RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL
New Scheme Based On AICTE Flexible Curricula
Computer Science and Engineering, III-Semester

CS 303 Digital Systems

Unit 1: Review of number systems and number base conversions. Binary codes, Boolean algebra, Boolean functions, Logic gates. Simplification of Boolean functions, Karnaugh map methods, SOP-POS simplification, NAND-NOR implementation.

Unit 2: Combinational Logic: Half adder, Half subtractor, Full adder, Full subtractor, look-ahead carry generator,BCD adder, Series and parallel addition, Multiplexer – demultiplexer, encoder- decoder, arithmetic circuits, ALU

Unit 3 : Sequential logic: flip flops, D,T, S-R, J-K Master- Slave, racing condition, Edge & Level triggered circuits, Shift registers, Asynchronous and synchronous counters, their types and state diagrams. Semiconductor memories, Introduction to digital ICs 2716, 2732 etc. & their address decoding. Modern trends in semiconductor memories such as DRAM, FLASH RAM etc. Designing with ROM and PLA.

Unit 4 : Introduction to A/D & D/A convertors & their types, sample and hold circuits, Voltage to Frequency & Frequency to Voltage conversion. Multivibrators :Bistable, Monostable, Astable, Schmitt trigger, IC 555 & Its applications. TTL, PMOS, CMOS and NMOS logic. Interfacing between TTL to MOS.

Unit 5 : Introduction to Digital Communication: Nyquist sampling theorem, time division multiplexing, PCM, quantization error, introduction to BPSK & BFSK modulation schemes. Shannon's theorem for channel capacity.

References:

1. Morris Mano, Digital Circuits & Logic Design, PHI
2. Gothman, Digital Electronics, PHI
3. Tocci, Digital Electronics, PHI
4. Mavino& Leach, Digital Principles & Applications, PHI
5. Taub and schilling, Digital Integrated electronics.
6. Simon Haykin, Introductionto Analog& Digital Communication, Wiley.
7. Lathi B.P., Modern analog& digital communication , Oxford University.

Subject Name: Digital Systems **Subject Code**: CS303

Subject Notes

UNIT-I

Review of number systems and number base conversions. Binary codes, Boolean algebra, Boolean functions, Logic gates. Simplification of Boolean functions, Karnaugh map methods, SOP-POS simplification, NAND-NOR implementation

1.1 NUMBER SYSTEMS:

BINARY NUMBER SYSTEM : - This number system has a base or radix of 2. The symbols or digits used in this system are 0 & 1.

OCTAL NUMBER SYSTEM : - This number system has a base or radix of 8. The symbols or digits used 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 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 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) BINARY TO DECIMAL CONVERSION:-

$(1111 . 11010)_2 = (?)_{10}$

Integeral part : $(1111)_2 = (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$= 8 + 4 + 2 + 1$

$= 15$

ie. $(1111)_2 = (15)_{10}$

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 $(1111.11010)_2 = (15.8125)_{10}$

2) DECIMAL TO BINARY CONVERSION :-

$(15 . 812)_{10} = (?)_2$

Integeral part :

2	15	1
2	7	1
2	3	1
	1	

ie. $(15)_{10} = (1111)_2$

Fractional Part :

$(0.812 \times 2) = 1.624 \rightarrow 1$

$(0.624 \times 2) = 1.248 \rightarrow 1$

$(0.248 \times 2) = 0.496 \rightarrow 0$

$(0.496 \times 2) = 0.992 \rightarrow 0$

$(0.992 \times 2) = 1.984 \rightarrow 1$

ie. $(0.812)_{10} = (0.11001)_2$

Thus $(15.812)_{10} = (1111.11001)_2$

3) OCTAL TO DECIMAL CONVERSION:-

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

<u>Integeral part</u> :

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

$= 40 + 7$

$= 47$

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

<u>Fractional Part</u> :

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

$= 0.25 + 0.0625 + 0.0097$

$= 0. 3222$

ie. $(0.245)_8 = (0. 3222)_{10}$

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

4) DECIMAL TO OCTAL CONVERSION :-

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

<u>Integeral part</u> :

8	303	7
8	37	5
	4	

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

<u>Fractional Part</u> :

$(0.322 \times 8)= 2.576 \rightarrow 2$

$(0.576 \times 8)= 4.608 \rightarrow 4$

$(0.608 \times 8)= 4.864 \rightarrow 4$

$(0.864 \times 8) = 6.912 \rightarrow 6$

$(0.912 \times 8)= 7.296 \rightarrow 7$

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

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

5) HEXADECIMAL TO DECIMAL CONVERSION :-

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

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

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

$= 3584 + 160 + 6$

$= 3750$

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

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

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

$= 0.125 + 0.0586 + 0.00244$

$= 0. 18604$

ie. $(0.2FA)_{16} = (0. 18604)_{10}$

Thus $(EA6 . 2FA)_8 = (3750 . 18604)_{10}$

6) DECIMAL TO HEXADECIMAL CONVERSION:

$(3750 . 365)_{10} = (?)_{16}$

Integeral part :

16	3750	6	→ 6
16	234	10	→ A
	14		→ E

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

Fractional Part :

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

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

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

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

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

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

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

7) BINARY TO HEXADECIMAL CONVERSION:

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

Integeral part :

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

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

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

RGPVNOTES.IN

Fractional Part :

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

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

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

8) HEXADECIMAL TO BINARY CONVERSION:

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

Integeral part :

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

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

Fractional Part :

$(0.2FA)_{16} = \{\underbrace{2}_{0010}\ \underbrace{F}_{1111}\ \underbrace{A}_{1010}\} = (0.001011111010)_2$

Thus $(99E.2FA)_{16} = (1001111010100110.001011111010)_2$

9) OCTAL TO BINARY CONVERSION:

$(404 . 245)_8 = (?)_2$

<u>Integeral part</u> :

$(404)_8 = \underbrace{4}\ \underbrace{0}\ \underbrace{4}$

$\{ 100 \quad 000 \quad 100 \} = (100000100)_2$

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

<u>Fractional Part</u> :

$(0 . 245)_8 = \underbrace{2}\ \underbrace{4}\ \underbrace{5}$

$\{ 010 \quad 100 \quad 101 \} = (0. 010100101)_2$

Thus $(404 . 245)_8 = (100000100 . 010100101)_2$

10) BINARY TO OCTAL CONVERSION:

$(10011110 . 00101)_2 = (?)_8$

Integeral part :

$(10011110)_2 = \{ 010, \quad 011, \quad 110 \}$

$= (\underbrace{010}_{2}, \quad \underbrace{011}_{3}, \quad \underbrace{110}_{6})_2$

ie. $(10011110)_2 = (236)_8$

Fractional Part :

$(0. 00101)_2 = \{ \underbrace{001}_{1}, \quad \underbrace{010}_{2} \}$

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

Thus $(10011110 . 00101)_2 = (236 . 12)_8$

11) OCTAL TO HEXADECIMAL CONVERSION:

$(174654 . 273054)_8 = (?)_{16}$

<u>Integeral part</u> :

$(174654)_8 = \underbrace{1}\ \underbrace{7}\ \underbrace{4}\ \underbrace{6}\ \underbrace{5}\ \underbrace{4}$

$\{ 001 \quad 111 \quad 100 \quad 110 \quad 101 \quad 100 \}$

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

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

<u>Fractional Part</u> : $\underbrace{2}\ \underbrace{7}\ \underbrace{3}\ \underbrace{0}\ \underbrace{5}\ \underbrace{4}$

$(0.273054)_8 =$ 010 111 011 000 101 100

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

ie. $(0.273054)_8 = (0.5D8B0)_{16}$

Thus $(174654.273054)_8 = (F9AC.5D8B)_{16}$

12) HEXADECIMAL TO OCTAL CONVERSION:

$(F9AC.5D8B)_{16} = (?)_8$

<u>Integeral part</u> :

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

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

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

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

<u>Fractional Part</u> :

$(0.5D8B)_{16} = \{\underbrace{0101}_{5}\ \underbrace{1101}_{D}\ \underbrace{1000}_{8}\ \underbrace{1011}_{B}\} = (0.010, 111, 011, 000, 101, 100)_2$

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

$= (0.273054)_8$

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

Thus $(F9AC.5D8B)_{16} = (174654.273054)_8$

1.2 CODES

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

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.

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 ⟶ (8421) BCD ⟶ ADD 3 i.e.(+0011) ⟶ 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.

1.3 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

1.4 BINARY ARITHMETIC:

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

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 $(+127)_{10}$.

1) 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**

2) 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**

3) 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.

4) 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).**

5) 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**

6) BINARY DIVISION:

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

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

1.5 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)

PGDVNOTES.IN

1.6 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.

1.7 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'

1.8 APPLICATION OF BOOLEAN ALGEBRA :

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

DCPU NOTES.IN

(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'

1.9 TECHNIQUES TO MINIMIZE THE BOOLEAN FUCTION:

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

1) <u>The Karnaugh map method (K-Map)</u>

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.

2) 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 X Y Z	Term	Designation
0	0 0 0	X'.Y'.Z'	m_0

1	0 0 1	X'.Y'.Z	m_1
2	0 1 0	X'.Y.Z'	m_2
3	0 1 1	X'.Y.Z	m_3
4	1 0 0	X.Y'.Z'	m_4
5	1 0 1	X.Y'.Z	m_5
6	1 1 0	X.Y.Z'	m_6
7	1 1 1	X.Y.Z	m_7

Example: Boolean expression in SOP form:
F = X'.Y'.Z' + X'.Y'.Z + X'.Y.Z + X.Y'.Z + X.Y.Z
F(A,B,C,D) = ∑m (0, 1, 3, 5, 7)

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

RGPVNOTES.IN

Example: Boolean expression in POS form:
F = (X'+Y+Z') (X'+Y'+Z) (X'+Y+Z) (X+Y'+Z) (X+Y+Z)
F(A,B,C,D) = ∏M (5,6,4,2,0)

1.10 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

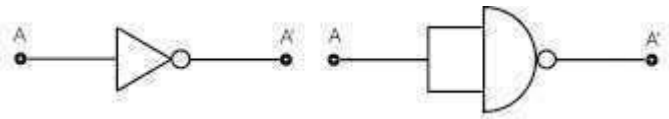
The minimized expressions are as follows: Y1 = B'.C' + A'.C' + A.B.C and Y2 = A + B'

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)

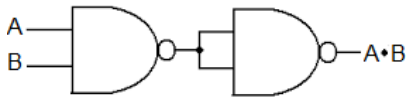
	C'.D'	C'.D	C.D	C.D'
A'.B'	1	**0**	1	1
A'.B	**0**	**0**	1	**0**
A.B	**0**	**0**	**0**	**0**
A.B'	1	1	**0**	1

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

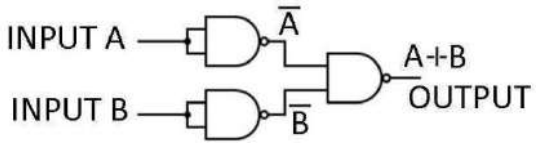
1.11 NAND-NOR Implementation:
NAND and NOR gate are called the universal gates and all the basic gates i.e. AND, OR, NOT and EX-OR can be implemented using these gates.
1.11.1 NAND Implementation:
(a) Implementing NOT gate using NAND Gate:



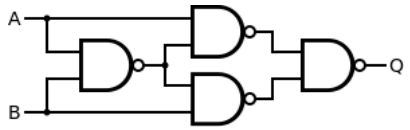
(a) Implementing AND gate using NAND Gate:



(c) Implementing OR gate using NAND Gate:

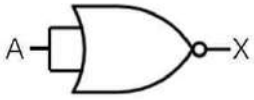


(d)) Implementing EX-OR gate using NAND Gate:

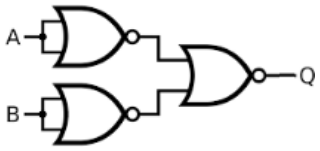


1.11.2 NOR Implementation:

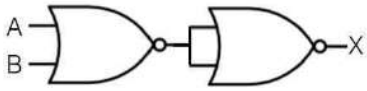
(a) Implementing NOT gate using NOR Gate:



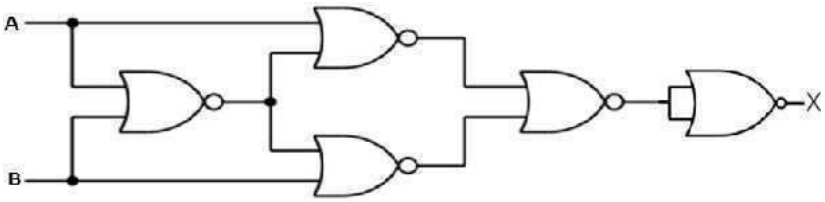
(a) Implementing AND gate using NOR Gate:



(c) Implementing OR gate using NOR Gate:

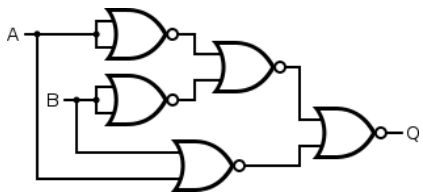


(d)) Implementing EX-OR gate using NOR Gate:



$$X = [(A + (A + B)')' + (B + (A + B)')']'$$
$$X = [A'B + AB']$$

Or



$$X = [(A' + B')' + (A + B)']'$$
$$X = [(A.B) + (A + B)']'$$
$$X = [(AB)'.\{(A + B)'\}']$$
$$X = [(A' + B').(A + B)]$$
$$X = [A'A + A'B + B'A + B'B]$$
$$X = [A'B + AB']$$

----------End of Unit 1----------

RGPVNOTES.IN

Subject Notes

UNIT-II

Combinational Logic: Half adder, Half subtractor, Full adder, Full subtractor, look- ahead carry generator,BCD adder, Series and parallel addition, Multiplexer – demultiplexer, encoder-decoder, arithmetic circuits, ALU

2.1 COMBINATIONAL LOGIC:

A combinational logic 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. The design of 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.
b) The number of input and required output variables is determined.
c) Truth table is derived.
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, BCD adder, Series and parallel adder, BCD adders, Look-ahead carry generator.

2.2 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 :

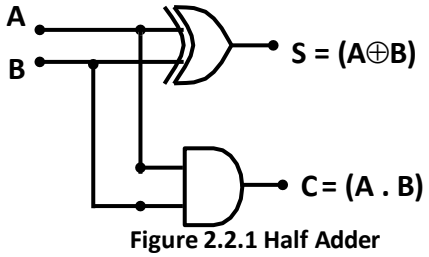


Figure 2.2.1 Half Adder

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

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

$= Y.C_{IN}(X + X') + X.Y.C_{IN}' + X.Y'.C_{IN}$

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

$= Y(C_{IN} + C_{IN}'.X) + X.Y'.C_{IN}$

$= Y(C_{IN} + X) + X.Y'.C_{IN}$

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

$= C_{IN}(Y + X.Y') + Y.X$

$= C_{IN}(Y + X) + Y.X$

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

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

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

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

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

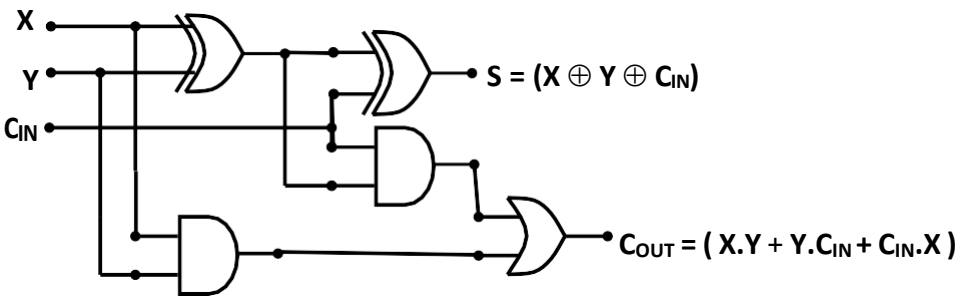


Figure 2.3.1 Full Adder

2.4 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

Simplified Boolean function for the outputs are derived using K-map:

K-Map for DIFFERENCE(D):

	B'	B
A'	0	1
A	1	0

D = A'.B + A.B'

K-Map for BORROW(BOR_{out}):

	B'	B
A'	0	1
A	0	0

BOR_{OUT} = A'.B

LOGIC DIAGRAM OF HALF SUBTRACTOR:

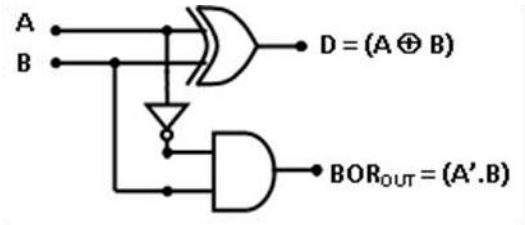


Figure 2.4.1 Half Subtractor

2.5 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 coloumn; 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

Simplified Boolean function for the outputs are derived using K-map:

K-Map for DIFFERENCE(D):

	$B'.BOR_{IN}'$	$B'.BOR_{IN}$	$B.BOR_{IN}$	$B.BOR_{IN}'$
A'	0	1	0	1
A	1	0	1	0

K-Map for BORROW(BOR_{out}):

	$B'.BOR_{IN}'$	$B'.BOR_{IN}$	$B.BOR_{IN}$	$B.BOR_{IN}'$
A'	0	1	1	1
A	0	0	1	0

$D = A'.B'.BOR_{IN} + A'.B.BOR_{IN}' + A.B'.BOR_{IN}' + A.B.BOR_{IN}$

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

RGPVNOTES.IN

LOGIC DIAGRAM OF FULL SUBTRACTOR:

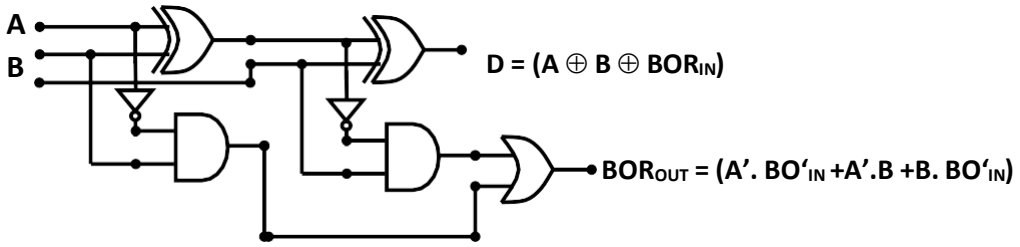


Figure 2.5.1 Full Subtractor

2.6 LOOK AHEAD CARRY GENERATOR:

Consider the full adder circuit,

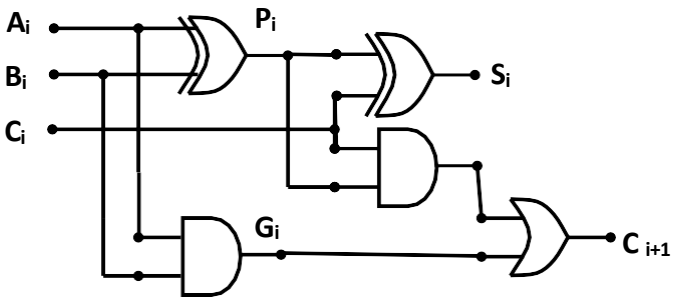


Figure 2.6.1 Full Adder

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:

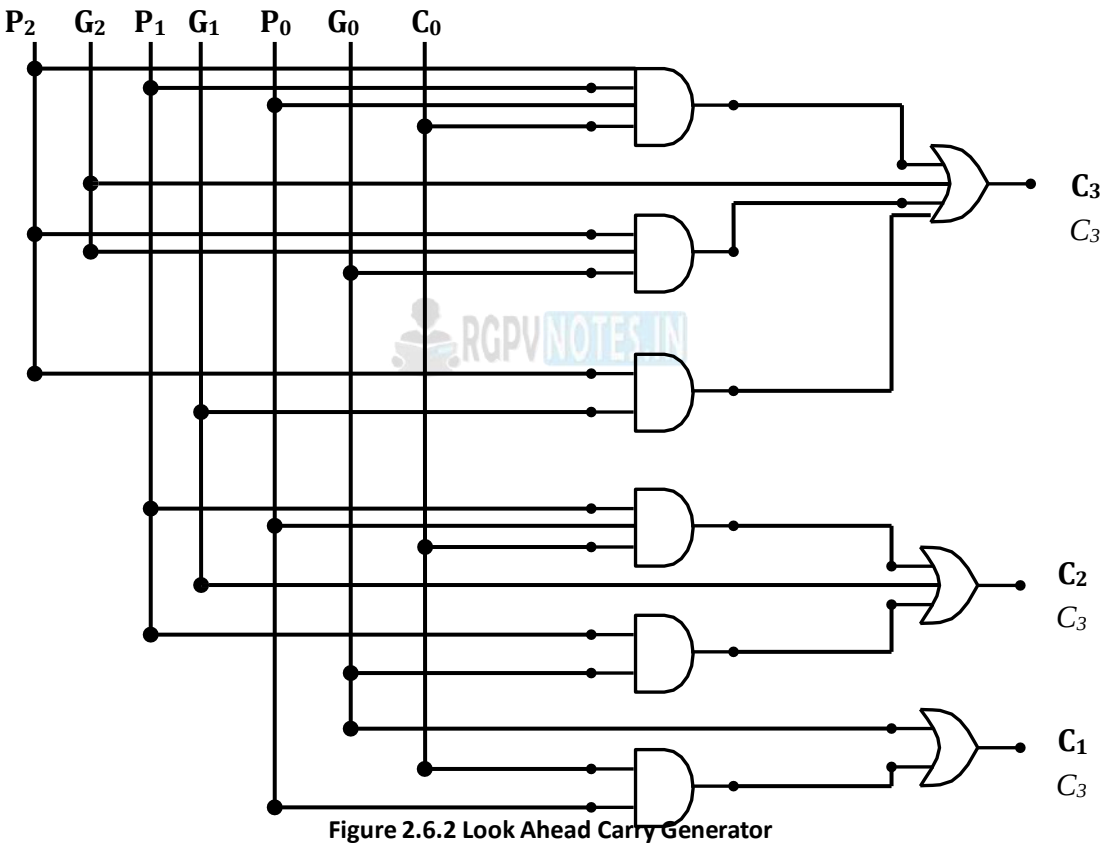


Figure 2.6.2 Look Ahead Carry Generator

2.7 BINARY CODED DECIMAL ADDER (BCD ADDER):

In BCD addition, 2-BCD digits are added in parallel, which produces a sum. If the sum is less than or equal to 9 , we get the valid BCD sum or if the sum is greater than 9 (non valid BCD sum) then we have to add 6(0110) to the sum to get the valid BCD sum. So, a logical circuit that performs the BCD addition is a BCD adder. A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit also in BCD.

Design of BCD adder:

In BCD, each input digit does not exceeds 9, so the output sum cannot be greater than 9+9+1=19, the 1 in sum being an input carry. Suppose, we apply 2-BCD digits to a 4-bit adder. The adder form the sum in binary an produce a result that may range from 0-19. These binary

numbers are listed in table below:

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

From the table, when binary sum is less than or equal to 9(1001), corresponding BCD number is identical or valid and no conversion is needed. When binary sum is greater than 9, we obtain a non-valid BCD representation. So to get a valid BCD representation, we add 6(0110) to the binary sum and also produces an output carry as required.

The condition for correction and output carry can be expressed by a Boolean function:

$$C = K + Z_8 . Z_4 + Z_8 . Z_2$$

Block diagram of BCD Adder:

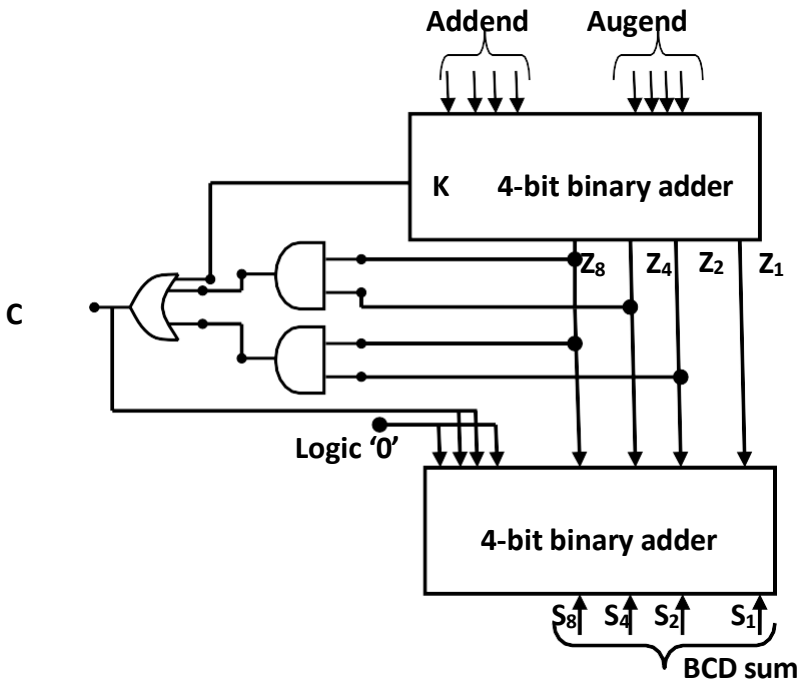


Figure 2.7.1 BCD Adder

From the block diagram of BCD adder, the binary numbers are labeled by symbols K,

Z8,Z4,Z2,Z1 where K is the carry, and the subscript under the letter Z represents the weight assigned to the bits. Letter C represent the output carry. The carry from the bottom binary adder can be ignored, since it supplies information already available at the output-carry terminal (C).

2.8 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.

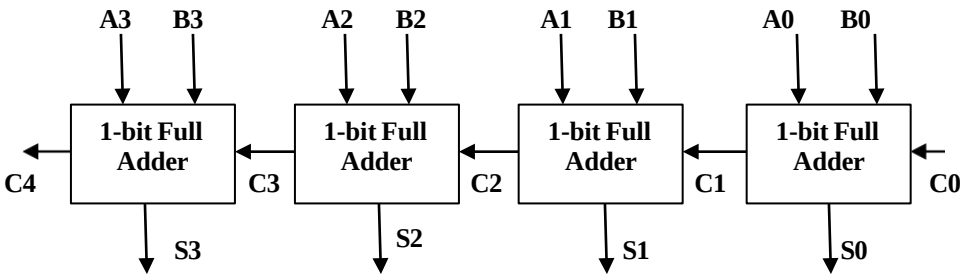


Figure 2.8.1 4-Bit Binary Parallel Adder

From figure 2.8.1, 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 2.8.1, 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.

2.9 MULTIPLEXER - DEMULTIPLEXER:

2.9.1 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:

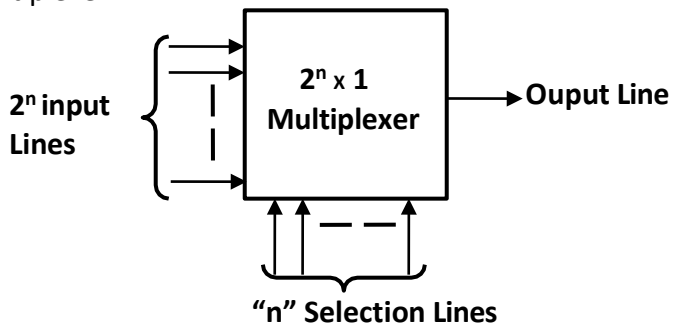


Figure 2.9.1.1 Multiplexer

4 x 1 Multiplexer:

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

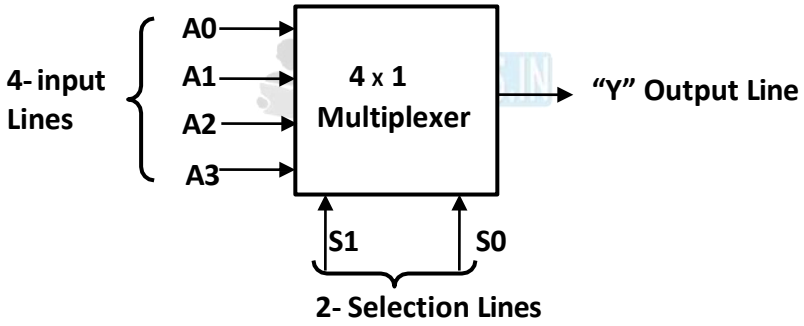


Figure 2.9.1.2 Block diagram of 4x1 Multiplexer

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

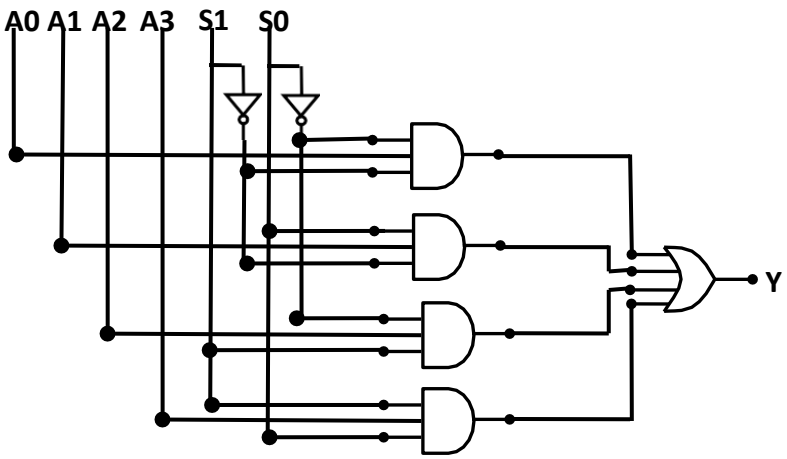


Figure 2.9.1.3 Logic diagram of 4x1 MUX

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

2.9.2 DEMULTIPLEXERS (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.

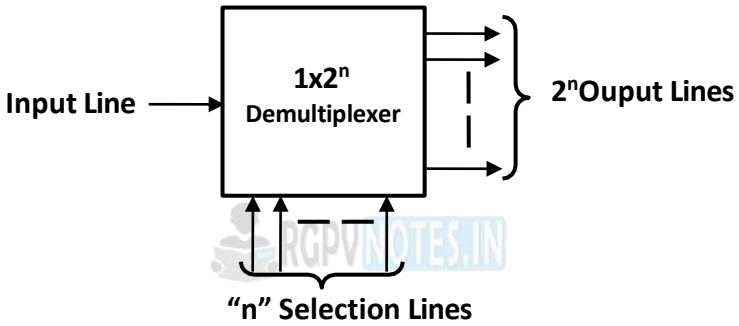


Figure 2.9.2.1 Block Diagram of Demultiplexer

1 x 4 Demultiplexer:

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

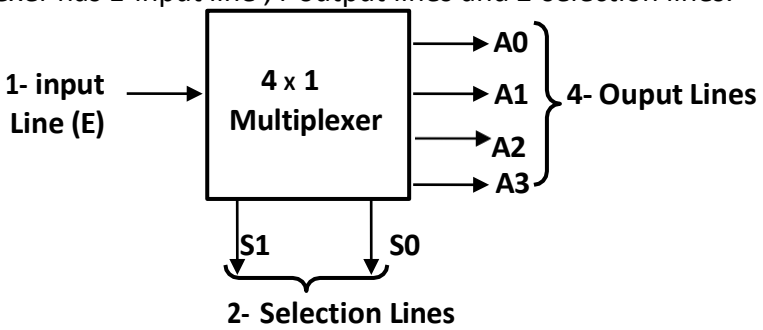


Figure 2.9.2.2 Block diagram of 1 x 4 Demultiplexer

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

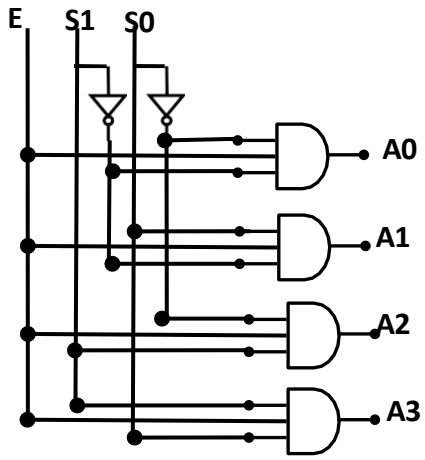


Figure 2.9.2.3 Logic diagram of 1 x 4 DEMUX

2.10 ENCODER DECODER:

2.10.1 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 "Z"is 1 for octal digits: 1,3,5,7. So, **$Z = D_1 + D_5 + D_3 + D_7$**

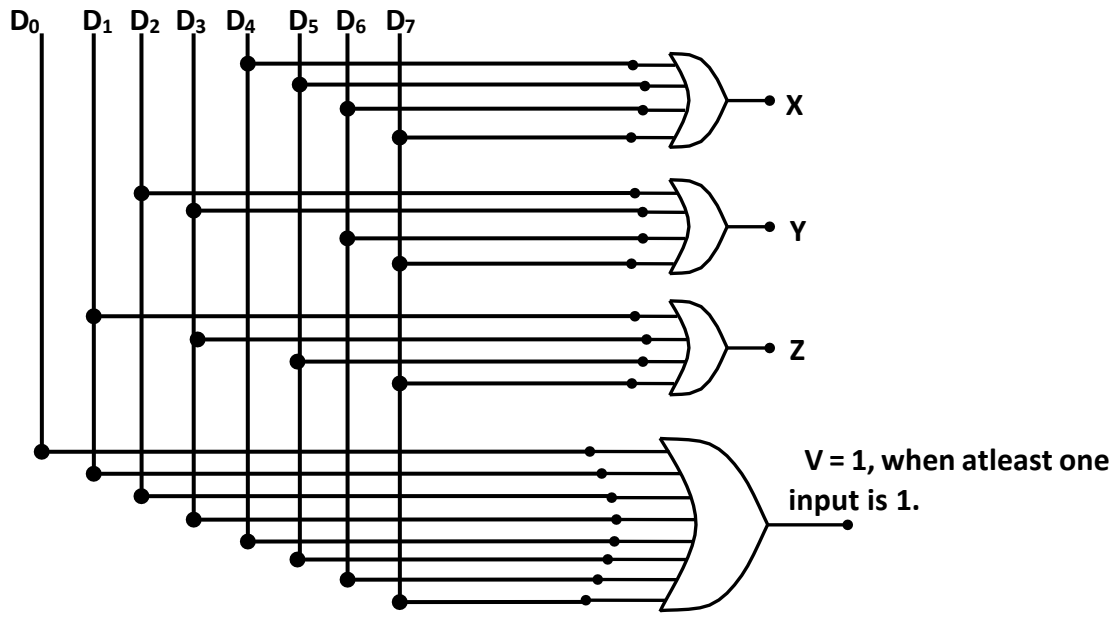


Figure 2.10.1.1:Logic diagram of 8 to 3 line encoder

The encoder in figure, 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.
Simillarly we can design priority encoder, decimal to BCD encoder, hexadecimal to binary encoder.

RGPVNOTES.IN

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:

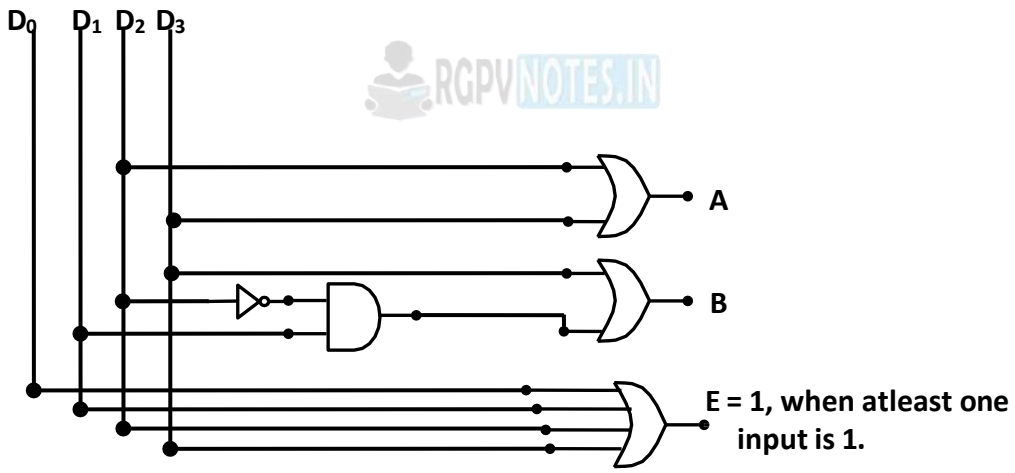


Figure 2.10.1.2: Logic diagram of 4 to 2 line priority

2.10.2 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.

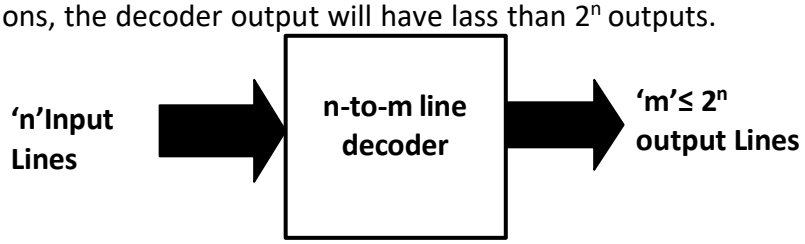


Figure 2.10.2.1 Block diagram of Decoder

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

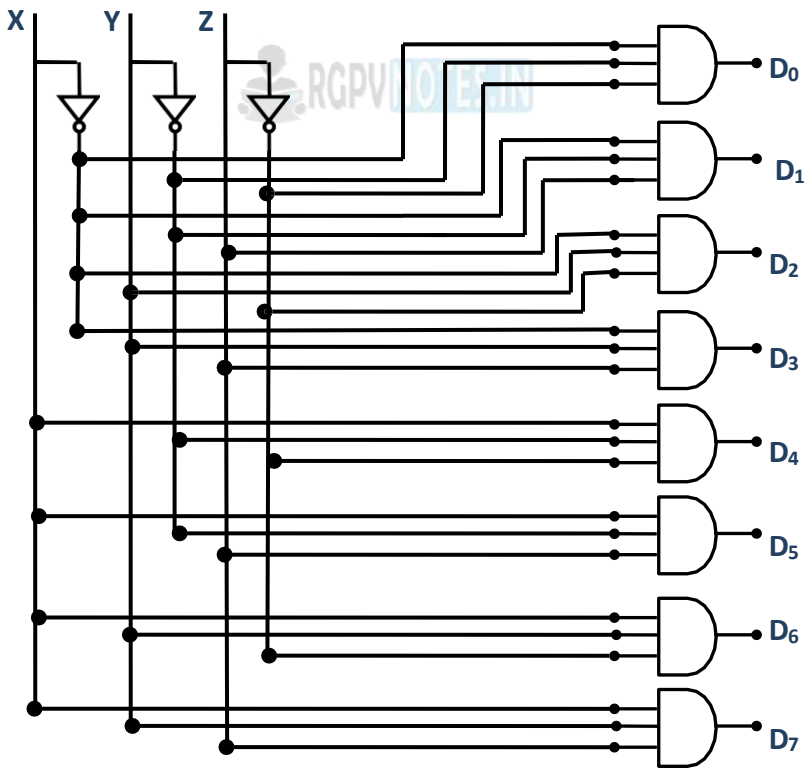


Figure 2.10.2.2: Logic diagram of 3 to 8 Line Decoder

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

2.11 ARITHMETIC CIRCUITS:

CODE CONVERTER:

1. Binary to Gray code converter:

Truth Table:

Inputs Binary				Outputs Gray			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

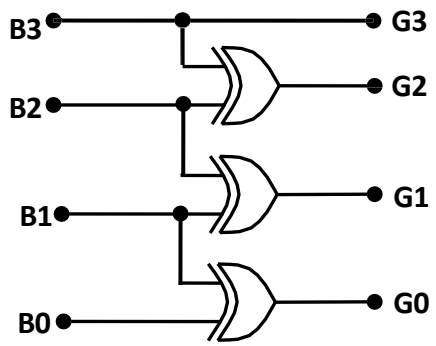


Figure 2.11.1.1: Binary to Gray Code Converter

Output expression is determine using K-map method, we get:

G3 = B3. **G1 = B2 ⊕ B1.**

G2 = B3 ⊕ B2. **G0 = B1 ⊕ B0.**

2. Gray to Binary code converter:

Truth Table:

Inputs Gray				Outputs Binary			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

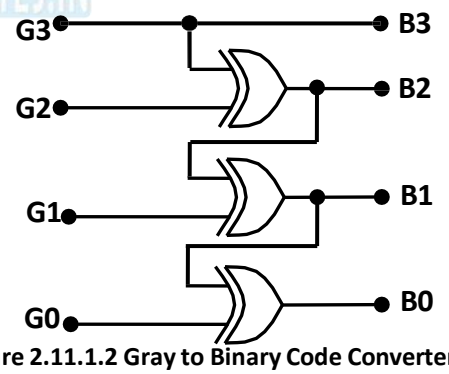


Figure 2.11.1.2 Gray to Binary Code Converter

Output expression is determine using K-map method, we get:

B3 = G3. **B1 = B2 ⊕ G1.**

B2 = G3 ⊕ G2. **B0 = B1 ⊕ G0.**

2.12 ARITHMETIC LOGIC UNIT (ALU)

ALU is a multioperation, combinational logic digital function. It can perform a set of basic arithmetic and a set of logical operations. ALU has a set of selection lines to select a particular operation. The selection lines are decoded within the ALU.

Figure 2.12.01 shows the block diagram of 4-bit ALU.

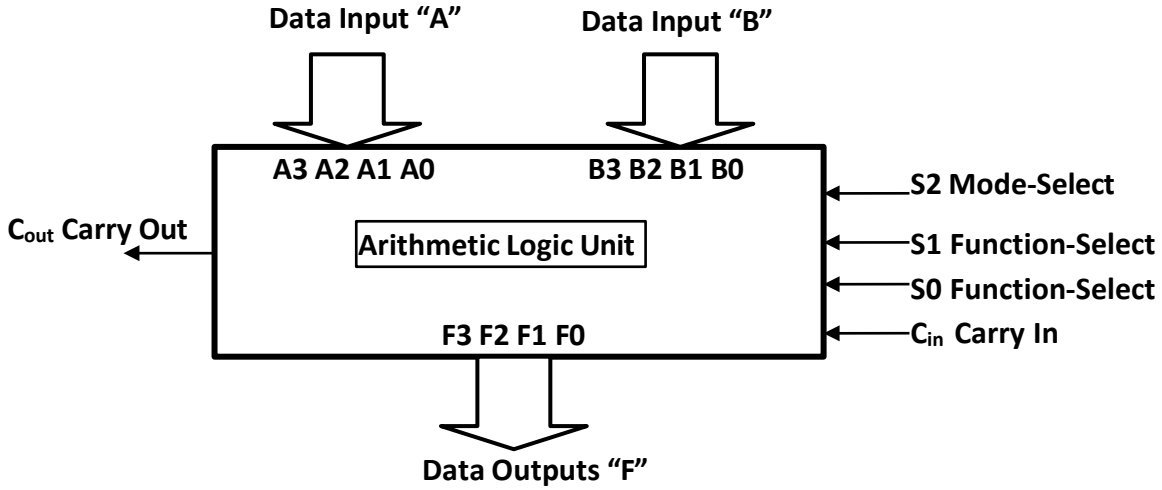


Figure 2.12.01 4 Bit Arithmatic & Logic Unit

Above block diagram, is of a 4-bit ALU. The 4-data inpus from register A are combined with 4-data inputs from register B to generate an operation at the F outputs. Mode select input S2 will specify whether the operations performed are arithmetic or logic. Two function-select S0 and S1 specify the particular arithmetic or logical operation to be generated. With 3-selection variables, it is possible to specify 4-arithmetic operations (with S2 in one state) and 4-logical operations (with S2 in another state). Input and output carries have meaning only during arithmetic operation.

DESIGN OF ALU:

RGPVNOTES.IN

Carried out in three stages: (i) Design of arithmetic section (ii) Design of logic section (iii) Finally the arithmetic section will be modified so that it can perform both arithmetic and logic operations.

DESIGN OF ARITHMETIC CIRCUIT:

The basic component of arithmetic section of ALU is parallel adder. Parallel adder is constructed with anumber of full adder circuits connected in cascade.
A 4-bit arithmetic circuit that performs eight arithmetic operations is shown in figure below.
The arithmetic operations implemented in the arithmetic circuit are listed in the below table.
The values to the Y inputs to the full adder circuits are a function of selection variables S1 and S0. Adding the value of Y to the value of A plus the C_{in} value gives the arithmetic operation.
The combinational circuit that inserted in each stage between external inputs A and B and the inputs of parallel adder X and Y, is a function of the arithmetic operations that are to be implemented. The combinational circuit will be different if the circuit generates different arithmetic operations.

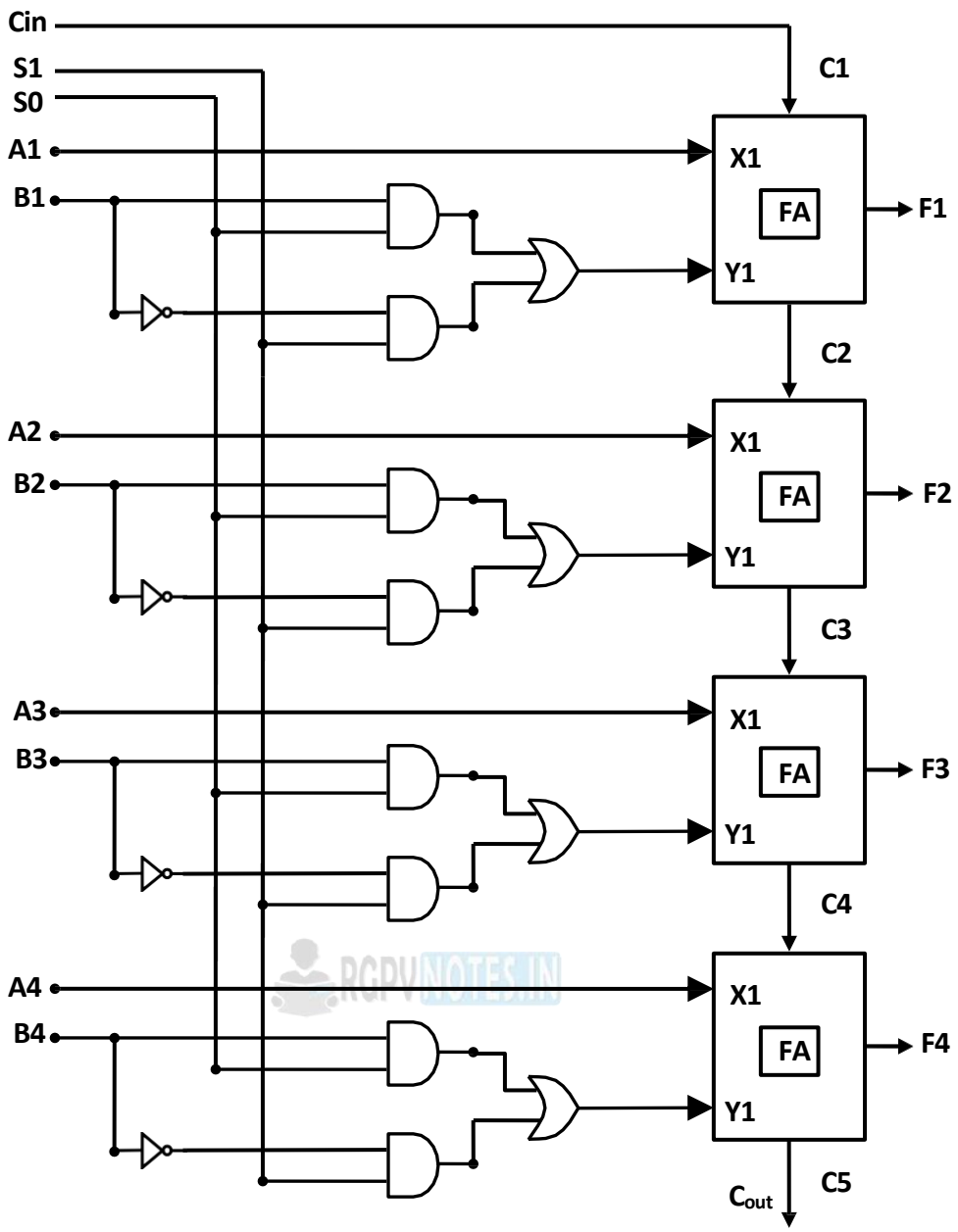


Figure 2.12.02: Logic Diagram of 4-Bit Arithmetic Circuit

Function Table for the arithmetic circuit:

Function Select			Y equals	Output equals	Function
S1	S0	C_{in}			
0	0	0	0	F=A	Transfer *A*
0	0	1	0	F= A+1	Increment *A*
0	1	0	B	F= A+B	Add *B* to *A*
0	1	1	B	F= A+B+1	Add *B* to *A* plus 1
1	0	0	B’	F= A+B’	Add 1’s complement of *B* to *A*
1	0	1	B’	F= A+B’+1	Add 2’s complement of *B* to *A*
1	1	0	All 1’s	F = A− 1	Decrement *A*
1	1	1	All 1’s	F =A	Transfer *A*

****End of Unit 2****

Subject Notes

UNIT-III

Sequential logic: flip flops, D,T, S-R, J-K Master- Slave, racing condition, Edge & Level triggered circuits, Shift registers, Asynchronous and synchronous counters, their types and state diagrams. Semiconductor memories, Introduction to digital ICs 2716, 2732 etc. & their address decoding, Modern trends in semiconductor memories such as DRAM, FLASH RAM etc. Designing with ROM and PLA.

3.1 SEQUELTIAL LOGIC:

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:

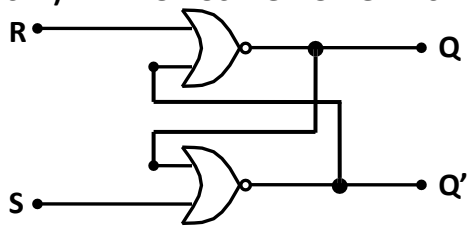


Figure 3.1.1 RS Flip Flop using NOR Gate

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:

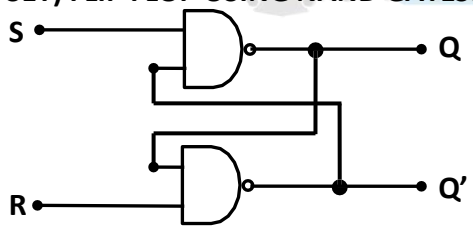


Figure 3.1.2 RS Flip Flop using NAND Gate

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:

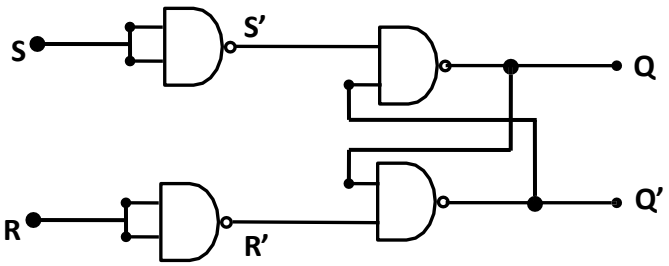


Figure 3.1.3 RS Flip Flop

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:

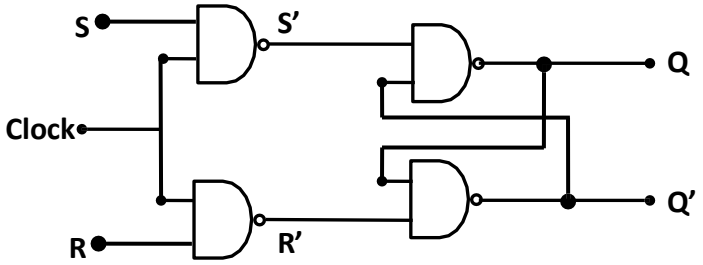


Figure 3.1.4 Clocked RS Flip Flop

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.

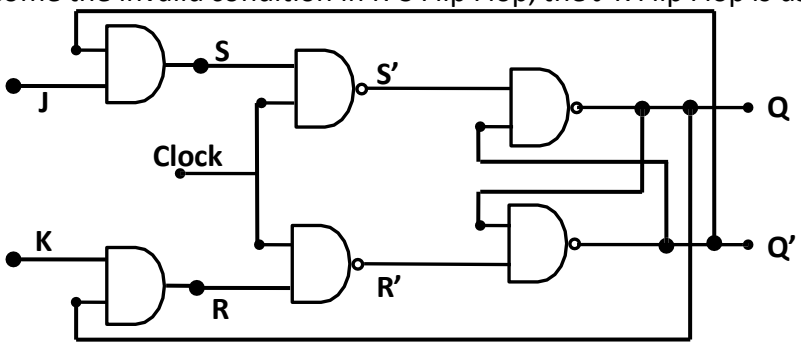


Figure 3.1.5 JK Flip Flop

Truth Table of J-K Flip Flop:

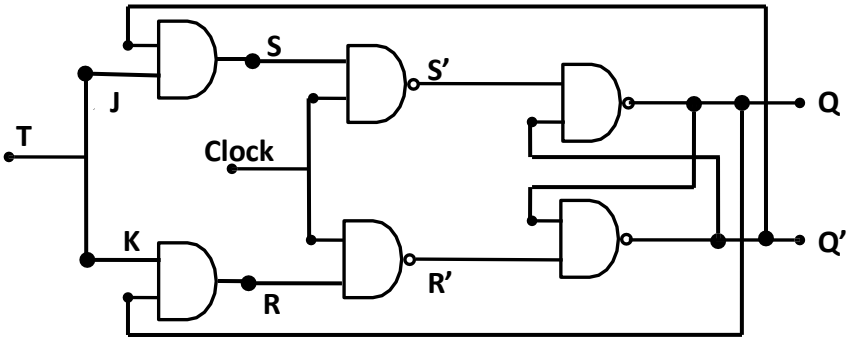
CLK	J	K	Qn	S	R	Q_{n+1} (output)	Remarks
1	0	0	0	0	0	0	Qn (No change)
1	0	0	1	0	0	1	
1	0	1	0	0	0	0	0 (RESET)
1	0	1	1	0	1	0	
1	1	0	0	1	0	1	1 (SET)
1	1	0	1	0	0	1	
1	1	1	0	1	0	1	Toggle or complement
1	1	1	1	0	1	0	

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 a problem in JK flip flop. To overcome the problem of race around condition, Master-Slave JK flip flop is used.

(e) 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.



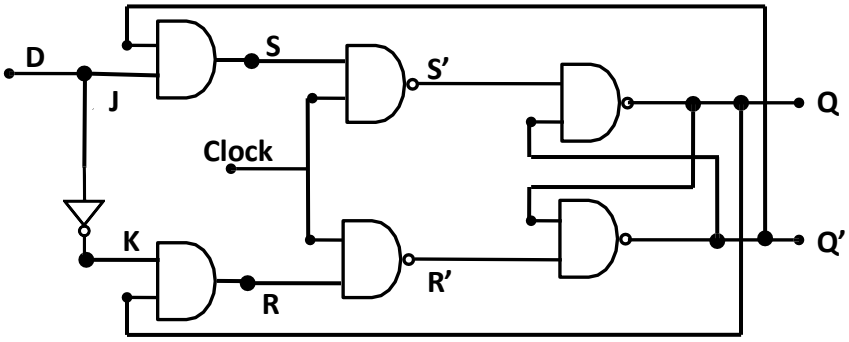
Truth Table:

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

Figure 3.1.6 JK Flip Flop

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 3.1.7 D Flip Flop

3.2 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 3.2.01 (a).

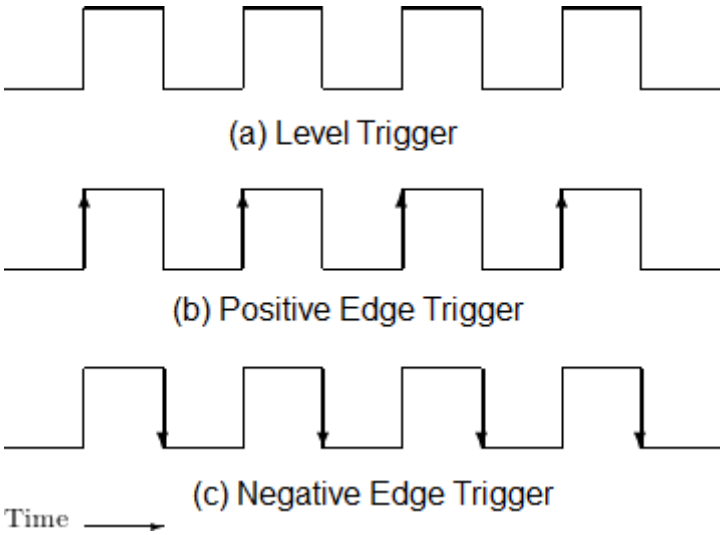


Figure 3.2.01 Level and Edge Triggering

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 3.2.01 (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.

3.3 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):

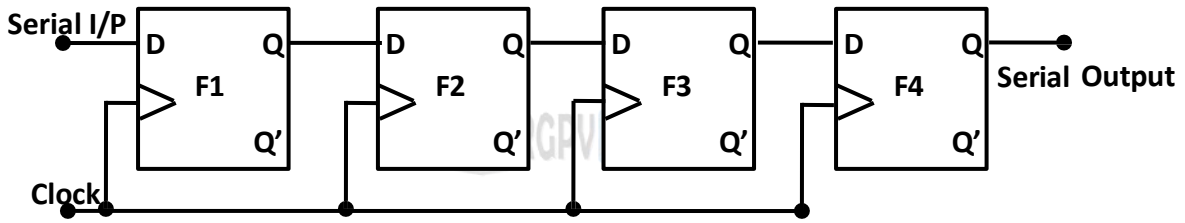


Figure 3.3.01Serial Input Serial Output Shift Register

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.

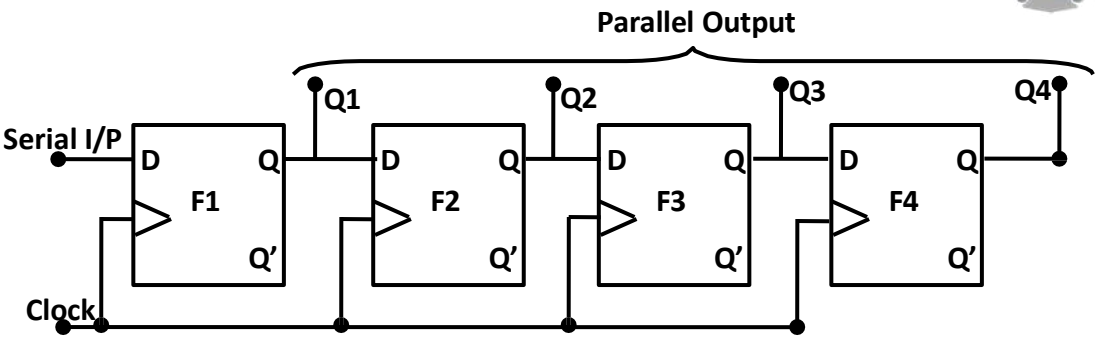


Figure 3.3.02 Serial Input Parallel Output Shift Register

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

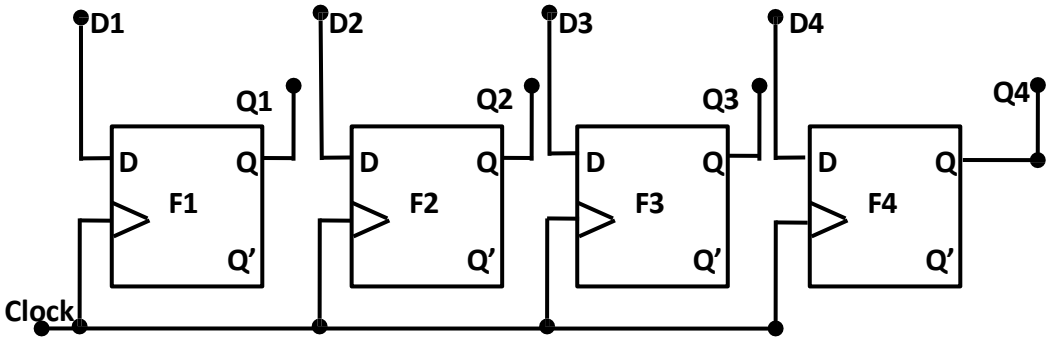


Figure 3.3.03 Parallel Input Parallel Output Shift Register

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.

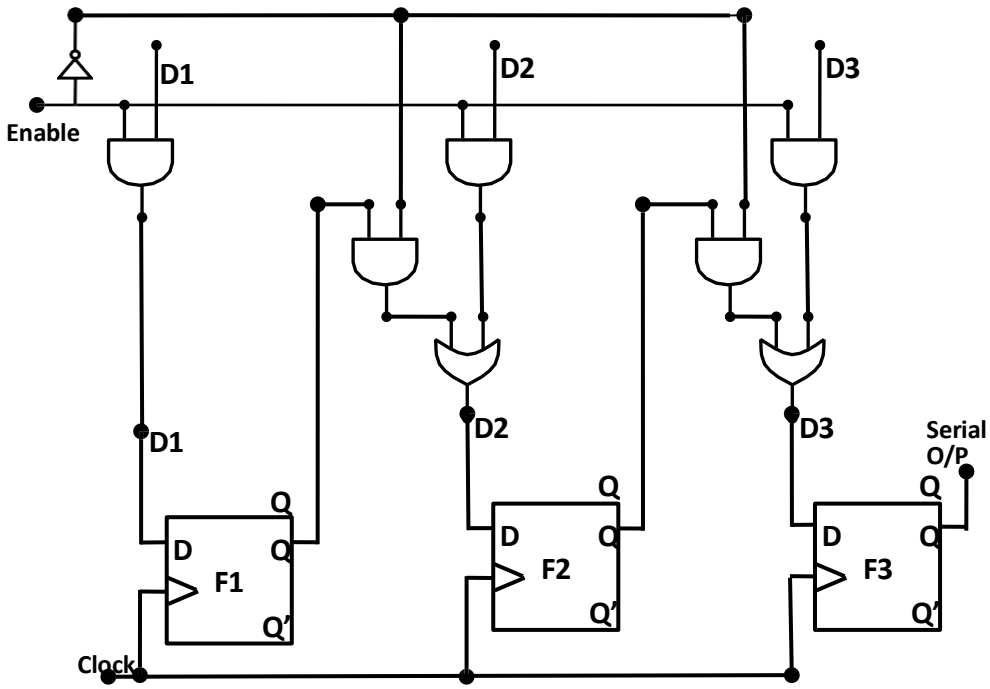


Figure 3.3.04 Parallel Input Serial Output Shift Register

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.The SISO 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.

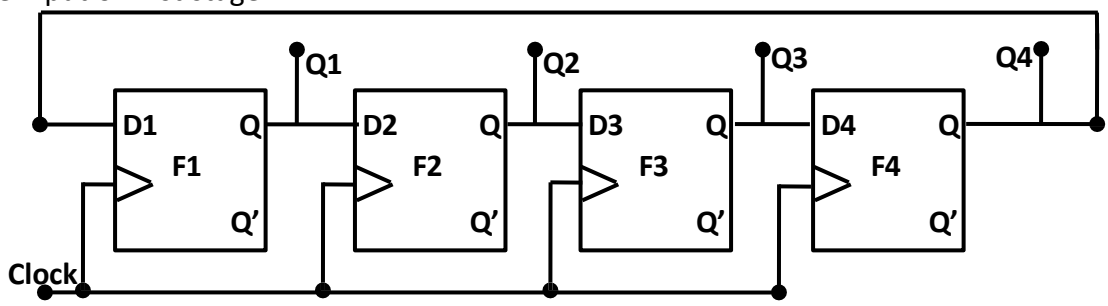


Figure3.3.05: 4-bit Ring Counter

	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

Table 3.3.01: 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.

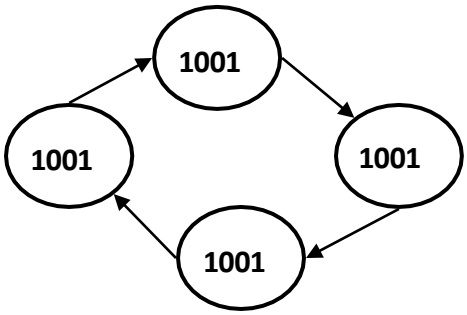


Figure 3.3.06: State Diagram of Ring Counter

3. JOHNSON 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.

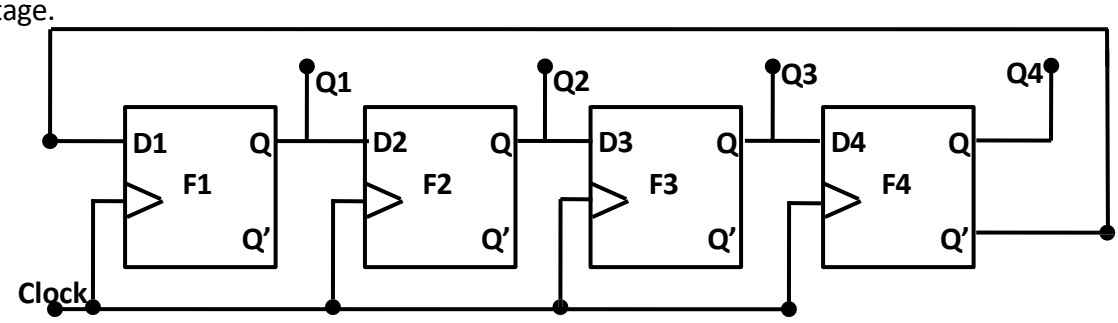


Figure3.3.07: 4-bit Johnson Counter

	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

4. SERIAL ADDER:

In serial adder, the pair of bits in A and B are transferred serially, one at a time, to the single full adder to produce a string of output bits for the sum.

Operation: In serial adder shown in below figure, register A stores augend bits and B stores addend bits. The two binary numbers are added serially stored in two shift registers. One pair of bits are added, one at a time, through full adder. The carry out is transferred to D Flip flop input. The output of D flip flop is used as an input carry to next pair bits. Register A is used for storing the SUM and augend bits.

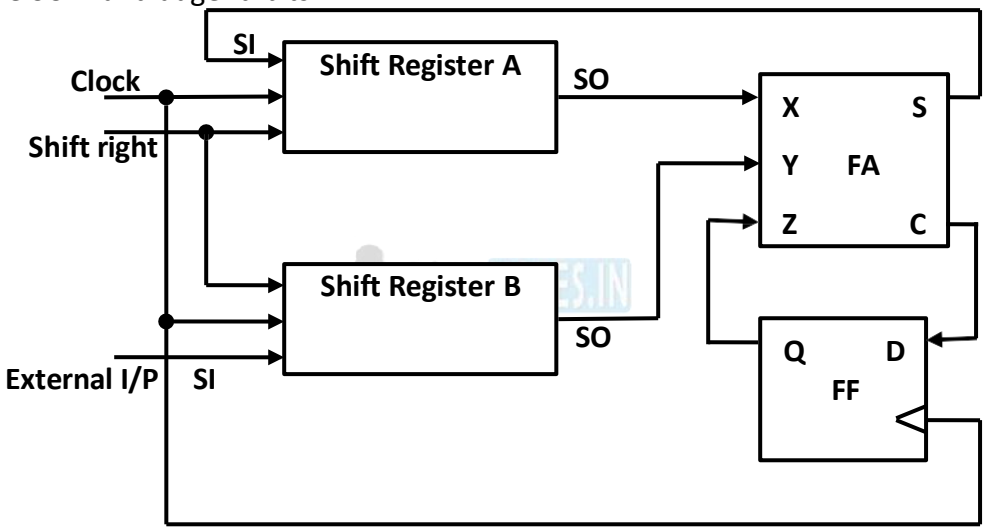


Figure3.3.08: Serial Adder

3.4 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):

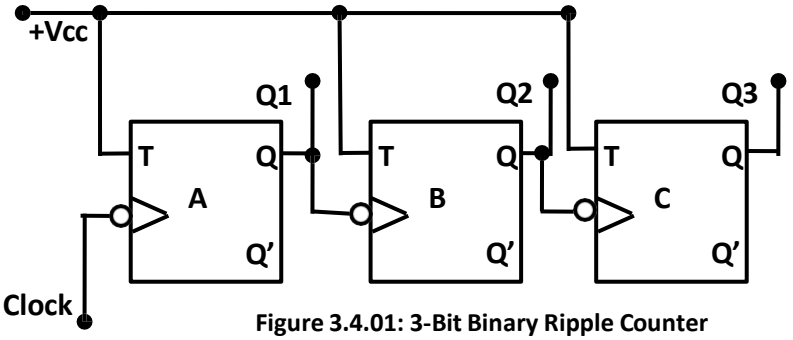


Figure 3.4.01: 3-Bit Binary Ripple Counter

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

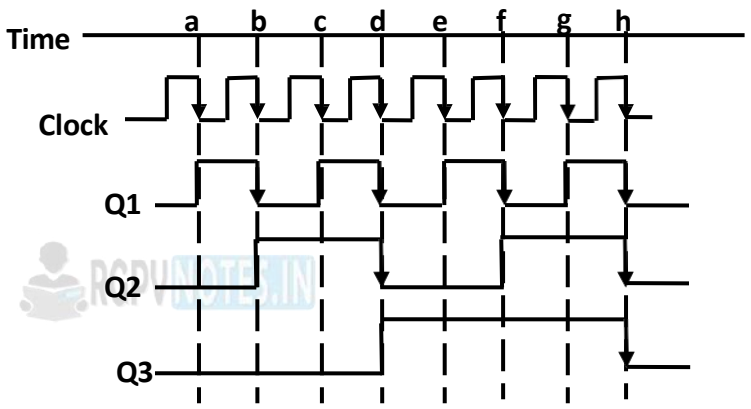


Figure 3.4.02: Waveforms

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:

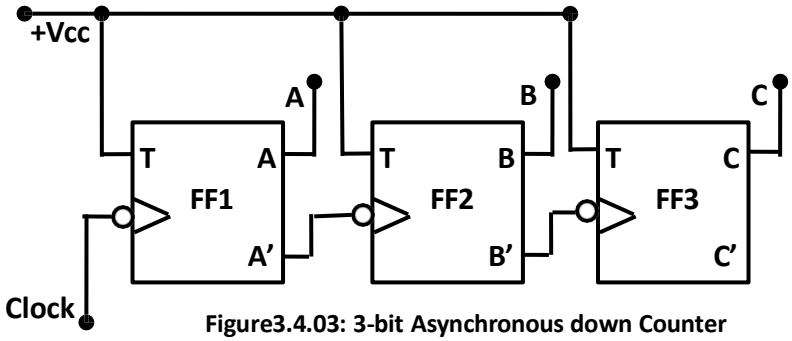


Figure3.4.03: 3-bit Asynchronous down Counter

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**

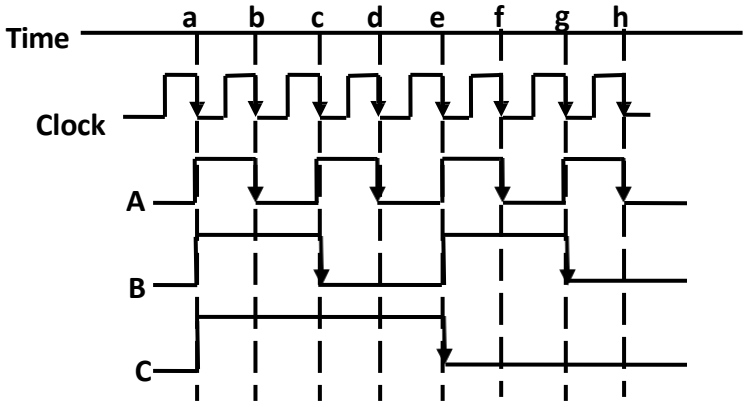


Figure 3.4.04: Waveforms

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:

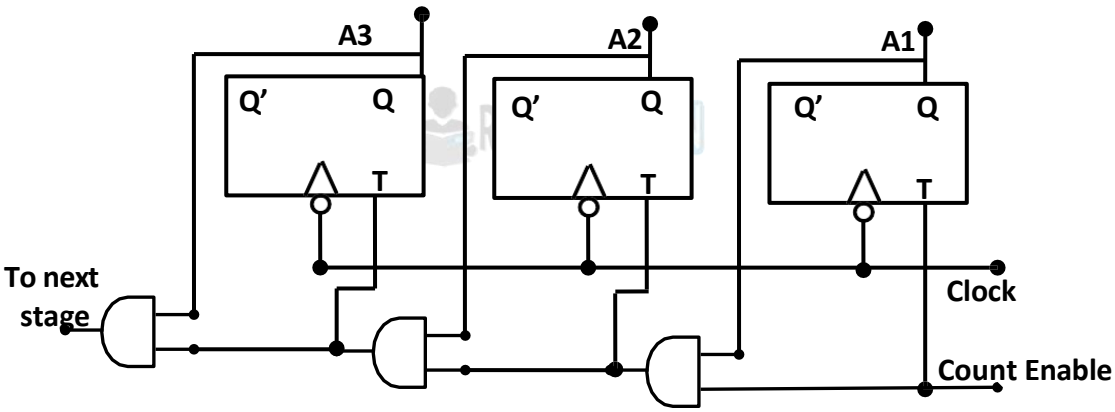


Figure 3.4.05: 3- Bit Synchronous Up Counter

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**

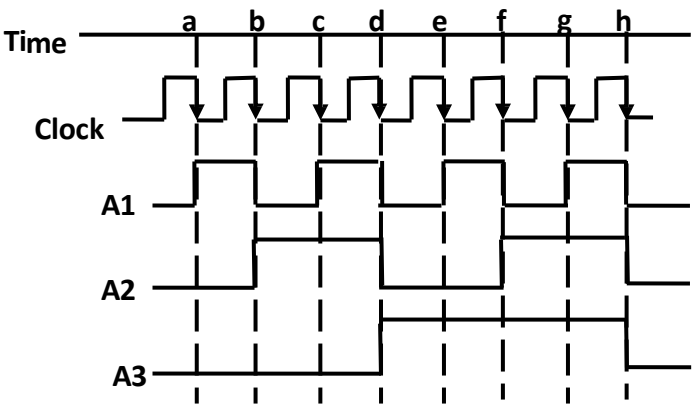


Figure 3.4.06: Waveforms

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

SYNCHRONOUS 3-BIT BINARY DOWN COUNTER :

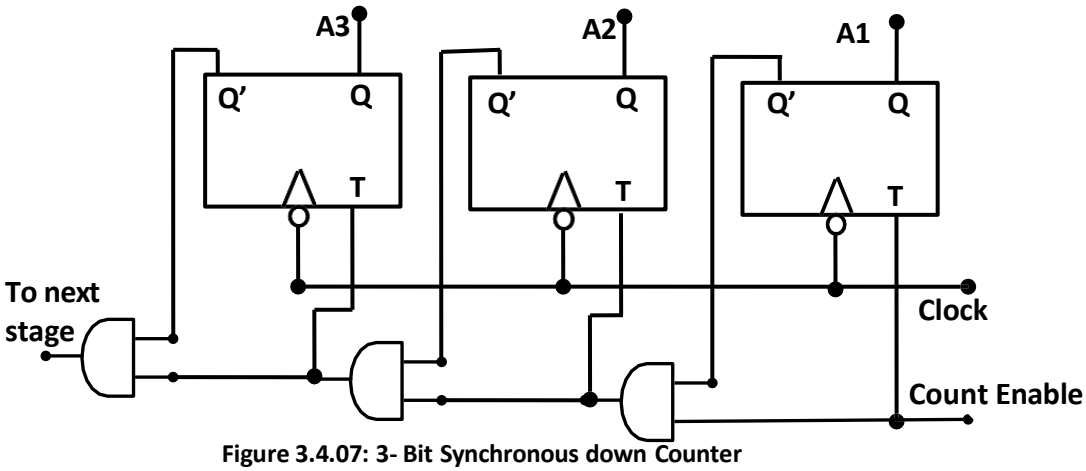
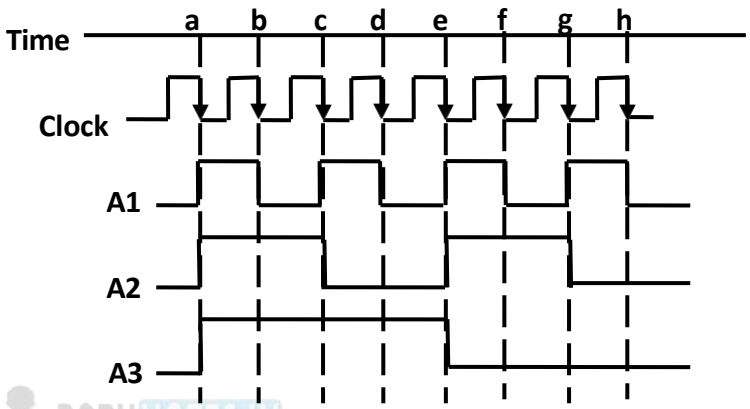


Figure 3.4.07: 3- Bit Synchronous down Counter

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**



RGPVNOTES.IN

Waveforms

Figure 3.4.08: Waveform and truth table of 3-bit binary down counter

3.5 SEMICONDUCTOR MEMORIES: MODERN TRENDS IN SEMICONDUCTOR MEMORIES SUCH AS DRAM, FLASH RAM ETC. DESIGNING WITH ROM AND PLA:

Memory is the ability to store information. The circuits or systems design, needs for data storage is called memory. A computer uses a number of memory devices of different technologies such as semiconductor memory, magnetic memory etc.

Memory Organization:

The basic element of a semiconductor memory is flip flop. There are number of memory locations in a memory chip, each location being ment for one word.

Size of a memory chip = M x N bits.

Where M represents the number of locations in memory and N represents the number of bits at each location.

Block diagram of memory device consists of M locations of memory and is defined by unique address and therefore for accessing any one of the M locations, P inputs are required, where $M = 2^p$. This is referred to as address lines. The number of inputs required to store the data into or read the data from any memory location is N.

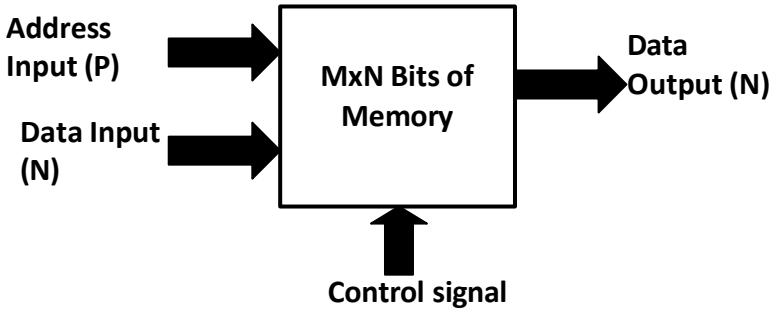


Figure 3.5.01: Block diagram of memory device

The number of distinct address possible with 'n' input variable is 2^n. The number of bits per word is equal to number of output lines.
Example: 1k-byte of memory chip has 1024 registers with 8-bits each.
The total capacity of memory having P address lines and N data lines is defined as 2^P x N.

Classification of Memories:

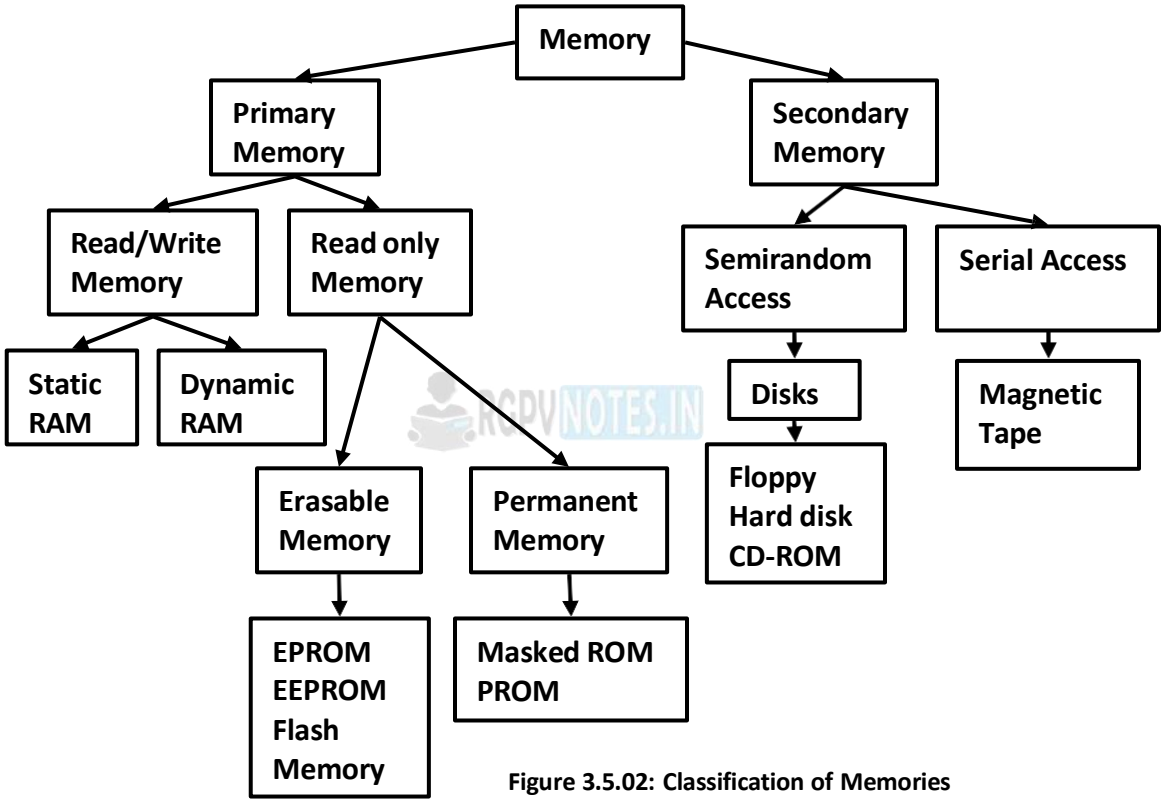


Figure 3.5.02: Classification of Memories

Volatile and Non volatile memories: The memory that required electrical energy to stored the information is called volatile memory. Information stored in memory is lost when electrical power is off. RAM is a volatile memory.
In a non volatile memory, the information once stored remain intact. ROM is non volatile.

Read Only Memory(ROM): It is used to store information which is permanent or semipermanent in nature. The permanent group includes: masked ROM and PROM and semipermanent group include: EPROM and EEPROM.
Masked ROM: Programming is done through masking and metallization process. User cannot write into this memory.
PROM:Programmable ROM: User can program(write) the PROM through PROM programmer.It can be programmed once only, user cannot rewrite this memory.
EPROM: Erasable Programmable ROM: User can rewrite this memory, many times. Program erasing is done using ultra violet light through a window over the memory chip. This window

is exposed in UV light. Entire information is erased at once when exposed in UV.

EEPROM: Electrical Erasable PROM: Information can be altered by using electrical signals at the register level rather than erasing all the information i.e information can be erased byte by byte.

Flash Memory: Erasing is done sector by sector not byte by byte.

RAM: Random Access Memory: Read and Write Memory: In such memories, the data stored at any location can be changed during the operation of the system. This type of memory is known as read write (RAM) memory. RAM is a volatile memory. Two types of RAM are there, static RAM and dynamic RAM.

Static RAM: The stored data will remain permanently stored as long as power supplied, without the need for periodically rewriting the data into memory. The basic memory cell is flip flops. One static RAM uses six transistors. Static RAM is faster than dynamic. More expensive and high durability.

Dynamic RAM: Stored data will not remain permanently stored, even with power applied, unless the data are periodically rewritten into memory i.e. the refresh operation. Basic memory cell is capacitor. Dynamic RAM uses 4-MOS transistors. Extra refreshing circuitry is present to accomplish refresh operation. Low power consumption.

ROM Organisation:

Diode Matrix ROM:

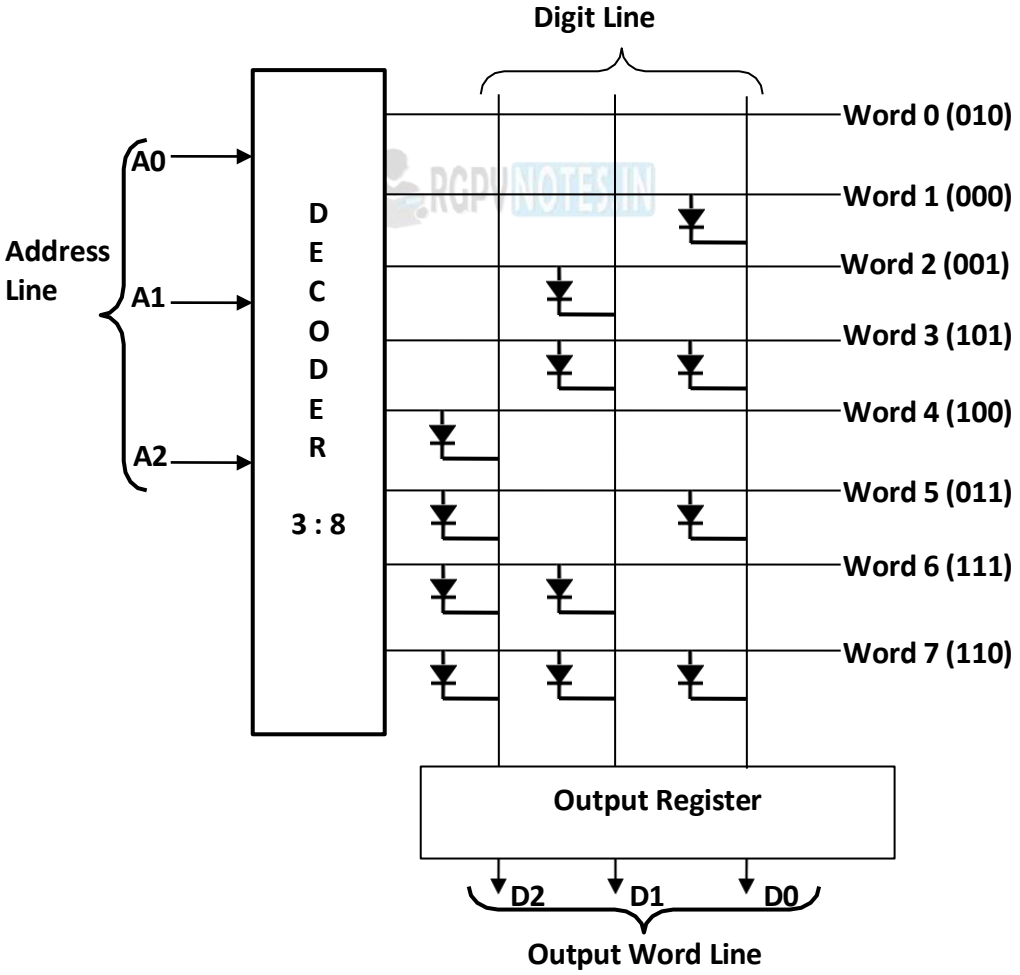


Figure3.5.03: 8-Bit ROM Array

From the above figure,8-bit ROM array, each horizontal row is a register. A diode present at the intersection represent logic 1 where as absence of a diode represent logic 0. The decoder selects one of the 8-word by increasing the voltage of the corresponding word line. This

forward bias the diodes, thus shorting the digit line to the word line. In this way the data is read across the output register.

Truth Table of 8-bit ROM Array:

Address Lines			Output of ROM		
A_2	A_1	A_0	D_2	D_1	D_0
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	1	1	0

Bipolar ROM: The bipolar transistor is used to connect the word line with digit line in place of diodes in diode matrix. Below figure shows the bipolar ROM cell.

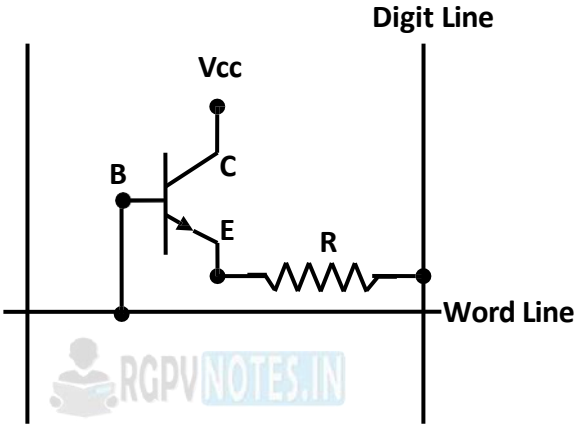


Figure 3.5.04: Bipolar ROM cell

When decoder selects one of the word by raising the voltage of corresponding word line, the emitter base junction of transistor is forward biased, which allows current to flow through resistor R, and tht bit of the word is a 1. When decoder output line is low, it is not selected and that bit of the word is a 0. The advantage of bipolar ROM over diode ROM are high packing density, low power consumption and high speed.

MOS ROM: In MOS ROM cell the gate and drain terminal of MOSFETare connected to word line and substrate and sourse terminals are connected to digit line as shown below.

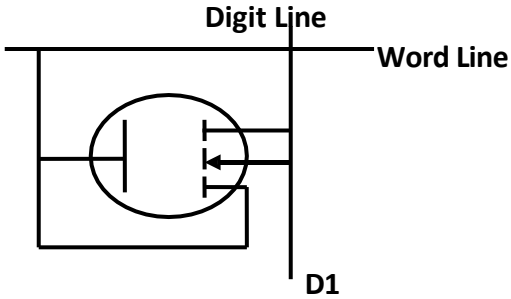


Figure 3.5.05: MOS ROM cell

The decoder selects one of the words by raising the voltage of the corresponding word line. MOSFET resistance is near zero ohms, thus shorting the digit lines to the word line and causing a high voltage to appear on these digit lines. A low at gate and drain terminals causes the FET to open, thus the digit line will remain to ground voltage. I this way the bits of the

addressed word are read.

PROGRAMMABLE LOGIC DESIGN (PLD):

PLD are special type of IC that contains a large number of logic gates whose interconnections are programmed by the user to generate the desired logic relationship between inputs and outputs. Types of PLDs:

a) Read only memories (ROMs)
b) Programmale logic array (PLAs)
c) Programmable array logic (PALs)
d) Simple Programmale logic array (SPLAs)
e) Complex Programmale logic array (CPLAs)
f) Field programmable gate arrays (FPGAs)

ROM as PLD: A ROM is a device that includes both the decoder and the OR gates within a single IC packages. A ROM of size MxN has M number of locations and each location can store N bit word. There are P number of address lines to access M locations hence the relation between P and M is given by $2^P = M$.

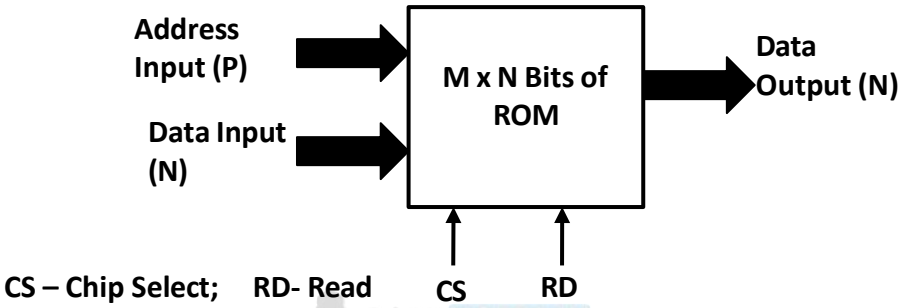


Figure 3.5.06: Block diagram of a ROM

There are N number of data lines, on which the stored data can be outputted, when the read (RD) input is active.

Internal Logic of a ROM:

Let us understand the internal logic of a 8 X 4 ROM. A 8 X 4 ROM consists of 8 words of 4 bit each. To access 8 locations, we have to use 3 input lines which form the binary words equivalent to 0 to 7 i.e. 000 to 111. Figure below shows the internal structure of ROM.

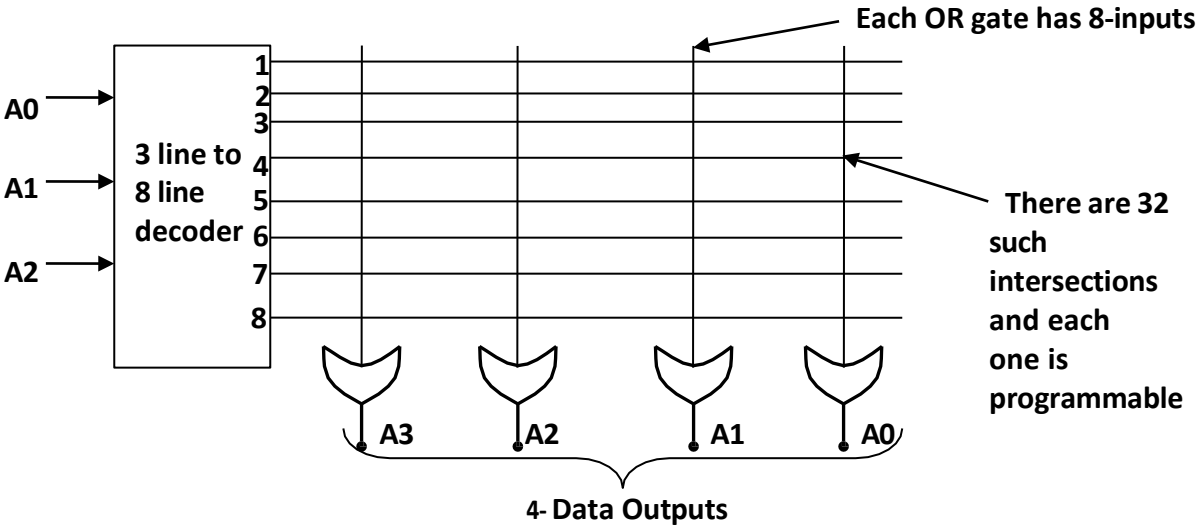


Figure 3.5.07: Internal structure of 8 x 4 ROM

The 3-inputs, A0,A1,A2 are decoded into 8 lines with the help of 3:8 line decoder. Each output of decoder represents a memory address. The 8-outputs are connected to each of the four OR gates. This means that each OR gate has 8-inputs, hence 32 internal connections. All 32 intersections are programmable (each interconnection is equivalent to a switch, ON or OFF, where ON means connection between two intersecting lines where as OFF means no connection). The programmable intersections is called as a crosspoint. The cross point is implemented using fuse link.

ROM TRUTH TABLE:

Location No.	Inputs			Outputs				
	A2	A1	A0	A3	A2	A1	A0	
0	0	0	0	**1**	**0**	**0**	**0**	**Data in location 0**
1	0	0	1	**1**	**1**	**0**	**0**	**Data in location 1**
2	0	1	0	**1**	**0**	**1**	**0**	**Data in location 2**
3	0	1	1	**0**	**0**	**1**	**1**	**Data in location 3**
4	1	0	0	**1**	**1**	**1**	**1**	**Data in location 4**
5	1	0	1	**0**	**0**	**0**	**0**	**Data in location 5**
6	1	1	0	**0**	**1**	**0**	**1**	**Data in location 6**
7	1	1	1	**1**	**0**	**0**	**1**	**Data in location 7**

The truth table shows that there are 3-input lines and 4-output lines. There are 8 locations (addresses) and at each location, an 4-bit word is stored.

Programming of a ROM:

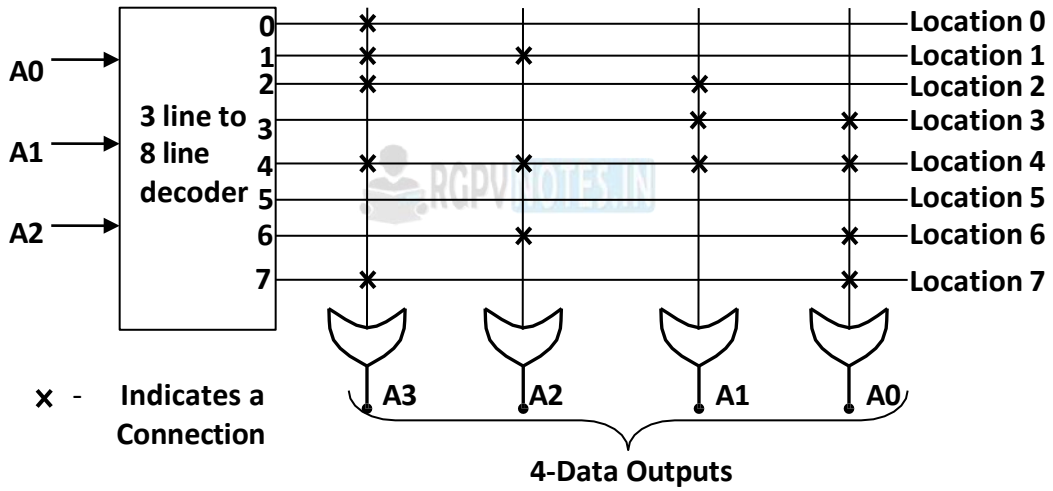


Figure 3.5.08: ROM Programmed according Truth Table

We consider location 0 of above figure. The word 1000 has been stored in this location. The 0's are stored by blowing OFF the fuse links between output 0 of decoder and input lines of OR gates, associated with A2,A1 and A0. The one 1 is marked by X.

PROGRAMMABLE LOGIC ARRAY (PLA):

Based on the idea that output logic functions can be realized in Sum –of – Product (SOP) form. As shown below in structure of PLA- PLA's inputs X1 Xn, pass through set of buffers and inverters (which provide both the true and complement of each input) into a circuit block called AND plane or AND array. AND plane outputs are set of product terms P1-------Pk i.e. each product terms can be configured to implement any AND function of X1 Xn. Product terms (P1----Pk) serves as the input to OR-Plne which produces the output F1 Fk. Each output can be configured to realize any sum of P1 Pk.

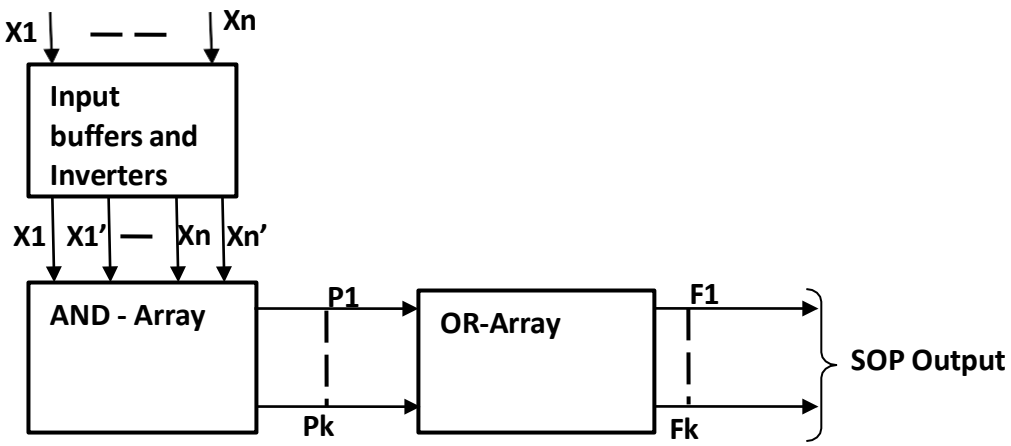


Figure 3.5.09: Block diagram of General Structure of PLA

Example: Implement the circuit using PLA with 3-inputs, 3-product terms and 2-outputs.

$F1(A,B,C) = \sum m(4,5,7)$ and $F2(A,B,C) = \sum m(3,5,7)$

Solution: On simplifying the function using K-Map method, we get the output expression as-

$F1 = A.B' + A.C$ and $F2 = A.C + B.C$

So, comparing F1 and F2, we get 3-product terms i.e. A.B', AC, BC

So, PLA table is-

Product Terms	Inputs			Outputs	
	A	B	C	F1	F2
A.C	1	-	1	1	1
B.C	-	1	1	-	1
A.B'	1	0	-	1	-

RGPVNOTES.IN

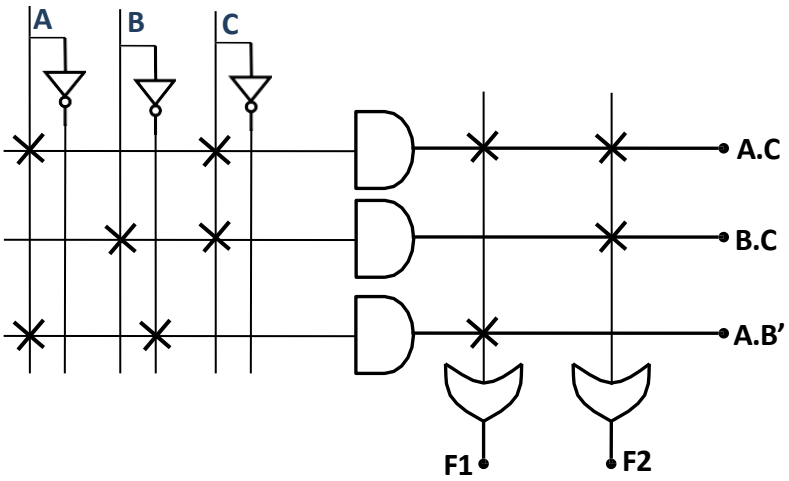


Figure 3.5.10 Circuit Implementation using PLA of above Example

3.6 INTRODUCTION TO DIGITAL ICS 2716 (NMOS EPROM), 2732 (NMOS EPROM) ETC.

Digital ICs M2716 is NMOS 16Kbit (2Kb x 8) UV EPROM and M2732 is NMOS 32Kbit (4Kb x 8) UV EPROM.

The M2716 is a 16,384 bit UV erasable and electrically programmable memory EPROM. The transparent window allows the user to expose the chip to ultraviolet light to erase the bit pattern.
VCC: Supply Voltage
VPP: Program Supply Voltage
G': Output Enable
E'P: Chip Enable
A0-A10: Address Line
Q0-Q7: Data Line

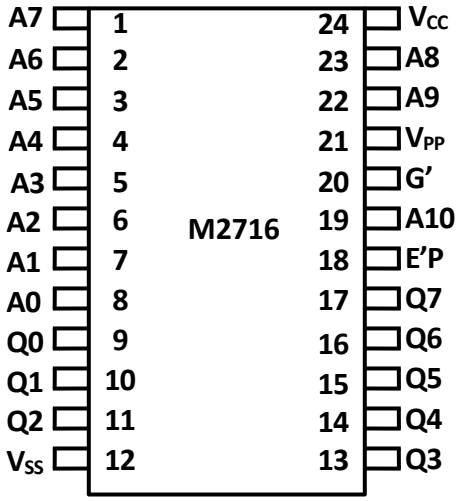


Figure 3.6.1 IC 2716 PIN Configuration

DEVICE OPERATION

The M2716 has 6 modes of operation shown in table below.

MODE	E'P	G'	V_{PP}	Q0-Q7
Read	V_{IL}	V_{IL}	V_{CC}	Data Out
Program	V_{IH} Pulse	V_{IH}	V_{PP}	Data In
Verify	V_{IL}	V_{IL}	V_{CC} or V_{PP}	Data Out
Program Inhibit	V_{IL}	V_{IH}	V_{PP}	Hi-Z
Deslect	X (V_{IH} or V_{IL})	V_{IH}	V_{CC}	Hi-Z
Standby	V_{IH}	X (V_{IH} or V_{IL})	V_{CC}	Hi-Z

V_{IH}: Input High Voltage; V_{IL}: Input Low Voltage

Read Mode. The M2716 read operation requires that G' = V_{IL}, E'P = V_{IL} and that addresses A0-A10 have been stabilized. Valid data will appear on the output pins after some switching time depending on which is limiting.

Deselect Mode. The M2716 is deselected by making G' = V_{IH}. This mode is independent of E'P and the condition of the addresses. The outputs are Hi-Z when G' = V_{IH}. This allows tied-OR of 2 or more M2716's for memory expansion.

Standby Mode (Power Down). The M2716 may be powered down to the standby mode by making E'P = V_{IH}. This is independent of G' and automatically puts the outputs in the Hi-Z state. The power is reduced to 25% (132 mW max) of the normal operating power. V_{CC} and V_{PP} must be maintained at 5V.

Programming

Table shows the 3 programming modes.

Program Mode. The M2716 is programmed by introducing "0"s into the desired locations. This is done 8 bits (a byte) at a time. Any individual address, sequential addresses, or addresses chosen at random may be programmed. Any or all of the 8 bits associated with an address location may be programmed with a single program pulse applied to the E'P pin. All input voltage levels including the program pulse on chip enable are TTL compatible. The programming sequence is: with V_{PP} = 25V, V_{CC} = 5V, G' = V_{IH} and E'P = V_{IL}, an address is selected and the desired data word is applied to the output pins (V_{IL} = "0" and V_{IH} = "1" for both address and data). After the address and data signals are stable the program pin is pulsed from V_{IL} to V_{IH} with a pulse width between 45ms and 55ms. Multiple pulses are not needed but will not cause device damage. No pins should be left open. A high level (V_{IH} or

RGPVNOTES.IN

higher) must not be maintained longer on the program pin during programming. M2716's may be programmed in parallel in this mode.

Program Verify Mode. The programming of the M2716 may be verified either one byte at a time during the programming or by reading all of the bytes out at the end of the programming sequence. This can be done with V_{PP} = 25V or 5V in either case. V_{PP} must be at 5V for all operating modes and can be maintained at 25V for all programming modes.

Program Inhibit Mode. The program inhibit mode allows several M2716's to be programmed simultaneously with different data for each one by controlling which ones receive the program pulse. All similar inputs of the M2716 may be paralleled. Pulsing the program pin (from V_{IL} to V_{IH}) will program a unit while inhibiting the program pulse to a unit will keep it from being programmed and keeping G' = V_{IH} will put its outputs in the Hi-Z state.

ERASURE OPERATION

The M2716 is erased by exposure to high intensity ultraviolet light through the transparent window. This exposure discharges the floating gate to its initial state through induced photo current. It is recommended that the M2716 be kept out of direct sunlight. The UV content of sunlight may cause a partial erasure of some bits in a relatively short period of time.

The M2716 to be erased should be placed 1 inch away from the lamp and no filters should be used. Lamps lose intensity as they age, it is therefore important to periodically check that the UV system is in good order.This will ensure that the EPROMs are being completely erased. Incomplete erasure will cause symptoms that can be misleading.

RGPVNOTES.IN

Subject Notes

UNIT-IV

Introduction to A/D & D/A convertors & their types, sample and hold circuits, Voltage to Frequency & Frequency to Voltage conversion. Multivibrators :Bistable, Monostable, Astable, Schmitt trigger, IC 555 & Its applications. TTL, PMOS, CMOS and NMOS logic. Interfacing between TTL to MOS.

ANALOG TO DIGITAL CONVERTER (ADC):

Figure 4.1 shows the general block diagram of ADC. An ADC takes an analog input voltage and after a certain amount of time produces a digital output code that represents the analog input.

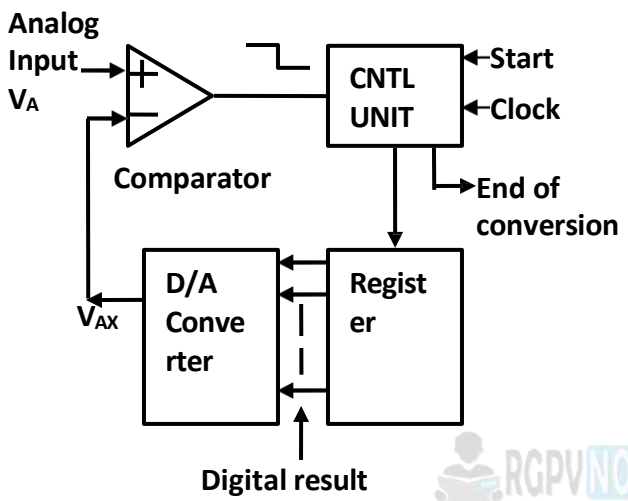


Figure 4.1: General block diagram of ADC

Timing for operation is provided by the input clock signal.

Control (CNTL) unit contains the logic circuitry for generating the proper sequence of operations in response to the start command, which initiates the conversion process.

OPAMP comparator has two analog inputs and a digital output that switches states, depending upon which analog input is greater.

Operation: START command pulse initiates the operation. At a rate determined by the clock, the control unit continually modifies the binary number that is stored in register. The binary number in register is converted into analog voltage, V_{AX}, by DAC. The comparator compares V_{AX} with V_A. When $V_{AX} < V_A$, comparator output stays HIGH. When $V_{AX} > V_A$, by atleast an amount equal to threshold voltage, the comparator output goes LOW and stops the process of modifying the register number. At this point V_{AX} is close approximation with V_A.

The digital number in the register, which is the digital equivalent of V_{AX}, is also the digital equivalent of V_A, within the resolution and accuracy of the system. The control logic activates the end of conversion signal, when the conversion is complete.

DIGITAL RAMP ADC (COUNTER TYPE ADC):

Figure 4.2 shows the diagram of digital ramp ADC.

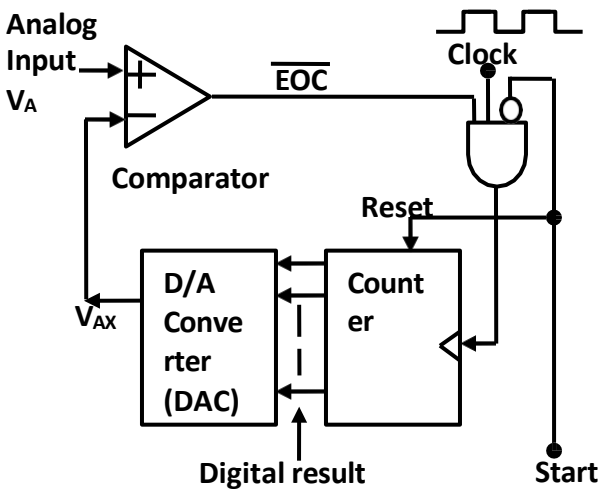
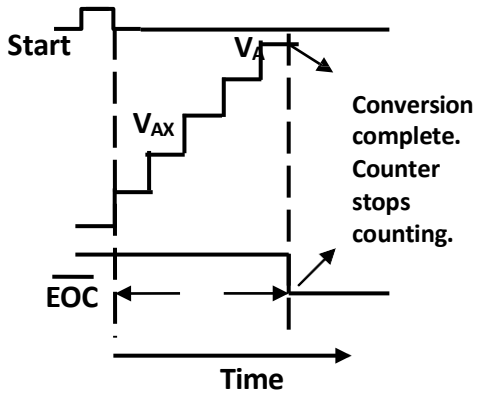


Figure 4.2: Digital Ramp ADC



Operation: Assume that V_A is positive. START pulse is applied to RESET the counter to 0 and AND gate is disabled. With all 0s as its input, DAC output will be V_{AX} = 0v. Since $V_{AX} < V_A$, comparator output, (EOC)', is HIGH. When START is LOW, AND gate is enabled and clock pulses get through to the counter. As the counter advances, DAC output, V_{AX}, increases one step at a time. This continues until $V_{AX} > V_A$ by an amount equal or greater than threshold voltage (typically 10 to 100µv). At this point comparator output, (EOC)', goes LOW and counter stop counting. The conversion process is now complete and the contents of the counter are the digital representation of V_A. Counter will hold the digital value until the next START pulse initiates a new conversion.

Conversion time, Tc, is the interval between the end of the START pulse and the activation of the (EOC)' output. Tc depends upon V_A.

For N-bit converter: $Tc_{(max)} = (2^N - 1)$ clock cycles. $Tc_{(avg)} = Tc_{(max)} / 2$ clock cycles.

Major disadvantage of digital ramp ADC is that it is not suitable for where the repetitive A/D conversion of a fast changing analog signal occurs. In this method the conversion time essentially doubles for each bit that is added to the counter.

SUCCESSIVE-APPROXIMATION ADC (SAC):

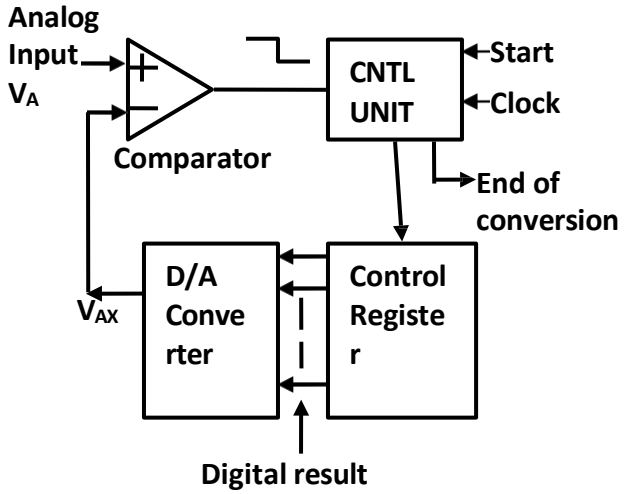
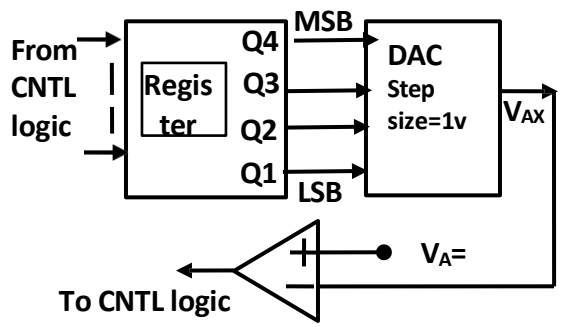


Figure 4.3: General block diagram of SAC

Figure 4.3 shows the general block diagram of SAC. Basic arrangement is similar to digital ramp ADC except that, instead of counter SAC uses a control register. The control logic modifies the contents of the register bit by bit until the register data are the digital equivalent of the analog input V_A within the resolution of the converter.

Most widely used ADC. Circuitry is more complex but much shorter conversion time

Operation of 4-bit SAC using DAC step size of 1Volt and V_A = 10.4 Volts:



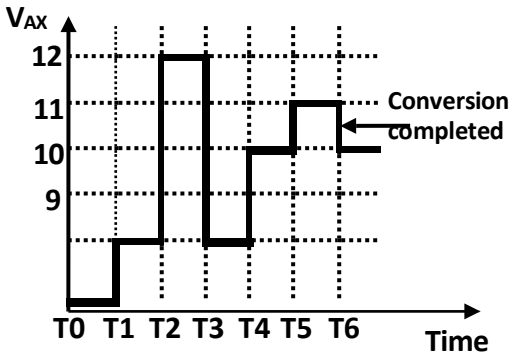


Figure 4.4: 4-bit SAC using step size 1V

Operation: Figure 4.4 shows the 4-bit SAC using DAC step size of 1Volt. Let assume that the analog input is V_A=10.4V.

At time T0, V_{AX} = 0V, i.e $V_A > V_{AX}$, comparator output is HIGH. Control logic clearing all bits so, Q3=Q2=Q1=Q0=0 ie [Q]= 0000.

At time T1, control logic (CNTL) sets MSB = 1. So [Q]= 1000. This produces V_{AX} = 8V. Since, $V_A > V_{AX}$, comparator output is HIGH. This HIGH tells the CNTL logic that the setting of MSB did not make V_{AX} exceeds V_A, so that MSB is kept at 1.

Now, CNTL logic proceeds to next lower bit, Q2. Q2=1 to produce [Q] = 1100 and V_{AX} = 12V at time T2. Since $V_{AX} > V_A$, comparator output goes LOW. The value of V_{AX} is too large, so CNTL logic then clears register contents back to 1000 ie V_{AX} = 8V. Thus, at T3, V_{AX} = 8V.

At time T4, CNTL logic sets the next lower bit Q1 = 1, ie [Q] = 1010 and V_{AX} = 10V. With $V_A > V_{AX}$, comparator output is HIGH and tells the CNTL logic to keep Q1 set at 1.

Final step, time T5, CNTL logic sets the next lower bit Q0 =1 ie [Q] = 1011 and V_{AX} = 11V. Since $V_{AX} > V_A$, comparator goes LOW to signal that V_{AX} is too large, and the CNTL logic clears back Q0 to 0 at time T6.

At this point, all of the register bits have been processed, the conversion is complete and the CNTL logic activates (EOC)' output to signal that is digital equivalent of V_A is now in the register. So, digital output for V_A = 10.4V is [Q] = 1010.

Conversion time,Tc, for SAC: The control logic goes to each register bit, set it to 1, decides whether or not to keep it at 1, and goes on to the next bit. The processing of each bit takes one clock cycle, so that the total conversion time for an N-bit SAC will be N-clock cycles.

Tc = N x 1 clock cycles

FLASH ADC: It is the highest speed ADC, but its circuitry requires much more than other

types ADC. Example: 6-bit flash ADC requires 63 analog comparators; 8-bit requires 255 comparators.

3-Bit Flash Converter:

Figure 4.5 shows the 4-bit flash converter.

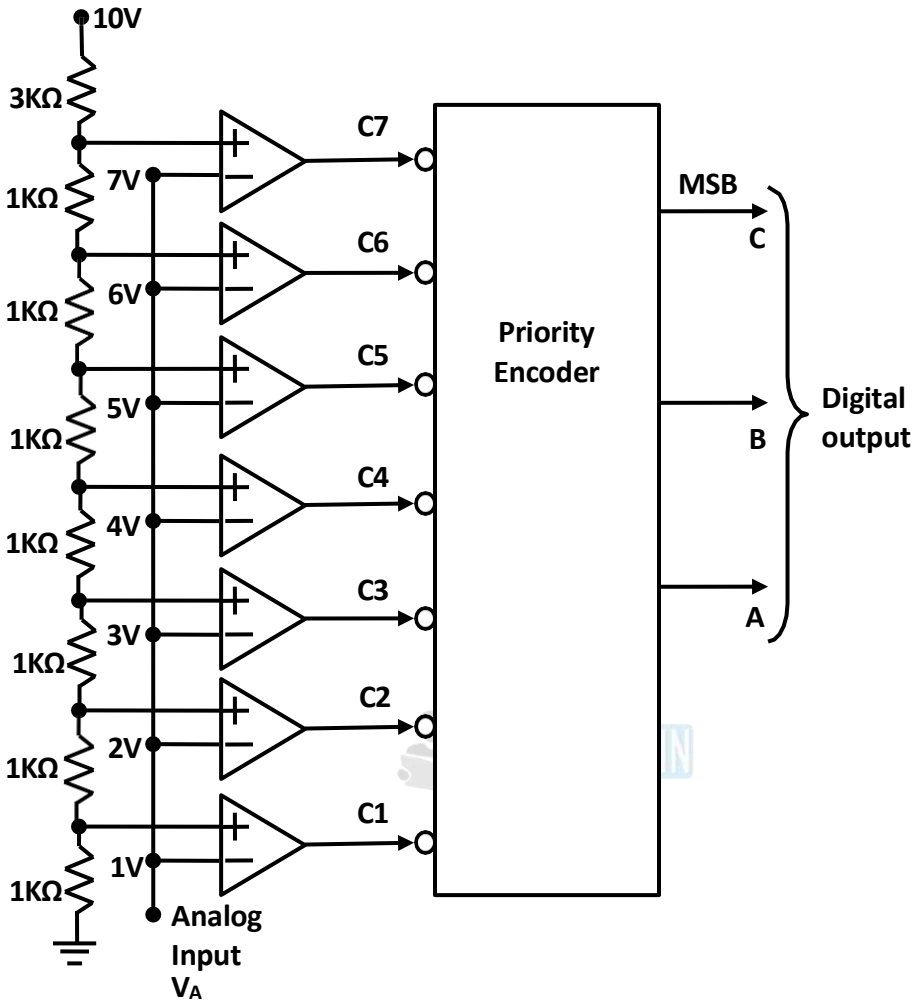


Figure 4.5: 4-bit Flash Converter

Operation: 3-bit flash converter has a resolution (step size) of 1V. Voltage divider set up a reference levels for each comparator, so that there are seven levels corresponding to 1V (weight of LSB), 2V, 3V, 4V, 5V, 6V and 7V (Full Scale). Analog input is connected to other input of each comparator. 3-bit flash converter ADC operation table is shown below:

Analog In	Comparator outputs							Digital Outputs		
(V_A)	C1	C2	C3	C4	C5	C6	C7	C	B	A
0V -1V	1	1	1	1	1	1	1	0	0	0
1V - 2V	0	1	1	1	1	1	1	0	0	1
2V - 3V	0	0	1	1	1	1	1	0	1	0
3 V- 4V	0	0	0	1	1	1	1	0	1	1
4V - 5V	0	0	0	0	1	1	1	1	0	0
5V - 6V	0	0	0	0	0	1	1	1	0	1
6V - 7V	0	0	0	0	0	0	1	1	1	0
>7V	0	0	0	0	0	0	0	1	1	1

TABLE of 3-bit Flash Converter ADC

With V_A < 1V, all comparator output is HIGH. With V_A > 1V, one or more comparators output

will be LOW. Comparator output is feed into active low priority encoder that generates a binary output corresponding to the highest numbered comparator output, that is LOW. For example, if V_A is between 3V – 4V, output C1, C2 and C3 will be LOW and all others are HIGH. Priority encoder will respond only to the LOW at C3 and will produce binary output CBA = 011.
Conversion Time, Tc of flash converter: Flash converter uses no clock signals. Conversion time depends only on the propagation delays of the comparators and encoder logic. So, flash converter has extremely short conversion times.

ADC USING VOLTAGE TO FREQUENCY CONVERTER:

Figure 4.6 shows the ADC using voltage to frequency converter.

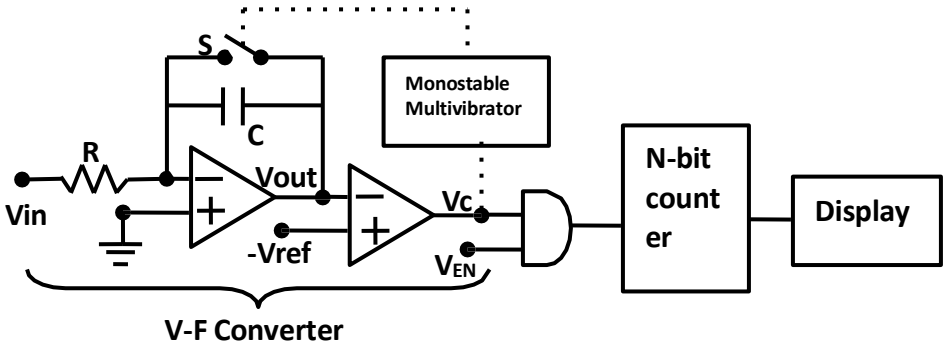


Figure 4.6: ADC using V-F converter

Voltage to frequency ADC does not require DAC. Instead it uses a linear voltage controlled oscillator (VCO), that produces an output frequency that is proportional to its input voltage. The analog input (Vin) that is to be converted, is applied to the VCO to generate the output frequency. This frequency is fed to the counter to be counted for a fixed time interval (V_{EN}). The final count is proportional to the value of the analog voltage. Circuit diagram of ADC using V-F converter shown in figure.
Operation: The Vin is applied to an integrator whose output is applied at the inverting terminal of a comparator. Non-inverting terminal is connected to –Vref. When switch S is open, Voltage Vout decreases linearly with time. Thus AND gate is disabled as long as Vout < Vref. As soon as Vout = Vref, the output Vc becomes positive, enabiling AND gate and hence counter starts counting. When the switch S is closed, the capacitor discharges and thereby returning the integrator output, Vout, to zero. After the delay time of multivibrator the switch S is again open and Vout starts decreasing again and ADC repeats its function.

FREQUENCY TO VOLATGE CONVERTER (INTEGRATING TYPE):

Block diagram of a voltage to frequency converter is shown in figure4.7 . The analog input is applied to an integrator. The integrator produces a ramp signal whose slope is proportional to the input voltage signal.

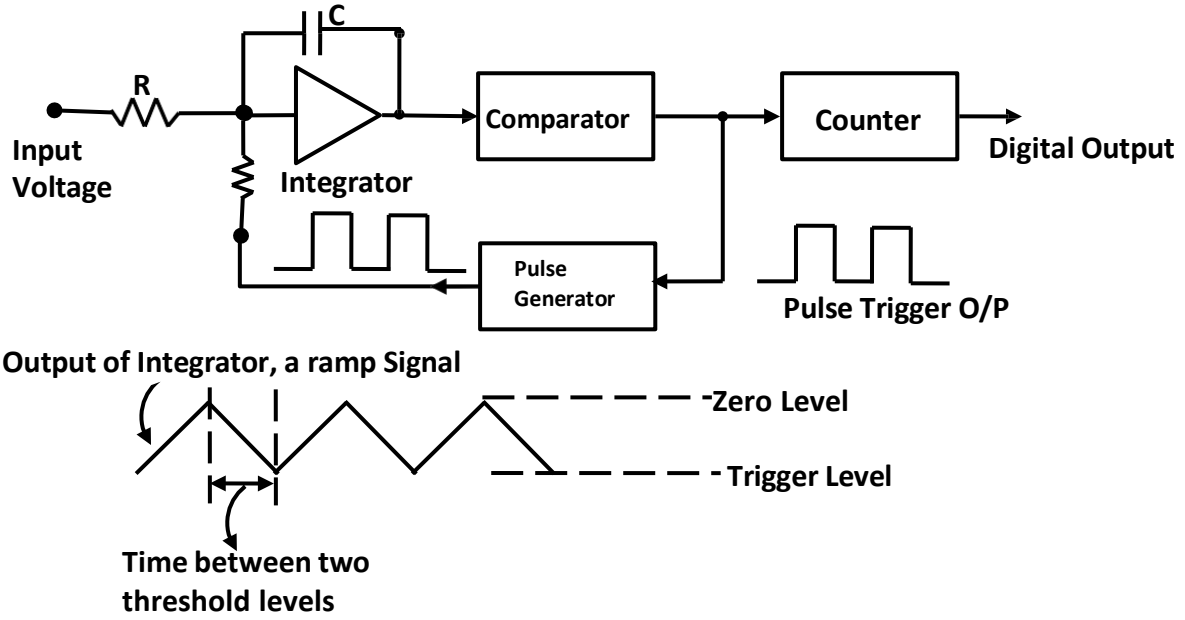


Figure 4.7: Frequency to Volatge converter

When this ramp signal reaches a preset threshold voltage level, a trigger pulse is produced. Also a current pulse is produced which discharges the capacitor of the integrator, after which a new ramp is initiated. The time between successive threshold level crossings is inversely proportional to the slope of the ramp. Since the slope of the ramp is proportional to the input analog voltage, hence the frequency of output pulses from the comparator is directly proportional to input voltage. The output frequency can be measured with the help of digital frequency meter.

SAMPLE AND HOLD CIRCUIT:

When an analog voltage is connected directly to the input of an ADC, the conversion process can be affected if the analog voltage is changing during the conversion time. This stability of conversion process can be improved by using a sample-and-hold circuit to hold the analog voltage constant while the A/D conversion is taking place.

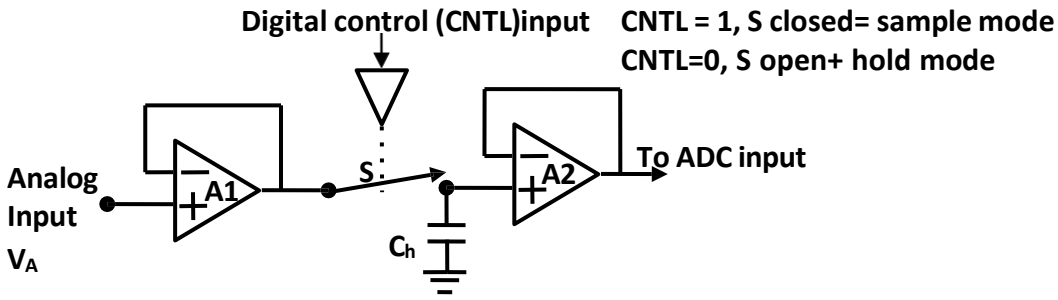


Figure4.8: Sample and Hold Circuit

Sample and hold circuit as shown in figure4.8, contains a unity gain buffer amplifier A1 that presents a high impedance to the analog signal and low output impedance that can rapidly charge the hold capacitor, C_h . Capacitor C_h is connected to output of A1 when digitally controlled switch is closed. This is called sample operation. The switch is closed long enough for C_h to charge to the current value of the analog input.

When switch opens, C_h will hold this voltage so that the output of A2 will apply this voltage to the ADC. The unity gain buffer amplifier A2 presents high input impedance that will not

discharge the capacitor voltage during the conversion time of the ADC.

DIGITAL TO ANALOG CONVERSION (DAC):

DAC is the process of taking digital code as input and converting it to a voltage or current that is proportional to digital value.

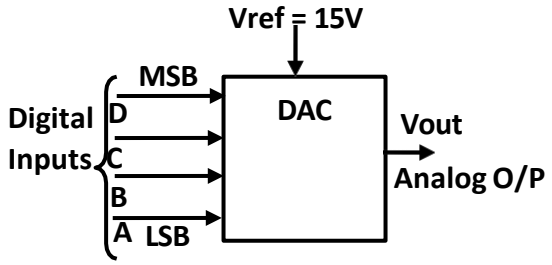


Figure 4.9:Block diagram 4-bit DAC

TABLE of 4-bit DAC:

Inputs				Output
D	C	B	A	Vout
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

From the block diagram as shown in figure 4.9 of 4-bit DAC ,Vref as input is used to determine the full scale output or maximum value that DAC can produce. For each input number, DAC output voltage is unique value. In general, **Analog output (Vout) = K x Digital Input;** Where, K is proportionality constant. In above block , DAC has K=1, so that, **Vout = 1 x Digital Input.**

Example: For digital input $(1100)_2 = (12)_{10}$, we obtain Vout = 1 x 12 = 12V.

RESOLUTION OR STEP SIZE OF DAC: Resolution of DAC is defined as the smallest change that can occur in the analog output as a result of a change in the digital input. Resolution is always equal to the weight of the LSB and also referred to as step size, since it is the amount that Vout will change as the digital input value is changed from one step to the next.

Resolution= K = A_{FS} / (2^n – 1) where, A_{FS} is analog full scale output; n is the number of bits

% Resolution = (Step size / A_{FS}) x 100 OR % Resolution = (1 / Total no. of steps) x 100

DAC USING OP-AMP SUMMING AMPLIFIER WITH BINARY WEIGHTED RESISTOR:

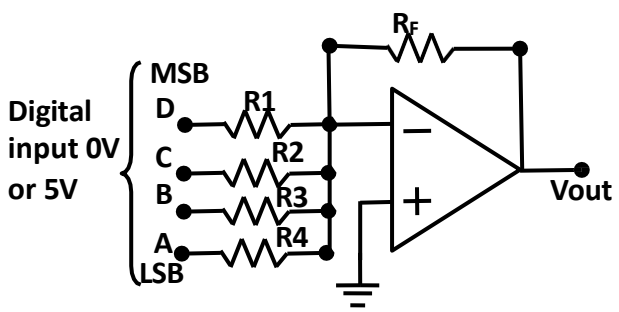


Figure4.10: OP-AMP as Summing Amplifier

Resistor R1 = 1KΩ
Resistor R2 = 2KΩ
Resistor R3 = 4KΩ
Resistor R_F = 8KΩ

From the figure4.10 , opamp as summing amplifier, inputs A, B, C and D are binary inputs that

are assumed to have a values either 0V or 5V. OP-AMP as summing amplifier, which produces the weighted sum of the input voltages. So that, **Vout = –(V_D + 1/2 V_C + 1/4 V_B + 1/8 V_A)**

So, the summing amplifier output is the analog voltage which represents a weighted sum of the digital inputs. The resolution of this DAC using opamp as summing amplifier using binary weighted resistor, is equal to the weighting of the LSB, which is 1 / 8 x 5 = 0.625V.

DAC USING R / 2R LADDER CIRCUIT:

In R/2R ladder circuit, the resistor values span a range of only 2 to 1.

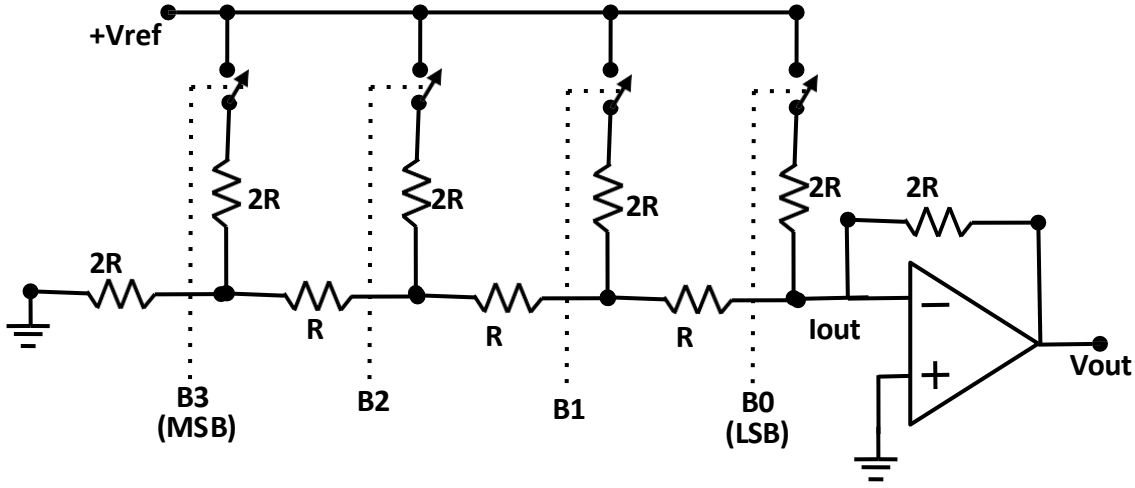


Figure 4.11: R/2R Ladder Circuit

In R/2R ladder network as shown in figure 4.11, only two different values are used, R and 2R. Current Iout depends on the positions of the 4- switches and the binary inputs B3, B2, B1 and B0, which controls the states of the switches.

Vout = (–Vref / 8) x B where "B" value of binary input from 0000 to 1111.

Example: Assume the Vref = 5V. What are the resolution and full scale output of this R/2R converter?

Solution: Resolution is equal to weight of LSB. Suppose, [B] = 0001 = $(1)_{10}$

Resolution = (-5V x 1)/ 8 = -0.625V. The full scale output occur for [B]=1111 = $(15)_{10}$

So, full scale output = (-5V x 15)/ 8 = -9.375V.

BISTABLE MULTIVIBRATOR:

If both the states of a multivibrator are stable i.e. the circuit which is in a particular state continues to remain in that state until it is triggered from an external source to change the state. Flip Flops are bistable multivibrator circuits as shown in figure 4.12.

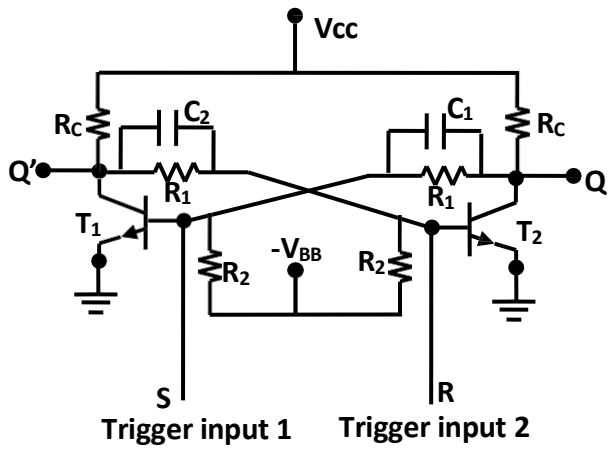
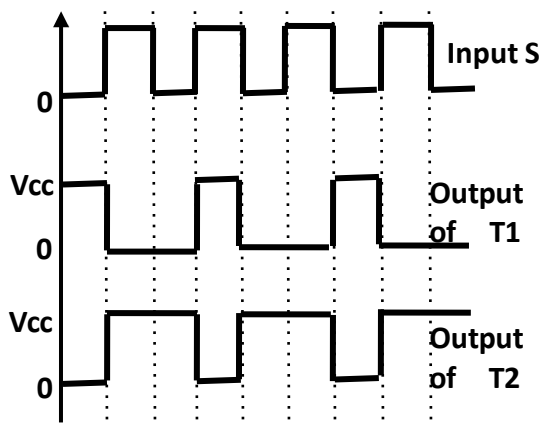


Figure4.12: Bistable Multivibrator



Transistor T_1 and T_2 are npn transistors and are resistively crosscoupled with each other. Q and Q' are outputs. Rc is the collector resistance. C_1 and C_2 are commutating capacitors, to fast turn OFF and ON of two transistors (C_1 for T_2 and C_2 for T_1). Supply $-V_{BB}$ and resistor R_2 are used to keep the base of the transistors at negative in one state and in other state provides large base current to drive the transistor into saturation.

Operation: When supply is ON, say T_1 is ON, so, V_{c1} = 0v, the base of T_2 is connected to V_{c1}, so T_2 is OFF. This makes $V_{c2} = V_{cc}$, since base of T_1 is connected to V_{c2}, so T_1 is ON. This is the first stable state (ie. T_1=ON and T_2=OFF ie. Q=1 and Q'=0).

To change the state of transistor T_2, a positive pulse is applied at the base of T_2. The OFF transistor T_2 will be forced to turn ON ($t_p > t_{on}$ ie. Pulse width should be greater than turn ON time of transistor). Thus V_{c2} is forced to 0v. Since base of T_1 is connected to V_{c2}, so T_1 is OFF. This is the second stable state (ie. T_1=OFF and T_2=ON ie. Q=0 and Q'=1).

MONOSTABLE MULTIVIBRATOR:

Monostable multivibrator as shown in figure 4.13 has one stable state and the other one is not stable (quasi stable).It is also called as one-shot multivibrator. Transition from stable state to quasi stable state is done by an external trigger pulse. After transition from stable to quasi stable state, the multivibrator remains in the quasi stable state for a definite period of time, decided by comoponents R and C, and then returns to the stable state automatically.

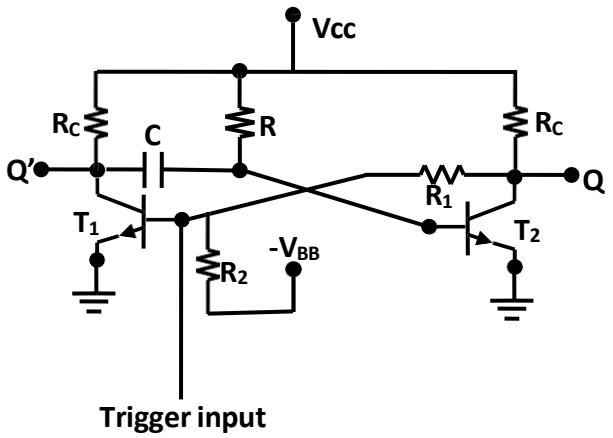
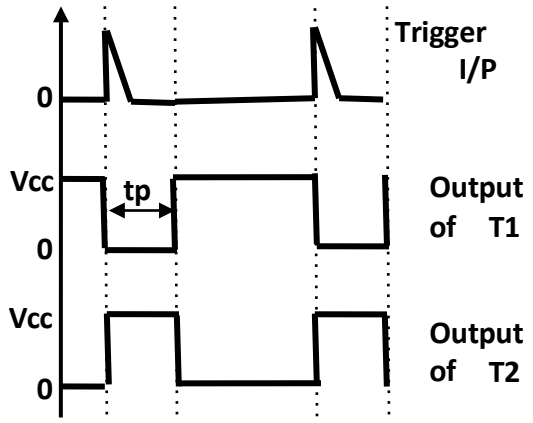


Figure4.13: Monostable Multivibrator



In this circuit the base of transistor T_2 is capacitively coupled to the collector of T_1, while the base of T_1 is resistively coupled to the collector of T_2.

Operation: When no trigger pulse applied, T_2 is ON, a proper base drive through Vcc and R to the base of T_2. So, V_{c2} = 0v. T_1 is OFF, because of resistively coupled of base T_1 with V_{c2}(V_{c1} = V_{cc}). At the instant when T_2 is ON, capacitor C charged towards V_{cc} through R_c. This is the stable state of multivibrator.

When a sufficient positive trigger pulse is applied to the base of T_1, T_1 is ON, so V_{c1} = 0v. Now the capacitor discharges through T_1 and R. Thus discharge current through R creates a negative potential at the base of T_2. Thus T_2 is OFF as long as the voltage drop across R is negative. This condition is quasi stable state. When the capacitor fully discharges the negative voltage at the base of T_2 reduces to zero and V_{cc} now drives the T_2 ON, so V_{c2} = 0v. So, T_1=OFF and V_{c1}=V_{cc}. This is the stable state and circuit remains in the stable state until the next trigger pulse is applied.

Time duration of quasi stable state or the pulse width: t_p = 0.693(R.C)

ASTABLE MULTIVIBRATOR:

Astable multivibrator as shown in figure 4.14, has no stable states but has two quasi stable states. The output oscillates between two quasi stable states without any external triggering ,therefore this circuit is also called as free running multivibrator. The output at the collector of transistors is a square wave, therefore also called as square wave generator.

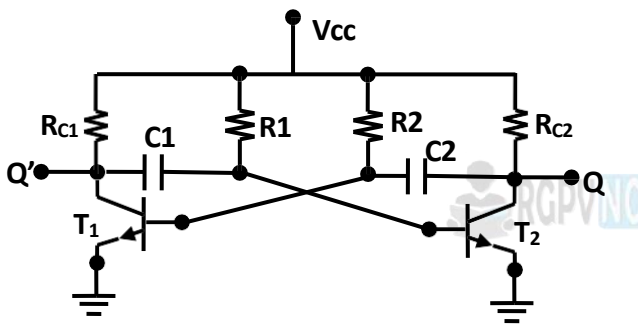
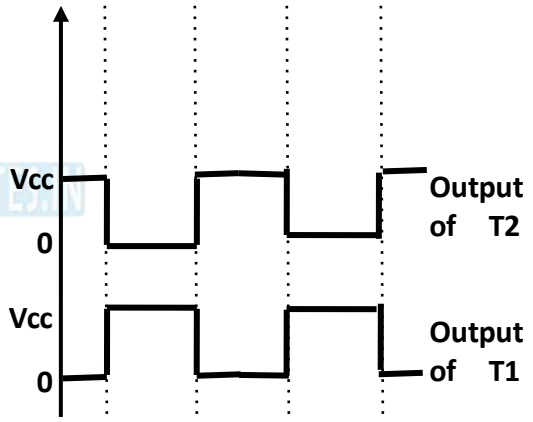


Figure4.14: Astable Multivibrator



Operation: Initially assume that T_1 is ON and T_2 is OFF due to circuit unbalance. So, V_{c1} = 0v and V_{c2} = V_{cc}. Since T_1 is ON, C_2 charges towards Vcc through R_{c2}. Meanwhile C_1 which was charged to V_{cc} when T_2 was ON will discharge through T_1 and R_1. This makes the potential at V_{B2} negative and causes T_2 to turn OFF. T_1 is kept ON by the base current provided by V_{cc} through R_2. The charging current of C_2 through R_{c2} has reduced to zero. The time duration for which T_2 is held OFF is determined by the R_1.C_1. Once T_2 turns ON due to the base drive from V_{cc} through R_1, then C_1 gets charged through R_{c1} and T_2. At the same time C_2 discharges through T_2 and R_2 making V_{B1} negative so that T_1 is turned OFF, thus V_{c1} = V_{cc}. T_1 is held OFF by the discharging current for the time duration R_2.C_2. After this T_1 turns ON and allow C_1 to discharge through T_1 and then C_2 recharges through T_1.

SCHMITT TRIGGER:

In digital circuits, fast waveforms are required so that the circuit remans in the active region for a very short time (of order of nano seconds) to eliminate the effects of noise or undesired parasitic oscillations causing malfunction of the circuit. Also if the rise time of the input waveform is long, it requires a large oupling capacitor. Therefore circuits which ca convert a slow-changing waveform (long rise time) into a fast-changing waveform (small rie time) are

required. The circuit which performs this waveform is known as Schmitt trigger.

In a Schmitt trigger circuit, the output is in one of the two levels, LOW or HIGH. From the figure 4.15,when the input voltage is rising, the level of the output changes when the input passes through a specific voltage V_{T+} (upper triggering level). Similarly, when the input voltage is falling, the level of the output changes when the input passes through a specific voltage V_{T-} (lower triggering level), the level of the output changes. V_{T+} (upper triggering level) is always greater than V_{T-} (lower triggering level). The difference of these two voltages is known as hysteresis as shown in figure 4.16.

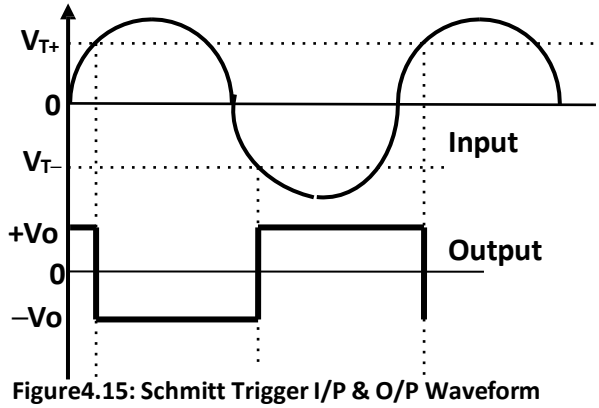


Figure4.15: Schmitt Trigger I/P & O/P Waveform

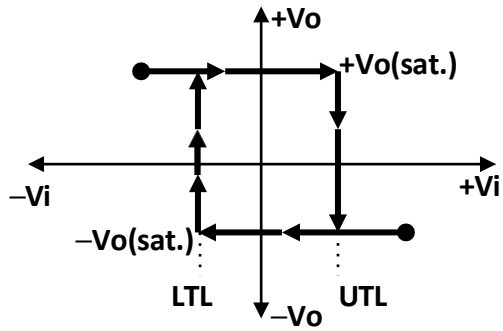


Figure4.16: Hysteresis Loop

Schmitt Trigger circuits:

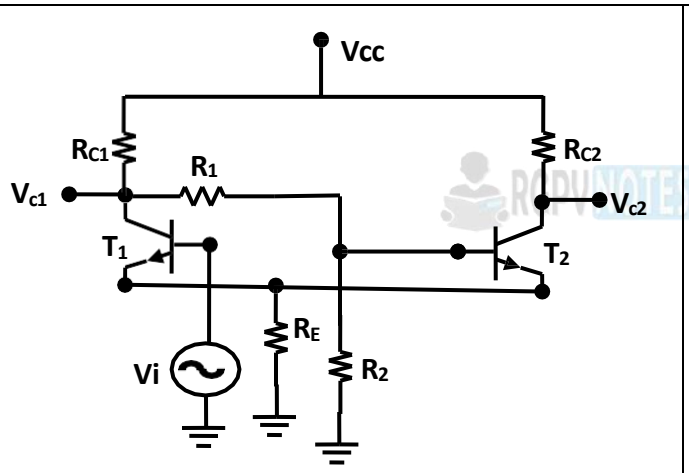


Figure4.17: Transistor Schmitt Trigger

From the figure 4.17 of transistor Schmitt trigger, resistor R_1 and R_2 are voltage divider resistors.

Operation: When Vi = 0V, when circuit is ON, T_2 is ON (V_{c2} = 0V). As T_2 is ON, there is a voltage drop across R_2. This drop acts as a reverse bias across emitter-base junction of T_1, due to which T_1 is OFF ($V_{c1} = V_{cc}$). This V_{cc} is coupled to base of T_2 through R_1. So T_2 is ON ie. In saturation($V_{c2} = V_{CE(sat)}$ = 0V).

When Vi = applied AC input, and approaches till it crosses V_{T+} (upper triggering level). Now, Vi > V_{T+} (upper triggering level), T_1 conducts. So, V_{c1} = 0V. This fall of voltage is coupled through resistor R_1 to the base of T_2, which reduces its forward bias voltage. So, T_1 = ON and T_2 = OFF ie. $V_{c1} = V_{CE(sat)}$ and $V_{c2} = V_{cc}$. The T_1 continues to conduct till the input voltage falls below V_{T-} (lower triggering level). When Vi > V_{T-} (lower triggering level), base-emitter junction of T_1 is reverse biased. So, T_1 is OFF. So, $V_{c1} = V_{cc}$ and $V_{c2} = V_{CE(sat)}$.

IC-555 TIMER:

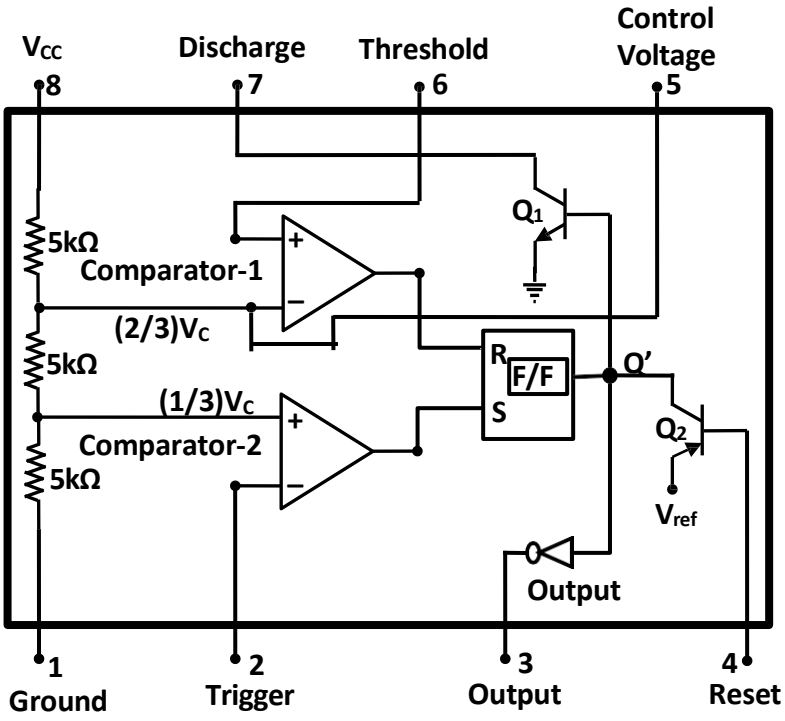


Figure 4.18: Functional block diagram of IC-555 Timer

Figure 4.18 shows the functional block diagram of IC-555 timer. It consists of a voltage divider network, which provides bias voltage of (2/3)Vcc to the inverting input of the comparator-1 and (1/3)Vcc to the non-inverting input of the comparator-2. These two voltages fix the comparator threshold voltage and also determine the timing interval. Electronically, possible to vary time by applying a modulation voltage to the control voltage input (pin-5).

If no such modulation is proposed, a 0.01µF capacitor is connected between control voltage and ground to bypass noise and ripple from supply. The other two inputs to the comparator are threshold and trigger inputs. The output of these two comparators, SET or RESET the flip flop, whose Q'output is fed to base of transistor Q_1. When Q' = high, Q_1 is ON and capacitor (externally connected between pin 7 and ground) will discharge.

The output stage is basically an inverting buffer stage used to provide a low output resistance and also to invert the flip flop output. Output stage has a capability of sourcing and sinking 200mA current. Q_2 (PNP transistor) whose emitter is connected to an internal reference voltage which is less than Vcc. When V_{ref} > Vcc (Pin-4 potential is less than Vcc), Q_2 is ON, which causes Q_1 to turn ON and output at pin-3 is brought to ground level.

Applications include oscillator, pulse generator, ramp and square wave generator, voltage monitor and may more applications.

<u>IC-555 AS MONOSTABLE MULTIVIBRATOR:</u>

Figure 4.19 (a) shows the circuit diagram of IC-555 as monostable multivibrator and (b) shows the waveform of trigger pulse, capacitor voltage and output pulse. Since it has only one stable state (output low), hence name monostable.

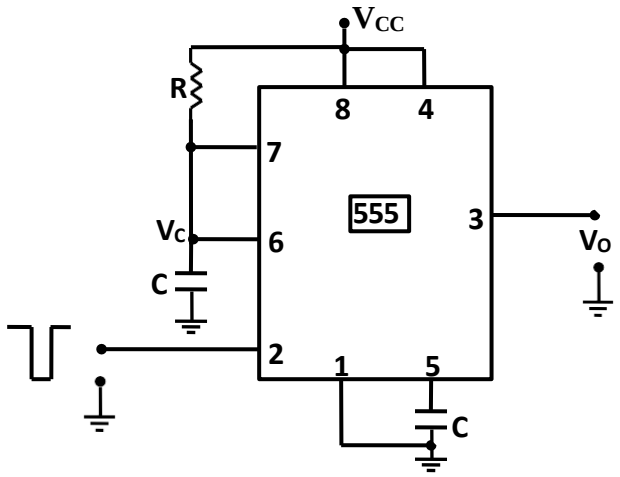


Figure 4.19(a): Monostable Multivibrator

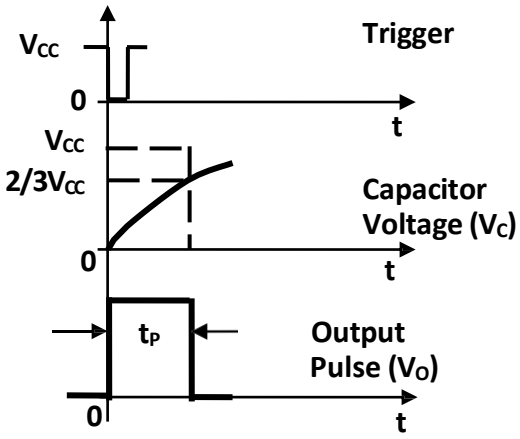


Figure 4.19(b): Waveforms

It is also called as one-shot multivibrator. From the circuit diagram, Pin-8 is connected to Vcc and pin-4 (reset pin) also connected to Vcc so that reset condition is disabled. The time interval for which the output remains high (t_P, pulse width) is decided by the external RC network. The capacitor C is connected between pin 7 and 1 so that it charges through the resistance R when the transistor Q_1 is OFF.

Operation: Initially, trigger pulse is high (Vcc), this drives the output of comparator-2 to low condition. As the capacitor C is in discharged state, pin-6 and 7 are at ground potential. The inputs to the flip flop will be S=R=0, hence Q' = high, so, Q_1 is ON, and C discharges to 0V ie. Vc = 0V. Since Q' = 1, output pin-3 = 0 is actually the stable state of multivibrator.

When the trigger input (negative trigger pulse) goes low (from Vcc to 0), comparator-2 output = high ie. S = 1. The comparator-1 output continue to be 0 ie. R=0, hence the flip flop is in set condition ie. Q' = 0, pin-3 = 1(High state). Since Q' = 0, transistor Q_1 is OFF and the capacitor C starts charging exponentially towards Vcc through the resistor R. When Vc becomes greater than ((2/3) Vcc), comparator-1 output changes form low to high ie. R = 1. Since the trigger input has returned back to Vcc from 0, comparator-2 output is equal to zero ie. S =0. So, S = 0 and R= 1, RS flip flop get RESET and Q' = 1. AS Q' = 1, transistor Q_1 = ON and capacitor C starts discharging towards zero through the transistor Q_1 and capacitor voltage Vc becomes zero. While discharging, when Vc < ((2/3) Vcc), the comparator-1 output goes to zero ie. R=0. Since the trigger input = Vcc, the comparator-2 output will be = 0 ie. S=0. Hence, S=0 and R=0, so no change in the Q' output condition and hence continuous to be High. Thus, pin-3 output = LOW (0-state).

The monostable multivibrator, thus goes from stable state into quasistable state and then returns back to the stable state after a time, $t_P = (1.1)R.C$

The output remains to be in LOW state until the next trigger pulse is applied to change the state.

IC-555 AS ASTABLE MULTIVIBRATOR:

Figure 4.20 (a) shows the circuit diagram of IC-555 as astable multivibrator and (b) shows the waveform of capacitor voltage and output pulse. Since no stable state, hence name astable.

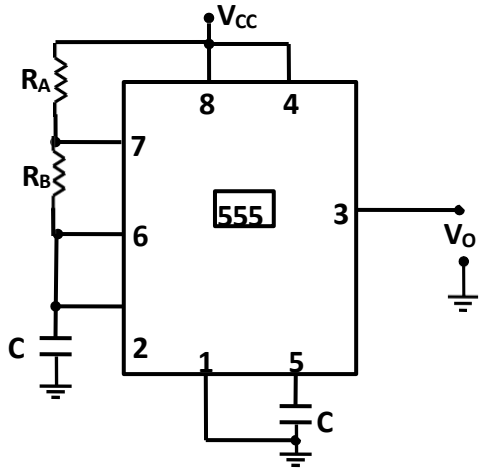


Figure 4.20(a): Astable Multivibrator

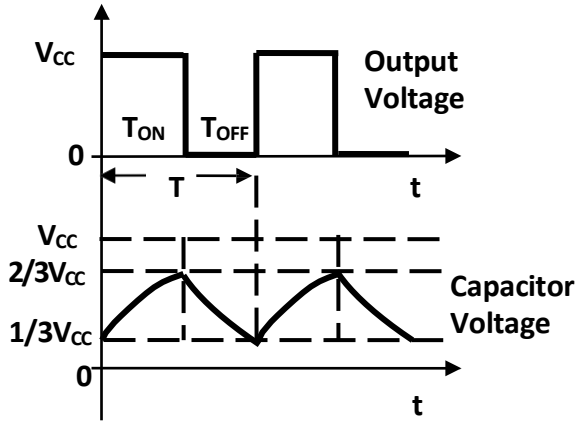


Figure 4.20(a): Waveforms

Astable multivibrator does not requires an external trigger pluse to change the output state, hence called as free-running multivibrator. The time duration for which the output will remain high or low is decided by the externally connected two resistors (R_A and R_B) and a capacitor (C).

Operation: Initially, when output is high (pin-3 = High), Flip flop output Q' = 0, hence transistor Q_1 is OFF. Now the capacitor C starts charging towards Vcc through R_A and R_B. As soon as the voltage across the capacitor Vc, becomes equal to[(2/3)Vcc], the comparator-1 output is high and will RESET the flip flop ie. Q' = 1. Hence the output = 0. As, Q' =1 , transistor Q_1 = ON and the capacitor C starts discharging through resistor R_B and transistor Q_1. During discharging mode of capacitor C, as soon as the voltage across the capacitor C becomes equal to [(1/3)Vcc], comparator-2 output will SET the flip flop, Q' = 0, and output = high. Then the cycle repeats.

Charging time duration of the capacitor C, is equal to the time the output is high is given by the expression: $t_C = T_{ON} = 0.69(R_A + R_B)C$

Discharging time duration of the capacitor C, is equal to the time the output is low is given by the expression: $t_d = T_{OFF} = 0.69(R_B)C$

Hence the total time period of output waveform: $T = t_C + t_d = T_{ON} + T_{OFF} = 0.69(R_A + 2R_B)C$

Hence, the frequency of oscillation is, $f_O = 1/T = \frac{1.45}{(R_A+2R_B)C}$

From the equation of frequency of oscillation fo, frequency is independent of the supply voltage Vcc.

Duty Cycle: Duty cycle is the ratio of the time during which the output is high (T_{ON}) to the total time period T.

% duty cycle = [T_{ON} / T] x 100 = $\frac{R_A+R_B}{R_A+2R_B} \times 100$

Applications: Astable multivibrator can be used to produce a square wave output. It can be used as a free running ramp generator.

DIGITAL IC LOGIC FAMILIES:

1. RTL- Resistor Transistor Logic
2. DTL-Diode Transistor Logic
3. TTL- Transistor Transistor Logic
4. ECL-Emitter Coupled Logic
5. I^2L- Integrated Injection Logic
6. PMOS- P-Channel Metal Oxide Semiconductor
7. NMOS- N-Channel Metal Oxide Semiconductor
8. CMOS- Complementary Metal Oxide Semiconductor

Characteristics of Digital IC's:

1. **FAN-IN**: Number of inputs connected to gate, without the degradation in the voltage levels.
2. **FAN-OUT**: Number of standard loads that the output of the gate can drive without degrading the normal operation.
3. **POWER DISSIPATION**: Power consumed by the gate, available from power supply.
4. **PROPAGATION DELAY**: Average transition time for the signal to propagate from input to output.
5. **NOISE MARGIN**: Noise margin is the limit of noise voltage which may be present without imparing or degrading the proper operation of the circuit.

TTL- Transistor Transistor Logic:

A) TTL-2 input NAND gate having totem pole (active pull-up) output stage:-

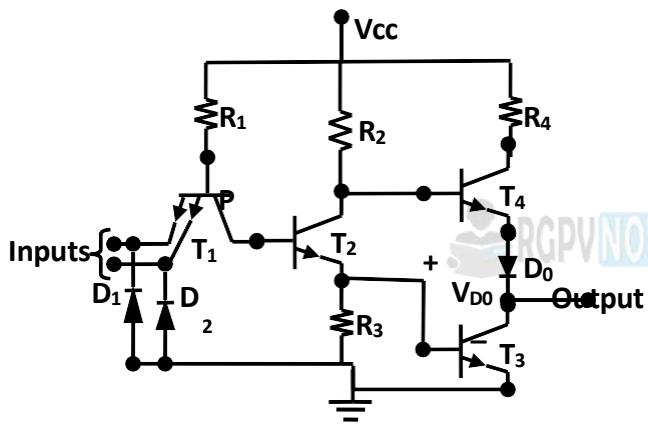


Figure 4.21: TTL- 2-Input NAND Gate(Totem Pole O/P)

The circuit diagram of a 2-input TTL NAND gate having an active pull-up (totem-pole) output stage is shown in figure 4.21. In this circuit, if one or both of the inputs are at logic 0, the corresponding B-E junction of T_1 will be forward biased, and the voltage at point P will become nearly equal to 0.7V which will keep the T_2 and T_3 OFF. (The voltage at P must be at least equal to 1.8V for turning T_2 and T_3 ON.) Therefore, the output voltage will be at logic 1, equal to Vcc – (drop across R_4)-(V_{CE} of T_4) – V_{D0}, which is nearly equal to 3.5V.

If both the inputs are held at logic 1 level, the B-E junctions of T_1 will be reverse biased, and the current flowing through R_1 and the C-B junction of T_1 will turn ON the transistors T_2 and T_3. Hence, the output voltage will be at logic 0, equal to V_{CEsat} of T_3. When the voltage at the input terminal corresponds to 1 level, the gate sinks an input current (reverse saturation current of the B-E junction of T_1), whereas when the voltage at the input terminal corresponds to 0 level, the gate sources an input current(forward current of the B-E junction of T_1).

Advantages: 1)when T4 is OFF and T3 is ON, no current through R4, so, no power dissipation. 2)If output is high, T4 is ON and T3 is OFF, hence T4 is acting in the emitter follower mode, its output impedance is low. Therefore output time constant for charging of any capacitive load is very short.	**Disadvantage:** T3 turns OFFmore slowly than T4 turns ON. So, before T3 completely turns OFF, T4 comes into conduction. So, for a very short duration of time both T3 and T4 are ON. This is called cross conduction and draws large current.

B) TTL- 2 input open-collector TTL NAND gate :

The circuit diagram of a 2-input open collector TTL NAND gate is shown in figure 4.23 and TTL-logic symbol (open collector) is shown in figure 4.22. Note that the collector of the transistor T_3 is floating. For the proper functioning of the device, this open collector terminal of T_3 must be tied to Vcc through resistor R, known as pull-up resistor (passive pull-up). Once a suitable pull-up resistor is connected, the characteristic of open collector and totem pole will be almost same.

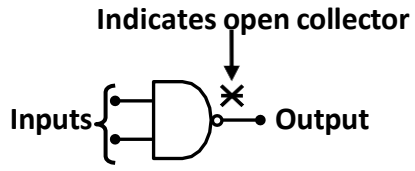


Figure 4.22: TTL- Logic Symbol (Open Collector)

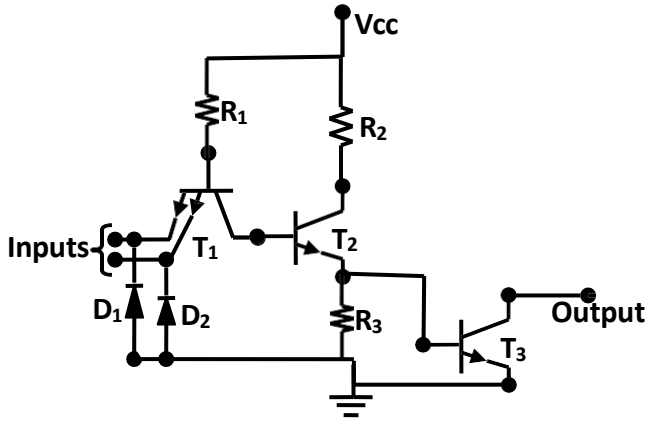


Figure 4.23: TTL- 2-Input NAND Gate(Open Collector O/P)

Advantage of open collector output is that wired ANDing becomes possible.
Wired ANDing as shown in figure 4.24, means tieing the outputs of gates together to obtain AND function.

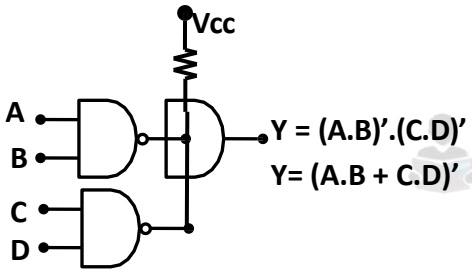


Figure 4.24: Wired ANDing of NAND gate O/P

It is possible to connect the outputs of two or more gates together.

Comparision of Totem-Pole and Open-Collector outputs:

Parameters	Totem-Pole	Open-collector
Circuit components on output side	T4 (pull up transistor), T3 (pull down transistor) and diode D0.	Only T3 (pull down transistor)
Wired ANDing	NO	Yes
External pull up resistor	Not required	Required
Power Dissipation	Low due to Pull up transistor.	High due to current flowing through external pull up resistor.
Speed	High	Low

C) TTL- TRI-STATE TTL Gate:

In a normal logic circuits there are two states of the output- LOW and HIGH. When a number of such outputs are connected to a common line, there are loading problems. To avoid this, tri-state outputs are used. In tri-state output circuits, there are three distinct states of which two are the logic 0 and logic 1 states and third is a high impedance state. In the figure 4.25 of a tri-state TTL inverter circuit, when control input is LOW, the drive is removed from T_1 and T_2 and the output is in the third state (High impedance). When the control input is high, the output is 1 or 0 depending on the input. Figure 4.26 logic symbol of tristate TTL inverter.

RGPVNOTES.IN

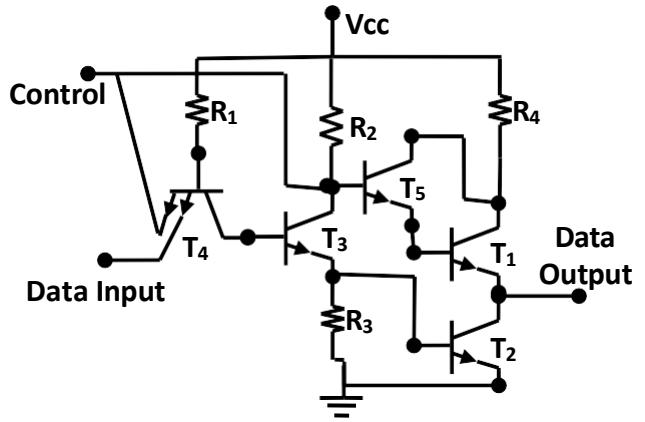


Figure4.25: Tri-state TTL inverter

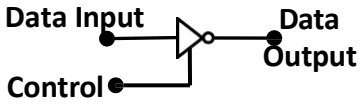
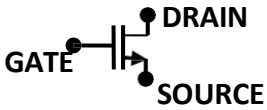
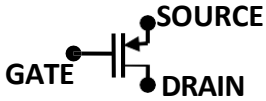
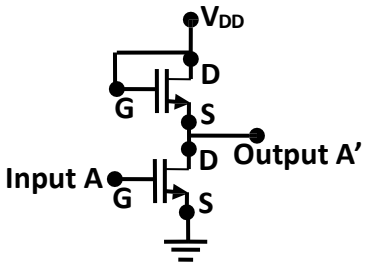
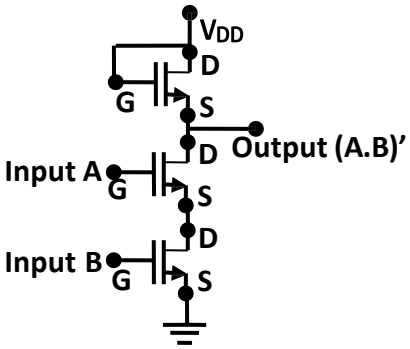


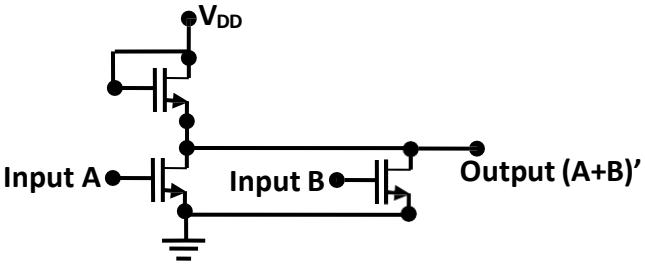
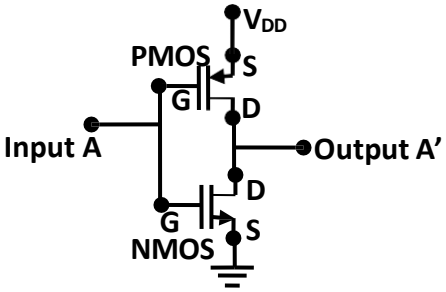
Figure 4.26 : Tri-state TTL inverter logic symbol

PMOS, NMOS, and CMOS logic:

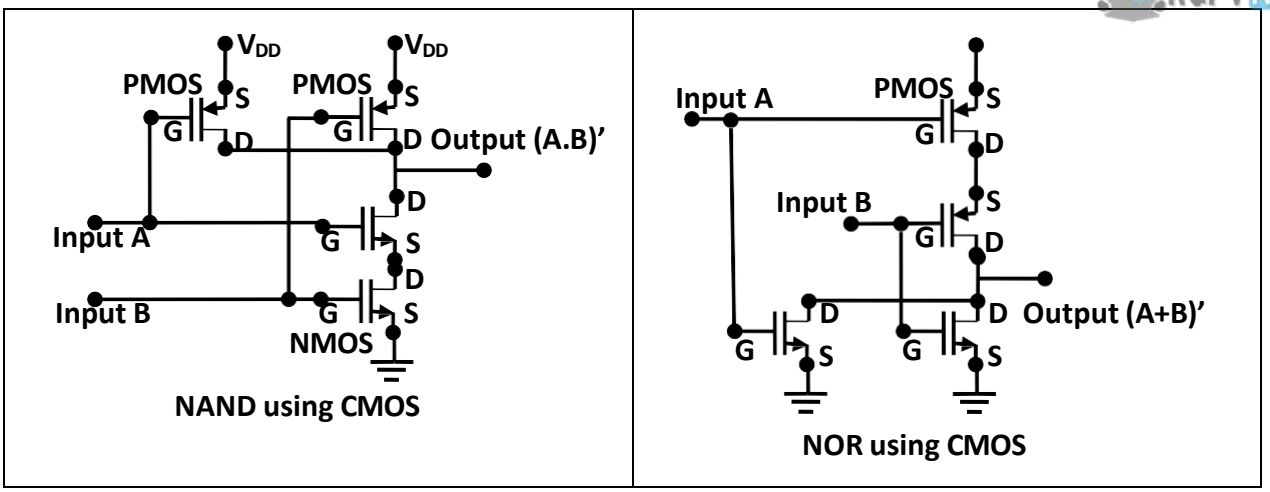
 Logic Symbol of NMOS	A NMOS switch is closed when controlling signal is HIGH. An arrow indicates the direction of positive current flow from drain (D) to source (S) in NMOS.
 Logic Symbol of PMOS	A PMOS switch is closed when controlling signal is LOW. An arrow indicates the direction of positive current flow from source (S) to drain (D) in PMOS. RGPVNOTES.IN

Realization of logic gates using MOS logic family:

 Inverter using NMOS	 NAND using NMOS

 NOR using NMOS	 Inverter using CMOS

RGPVNOTES.IN



Advantages of MOS logic: 1)Low power dissipation. 2)Excellent noise immunity. 3)High packing density. 4)Wide range of supply voltages (+3V to +18V)

INTERFACING BETWEEN TTL to MOS:

Interfacing refers the way a driving device is connected to a loading device. Here TTL is the driving device and CMOS is the loading device. TTL device needs a supply voltage of 5V and CMOS needs of +3V to +15V.

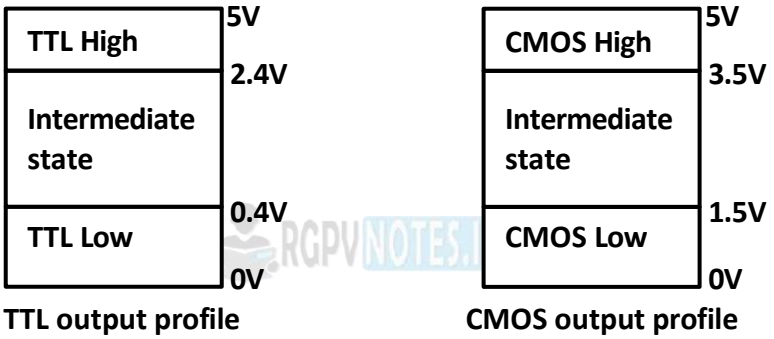
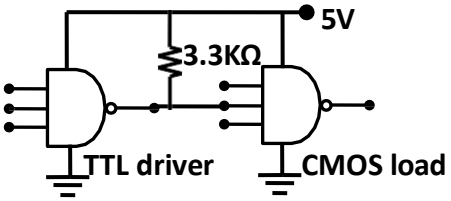
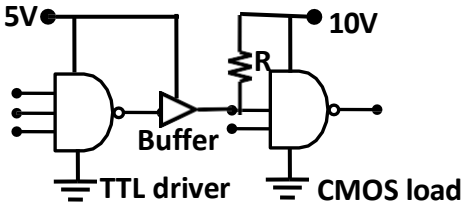


Figure 4.26: Outprofile of TTL and CMOS

Figure 4.26 shows the output profile of TTL and CMOS, a TTL low fits inside the CMOS low ie. CMOS load always interprets the TTL low state drive as a low. The problem is with TTL high state ie. there is indeterminate action ie. no reliable operation. So, the standard solution is to use a pull up resistor between TTL driver and CMOS load. It raises the high state to approximate 15V.

 **Figure 4.27:TTL driver & CMOS interface**	In figure 4.27, When TTL output is low, 3.3K resistor is grounded, therefore TTL driver sink current of 1.52mA. when TTL output is high, pull up resistor, output raises above +2.4V ie. The supply voltage is pulling up the output upto +5V (1.52mAx3.3K). If more than one TTL chip is being interfaced to the CMOS load, connect each TTL driver to a separate pull-up resistor.
 **Figure4.28:TTL driving high voltage CMOS**	If CMOS IC is operated with V_{DD} greater than 5V then the TTL driving high voltage CMOS design is used as shown in figure 4.28. Buffer is used to interface between TTL and CMOS. IC7406 as inverting buffer has an output voltage rating of 30V.

RGPVNOTES.IN

Unit 5

Syllabus:

Introduction to Digital Communication: Nyquist sampling theorem, time division multiplexing, PCM, quantization error, introduction to BPSK & BFSK modulation schemes, Shannon's theorem for channel capacity.

5.1 Introduction to Digital Communication

In digital communication, the information is first converted in to the digital form and then transmitted. First the signal is converted in to electrical form using an input transducer. Then the analog signal obtained is quantized as well as sampled to get the digital signal. The sampling rate and the number of quantization levels thus determine the quality of the digital signal. More the number of samples per seconds as well as more the quantization levels will be there, better the quality of the digital signal. The block diagram of a digital communication system is explained in figure 5.1.1.

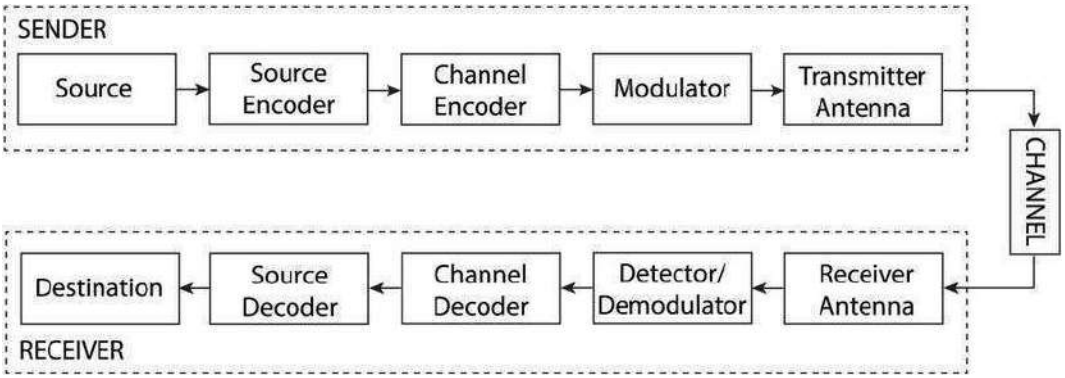


Figure 5.1.1 Digital Communication System

The information generated by the source is coded by a source coder. The aim of the source coding is to represent the given information in minimum number of bits per sample. Then channel coder allows the maximum utilization of the channel, through which the information is to be transmitted. Then by selecting a suitable modulation technique the data is modulated and transmitted using the transmitter antenna or through the media.

At the receiver the received signal is first sent to the demodulator, which demodulates the signal and then data is decoded by the channel decoder and the source decoder. Now the information received is in its digital form which can be extracted after conversion in to its original form.

5.2 Sampling

Sampling is defined as, "The process of measuring the instantaneous values of continuous-time signal in a discrete form."

Sample is a piece of data taken from the whole data which is continuous in the time domain. When a source generates an analog signal and if that has to be digitized, having 1s and 0s i.e., High or Low, the signal has to be discretized in time. This discretization of analog signal is called as Sampling. The following figure indicates a continuous-time signal x (t) and a sampled signal x_s (t). When x (t) is multiplied by a periodic impulse train, the sampled signal x_s (t) is obtained.

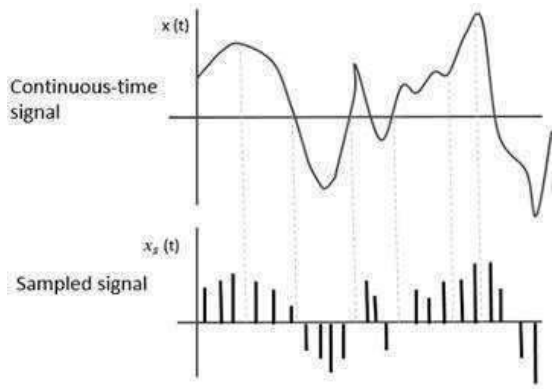


Figure 5.2.1 Sampling

Sampling Rate

To discretize the signals, the gap between the samples should be fixed. That gap can be termed as a sampling period Ts.

$$Sampling\ Frequency = 1/Ts = fs$$

Where,

$$Ts = \ sampling\ time$$
$$fs = \ sampling\ frequency\ or\ the\ sampling\ rate$$

Sampling frequency is the reciprocal of the sampling period. This sampling frequency can be simply called as Sampling rate. The sampling rate denotes the number of samples taken per second, or for a finite set of values.

Nyquist Rate

For an analog signal to be reconstructed from the digitized signal, the sampling rate should be highly considered. The rate of sampling should be such that the data in the message signal should neither be lost nor it should get over-lapped. Hence, a rate was fixed for this, called as Nyquist rate.

Suppose that a signal is band-limited with no frequency components higher than W Hertz. That means, W is the highest frequency. For such a signal, for effective reproduction of the original signal, the sampling rate should be twice the highest frequency.

Which means, $fS = 2\mathrm{W}$

Where,

$$f_S = \ sampling\ rate$$
$$\mathrm{W} = \ highest\ frequency$$

This rate of sampling is called as Nyquist rate. A theorem called, Sampling Theorem, was stated on the theory of this Nyquist rate.

5.2.1 Nyquist Sampling Theorem

The sampling theorem, i.e. Nyquist theorem, states that, "a signal can be recovered exactly from its samples if the sampling rate f_s is greater or equals to the twice the maximum frequency W."

$$f_s \geq 2\mathrm{W}$$

Proof: Consider a continuous time signal x(t). The spectrum of x(t) is a band limited to f_m Hz i.e. the spectrum of x(t) is zero for $|\omega|>\omega_m$.

Sampling of input signal x(t) can be obtained by multiplying x(t) with an impulse train δ(t) of period T_s. The output of multiplier is a discrete signal called sampled signal which is represented with y(t) in the following diagrams:

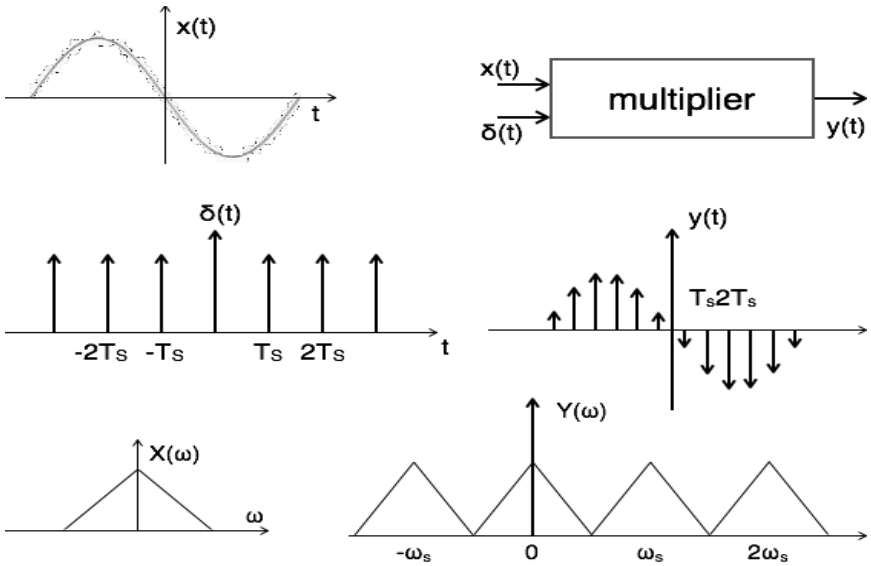


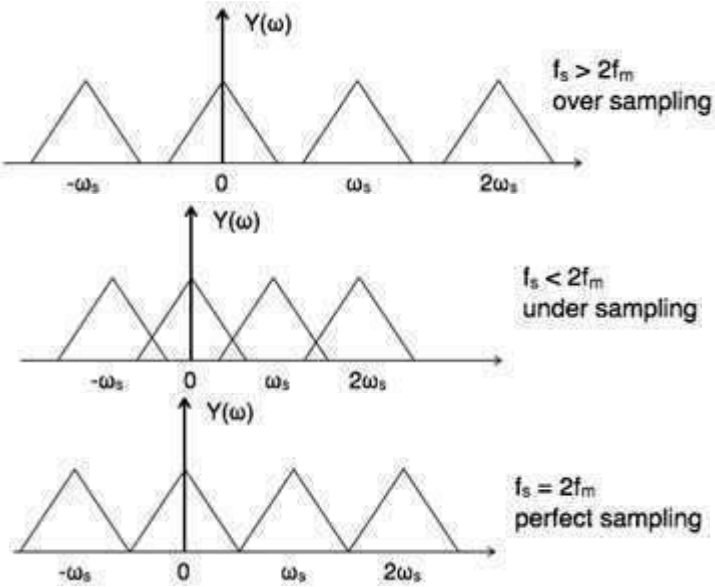
Figure 5.2.2 Sampling of signal x(t)

Here, you can observe that the sampled signal takes the period of impulse. Possibility of sampled frequency spectrum with different conditions is given by the following diagrams:

(a) Over Sampling

(b) Under Sampling

(c) Perfect sampling



Therefore for proper reproduction of the signal from its samples, the sampling frequency must be atleast twice of the maximum frequency of the message signal.

5.2.2 Signals Sampling Techniques

There are three types of sampling techniques:

- Impulse sampling.
- Natural sampling.
- Flat Top sampling.

(a) Impulse Sampling

Impulse sampling can be performed by multiplying input signal x(t) with impulse train $\mathrm{Z}\infty n = -\infty \eth(t - nT)$ of period 'T'. Here, the amplitude of impulse changes with respect to amplitude of input signal x(t). The output of sampler is given by

RGPVNOTES.IN

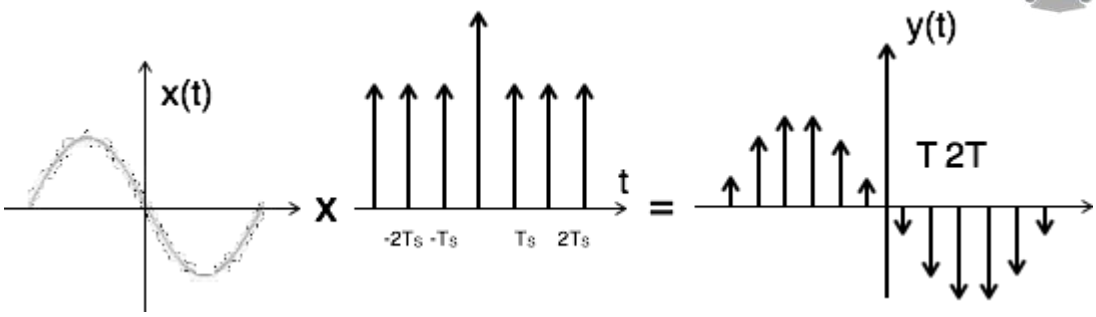


Figure 5.2.2.1 Impulse Sampling

This is called ideal sampling or impulse sampling. You cannot use this practically because pulse width cannot be zero and the generation of impulse train is not possible practically.

(b) Natural Sampling

Natural sampling is similar to impulse sampling, except the impulse train is replaced by pulse train of period T. i.e. you multiply input signal x(t) to pulse train $\sum_{n=-\infty}^{\infty} P(t-nT)$ as shown below

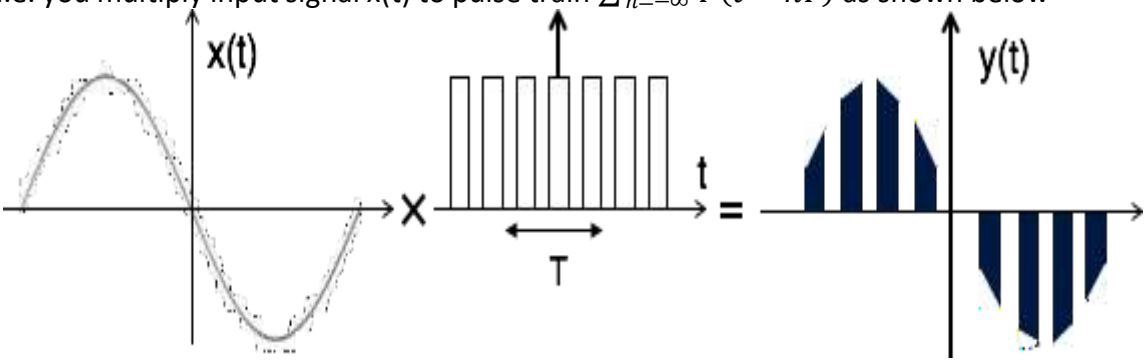


Figure 5.2.2.2 Natural Sampling

(c) Flat Top Sampling

During transmission, noise is introduced at top of the transmission pulse which can be easily removed if the pulse is in the form of flat top. Here, the top of the samples are flat i.e. they have constant amplitude. Hence, it is called as flat top sampling or practical sampling. Flat top sampling makes use of sample and hold circuit.

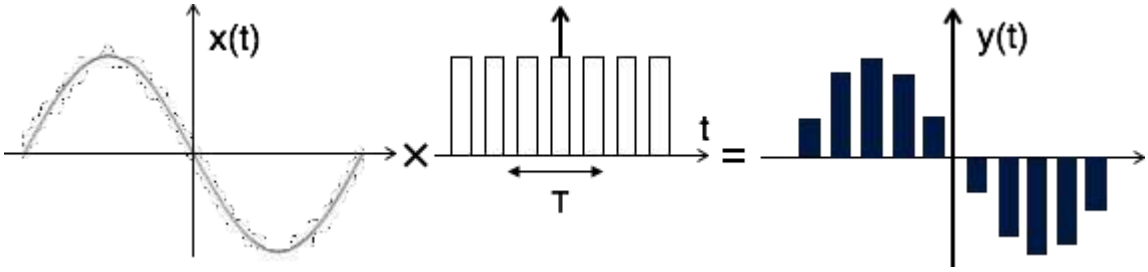


Figure 5.2.2.3 Flat Top Sampling

5.3 Time Division Multiplexing

The sampling theorem allows us to multiplex the samples. That is, the transmission of the message samples engages the communication channel for only a fraction of the sampling interval, and for the rest of the interval transmitting the samples of the other signals. We thereby obtain a time-division multiplex (TDM) system, which enables the joint utilization of a common communication channel by a number of independent message sources without mutual interference among them. The concept of TDM is illustrated by the block diagram shown in Fig. 5.3.1. Each input message signal is first restricted in bandwidth by a low-pass anti-aliasing filter to remove the frequencies that are nonessential to an adequate signal representation. The low-pass filter outputs are then applied to a commutator, which is usually, implemented using electronic switching circuitry. The function of the commutator is twofold: (1) to take a

narrow sample of each of the N input messages at a rate that is slightly higher than Nyquist rate 2W, where W is the cutoff frequency of the anti-aliasing filter, and (2) to sequentially interleave these N samples inside the sampling interval Indeed, this latter function is the essence of the time-division multiplexing operation.

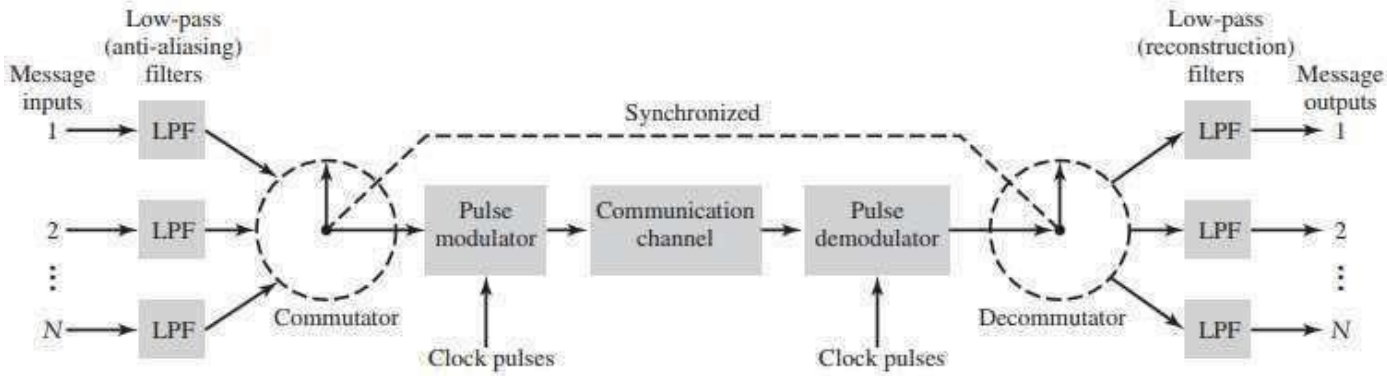


Figure 5.3.1 Time Division Multiplexing

Following the commutation process, the multiplexed signal is applied to a pulse modulator, the purpose of which is to transform the multiplexed signal into a form suitable for transmission over the common channel. It is clear that the use of TDM introduces a bandwidth expansion factor N, because the scheme must squeeze N samples derived from N independent message sources into a time slot equal to one sampling interval. At the receiving end of the system, the received signal is applied to a pulse demodulator, which performs the reverse operation of the pulse modulator. The narrow samples are distributed to the appropriate low-pass reconstruction filters by means of a decommutator, which operates in synchronism with the commutator in the transmitter. This synchronization is essential for a satisfactory operation of the system. Thus the samples are separated at the receiver section.

5.4 Pulse Code Modulation:

A signal which is to be quantized before transmission is sampled as well. The quantization is used to reduce the effect of noise and the sampling allows us to do the time division multiplexing. The combined operation of sampling and quantization generate a quantized PAM waveform i.e. a train of pulses whose amplitude is restricted to a number of discrete levels.

Rather than transmitting the sampled values itself, we may represent each quantization level by a code number and transmit the code number. Most frequently the code number is converted in to binary equivalent before transmission. Then the digits of the binary representation of the code are transmitted as pulses. This system of transmission is called binary **Pulse Code Modulation**. The whole process can be understood by the figure 5.4.1.

PCM Transmitter:

Basic Blocks:

1. Anti aliasing Filter, 2. Sampler, 3. Quantizer, 4. Encoder

An anti-aliasing filter is basically a filter used to ensure that the input signal to sampler is free from the unwanted frequency components. For most of the applications these are low-pass filters. It removes the frequency components of the signal which are above the cutoff frequency of the filter. The cutoff frequency of the filter is chosen such it is very close to the highest frequency component of the signal.

Sampler unit samples the input signal and these samples are then fed to the Quantizer which outputs the quantized values for each of the samples. The quantizer output is fed to an encoder which generates the binary code for every sample.

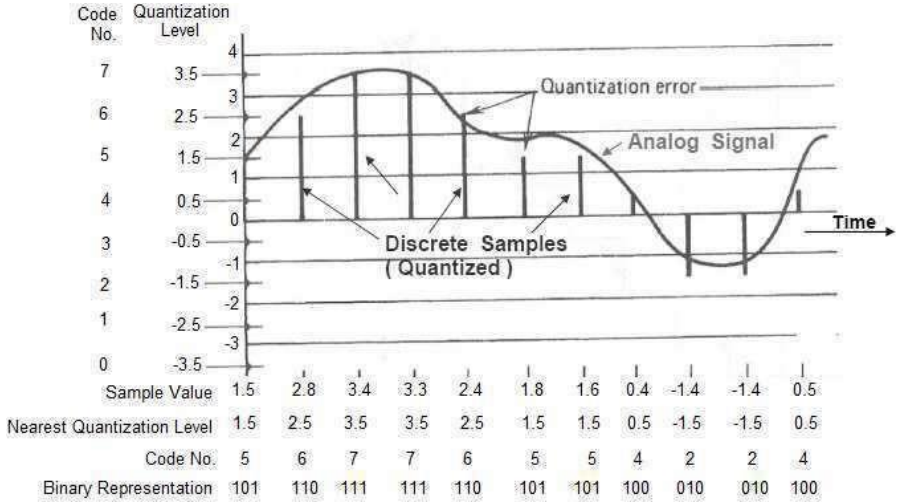


Figure 5.4.1 Pulse Code Modulation

The quantizer and encoder together are called as analog to digital converter.

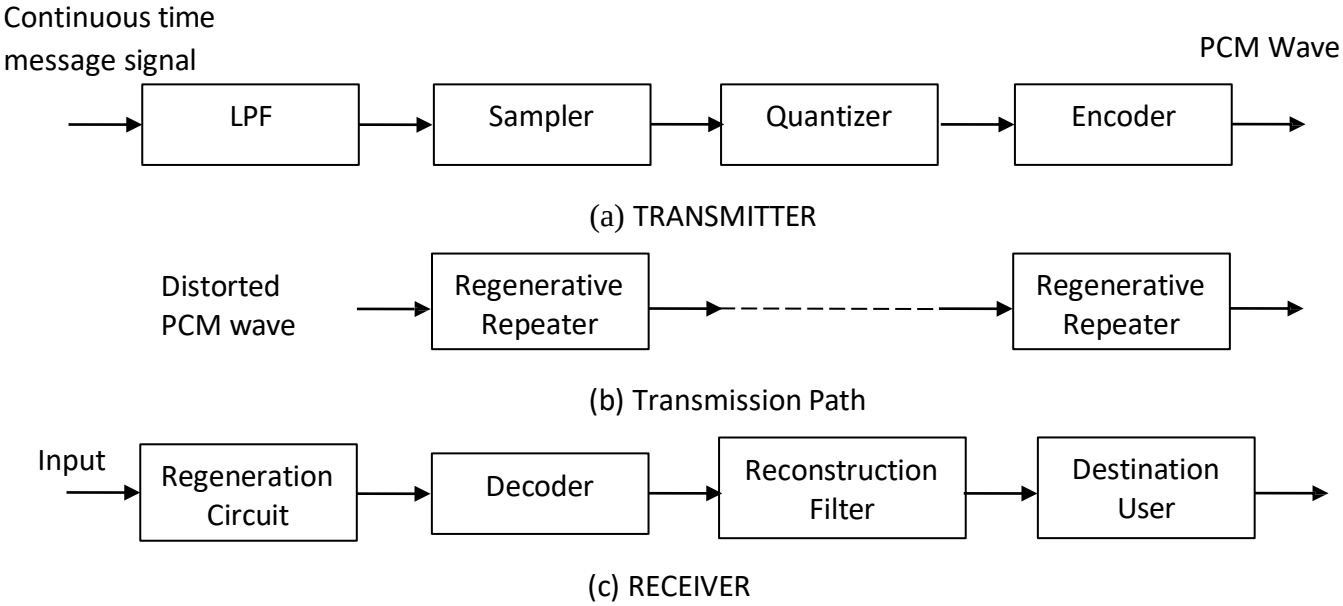


Figure 5.4.2 PCM System Basic Block Diagram

Advantages of Pulse Code Modulation:

- Pulse code modulation will have low noise addition and data loss is also very low.
- Pulse code modulation is used in music play back CD's and also used in DVD for data storing whose sampling rate is bit higher.
- Pulse code modulation can be used in storing the data.
- PCM can encode the data also.
- Multiplexing of signals can also be done using pulse code modulation. Multiplexing is nothing for adding the different signals and transmitting the signal at same time.
- Pulse code modulation permits the use of pulse regeneration.

Disadvantages:

- Pulse code modulation requires large bandwidth
- Specialized circuitry is required for transmitting and also for quantizing the samples at same quantized levels.
- We can do encoding using pulse code modulation but we need to have complex and special circuitry.
- Pulse code modulation receivers are cost effective when we compared to other modulation receivers.

- Developing pulse code modulation is bit complicated and checking the transmission quality is also difficult and takes more time.
- Channel bandwidth should be more for digital encoding.
- PCM systems are complicated when compared to analog modulation methods and other systems.
- Decoding also needs special equipment's and they are also too complex.

Applications of Pulse Code Modulation (PCM):

- Pulse code modulation is used in telecommunication systems, air traffic control systems etc.
- Pulse code modulation is used in compressing the data that is why it is used in storing data in optical disks like DVD, CDs etc. PCM is even used in the database management systems.
- Pulse code modulation is used in mobile phones, normal telephones etc.
- Remote controlled cars, planes, trains use pulse code modulations.

5.5 Quantization Error

Quantization: In the process of quantization we create a new signal $m_q(t)$, which is an approximation to $m(t)$. The quantized signal $m_q(t)$, has the great merit that it is separable from the additive noise.
The operation of quantization is represented in figure 2.7.1. Here we have a signal m(t), whose amplitude varies in the range from V_H to V_L as shown in the figure.
We have divided the total range in to M equal intervals each of size S, called the step size and given by

$$S = \Delta = \frac{(V_H - V_L)}{M}$$

In our example M=8. In the centre of each of this step we located quantization levels $m0, m1, m2, \ldots m7$. The $m_q(t)$ is generated in the following manner-
Whenever the signal $m(t)$ is in the range Δ_0, the signal $m_q(t)$ maintains a constant level m_0, whenever the signal $m(t)$ is in the range Δ_1, the signal $m_q(t)$ maintains a constant level m_1 and so on. Hence the signal $m_q(t)$ will found all times to one of the levels $m0, m1, m2, \ldots m7$. The transition in $m_q(t)$ from $m0$ to $m1$ is made abruptly when $m(t)$ passes the transition level L_{01}, which is mid way between m_0 and m_1 and so on.

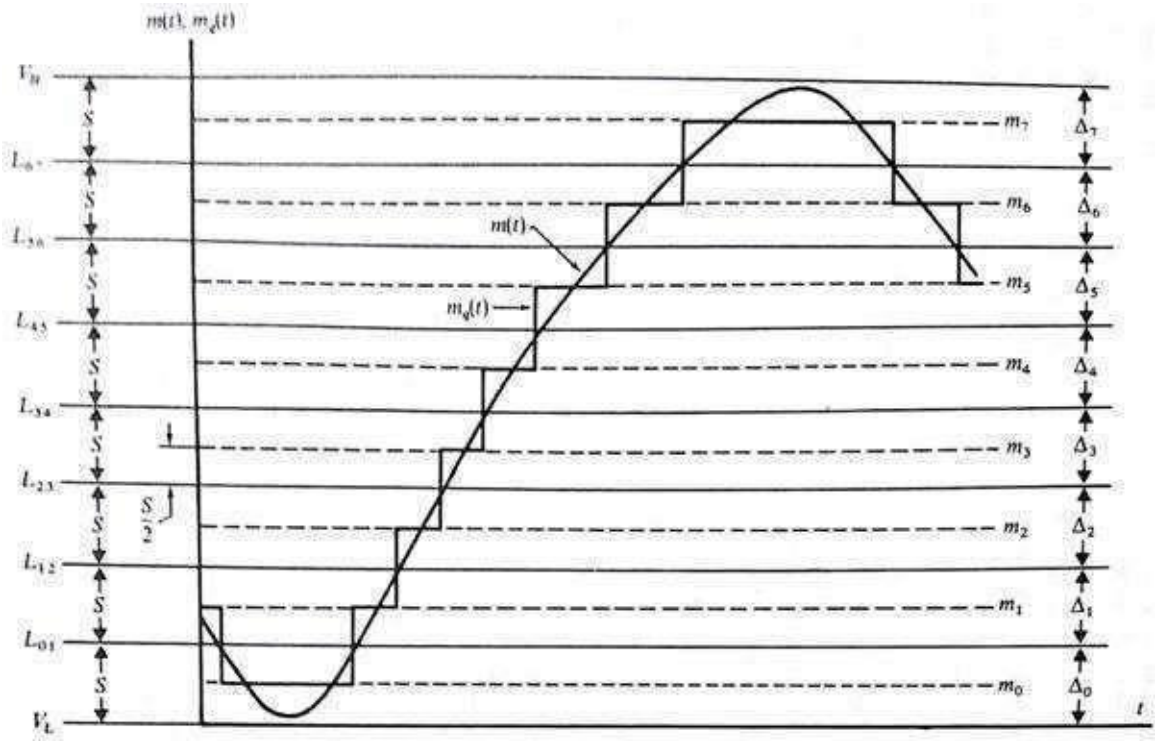


Figure 5.5.1 Quantization of signals

Using quantization of signals, the effect of noise can be reduced significantly. The difference between $m(t)$ and $m_q(t)$ can be regarded as noise and is called quantization noise.

$$quantization\ noise = m(t) - m_q(t)$$

Also the quantized signal and original signal differs from one another in a ransom manner. This difference or error due to quantization process is called quantization error and is given by

$$e = m(t) - m_k$$

when $m(t)$ happens to be close to quantization level m_k, quantizer output will be m_k.

Quantization Error:

For any system, during its functioning, there is always a difference in the values of its input and output. The processing of the system results in an error, which is the difference of those values.

The difference between an input value and its quantized value is called a Quantization Error. A Quantizer is a logarithmic function that performs Quantization (rounding off the value). An analog-to-digital converter (ADC) works as a quantizer.

The following figure illustrates an example for a quantization error, indicating the difference between the original signal and the quantized signal.

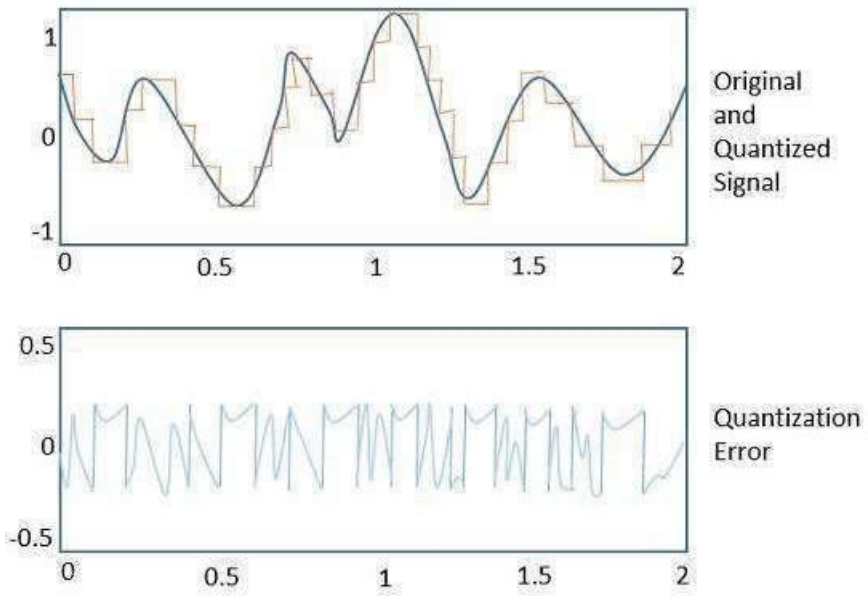


Figure 5.5.2 Quantization Error

Quantization Noise:

It is a type of quantization error, which usually occurs in analog audio signal, while quantizing it to digital. For example, in music, the signals keep changing continuously, where regularity is not found in errors. Such errors create a wideband noise called as Quantization Noise.

5.6 Digital Modulation

It provides more information capacity, high data security, quicker system availability with great quality communication. Hence, digital modulation techniques have a greater demand, for their capacity to convey larger amounts of data than analog modulation techniques.

There are many types of digital modulation techniques and also their combinations, depending upon the need. Of them all, we will discuss the prominent ones.

ASK – Amplitude Shift Keying

The amplitude of the resultant output depends upon the input data whether it should be a zero level or a variation of positive and negative, depending upon the carrier frequency.

FSK – Frequency Shift Keying

The frequency of the output signal will be either high or low, depending upon the input data applied.

PSK – Phase Shift Keying

The phase of the output signal gets shifted depending upon the input. These are mainly of two types, namely Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK), according to the number of phase shifts. The other one is Differential Phase Shift Keying (DPSK) which changes the phase according to the previous value.

Amplitude Shift Keying (ASK) is a type of Amplitude Modulation which represents the binary data in the form of variations in the amplitude of a signal.
Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input.
The following figure represents ASK modulated waveform along with its input.

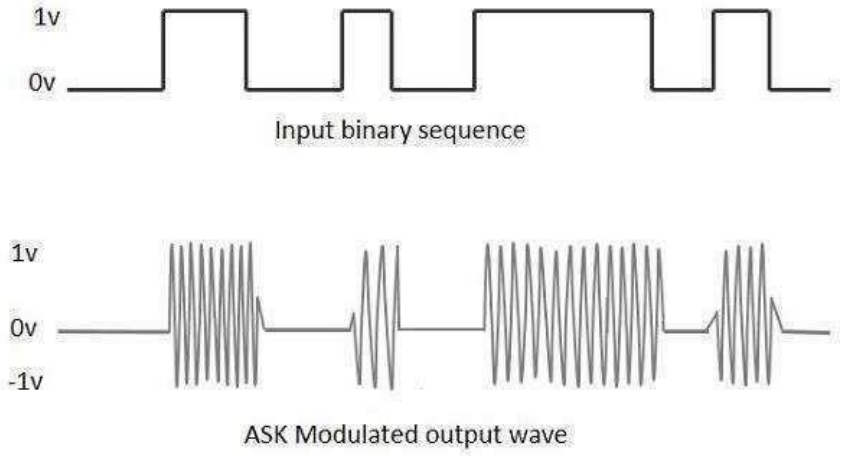


Figure 5.6.1 ASK Modulation

To find the process of obtaining this ASK modulated wave, let us learn about the working of the ASK modulator.

ASK Modulator
The ASK modulator block diagram comprises of the carrier signal generator, the binary sequence from the message signal and the band-limited filter. Following is the block diagram of the ASK Modulator.

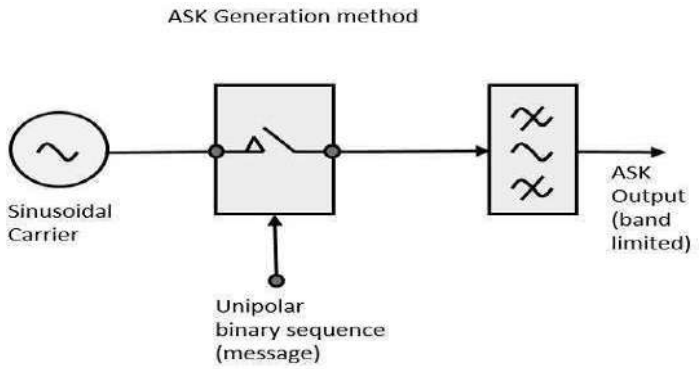


Figure 5.6.2 ASK Modulator

The carrier generator sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes the switch, allowing a carrier wave. Hence, the output will be the carrier signal at high input. When there is low input, the switch opens, allowing no voltage to appear. Hence, the output will be low.
The band-limiting filter, shapes the pulse depending upon the amplitude and phase characteristics of the band-limiting filter or the pulse-shaping filter.

ASK Demodulator
There are two types of ASK Demodulation techniques. They are −

- Asynchronous ASK Demodulation/detection

- Synchronous ASK Demodulation/detection

The clock frequency at the transmitter when matches with the clock frequency at the receiver, it is known as a Synchronous method, as the frequency gets synchronized. Otherwise, it is known as Asynchronous.

Asynchronous ASK Demodulator

The Asynchronous ASK detector consists of a half-wave rectifier, a low pass filter, and a comparator. Following is the block diagram for the same.

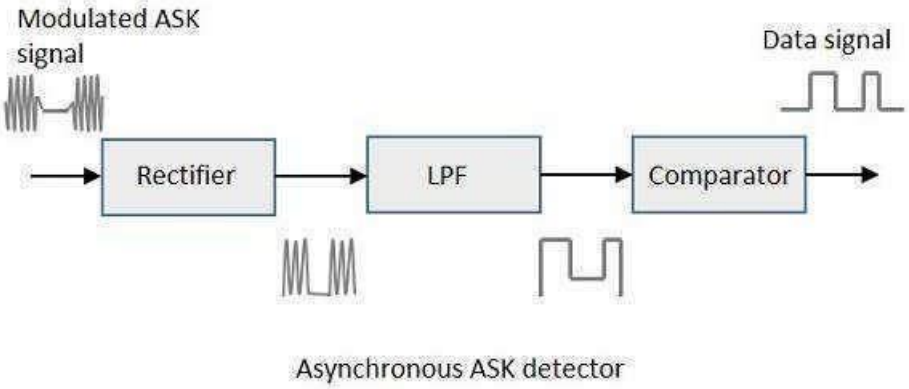


Figure 5.6.3 ASK Demodulator

The modulated ASK signal is given to the half-wave rectifier, which delivers a positive half output. The low pass filter suppresses the higher frequencies and gives an envelope detected output from which the comparator delivers a digital output.

Synchronous ASK Demodulator

Synchronous ASK detector consists of a Square law detector, low pass filter, a comparator, and a voltage limiter. Following is the block diagram for the same.

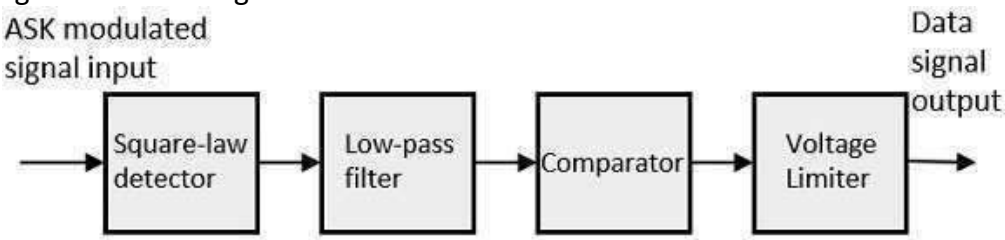


Figure 5.6.4 Synchronous ASK Demodulator

The ASK modulated input signal is given to the Square law detector. A square law detector is one whose output voltage is proportional to the square of the amplitude modulated input voltage. The low pass filter minimizes the higher frequencies. The comparator and the voltage limiter help to get a clean digital output.

5.7 Frequency Shift Keying (FSK)

FSK is the digital modulation technique in which the frequency of the carrier signal varies according to the digital signal changes. FSK is a scheme of frequency modulation. The output of a FSK modulated wave is high in frequency for a binary High input and is low in frequency for a binary Low input. The binary 1s and 0s are called Mark and Space frequencies. The following image is the diagrammatic representation of FSK modulated waveform along with its input.

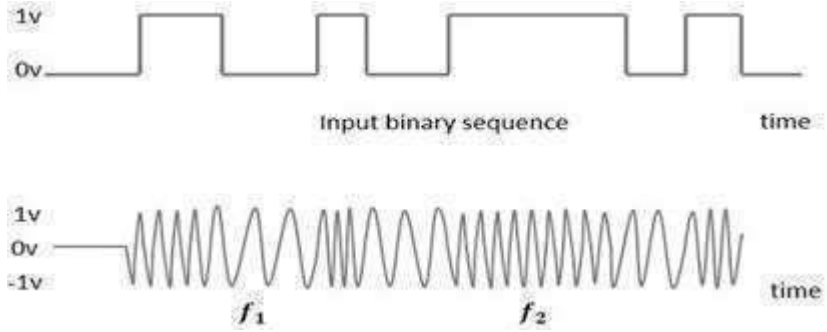


Figure 5.7.1 Frequency Shift Keying (FSK)

To find the process of obtaining this FSK modulated wave, let us know about the working of a FSK modulator.

FSK Modulator

The FSK modulator block diagram comprises of two oscillators with a clock and the input binary sequence. Following is its block diagram.

FSK Transmitter

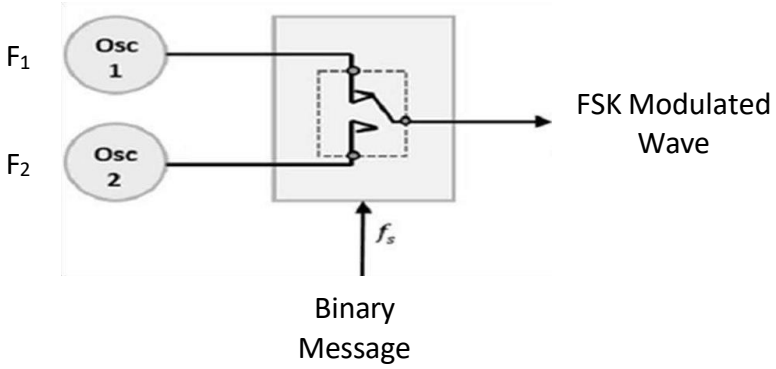


Figure 5.7.2 FSK Modulator

The two oscillators, producing a higher and a lower frequency signals, are connected to a switch along with an internal clock. To avoid the abrupt phase discontinuities of the output waveform during the transmission of the message, a clock is applied to both the oscillators, internally. The binary input sequence is applied to the transmitter so as to choose the frequencies according to the binary input.

FSK Demodulator

There are different methods for demodulating a FSK wave. The main methods of FSK detection are asynchronous detector and synchronous detector. The synchronous detector is a coherent one, while asynchronous detector is a non-coherent one.

Asynchronous FSK Detector

The block diagram of Asynchronous FSK detector consists of two band pass filters, two envelope detectors, and a decision circuit. Following is the diagrammatic representation.

RGPVNOTES.IN

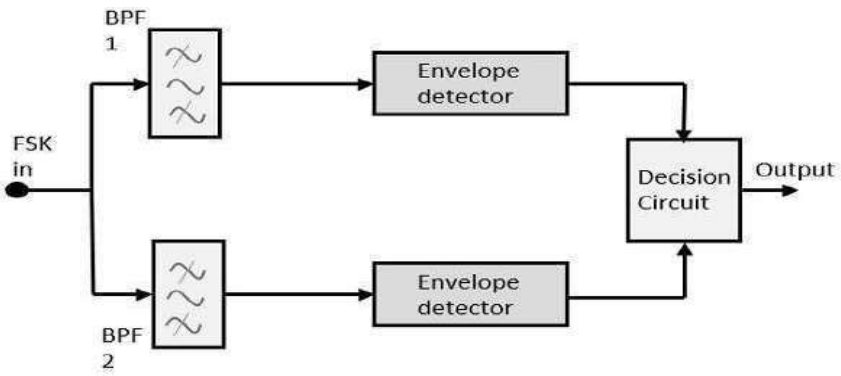


Figure 5.7.3 Asynchronous FSK Detector

The FSK signal is passed through the two Band Pass Filters (BPFs), tuned to Space and Mark frequencies. The output from these two BPFs look like ASK signal, which is given to the envelope detector. The signal in each envelope detector is modulated asynchronously.

The decision circuit chooses which output is more likely and selects it from any one of the envelope detectors. It also re-shapes the waveform to a rectangular one.

Synchronous FSK Detector

The block diagram of Synchronous FSK detector consists of two mixers with local oscillator circuits, two band pass filters and a decision circuit. Following is the diagrammatic representation.

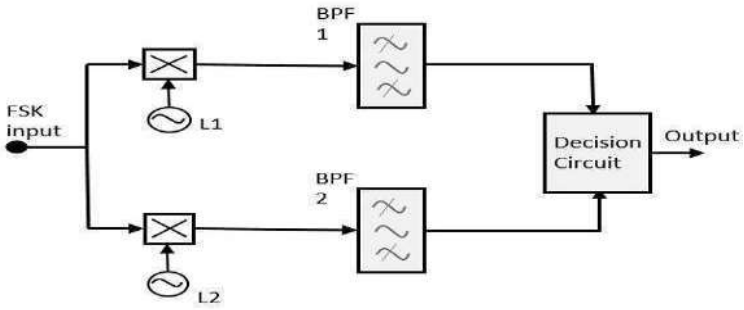


Figure 5.7.4 Synchronous FSK Detector

The FSK signal input is given to the two mixers with local oscillator circuits. These two are connected to two band pass filters. These combinations act as demodulators and the decision circuit chooses which output is more likely and selects it from any one of the detectors. The two signals have a minimum frequency separation.

For both of the demodulators, the bandwidth of each of them depends on their bit rate. This synchronous demodulator is a bit complex than asynchronous type demodulators.

5.8 Phase Shift Keying (PSK)

PSK is the digital modulation technique in which the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time. PSK technique is widely used for wireless LANs, bio-metric, contactless operations, along with RFID and Bluetooth communications.

PSK is of two types, depending upon the phases the signal gets shifted. They are −

Binary Phase Shift Keying (BPSK)

This is also called as 2-phase PSK or Phase Reversal Keying. In this technique, the sine wave carrier takes two phase reversals such as 0° and 180°.

BPSK is basically a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, for message being the digital information.

BPSK Modulator

The block diagram of Binary Phase Shift Keying consists of the balance modulator which has the carrier sine wave as one input and the binary sequence as the other input. Following is the diagrammatic representation.

BPSK Modulator

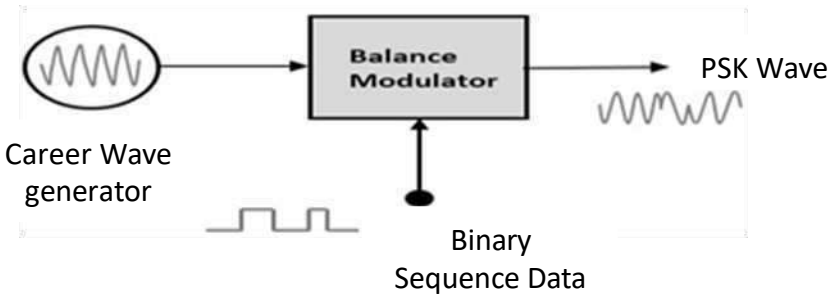


Figure 5.8.1 BPSK Modulator

The modulation of BPSK is done using a balance modulator, which multiplies the two signals applied at the input. For a zero binary input, the phase will be 0° and for a high input, the phase reversal is of 180°. Following is the diagrammatic representation of BPSK Modulated output wave along with its given input.

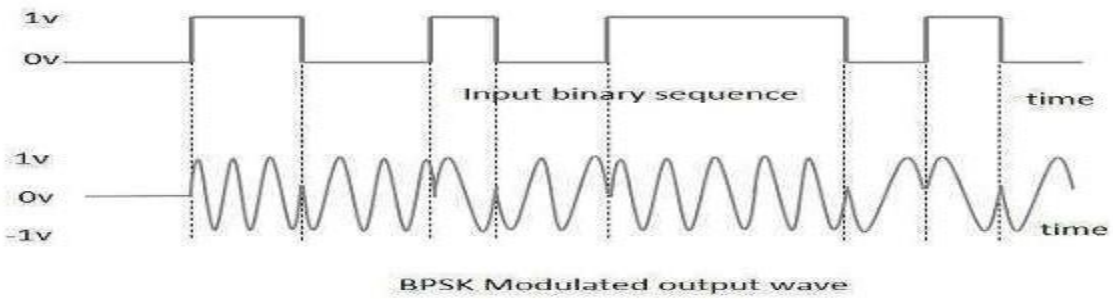


Figure 5.8.2 BPSK Modulated Waveform

The output sine wave of the modulator will be the direct input carrier or the inverted (180° phase shifted) input carrier, which is a function of the data signal.

BPSK Demodulator

The block diagram of BPSK demodulator consists of a mixer with local oscillator circuit, a band pass filter, a two-input detector circuit. The diagram is as follows.

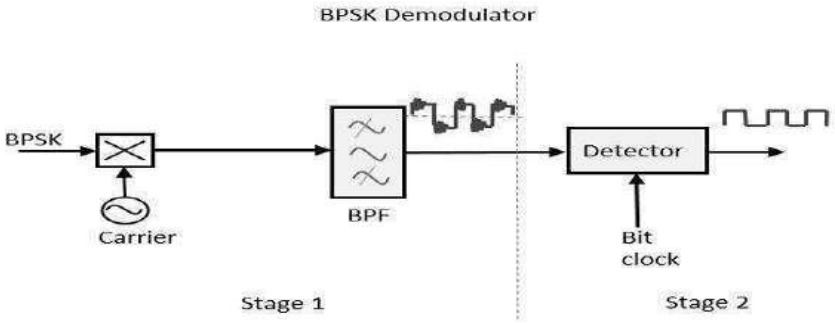


Figure 5.8.3 BPSK Demodulator

By recovering the band-limited message signal, with the help of the mixer circuit and the band pass filter, the first stage of demodulation gets completed. The base band signal which is band limited is obtained and this signal is used to regenerate the binary message bit stream.

In the next stage of demodulation, the bit clock rate is needed at the detector circuit to produce the original binary message signal. If the bit rate is a sub-multiple of the carrier frequency, then the bit clock regeneration is simplified. To make the circuit easily understandable, a decision-making circuit may also be inserted at the 2nd stage of detection.

RGPVNOTES.IN

The Quadrature Phase Shift Keying (QPSK) is a variation of BPSK, and it is also a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, which sends two bits of digital information at a time, called as bigits.

Instead of the conversion of digital bits into a series of digital stream, it converts them into bit pairs. This decreases the data bit rate to half, which allows space for the other users.

5.9 Shannon's Theorem for Channel Capacity:

Channel Capacity

We have so far discussed mutual information. The maximum average mutual information, in an instant of a signaling interval, when transmitted by a discrete memory less channel, the probabilities of the rate of maximum reliable transmission of data, can be understood as the channel capacity.
It is denoted by C and is measured in bits per channel use.

Shannon Limit

Shannon-Hartley equation relates the maximum capacity (transmission bit rate) that can be achieved over a given channel with certain noise characteristics and bandwidth. For an AWGN the maximum capacity is given by

$$C = B \log_2 (1 + \frac{S}{N})$$

Here CC is the maximum capacity of the channel in bits/second otherwise called Shannon's capacity limit for the given channel, BB is the bandwidth of the channel in Hertz, SS is the signal power in Watts and NN is the noise power, also in Watts. The ratio S/N is called Signal to Noise Ratio (SNR)**.** It can be ascertained that the maximum rate at which we can transmit the information without any error, is limited by the bandwidth, the signal level, and the noise level. It tells how many bits can be transmitted per second without errors over a channel of bandwidth B Hz, when the signal power is limited to S Watt and is exposed to Gaussian White Noise of additive nature.