per second	12	25-250	30	10	2	300	70
Speed Power Product	144	< 1	300	100	100	60	d.c. -0.7

Table 4.9.1 Comparison of Various Characteristics of Logic Families

------------------------END OF UNIT----------------------------

Subject Name: Digital Circuits & System **Subject Code**: IT 305

Subject Notes

UNIT-V

Clocks and timing circuits: Bistable, Monostable & Astable multivibrator, Schmitt trigger circuit, Introduction of Analog to Digital & Digital to Analog converters, Display devices, 7 and 16 segment LED display, LCD.

5.1 Clock and Timing circuits:

Circuits characterized by the existence of some well-defined states, amongst which take place fast transitions, called switching processes. A switching process is a fast change in value of a current or a voltage, the fast process implying the existence of positive reaction loops, or negative resistances. The switching can be triggered from outside, by means of command signals, or from inside, by slow charge accumulation and the reaching of a critical state by certain electrical quantities in the circuit. Circuits have two, well defined states, which can be either stable or unstable.

A stable state is a state, in which the circuit, in absence of a driving signal, can remain for an unlimited period of time The circuit can remain in an unstable state only for a limited period of time, after which, in the absence of any exterior command signals, it switches into the other state.

The multivibrator circuits can be grouped, according to their number of stable (steady) states, into: - flip-flops (bistable circuits) with both states being stable - monostable circuits, having a stable and an unstable state - astable circuits, with both states being unstable

5.2 Schmitt Trigger or Regenerative Comparator Circuit:

A Schmitt trigger circuit is also called a regenerative comparator circuit. The circuit is designed with a positive feedback and hence will have a regenerative action which will make the output switch levels. Also, the use of positive voltage feedback instead of a negative feedback, aids the feedback voltage to the input voltage, instead of opposing it. The use of a regenerative circuit is to remove the difficulties in a zero-crossing detector circuit due to low frequency signals and input noise voltages. Shown below is the circuit diagram of a Schmitt trigger. It is basically an inverting comparator circuit with a positive feedback. The purpose of the Schmitt trigger is to convert any regular or irregular shaped input waveform into a square wave output voltage or pulse. Thus, it can also be called a squaring circuit.

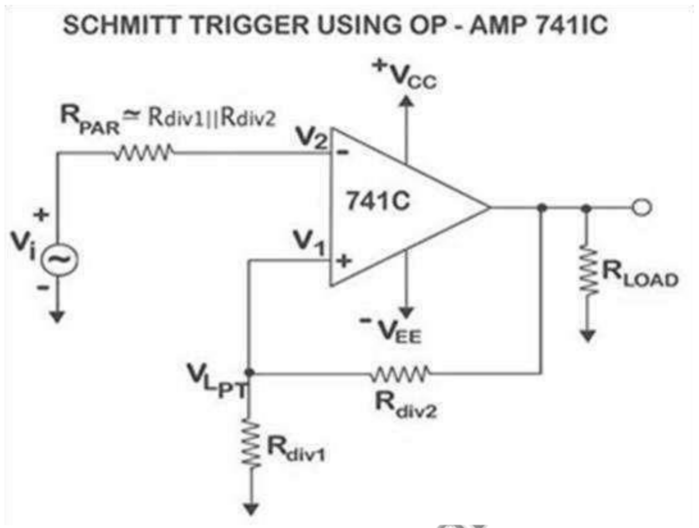


Fig 5.2.1 Schmitt Trigger Circuit Using Op-Amp uA741 IC

As shown in the circuit diagram, a voltage divider with resistors Rdiv1 and Rdiv2 is set in the positive feedback of the 741 IC op-amp. The same values of Rdiv1 and Rdiv2 are used to get the resistance value R_{par} = Rdiv1||Rdiv2 which is connected in series with the input voltage. R_{par} is used to minimize the offset problems. The voltage across R1 is feedback to the non-inverting input.

The input voltage Vi triggers or changes the state of output V_{out} every time it exceeds its voltage levels above a certain threshold value called Upper Threshold Voltage (V_{upt}) and Lower Threshold Voltage (V_{lpt}).

Let us assume that the inverting input voltage has a slight positive value. This will cause a negative value in the output. This negative voltage is fedback to the non-inverting terminal (+) of the op-amp through the voltage divider. Thus, the value of the negative voltage that is fedback to the positive terminal becomes higher. The value of the negative voltage becomes again higher until the circuit is driven into negative saturation (-Vsat). Now, let us assume that the inverting input voltage has a slight negative value. This will cause a positive value in the output. This positive voltage is feedback to the non-inverting terminal (+) of the op-amp through the voltage divider. Thus, the value of the positive voltage that is feedback to the positive terminal becomes higher. The value of the positive voltage becomes again higher until the circuit is driven into positive saturation (+Vsat). This is why the circuit is also named a regenerative comparator circuit.

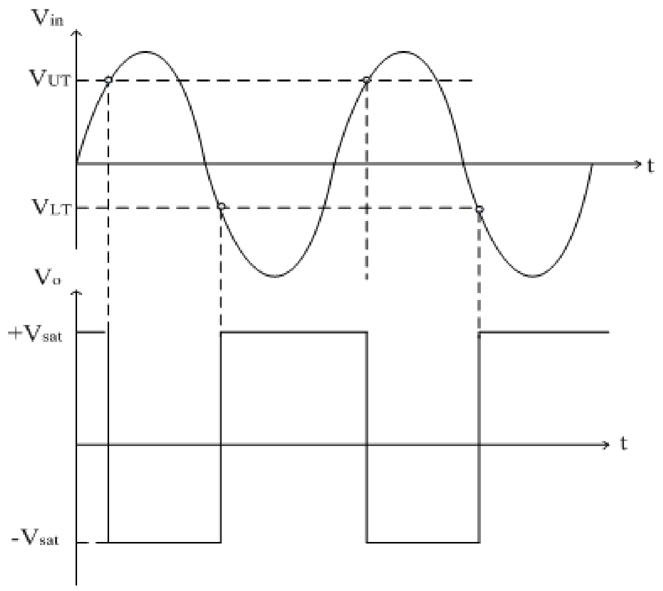


Fig 5.2.2 Schmitt Trigger Input and Output Waveform

Working :

When Vout = +Vsat, the voltage across Rdiv1 is called Upper Threshold Voltage (Vupt). The input voltage, Vin must be slightly more positive than V_{upt} inorder to cause the output Vo to switch from +Vsat to -Vsat. When the input voltage is less than V_{upt}, the output voltage Vout is at +Vsat.

Upper Threshold Voltage, $V_{upt} = +V_{sat}$ (Rdiv1/[Rdiv1+Rdiv2])

When Vout = -Vsat, the voltage across Rdiv1 is called Lower Threshold Voltage (Vlpt). The input voltage, Vin must be slightly more negative than V_{lpt} inorder to cause the output Vo to switch from -Vsat to +Vsat. When the input voltage is less than V_{lpt}, the output voltage Vout is at -Vsat.

Lower Threshold Voltage, V_{lpt} = -Vsat (Rdiv1/[Rdiv1+Rdiv2])

If the value of V_{upt} and V_{lpt} are higher than the input noise voltage, the positive feedback will eliminate the false output transitions. With the help of positive feedback and its regenerative behaviour, the output voltage will switch fast between the positive and negative saturation voltages.

Hysteresis Characteristics

Since a comparator circuit with a positive feedback is used, a dead band condition hysteresis can occur in the output. When the input of the comparator has a value higher than V_{upt}, its output switches from +Vsat to -Vsat and reverts back to its original state, +Vsat, when the input value goes below V_{lpt}. This is shown in the figure below. The hysteresis voltage can be calculated as the difference between the upper and lower threshold voltages.

$V_{hysteresis} = V_{upt} – V_{lpt}$

Substituting the values of V_{upt} and V_{lpt} from the above equations:

$V_{hysteresis}$= +Vsat (Rdiv1/Rdiv1+Rdiv2) – {-Vsat (Rdiv1/Rdiv1+Rdiv2)}

$V_{hysteresis}$ = (Rdiv1/Rdiv1+Rdiv2) {+Vsat – (-Vsat)}

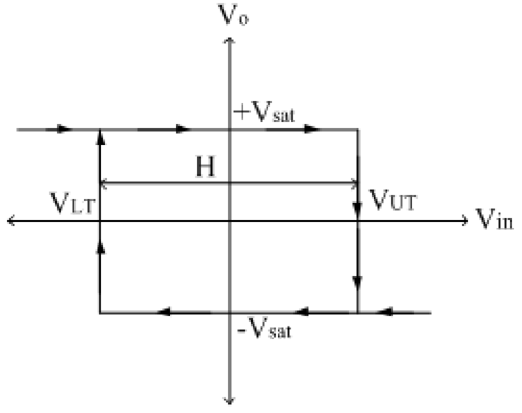


Fig 5.2.3 Schmitt-Trigger-Hysteresis Characteristics

Applications of Schmitt Trigger

Schmitt trigger is mostly used to convert a very slowly varying input voltage into an output having abruptly varying waveform occurring precisely at certain predetermined value of input voltage. Schmitt trigger may be used for all applications for which a general comparator is used. Any type of input voltage can be converted into its corresponding square signal wave. The only condition is that the input signal must have large enough excursion to carry the input voltage beyond the limits of the hysteresis range. The amplitude of the square wave is independent of the peak-to-peak value of the input waveform.

5.3 Introduction of Analog to Digital & Digital to Analog converters

In general, there are two types of electrical signals – the analog signal and the digital signal. The linear electronics circuit like linear amplifier, transducers; generate such signals, while the computers generate digital signals. In analog signal, voltage level changes smoothly or linearly with respect to time, while in digital signal voltage level changes abruptly or suddenly. Sometimes it is required to connect output of a transducer to input of a digital circuit, like a digital display. If they are connected directly then the results are quite unsatisfactory. Because the digital circuit cannot 'understand' the linear signals of transducer. Hence, we will NOT get satisfactory digital output from the circuit. a) Therefore, whenever we need to interface (connect) analog and digital circuits, we need to convert either analog signal into proportional digital signal or vice versa. In this way, we need the A/D and D/A conversion. Examples: A/D conversion is required is digital voltmeter, A/D and D/A conversions, both, are required in temperature controller, A/D conversion is required in digital recording system etc.

Analog to Digital Converters (ADC)

The process of converting analog signal into equivalent digital signal is called A/D conversion. The electronics circuit, which does this process, is called A/D converter. The circuit has only one input with 'n' number of digital outputs. The basic idea of this process is given below –

Simultaneous A/D converter –A comparator compares the unknown voltage with a known value of voltage and then produces proportional output (i.e. It will produce either a 1 or a 0). This principle is basically used in the above circuit. Here three comparators are used. Each has two inputs. One input of each comparator is connected to analog input voltage.

The other input terminals are connected to fixed reference voltage like +3/4V, +V/2 and +V/4. Now the circuit can convert analog voltage into equivalent digital signal. Since the analog output voltage is connected in parallel to all the comparators, the circuit is also called as parallel A/D converter.

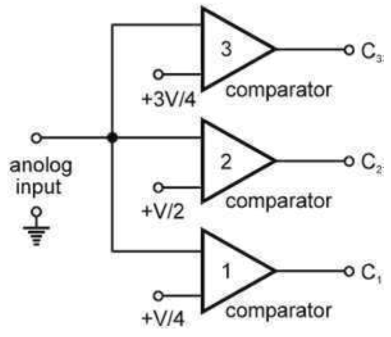


Fig 5.3.1Simultaneous A/D Converter

Working – Here each comparator is connected to a reference voltage of +3/4V, +V/2 and +V/4 with their outputs as $C_3C_2C_1$ respectively. Now suppose the analog input voltage change from 0 – 4V, then the actual values of reference voltages will be +3/4V = 3V, +V/2 =2V and +V/4 =1V. Now there will be following conditions of outputs of the circuit–

1) When input voltage is between 0 and 1V, the output will be $C_3C_2C_1$ =000.
2) When input voltage>1V: upto2V,the output will be $C_3C_2C_1$=001.
3) When input voltage>2V: upto3V,the output will be $C_3C_2C_1$=011.
4) When input voltage>3V:upto 4V,the output will be $C_3C_2C_1$=111.

In this way, the circuit can convert the analog input voltage into its equivalent or proportional binary number in digital style.

Counter type A/D converter–Here one input of comparator is connected to the analog signal (i.e.unknown voltage) and other is connected to the analog output of binary ladder (R–2' ladder, at point 'P'). The clock–input block produces square waves. They are connected to the gate–control block. When output of comparator is HIGH (logic–1), it connects the clock pulses to the input of counter. However, when its output is LOW (logic–0), it cuts off the clock pulses, so that counting stops. The number of clock pulses connected depends on the value of analog voltage.

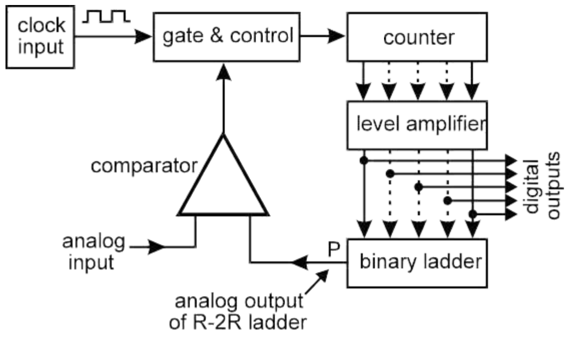


Fig 5.3.2 Simultaneous A/D Converter

The higher is the analog voltage the greater is the number of clock pulses counted by the counter. Then counter produces proportional binary output. Its output is fed to the level amplifier. It amplifies the output voltage of counter proportionally. This output is then fed to binary ladder. The binary ladder produces proportional analog voltage at point ‘P’.
Now when then circuit is switched on, initially the counter is RESET to 0000. Suppose we connect a known value of analog voltage = 9V. Therefore, there will be a difference in input voltages of comparator. Hence, output of comparator will be HIGH (logic–1). The counter will start counting the clock pulses and will produce equivalent binary number. Its output will sequentially change from 0001, 0010, 0011 and so on. When the output is 1001, which is = 9V, both inputs of comparator will be at same voltage. Therefore, its output will be LOW (logic–0). So the gate will be cutoff and the counter will stop counting. The final digital output will be 1001 = 9V. In this way, any analog voltage can be converted into its equivalent digitaloutput.

Successive Approximation Method – the basic drawback of counter method (given above) is that it has longer conversion time. Because it always starts from 0000 at every measurement, until the analog voltage is matched. This drawback is removed in successive approximation method. In the adjacent figure, the method of successive approximation technique is shown. When unknown voltage (Va) is applied, the circuit starts up from 0000, as shown above. The output of SAR advances with each MSB.

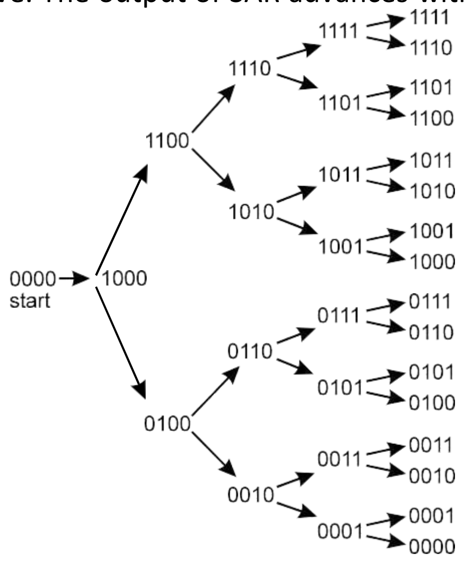


Fig 5.3.3 SAR Approximation method

The output of SAR does not increase step–by–step in BCD bus pattern, but individual bit becomes high–starting from MSB. Then by comparison, the bit is fixed or removed. Thus, it sets first MSB (1000), then the second MSB (0100) and so on. Every time, the output of SAR is converted to equivalent analog voltage by binary ladder. It is then compared with applied unknown voltage (Va). The comparison process goes on, in binary search style, until the binary equivalent of analog voltage is obtained. In this way following steps are carried out during conversion

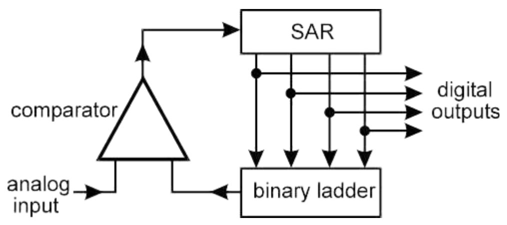


Fig 5.3.4 SAR Type A/D Converter

1) The unknown analog voltage (Va) is applied.
2) Starts up from 0000 and sets up first MSB1000.
3) If Va> 1000, the first MSB is fixed.
4) If Va< 1000, the first MSB is removed and second MSB is set.
5) The fixing and removing the MSBs continues upto last bit (LSB), until equivalent binary output is obtained. The block diagram of successive approximation A/D converter is given below–

Digital to Analog Converters:

Digital to Analog Converters (DAC) The process of converting digital signal into equivalent analog signal is called D/A conversion. The electronics circuit, which does this process, is called D/A converter. The circuit has 'n' number of digital data inputs with only one output. Basically, there are two types of D/A converter circuits:

Weighted resistors D/A converter circuit and Binary ladder or R–2R ladder D/A converter circuit.

Digital to Analog Converter using the Summing Amplifier

The following diagram shows a 3 bit digital to analog converter implemented using a summing opamp amplifer.

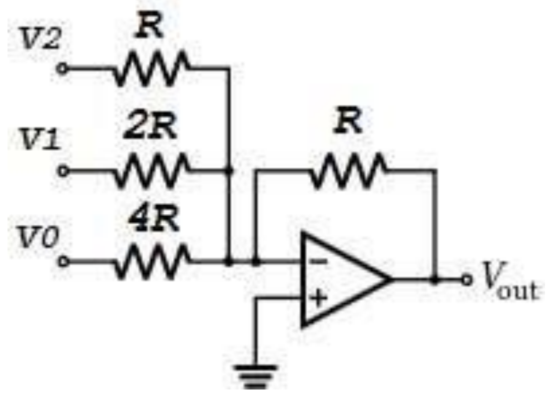


Fig 5.3.5 Weighted Position D/A Converter

From the summing amplifier circuit, the output voltage is if V1 be input voltage at MSB (most significant bit), V2 be input voltage at next bit and so on then for four bit DAC we can write,

(V1/R+V2/2R+V3/4R)=Vout/R ------(1)

Note: Here V_1, V_2, V_3, V_4, will be Vref if digital input is 1 or otherwise it will be zero.

Hence output voltage can be found as:

However Binary weighted DAC doesn't work for multiple or higher bit systems as the value of resistance doubles in each case.

From above we can conclude the following

- The inputs can be thought of as a binary number, one that can run from 0 to 7.
- V2 is the MSB (most significant bit) and V0 is the LSB (least significant bit).
- The output is a voltage that is proportional to the binary number input.
- The resolution of this DAC is 3 (the number of bits) or -0.25V (the step size).

- To have more bits, add an additional resistor for each additional bit. Note the relationship between adjacent resistor values.

R-2R Binary Ladder Digital to Analog Converter

The R-2R Digital to Analog Converter uses only two resistance values R and 2R regardless of the number of bits of the converter compared to the summing amplifier implementation where each bit resistor has a different value. The circuit shown is a 3 bit DAC.

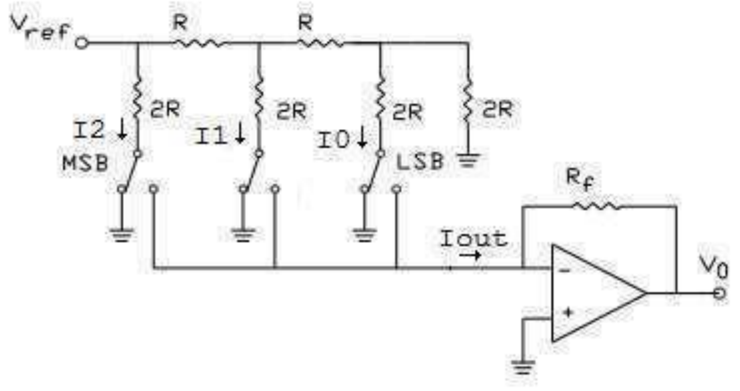


Fig 5.3.5 R-2R Ladder Type D/A Converter

Depending on the state of bit B2, B1 or B0, the respective current I2, I1 or I0 is switched either to ground or to V- of the op amp. Thus

Iout=B2*I2+B1*I1+B0* -------(1)

Thus the R-2R network can be seen to be like a current source whose output depends on switch setting B2, B1, B0 that controls I2, I1, I0 respectively

Iout=Vref/8R (4B2+2B1+B0) -------(2)

5.4 Display devices: Display devices are the output devices for presentation of information in text or image form. An output device is a thing that provides a way to show information to the outside world. For displaying the information in an appropriate manner these devices must be controlled by some other external devices. Controlling can be done by interfacing these displays with the controlling devices. Some displays can show digits and alphanumeric characters only. Some displays can show images and all type of characters. Most commonly used displays are CRT, LEDs, 7-segment display, LCD, GLCD etc.

7 and 16 segment LED display:

7-Segment LED Display:

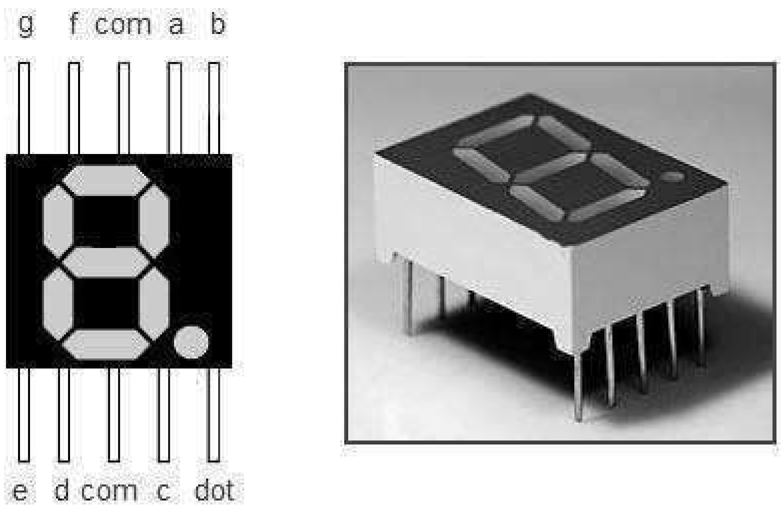
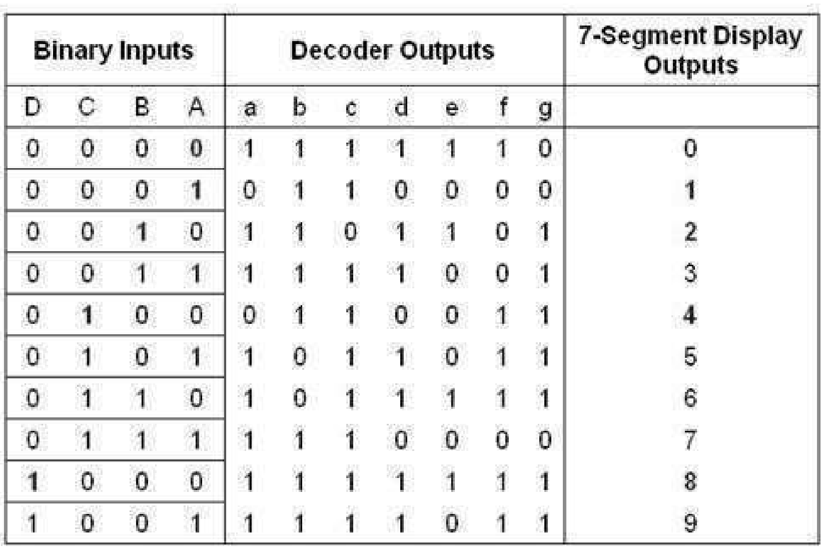


Fig 5.4.1 7-segment LED Display

7-Segment LED display can be used for displaying digits and few characters. A seven segment display consists of 7 LEDs arranged in the form of Square ‘8’ and a single LED as dot character. Different characters can be displayed by selecting the required LED segments. A 7 seven segment display is an electronic display, which displays 0-9 digital information. They are available in common cathode mode and common anode mode. There are state lines in LED, anode is given to positive terminal and cathode is given to negative terminal then LED will glow.

In common cathode, the negative terminals of all LEDs are connected to the common pins to ground and a particular LED glows when its corresponding pin is given high. The cathodes of all LEDs are connected together to a single terminal and the anodes of all LEDs are left alone.

In common anode arrangement, the common pin is given a high logic and the LED pins are given low to display a number. In common anode, all the anodes are connected together and all the cathodes are left alone. Thus when we gives first signal is high or 1 then only there is a lean in display if not there is no lean in display.LED pattern for displaying digits using 7-segment display.

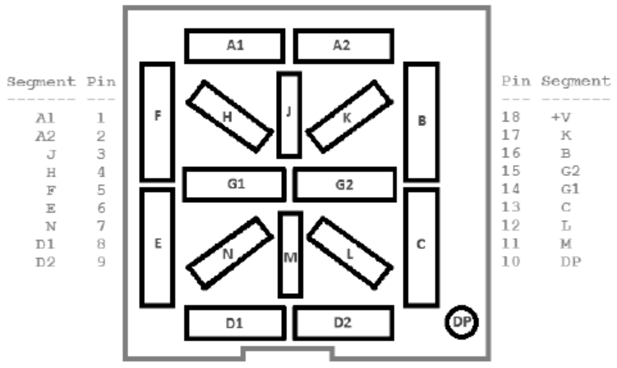


Binary Inputs				Decoder Outputs							7-Segment Display Outputs
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

Table 5.4.1 Table LED pattern for displaying digits using 7-segment display

16-Segment LED Display:

Sixteen-segment displays were originally designed to display alphanumeric characters (Latin letters and Arabic digits). Later they were used to display Thai numerals and Persian characters.Non-electronic displays using this pattern existed as early as 1902.



Segment	Pin
A1	1
A2	2
J	3
H	4
F	5
E	6
N	7
D1	8
D2	9

Pin	Segment
18	+V
17	K
16	B
15	G2
14	G1
13	C
12	L
11	M
10	DP

Fig **5.4.2 16-segment LED Display**

Liquid Crystal Display (LCD):

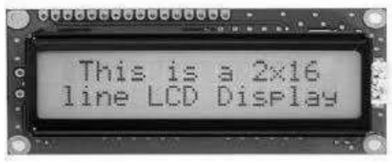
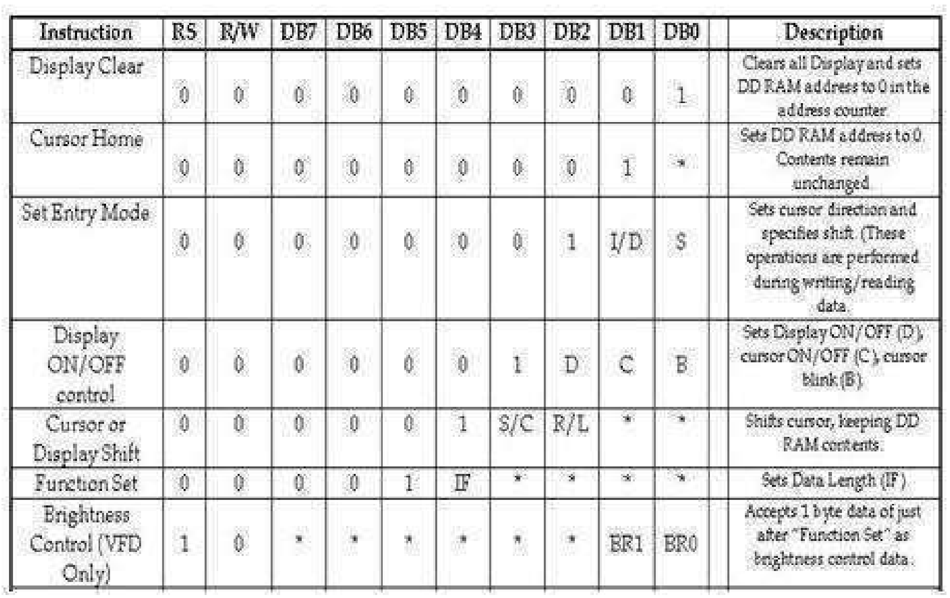


Fig 5.4.3 LCD Display

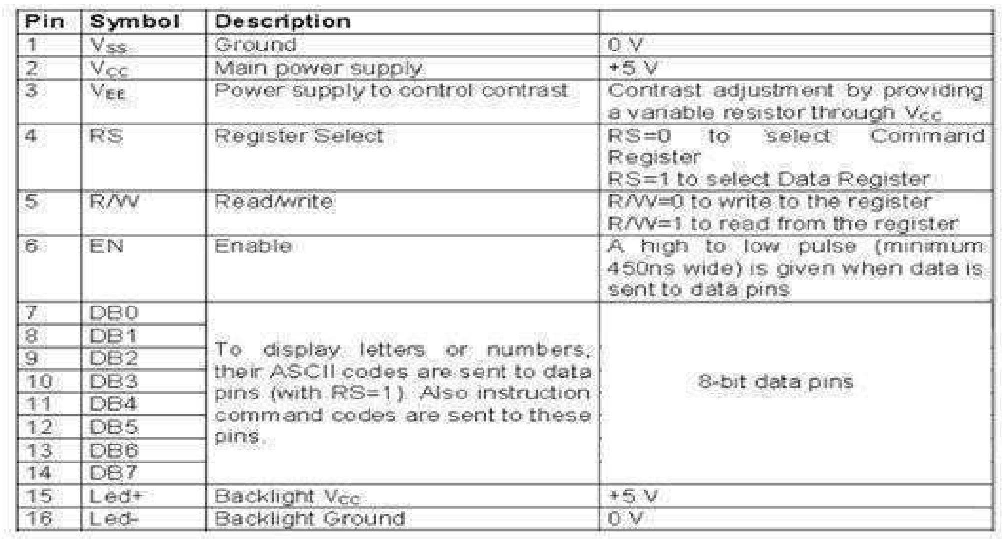
Liquid crystal display (LCD) has material which joins together the properties of both liquid and crystals. They have a temperature range within which the particles are essentially as mobile as they might be in a liquid, however are gathered together in an order form similar to a crystal.

The LCD is much more informative output device than a single LED. The LCD is a display that can easily show characters on its screen. They have a couple of lines to large displays. Some LCDs are specially designed for specific applications to display graphic images. 16×2 LCD (HD44780) module is commonly used. These modules are replacing 7-segments and other multi-segment LEDs. LCD can be easily interfaced with microcontroller to display a message or status of the device. It can be operated in two modes: 4-bit mode and 8-bit mode. This LCD has two registers namely command register and data register. It is having three selection lines and 8 data lines. By connecting the three selection lines and data lines with the microcontroller, the messages can be displayed on LCD.



Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		Description
Display Clear	0	0	0	0	0	0	0	0	0	1		Clears all Display and sets DD RAM address to 0 in the address counter
Cursor Home	0	0	0	0	0	0	0	0	1	*		Sets DD RAM address to 0. Contents remain unchanged
Set Entry Mode	0	0	0	0	0	0	0	1	I/D	S		Sets cursor direction and specifies shift. (These operations are performed during writing/reading data.
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B		Sets Display ON/OFF (D), cursor ON/OFF (C), cursor blink (B).
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*		Shifts cursor, keeping DD RAM contents.
Function Set	0	0	0	0	1	IF	*	*	*	*		Sets Data Length (IF)
Brightness Control (VFD Only)	1	0	*	*	*	*	*	*	BR1	BR0		Accepts 1 byte data of just after "Function Set" as brightness control data.

Table 5.4.2 Instruction/Commands for LCD operation



Pin	Symbol	Description	
1	V_{SS}	Ground	0 V
2	V_{CC}	Main power supply	+5 V
3	V_{EE}	Power supply to control contrast	Contrast adjustment by providing a variable resistor through V_{CC}

4	RS	Register Select	RS=0 to select Command Register
			RS=1 to select Data Register

5	R/W	Read/write	R/W=0 to write to the register R/W=1 to read from the register
6	EN	Enable	A high to low pulse (minimum 450ns wide) is given when data is sent to data pins
7	DB0	To display letters or numbers, their ASCII codes are sent to data pins (with RS=1). Also instruction command codes are sent to these pins.	8-bit data pins
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	Led+	Backlight V_{CC}	+5 V
16	Led-	Backlight Ground	0 V

Table 5.4.3 Pin Connections for LCD operation

In the above Table selected lines EN, R/W, RS will be used for controlling the LCD display. EN pin will be used for enabling the LCD display for communicating with microcontroller. RS will be used for register selection.

When RS is set microcontroller will send instructions as data and when RS is clear microcontroller will send the instructions as commands. For writing data RW should be 0 and for reading RW should be 1.

OTHER TYPES OF DISPLAYS:

Plasma Displays

- Plasma: Cloud of ionized particles
  These produce light when electrons change energy levels
  Electrodes in front and back of sealed chambers with neon gas and mercury vapors
  Not suitable in high altitude
  No backlight, produce deeper black
  600Hz is refresh rate
- Organic Light Emitting Diode (OLED)
  Organic light emitting compound placed b/w anode-cathode emits light when excited by current
  No backlight, deeper black
  Thin, light weight, flexible
  Better contrast ratio
  Reflective thus filters required
  Active matrix (AMOLED)
  Passive matrix (PMOLED)
  Lesser power than lcds
  Super AMOLED use one cell TSP thus thinner touch layer over the display
  Super AMOLED plus has sub pixels for clearer display

-----------------------------END OF UNIT---------------------------