GOVERNMENT POLYTECHNIC, BHUBANESWAR
Government of Odisha
ସରକାରୀ ବହୁବୃତ୍ତି ଅନୁଷ୍ଠାନ, ଭୁବନେଶ୍ୱର

LECTURE NOTES
ON
DIGITAL ELECTRONICS

4th Semester
Information Technology

Prepared by:
P. Bhawani
Lecturer in Electronics & Telecommunication Engineering
Government polytechnic Bhubaneswar

UNIT-1: BASICS OF DIGITAL ELECTRONICS

Digital electronics *is* a field of electronics that deals with digital data in the form of codes. There are only two codes in digital electronics, and they are 0 and 1. 0 is considered to be low logic while 1 is considered to be high logic.

Important terms:

Bit: A bit is the smallest unit of information. Bits in computer are grouped to form a lager unit of information

Byte: A byte is a combination of eight bits. Eight bits represent a character and is called a byte.

Nibble: A nibble is a combination of four bits, in other words a nibble is half a byte.

Word: a word is a combination of 16 bits, 32 bits or 64 bits depending on the computer. 16 is known as quad word.

Kilobyte (kb): A kilobyte has a memory unit of 1024 (2^{10})

Megabyte: A megabyte has a memory unit of 1,048,576 (2^{20}). This is approximately 1 million bytes.

Gigabytes (Gb): A gigabyte has a memory unit of 1,073,741,824 (2**30). This approximately billion bytes.

Terabytes: a terabyte has a memory unit of (2**40). This is approximately 1 trillion bytes.

Note:

1024 byte = 1 kilobyte (1KB)

1024 kilobytes = 1 megabytes (1MB)

1024 megabyte = 1 gigabytes (1 GB)

1024 gigabytes =1 terabyte (1TB)

1.1 Number System-Binary, Octal, Decimal, Hexadecimal - Conversion from one system to another number system.

In digital electronics, the number system is used for representing the information.

The four most common number system are:

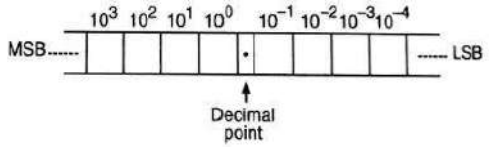
1. Decimal number system (Base- 10)
2. Binary number system (Base- 2)
3. Octal number system (Base-8)
4. Hexadecimal number system (Base- 16)

The **Base or Radix** of the number system is the total number of digit used in the number system.

For example, in decimal number system, it represents digit from 0-9 then the base of the system is 10.

1.Decimal Number System

- A number system which uses digits 0 to 9 to represent a number is called as Decimal Number System.
- It is also called as base 10 number system.
- Positional weight:

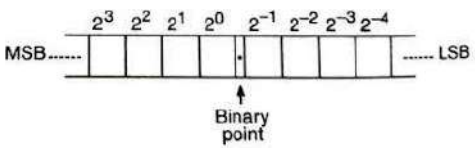


Example

Decimal Number: 2479.08_{10}

2.Binary Number System

- A Number system where a number is represented by using only two digits (i.e 0 & 1) is called as Binary number system.
- It is also called as base 2 number system.
- Positional weight:

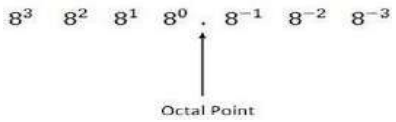


Example

Binary Number: 10101_2

3.Octal Number System

- A number system which uses digits 0 to 7 to represent a number is called as Octal Number System.
- It is also called as base 8 number system
- Positional weight:



MSB LSB

Example

Octal Number: 12570_8

4.Hexadecimal Number System

- A number system which uses 16 digits (10 digits i.e 0 to 9 and 6 letters i.e A, B, C, D, E, F) to represent a number is called as Hexadecimal Number System.
- It is also called as base 16 number system.
- Positional weight:

Example:

Hexadecimal Number: $19FDE.\ 7A_{16}$

Conversion of Number System

Binary to Decimal

Example 1:

$(10100011)_2 = (1 \times 2^7) + (0 \times 2^6) + (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$= 128 + 0 + 32 + 0 + 0 + 0 + 2 + 1$

$= 163$

Hence the decimal equivalent number of 10100011_2 is given as: 163_{10}

Example 2:

$(1101.0111)_2 = (1\times2^3) + (1\times2^2) + (0\times2^1) + (1\times2^0) + (0\times2^{-1}) + (1\times2^{-2}) + (1\times2^{-3}) + (1\times2^{-4})$

$= 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8 + 1/16$

$= 8 + 4 + 0 + 1 + 0 + 0.25 + 0.125 + 0.0625 = 13.4375_{10}$

Hence the decimal equivalent number of 1101.0111_2 is given as: 13.4375_{10}

Octal to Decimal

Example 1:

$(234)_8 = 2\times8^2 + 3\times8^1 + 4\times8^0$

$= 2\times64 + 3\times8 + 4\times1$

$= 128 + 24 + 4$

$=156$

Hence the decimal equivalent number of $(234)_8$ is given as $(156)_{10}$

Example 2:

$(137.24)_8 = (1 \times 8^2) + (3 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) + (4 \times 8^{-2})$

$= 64+24+7+2/8+4/64$

$= (95.3125)_{10}$

Hence the decimal equivalent number of $(137.24)_8$ is given as $(95.3125)_{10}$

Hexadecimal to Decimal

Example 1

Convert $(E\ F.\ B1\ )_{16} = (?\ )_{10}$

$= E \times 16^1 + F \times 16^0 + B \times 16^{-1} + 1 \times 16^{-2}$
$= 14 \times 16 + 15 \times 1 + 11 \times (1 / 16) + 1 \times (1 / 256)$
$= 224 + 15 + (0.6875) + (0.00390625)$
$= 239 + 0.6914$
$= 239.691406$
Therefore $(EF.B1)_{16} = (239.691406)_{10}$

Decimal to Binary

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder.

Example 1:

$(43)_{10}=(?)_2$

		Remainder
2	43	
2	21	1
2	10	1
2	5	0
2	2	1
2	1	0
	0	1

MSB

The remainders are to be read from bottom to top to obtain the binary equivalent.

$(43)_{10} = (101011)_2$

Example 2:

$(105.692)_{10} = (?)_2$

		Remainder	
2	105	1	LSB
2	52	0	
2	26	0	
2	13	1	
2	6	0	
2	3	1	MSB
	1		

.692 x 2 = 1.384	1	MSB
.384 x 2 = 0.768	0	
.768 x 2 = 1.536	1	
.536 x 2 = 1.072	1	LSB

$(105.692)_{10} = (1101001.1011)_2$

Decimal to Octal

Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder.

Example 1:

$(473)_{10} =(?)_8$

		Remainder
8	473	
8	59	1
8	7	3
	0	7

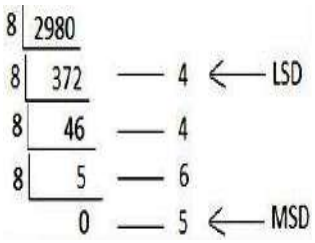
MSD

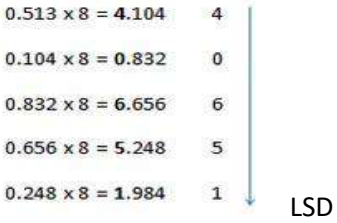
Reading the remainders from bottom to top,

$473_{10} = 731_8$

Example 2:

Convert $(2980.513)_{10}$ to $(?)_8$





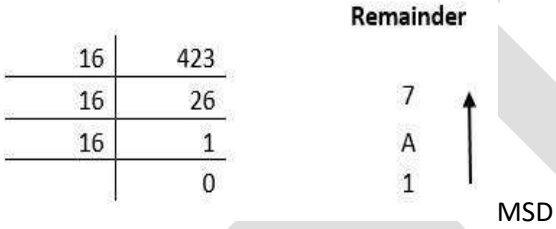
$(2980.513)_{10} = (5644.40651)_8$

Decimal to Hexadecimal

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder.

Example 1:

$(423)_{10} = (?)_{16}$

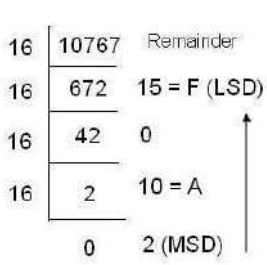


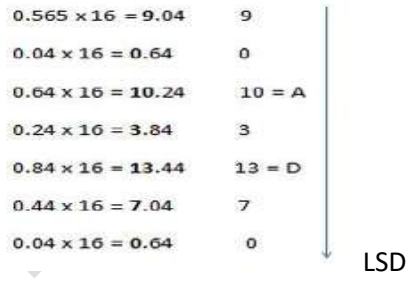
Reading the remainders from bottom to top we get,

$(423)_{10} = (1A7)_{16}$

Example 2:

$(10767.565)_{10} = (?)_{16}$





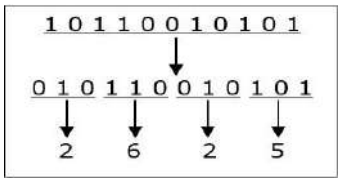
$(10767.565)_{10} = (2A0F.90A3D70)_{16}$

Binary to Octal and Vice Versa

To convert a binary number to octal number, these steps are followed −

- Starting from the least significant bit, make groups of three bits.
- If there are one or two bits less in making the groups, 0s can be added after the most significant bit
- Convert each group into its equivalent octal number

Example: $(10110010101)_2 = (?)_8$



$10110010101_2 = 2625_8$

To convert an **octal number to binary**, each octal digit is converted to its 3-bit binary equivalent according to this table.

Octal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	000	001	010	011	100	101	110	111

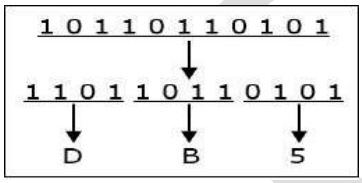
Example: $54673_8 = 101100110111011_2$

Binary to Hexadecimal

To convert a binary number to hexadecimal number, these steps are followed –

- Starting from the least significant bit, make groups of four bits.
- If there are one or two bits less in making the groups, 0s can be added after the most significant bit.
- Convert each group into its equivalent hexadecimal number.

Example: $(10110110101)_2 = (?)_{16}$



$10110110101_2 = DB5_{16}$

To convert a **hexadecimal number to binary**, each hexadecimal digit is converted to its 4-bit binary equivalent according to this table.

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

Octal to Hexadecimal and vice versa

- While converting from octal to hexadecimal, first convert the octal number into binary digit and then further from binary to hexadecimal.

Convert 536 from octal to hexadecimal number

$(536)_8$ = (101) (011) (110)

$=(101011110)_2$

Now forming the group of 4 binary bits to obtain its hexadecimal equivalent,

$(101011110)_2$= (0001) (0101) (1110)

$= (15E)_{16}$

So the hexadecimal number of 536 is 15E.

- While converting from hexadecimal to octal, first convert the hexadecimal number into binary digit and then further from binary to octal.

Convert 15E hexadecimal number to Octal number

$(15E)_{16}$ = (0001) (0101) (1110)

$= (000101011110)_2$

Now forming the group of 3 binary bits to obtain its octal equivalent,

$(000101011110)_2$ = (101) (011) (110) = $(536)_8$

Number System Relationship

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

1.2 Arithmetic Operation-Addition, Subtraction, Multiplication, Division, 1"s & 2"s complement of Binary numbers& Subtraction using complements method

ADDITION:

Binary addition refers to adding more than one binary number by using of binary addition rules.
The four rules of binary addition are:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 with a carry-over of 1

Find the sum of the following numbers:

i) 10101 and 11011

Solution:
10101 and 11011

```
  1 1 1 1    Carry overs
  1 0 1 0 1
  1 1 0 1 1
-------------
1 1 0 0 0 0
```

ii) 11001 and 111

Solution:
11001 and 111

```
  1 1 1 1    Carry overs
  1 1 0 0 1
      1 1 1
-------------
1 0 0 0 0 0
```

iii) 10101.101 and 1101.011

Solution:
10101.101 and 1101.011

```
  1 1   1 1  1 1    Carry overs
  1 0 1 0 1 . 1 0 1
    1 1 0 1 . 0 1 1
-------------------
1 0 0 0 1 1 . 0 0 0
```

SUBTRACTION:

Binary subtraction is the process of subtracting binary numbers by using binary subtraction rules.
The four rules of binary subtraction are:
0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 with a borrow of 1

Subtract the following numbers:
i) 101 from 1001
Solution:

1 Borrow
1 0 0 1
1 0 1
1 0 0

ii) 111 from 1000

Solution:
111 from 1000

1 Borrow
1 0 0 0
1 1 1
0 0 0 1

iii) 11010.101 from 101100.011

Solution:

11010.101 from 101100.011

1 1 1 Borrow

1 0 1 1 0 0 . 0 1 1
1 1 0 1 0 . 1 0 1
1 0 0 0 1 . 1 1 0

MULTIPLICATION:

Binary multiplication refers to multiplying binary numbers by using binary multiplication rules.

The rules for multiplication

$0 \times 0 = 0$
$1 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 1 = 1 (there\ is\ no\ carry\ or\ borrow\ for\ this)$

Examples of Binary Multiplication

1)

```
  10101
×   101
-------
  10101
 000000
1010100
-------
1101001
```

2)

```
    1011.01
×    110.1
----------
  101101
 0000000
1011010
1011010000
----------
1001001.001
```

DIVISION:

In Binary division, two binary numbers are divided by using binary division rules.
The main rules of the binary division include:

- 1÷1 = 1
- 1÷0 = 0
- 0÷1 = Meaningless
- 0÷0 = Meaningless

Evaluate:

(i) 11001 ÷ 101

Solution:

```
101)  11001  (101
      101
       101
       101
```

Hence the quotient is 101

(ii) 11101.01 ÷ 1100

Solution:

```
1100)  11101.01  (10.0111
       1100
         10101
         1100
           0010
           1100
             1100
             1100
```

Hence the quotient is 10.0111

SIGNED-MAGNITUDE REPRESENTATION

In digital system, a number consists of a magnitude and a symbol which indicates whether the magnitude is positive or negative.

In binary numbers system an extra bit position is used to represent the sign. This extra bit is called the SIGN BIT and is placed before the magnitude of the number to be represented. Generally, the MSB is the sign bit and the convention is that when the sign bit is 0, the number represented is positive and when the sign bit is 1, the number is negative.

$(00000000)_2 = +(0)_{10}$

$(10000000)_2 = -(0)_{10}$

1'S & 2'S COMPLEMENT OF BINARY NUMBERS

The 1's complement can be defined as the inverse of a binary number. That means it can be obtained by substituting each 1 by 0 and 0 by 1.

Example:
Binary number-101100
1's complement = 010011
The 2's complement of a binary number is 1 added to the 1's complement of the binary number.
Example:

Binary number=101100

1's complement = 010011

2's complement= 1's complement + 1

=010100

SUBTRACTION USING COMPLEMENTS METHOD

1's complement subtraction:

- In 1's complement subtraction, add the 1's complement of subtrahend to minuend.
- If there is carry, then the carry is added to the LSB. This is called end around carry.
- If MSB is 0, the result is Positive.
- If the MSB is 1 then the result is Negative and is in its 1's complement form. Then take its 1's complement to get magnitude in binary.

Evaluate:
(i) 110101 – 100101
Solution:
1's complement of 10011 is 011010. Hence

Minued - 1 1 0 1 0 1

1's complement of subtrahend - 0 1 1 0 1 0

Carry over - 1 0 0 1 1 1 1

+ 1

0 1 0 0 0 0

As MSB is 0, so the number is positive. Hence the difference is 10000

(ii) 101011 – 111001
Solution:
1's complement of 111001 is 000110. Hence

Minued - 1 0 1 0 1 1

1's complement - 0 0 0 1 1 0

1 1 0 0 0 1

As MSB is 1, the number is negative. Take 1's complement of 10001
Hence the difference is – 01 1 1 0
(iii) 1011.001 – 110.10
Solution:
1's complement of 0110.100 is 1001.011 Hence

Minued - 1 0 1 1 . 0 0 1

1's complement of subtrahend - 1 0 0 1 . 0 1 1

Carry over - 1 0 1 0 0 . 1 0 0

+1

0 1 0 0 . 1 0 1

Hence the required difference is 100.101

2's complement subtraction:

- In 2's complement, add 2's complement of subtrahend to the minuend.
- If there is a carry ignore it.
- If the MSB is 0, the result is Positive.
- If the MSB is 1 the result is Negative and is in 2's complement form. Then take its complement to get the magnitude in binary.

Evaluate:
10110 – 11010
Solution:
2's complement of 11010 is (00101 + 1) i.e. 00110. Hence

Minued - 1 0 1 1 0

2's complement of subtrahend- 0 0 1 1 0

Result of addition - 1 1 1 0 0

As MSB is 1, the number is negative. Take 2's complement of 11100
So, 1's complement of 11100 is 00011
2's complement is 00011+1 =00100
Hence the difference is – 100.

1.3 Digital Code & its application & distinguish between weighted & non-weight Code, Binary codes, excess-3 and Gray codes.

- Digital coding is the process of using binary digits to represent letters, characters and other symbols in a digital format.
- The most widely used Digital Codes are ASCII (American Standard Code Information Interchange), BCD (Binary Coded Decimal), EBCDIC (Extended Binary Coded Decimal Interchange Code) and Unicode.

Binary codes can be classified into two types.

- Weighted codes
- Unweighted codes

In weighted codes, each digit is assigned a specific weight according to its position.
Examples:8421code,2421code
The Non - Weighted Code are not positionally weighted. In other words, codes that are not assigned with any weight to each digit position.
Examples: Excess 3 code, Grey code

Excess 3 code

- Excess-3 code doesn't have any weights. So, it is a non-weighted code.
- Excess-3 code can be obtained by adding 3 to each decimal digit then it can be represented by using 4-bit binary number for each digit. Hence, it is called as Excess 3 code.
- It is a self-complementing code.

Convert the following numbers to Excess-3 code

(a) 87 **(b) 159**

Solution:

(a)

8	7
+3	+3
11	10
↓	↓
1011	1010

(b)

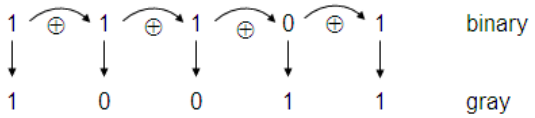
1	5	9
+3	+3	+3
4	8	12
↓	↓	↓
0100	1000	1100

Grey Code

- The Grey code is a non-weighted code.
- No specific weight assigned to the bit positions.
- It exhibits only a single change from one code word to the next sequence.

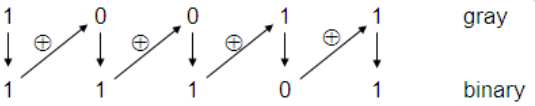
Conversion of a Binary number to Gray code

- The first bit(MSB) of the Gray code is the same as the first bit of the binary number
- The second bit of the Gray code equals the exclusive-OR, of the first and second bits of the binary number, i.e. it will be 1 if these binary code bits are different and 0 if they are the same.
- The third Gray code bit equals the exclusive-OR of the second and third bits of the binary number, and so on.



Conversion from Gray code to Binary

- The first binary bit(MSB) is the same as that of the first the Gray code bit.
- The second bit of the binary code equals the exclusive-OR, of the MSB and second bits of the gray code.
- Step 2 is repeated.

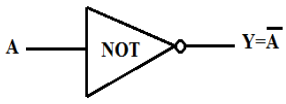


1.4 Logic gates: AND,OR,NOT,NAND,NOR, Exclusive-OR, Exclusive-NOR--Symbol, Function, expression, truth table & timing diagram

- Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on certain **logic**.
- In digital system, the basic gates are AND Gate, OR gate, NOT gate.
- NAND and NOR are called **universal gates** because all the other gates can be derived from it.
- Truth table is the list of all possible combination of input and the corresponding output.

NOT gate

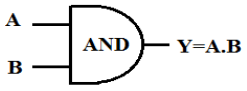
The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as A'.



Input	Output
A	$Y=\overline{A}$
0	1
1	0

AND gate

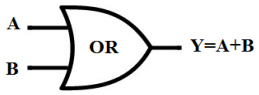
The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B.



Inputs		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

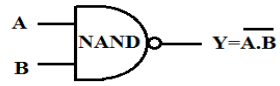
The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.



Inputs		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NAND gate

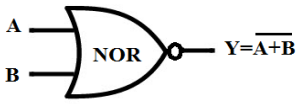
This is an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low.



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

NOR gate

This is an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.



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

EX-OR gate

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EX-OR operation.

Inputs		Output
A	B	Y=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

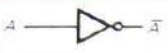
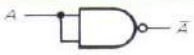
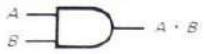
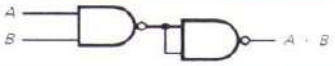
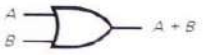
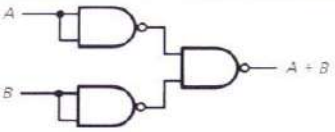
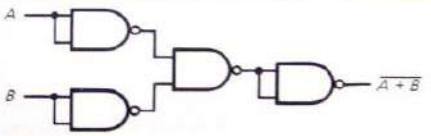
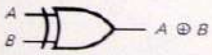
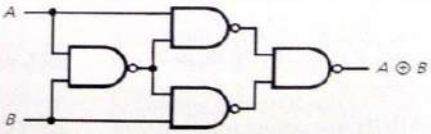
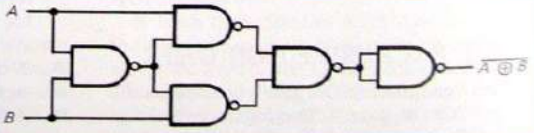
EXNOR gate

The '**Exclusive-NOR**' gate circuit does the opposite to the EXOR gate. It give low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

Inputs		Output
A	B	$Y=\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

1.5 Universal Gates& its Realisation

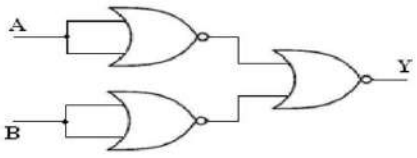
NAND and NOR are called **universal gates** because all the other gates can be derived from it.

LOGIC FUNCTION	SYMBOL	CIRCUIT USING NAND GATES ONLY
Inverter		
AND		
OR		
NOR		
XOR		
XNOR		

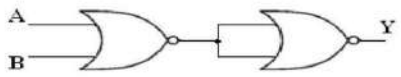
II. Implementation using NOR gate:

(a) NOT gate: $Y = A'$

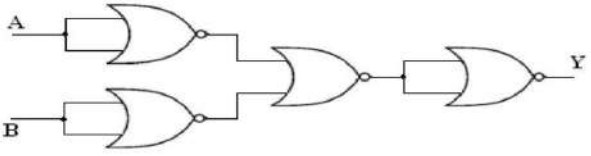
(b) AND gate: $Y = A \cdot B$



(c) OR gate: $Y = A + B$



(d) NAND gate: $Y = (AB)'$



(e) Ex-NOR gate: $Y = A \odot B = (A \oplus B)'$

1.6 Boolean algebra, Boolean expressions, Demorgan's Theorems.

- Boolean algebra is the set of rules used to simplify the logic expression without changing its functionality.
- It is used when number of variable is less (1,2,3)

Boolean Theorem:

1. INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.

$\overline{\overline{A}} = A$

2. AND law

These laws use the AND operation. Therefore, they are called as **AND** laws.

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

3. OR law

These laws use the OR operation. Therefore, they are called as **OR** laws.

(i) $A + 0 = A$ (ii) $A + 1 = 1$

(iii) $A + A = A$ (iv) $A + \overline{A} = 1$

4. Commutative law

(i) $A.B = B.A$ (ii) $A + B = B + A$

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

5. Associative law

(i) $(A.B).C = A.(B.C)$ (ii) $(A + B) + C = A + (B + C)$

This law states that when ORing/ANDing more than two variables, the result is the same regardless of the grouping of the variables.

6. Distributive law

Distributive law states the following condition.

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

This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the product.

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

This law states that ANDing two or more variables and then ORing the result with a single variable is equivalent to ORing the single variable with each of the two or more variables and then ANDing the sum.

7. Absorption Laws

Absorption law involves in linking of a pair of binary operations.

i. $A+AB = A$

ii. $A(A+B) = A$

iii. $A+\bar{A}B = A+B$

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

8.Consensus Laws

i. $AB + A'C + BC = AB + A'C$

ii. $(A+B).(A'+C).(B+C) = (A+B).(A'+C)$

DEMORGAN'S THEOREM

- The first theorem states that the complement of a product is equal to the sum of the complements. That is if the variables are A and B, then

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

- The second theorem states that, the complement of a sum is equal to the product of the complements. In equation form, this can be written as

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

- Demorgan's Theorem can be proved for any number of variables; proof of these two theorems for two variables can be found in the below truth table

A	B	$\overline{A}$	$\overline{B}$	A+B	$A \cdot B$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$
0	0	1	1	0	0	1	1	1	1
0	1	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	0	0	0	0

A study of truth table makes clear that columns 7 and 8 are equal. Therefore, $\overline{A + B} = \overline{A}\;\overline{B}$

Similarly, columns 9 and 10 are equal. Therefore, $\overline{A\,B} = \overline{A} + \overline{B}$

1.7 Represent Logic Expression: SOP & POS forms

Sum of Product (SOP)

- Sum of Product is the abbreviated form of SOP. Sum of product form is a form of expression in Boolean algebra in which different product terms of inputs are being summed together.
- This product is not arithmetical multiply but it is Boolean logical AND and the Sum is Boolean logical OR.
- Example:
  AB+A'C+AB'C'
- There are few different forms of Sum of Product.
  - Canonical SOP Form
  - Non-Canonical SOP Form
  - Minimal SOP Form
- In Canonical SOP form, each product term contains all literals, either with complement or without complement. So, these product terms are nothing but the min terms. Hence, canonical SOP form is also called as sum of min terms form.
- The minterms for two variable (x and y) are x'y', x'y, xy' and xy. For a three variable (x, y and z) minterms are: x'y'z', x'y'z, x'yz', x'yz, xy'z', xy'z, xyz' and xyz.
- If the variable value is 1, the variable is without its complement. If the variable value is 0, take its complement.
- The minterms can be represented by m_0, m_1, m_2, m_3……; the suffix indicates the decimal code corresponding to the minterm combination.
- The expression of the canonical SOP is denoted with sign summation (∑), and the minterms in the bracket are taken when the output is true.
- The Boolean function of output is, f = p'qr + pq'r + pqr' + pqr. This is the canonical SOP form of output, f. It can be represent this function in following two notations.
  f=m3+m5+m6+m7
  f=∑ m (3,5,6,7)

Product of Sum(POS)

- **Product of Sum is the abbreviated for POS.** The product of Sum form is a form in which products of different sum terms of inputs are taken.
- These are not arithmetic product and sum but they are logical Boolean AND and OR respectively.
- Example:

  (A+B+C')(A+B')(A'+B'+C)
- There are few different forms of Sum of Product.
  - Canonical SOP Form
  - Non-Canonical SOP Form
  - Minimal SOP Form
- In Canonical POS, each sum term contains all literals either with complement or without complement. So, these sum terms are nothing but the Max terms. Hence, canonical POS form is also called as product of Max terms form.
- The maxterms for two variable (x and y) are x'+y', x'+y, x+y' and x+y. For three variable (x, y and z) Boolean function, the possible maxterms are: x + y + z, x + y + z', x + y' + z, x + y' + z', x' + y + z, x' + y + z', x' + y' + z and x' + y' + z'.
- If the variable value is 0, the variable without its complement.

  If the variable value is 1, take its complement.
- The maxterms can be represented by M_0, M_1, M_2, M_3……; the suffix indicates the decimal code corresponding to the maxterm combination.
- The expression this is denoted by ∏ and the max terms in the bracket are taken when the output is false.
- The Boolean function of output is, f = p+q+r. p+q+r'. p+q'+r. p'+q+r. This is the canonical POS form of output, f. We can also represent this function in following two notations.
  $f=M_0.M_1.M_2.M_4$
  f=∏ M (0,1,2,4)

Conversion of SOP form to standard SOP form or Canonical SOP form

For getting the standard SOP form of the given non-standard SOP form, we will add all the variables in each product term which do not have all the variables. By using the Boolean algebraic law, (x + x' = 1) and by following the below steps we can easily convert the normal SOP function into standard SOP form.

- Multiply each non-standard product term by the sum of its missing variable and its complement.
- Repeat step 1, until all resulting product terms contain all variables
- For each missing variable in the function, the number of product terms doubles.

Example:

Convert the non standard SOP function F = AB + A C + B C

F = A B + A C + B C

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

= A B C + A B C' + A B C + A B' C + A B C + A' B C

= A B C + A B C' + A B' C + A' B C

So, the standard SOP form of non-standard form is F = A B C + A B C' + A B' C + A' B C

Conversion of POS form to standard POS form or Canonical POS form

For getting the standard POS form of the given non-standard POS form, we will add all the variables in each product term that do not have all the variables. By using the Boolean algebraic law (x * x' = 0) and by following the below steps, we can easily convert the normal POS function into a standard POS form.

- By adding each non-standard sum term to the product of its missing variable and its complement, which results in 2 sum terms
- Applying Boolean algebraic law, x + y z = (x + y) * (x + z)
- By repeating step 1, until all resulting sum terms contain all variables

By these three steps, we can convert the POS function into a standard POS function.

Example:

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

Conversion of SOP form to POS form

For getting the POS form of the given SOP form expression, we will change the symbol ∑ to ∏. After that, we will write the numeric indexes of the variables which are missing in the boolean function.

There are the following steps used to convert the SOP function F = ∑ x, y, z (0, 2, 3, 5, 7) = x' y' z' + x' y z' + x' y z + xy'z + xyz into POS: F=∏ x, y, z (1,4,6) = (x+ y+ z') (x'+ y+ z) (x'+ y'+ z)

Conversion of POS to SOP form

For getting the SOP form from the POS form, we have to change the symbol ∏ to ∑. After that, we write the numeric indexes of missing variables of the given Boolean function.

There are the following steps to convert the POS function F = Π x, y, z (2, 3, 5) = (x +y'+ z) (x +y' +z')(x'+ y+ z') into SOP form: F= ∑ x, y, z (0, 1, 4, 6, 7)=x'y'z'+x'y'z+xy'z'+xyz'+xyz

1.8 Karnaugh map (3 & 4 Variables)&Minimization of logical expressions ,don't care conditions

- Karnaugh map or K-Map method is most suitable method for minimizing Boolean functions. With the help of the K-map method, we can find the simplest POS and SOP expression, which is known as the minimum expression.
- It is a graphical method, which consists of 2^n cells for 'n' variables. The adjacent cells are differed only in single bit position.
- Just like the truth table, a K-map contains all the possible values of input variables and their corresponding output values.
- The K-map method is used for expressions containing 2, 3, 4, and 5 variables.
- The K-map grid is filled using 0's and 1's. It is solved by making groups.

Karnaugh Map Simplification Rules-

To minimize the given Boolean function,

- Draw a K Map according to the number of variables it contains.
- Fill the K Map with 0's and 1's according to its function.
- Then, minimize the function in accordance with the following rules.

Rule-01:

- Either group 0's with 0's or 1's with 1's but we cannot group 0's and 1's together.
- X representing don't care can be grouped with 0's as well as 1's.

Rule-02:

- Groups may overlap each other.

Rule-03:

- We can only create a group whose number of cells can be represented in the power of 2.
- In other words, a group can only contain 2^n i.e. 1, 2, 4, 8, 16 and so on number of cells.

Rule-04:

- Groups can be only either horizontal or vertical.
- Groups cannot be of diagonal or any other shape.

Rule-05:

Each group should be as large as possible.

Rule-06:

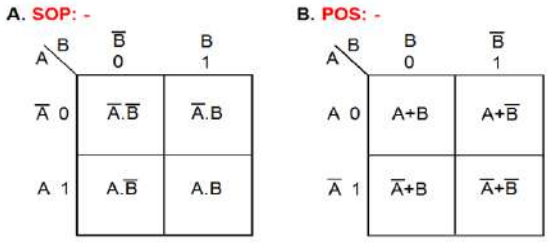
- Opposite grouping and corner grouping are allowed.

Rule-07:

- There should be as few groups as possible.

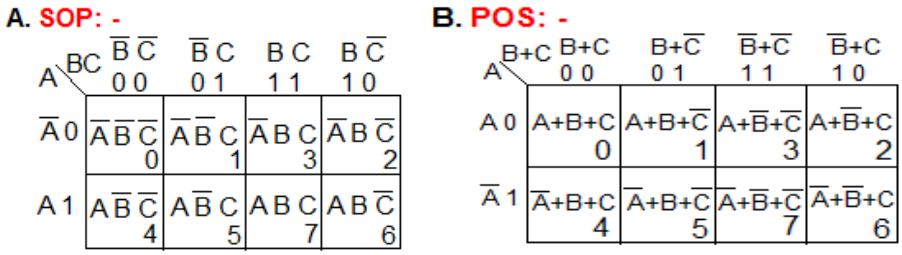
2 Variable K-Map

The number of cells in 2 variable K-map is four, since the number of variables is two. The following figure shows **2 variable K-Map**.



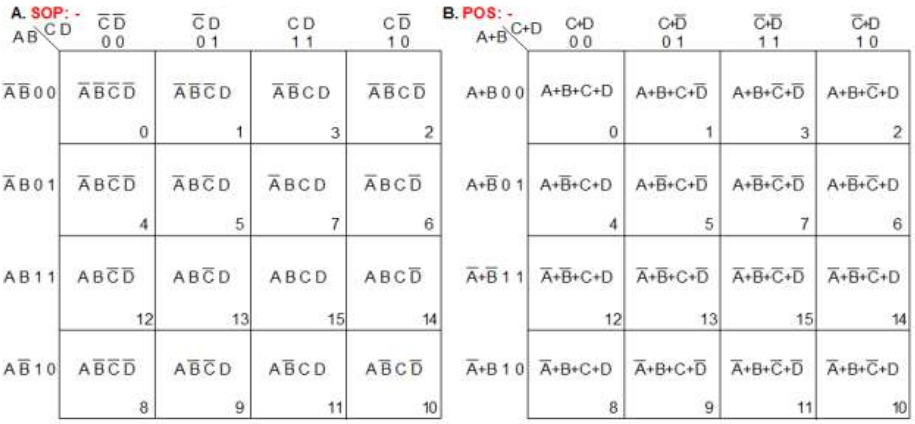
3 Variable K-Map

The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows **3 variable K-Map**.



4 Variable K-Map

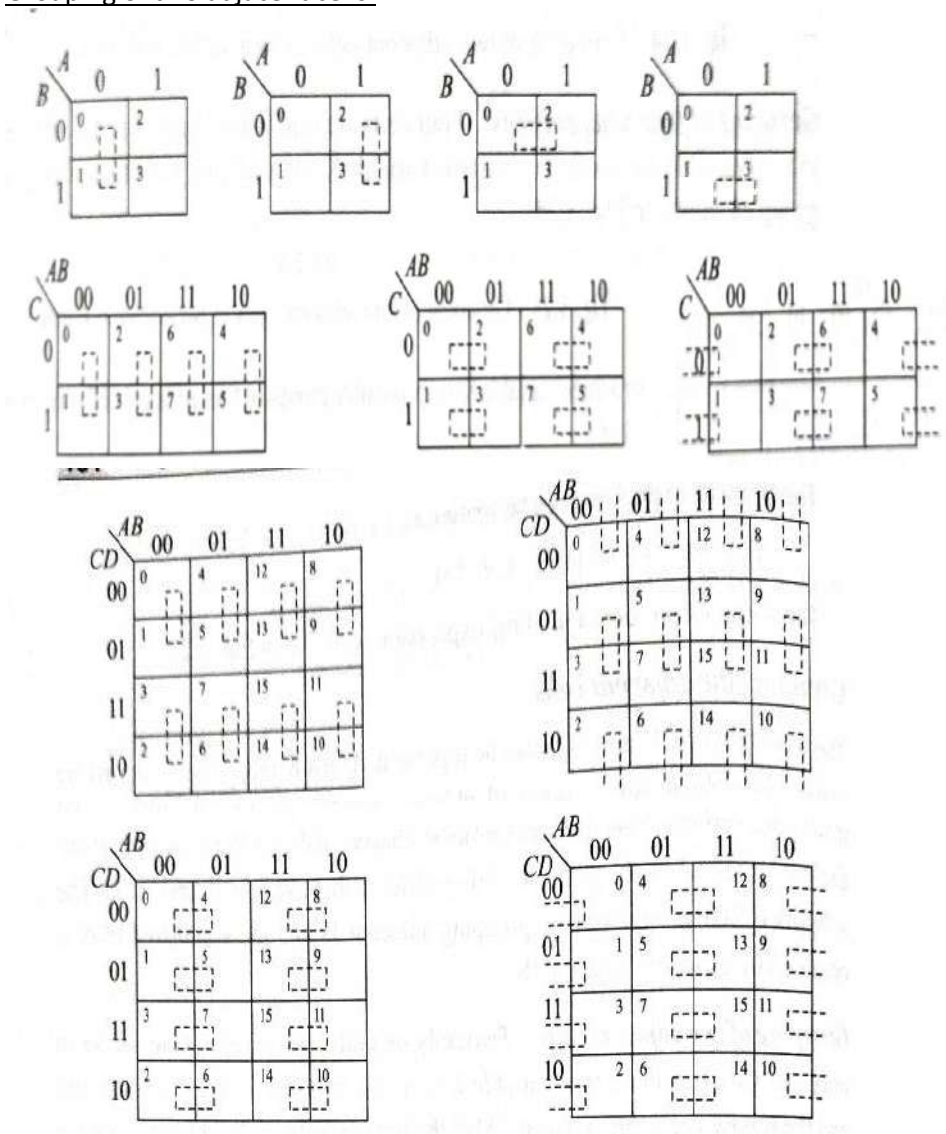
The number of cells in 4 variable K-map is sixteen, since the number of variables is four. The following figure shows **4 variable K-Map**.



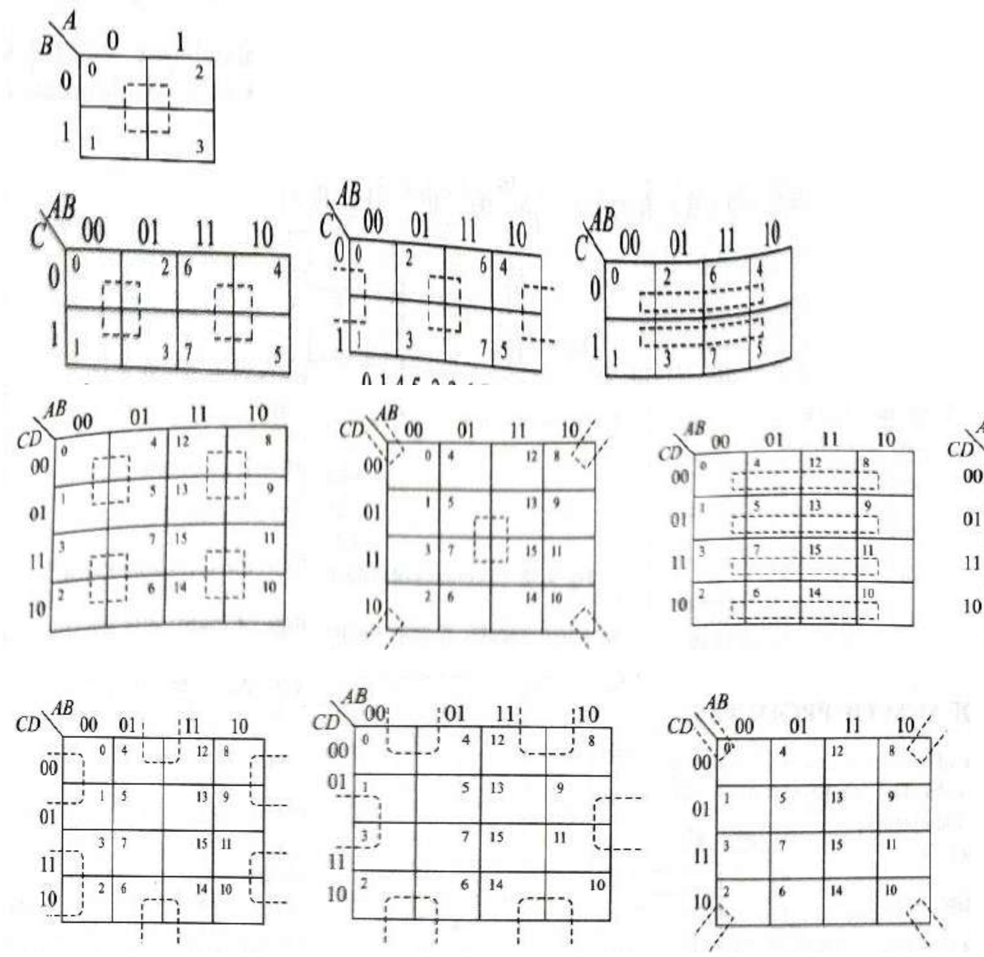
Don't Care condition

- The "Don't Care" conditions allow us to replace the empty cell of a K-Map to form a grouping of the variables. While forming groups of cells, we can consider a "Don't Care" cell as either 1 or 0 or we can simply ignore that cell. Therefore, "Don't Care" condition can help us to form a larger group of cells.
- A Don't Care cell can be represented by a cross(X) or (d) in K-Maps representing an invalid combination.

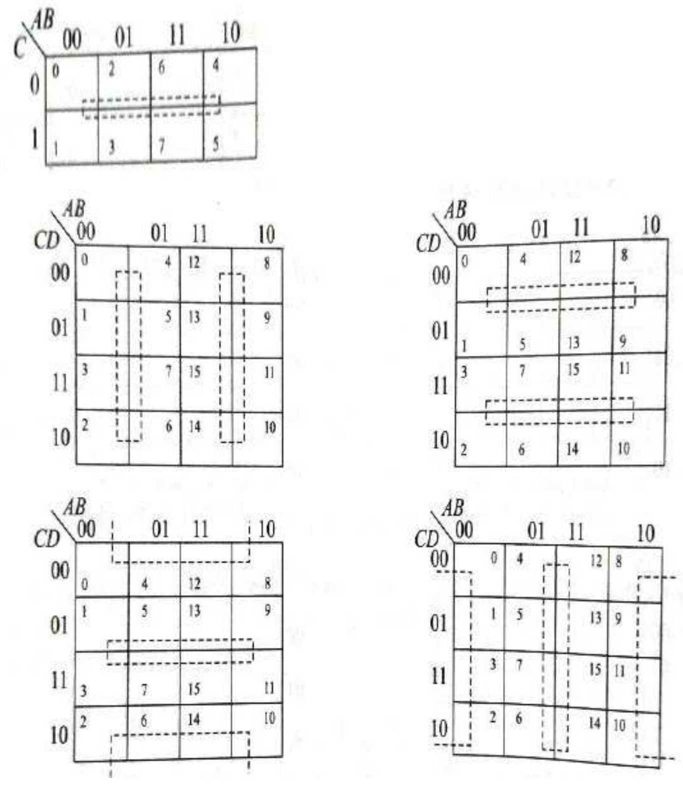
Grouping of two adjacent cells:



Grouping of four adjacent cells:

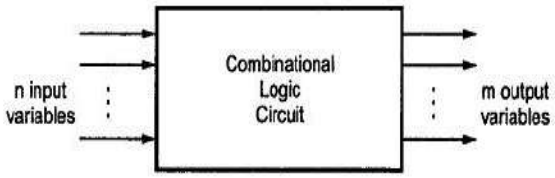


Grouping of eight adjacent cells:



UNIT-2: COMBINATIONAL LOGIC CIRCUITS

- A combinational circuit is the digital logic circuit in which the output depends on present combination of inputs, regardless of the previous input.



- The input variables, logic gates, and output variables are the basic components of the combinational logic circuit.
- There are different types of combinational logic circuits, such as Adder, Subtractor, Decoder, Encoder, Multiplexer, and De-multiplexer.

Design procedure of Combinational circuits

- Find the required number of input variables and outputs from given specifications.
- Formulate the Truth table. If there are 'n' input variables, then there will be 2^n possible combinations. For each combination of input, find the output values.
- Find the Boolean expressions for each output. If necessary, simplify those expressions.
- Implement the above Boolean expressions corresponding to each output by using Logic gates.

2.1 Half adder, Full adder, Half Subtractor, Full Subtractor, Serial and Parallel Binary 4 bit adder.

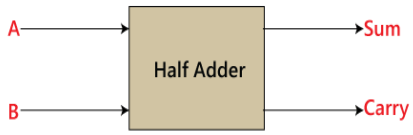
ADDER:

Adder circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output.
Adder circuits are of two types: Half adder and Full adder

HALF ADDER:

- Half adder is a combinational arithmetic circuit that adds two single bit numbers and produces a sum and carry as the output.
- There are two inputs named as A and B and the outputs are named as Sum (S) and Carry (C).

Block diagram



Truth Table

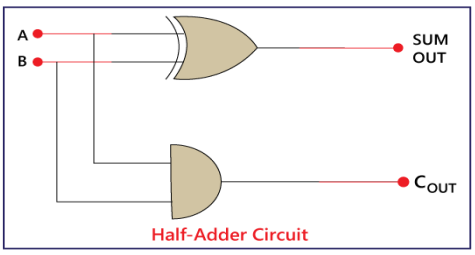
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From Truth table, the Boolean functions for each output as

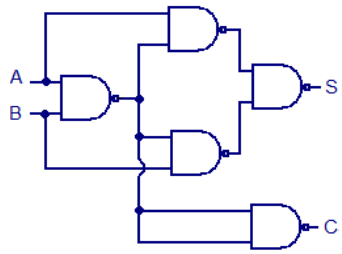
$S=A \oplus B$

$C=AB$

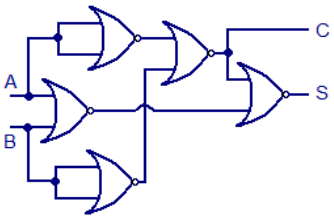
Logical diagram



NAND gates or NOR gates can be used for realizing the half adder in universal logic and the relevant circuit diagrams are shown in the figure below.



Half adder using NAND logic

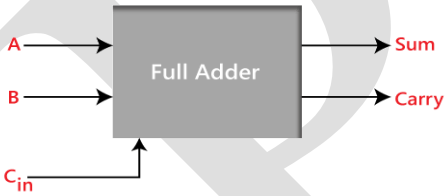


Half adder using NOR logic

FULL ADDER

Full Adder is the adder circuit which adds three single bit inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C_{in}. The output carry is designated as Carry and the normal output is designated as S which is SUM.

Block diagram



Truth Table

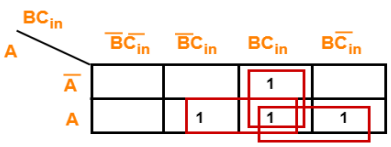
Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-map for SUM and CARRY

For S:

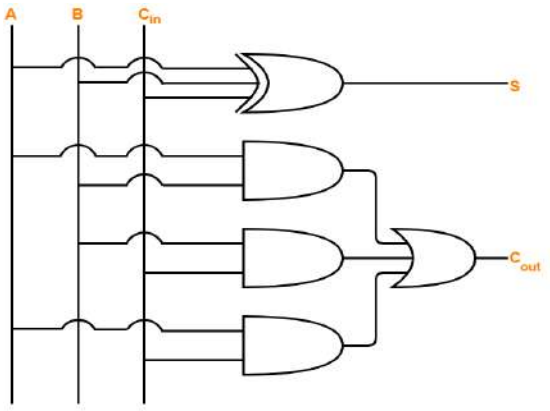
$S = A \oplus B \oplus C_{in}$

For C_{in}:



$C_{out} = AB + BC_{in} + C_{in}A$

Logical Diagram:



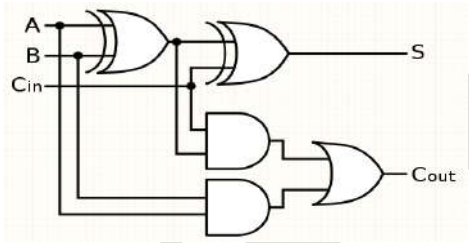
OR

SUM = $A \oplus B \oplus C_{in}$

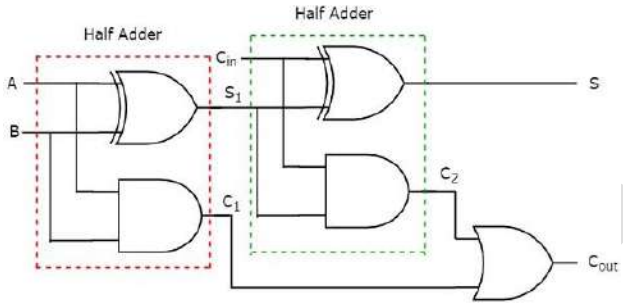
And

CARRY

$$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$
$$= C_i(\bar{A}B + A\bar{B}) + AB(\bar{C}_i + C_i)$$
$$= C_i(A \oplus B) + AB$$



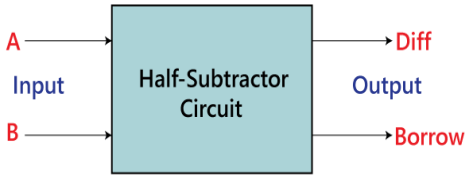
Full Addrer Using Half adder



HALF SUBTRACTOR

The half subtractor is a combinational circuit. It has two inputs and two outputs. This circuit is used to subtract two single bit binary numbers A and B. The **'difference'** and **'borrow'** are two output states of the half subtractor.

Block diagram

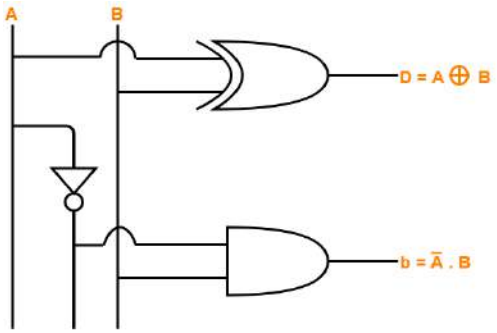


Truth Table

Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The SOP form of the **Diff** and **Borrow** is as follows:
Diff=A'B+AB'
Borrow = A'B

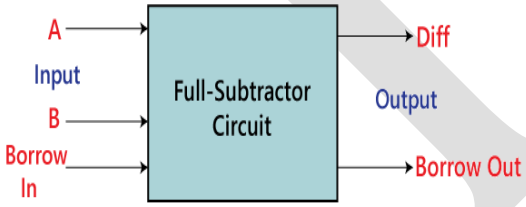
Logical Diagram



FULL SUBTRACTOR

The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.

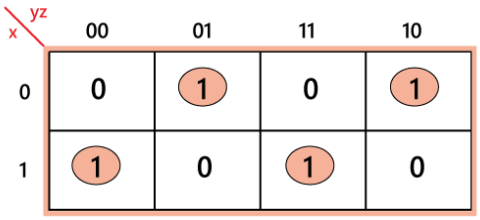
Block diagram

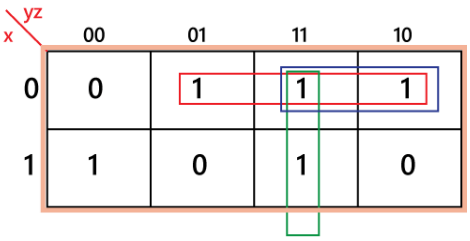


Truth Table

Inputs			Outputs	
A	B	$Borrow_{in}$	Diff	Borrow
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

The SOP form can be obtained with the help of K-map as:

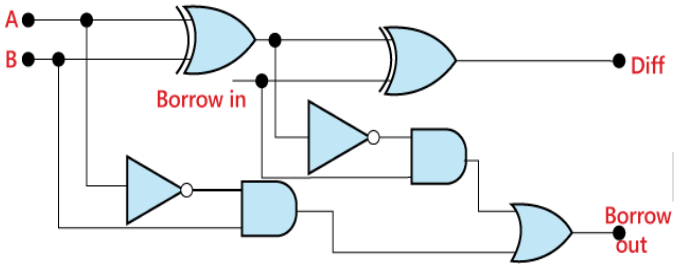




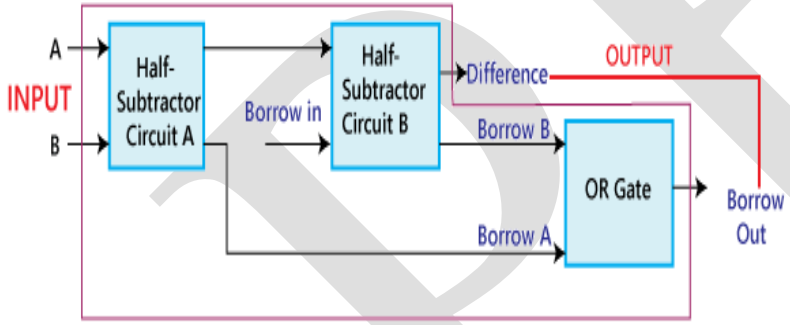
Diff=xy' z'+x' y' z+xyz+x'yz'

Borrow=x' z +x' y + yz

Logic circuit Diagram



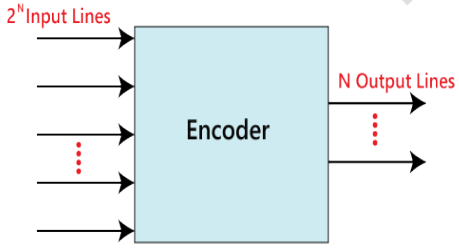
Full Subtractor Circuit using half subtractor:



2.2 Multiplexer (4:1), De- multiplexer (1:4), Decoder, Encoder, Digital comparator (3 Bit)

ENCODER

The combinational circuits that change the binary information in the form of 2^N input lines into N output lines are known as **Encoders**.

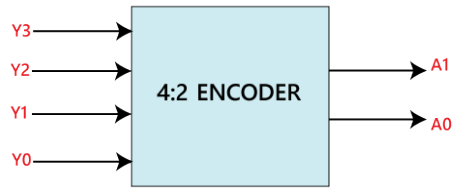


There are various types of encoders which are as follows:

4 to 2 line Encoder:

In 4 to 2 line encoder, there are total of four inputs, i.e., Y_0, Y_1, Y_2, and Y_3, and two outputs, i.e., A_0 and A_1. In 4-input lines, one input-line is set to true at a time to get the respective binary code in the output side.

Block Diagram:



Truth Table:

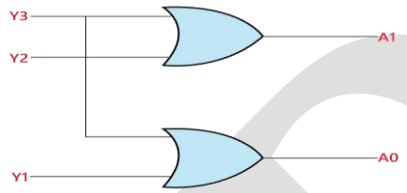
INPUTS				OUTPUTS	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

The logical expression of the term A0 and A1 is as follows:

$A_1=Y_3+Y_2$

$A_0=Y_3+Y_1$

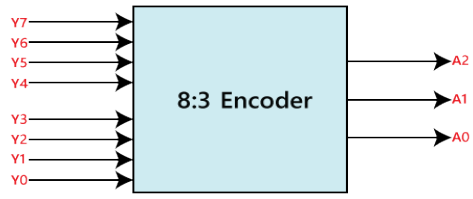
Logical circuit of the above expressions is given below:



8 to 3 line Encoder:

The 8 to 3 line Encoder is also known as **Octal to Binary Encoder**. In 8 to 3 line encoder, there is a total of eight inputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2. In 8-input lines, one input-line is set to true at a time to get the respective binary code in the output side.

Block Diagram:



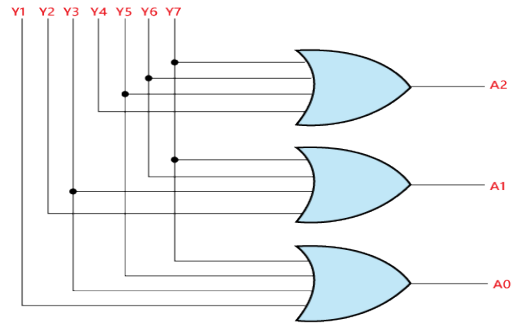
Truth Table:

INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

The logical expression of the term A0, A1, and A2 are as follows:

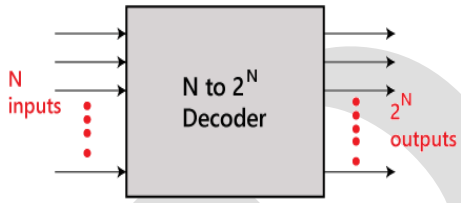
$A_2=Y_4+Y_5+Y_6+Y_7$
$A_1=Y_2+Y_3+Y_6+Y_7$
$A_0=Y_7+Y_5+Y_3+Y_1$

Logical circuit of the above expressions is given below:



DECODER

The combinational circuit that change the binary information in the form of N input lines into 2^N output lines is known as **Decoders**. In simple words, the **Decoder** performs the reverse operation of **Encoder**. At a time, only one input line is activated for simplicity.

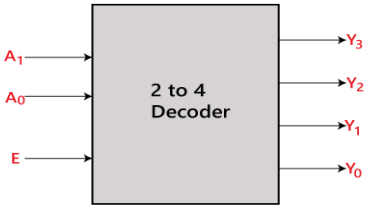


There are various types of decoders which are as follows:

2 to 4 line decoder:

In the 2 to 4 line decoder, there is a total of three inputs, i.e., A_0, and A_1 and E and four outputs, i.e., Y_0, Y_1, Y_2, and Y_3. For each combination of inputs, when the enable 'E' is set to 1, one of these four outputs will be 1.

Block Diagram:



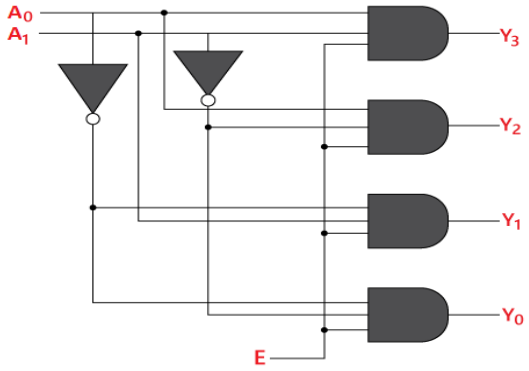
Truth Table:

Enable	INPUTS		OUTPUTS			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

The logical expression of the term Y0, Y0, Y2, and Y3 is as follows:

$Y_3=E.A_1.A_0$
$Y_2=E.A_1.A_0'$
$Y_1=E.A_1'.A_0$
$Y_0=E.A_1'.A_0'$

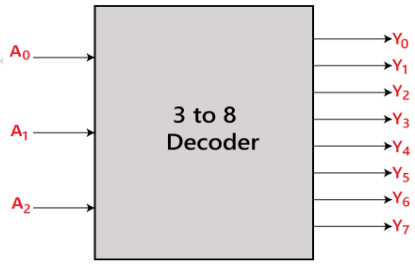
Logical circuit of the above expressions is given below:



3 to 8 line decoder:

The 3 to 8 line decoder is also known as **Binary to Octal Decoder**. In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2. This circuit has an enable input 'E'. Just like 2 to 4 line decoder, when enable 'E' is set to 1, one of these four outputs will be 1.

Block Diagram:



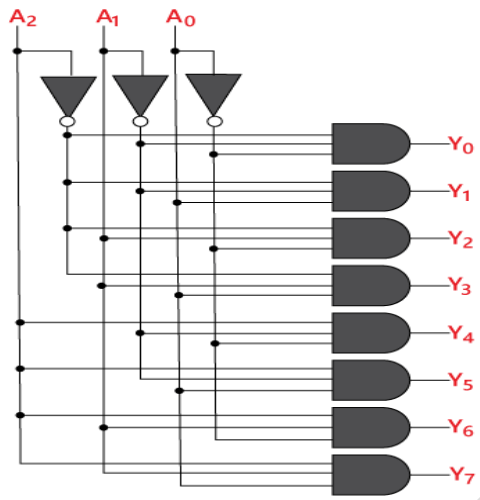
Truth Table:

Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

The logical expression of the term Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 is as follows:
$Y_0=A_0'.A_1'.A_2'$
$Y_1=A_0.A_1'.A_2'$
$Y_2=A_0'.A_1.A_2'$
$Y_3=A_0.A_1.A_2'$
$Y_4=A_0'.A_1'.A_2$

$Y_5=A_0.A_1'.A_2$
$Y_6=A_0'.A_1.A_2$
$Y_7=A_0.A_1.A_2$

Logical Diagram



Digital Comparator

- A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers (consider A and B) and determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.
- Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B. The below figure shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.

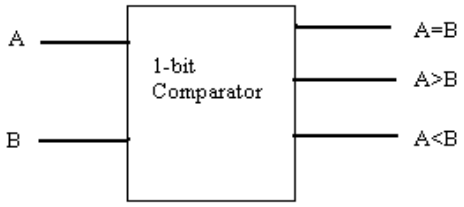
Types of Magnitude Comparators

There are different kinds of magnitude comparators which include the following.

1-bit Magnitude Comparator

A comparator that compares two binary bits and produces three outputs based on the relative magnitudes of given binary bits is called a 1-bit magnitude comparator.

Block Diagram



Truth **Table**

Inputs		*Outputs*		
A	*B*	*A > B*	*A = B*	*A < B*
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

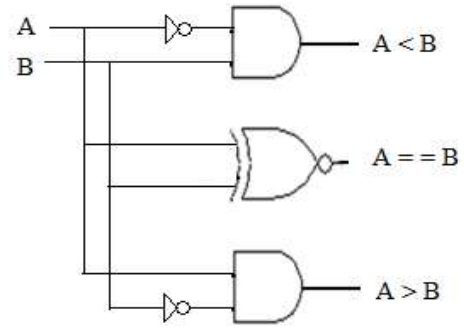
The truth table derives the expressions of A<B, A>B and A=B as below

A<B – A'B

A>B – AB'

A=B – A'B'+AB= AΘB

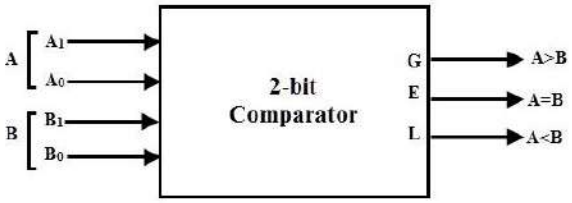
With these expressions, the Circuit diagram can be as follows



2-bit Magnitude Comparator

A comparator that compares two binary numbers (each number having 2 bits) and produces three outputs based on the relative magnitudes of given binary bits is called a 2-bit magnitude comparator.

Block Diagram



Truth Table

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

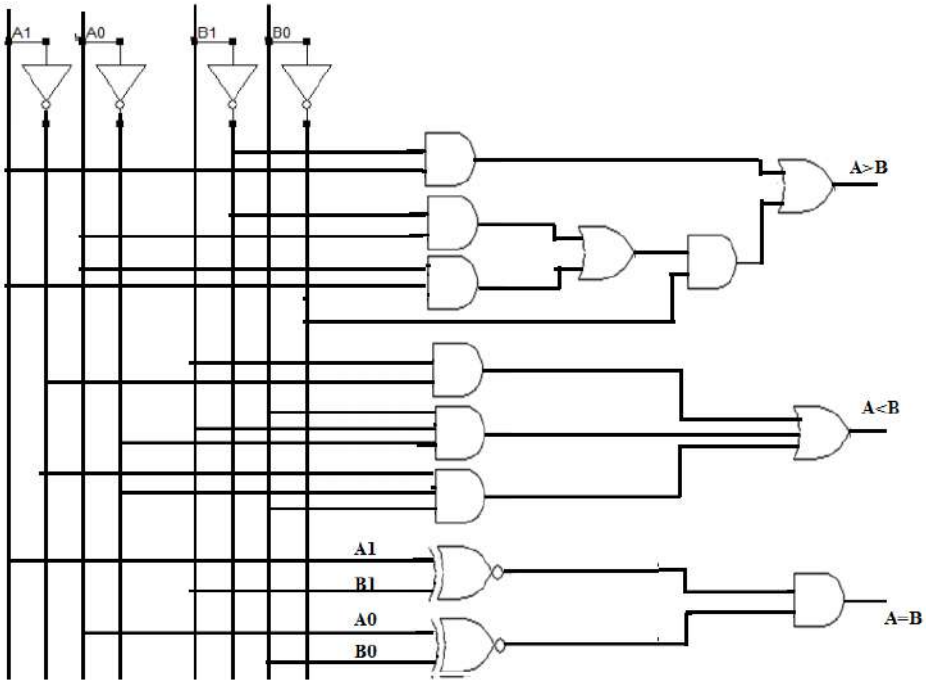
The truth table derives the expressions of A<B, A>B, and A=B as below

$A<B – A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$

$A>B – A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

$A=B – (A_0ΘB_0)\ (A_1ΘB_1)$

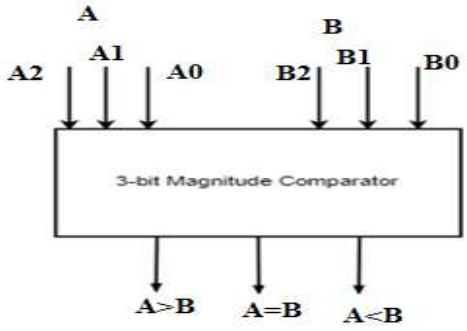
With these expressions, the Circuit diagram can be as follows



3-bit Magnitude Comparator

A comparator that compares two binary numbers (each number having 3 bits) and produces three outputs based on the relative magnitudes of given binary bits is called a 3-bit magnitude comparator.

Block Diagram



The output is **A= B** in the cases of

$A_0 = B_0$, $A_1 = B_1$, $A_2 = B_2$

Then **$A=B = (A_0 \odot B_0)(A_1 \odot B_1)(A_2 \odot B_2)$**

The output is **A< B** in the cases of

$A_2 < B_2$

$A_2 = B_2$ then **$A_1 < B_1$**

$A_2 = B_2$, $A_1 = B_1$ then **$A_0 < B_0$**

$A<B = A_2'B_2 + [(A_2 \odot B_2) \times A_1'B_1] + [(A_2 \odot B_2) \times [(A_1 \odot B_1) \times A_0'B_0]$

The output is **A> B i**n the cases of

$A_2 > B_2$

$A_2 = B_2$ then **$A_1 > B_1$**

$A_2 = B_2$, $A_1 = B_1$ then $A_0 > B_0$

$A>B = A_2B_2' + + [(A_2 \odot B_2) \times A_1B_1'] + [(A_2 \odot B_2) \times [(A_1 \odot B_1) \times A_0B_0']$

Circuit Diagram

MULTIPLEXER

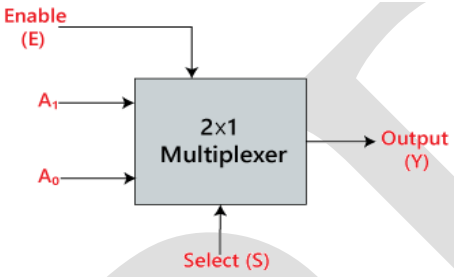
- **Multiplexer** is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.
- Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**

There are various types of the multiplexer which are as follows:

2×1 Multiplexer:

In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S, one of these 2 inputs will be connected to the output.

Block Diagram:



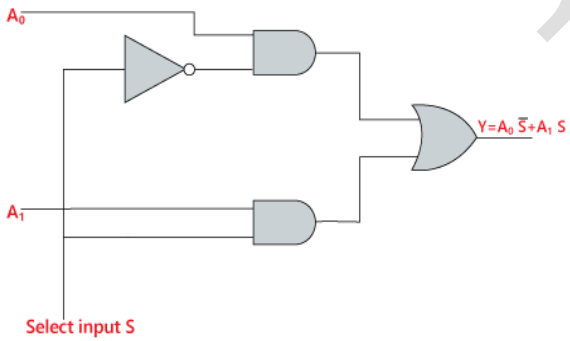
Truth Table:

INPUTS	Output
S_0	Y
0	A_0
1	A_1

The logical expression of the term Y is as follows:

$Y=S_0'.A_0+S_0.A_1$

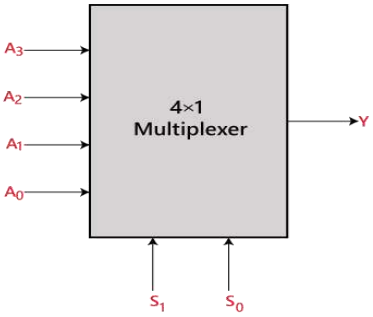
Logical circuit of the above expression is given below:



4×1 Multiplexer:

In the 4×1 multiplexer, there is a total of four inputs, i.e., A_0, A_1, A_2, and A_3, 2 selection lines, i.e., S_0 and S_1 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S^0 and S_1, one of these 4 inputs are connected to the output.

Block Diagram:



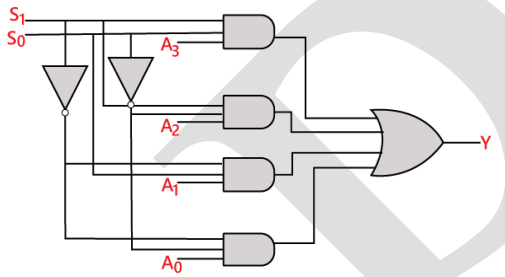
Truth Table:

INPUTS		Output
S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

The logical expression of the term Y is as follows:

$Y=S_1' S_0' A_0+S_1' S_0 A_1+S_1 S_0' A_2+S_1 S_0 A_3$

Logical circuit of the above expression is given below:



DE-MULTIPLEXER

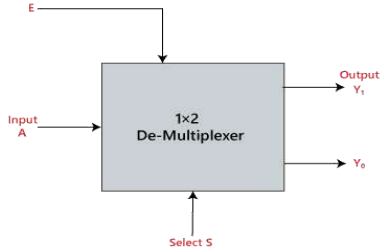
- De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, 'n' selection lines and maximum of 2^n outputs. The input will be connected to one of these outputs based on the values of selection lines.
- Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as De-Mux.

There are various types of De-multiplexer which are as follows:

1×2 De-multiplexer:

In the 1 to 2 De-multiplexer, there are only two outputs, i.e., Y_0, and Y_1, 1 selection lines, i.e., S_0, and single input, i.e., A. On the basis of the selection value, the input will be connected to one of the outputs.

Block Diagram:



Truth Table:

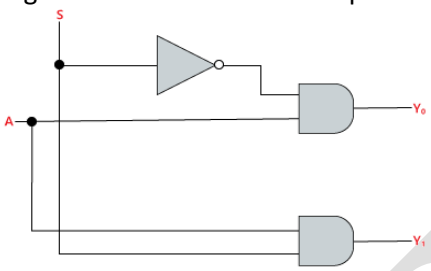
INPUTS	Output	
S_0	Y_1	Y_0
0	0	A
1	A	0

The logical expression of the term Y is as follows:

$Y_0=S_0'.A$

$Y_1=S_0.A$

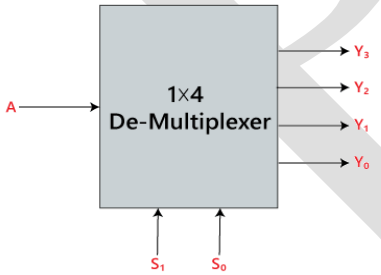
Logical circuit of the above expressions is given below:



1×4 De-multiplexer:

In 1 to 4 De-multiplexer, there are total of four outputs, i.e., Y_0, Y_1, Y_2, and Y_3, 2 selection lines, i.e., S_0 and S_1 and single input, i.e., A. On the basis of the combination of inputs which are present at the selection lines S_0 and S_1, the input be connected to one of the outputs.

Block Diagram:



Truth Table:

INPUTS		Output			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

The logical expression of the term Y is as follows:

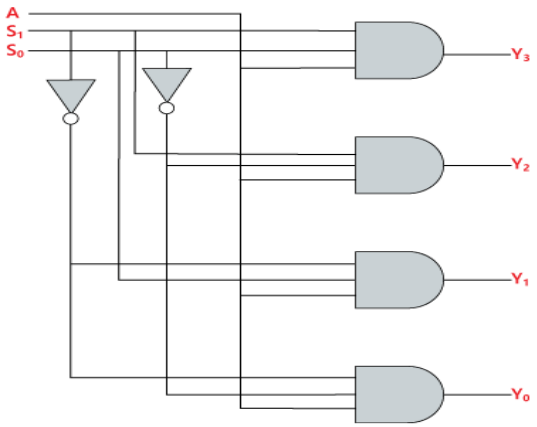
$Y_0=S_1'\ S_0'\ A$

$y_1=S_1'\ S_0\ A$

$y_2=S_1\ S_0'\ A$

$y_3=S_1\ S_0\ A$

Logical circuit of the above expressions is given below:



2.3 Seven segment Decoder (Definition, relevance, gate level of circuit Logic circuit, truth table, Applications of above)

- Seven segment displays are the output display device that provides a way to display information in the form of image or text or decimal numbers.
- One of the most commonly used decoder drive is a BCD to 7-segment decoder. In this decoder the outputs are used to drive seven segments.
- It is widely used in digital clocks, basic calculators, electronic meters, and other electronic devices that display numerical information.
- BCD to seven segment decoder has four input lines A, B, C and D and 7 output lines a, b, c, d, e, f and g. As the numbers which are greater than 9 are not a permitted combination it is assumed that they will not occur.

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

From the above truth table, the Boolean expressions of each output functions can be written as

$a = F_1 (A, B, C, D) = \sum m (0, 2, 3, 5, 7, 8, 9)$

$b = F_2 (A, B, C, D) = \sum m (0, 1, 2, 3, 4, 7, 8, 9)$

$c = F_3 (A, B, C, D) = \sum m (0, 1, 3, 4, 5, 6, 7, 8, 9)$

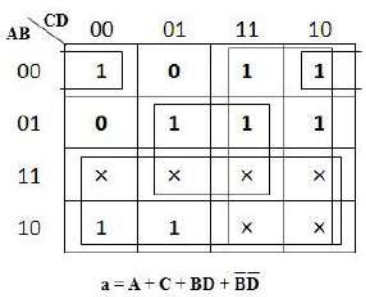
$d = F_4 (A, B, C, D) = \sum m (0, 2, 3, 5, 6, 8)$

$e = F_5 (A, B, C, D) = \sum m (0, 2, 6, 8)$

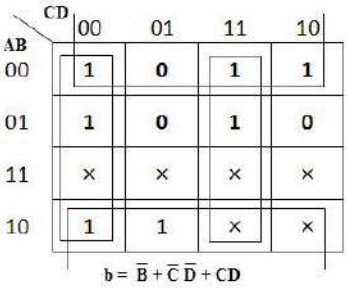
$f = F_6 (A, B, C, D) = \sum m (0, 4, 5, 6, 8, 9)$

$g = F7 (A, B, C, D) = \sum m (2, 3, 4, 5, 6, 8, 9)$

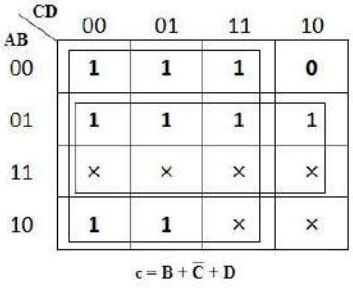
K-Map Simplification



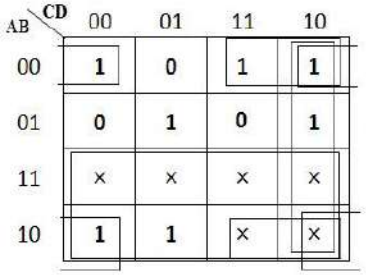
$a = A + C + BD + \overline{B}\,\overline{D}$



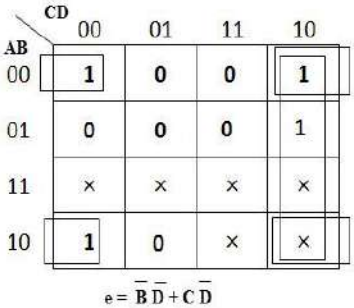
$b = \overline{B} + \overline{C}\,\overline{D} + CD$



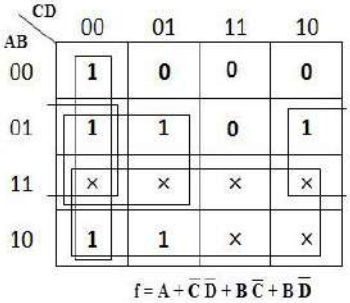
$c = B + \overline{C} + D$



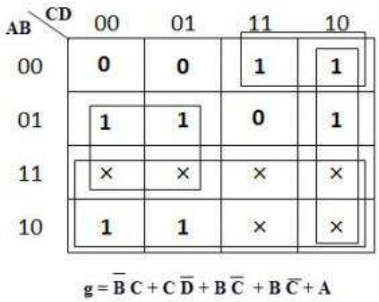
$d = \overline{B}\,\overline{D} + C\,\overline{D} + B\,\overline{C}\,D + \overline{B}\,C + A$



$e = \overline{B}\,\overline{D} + C\,\overline{D}$



$f = A + \overline{C}\,\overline{D} + B\,\overline{C} + B\,\overline{D}$



$g = \overline{B}\,C + C\,\overline{D} + B\,\overline{C} + B\,\overline{C} + A$

From the above simplification, we get the output values as

$a = A + C + BD + \overline{B}\,\overline{D}$

$b = \overline{B} + \overline{C}\,\overline{D} + CD$

$c = B + \overline{C} + D$

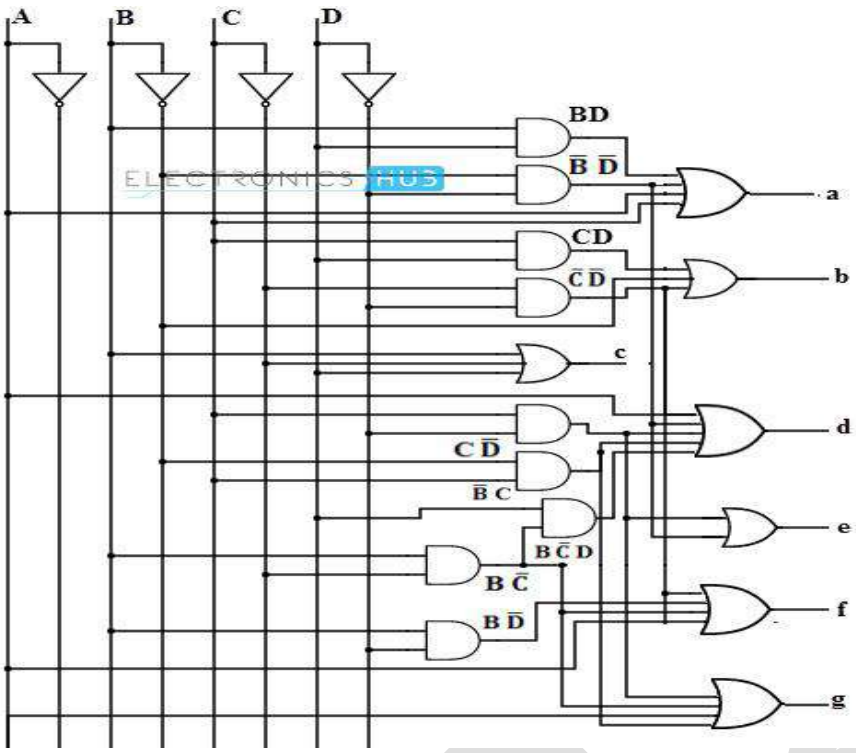
$d = \overline{B}\,\overline{D} + C\,\overline{D} + B\,\overline{C}\,D + \overline{B}\,C + A$

$e = \overline{B}\,\overline{D} + C\,\overline{D}$

$f = A + \overline{C}\,\overline{D} + B\,\overline{C} + B\,\overline{D}$

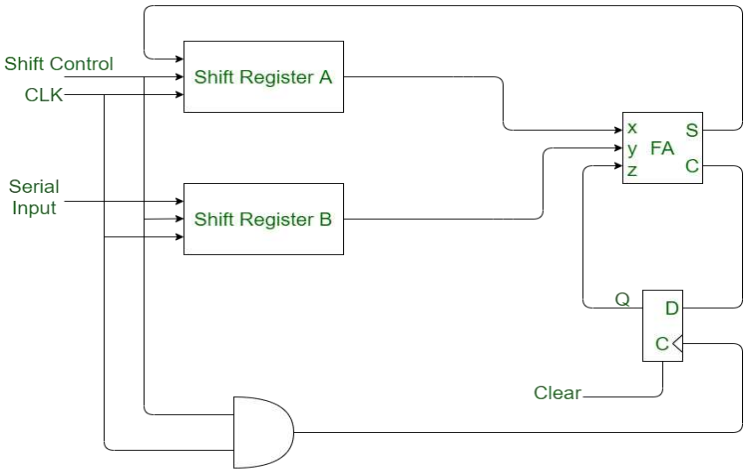
$g = A + B\,\overline{C} + \overline{B}\,C + C\,\overline{D}$

The logic circuit for each output signal of 7- segment display as follow



SERIAL BINARY ADDER

- Serial binary adder is a combinational logic circuit that performs the addition of two binary numbers in serial form. Serial binary adder performs bit by bit addition. Two shift registers are used to store the binary numbers that are to be added.
- A single full adder is used to add one pair of bits at a time along with the carry. The carry output from the full adder is applied to a D flip-flop. After that output is used as carry for next significant bits. The sum bit from the output of the full adder can be transferred into a third shift register.



Shift Registers :

Shift Register is a group of flip flops used to store multiple bits of data. There are two shift registers used in the serial binary adder. In one shift register augend is stored and in other shift register addend is stored.

Full Adder :

Full adder is the combinational circuit which takes three inputs and gives two outputs as sum and carry. The circuit adds one pair at a time with the help of it.

D Flip-flop :

The carry output from the full adder is applied on the D flip-flop. Further, the output of D flip-flop is used as a carry input for the next pair of significant bits.

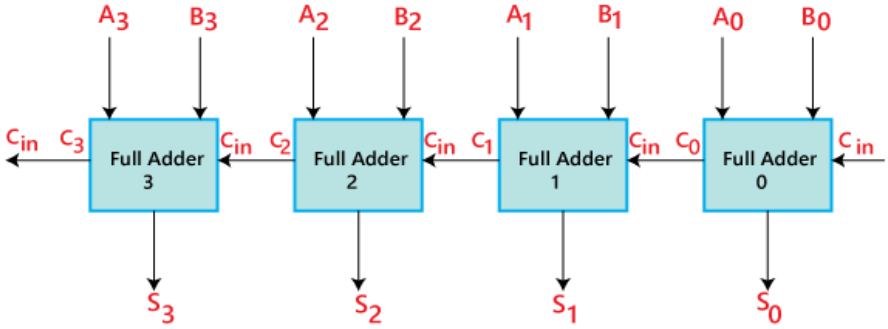
Working Process:

Following is the procedure of addition using serial binary adder:

- **Step-1:**
  The two shift registers A and B are used to store the numbers to be added.
- **Step-2:**
  A single full adder is used to add one pair of bits at a time along with the carry.
- **Step-3:**
  The contents of the shift registers shift from left to right and their output starting from a and b are fed into a single full adder along with the output of the carry flip-flop upon application of each clock pulse.
- **Step-4:**
  The sum output of the full adder is fed to the most significant bit of the sum register.
- **Step-5:**
  The content of sum register is also shifted to right when clock pulse is applied.
- **Step-6:**
  After applying four clock pulse the addition of two registers (A & B) contents are stored in sum register.

PARALLEL BINARY ADDER

- The **Parallel binary adder** is a **combinational circuit** consists of various full adders in parallel structure so that more than 1-bit numbers are to be added efficiently.
- The arrangement of parallel adder can be done by arranging the full adders (FAs) in a chain model where the carry output from every full adder (FA1) can be linked to the carry input of the next full adder (FA2) within the chain.

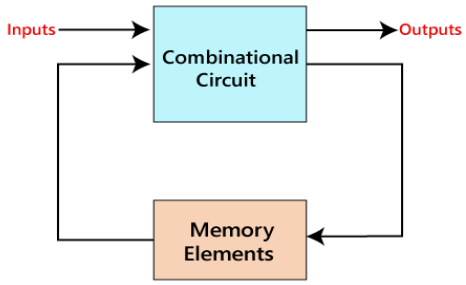


- 4- bit Parallel Adder is designed using 4 Full Adders FA_0, FA_1, FA_2, FA_3.
- The 'A' and 'B' are the augend, and addend bits are defined by the subscript numbers. The subscripts start from right to left, and the lower-order bit is defined by subscript '0'.
- Full Adder FA_0 adds A_0, B_0 along with carry C_{in} to generate Sum S_0 and Carry bit C_0 and this Carry bit is connected to FA_1.
- FA_1 accepts this Carry C_0 and adds with its inputs A_1 and B_1 to generate Sum S_1 and Carry C_1. This bit C_1 is connected to FA_2. This process continues till last Full Adder.
- The S0, S1, S2, and S3 are the sum outputs that produce the sum of augend and addend bits.
- The inputs for the input variable 'A' and 'B' are fetched from different source registers.
- The outcome produced by adding both input variables is stored into either third register or to one of the source registers.

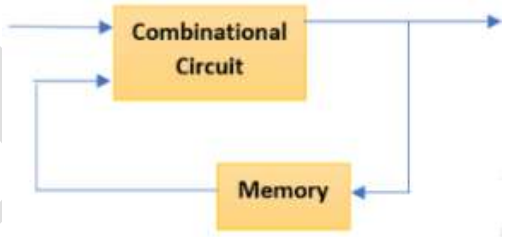
UNIT-3: SEQUENTIAL LOGIC CIRCUITS

The sequential circuit is an Electronic circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs.

So, the sequential circuit contains the combinational circuit and its memory storage elements.

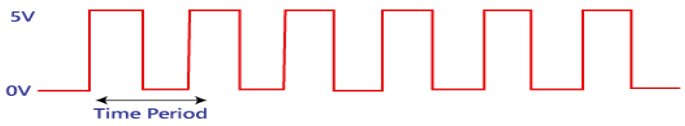


Distinguish between combinational and sequential circuit

Combinational circuit	Sequential circuit
The outputs of the combinational circuit depend only on the present inputs.	The outputs of the sequential circuits depend on both present inputs and present state(previous output).
The feedback path is not present in the combinational circuit	The feedback path is not present in the sequential circuits
In combinational circuits, memory elements are not required.	In the sequential circuit, memory elements play an important role and require.
The clock is not required for combinational circuits	The clock is required for sequential circuit
The combinational circuit is simple to design.	It is not simple to design a sequential circuit.
Example: Adder, Subtractor, Encoder etc	Example: Counter, Register etc
	

Clock signal

A clock signal is nothing but a train of pulses that oscillates between a high and a low state that is, which alternates between 0 and 1 values.



A clock signal is considered as the square wave. Sometimes, the signal stays at logic, either high 5V or low 0V, to an equal amount of time.

Triggering

Triggering basically means switching. It change the current state from low state to high state means from 0 to 1 or vice versa.

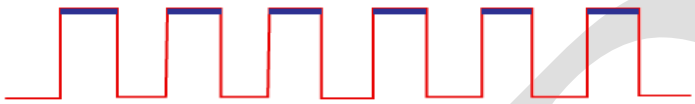
These are two types of triggering in sequential circuits:

1. **Level triggering**

The logic High and logic Low are the two levels in the clock signal. In level triggering, when the clock pulse is at a particular level, only then the circuit is activated. There are the following types of level triggering:

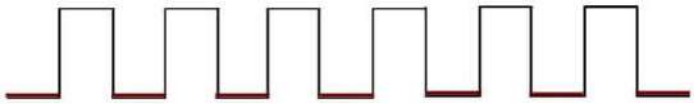
Positive level triggering

In a positive level triggering, the signal with Logic High occurs. So, in this triggering, the circuit is operated with such type of clock signal.



Negative level triggering

In negative level triggering, the signal with Logic Low occurs. So, in this triggering, the circuit is operated with such type of clock signal.



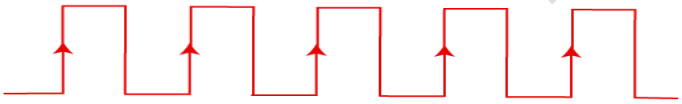
2. Edge triggering

In clock signal of edge triggering, two types of transitions occur, i.e., transition either from Logic Low to Logic High or Logic High to Logic Low.

Based on the transitions of the clock signal, there are the following types of edge triggering:

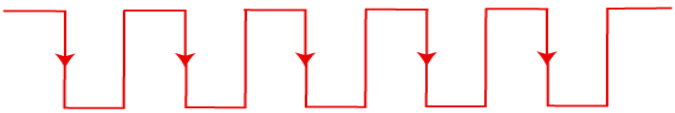
Positive edge triggering

The transition from Logic Low to Logic High occurs in the clock signal of positive edge triggering. So, in positive edge triggering, the circuit is operated with such type of clock signal.



Negative edge triggering

The transition from Logic High to Logic low occurs in the clock signal of negative edge triggering. So, in negative edge triggering, the circuit is operated with such type of clock signal.



3.1 Principle of flip-flops operation, its Types

- A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs.
- Flip flop is formed using logic gates such as two NAND and NOR gates which are in turn made of transistors.
- As flip flops have two stable states, hence they are called as Bistable multivibrators. The two stable states are High (logic 1) and Low (logic 0).
- Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops and latches are used as data storage elements.

Latches vs Flip-Flops

Latch	Flip Flop
A Latch is a bistable device, and the state of the latch is represented as 0 and 1.	Flip-Flop is also a bistable device and there are two stable states of Flip-Flop, which are represented as 0 and 1.
The clock signal is absent.	The clock signal is present.
A Latch is a level triggered device.	A Flip Flop is a level triggered device.
Latches are fast as compared to the Flip-Flop.	Flip-Flops are slow as compared to the latches.
It requires comparatively less power.	It requires more power.
The latch is asynchronous because latch does not work on the basis of the time signal.	Flip-Flop is synchronous because flip-flop work on the basis of the clock signal.

Types of flip flops

Based on their operations, flip flops are basically 4 types. They are

i. S-R flip flop

ii. D flip flop

iii. J-K flip flop

iv. T flip flop

3.2 SR Flip Flop using NAND, NOR Latch (unclocked)

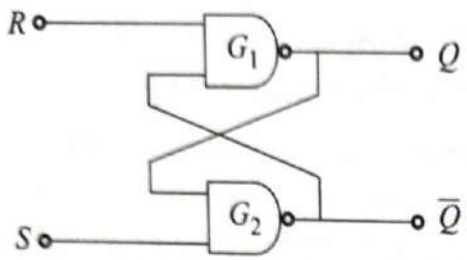
The **SR flip-flop** has two inputs, one is “SET” labelled **S** and other is “RESET” labelled **R**.

The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level “1” or logic “0” depending upon this set/reset condition.

Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to its current state or history. The term “Flip-flop” relates to the actual operation of the device, as it can be “flipped” into one logic Set state or “flopped” back into the opposing logic Reset state.

SR Flip Flop using NAND:

The basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates, there is feedback from each output to one of the other NAND gate inputs. It consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement Q̄.



S	R	Q	$\bar{Q}$
0	0	Race	Race
0	1	0	1
1	0	1	0
1	1	Same as previous	Same as previous

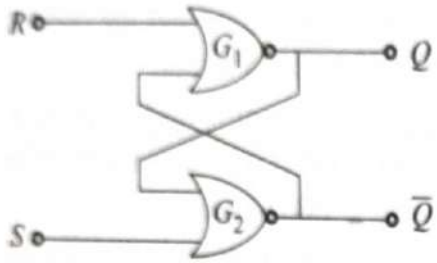
Operation

- When S=0 and R= 0, then both the outputs of gate G1 and G2 try to become 1, which is not possible. There is a race between the outputs.
- When S=0 and R= 1, then the output of G2 is 1 and that of G1 is 0 (Q=0) and the memory is said to be reset.
- When S=1 and R= 0, then the output of G2 is 0 and that of G1 is 1 (Q=1) and the memory is said to be set.
- When S=1 and R= 1, then the output remains unchanged.

According to inputs S and R, the data logic 0 or logic 1 can be loaded or stored into the logic circuit.

SR Flip Flop using NOR:

The basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NOR gates, there is feedback from each output to one of the other NOR gate inputs. It consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement Q̄.



S	R	Q	$\bar{Q}$
0	0	Same as previous	Same as previous
0	1	0	1
1	0	1	0
1	1	Race	Race

Operation

- When S=0 and R= 0, the output remains unchanged.
- When S=0 and R= 1, then the output of G1is 0 (Q=0) and that of G2 is 1 and the memory is reset.
- When S=1 and R= 0, then the output of G1 is 1 (Q=1) and that of G2 is 0 and the memory is set.
- When S=1 and R= 1, then the outputs of both the gates try to become 0, which is not possible, and there is a race between the outputs.

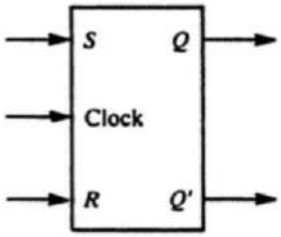
3.3 C l o c k e d SR, D, JK, T, JK Master Slave flip-flops-Symbol, logic Circuit, truth table and applications

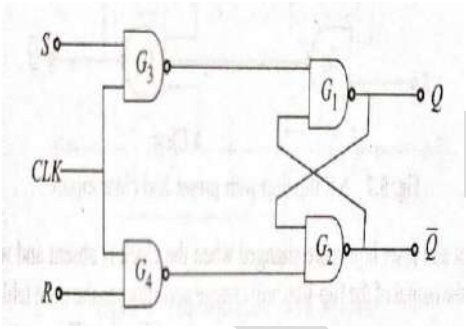
Clocked SR flip-flop

A clocked SR flip-flop is designed by adding two NAND gates to a basic NAND gate un-clocked flip-flop. These two NAND gates are used to apply clock pulse. The simple S-R flip flop will be set or reset any time by changing the inputs S and R.

Operation:

When the clock is absent (CLK=0), the outputs of both the gates G3 and G4 are 1 and there is no change in the outputs of G1 and G2. When the clock is present (CLK=1), the outputs of G3 and G4 are the function of the inputs S and R.





C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

- When S=0 and R=0, then the outputs of G3 and G4 are 1, the outputs of G1 and G2 are unchanged.
- When S=0 and R=1, then the output of G3 is 1 and that of G4 is 0, the output of G1 is 0 and that of G2 is 1. The flip-flop is reset.
- When S=1 and R=0, then the output of G3 is 0 and that of G4 is 1, the output of G1 is 1 and that of G2 is 0. The flip-flop is set.
- When S=1 and R=1, then the outputs of G3 and G4 are 0, and the outputs of G1 and G2 try to become1, which is not possible. There is a race and the output is undefined and it is known as forbidden state.

S_n and R_n are the present inputs, Q_n is the present output, and Q_{n+1} is the output after the clock.

Case 1: For S=0, R=0 and CLK=0, the flip-flop simply remains in its present state. That is, Q remains unchanged. Even for S=0, R=0 and CLK=1, the flip-flop remains in its present state. This condition will not affect the outputs of flip-flop.

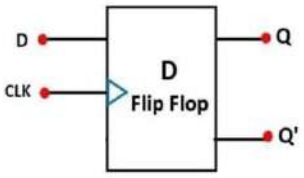
Case 2: For S=0, R=1 and CLK=0, the flip-flop remains in its present state. But, when CLK=1, the NAND gate-1 output will go to 1 and the NAND gate 2 output will go to 0. Now a 0 NAND gate 4 input forces $\bar{Q}$= 1 which in turn results in NAND gate 3 output Q=0. Thus, for S=0, R=1 and CLK=1, the flip-flop RESET to the 0 state.

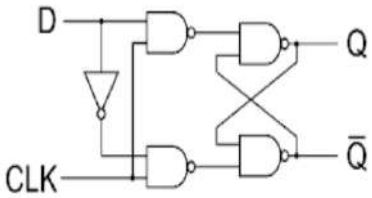
Case 3: For S=1, R=0 and CLK=0, the flip-flop remains in its present state. But for S=1, R=0 and CLK=1, the set state of the flip-flop is reached. This causes the NAND gate 1 output to go 0 and the NAND gate 2 output to 1. Now, a 0 at NAND gate 3 input forces Q to 1 which in turn forces NAND gate 4 output $\bar{Q}$ to 0.

Case 4: An indeterminate condition occurs when all the inputs, namely CLK, S and R are equal to 1. This condition results in 0's in the outputs of gate 1 and 2 and 1's in both outputs Q and $\bar{Q}$. When the CLK input goes back to 0 (while S and R remain at 1), it is not possible to determine the next state, as it depends on whether the output of gate 1 or gate 2 goes to 1 first.

D FLIP-FLOP

The D flip-flop has only one input called the Delay (D) input and outputs Q and Q̄. It can be constructed from an S-R flip-flop by inserting an inverter between S and R and assigning the symbol D to the S input.





D Flip Flop		
Input	Output	
D	Q	Q^
0	0	1
1	1	0

Operation:

- When the CLK input is LOW, the D input has no effect, since the set and reset inputs of the NAND flip-flop are kept HIGH.
- When the CLK goes HIGH, the Q output will take on the value of the D input.
  If CLK=1 and D=1, the NAND gate-1 output goes 0 and output of NAND gate goes 1. The basic NAND S-R flip-flop output will be 1, i.e it follows D input. Similarly, for CLK =1 and D =0, the flip-flop output will be 0. If D changes while the CLK is HIGH, Q will follow and change quickly.

It is clear that the next state of the flip-flop at time Q_{n+1} follows the value of the input D when the clock pulse is applied. As transfer of data from the input to the output is delayed, it is known as delay(D) flip-flop.

The delay flip-flop is either used as a delay or as a latch to store 1 bit of binary information.

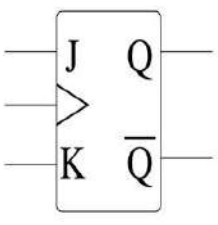
JK FLIP-FLOP:

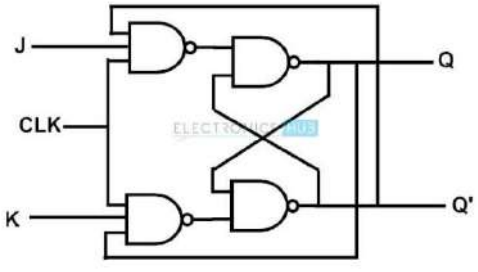
- JK flip flop is a modified version of SR flip flop. Logic diagram consists of three input NAND gates replacing the two input NAND gates in SR flip flop and the inputs are replaced with J and K from S and R.
- The design of the JK flip flop is such that the three inputs to one NAND gate are J, clock signal along with a feedback signal from Q' and the three inputs to the other NAND are K, clock signal along with a feedback signal from Q. This arrangement eliminates the indeterminate state in SR f lip flop.

Operation

- When both the inputs J and K are LOW, then Q returns its previous state value i.e. it holds the previous data.

When we apply a clock pulse to the J K flip flop and the J input is low then irrespective of the other NAND gates, the NAND gate-1 output becomes HIGH. In the same manner, if the K input is low then output of NAND gate-2 is also HIGH. So thus the output remains in the same state i.e. no change in the state of flip flop.





J	K	$Q_{(t+1)}$	
0	0	$Q_{(t)}$	*unchanged*
0	1	0	*reset*
1	0	1	*set*
1	1	$\bar{Q}_{(t)}$	*output inversion*

- When J is LOW and K is HIGH, then flip flop will be in Reset state i.e. Q = 0, Q' = 1.

When we apply a clock pulse to the J K flip flop and the inputs are J is low and K is high the output of the NAND gate connected to J input becomes 1. Then Q becomes 0. This will reset the flip flop again to its previous state. So the Flip flop will be in RESET state.

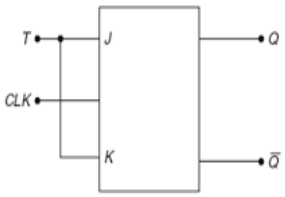
- When J is HIGH and K is LOW, then flip – flop will be in Set state i.e. Q = 1, Q' = 0

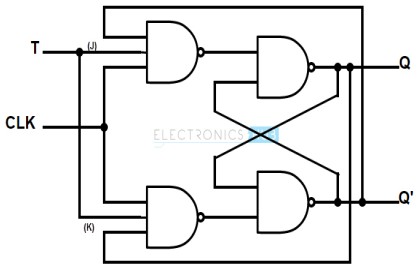
When we apply a clock pulse to the J K flip flop and the inputs are J is high and K is low the output of the NAND gate connected to K input becomes 1. Then Q' becomes 0. This will set the flip flop with the high clock input. So the Flip flop will be in SET state.

- When both the inputs J and K are HIGH, then flip – flop is in Toggle state. This means that the output will complement of the previous state.

T FLIP-FLOP:

A T flip flop is basically a single input version of JK flip flop. This modified form of JK flip-flop is obtained by connecting both inputs J and K together. This flip-flop has only one input along with the clock input. These flip-flops are called T flip-flops because of their ability to complement its state (i.e.) Toggle, hence the name Toggle flip-flop.





T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

- Here J and K input of the JK Flip-Flop is connected together and given to the T input.
- If the T input is in 0 state (i.e., J = K = 0) prior to a clock pulse, the Q output will not change with the clock pulse.
- If the T input is in 1 state (i.e., J = K = 1) prior to a clock pulse, the Q output will change to Q' with the clock pulse.
- We can conclude that if T = 1 and the device is clocked, then the output toggles its state.

Applications of flip flops.

- Counters
- Frequency Divider Circuit
- Shift Registers
- Storage Registers
- Data storage
- Data transfer

3.4 Concept of Racing and how it can be avoided

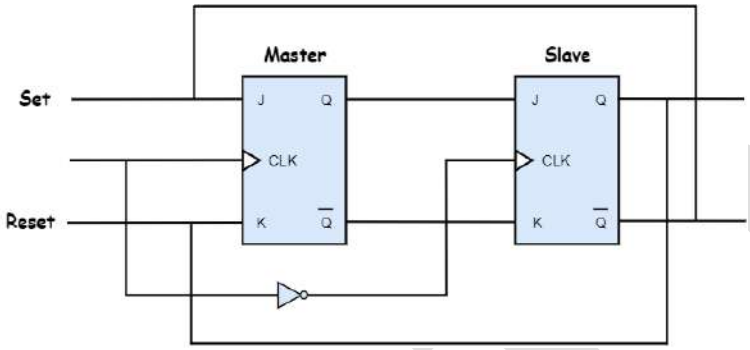
When J=K=1 and if Q=0, a clock pulse of width t_p is applied, the output will change from 0 to 1 after a time interval Δt, where Δt is the propagation delay through two NAND gates in series. Now after Δt, we have J=K=1 and Q=1 and after, another interval of Δt, output Q will become 0. Hence the output will oscillate back and forth between 0 and 1 in the duration t_p of the clock pulse width. So at the end of the clock pulse, the value of Q is ambiguous. This situation is known as Race- Around condition.

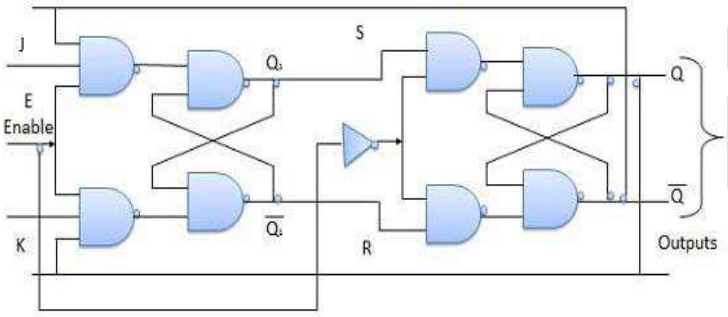
The race around condition can be avoided if $t_p<\Delta t$. Lumped delay lines can be used in series with the feedback connections in order to increase the loop delay beyond and hence to prevent the race around difficulty. The race around condition can also be avoided in Master- Slave flip-flop.

MASTER SLAVE JK FLIP FLOP:

- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration.
- The first flip-flop is called the *master*, and it is driven by the positive clock cycle. The second flip-flop is called the *slave*, and it is driven by the negative clock cycle.
- During the positive clock cycle, master flip-flop gives the intermediate output but slave flip-flop will not give the final output.
- During the negative clock cycle, slave flip-flop gets activated and copies the previous output of the master flip-flop and produces the final output.

Circuit Diagram





Truth Table

Inputs			Outputs		Comments
E	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	0	Q_n	$\overline{Q}_n$	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	$\overline{Q}_n$	Q_n	Toggle

Working of a master slave flip flop

J = K = 0 (No change)

- When clock = 0, the slave becomes active and master is inactive. But since the S and R inputs have not changed, the slave outputs will also remain unchanged. Therefore, outputs will not change if J = K =0.

J = 0 and K = 1 (Reset)

- Clock = 1 − Master active, slave inactive. Therefore, outputs of the master become Q_1 = 0 and Q_1 bar = 1. That means S = 0 and R =1.
- Clock = 0 − Slave active, master inactive. Therefore, outputs of the slave become Q = 0 and Q bar = 1.

- Again clock = 1 − Master active, slave inactive. Therefore, even with the changed outputs Q = 0 and Q bar = 1 fed back to master, its output will be Q1 = 0 and Q1 bar = 1. That means S = 0 and R = 1.
- Hence with clock = 0 and slave becoming active the outputs of slave will remain Q = 0 and Q bar = 1. Thus we get a stable output from the Master slave.

J = 1 and K = 0 (Set)

- Clock = 1 − Master active, slave inactive. Therefore, outputs of the master become Q_1 = 1 and Q_1 bar = 0. That means S = 1 and R =0.
- Clock = 0 − Slave active, master inactive. Therefore, outputs of the slave become Q = 1 and Q bar = 0.
- Again clock = 1 − then it can be shown that the outputs of the slave are stabilized to Q = 1 and Q bar = 0.

J = K = 1 (Toggle)

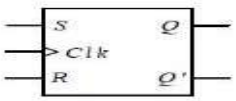
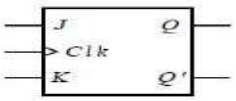
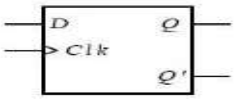
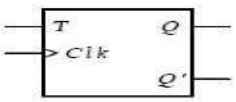
- Clock = 1 − Master active, slave inactive. Outputs of master will toggle. So S and R also will be inverted.
- Clock = 0 − Slave active, master inactive. Outputs of slave will toggle.

These changed output are returned back to the master inputs. But since clock = 0, the master is still inactive. So it does not respond to these changed outputs. This avoids the multiple toggling which leads to the race around condition. The master slave flip flop will avoid the race around condition.

****

Characteristics Table: It defines the next state of flip-flop in terms of flip-flop input and current state.

Excitation Table: It defines the flip-flop input variable as function of the current state and next state.

Name / Symbol	Characteristic (Truth) Table	Excitation Table
SR 	S R Q Qnext 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 × 1 1 1 ×	Q Qnext S R 0 0 0 × 0 1 1 0 1 0 0 1 1 1 × 0
JK 	J K Q Qnext 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0	Q Qnext J K 0 0 0 × 0 1 1 × 1 0 × 1 1 1 × 0
D 	D Q Qnext 0 × 0 1 × 1	Q Qnext D 0 0 0 0 1 1 1 0 0 1 1 1
T 	T Q Qnext 0 0 0 0 1 1 1 0 1 1 1 0	Q Qnext T 0 0 0 0 1 1 1 0 1 1 1 0

Conversion of Flip Flops

- Consider the characteristic table of desired flip-flop.
- Obtain the Excitation Table for the given Flip-Flop from its Truth Table
- Get the simplified expressions for each excitation input. If necessary, use K-Maps for simplifying.
- Draw the circuit diagram of simplified expressions.

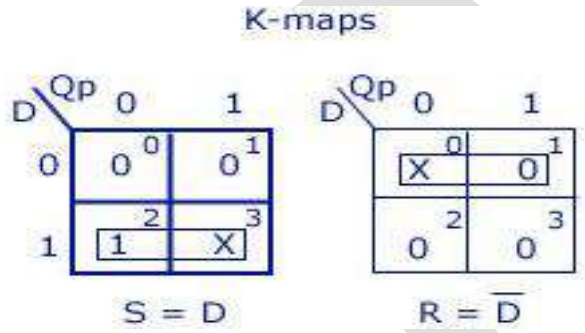
SR Flip Flop to D Flip Flop

Here, the given flip-flop is SR flip-flop and the desired flip-flop is D flip-flop. The following table shows the characteristic table of D flip-flop along with the excitation inputs of SR flip-flop.

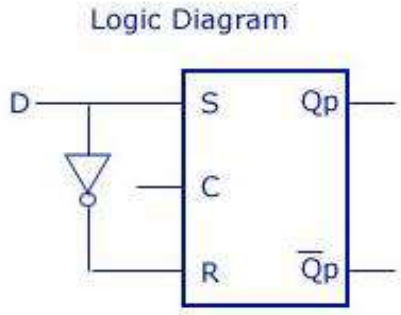
Conversion Table

D Input	Outputs Qp	Outputs Qp+1	S-R Inputs S	S-R Inputs R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

The **k-Maps** for S & R are shown below.



The **circuit diagram** of D flip-flop is shown in the following figure.



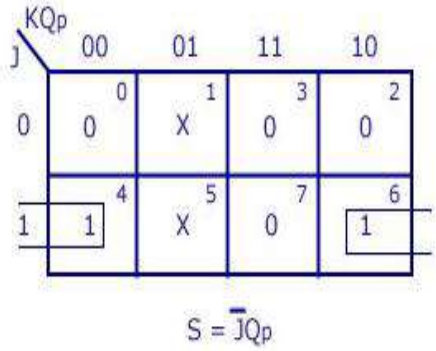
SR Flip Flop to JK Flip Flop

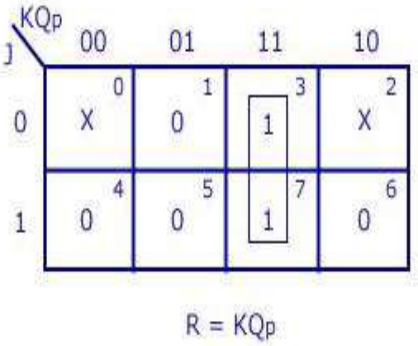
Here, the given flip-flop is SR flip-flop and the desired flip-flop is JK flip-flop. The following table shows the characteristic table of JK flip-flop along with the excitation inputs of SR flip-flop.

Conversion Table

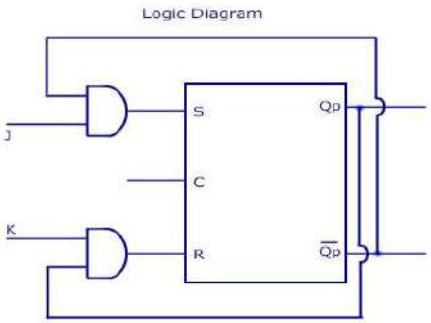
J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

The **k-Maps** for S & R are shown below





The **circuit diagram** of JK flip-flop is shown in the following figure.

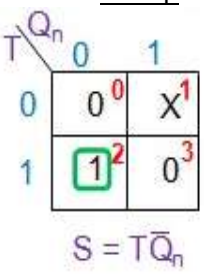


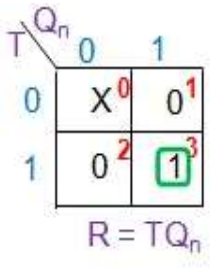
SR Flip Flop to T Flip Flop

Conversion Table

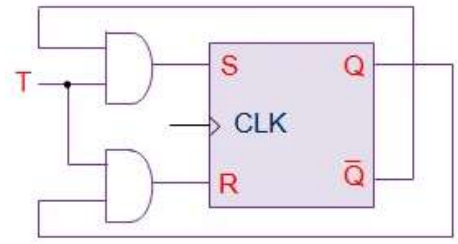
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

K-Map





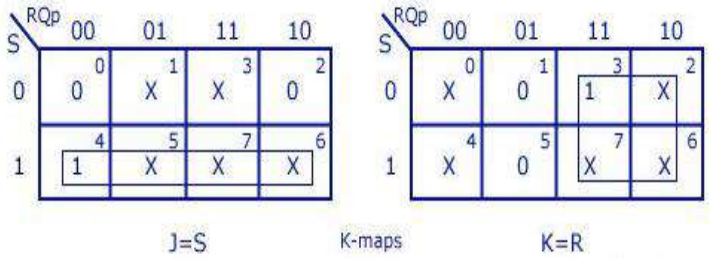
Circuit Diagram



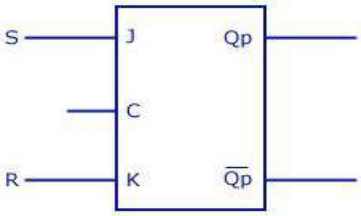
JK Flip Flop to SR Flip Flop

Conversion Table

S-R Inputs S	R	Outputs Qp	Qp+1	J-K Inputs J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	



Logic Diagram

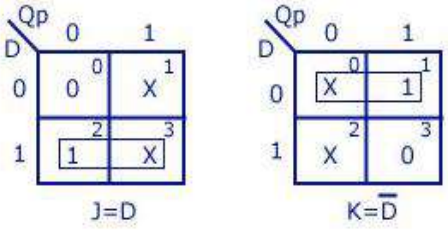


JK Flip Flop to D Flip Flop

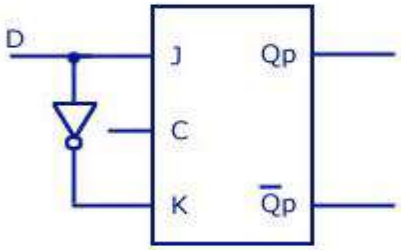
Conversion Table

D Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps



Logic Diagram

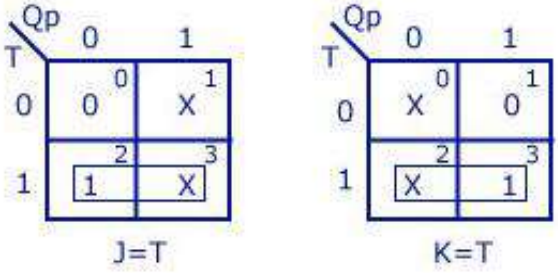


JK Flip Flop to T Flip Flop

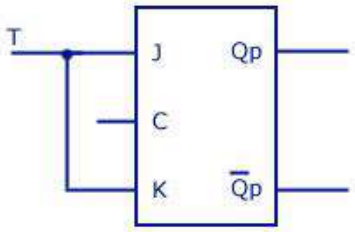
Conversion Table

T Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps



Logic Diagram

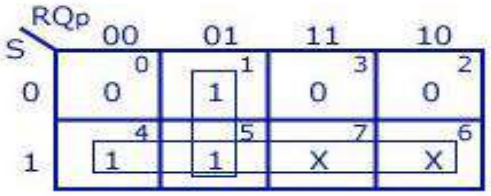


D Flip Flop to SR Flip Flop

Conversion Table

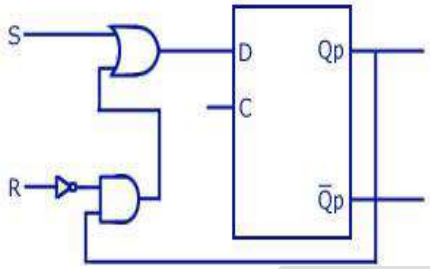
S-R Inputs S	R	Outputs Qp	Qp+1	D Input
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	Invalid		Dont care
1	1	Invalid		Dont care

K-map



$D = S+\bar{R}Q_n$

Logic Diagram

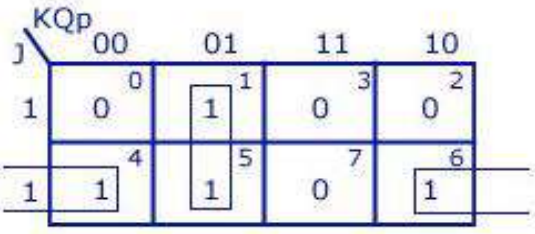


D Flip Flop to JK Flip Flop

Conversion Table

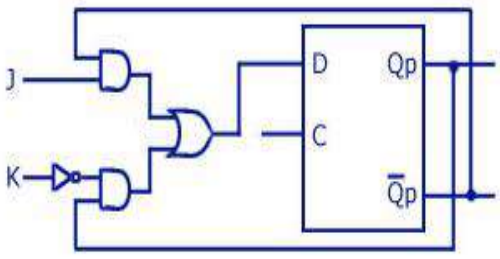
J-K Input J	K	Outputs Qp	Qp+1	D Input
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

K-map



$D = J\bar{Q}p + \bar{K}Qp$

Logic Diagram

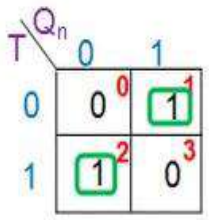


D Flip Flop to T Flip Flop

Conversion Table

T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

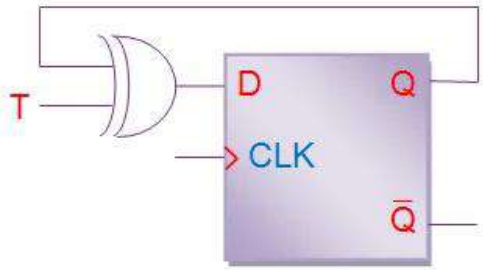
K-Map



$D = T\bar{Q}_n + \bar{T}Q_n$

$= T \oplus Q_n$

Circuit Diagram

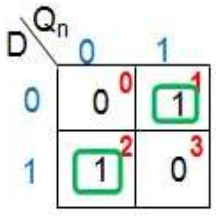


T Flip Flop to D Flip Flop

Conversion Table

D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

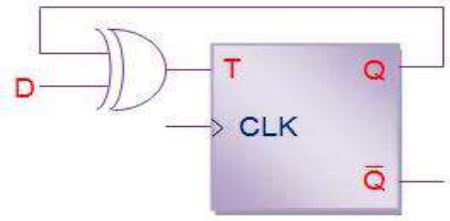
K-Map



$D = D\bar{Q}_n + \bar{D}Q_n$

$= D \oplus Q_n$

Circuit Diagram

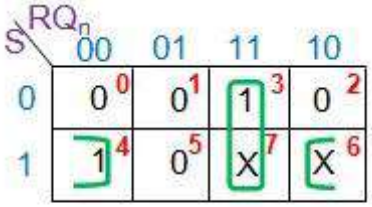


T Flip Flop to SR Flip Flop

Conversion Table

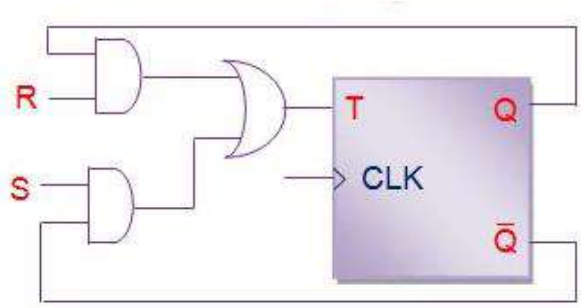
K-Map

S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	invalid		X
1	1	invalid		X



$$T = S\bar{Q}_n + RQ_n$$

Circuit Diagram

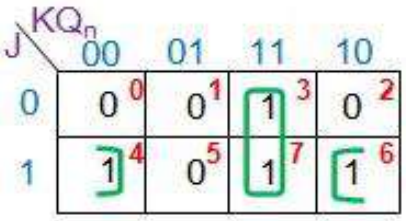


T Flip Flop to JK Flip Flop

Conversion Table

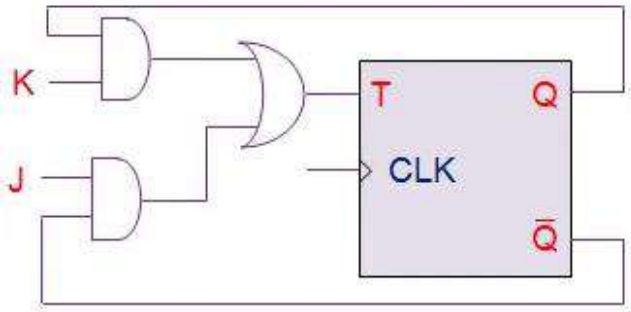
J	K	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

K-Map



$$T = J\bar{Q}_n + KQ_n$$

Circuit Diagram



UNIT-4: REGISTERS, MEMORIES & PLD

Flip-flop is a single bit memory cell which can be used for storing the digital data. To increase the storage capacity in terms of number of bits, group of flip-flop is used. Such a group of flip-flop is known as a Register. The n-bit register will consist of **n** number of flip-flop and it is capable of storing an n-bit word.

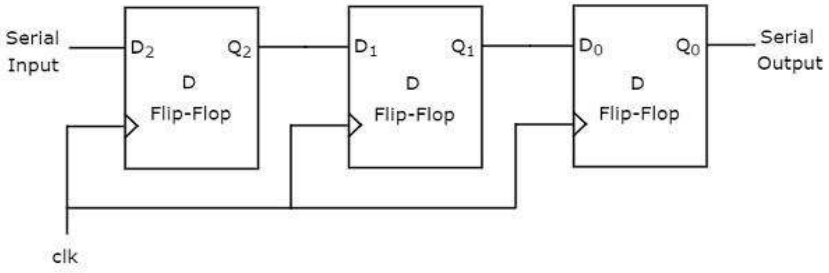
The binary data in a register can be moved within the register from one flip-flop to another. The registers that allow such data transfers are called as shift registers. There are four mode of operations of a shift register.

- Serial Input Serial Output
- Serial Input Parallel Output
- Parallel Input Serial Output
- Parallel Input Parallel Output

4.1 Shift Registers-Serial in Serial -out, Serial- in Parallel-out, Parallel in serial out and Parallel in parallel out

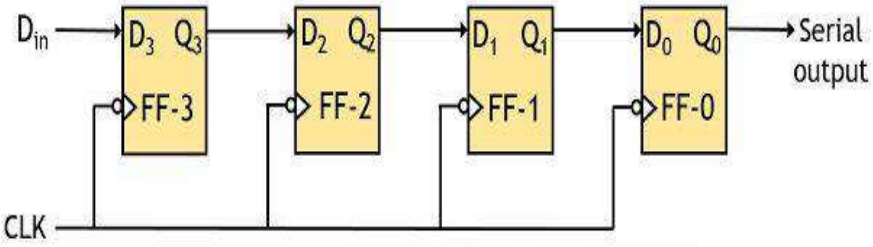
Serial In Serial Out (SISO) Shift Register

- The shift register, which allows serial input and produces serial output is known as Serial In Serial Out (SISO) shift register.



- This block diagram consists of three D flip-flops, which are cascaded. That means, output of one D flip-flop is connected as the input of next D flip-flop.
- All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, the bits are serially send from the input of left most D flip-flop. Hence, this input is also called as serial input.
- For every positive edge triggering of clock signal, the data shifts from one stage to the next. So, the bits are serially received from the output of right most D flip-flop. Hence, this output is also called as serial output.

Operation of SISO Shift Register



Initially, consider all the flip-flops are at reset mode. Thus the output of each flip-flop in the arrangement is logic low i.e., 0.

Here we have assumed a shift right mode circuit as the data input is present at the left end while the stored bit is getting shifted towards the right so as to provide serial output.

Suppose we have to insert '1111' inside the shift register. Initially, as the device is in reset mode thus the output of each register will be low, thereby providing output of all the 4 registers as 0000.

- Now beginning with LSB of the data to be inserted so 1 is provided as input to the circuit i.e., D_3 = 1. But as initially the output of all the flip-flops were 0. Therefore, D_2, D_1 and D_0 will be 0. While input D_3 = 1 will cause Q_3 to be 1. Thus the overall output will be **1000**.
- Further, when another data input bit i.e., 1 is provided at D_3. Then again this will cause Q_3 to be 1, but as Q_3 is provided as input to D_2. Therefore, this will cause Q_2 to be 1, while rest all other outputs will be 0. Thus for a second falling edge, we will get 11 at the stored bit inside the register, thereby giving the overall output as **1100**.
- Similarly, when 3rd input bit '1' is provided then previous output Q_2 will cause the input D_1 to be 1. This will provide output Q_3, Q_2 and Q_1 as 1 while Q_0 will still remain 0. Thus the overall output will be **1110**.
- Furthermore, when MSB of the data is provided as input, then 1 at Q_1 will cause D_0 input to be logic high. So, this will cause Q_0 to be 1.

Thus in this way shift register stores '**1111**'thereby showing in the output.

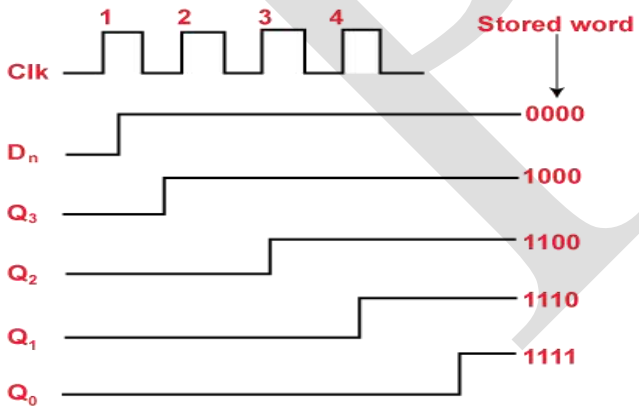
The figure below represents the truth table for the 4-bit SISO shift register:

Truth Table

	Clk	$D_n=Q_3$	$Q_3=D_2$	$Q_2=D_1$	$Q_1=D_0$	Q_0
Initially			0	0	0	0
(1)	↓	1	1	0	0	0
(2)	↓	1	1	1	0	0
(3)	↓	1	1	1	1	0
(4)	↓	1	1	1	1	1

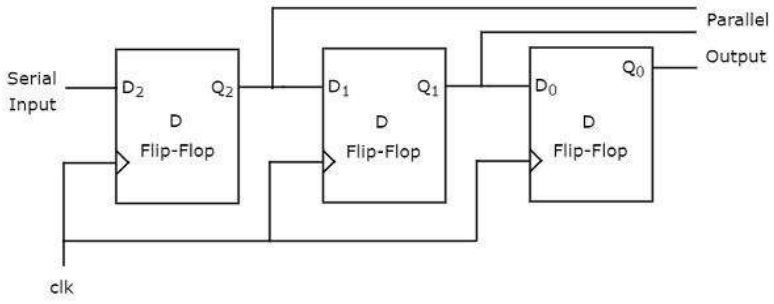
→ Direction of data travel

Waveforms



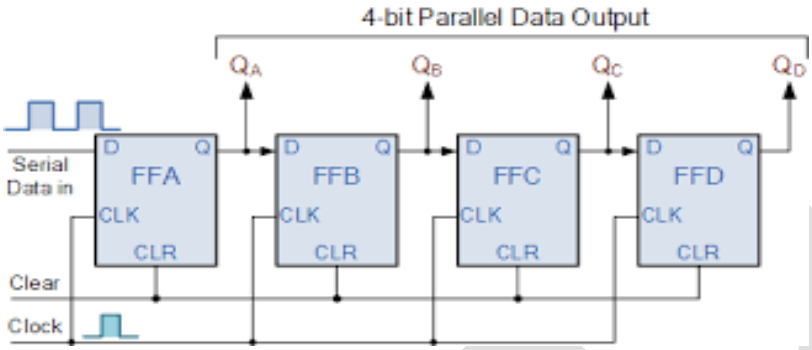
Serial In Parallel Out (SIPO) Shift Register

- The shift register, which allows serial input and produces parallel output is known as Serial In Parallel Out (SIPO) shift register.



- This circuit consists of three D flip-flops, which are cascaded. That means, output of one D flip-flop is connected as the input of next D flip-flop.
- All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, the bits serially send from the input of left most D flip-flop. Hence, this input is also called as **serial input**.
- For every positive edge triggering of clock signal, the data shifts from one stage to the next.
- In this case, the outputs of each D flip-flop access in parallel. So, we get **parallel outputs** from this shift register.

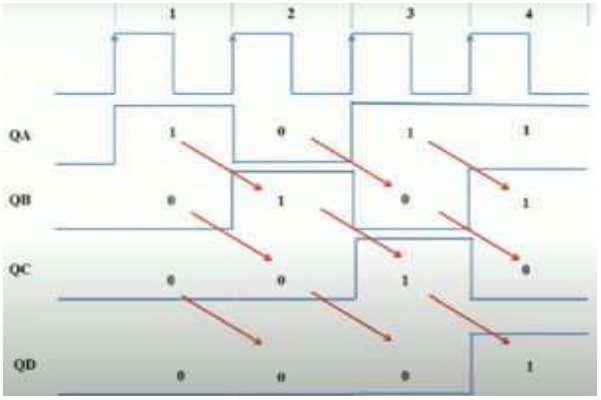
Operation of SIPO Shift Register



- Initially, all the output will become zero so without CLK pulse; all the data will become zero. Let's take a 4-bit data input example like 1101. If we apply the first clock pulse '1' to the first flip flop, the data to be entered into the FF and Q_A becomes '1', and remaining all the outputs like Q_B, Q_C and Q_D will become zero. So the first data output is '1000'
- If we apply the second clock pulse as '0' to the first flip flop then Q_A becomes '0', Q_B becomes '0', Q_C becomes '0' and Q_D becomes '0'. So the second data output will become '0100' due to the shift right process.
- If we apply the third clock pulse as '1' to the first flip flop then Q_A becomes '1', Q_B becomes '0', Q_C becomes '1' and Q_D becomes '0'. So the third data output will become '1010' due to the shift right process.
- If we apply the fourth clock pulse as '1' to the first flip flop then Q_A becomes '1', Q_B becomes '1', Q_C becomes '0' and Q_D becomes '1'. So the third data output will become '1101' due to the shift right process.

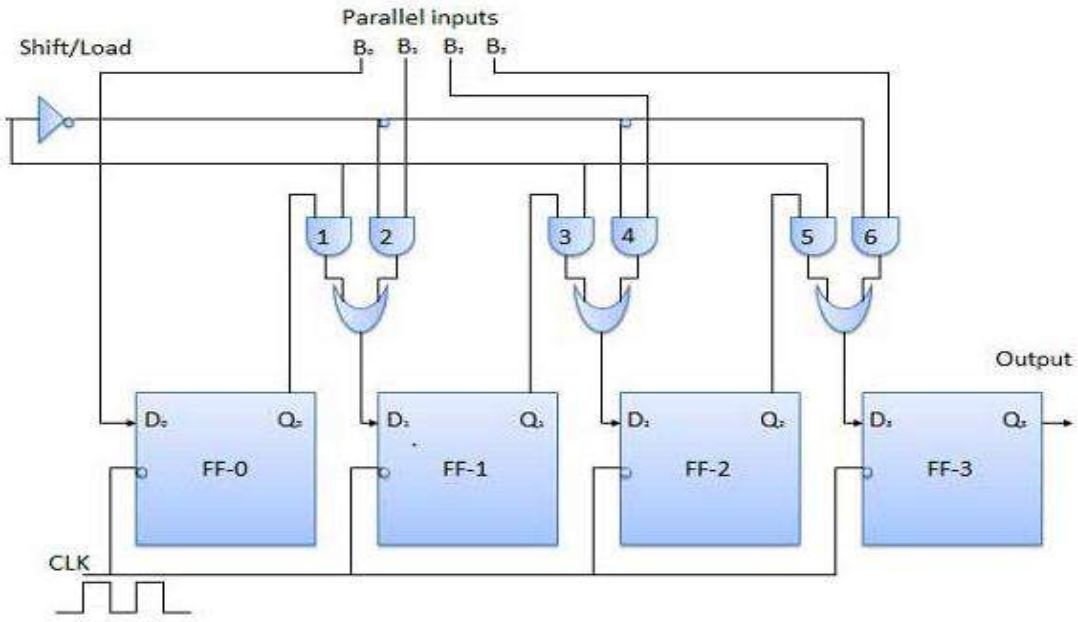
CLK Pulse	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

Timing Diagram



Parallel In Serial Out (PISO) Shift Register

The shift register, which allows parallel input and produces serial output is known as Parallel In Serial Out (PISO) shift register.



In 4bit parallel in serial out shift register, B0, B1, B2, and B3 are the four parallel data input lines and *SHIFT* / *LOAD* (*SH* / *LD*) is a control input that allows the four bits of data at B0, B1, B2, and B3 inputs to enter into the register in parallel or shift the data in serial.

Operation of PISO Shift Register

- In the above-shown PISO shift register circuit, the input data is applied to the input pins of the shift registers from D_0 to D_3 at the same time. After that, it is read out from the shift register serially 1-bit at a time from input pins on every CLK cycle. Here, one CLK pulse is enough to load the 4-bit of data but four pulses are required to unload all the four bits.
- In this parallel input serial output (PISO) shift register circuit, logic gates are used. One control signal (Shift/Load) is used to control the parallel input and serial output. After that, NOT gate outputs are connected to 'G2', 'G4', and 'G6', and the other inputs of G2, G4 & G6 are B1, B2 & B3. Here, 'B0' is directly connected to D0 of the first flip flop.
- The direct control signal is connected to one input of the 'G1', 'G3' & 'G5' and one more input of the 'G1', 'G3' & 'G5' are connected to the outputs of Flip Flops like Q0, Q1, and Q2. The OR gate is connected to the second, third, and fourth Flip Flop's inputs like D_1, D_2, and D_3. All the flip flops are to be connected in a single CLK pulse and the FFs outputs will be in the serial data output.
- Now we are going to see how the data is loaded. Here, we are choosing the input data as 1101 then B0=1, B1=1, B2=0 & B3=1. When the control signal applied to NOT gate is '0' then its o/p will

become‘1’ and G2, G4 & G6 will enable, and ‘G1’, ‘G3’ & ‘G5’ will disable. So all the inputs are loaded and after that OR gates are also enabled and the data is to be loaded to the input of each Flip Flop.

- Now we are applying the control signal ‘1 to NOT gate then the output of this gate will become ‘0’ then G1’, ‘G3’ & ‘G5’ are enabled. Once the CLK pulse is applied to FFs, then the data ‘1101’ is shifted to the right side from one OR gate to the other.

The PISO shift register truth table is shown below.

Now the control signal applied is ‘0’ then the data to be loaded and the data will become 1101.

D_0	D_1	D_2	D_3
1	1	0	1

Now the control signal applied is ‘1’ and the CLK pulse ‘1’ is applied then the data is shifted like Q_A becomes ‘1’, Q_B becomes ‘1’, Q_C becomes ‘0’ and Q_D becomes ‘1’ as shown in the following table.

CLK	Q_A	Q_B	Q_C	Q_D(Data Output)
0	0	0	0	0
1	1	1	0	1
2	0	1	1	0
3	0	0	1	1
4	0	0	0	1

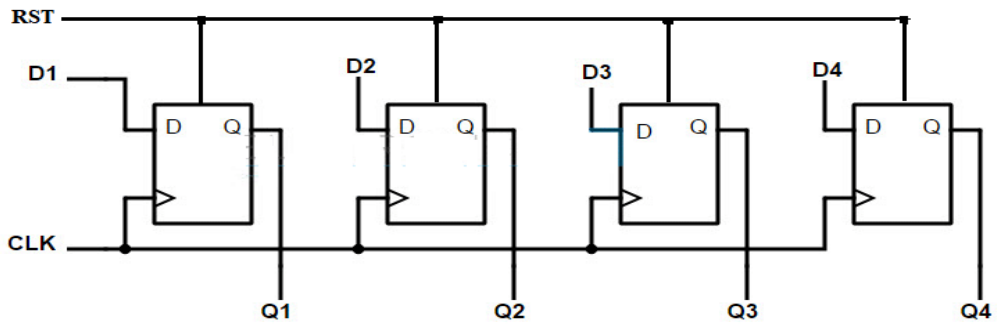
If the CLK pulse ‘2’ is applied then the data is shifted like Q_A becomes ‘0’, Q_B becomes ‘1’, Q_C becomes ‘1’ and Q_D becomes ‘0’ as shown in the following table.
If the CLK pulse ‘3’ is applied then the data is shifted like Q_A becomes ‘0’, Q_B becomes ‘0’, Q_C becomes ‘1’ and Q_D becomes ‘1’ as shown in the following table.
If the CLK pulse ‘4’ is applied then the data is shifted like Q_A becomes ‘0’, Q_B becomes ‘0’, Q_C becomes ‘0’ and Q_D becomes ‘1’ as shown in the following table.

PISO Shift Register Timing Diagram

Do it your self

Parallel In - Parallel Out (PIPO) Shift Register

- The shift register, which allows parallel input and produces parallel output is known as Parallel In − Parallel Out (PIPO) shift register.



- The circuit consists of four D flip-flops which are connected. The clear (CLR/RST) signal and clock signals are connected to all the 4 flip flops.
- In this type of register, there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

- Data is given as input separately for each flip flop and in the same way, output also collected individually from each flip flop.
- From this above diagram, parallel inputs to be applied at D_1, D_2, D_3, and D_4 inputs are directly connected to the D inputs of the respective flip flops.
- On applying the clock transitions, these inputs are entered into the register and are immediately available at the outputs Q_1, Q_2, Q_3, and Q_4.

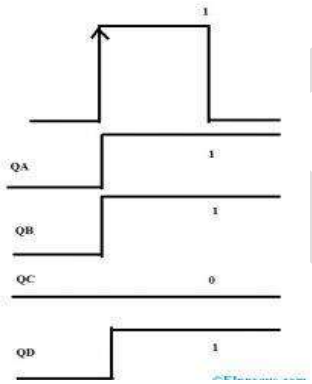
Now if we take the data input is 1101 which is loaded in each flip flop, initially, the output will become 0000. If we apply the first CLK pulse '1' then the input data will be shifted from input D_A to Q_A, so the output will become 1101.

Truth Table

CLK Pulse	Q1	Q2	Q3	Q4
0	0	0	0	0
1	1	1	0	1

Timing Diagram

Here positive edge clock input is used, at that time the transition can take place. So the input data is to be shifted to output, so Q_A is '1', Q_B is '1', Q_C is '0' and Q_D is '1'. This is the output data.



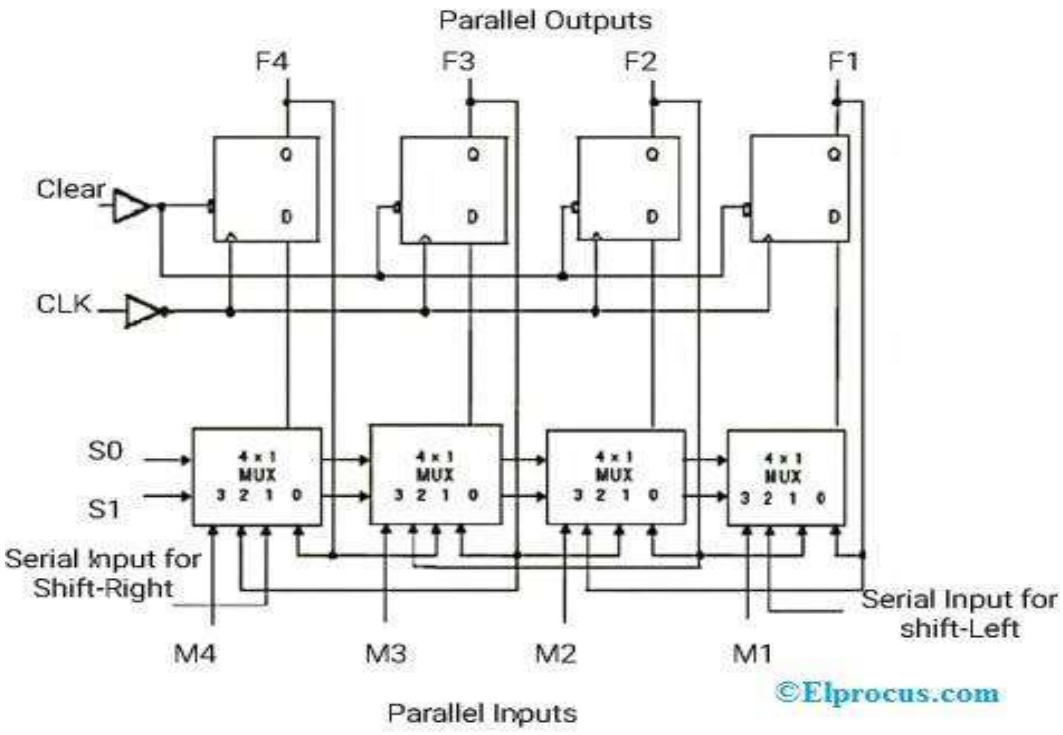
4.2 Universal shift registers-Applications.

- A register that can store the data and shifts the data towards the right and left along with the parallel load capability is known as a universal shift register. It can be used to perform input/output operations in both serial and parallel modes.
- Unidirectional shift registers and bidirectional shift registers are combined together to get the design of the universal shift register.
- It is also known as a parallel-in-parallel-out shift register or shift register with the parallel load.
- Universal shift registers are capable of performing 3 operations as listed below.

- **Parallel load operation –** stores the data in parallel as well as the data in parallel
- **Shift left operation –** stores the data and transfers the data shifting towards left in the serial path
- **Shift right operation –** stores the data and transfers the data by shifting towards right in the serial path.

Design of Universal Shift Register

The design of a 4-bit universal shift register using multiplexers and flip-flops is shown below.



Universal Shift Register Design

- S0 and S1 are the selected pins that are used to select the mode of operation of this register. It may be shift left operation or shift right operation or parallel mode.
- Pin-0 of first 4×1 Mux is fed to the output pin of the first flip-flop. Observe the connections as shown in the figure.
- Pin-1 of the first 4X1 MUX is connected to serial input for shift right. In this mode, the register shifts the data towards the right.
- Similarly, pin-2 of 4X1 MUX is connected to the serial input for shift-left. In this mode, the universal shift register shifts the data towards the left.
- M1 is the parallel input data given to the pin-3 of the first 4×1 MUX to provide parallel mode operation and stores the data into the register.
- Similarly, remaining individual parallel input data bits are given to the pin-3 of related 4X1 MUX to provide parallel loading.

Universal Shift Register Working

- From the above figure, selected pins the mode of operation of the universal shift register. Serial input shifts the data towards the right and left and stores the data within the register.
- Clear pin and CLK pin are connected to the flip-flop.
- M0, M1, M2, M3 are the parallel inputs while F0, F1, F2, F3 are the parallel outputs of flip-flops
- When the input pin is active HIGH, then the universal shift register loads / retrieve the data in parallel. In this case, the input pin is directly connected to 4×1 MUX
- When the input pin (mode) is active LOW, then the universal shift register shifts the data. In this case, the input pin is connected to 4×1 MUX via NOT gate.
- When the input pin (mode) is connected to GND (Ground), then the universal shift register acts as a Bi-directional shift register.
- To perform the shift-right operation, the input pin is fed to the 1st AND gate of the 1st flip-flop via serial input for shit-right.
- To perform the shift-left operation, the input pin is fed to the 8th AND gate of the last flip-flop via input M.

- If the selected pins S0= 0 and S1 = 0, then this register doesn't operate in any mode. That means it will be in a Locked state or no change state even though the clock pulses are applied.
- If the selected pins S0 = 0 and S1 = 1, then this register transfers or shifts the data to left and stores the data.
- If the selected pins S0 = 1 and S1 = 0, then this register shifts the data to right and hence performs the shift-right operation.
- If the selected pins S0 = 1 and S1 = 1, then this register loads the data in parallel. Hence it performs the parallel loading operation and stores the data.

S0	**S1**	**Mode of Operation**
0	0	Locked state (No change)
0	1	Shift-Left
1	0	Shift-Right
1	1	Parallel Loading

From the above table, we can observe that this register operates in all modes with serial/parallel inputs using 4×1 multiplexers and flip-flops.

Applications

- Used in micro-controllers for I/O expansion
- Used as a serial-to-serial converter
- Used as a parallel-to-parallel data converter
- Used as a serial-to-parallel data converter.
- Used in serial – to – serial data transfer
- Used in parallel data transfer.
- Used in time delay applications
- Used as frequency counters, binary counters, and Digital clocks

4.3 Types of Counter & applications

A counter is a sequential circuit, which counts the number of pulses produced by the clock input.

It can count the pulses and so an n-bit binary counter can count up to n bits. The n-bit counter will have n number of flip flops and has 2^n distinct output states.

Types of counter in Digital circuits

Counters are constructed with a series of flip-flops. Based on the input and the clock pulses given to the flip-flops, there are several types of counter as listed below.

1. Asynchronous counter
2. Synchronous counter
3. Ring counter
4. Johnson counter

Asynchronous counter

It consists of a series of flip flops, in which the output of each flip flop is connected to the clock input of the next higher-order flip flop. The flip flops in the asynchronous counter are triggered individually, that is, they are not synchronized. It is also called a **Ripple counter**.

Synchronous counter

If each flip flop in the counter is triggered at the same time through the clock pulse input, it is said to be synchronous counter.

Modulus of a counter

- Modulus of a counter is the no. of different states through which the counter progress during its operation.
- The modulus of a counter is given as: 2^n where n = number of flip-flops.

Synchronous Counter	**Asynchronous Counter**
1. All the flip-flops are triggered simultaneously with the same clock.	1. Different Flip-flops are triggered with different clocks.
2. Operation is faster	2. Operation is slower
3. Any required sequence can be designed.	3. Will operate only in a fixed count sequence.
4. No decoding error occurs	4. Decoding error occurs due to 'tpd'.
5. Designing is complex as the number of states increases.	5. Designing is easy for more states.
6. E.g. Ring Counter, Johnson Counter	6. E.g. Ripple UP counter, Ripple DOWN counter.
7. Its design necessitates the use of an additional combinational circuit. This circuit becomes more difficult to understand.	7. In comparison to synchronous counters, a circuit is clean.
8. The input clock signal causes all of the flip-flops in the counter to change state at the same time.	8. The flip-flops' clock inputs are not changed by the same clock signal.
9. In synchronous counters, there is no inherent propagation delay.	9. From one flip-flop to the next, there is a subsequent propagation delay.

Applications:

- Counting any digital pulse
- Frequency division
- Digital clocks

4.4 Binary counter, Asynchronous ripple counter (UP & DOWN), Decade counter. Synchronous counter, Ring Counter.

The asynchronous or ripple counter consists of series of flip-flops which are not synchronized by the same clock pulse.

Design steps of asynchronous counter

- Find the number of flip flops using $2^n \geq N$, where N is the number of states and n is the number of flip flops.
- Choose the type of flip flop.
- Draw the truth table for asynchronous counter.
- Use K-map to derive the flip flop reset input functions.
- Draw the logic circuit diagram.

Binary counter/BCD ripple counter

Step 1:

BCD is the 4-bit number and there are 9 valid states in a 4-bit BCD. Hence 4 flip-flops should be used in the design.

The valid states are 0000, 0001, 0010, … 1001.

Step 2:

Let us design the BCD ripple / asynchronous counter with JK flip-flops.

Step 3:

Clock	BCD Counter				Output of Reset Logic Y
	Q_D	Q_C	Q_B	Q_A	
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1

Step 4:

The above truth table for BCD counter is implemented in a Karnaugh map to get the reset logic function.

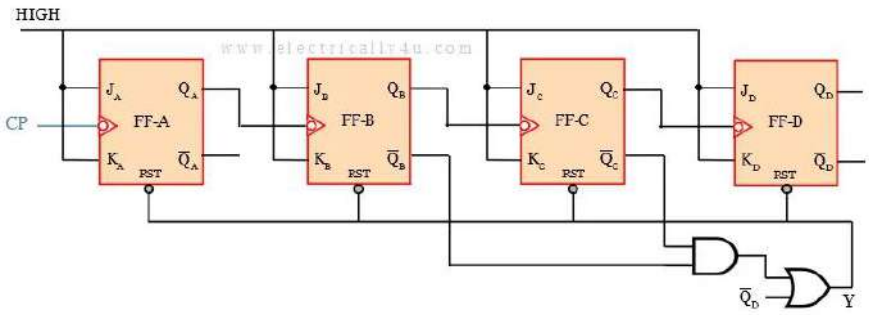
Q_DQ_C \ Q_BQ_A	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

$$Y = \overline{Q}_D + \overline{Q}_C\overline{Q}_B$$

Step 5:

It is drawn with 4 flip-flops. Since it is an asynchronous counter, the clock pulse is given only to the first flip-flop. For the other flip-flops, the output of the previous flip-flop is given as the clock pulse for the next flip-flop.

The expression obtained from the K-map is drawn using combinational circuits. The output Y is given as RESET for all the flip-flops.



Mod-6 asynchronous counter

Step 1:

- Mod-6 counter represents that the counter will have 6 states. Thus, N =6.
- The number of flip-flops used for counter design is determined using the formula, $2^n \geq N$.
- By trial and error method, the value of n is found to be 3. That is the number of flip-flops, n = 3.
- Hence the 6 counter states are 000, 001, 010, 011, 100, 101.

Step 2:

Let us design the mod-6 asynchronous counter with JK flip-flops.

Step 3:

Clock	Q_C	Q_B	Q_A	Output of Reset Logic Y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Step 4:

The above truth table for Mod-6 counter is implemented in a Karnaugh map to get the reset logic function.

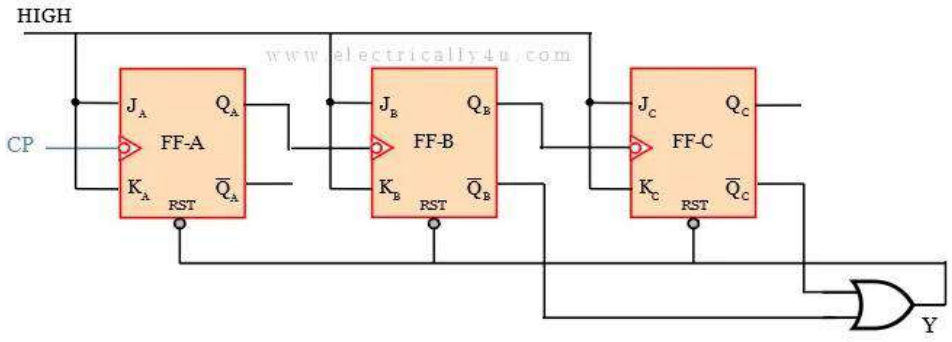
Q_C \ Q_BQ_A	00	01	11	10
0	1	1	1	1
1	1	1	0	0

$$Y = \overline{Q}_C + \overline{Q}_B$$

Step 5:

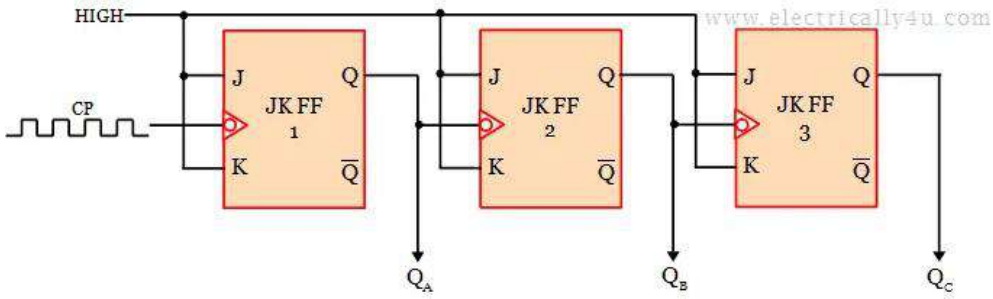
It is drawn with 3 flip-flops. Since it is an asynchronous counter, the clock pulse is given only to the first flip-flop. For the other flip-flops, the output of the previous flip-flop is given as the clock pulse for the next flip-flop.

The expression obtained from the K-map is drawn using combinational circuits. The output Y is given as RESET for all the flip-flops.



3-bit asynchronous up counter

For 3-bit asynchronous or ripple up counter, 3 T-flip-flops are used.This counter consists of 2^3 = 8 count states(000, 001, 010, 011, 100, 101, 110, 111). The counter counts the incoming pulses starting from 0 to 7.

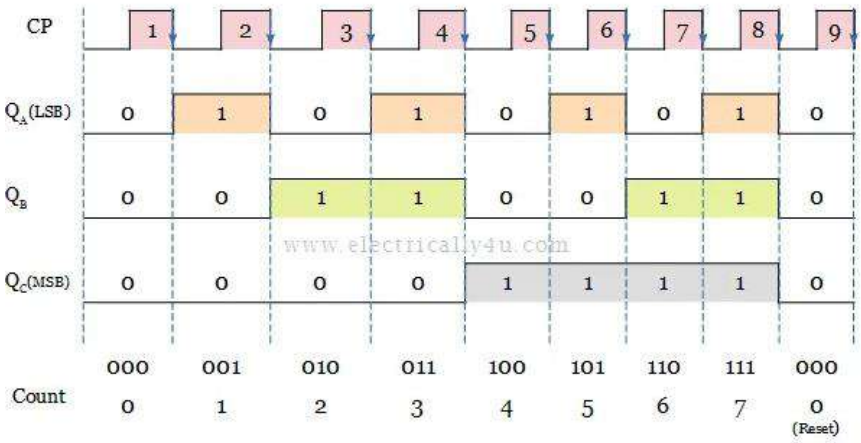


The above circuit shows the circuit diagram of a 3-bit asynchronous up counter, in which the clock pulse is given as clock input for JK FF1. For the other flip-flops, the clock input is fed from the output of previous flip-flops.

The clock pulse count is noted at the output of each flip-flop($Q_CQ_BQ_A$), where Q_A is the LSB and Q_C is the MSB.

At the falling edge of each clock pulse, the output of JF FF1 toggles. For each logic HIGH output(Q_A = 1) of JK FF1, at its falling edge, JK FF2 will toggle the output(Q_B). Similarly, for each logic HIGH output(Q_B = 1) of JK FF2, JK FF3 will toggle the output(Q_C).

The below figure shows the timing diagram of the 3-bit ripple counter, which shows the change of state of each flip-flop during each clock pulse. In this type, the counter resets to 000, after counting up to 111.

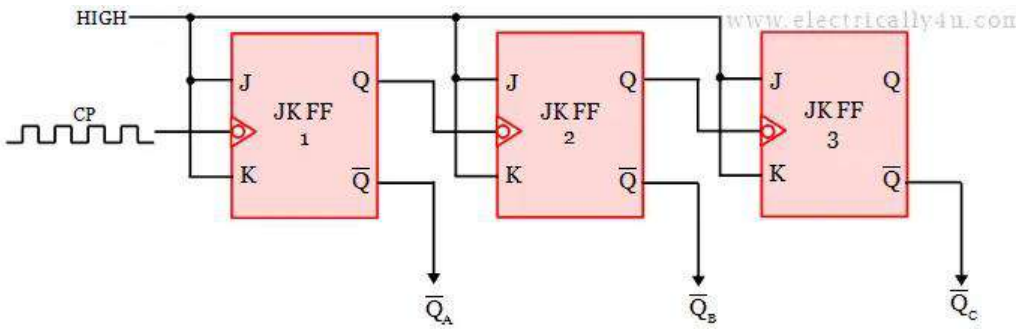


3-bit asynchronous down counter

The down counter will count the clock pulses from maximum value to zero. In other words, for each clock pulse, the count value is decremented.

The below diagram shows the 3-bit asynchronous down counter. Since it is a 3-bit counter, 3 negative edge-triggered flip-flops are used. The clock pulse input is given only to the first flip-flop. The clock input of the remaining flip-flops is triggered by the Q output of the previous flip-flop.

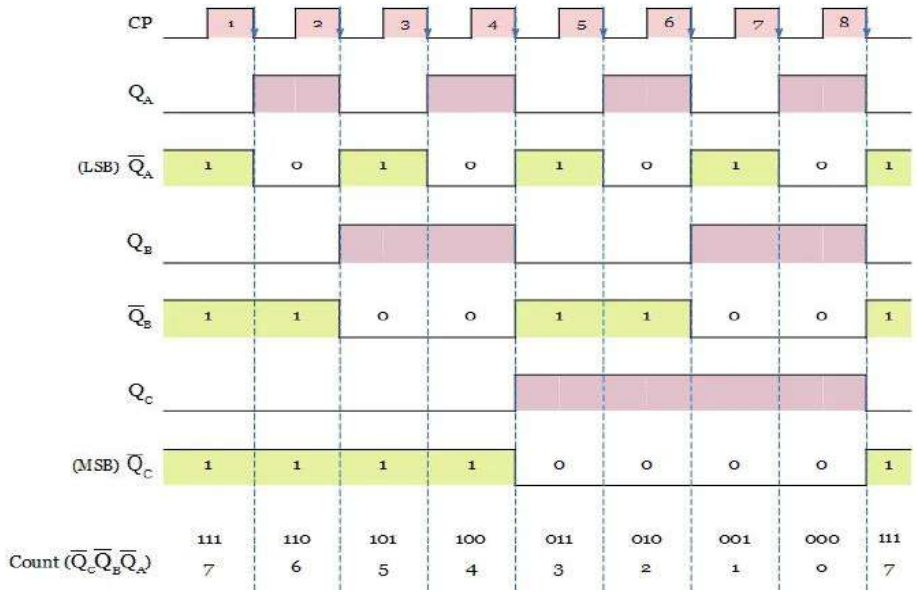
Since it is down counter, the 3-bit count value is measured from the (Q_C'Q_B'Q_A'), where Q_C' is the MSB and Q_A' is the LSB.



The operation is the same as that of the 3-bit asynchronous up counter. If the output is taken at the normal Q output of each flip flop, then it is an up counter. If the output is taken at the complemented output (Q') of each flip flop, it is said to be the down counter.

The change of state of each flip flop with respect to the clock pulse and the count value is shown in the below timing diagram.

In this diagram, the output waveform Q_C, Q_B, Q_A represents the normal Q output of JK FF3, JK FF2 and JK FF1 respectively. In the below waveform, Q_C', Q_B', Q_A' represents the complemented output from of JK FF3, JK FF2 and JK FF1 respectively.



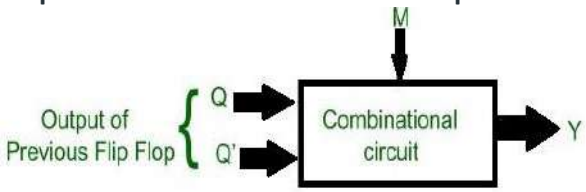
The count value can be observed from the complemented outputs. The counter value starts from 111 and decrements its value for each clock pulse. After reaching 000, the counter resets to the maximum value(111) and starts to decrement again for the next clock pulse.

Asynchronous ripple counter (UP & DOWN)

Up/Down counter is the combination of both the counters in which we can perform up or down counting by changing the Mode control input.
In this a mode control input (say M) is used for selecting up and down mode.

A combinational circuit is required between each pair of flip-flop to decide whether to do up or do down counting.

For n = 3, i.e for 3 bit counter –
Maximum count = $\mathbf{2^n - 1}$ and number of states are $\mathbf{2^n}$.
Steps involve in design are :
Step 1 : **Decision for Mode control input –**

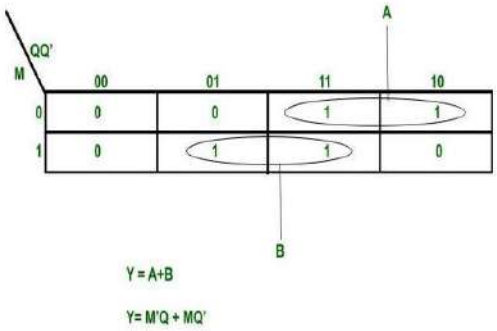


When M = 0, then Y= Q, therefore it will perform Up counting (As discussed above).
When M = 1, then Y= Q' therefore it will perform Down counting (As discussed above).
Combinational circuit is required for deciding mode control(i.e whether counter will perform Up counting or Down counting).
So the all possible combinations are –

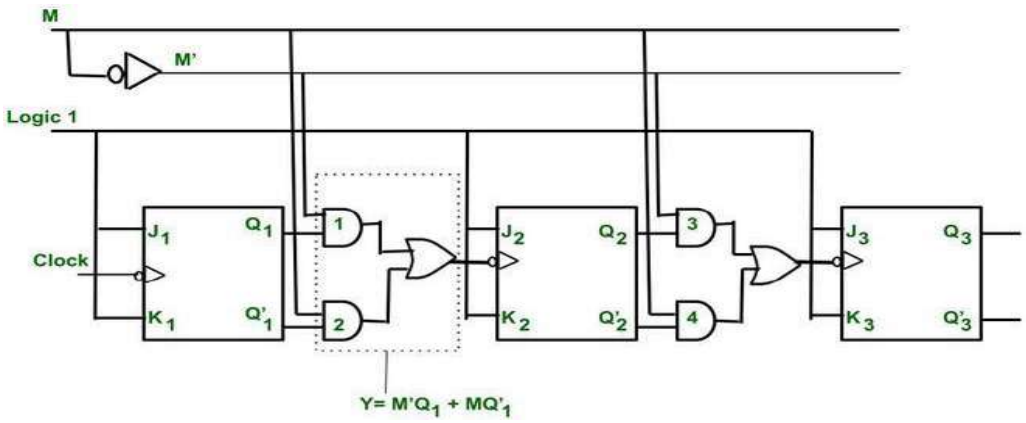
Y = Q when M= 0
Y = Q' when M=1

M	Q	Q'	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

K-map for finding output Y that will be given as clock to next FF.

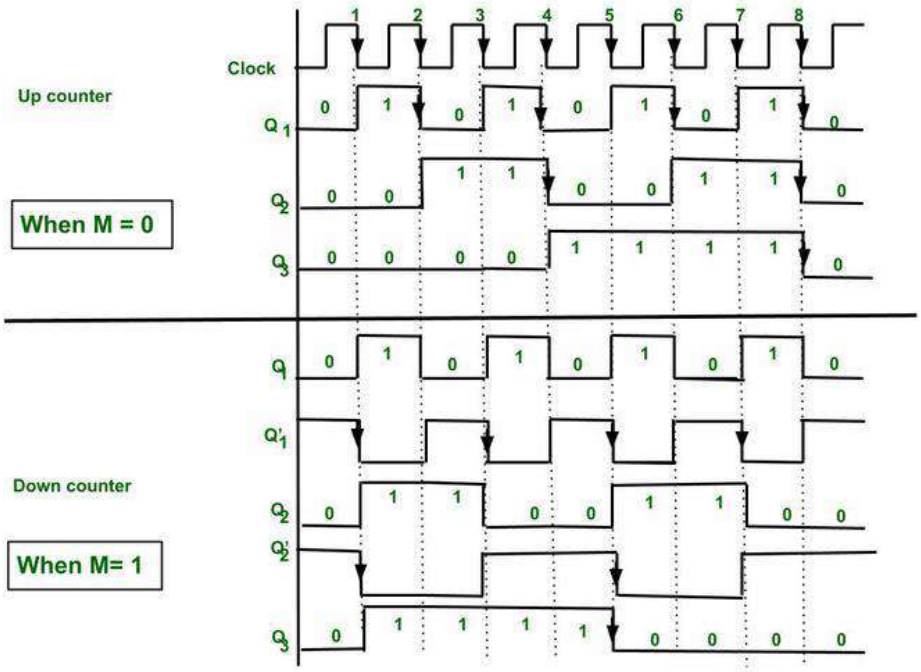


Step 2 : **Insertion of Combinational logic between every pair of FFs –**



Timing diagram :

Initially $Q_3 = 0$, $Q_2 = 0$, $Q_1 = 0$.



Case 1 – When M=0, then M' =1.
Put this in **Y= M'Q + MQ'= Q** So Q is acting as clock for next FFs.
Therefore, the counter will act as Up counter.

Explanation of Up counter –

- The 1st FF is connected to logic 1. Therefore, it will toggle for every falling edge.
- The 2nd FF input is connected to Q_1.Therefore it changes its state when Q_1= 1 and there is falling edge of clock.
- Similarly, 3rd FF is connected to Q_2. Therefore, it changes its state when Q_2= 1 and there is falling edge of clock.
- By this we can generate counting states of Up counter.
- After every 8th falling edge the counter is again reaching to state 0 0 0. Therefore, it is also known as divide by 8 circuit or mod 8 counter.

Case 2 – When M=1, then M' =0.
Put this in **Y= M'Q + MQ'= Q'**. So Q' is acting as clock for next FFs.
Therefore, the counter will act as Down counter.

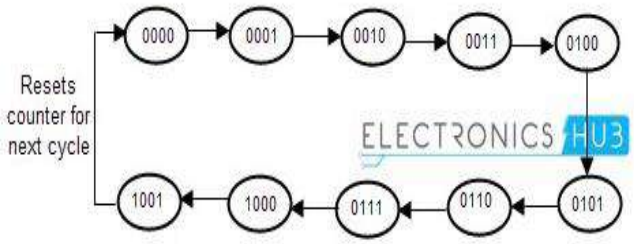
Explanation of Down counter –

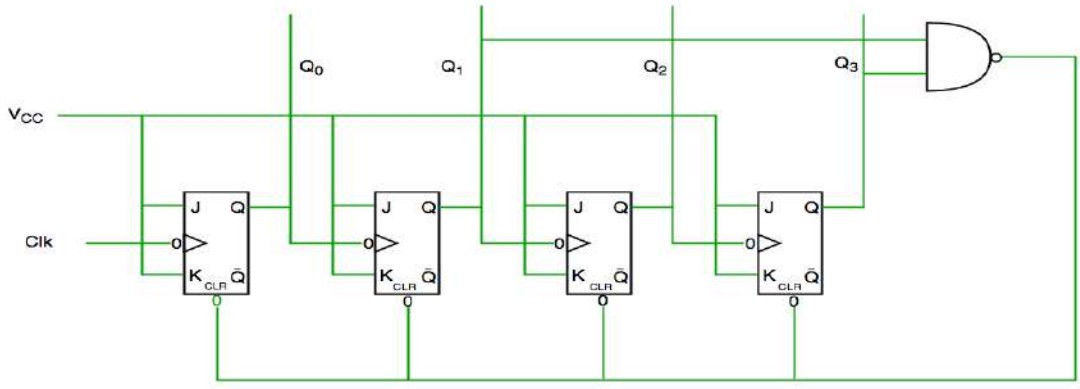
- The 1st FF is connected to logic 1. Therefore, it will toggle for every falling edge.
- The 2nd FF input is connected to Q'_1.Therefore it changes its state when Q'_1= 1 and there is falling edge of clock.
- Similarly, 3rd FF is connected to Q'_2. Therefore, it changes its state when Q'_2= 1 and there is falling edge of clock.
- By this we can generate counting states of down counter.
- After every 8th falling edge the counter is again reaching to state 0 0 0. Therefore, it is also known as divide by 8 circuit or mod 8 counter.

Decade counter

A decade counter also termed as BCD (Binary Coded Decimal) counter is a series type of digital counter which is designed to count ten digits. It performs the operation of resetting automatically when there is a new clock input signal. As the counter counts ten unique combinations of the applied input, it is termed a decade counter. The values that a BCD counter counts are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in binary format.

State Diagram of Decade Counter





A Decade counter which is designed with JK flip flop. The outputs from J and K terminals are connected to logic '1'. The clock signal's input in each flip flop is connected to the subsequent flip flop excepting the last flip flop. The NAND gate output is connected parallelly to the CLR signal for all the flip-flops.

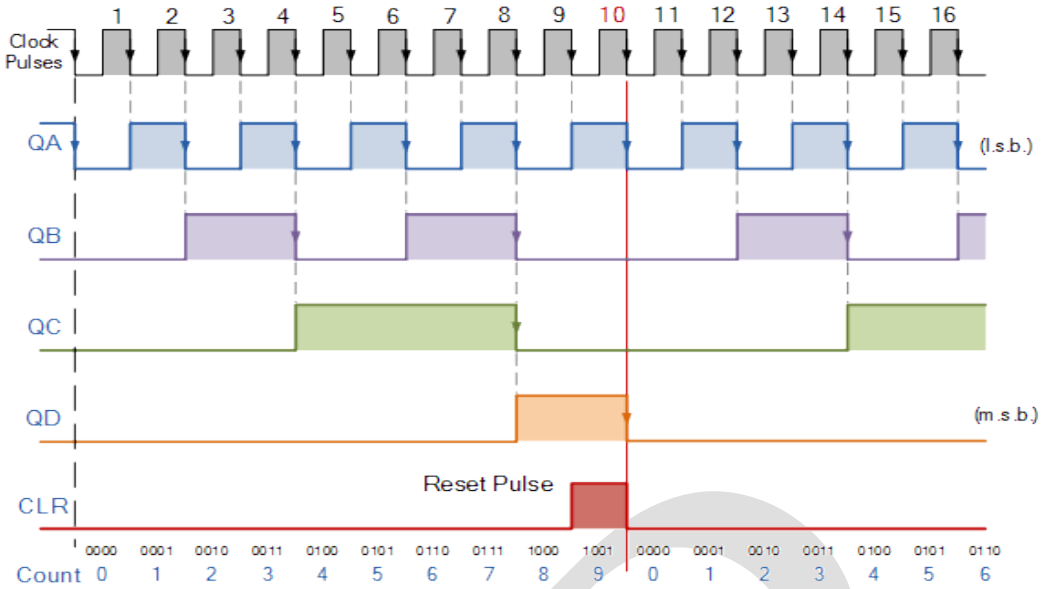
Operation

In the decade counter, when the device is at REST mode, the count equals '0' which is 0000 in binary and this is the initial phase of the counter cycle. When a clock signal is connected as an input to the circuit, then the circuit starts to count the binary digits in sequence. The initial clock pulse can be able to count to 9 digits means up to 1001. Then with the arrival of the next clock pulse, the count increases to 10 which is 1010. Output of the NAND gate is at a low position when inputs are high. This output is then connected to the CLR signal as input so that it resets the entire flip-flop stages in the BCD counter. This corresponds that the pulse count from 1001 will reset and tend to initiate from '0000'.

Decade Counter Truth Table

Clock Count	Output bit Pattern				Decimal Value
	QD	QC	QB	QA	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	Counter Resets its Outputs back to Zero				

Decade Counter Timing Diagram



Synchronous counter

The synchronous counters count the number of clock pulses received at its input. The synchronous counter uses the same clock signal from the same source and at exactly the same time. Generally, it is constructed using either JK flip flop or T flip flop. Synchronous counters use edge-triggered flip-flops. These flip-flops change the state during the next clock pulse.

Design steps of synchronous counter

1. Find the number of flip flops using $2n \geq N$, where N is the number of states and n is the number of flip flops.
2. Choose the type of flip flop.
3. Draw the state diagram of the counter.
4. Draw the excitation table of the selected flip flop and determine the excitation table for the counter.
5. Use K-map to derive the flip flop input functions.

3-bit synchronous up counter

Step 1:

A flip flop stores only one bit, hence for a 3 bit counter, 3 flip flops(n=3) are needed to design the counter.

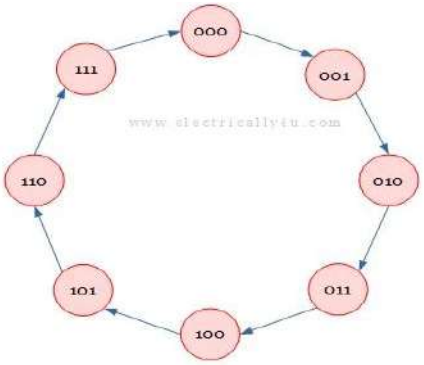
Number of states = $2^n = 2^3$ = 8 states (000, 001, 010, 011, 100, 101, 110, 111)

Step 2:

Since the type of flip flop is given in the problem, let us use JK flip flops.

Step 3:

The state diagram for the counter is drawn as below.



Step 4:

We know, the excitation table for JK flip flop is given by,

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

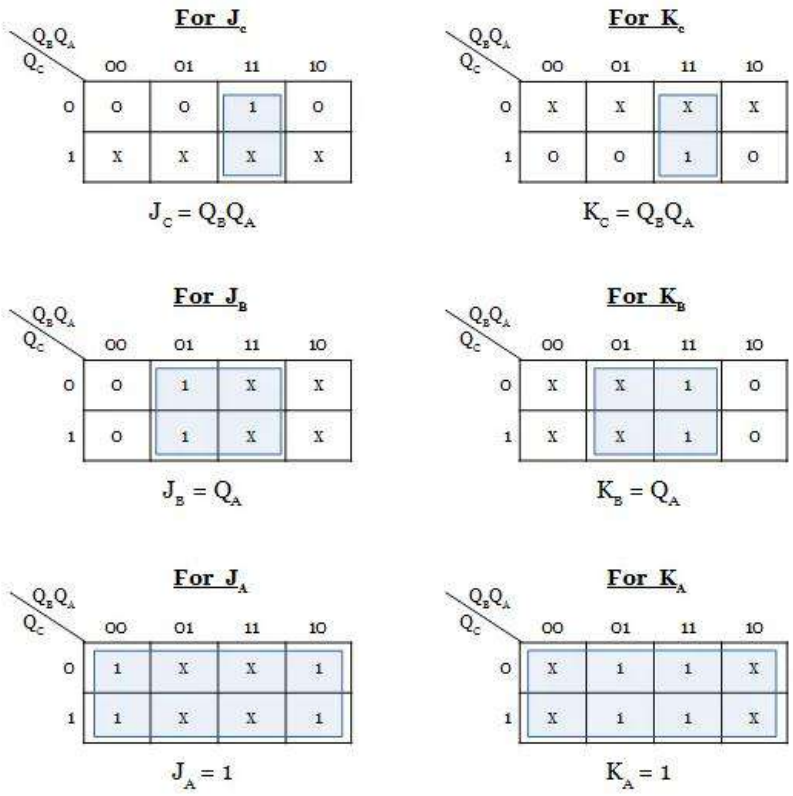
Now the excitation table for the 3-bit synchronous counter is determined from the excitation table of JK flip flop.

The excitation table is framed for 8 states of the counter. Since 3 flip-flops are used in the design, the present state, next state and flip flop inputs for each flip flop are considered.

Clock	Present State			Next State			Flip flop Inputs					
	Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
1	0	0	0	0	0	1	0	X	0	X	1	X
2	0	0	1	0	1	0	0	X	1	X	X	1
3	0	1	0	0	1	1	0	X	X	0	1	X
4	0	1	1	1	0	0	1	X	X	1	X	1
5	1	0	0	1	0	1	X	0	0	X	1	X
6	1	0	1	1	1	0	X	0	1	X	X	1
7	1	1	0	1	1	1	X	0	X	0	1	X
8	1	1	1	0	0	0	X	1	X	1	X	1

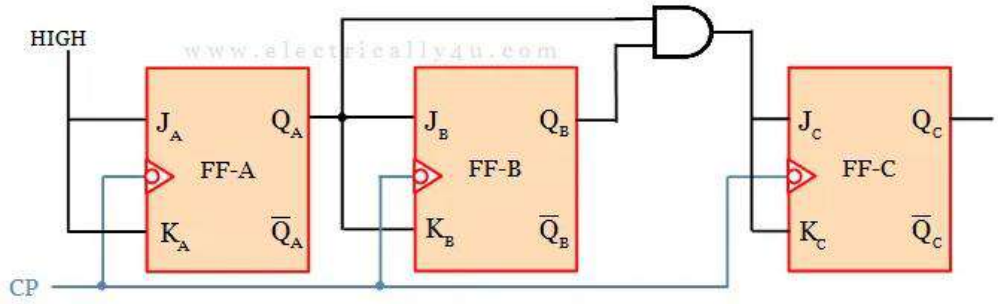
Step 5:

Using Karnaugh maps, the input functions for the 3 flip flops are derived. The present states are the input for all the flip-flops. Since there are three inputs(Q_C, Q_B, Q_A), 8 cell K-map is used.



Step 6:

The logic diagram of the 3-bit synchronous counter is drawn as follows. Draw the 3 JK flip-flops. The common clock pulse input is given to all the flip-flops. The inputs for each flip-flop are drawn as per the logic functions derived in the previous step.



Mod-6 synchronous up Counter.

Step 1:

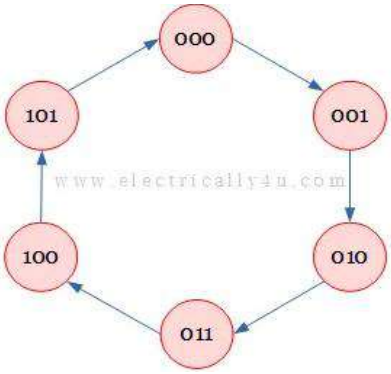
- Mod-6 counter represents that the counter will have 6 states. Thus, N =6.
- The number of flip-flops used for counter design is determined using the formula, $2n \geq N$.
- By trial and error method, the value of n is found to be 3. That is the number of flip-flops, n = 3.
- Hence the 6 counter states are 000, 001, 010, 011, 100, 101.

Step 2:

Let us choose the JK- flip flop to design the Mod-6 synchronous up counter.

Step 3:

The state diagram for the mod-6 counter with 6 states is drawn as below.



Step 4:

The excitation table for JK flip flop is given by,

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

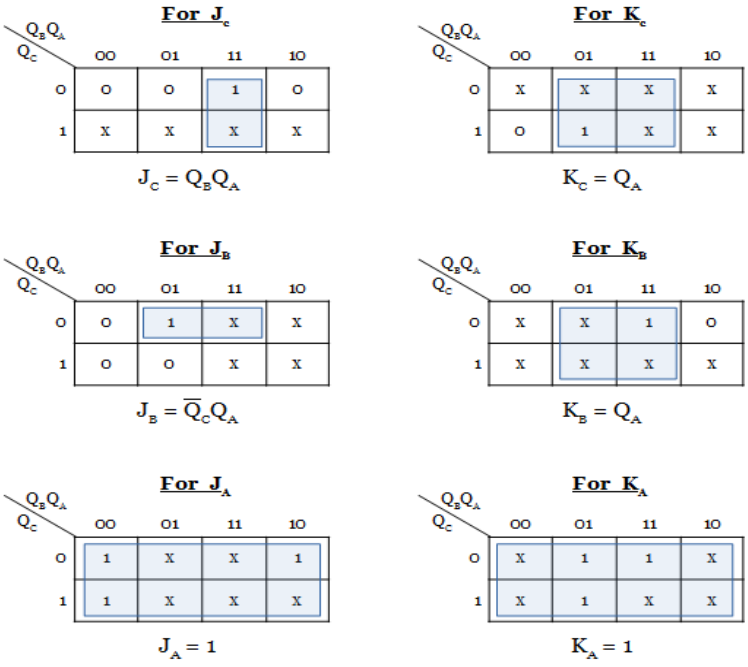
Now the excitation table for the mod-6 synchronous counter is determined from the excitation table of JK flip flop.

The excitation table is framed for 6 states of the counter. Since 3 flip-flops are used in the design, the present state, next state and flip flop inputs for each flip flop are considered.

Clock	Present State			Next State			Flip flop Inputs					
	Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
1	0	0	0	0	0	1	0	X	0	X	1	X
2	0	0	1	0	1	0	0	X	1	X	X	1
3	0	1	0	0	1	1	0	X	X	0	1	X
4	0	1	1	1	0	0	1	X	X	1	X	1
5	1	0	0	1	0	1	X	0	0	X	1	X
6	1	0	1	0	0	0	X	1	0	X	X	1
-	1	1	0	X	X	X	X	X	X	X	X	X
-	1	1	1	X	X	X	X	X	X	X	X	X

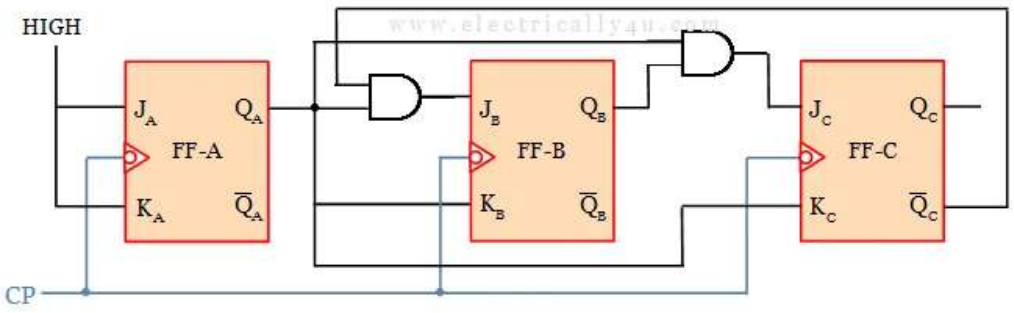
Step 5:

Using K-maps, the input functions for the 3 flip flops are derived. The present states are the input for all the flip-flops. Since there are three inputs(Q_C, Q_B, Q_A), 8 cell K-map is used.



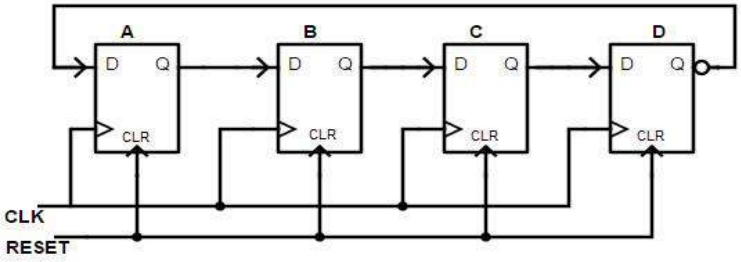
Step 6:

The logic diagram of the mod-6 synchronous up counter is drawn as follows. Draw the 3 JK flip-flops. The inputs for each flip-flop are drawn as per the logic functions derived in the previous step. The clock pulse is given to all the flip-flops.



Ring Counter

- The ring counter is a cascaded connection of flip flops, in which the output of last flip flop is connected to input of first flip flop. In ring counter if the output of any stage is 1, then its reminder is 0. The Ring counters transfers the same output throughout the circuit.
- That means if the output of the first flip flop is 1, then this is transferred to its next stage i.e. 2nd flip flop. By transferring the output to its next stage, the output of first flip flop becomes 0. And this process continues for all the stages of a ring counter. If we use n flip flops in the ring counter, the '1' is circulated for every n clock cycles.

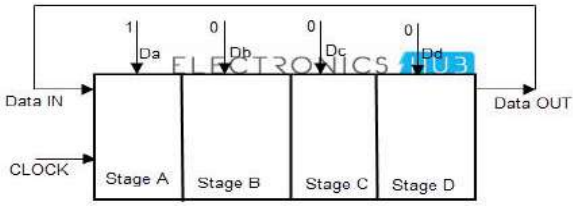


Here we design the ring counter by using D flip flop. This is a Mod 4 ring counter which has 4 D flip flops connected in series. The clock signal is applied to clock input of each flip flop, simultaneously and the RESET pulse is applied to the CLR inputs of all the flip flops.

Operation of Ring Counter

Initially, all the flip flops in ring counter are reset to 0 by applying CLEAR signal. Before applying the clock pulse, we apply the PRESET pulse to the flip flops which assigns the value '1' to the ring counter circuit. For each clock signal, the data circulates among all the 4 flip flop stages of ring counter.

Circulation of data in Ring counters



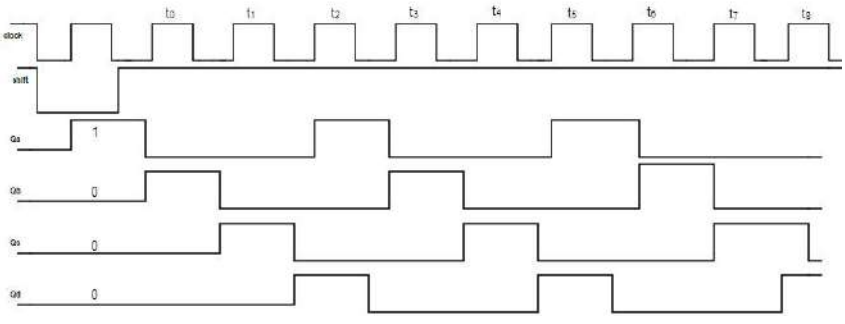
- We know that the ring counter is similar to that of the shift registers connected in series. The above diagram shown the four stages of flip flops as the parallel in serial our shift registers, with data inputs D0, D1, D2 and D3.
- The data circulation in ring counter is explained below. By passing the reset signal, initially the flip flops are at RESET state. When the PRESET is applied to the ring counter the input of the circuit becomes 1.
- This input is connected to the first flip flop in the series, so that the flip flop QA is set to 1 and all other outputs of remaining flip flops will be low.
- If we make the data input of the flip flop 'A' to low, this gives us the data pulse as 0 1 0. Then for the second clock signal, the output of first flip flop will again change and then the output of 'B' will become high. This means the data pulse 0 0 1 occurs.
- In this way, as the clock signal and input of first flip flop changes, the output of the other flip flops changes. As the output of last flip flop in series is connected to the input of the first flip flop, the data sequence rotates or circulates in the ring counter.

Truth table of ring counter

Q_0	Q_1	Q_2	Q_3
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

When CLEAR input CLR = 0, then all flip flops are set to 1. When CLEAR input CLR = 1, the ring counter starts its operation. For one clock signal, the counter starts its operation. On next clock signal, the counter again resets to 0000. Ring counter has 4 sequences: 0001, 0010, 0100, 1000, 000.

Timing diagram of Ring Counter



**All 4 bit possible Asynchronous /Synchronous counter: Do it yourself

4.5 Concept of memories-RAM, ROM, static RAM, dynamic RAM, PSRAM

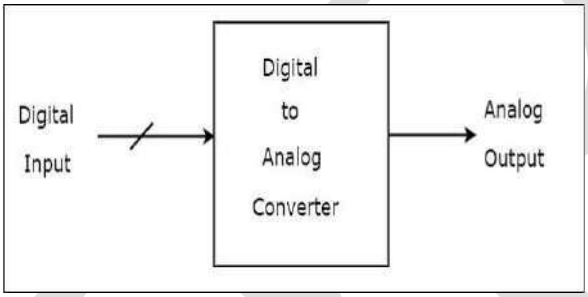
UNIT-5: A/D AND D/A CONVERTERS

5.1 Necessity of A/D and D/A converters.

- To convert the data from one form into another form
- Electrically sophisticated and high-speed processing are performed digitally in CPUs and DSPs.

******To convert one form of signal with other

A Digital to Analog Converter (DAC) converts a digital input signal into an analog output signal. The digital signal is represented with a binary code, which is a combination of bits 0 and 1.

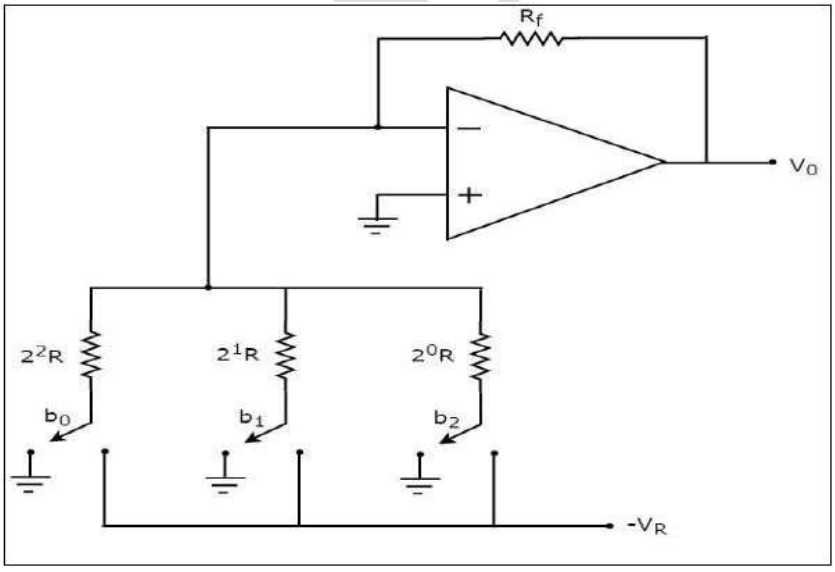


There are two types of DACs

- Weighted Resistor DAC
- R-2R Ladder DAC

5.2 D/A conversion using weighted resistors methods.

A weighted resistor DAC produces an analog output, which is almost equal to the digital (binary) input by using binary weighted resistors in the inverting adder circuit.



- It uses binary weighted resistor 2^1R, 2^2R……2^nR and summing amplifier.

- The MSB input is connected with lowest resistor and towards LSB the resistance value is made twice of previous resistor.
- The purpose of increasing the resistor value is to pass minimum current through the LSB resistance while maximum current through MSB resistance.
- Here n number of electronic switches are used to provide digital input.
  When switch is connected to ($-V_R$), b1=1
  When switch is connected to ground b_1 =0

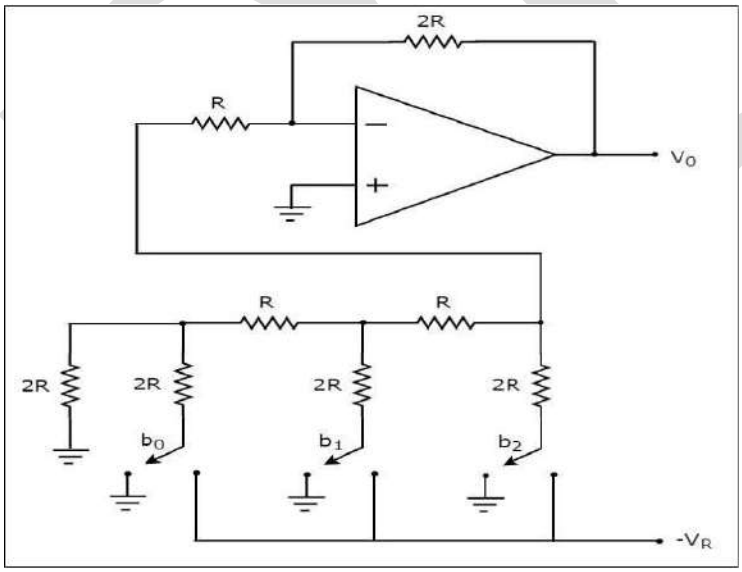
- Output voltage= $V_R\left(\frac{b1}{2}+\frac{b2}{22}+\ldots\ldots\ldots\frac{bN}{2N}\right)$
- Example:

The disadvantages of a binary weighted resistor DAC are as follows

- The difference between the resistance values corresponding to LSB & MSB will increase as the number of bits present in the digital input increases.
- It is difficult to design more accurate resistors as the number of bits present in the digital input increases.

5.3 D/A conversion using R-2R ladder (Weighted resistors) network.

- R-2R ladder D/A converter produces an analog output, which is almost equal to the digital input by using R-2R ladder network in the inverting adder circuit.

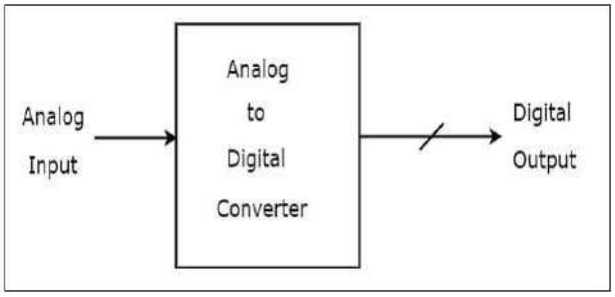


- In R-2R ladder D/A converter, resistors of only two values i.e R and 2R are used.
- The inverting input terminal of the op-amp acts as summing junction for the ladder inputs.
- Output voltage= $V_R\left(\frac{bN}{2N}+\ldots\ldots+\frac{b2}{22}+\frac{b1}{2}\right)$
- Example

The advantages of a R-2R Ladder DAC are as follows −

- R-2R Ladder DAC contains only two values of resistor: R and 2R. So, it is easy to select and design more accurate resistors.
- If more number of bits are present in the digital input, then we have to include required number of R-2R sections additionally.

********An Analog to Digital Converter **(ADC)** converts an analog signal into a digital signal. The digital signal is represented with a binary code, which is a combination of bits 0 and 1.



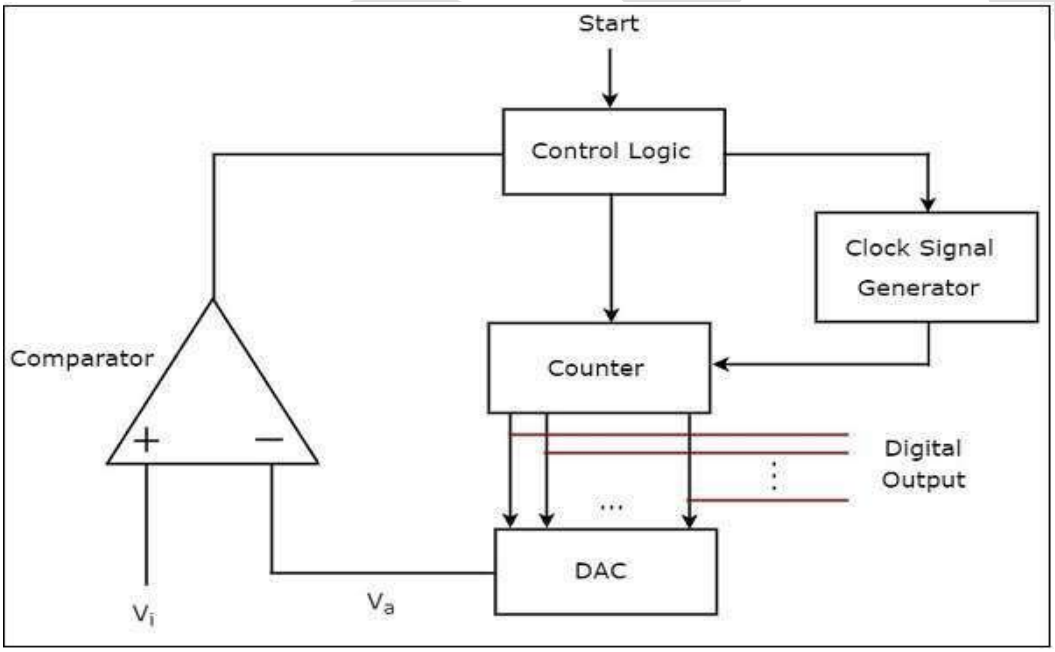
There are 3 types of ADCs

- Counter type ADC
- Successive Approximation ADC
- Flash type ADC

5.4 A/D conversion using counter method.

A counter type ADC produces a digital output, which is approximately equal to the analog input by using counter operation internally.

The block diagram of a counter type ADC is shown in the following figure –



The counter type ADC mainly consists of 5 blocks: Clock signal generator, Counter, DAC, Comparator and Control logic.

The working of a counter type ADC is as follows –

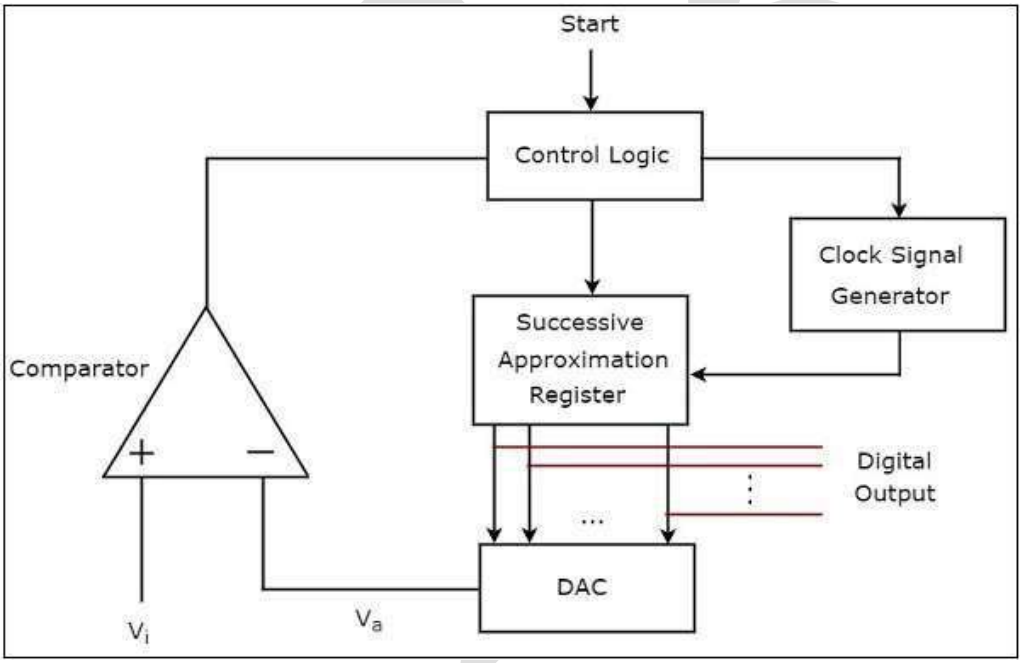
- The control logic resets the counter and enables the clock signal generator in order to send the clock pulses to the counter, when it received the start commanding signal.
- The counter gets incremented by one for every clock pulse and its value will be in binary (digital) format. This output of the counter is applied as an input of DAC.

- DAC converts the received binary (digital) input, which is the output of counter, into an analog output. Comparator compares this analog value, with the external analog input value.
- The output of comparator will be '1' as long as Vi is greater than. The operations mentioned in above two steps will be continued as long as the control logic receives '1' from the output of comparator.
- The output of comparator will be '0' when Va is less than or equal to Vi. So, the control logic receives '0' from the output of comparator. Then, the control logic disables the clock signal generator so that it doesn't send any clock pulse to the counter.
- At this instant, the output of the counter will be displayed as the digital output. It is almost equivalent to the corresponding external analog input value.

5.5 A/D conversion using Successive approximate method

A successive approximation type ADC produces a digital output, which is approximately equal to the analog input by using successive approximation technique internally.

The block diagram of a successive approximation ADC is shown in the following figure



The successive approximation ADC mainly consists of 5 blocks− Clock signal generator, Successive Approximation Register (SAR), DAC, comparator and Control logic.

The working of a successive approximation ADC is as follows −

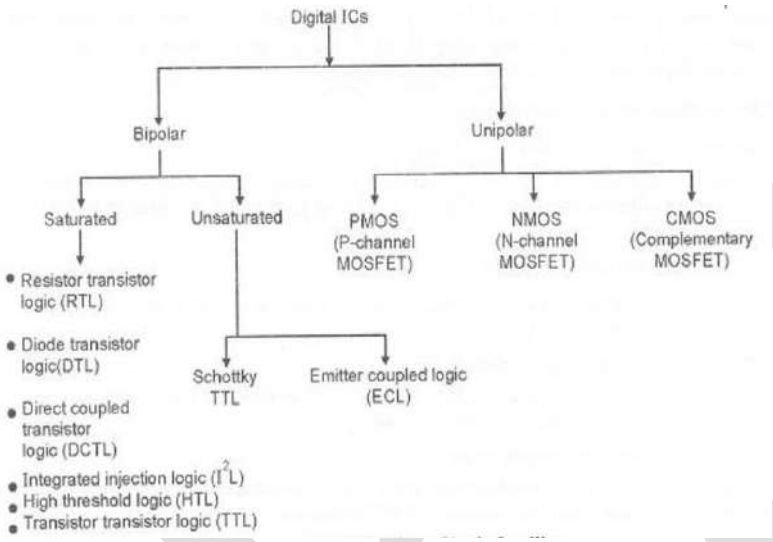
- The control logic resets all the bits of SAR and enables the clock signal generator in order to send the clock pulses to SAR, when it received the start commanding signal.
- The binary (digital) data present in SAR will be updated for every clock pulse based on the output of comparator. The output of SAR is applied as an input of DAC.
- DAC converts the received digital input, which is the output of SAR, into an analog output. The comparator compares this analog value Va with the external analog input value Vi.
- The output of a comparator will be '1' as long as Vi is greater than Va. Similarly, the output of comparator will be '0', when Vi is less than or equal to Va .

- The operations mentioned in above steps will be continued until the digital output is a valid one.

UNIT-6: LOGIC FAMILIES

6.1 Various logic families &categories according to the IC fabrication process

- A logic family is a collection of different integrated circuit chips that have similar input, output and internal circuit characteristics but that perform different logic functions.
- The circuit design of the basic gate of each logic family is the same.
- The most important parameters for evaluating and comparing logical families include: logic levels, power dissipation, propagation delay, noise margin, Fan out



6.2 Characteristics of Digital ICs- Propagation Delay, fan-out, fan-in, Power Dissipation ,Noise Margin ,Power Supply requirement &Speed with Reference to logic families.

Propagation Delay

The time required to change the output from one logic state to another logic state after input is applied, is called the propagation delay of logic circuit. Its unit is nanoseconds.

fan-out

It is the no of the same gates that can be driven by a gate. High fan out is beneficial, as this reduces the requirement for additional drivers to drive more gates.

fan-in

The fan-in of a logic gate is defined as the number of inputs (coming from similar circuits) that it can handle properly.

Power Dissipation

This is the amount of power dissipated in an IC. It is determined by the current, Icc. that it draws from the Vcc supply and equals Vcc Icc. This power is specified in mW. Lower power dissipation is desirable feature for any IC.

Noise Margin

Noise margin is *the amount of noise that a CMOS circuit could withstand without compromising the operation of circuit*.

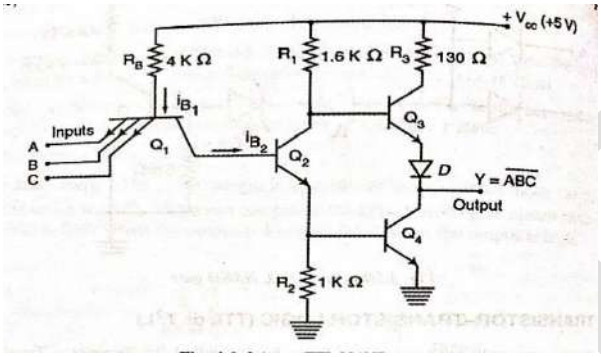
Power Supply requirement

Every IC requires a certain amount of electrical power to operate. The power is supplied by one or more power-supply voltage connected to the power pin (or pins) on the chip. Usually there is only one power-supply terminal on the chip and it is marked Vcc for TTL or VDD for for MOS devices. Obviously low power consumption is desirable features in any digital ICs.

6.3 Features, circuit operation &various applications of TTL(NAND), CMOS (NAND & NOR)

Transistor Transistor Logic (TTL)

The basic circuit for TTL logic family is the NAND gate. The TTL circuit uses a special single multi- emitter transistor that is fabricated with several emitters at its input. The number of emitters used depends on the desired fan-in of the circuit.



Circuit operation:

- When any one of the input is logic 0, the emitter junctions of transistor T1 are forward biased. The required voltage to conduct T2 and T3 is greater than the voltage available at the base of T1 and hence T2 and T3 are in cut-off. The output voltage is equal to the supply voltage Vcc, output is in logic 1 state.
- When all the inputs are in logic 1 state, the emitter junction of T1 are reverse biased and the current supply by the source is sufficient to operate T2 and T3 in saturation and the output is in logic 0 state.

Application:

- Switching device in driving lamps and relays
- Processors of mini computers
- Used in printers and video display terminals

CMOS NAND:

1.18.2 CMOS NAND Gate

Figure 1.45 shows a two-inputs CMOS logic NAND gate. It consists of two p-channel and two n-channel MOSFETs. p-channel MOSFETs are connected in parallel and n-channel MOSFETs are connected in series. Here, Q_1 and Q_2 are p-channel MOSFETs and Q_3 and Q_4 are n-channel MOSFETs.

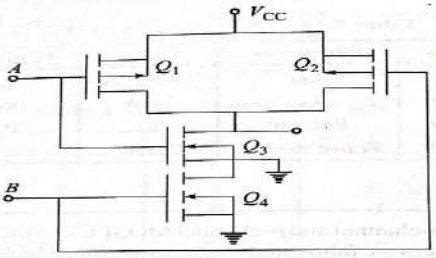


Fig. 1.45 CMOS NAND gate

Operation

- When the inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF, and the output is high (V_{DD}).
- When any one of the inputs is low (0 V or –ve), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON, and the output is high.
- When the inputs are high (+ve voltage), Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON, and the output is low.

The operation of the circuit is summarized in Table 1.16.

Table 1.16 Operations of CMOS NAND gate

A	B	Q_1	Q_2	Q_3	Q_4	V_O
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

CMOS NOR

1.18.3 CMOS NOR Gate

Figure 1.46 shows a two-inputs CMOS logic NOR gate. It consists of two p-channel and two n-channel MOSFETs. n-channel MOSFETs are connected in parallel and p-channel MOSFETs are connected in series. Here, Q_1 and Q_2 are p-channel MOSFETs and Q_3 and Q_4 are n-channel MOSFETs. When the input is low, p-channel MOSFETs are ON and n-channel MOSFETs are OFF. When the input is high, p-channel is OFF and n-channel is ON.

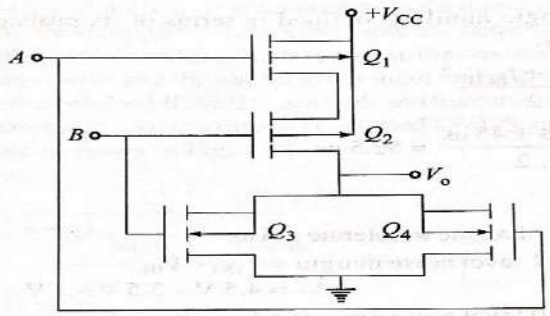


Fig. 1.46 CMOS NOR gate

Operation

- When inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF, and the output is high (V_{DD}).
- When any one of the inputs is low (0 V or –ve), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON, and the output is low.
- When inputs are high (+ve voltage), Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON, and the output is low.

Table 1.17 Operations of CMOS NOR gate

A	B	Q_1	Q_2	Q_3	Q_4	V_O
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0