2 Mark Questions

1. State the base of a Number System

   The base or radix of a number system is the number of symbols or digits present in that number system. For example in Binary number system base is 2 .That is the number of symbols used in Binary number system is 2, 0 and 1

2. One's complement of 1011 is

   0100

3. List two universal gates

   NAND and NOR

4. List two basic gates

   AND ,OR and NOT

5. Write two examples for non-weighted codes

   Gray code and Excess-3 code

6. Write two examples for weighted codes

   BCD and 8421 code

7. Convert binary 11010 to decimal

   $0\times2^0+1\times2^1+0\times2^2+1\times2^3+1\times2^4$

   $=0 + 2 + 0 +8 + 16$

   $=26$

   $(11010)_2 = (26)_{10}$

8. State the number of symbols or digits used in a

Number System	Base	Symbols or Digits
Decimal Number system	10	0 to 9
Binary Number	2	0, 1

system

Octal Number system	8	0 to 7
Hexadecimal Number system	16	0 to 9, Letters used: A- F

9. Full form of ASCII

   American Standard Code for Information Interchange

10. Define binary codes

    The digital data is represented, stored and transmitted as group of binary bits. This group is also called as binary code. The binary code is represented by the number as well as alphanumeric letter.

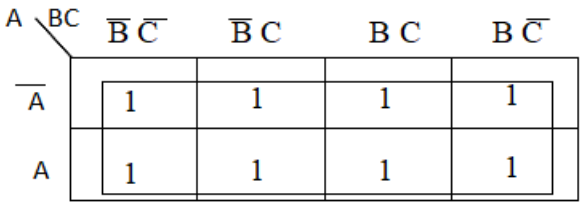
11. Give any two alphanumeric codes

    ASCII and EBCDIC

12. Define Minterm and Maxterm

    Minterms are AND terms with every variable present in either normal or complemented form. Maxterms are OR terms with every variable in normal or complemented form.

13. Reduce the expression f(A,B,C)=∑m(0,1,2,3,4,5,6,7)



=1

14. Name the flipflop used for data storage

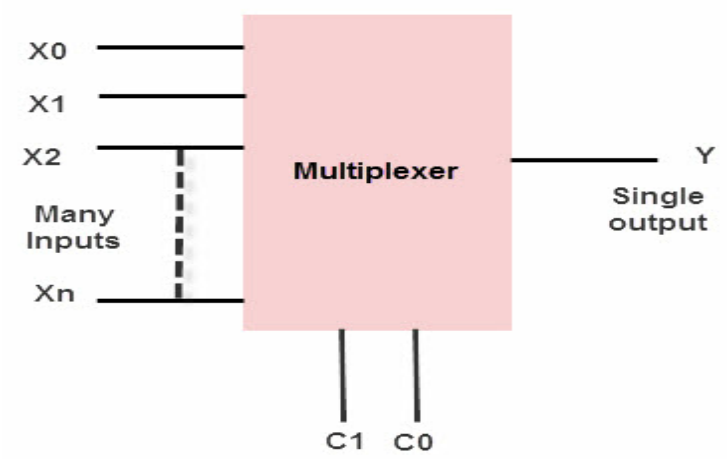
    D flipflop

15. Name an error detecting code

    Hamming code.

16. Define a 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.**



17. List two types of sequential circuits based on timing of signals

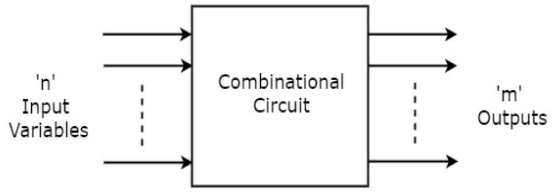
    - Synchronous sequential circuits
    - Asynchronous sequential circuits

18. A group of 4 bits is called a …….. and group of 8 bit is called a ……..

    Nibble , Byte

19. Describe combinational circuits

    The output of combinational circuit at any instant of time, depends only on the present input terminals. The combinational circuit do not use any memory.



This combinational circuit has 'n' input variables and 'm' outputs. Each combination of input variables will affect the output(s).

20. Name the flip-flop used to construct a Ripple counter

    J K flip flop Or T FF

21. Name an error detecting and correcting codes

    Hamming codes

22. What is the function of an encoder

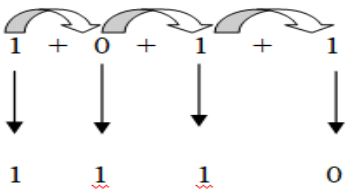
    An **Encoder** is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits.

23. Define a flip-flop

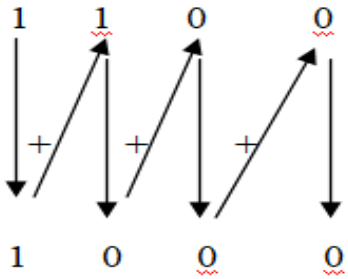
- Flip-flop is a 1 bit memory cell which can be used for storing the digital data.
- Bi stable devices having two stable states 0 and 1
- flip-flops are edge sensitive

Essay Questions

1. Convert 1011 to gray code and gray code 1100 to binary

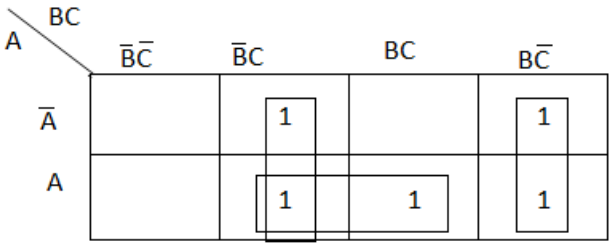


$(1011)_2=(1110)_{\text{gray code}}$



$(1100)_{gray\ code}=(1000)_2$

2. Map the expression f=A'B'C+AB'C+A'BC'+ABC'+ABC



$\bar{B}C + B\bar{C} + AC$

3. Demonstrate a one bit comparator

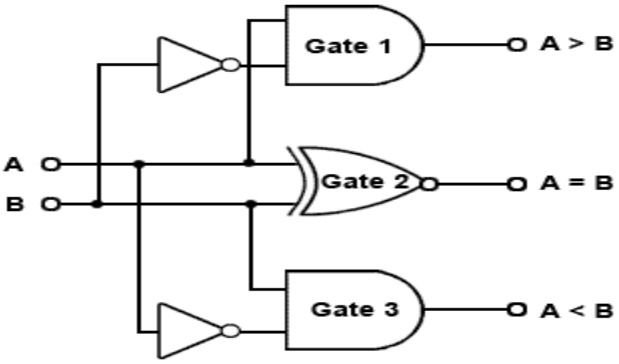
1-Bit Magnitude Comparator –

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers

Logic symbol	Truth table

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Circuit diagram



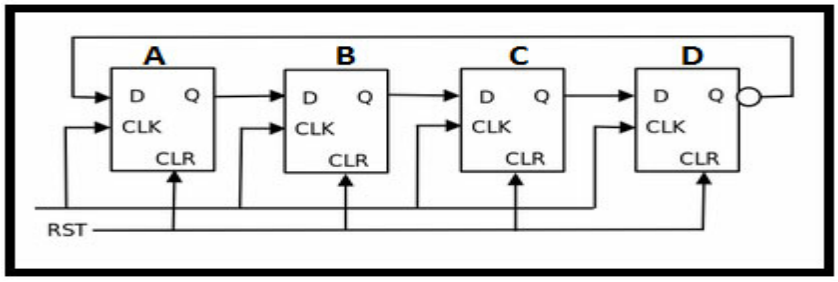
4. Compare sequential and combinational circuits

Combinational Circuits	Sequential Circuits
1. Outputs depend only on present inputs.	1. Outputs depend on both present inputs and present state.
2. Feedback path is not present.	2. Feedback path is present.
3. Memory elements are not required.	3. Memory elements are required.
4. Clock signal is not required.	4. Clock signal is required.
5. Easy to design.	5. Difficult to design.
6. **Examples –** Encoder, Decoder, Multiplexer, Demultiplexer	6. **Examples –** Flip-flops, Counters

5. Draw a 4 bit ring counter using D flip-flop

Ring Counter

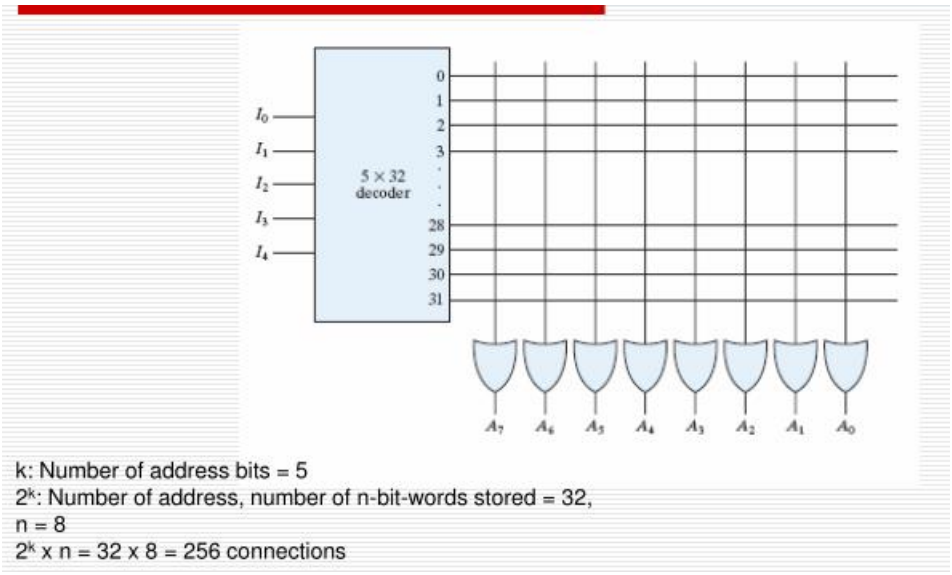
Ring counter is a basic application of shift registers. It is formed by the feedback of the output to its own input. This counter has N states where N denotes the number of flip-flops in the ring counter.



Q_A	Q_B	Q_C	Q_D
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

A 4 bit ring counter circuit is shown in the figure above. It consists of 4 D-flipflops, FFA, FFB, FFC and FFD. Each of these D flip-flops has an input D and output Q. At first, a **CLEAR** signal is applied to the flip-flops to RESET the outputs to zero. Then a **PRESET** pulse is applied to the flip-flop FFA before the clock pulse is given. This step allows putting the value '1' to the ring counter circuit. When each time the clock pulse is given, the counter circulates the data among all the four flip-flops.

6. Draw an internal diagram of 32× 8 ROM



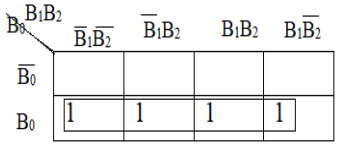
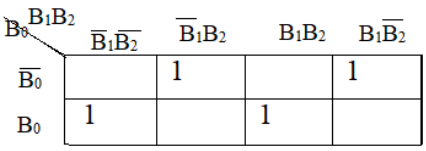
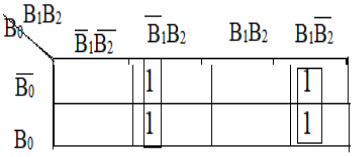
k: Number of address bits = 5
2^k: Number of address, number of n-bit-words stored = 32,
n = 8
2^k x n = 32 x 8 = 256 connections

7. Design and implement a 3 bit Binary to Gray code converter

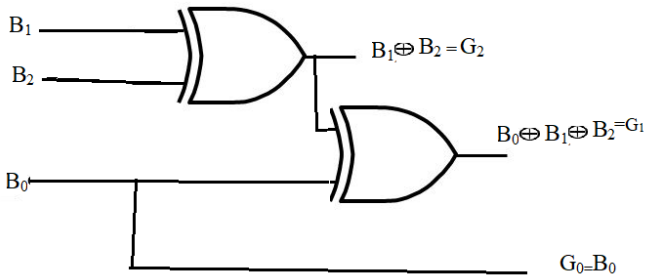
Truth table

Input			Output		
B_0	B_1	B_2	G_0	G_1	G_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

K map for G_0	K map for G_1	K map for G_2

 $G_0=B_0$	 $G_1=B_0B_1B_2+B_0B_1B_2+B_0B_1B_2+B_0B_1B_2$ $G_1=\overline{B}_0\overline{B}_1B_2+\overline{B}_0B_1\overline{B}_2+B_0\overline{B}_1\overline{B}_2+B_0B_1B_2$ $B_2\ (\overline{B}_0\overline{B}_1+B_0\,B_1) + \overline{B_2}\ (\overline{B}_0\,B_1+B_0\,\overline{B_1})$ $B_2\ (\overline{B_0 \oplus B_1}) + \overline{B_2}\ (B_0 \oplus B_1)$ $B_0 \oplus B_1 \oplus B_2$	 $\overline{B}_1B_2 + B_1\overline{B}_2$ $B_1 \oplus B_2$

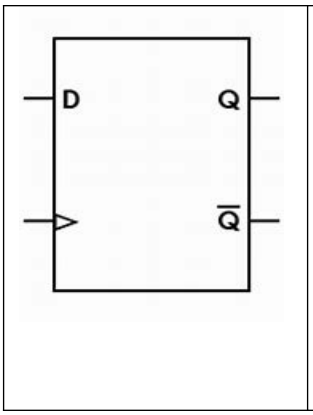
Circuit diagram

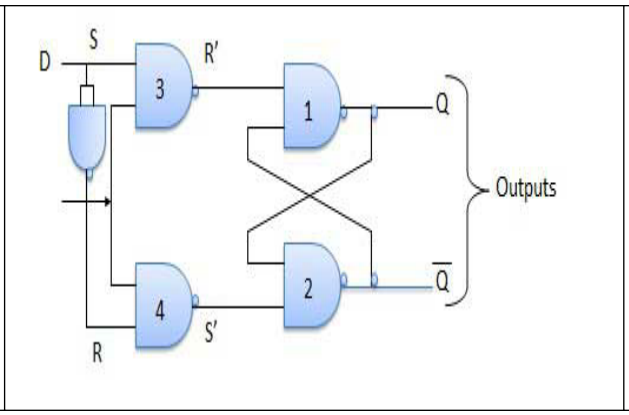


8. Write a short note on D flip flop ,draw the logic symbol and truth table for a D flip flop

   Delay Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs. It has only one input. The input data is appearing at the output after some time. Due to this data delay between i/p and o/p, it is called delay flip flop. S and R will be the complements of each other due to NAND inverter. Hence S = R = 0 or S = R = 1, these input condition will never appear. This problem is avoid by SR = 00 and SR = 11 conditions.

Logic symbol	Circuit diagram	





Inputs	Outputs		Comments
D	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	1	Rset
1	1	0	Set

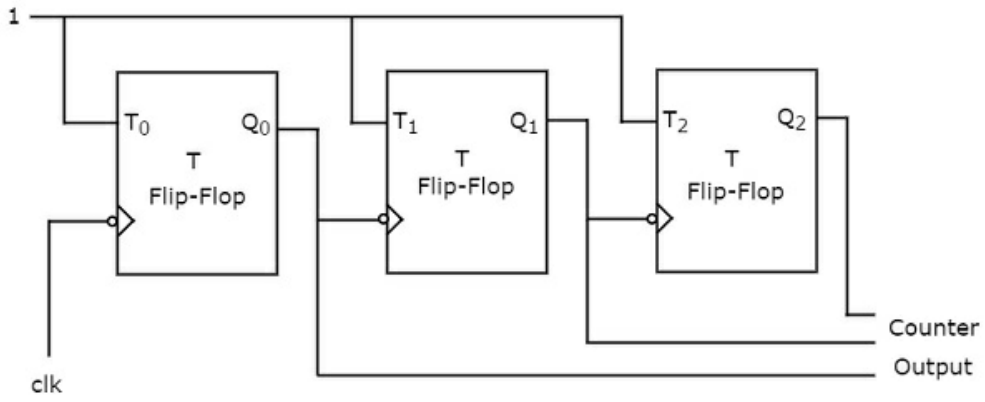
As long as the clock input is low, changes at the D input make no difference to the outputs.).

Provided that the CK input is high (at logic 1), then whichever logic state is at D will appear at output Q and (unlike the SR flip-flops) Q' is always the inverse of Q).

If D = 1, then S must be 1 and R must be 0, therefore Q is SET to 1.

If D = 0 then R must be 1 and S must be 0, causing Q to be reset to 0.

9. Draw a 3 bit asynchronous counter using T flipflop



The 3-bit Asynchronous binary up counter contains three T flip-flops and the T-input of all the flip-flops are connected to '1'. All these flip-flops are negative edge triggered but the outputs change asynchronously. The clock signal is directly applied to the first T flip-flop. So, the output of first T flip-flop **toggles** for every negative edge of clock signal.

The output of first T flip-flop is applied as clock signal for second T flip-flop. So, the output of second T flip-flop toggles for every negative edge of output of first T flip-flop.

Similarly, the output of third T flip-flop toggles for every negative edge of output of second T flip-flop, since the output of second T flip-flop acts as the clock signal for third T flip-flop.

Assume the initial status of T flip-flops from rightmost to leftmost is Q2Q1Q0=000. Here, Q2 & Q0 are MSB & LSB respectively. We can understand the **working** of 3-bit asynchronous binary counter from the following table.

No of negative edge of Clock	Q_2(MSB)	Q_1	Q_0(LSB)
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Here Q0 toggled for every negative edge of clock signal. Q1 toggled for every Q0 that goes from 1 to 0, otherwise remained in the previous state. Similarly, Q2 toggled for every Q1 that goes from 1 to 0, otherwise remained in the previous state.
The initial status of the T flip-flops in the absence of clock signal is Q2Q1Q0=000. This is incremented by one for every negative edge of clock signal and reached to maximum value at 7^{th} negative edge of clock signal. This pattern repeats when further negative edges of clock signal are applied.

10. Describe the need of DAC and ADC in digital systems

Need for ADC

- To sample and digitally analyze signals for computational purpose
- To digitize speed of video signals for filtering by digital circuits

- To feed the digital signals to computers

Need for DAC

- To display the digital output of a digital system in analog form
- To reconstruct the analog signal
- To synthesize the speed, video signals etc

11. List and explain different types of ROMs

ROM stands for **Read Only Memory**. The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture. A ROM stores such instructions that are required to start a computer. This operation is referred to as **bootstrap**. ROM chips are not only used in the computer but also in other electronic items like washing machine and microwave oven.

A. MROM (Masked ROM)

The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs, which are inexpensive.

B. PROM (Programmable Read Only Memory)

PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

C. EPROM (Erasable and Programmable Read Only Memory)

EPROM can be erased by exposing it to ultra-violet light for a duration of up to 40 minutes. Usually, an EPROM eraser achieves this function. During programming, an electrical charge is trapped in an insulated gate region. The charge is retained for more than 10 years because the charge has no leakage path. For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid). This exposure to ultra-violet light dissipates the charge. During normal use, the quartz lid is sealed with a sticker.

D. EEPROM (Electrically Erasable and Programmable Read Only Memory)

EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10 ms (millisecond). In EEPROM, any location can be selectively

erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of reprogramming is flexible but slow.

Advantages of ROM

The advantages of ROM are as follows −

- Non-volatile in nature
- Cannot be accidentally changed
- Cheaper than RAMs
- Easy to test
- More reliable than RAMs
- Static and do not require refreshing
- Contents are always known and can be verified

12. State the advantage of performing subtraction using 2's complement method.Perform 2's complement subtraction for the following binary numbers

- The primary advantage of two's complement over one's complement is that two's complement only has one value for zero. One's complement has a "positive" zero and a "negative" zero.
- To add numbers using one's complement you have to first do binary addition, then add in an end-around carry value.
- Two's complement has only one value for zero, and doesn't require carry values.

a) 110000-10101

110000-10101

No of bits in 110000 6

No of bits in 10101 is 5

Make the no of bits are equal by adding zeros to the left of 10101

110000-10101

No of bits in 110000 6

No of bits in 10101 is 5

Make the no of bits are equal by adding zeros to the left of 10101

110000-

010101

2's complement of 010101 is(101010+1=101011)

Add 2's complement of 010101 to 110000

Ie,

1 1 0 0 0 0+

1 0 1 0 1 1

0 1 1 0 1 1 and carry=1

Omit this carry

Result is 1 1 0 1 1

b) 1001-101000

No of bits in 1001-4

No of bits 101000 -6

Make no of bits are equal in 1001 as 001001

2's complement of 101000(1's complement of 101000 +1) is 010111+1 =

0 1 0 1 1 1 +

1

---

0 1 1 0 0 0

Add this 2's complement to 001001

0 0 1 0 0 1 +

0 1 1 0 0 0

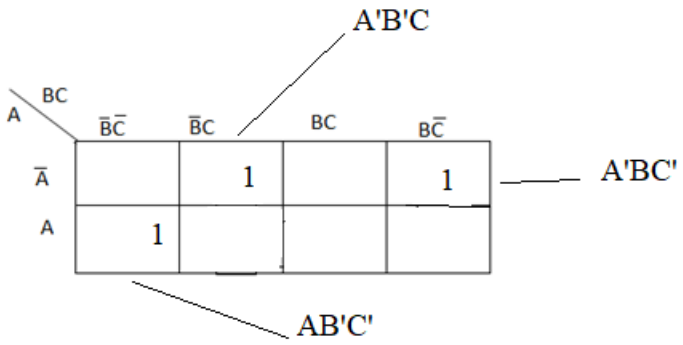
---

1 0 0 0 0 1

There is no carry result is in 2's complement form and is negative ie 2's complement of 1 0 0 0 0 1

(011110+1)= - 011111

13. Map the expression f=A'B'C+A'BC'+AB'C'



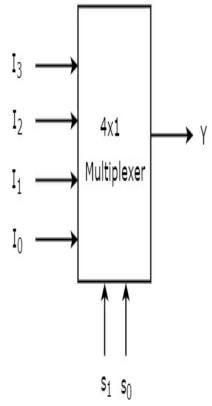
A'B'C+A'BC'+AB'C'

14. Draw the logic diagram of a four input Multiplexer

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & $s_0$and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.

Logic symbol	Truth table



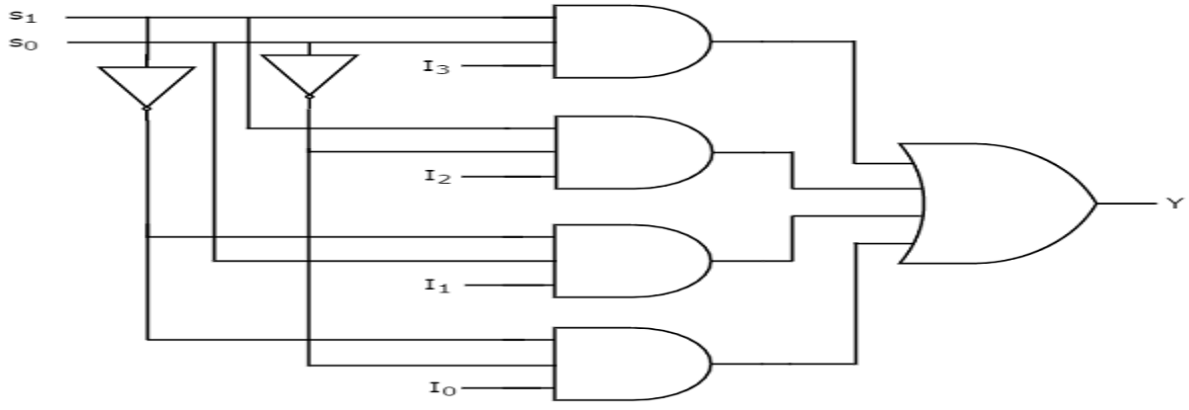
Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the **Boolean function** for output, Y as

Y=S1'. S0'. I0+S1'. S0. I1+S1.S0.'I2+S1.S0.I2

We can implement this Boolean function using Inverters, AND gates & OR gate. The **circuit diagram** of 4x1 multiplexer is shown in the following figure.

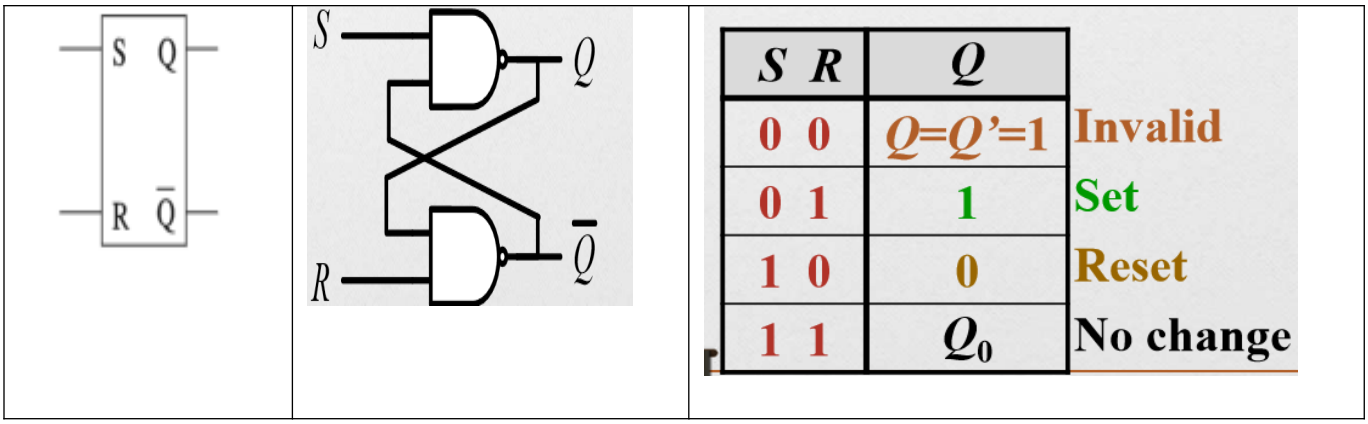
Circuit Diagram



15. Describe SR latch using NAND gates

SR Latch is also called as **Set Reset Latch**. SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates It consist of two inputs named S for Set and R for Reset.It has two useful states , when output Q=1 and Q'=0 ,and latch is said to be in Set State. When Q=0 and Q'=1 , it is in Reset Sate.

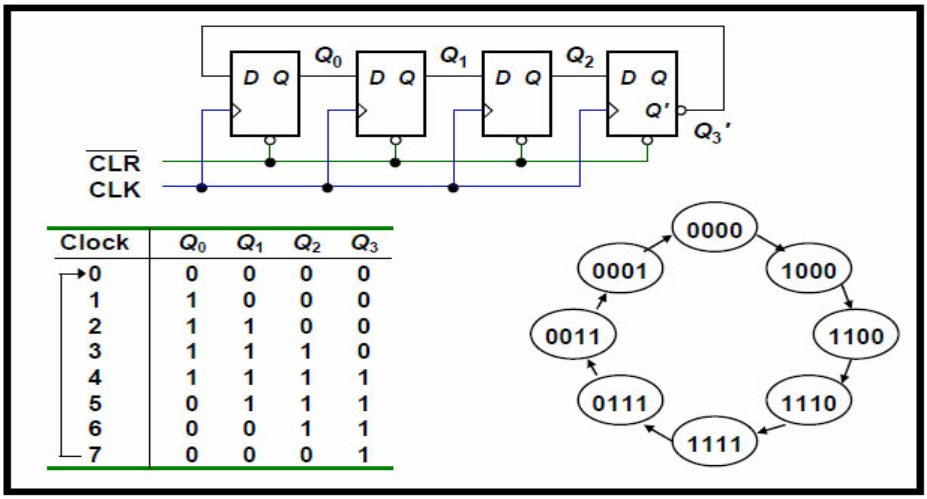
Logic symbol	Circuit diagram	Truth table



16. Construct a Johnson counter using D flip flop

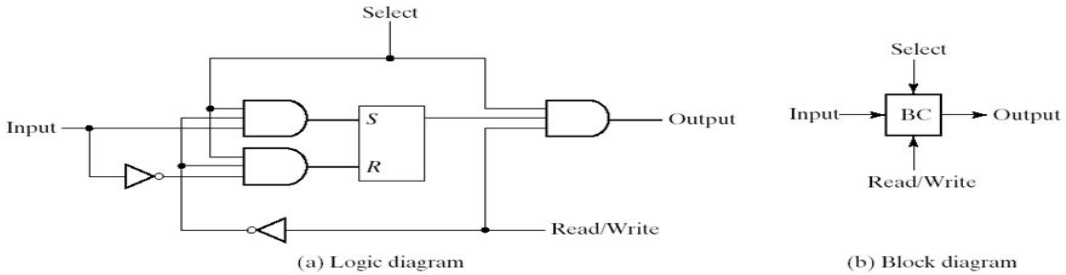
Johnson Counter

Johnson Counter also known as **Twisted Ring Counter** is another basic application of shift registers with a feedback. Here the feedback is given from the inverted output of the last flip flop to the input of the first flip-flop.Figure below shows a 4-bit Johnson counter. It consists of four flip-flops FF0, FF1, FF2 and FF3. Here the inverted output of the last flip-flop FF3 is given as feedback to the input of the first flip-flop FF0. Here, at first four logic zeros will be passed to the flip-flops. When clock pulses are given "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" outputs will be obtained and the sequence will repeat for the next clock pulses.



17. Illustrate the logic diagram of a memory cell

- A memory cell



18. Reduce the expression F=A+B[AC+(B+C')D]

F=A+B[AC+BD+C'D]

=A+ABC+BD+BC'D

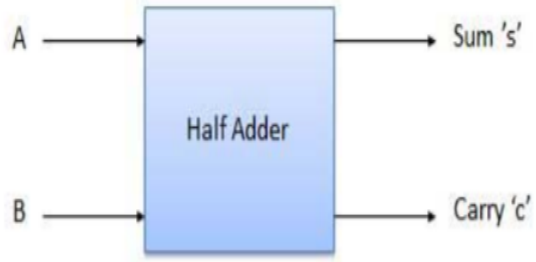
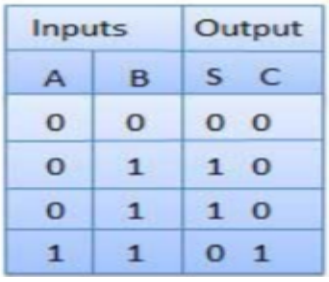
=BD(1+C')+A(1+BC)

=BD+A(1+BC)

19. Design a Half adder with truth table ,expressions and logic diagram

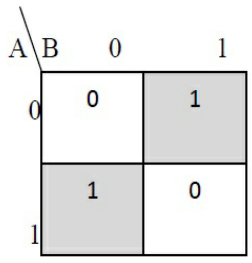
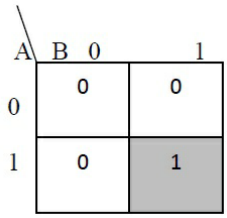
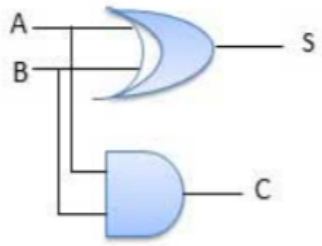
Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary numbers A and B. This circuit has two outputs carry and sum.

Sum S = A ⊕ B = AB' + A' B Carry C= AB

Logic Symbol	Truth Table
	

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
0	1	1	0
1	1	0	1

K map for Sum	K Map for carry	Circuit diagram

 Sum = $\overline{A}$ B + A $\overline{B}$.= A ⊕ B	 Carry =A.B	

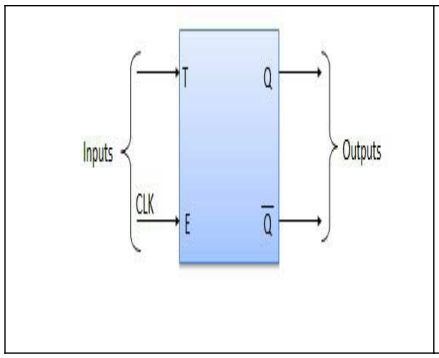
20. Differentiate between synchronous and asynchronous sequential circuits

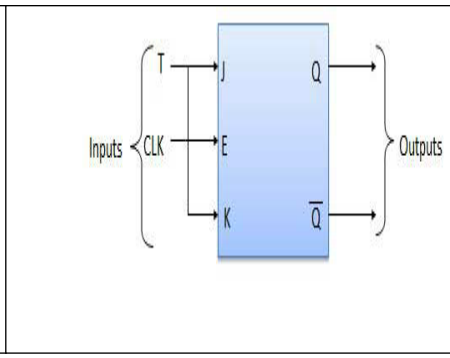
Synchronous Sequential Circuit	Asynchronous Sequential Circuit
• It is easy to design.	• It is difficult to design.
• A clocked flip flop acts as memory element.	• An unclocked flip flop or time delay is used as memory element.
• They are slower as clock is involved.	• They are comparatively faster as no clock is used here.
• The states of memory element is affected only at active edge of clock, if input is changed.	• The states of memory element will change any time as soon as input is changed.

21. Explain the working of a T flip flop with logic diagram and truth table

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together.

Symbol Diagram	Block Diagram	Truth Table



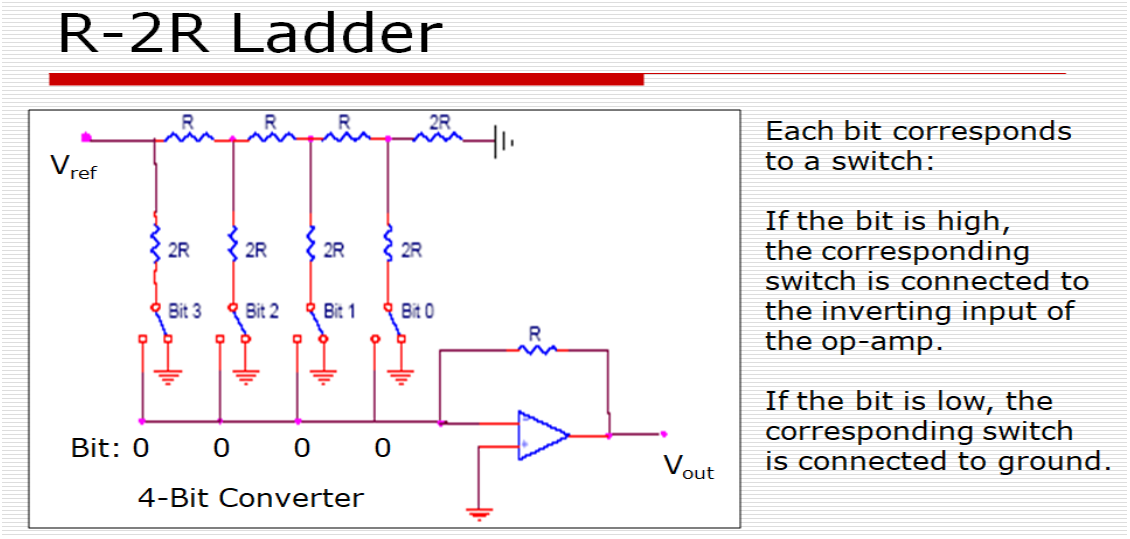


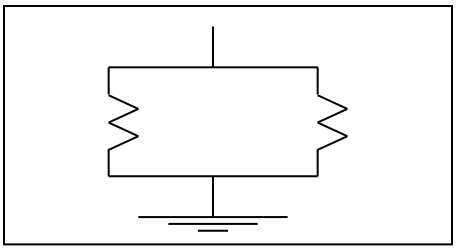
Inputs		Outputs		Comments
E	T	Q_{n+1}	$\bar{Q}_{n+1}$	
1	0	Q_n	$\bar{Q}_n$	No change
1	1	$\bar{Q}_n$	Q_n	Toggle

Operation

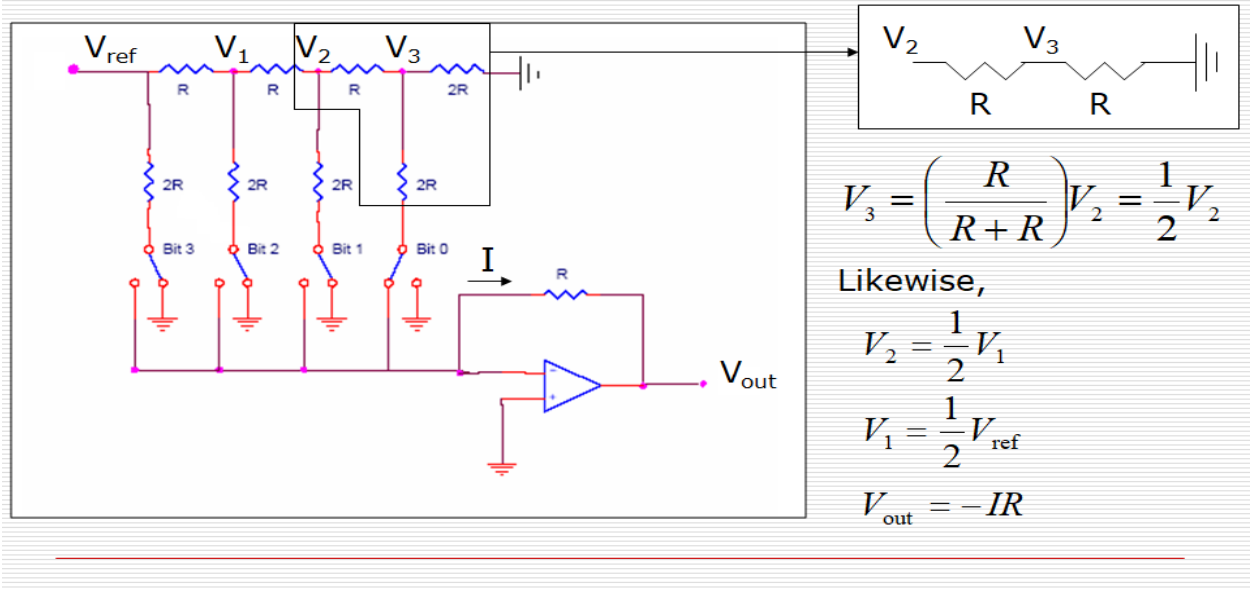
S.N.	Condition	Operation
1	**T = 0, J = K = 0**	The output Q and Q'won't change
2	**T = 1, J = K = 1**	Output will toggle corresponding to every leading edge of clock signal.

22. Describe the working of R-2R DAC/Explain 4 bit DAC





$$R_{eq} = \frac{(2R)(2R)}{(2R+2R)} = R$$



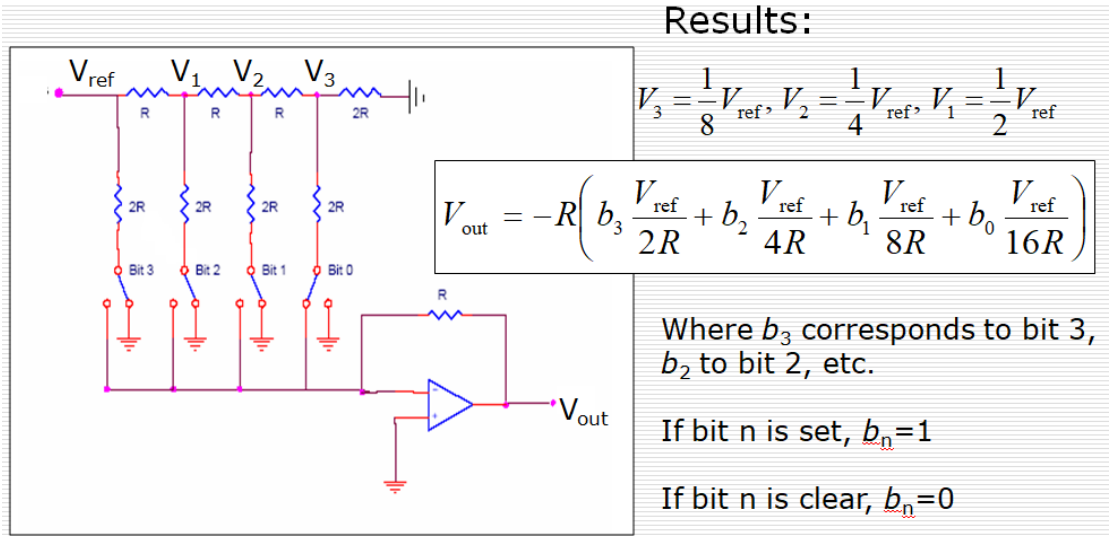
$$V_3 = \left(\frac{R}{R+R}\right)V_2 = \frac{1}{2}V_2$$

Likewise,

$$V_2 = \frac{1}{2}V_1$$

$$V_1 = \frac{1}{2}V_{ref}$$

$$V_{out} = -IR$$



Results:

$$V_3 = \frac{1}{8}V_{ref},\ V_2 = \frac{1}{4}V_{ref},\ V_1 = \frac{1}{2}V_{ref}$$

$$V_{out} = -R\left(b_3\frac{V_{ref}}{2R} + b_2\frac{V_{ref}}{4R} + b_1\frac{V_{ref}}{8R} + b_0\frac{V_{ref}}{16R}\right)$$

Where b_3 corresponds to bit 3, b_2 to bit 2, etc.

If bit n is set, b_n=1

If bit n is clear, b_n=0

23. Explain about Alphanumeric codes

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

items. The following are alphanumeric codes are very commonly used for the data representation.

ASCII and EBCDIC

ASCII

- American Standard Code for Information Interchange.
- ASCII code is a 7-bit code
- ASCII code is more commonly used worldwide

EBCDIC

- Extended Binary Coded Decimal Interchange Code
- EBCDIC is used primarily in large IBM computers
- EBCDIC is an 8-bit code

24. Convert

a) decimal 251 to Octal

Number	Quotient	Reminder
251/8	31	3
31/8	3	7
3/8	0	3

Collect reminders from bottom to top -373

$(251)_{10}=(373)_8$

b) B9 to decimal

$9\times16^0 + 11(B) \times16^1$

=9+11*16

=185

$(B9)_{16}=(185)_{10}$

25. Briefly explain Excess-3 code and gray code

Excess-3 code

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

Gray Code

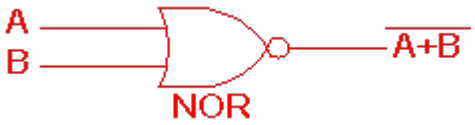
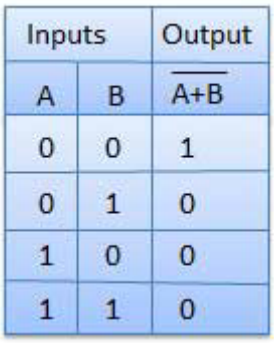
It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time the decimal number is incremented

As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

26. Explain the behavior of universal gates with logic diagram and truth table (*2 times)*

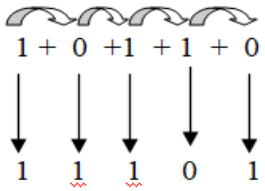
NAND gate and NOR gate are universal gates

Gate	Function	Logic diagram	Truth table
NAND Gate	A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.		Inputs: A, B; Output: $\overline{AB}$ 0 0 1 0 1 1 1 0 1 1 1 0

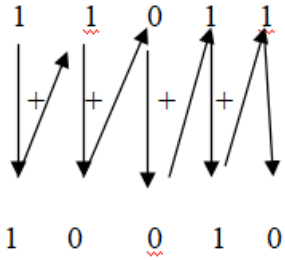
NOR Gate	A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.		

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

27. Convert $(10110)_2$ to gray code and $(11011)_{gray}$ to binary

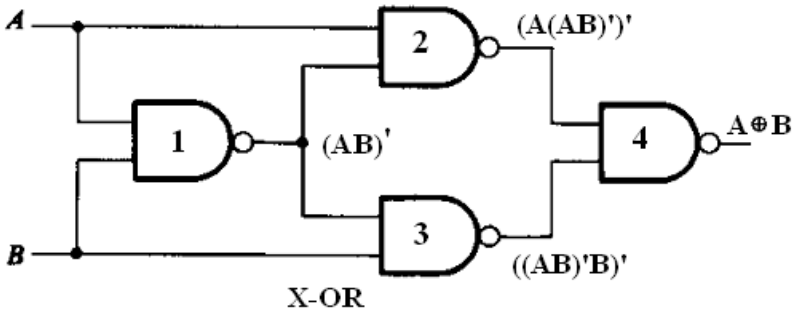


$(10110)_2 = (11101)_{gray}$



$(11011)_{gray} = (10010)_2$

28. Implement X-OR gate using NAND gates



29. Convert the following SOP into standard SOP

Y=A+B'C

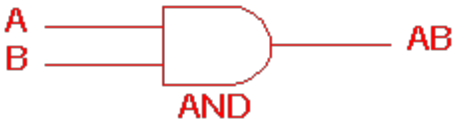
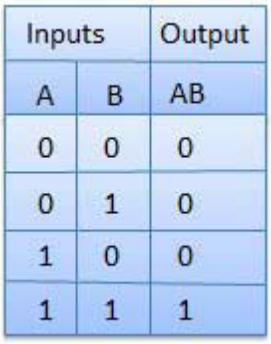
=A(B+B')(C+C')+B'C(A+A')

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

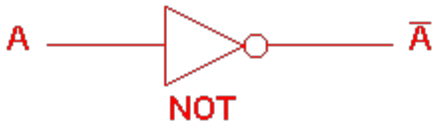
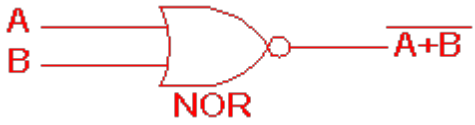
=ABC+ABC'+AB'C+AB'C'+AB'C+A'B'C

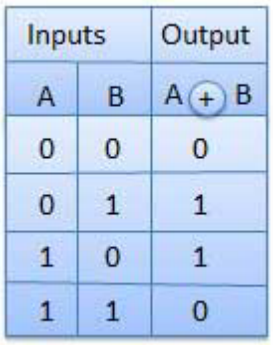
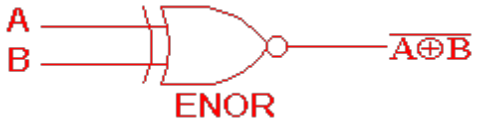
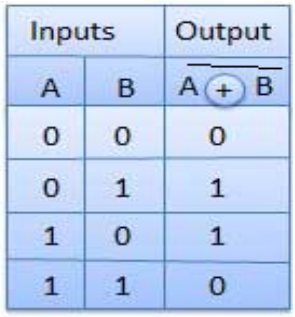
=ABC+ABC'+AB'C+AB'C'+A'B'C'

30. Draw the symbol and truth table of logic gates

Gate	Function	Logic diagram	Truth Table
AND gate	AND gate operation is logical multiplication . It has n input (n >= 2) and one output. Y= A.B		

Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

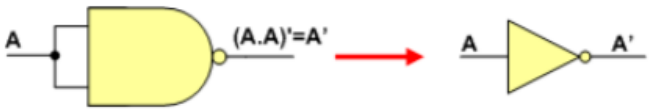
<table>
<tr>
<td>OR Gate</td>
<td>OR gate operation is logical addition It has n input (n >= 2) and one output.
Y=A+B</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Inputs</th><th>Output</th></tr>
<tr><td>A</td><td>B</td><td>A + B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
<tr>
<td>NOT Gate</td>
<td>NOT gate is also known as Inverter. It has one input A and one output Y.
Y=A ——</td>
<td> 
</td>
<td>
<table>
<tr><th>Inputs</th><th>Output</th></tr>
<tr><td>A</td><td>B</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>NAND Gate</td>
<td>A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Inputs</th><th>Output</th></tr>
<tr><td>A</td><td>B</td><td>$\overline{AB}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>NOR Gate</td>
<td>A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Inputs</th><th>Output</th></tr>
<tr><td>A</td><td>B</td><td>$\overline{A+B}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

X-OR Gate	The exclusive -OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.		
EX-NOR gate	The exclusive -NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.		

31. Draw AND, OR ,NOT and XOR using NAND gate only

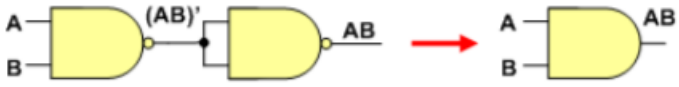
NOT gate

1. All NAND input pins connect to the input signal **A** gives an output **A'**.



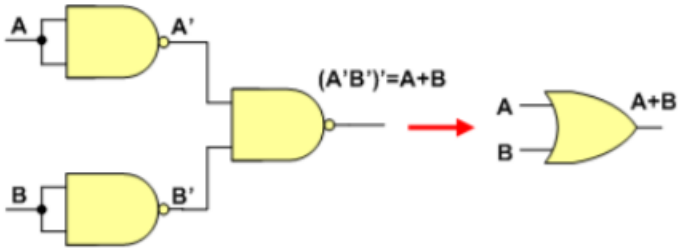
Implementing AND Using only NAND Gates

An AND gate can be replaced by NAND gates as shown in the figure (The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter).

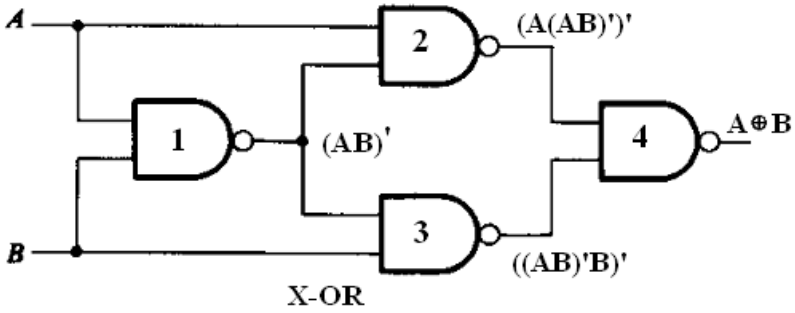


Implementing OR Using only NAND Gates

An OR gate can be replaced by NAND gates as shown in the figure (The OR gate replaced by a NAND gate with all its inputs complemented by NAND gate inverters)



Implementing XOR using NAND gates



32. Convert the following

a) $(237.25)_{16}$ to Octal

Convert each digit to four bit hexadecimal

2	3	7	.	2	5
0010	0011	0111	.	0 010	0101

Regroup 4 bit group to 3 bit group and write corresponding octal digit for each 3 bit group

001 000 110 111. 001 001 010

1 0 6 7 . 1 1 2

Therfore $(237.25)_{16}=(1067.112)_8$

b) $(11010011.1111)_2$ to Hexadecimal

1101 0011 . 1111

D(13) 3 . F (15)

$(11010011.1111)_2= (D3.F)_{16}$

c) 232.17 to Binary

Integer part is 232.Division of 232 and successive quotients with base 2.

Operation	Quotient	Remainder(LSB)
232/2	116	0
116/2	58	0
58/2	29	1
29/2	14	1
14/2	7	0
7/2	3	1
3/2	1	1
½	0	1(MSB)

232=$(11101100)_2$

Fractional part is0.17.Multiplication of 0.17 and successive fractions with base 2.

Operation	Result	Carry
0.17×2	0.34	0
0.34×2	0.68	0
0.68×2	1.36	1
0.36×2	0.72	0

$(0.17)=(0010)_2$

$232.17=(11101100.0010)_2$

33. Describe Laws of Boolean algebra

AND law	(i) $A.0 = 0$ (ii) $A.1 = A$ (iii) $A.A = A$ (iv) $A.\overline{A} = 0$
OR law	(i) $A + 0 = A$ (ii) $A + 1 = 1$ (iii) $A + A = A$ (iv) $A + \overline{A} = 1$
Commutative law	(i) $A.B = B.A$ (ii) $A + B = B + A$
Associative law	(i) $(A.B).C = A.(B.C)$ (ii) $(A + B) + C = A + (B + C)$
Distributive law	$A.(B + C) = A.B + A.C$
INVERSION law	$\overline{\overline{A}} = A$
De Morgan's Law	$\overline{A.B} = \overline{A} + \overline{B}$ NAND = Bubbled OR

	$\overline{A+B} = \overline{A}.\overline{B}$ NOR = Bubbled AND

34. .Convert the following

a) $(10110.0101)_2$ to Hexadecimal

0001 0110 . 0101

1 6 . 5

b) $(F4B.11)_{16}$ to Binary

Convert each hexadecimal digit into 4 bit binary

F 4 B . 1 1

1111 0100 1011 0001 0001

c) $(26.24)_8$ to decimal

$2\times8^1 + 6\times8^0 + 2\times8^{-1} + 4\times 8^{-2}$

=16+ 6 +2/8+4/64

=22+0.25+0.0625

$=(22.31)_{10}$

d) Decimal 85.25 to Octal

Integer part is 85.Division of 85 and successive quotients with base 8.

Operation	Quotient	Remainder(LSB)
85/8	10	5
10/8	1	2
1/8	0	1(MSB)

$(85)_{10}= (125)_8$

Fractional part is 0.25.Multiplication of 0.25 and successive fractions with base 8.

Operation	Result	Carry
0.25×8	2.00	2
0.00×8	0.00	0

$(0.25)_{10}=(20)_8$

$(85.25)_{10}=(125.2)_8$

35. Using theorems of Boolean algebra, prove the following

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

LHS (A+B).(A+C)

=A.A+A.C+B.A+B.C

=A+A.C+A.B+B.C [A.A=A

=A(1+C)+A.B+B.C

=A+A.B+B.C [1+C=1

=A(1+B)+B.C [1+B=1

=A+B.C

=RHS

36. Convert the following

a) $(2A0F.12)_{16}$ to Octal

Convert each hexadecimal digit into 4 bit binary group

2 A(10) 0 F(15) . 1 2 0010 1010 0000 1111 . 0001 0010
Regroup 4 bit group into 3 bit group and write corresponding Octal digit for eac3 bit group 000 010 101 000 001 111 .000 100 100 0 2 5 0 1 7 . 0 4 4

b) $(267.11)_8$ to Hexadecimal

Convert each octal digit into 3 bit binary group 2 6 7 . 1 1 010 110 111 . 001 001
Regroup 3 bit group into 4 bit group and write corresponding Hexadecimal for eac 4 bit group 0000 1011 0111. 0010 0100 0 B (11) 7 . 2 4

$(267.11)_8 = (B7.24)_{16}$

c) 167.12 to binary

Integer part is 167.Division of 167 and successive quotients with base 2.

Operation	Quotient	Remainder(LSB)
167/2	83	1
83/2	41	1
41/2	20	1
20/2	10	0
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1(MSB)

$167=(10100111)_2$

Fractional part is 0.12.Multiplication of 0.12 and successive fractions with base 2.

Operation	Result	Carry
0.12×2	0.24	0
0.24×2	0.48	0
0.48×2	0.96	0
0.96×2	1.92	1

$(0.12)=(0001)_2$

$167.12=(10100111.0001)_2$

24. Convert the following SOP into standard SOP

Y=A+B'C

Missing variable in first AND term is B and C

Missing variable in second AND term is A

Therefore

A(B+B')(C+C')+(A+A')B'Ç

(AB+AB')(C+C')+AB'Ç+A'B'Ç

=ABC+ABC'+AB'Ç+AB'Ç'+AB'Ç+A'B'Ç

=ABC+ABC'+AB'Ç+AB'C'+A'B'C

Standard SOP form of A+B'C is ABC+ABC'+AB'Ç+AB'C'+A'B'C

25. By using Boolean algebra ,Prove the following

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

LHS=

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

=A+A.C+A.B+B.C [A.A=A

=A[1+C]+A.B+B.C

=A+A.B+B.C [1+C=1

=A[1+B]+B.C

=A+B.C [1+B=1

=RHS

26. Expand A'+B' to min terms and max terms

Minterms

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

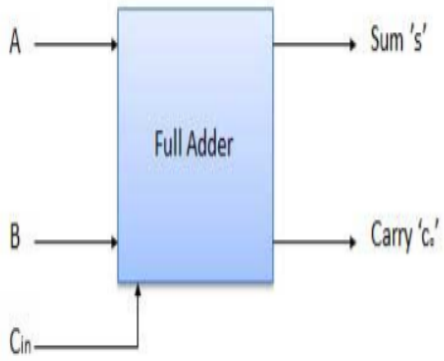
A'B+A'B'+AB'+A'B'

A'B+AB'+A'B'

Maxterms

27. Design a full adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

Block diagram	Truth Table
	

Inputs			Output	
A	B	Cin	S	Co
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	K Map for Carry

A \ BC_{IN}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$\text{Sum} = \overline{A}\,\overline{B}\,C_{in} + \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + A\,B\,C_{in}$$

$$= C_{in}\,(\overline{A}\,\overline{B} + AB) + \overline{C}_{in}\,(\overline{A}\,B + A\,\overline{B})$$

$$= C_{in}\,(A \odot B) + \overline{C}_{in}\,(A \oplus B)$$

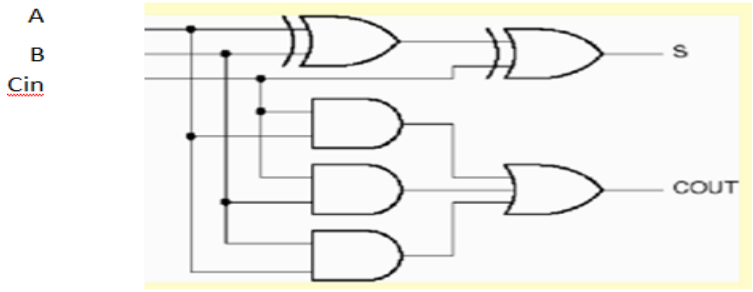
$$= C_{in}\,(\overline{A \oplus B}) + \overline{C}_{in}\,(A \oplus B)$$

$$= C_{in} \oplus (A \oplus B)$$

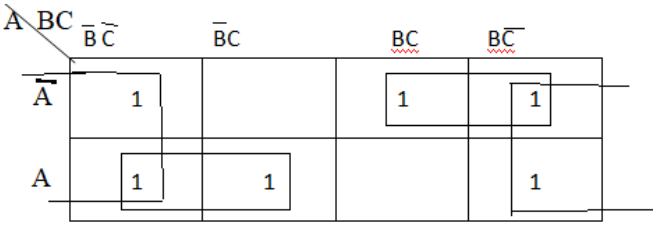
A \ BC_{IN}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_{out} = AB + A\,C_{in} + B\,C_{in}$

Circuit Diagram

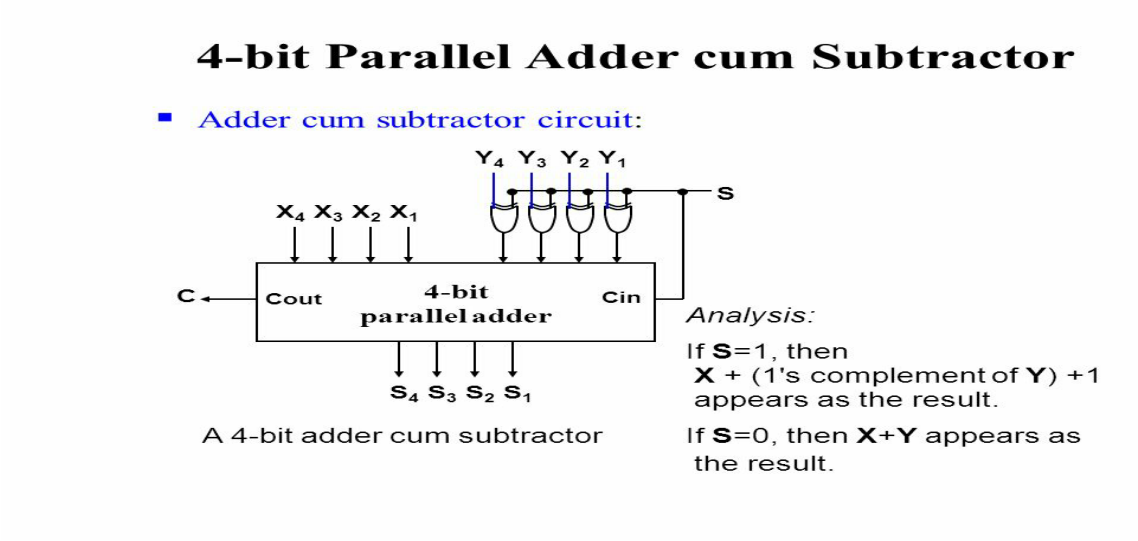


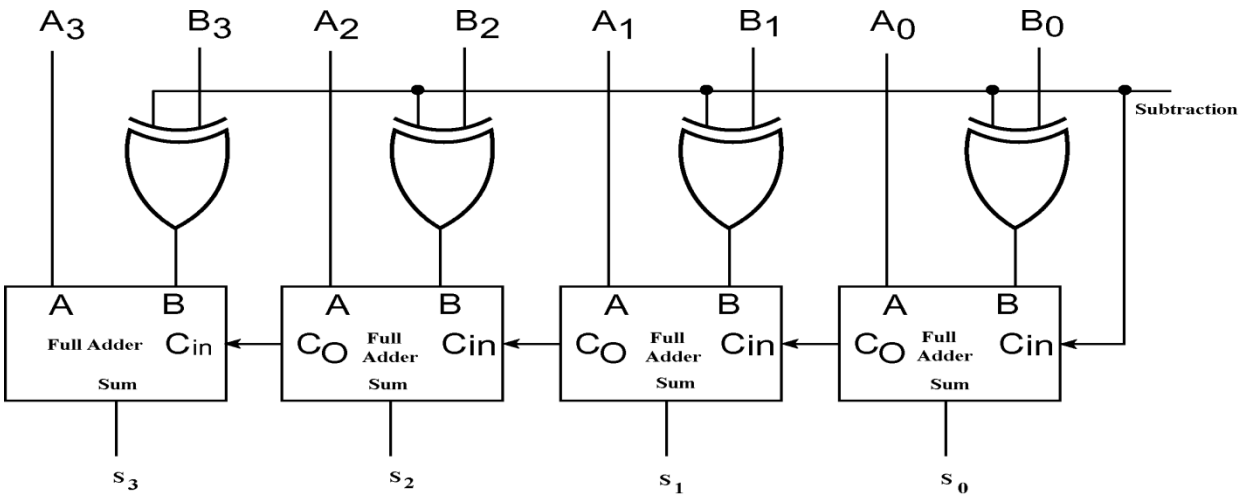
28. Reduce the expression f=∑(0,2,3,4,5,6) using k map



$\overline{C} + A\overline{B} + \overline{A}B$

29. Demonstrate a 4 bit adder-subtractor with suitable neat diagram





The control input is controls the addition or subtraction operation.When the SUBTRACTION input is logic '0' , the B3 B2 B1 B0 are passed to the full adders. Hence, the output of the full adders is the addition of the two numbers

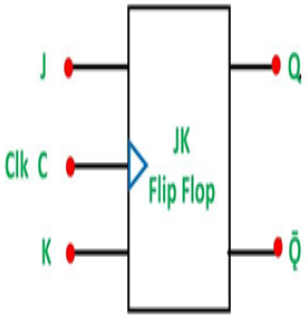
When the SUBTRACTION input is logic '1' , the B3 B2 B1 B0 are complemented. Further, the SUBTRACTION logic '1' input is also work as Cin for the LSB full adder, due to which 2's complement addition can be carried out

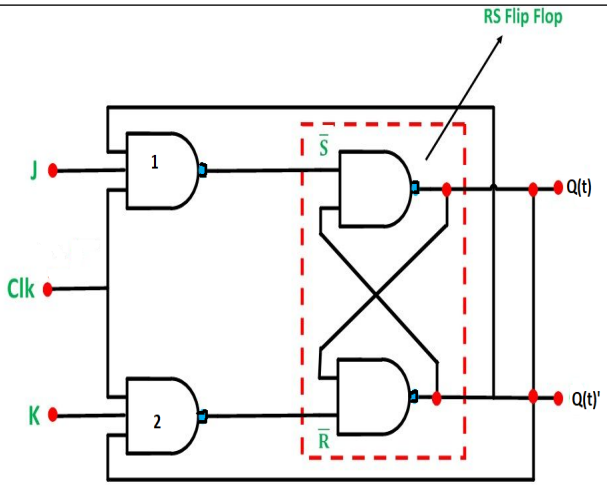
30. Demonstrate a JK flip flop with truth table

The sequential operation of the JK Flip Flop is same as for the RS flip-flop with the same SET and RESET input. The difference is that the JK Flip Flop does not have the

invalid input states of the RS Latch (when S and R are both 1).

Logic symbol	Circuit diagram	Truth table





J	K	$Q_{(t+1)}$
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'

Operation

- When J and K are both 0 the flip-flop is inhibited, Q is the same after the Clk pulse as it was before; there is no change at the output.
- When J and K are at different logic levels, then after the CK pulse, Q and Q will take up the same states as J and K. For example, if J = 1 and K = 0, then on the trailing (negative going) edge of a clock pulse, the Q output will be set to 1, and if K = 1 and J = 0 then the Q output is reset to logic 0 on the trailing edge of a clock pulsee D input with J.
- When logic 1 is applied to both J and K, the output toggles at the trailing edge of each clock pulse, just like a toggle flip-flop

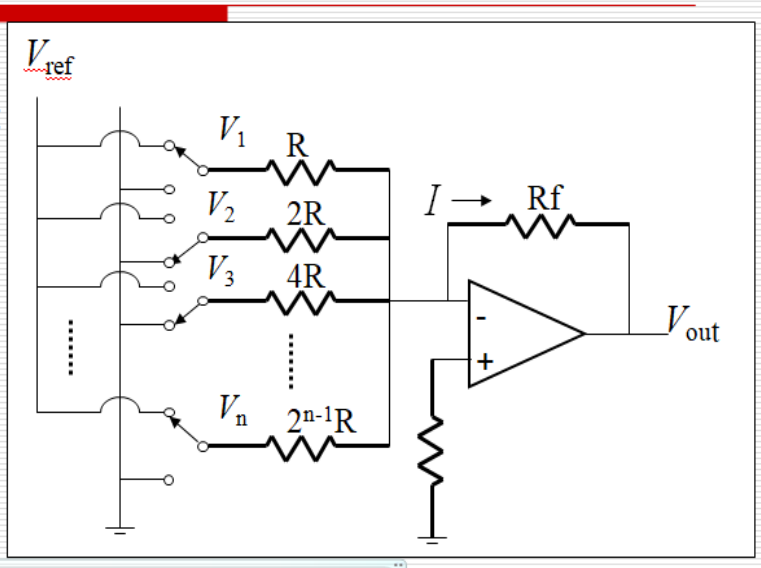
31. Explain a binary weighted resister type DAC

Binary Weighted Resistor

Voltages V_1 through V_n are either V_{ref} if corresponding bit is high or ground if corresponding bit is low

V_1 is most significant bit

V_n is least significant bit



MSB

$$V_{out} = -IR_f = -R_f\left(\frac{V_1}{R} + \frac{V_2}{2R} + \frac{V_3}{4R} + \cdots \frac{V_n}{2^{n-1}R}\right)$$

LSB

If $R_f = R/2$

$$V_{out} = -IR_f = -\left(\frac{V_1}{2} + \frac{V_2}{4} + \frac{V_3}{8} + \cdots \frac{V_n}{2^n}\right)$$

For example, a 4-Bit converter yields

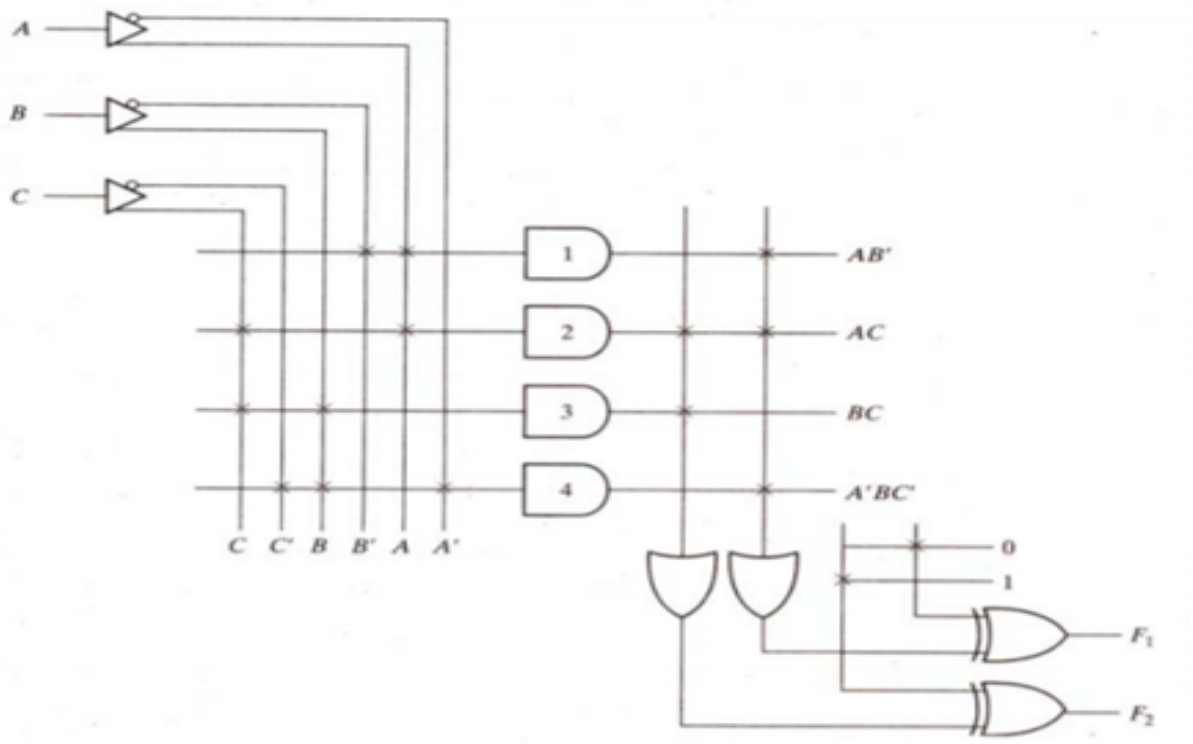
$$V_{out} = -V_{ref}\left(b_3\frac{1}{2} + b_2\frac{1}{4} + b_1\frac{1}{8} + b_0\frac{1}{16}\right)$$

Where b_3 corresponds to Bit-3, b_2 to Bit-2, etc.

32. Draw a logic diagram to implement the Boolean functions

F1=AB’+AC+A’BC’

F2=(AC+BC)’ in PLA with a PLA programming table



The product terms generated in each AND gate are listed along the output of the gate in the diagram. The product term is determined from the inputs whose crosspoints are connected and marked with a X . The output of an OR gate gives the logic sum of the selected product terms. The output may be complemented or left in its true form depending on the connection for one of the XOR gate inputs.

The PLA programming table consists of three sections .The first section lists the product terms numerically. The second section specifies the required paths between inputs and AND gates. The third section specifies the paths between AND and OR gates. For each output variable, we may have a T (for True) or C (for complement) for programming the XOR gate. The product terms listed on the left are not part of the table, they are included for reference only. For each product term the inputs are marked with 1,0 or --. I f a variable in the product term appears in its true form, the corresponding input var iable is marked with a 1. If it appears complemented the corresponding input variable is marked with a 0. If the variable is absent in the product term it is marked with a dash. The paths between the inputs and the AND gates are specified under the column heading inputs in the programming table. A 1 in the input column specifies a connection from the input variable to the AND gate. A 0 in the input column specifies a connection from the complement of the variable to input of the AND gate. A dash specifies a blown fuse in both the input variable and its complement. The paths between the AND gates are specified under the column heading outputs. The output variables are marked with 1's for those product terms that are included in the function. Each product term that has a 1 in the output column requires a path from the output of the AND gate to the input of the OR gate behaves like a 0.

PLA Programming Table

		Inputs			Outputs (T)	(C)
	Product Term	**A**	**B**	**C**	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

33. Develop a programming table for PAL for Boolean functions

w=ABC'+A'B'CD'

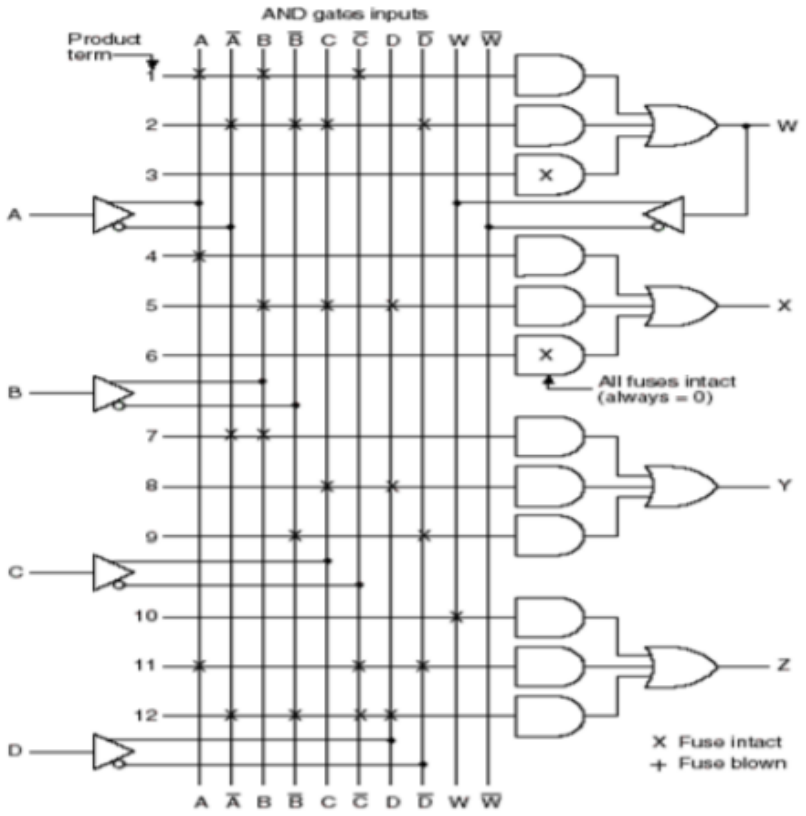
x=A+BCD

y=A'B+CD+B'D'

z=ABC'+A'B'CD'+AC'D'+A'B'C'D

Note that the function for Z has four product terms. The logical sum of two of these terms is equal to W. Thus, by using W, it is possible to reduce the number of terms for Z from four to three, so that the function can fit into the given PAL device. The PAL programming table is similar to the table used for the PLA, except that only the inputs of the AND gates need to be programmed.

Product term	AND Inputs A	B	C	D	W	Outputs
1	1	1	0	—	—	$W = A B \overline{C}$
2	0	0	1	0	—	$+\overline{A}\,\overline{B} C \overline{D}$
3	—	—	—	—	—	
4	1	—	—	—	—	$X = A$
5	—	1	1	1	—	$+BCD$
6	—	—	—	—	—	
7	0	1	—	—	—	$Y = \overline{A} B$
8	—	—	1	1	—	$+CD$
9	—	0	—	0	—	$+\overline{B}\,\overline{D}$
10	—	—	—	—	1	$Z = W$
11	1	—	0	0	—	$+A\overline{C}\,\overline{D}$
12	0	0	0	1	—	$+\overline{A}\,\overline{B}\,\overline{C} D$



34. Draw a logic diagram to implement the Boolean function

F1=(AB+AC+BC)'

F2=AB+AC+ A'B'C' in PLA

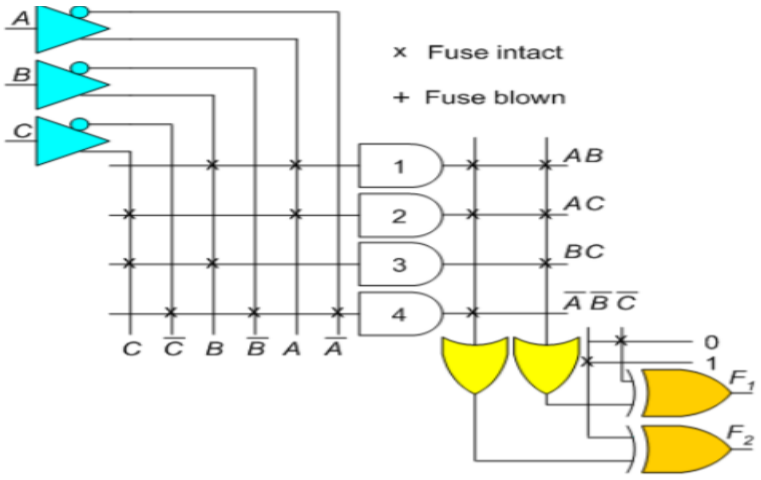
This gives only 4 distinct product terms: AB, AC, BC, and A'B'C'.

So the PLA table will be as follows:

PLA programming table

	Product term	Inputs A B C	Outputs (C) F_1	Outputs (T) F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$\overline{A}\overline{B}\overline{C}$	4	0 0 0	–	1

For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form , the corresponding input variable is marked with a 1. A 1 in the Inputs column specifies a path from the corresponding input to the input of the AND gate that forms the product term. A 0 in the Inputs column specifies a path from the corresponding complemented input to the input of the AND gate. A dash specifies no connection. The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input. In the Outputs column, a T (true) specifies that the other input of the corresponding XOR gate can be connected to 0, and a C (complement) specifies a connection to 1. Note that output F1 is the normal (or true) output even though a C (for complement) is marked over it. This is because F1' is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function F1' to produce the true F1 output as its second input is connected to logic 1.



35. List merits and demerits of K map

ADVANTAGES

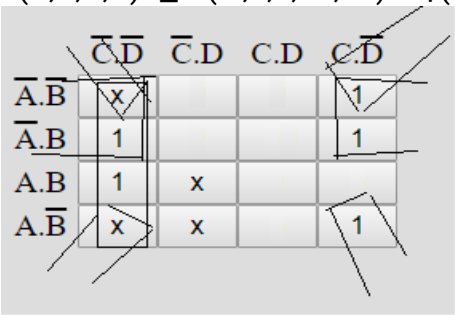
- Reduces extensive calc.
- Reduces expression without Boolean theorems
- Used for minimizing circuits
- Less time consuming
- Less space consuming

DISADVANTAGES

- Tedious for more than 5 variables
- Some examples are solved in few seconds by Boolean theorems easily

36. Simplify the following function using K map and draw the logic circuit for the simplified function

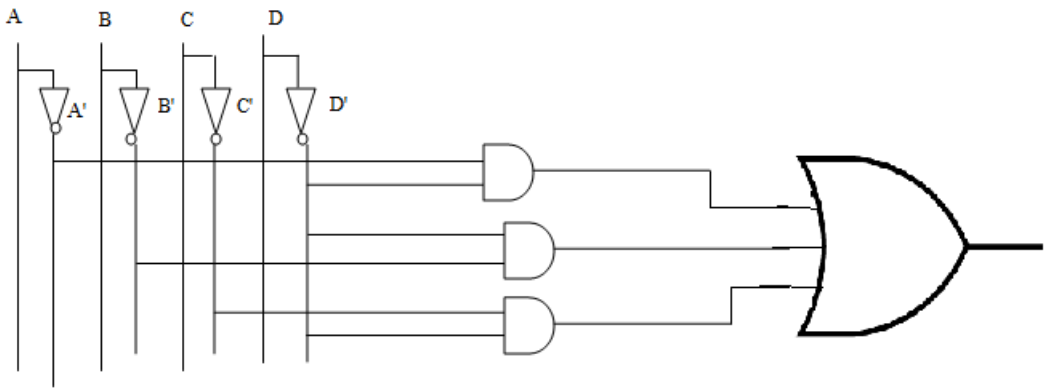
F(A,B,C,D)=∑m(2,4,6,10,12) + ¢(0,8,9,13)



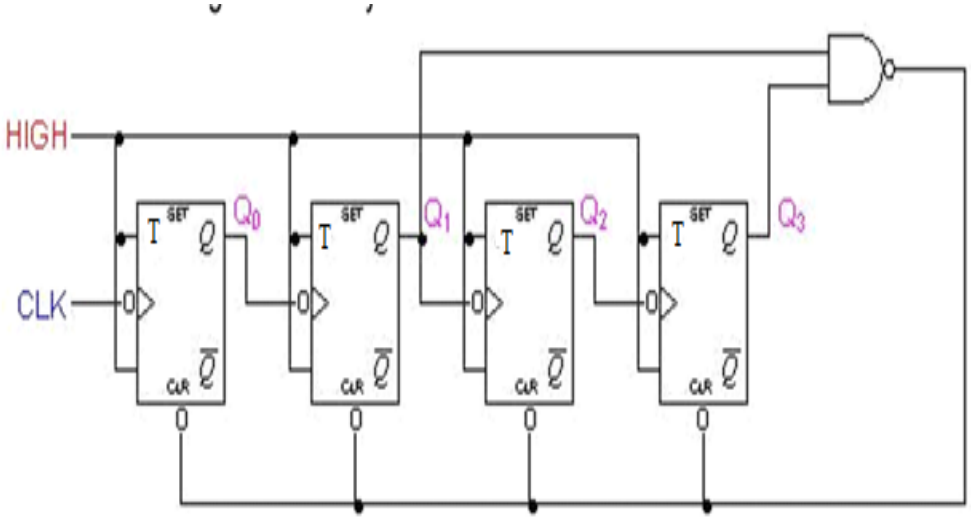
Groups

(0,2,4,6)	$\overline{A}.\overline{D}$
(0,2,8,10)	$\overline{B}.\overline{D}$
(0,4,8,12)	$\overline{C}.\overline{D}$

y = A'D' + B'D' + C'D'



37. Design and implement a mod-10 asynchronous counter using T flip flops and explain its working

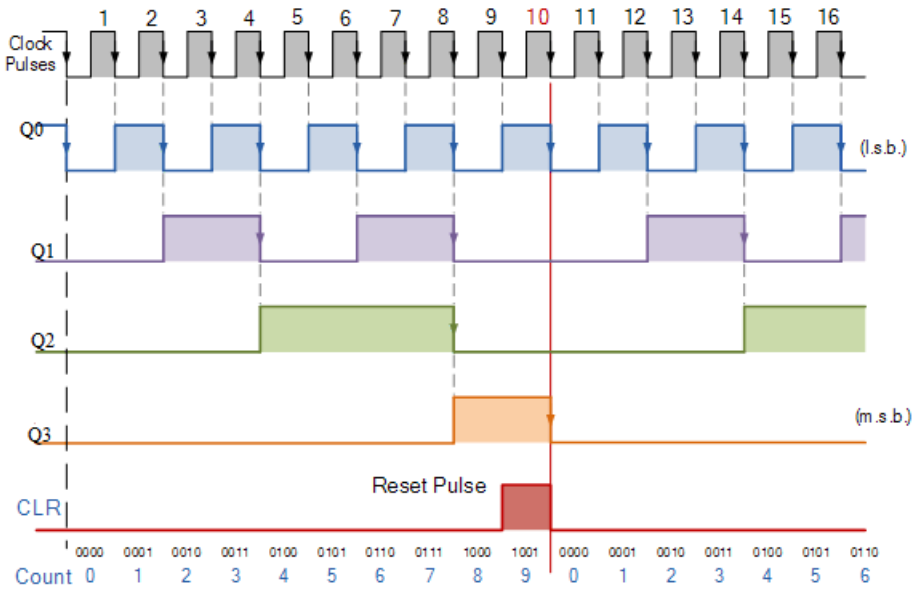


- **The sequence of the decade counter is shown in the table below:**

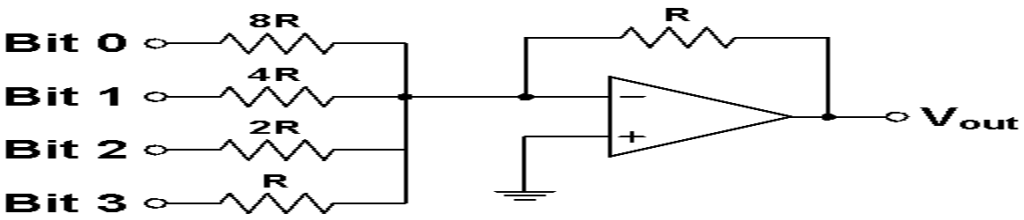
Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

- **Once the counter counts to ten (1010), all the flip-flops are being cleared. Notice that only Q1 and Q3 are used to decode the count of ten. This is called partial decoding, as none of the other states (zero to nine) have both Q1 and Q3 HIGH at the same time.**

Decade Counter Timing Diagram



38. Explain 4-bit DAC with neat block diagram



The resistor with the lowest value R corresponds to the highest weighted binary input Bit 3 (MSB) [2^3 = 8], and 2R, 4R, 8R correspond to the binary weights of Bit 2 (2^2 = 4), Bit 1 (2^1 = 2), and Bit 0 (LSB) [2^0 = 1] respectively. The relationship between the digital inputs (Bit 0 to Bit 3) and the analog output VOUT is as follow:

$$V_{OUT} = -V_{ref}\left(\frac{1}{1}Bit3 + \frac{1}{2}Bit2 + \frac{1}{4}Bit1 + \frac{1}{8}Bit0\right)$$

39. Explain the technique of error detection and correction using humming codes with example/Write notes on error correction codes

Hamming code is useful for both detection and correction of error present in the received data. This code uses multiple parity bits and we have to place these parity bits in the positions of powers of 2.

the required number of parity bits.

$$40.\ 2^p \geq m+p+1$$

Where,

'm' is the number of bits in the binary code

'p' is the number of parity bits

Therefore, the number of bits in the Hamming code is equal to m + p.

Error Detection & Correcting Code: Hamming Code

- **Locating Parity Bits**: Parity bits, interspersed in the code word along with data bits, are located at positions *2^0, 2^1 and 2^2 from left* of the message block and are as shown below:

  P_1 P_2 D_3 P_4 D_5 D_6 D_7 Where "**D**" bits are data bits and "**P**"

- **Setting Values of Parity bits**: Parity bits calculated in following manner:-

  P_1 : set to 0 or 1 to create even parity for bits 1, 3, 5 & 7 (P_1 D_3 D_5 D_7)
  P_2 : set to 0 or 1 to create even parity for bits 2, 3, 6 & 7 (P_2 D_3 D_6 D_7)
  P_4 : set to 0 or 1 to create even parity for bits 4, 5, 6 & 7 (P_4 D_5 D_6 D_7)

- **Error Detection at Receiver**: Received hamming code ($P_1P_2D_3P_4D_5D_6D_7$) is decoded to detect and correct any error. Bits *1,3,5,7*, bits *2,3,6,7* and bits *4,5,6,7* are all checked for *even parity*.

- If there is an error, location of error bit can be found by forming a *3-bit binary number $c_3c_2c_1$* where c_3= *XORing* of bits 4, 5, 6 & 7
  c_2= *XORing* of bits 2, 3, 6 & 7
  c_1= *XORing* of bits 1, 3, 5 & 7

Example :

Error Detection & Correcting Code
Hamming Code

Example: Received 7-bit Hamming Code → 1 1 1 1 0 0 1
P_1 P_2 D_3 P_4 D_5 D_6 D_7

Step-1: Analyze bits 1, 3, 5, 7 (1 1 0 1) of which bit-1 is parity bit P_1.
Error is present as number of 1's is odd.
Derive c_1 by XORing (1 1 0 1) = 1

Step-2: Analyze bits 2, 3, 6, 7 (1 1 0 1) of which bit-1 is parity bit P_2.
Error is present as number of 1's is odd.
Derive c_2 by XORing (1 1 0 1) = 1

Step-3: Analyze bits 4, 5, 6, 7 (1 0 0 1) of which bit-1 is parity bit P_4.
Error is not present as number of 1's is even.
Derive c_3 by XORing (1 0 0 1) = 0

Step-4: Locating Bit Error Position:
Error word is $c_3c_2c_1$ = 011 →
Decimal equivalent is 3. *Thus bit 3 (from left) is in error.*

Step-5: Locating Error Bit & Correcting: By taking complement of bit 3, we get the corrected code, which is

1 1 **0** 1 0 0 1
P_1 P_2 D_3 P_4 D_5 D_6 D_7

41. Decode the message"1001001" coded in 7 bit Hamming code ,assuming that at most a single error occurred in the code

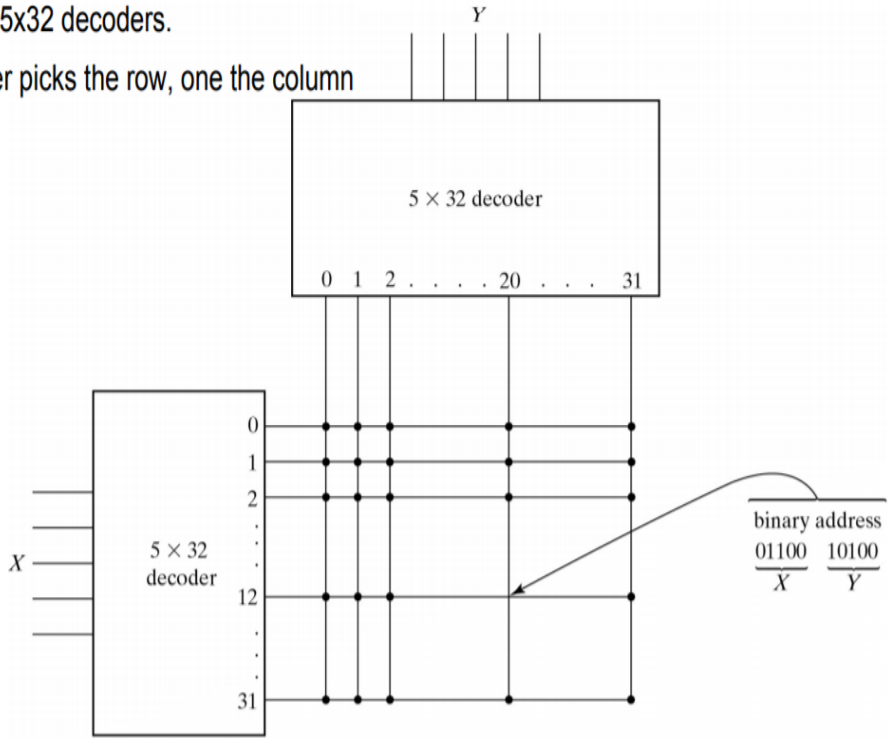
Received message is 1001001
Analyze 1001001
(Even Parity Considered)
Bits 1,3,5,7 (1001) → no error → c1= 0
Bits 2,3,5,7 (0001) → error → c2= 1
Bits 4,5,6,7 (1001) → no error → c3= 0

Error word is c3c2c1=010 = decimal 2
Corrected bit pattern is obtained by complementing 2nd bit → 1 1 01001

42. Draw and explain two dimensional decoding structure for a 1 K-memory

Instead of using a single 10 x 1024 decoder
we use two 5x32 decoders.

One decoder picks the row, one the column



ts per gate. n
ber of inputs
we use two

ments

Theorem 1: Complement of the sum is equal to product of complements

44. Describe and prove Demorgan's theorem

Theorem 1: The compliment of the product of two variables is equal to the sum of the compliment of each variable.

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

NAND = Bubbled OR

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

Theorem 2: The compliment of the sum of two variables is equal to the product of the compliment of each variable.

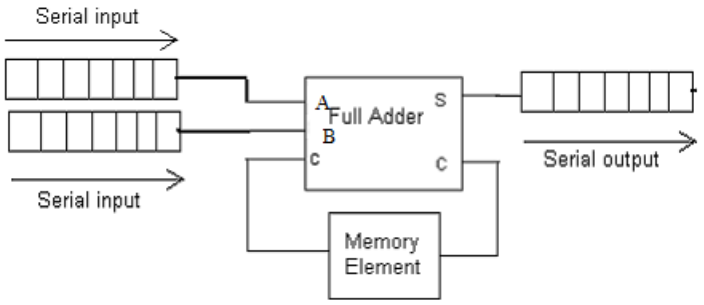
$\overline{A+B} = \overline{A}.\overline{B}$

NOR = Bubbled AND

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

45. Illustrate the working of a serial adder

In serial adders, pairs of bits are added simultaneously during each clock cycle. Two right-shift registers are used to hold the numbers (*A* and *B*) to be added, while one left-shift register is used to hold the sum (*S*).



A	*B*	*S*	s_i	c_{i+1}
1011	0011	0000	0	1
0101	0001	1000	1	1
0010	0000	1100	1	0
0001	0000	1110	1	0

46. Explain different types of shift registers with data shifting diagrams

We know that one flip-flop can store one-bit of information. In order to store multiple bits of information, we require multiple flip-flops. The group of flip-flops, which are used to hold (store) the binary data is known as **register**.

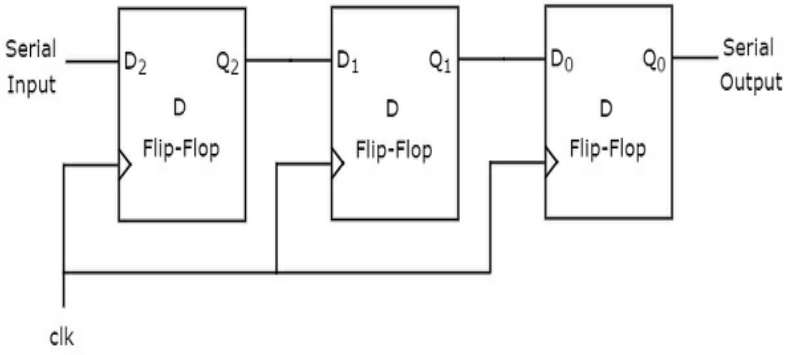
If the register is capable of shifting bits either towards right hand side or towards left hand side is known as **shift register**. An 'N' bit shift register contains 'N' flip-flops. Following are the four types of shift registers based on applying inputs and accessing of outputs.

- Serial In - Serial Out shift register
- Serial In - Parallel Out shift register

- Parallel In - Serial Out shift register
- Parallel In - Parallel Out shift register

E. 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. The **block diagram** of 3-bit SISO shift register is shown in the following figure.



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, we can send the bits serially 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, we can receive the bits serially from the output of right most D flip-flop. Hence, this output is also called as **serial output**.

Example

Let us see the working of 3-bit SISO shift register by sending the binary information **"011"** from LSB to MSB serially at the input.

Assume, initial status of the D flip-flops from leftmost to rightmost is Q2Q1Q0=000. We can understand the **working of 3-bit SISO shift register** from the following table.

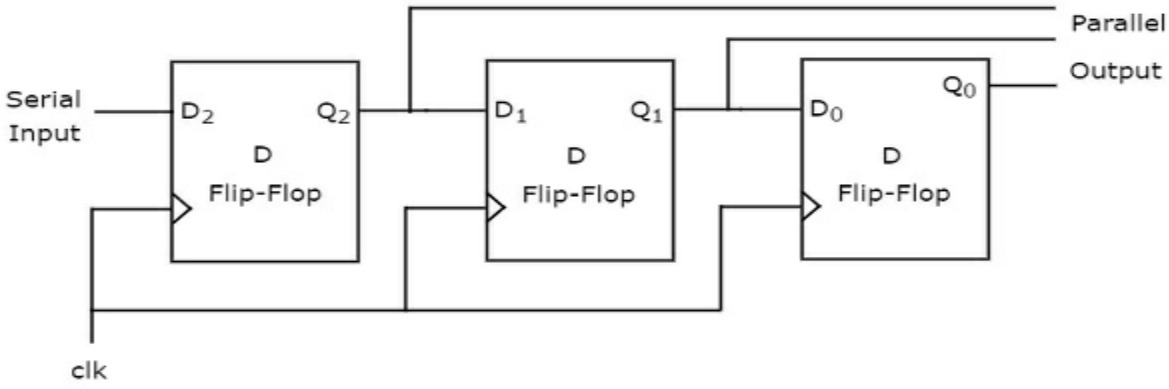
No of positive edge of Clock	Serial Input	Q_2	Q_1	Q_0
0	-	0	0	0
1	1(LSB)	1	0	0

2	1	1	1	0
3	0(MSB)	0	1	1(LSB)
4	-	-	0	1
5	-	-	-	0(MSB)

The initial status of the D flip-flops in the absence of clock signal is Q2Q1Q0=000. Here, the serial output is coming from Q0. So, the LSB (1) is received at 3rd positive edge of clock and the MSB (0) is received at 5thpositive edge of clock. Therefore, the 3-bit SISO shift register requires five clock pulses in order to produce the valid output. Similarly, the **N-bit SISO shift register** requires **2N-1** clock pulses in order to shift 'N' bit information.

F. 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. The **block diagram** of 3-bit SIPO shift register is shown in the following figure.



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, we can send the bits serially 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, we can access the

outputs of each D flip-flop in parallel. So, we will get **parallel outputs** from this shift register.

Example

Let us see the working of 3-bit SIPO shift register by sending the binary information **"011"** from LSB to MSB serially at the input.

Assume, initial status of the D flip-flops from leftmost to rightmost is Q2Q1Q0=000. Here, Q2 & Q0 are MSB & LSB respectively. We can understand the **working of 3-bit SIPO shift register** from the following table.

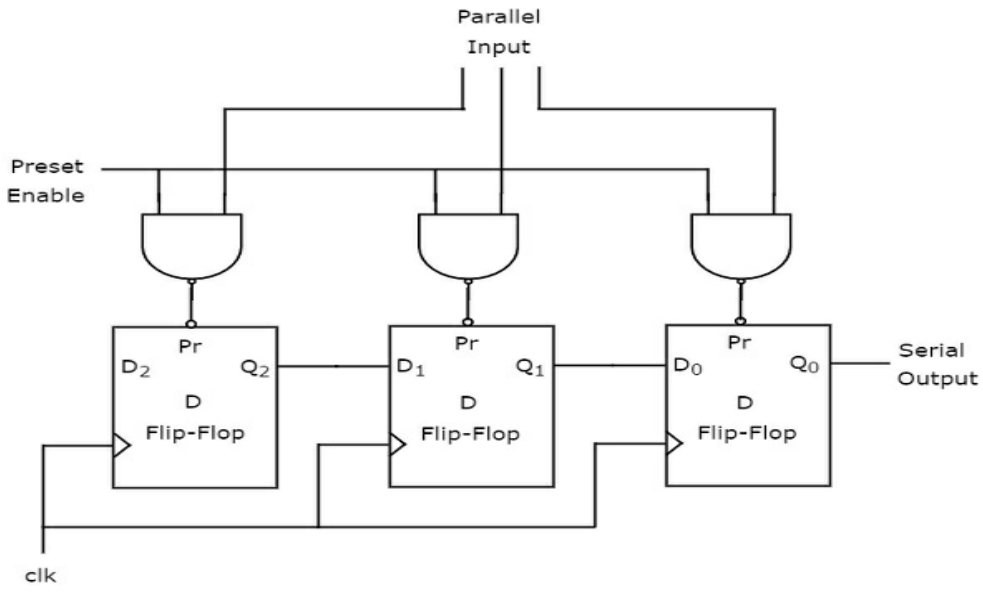
No of positive edge of Clock	Serial Input	Q_2(MSB)	Q_1	Q_0(LSB)
0	-	0	0	0
1	1(LSB)	1	0	0
2	1	1	1	0
3	0(MSB)	0	1	1

The initial status of the D flip-flops in the absence of clock signal is Q2Q1Q0=000. The binary information **"011"** is obtained in parallel at the outputs of D flip-flops for third positive edge of clock.

So, the 3-bit SIPO shift register requires three clock pulses in order to produce the valid output. Similarly, the **N-bit SIPO shift register** requires **N** clock pulses in order to shift 'N' bit information.

G. 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. The **block diagram** of 3-bit PISO shift register is shown in the following figure.



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, we can apply the **parallel inputs** to each D flip-flop by making Preset Enable to 1. For every positive edge triggering of clock signal, the data shifts from one stage to the next. So, we will get the **serial output** from the right most D flip-flop.

Example

Let us see the working of 3-bit PISO shift register by applying the binary information **"011"** in parallel through preset inputs.

Since the preset inputs are applied before positive edge of Clock, the initial status of the D flip-flops from leftmost to rightmost will be Q2Q1Q0=011. We can understand the **working of 3-bit PISO shift register** from the following table.

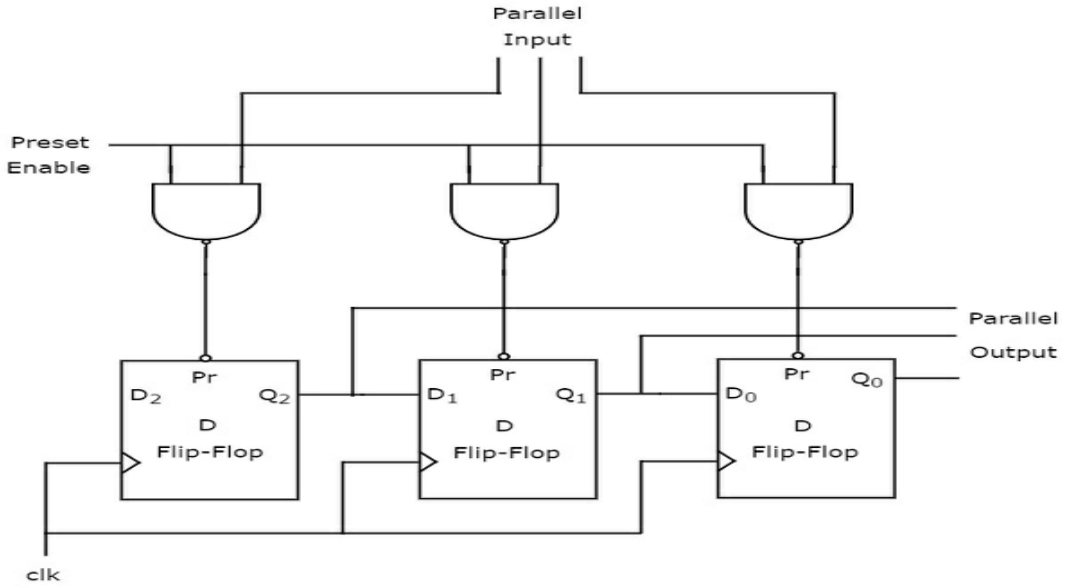
No of positive edge of Clock	Q_2	Q_1	Q_0
0	0	1	1(LSB)
1	-	0	1
2	-	-	0(LSB)

Here, the serial output is coming from Q0. So, the LSB (1) is received before applying

positive edge of clock and the MSB (0) is received at 2nd positive edge of clock.
Therefore, the 3-bit PISO shift register requires two clock pulses in order to produce the valid output. Similarly, the **N-bit PISO shift register** requires **N-1** clock pulses in order to shift 'N' bit information.

H. 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 **block diagram**of 3-bit PIPO shift register is shown in the following figure



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, we can apply the **parallel inputs** to each D flip-flop by making Preset Enable to 1. We can apply the parallel inputs through preset or clear. These two are asynchronous inputs. That means, the flip-flops produce the corresponding outputs, based on the values of asynchronous inputs. In this case, the effect of outputs is independent of clock transition. So, we will get the **parallel outputs** from each D flip-flop.

Example:Let us see the working of 3-bit PIPO shift register by applying the binary information **"011"** in parallel through preset inputs.

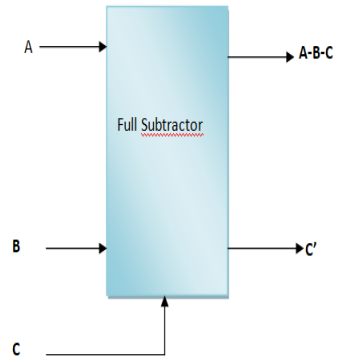
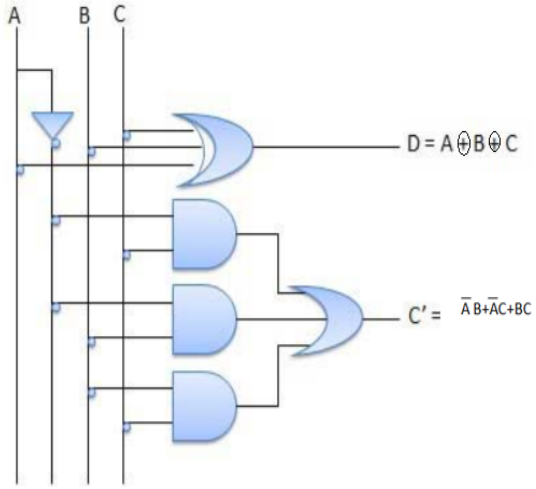
Since the preset inputs are applied before positive edge of Clock, the initial status of the D flip-flops from leftmost to rightmost will be Q2Q1Q0=011. So, the binary information **"011"** is obtained in parallel at the outputs of D flip-flops before applying positive edge of clock.
Therefore, the 3-bit PIPO shift register requires zero clock pulses in order to produce the valid output. Similarly, the **N-bit PIPO shift register** doesn't require any clock pulse

in order to shift ‘N’ bit information.

47. Design a full subtractor

a. Full Subtractors

The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

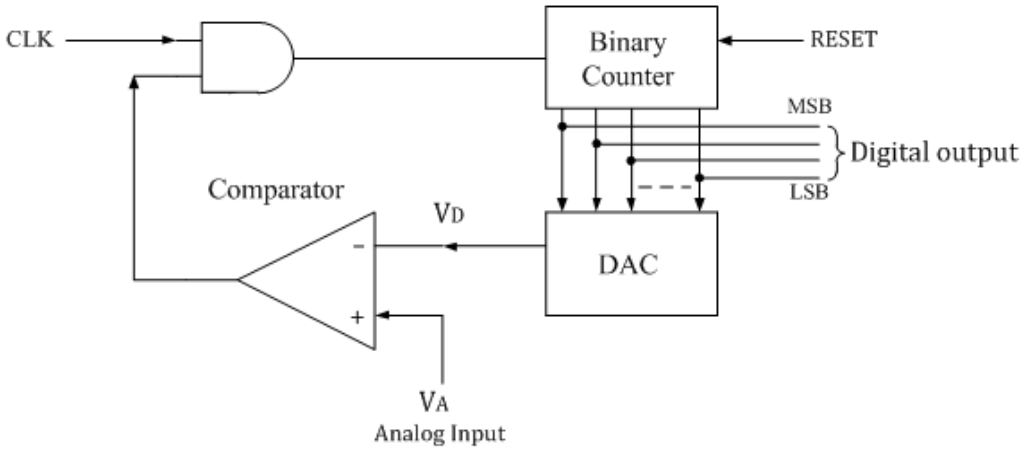
Block diagram	Truth Table	Circuit Diagram
	V	

Inputs			Output	
A	B	C	(A-B-C)	C'
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

48. Explain Counter type ADC

The counter type ADC is constructed using a binary counter, DAC and a comparator. The output voltage of a DAC is VD which is equivalent to corresponding digital input to DAC.

The following figure shows the n-bit counter type ADC.



Operation:

The n-bit binary counter is initially set to 0 by using reset command. Therefore the digital output is zero and the equivalent voltage VD is also 0V.

When the reset command is removed, the clock pulses are allowed to go through AND gate and are counted by the binary counter.

The D to A converter (DAC) converts the digital output to an analog voltage and applied as the inverting input to the comparator. The output of the comparator enables the AND gate to pass the clock.

The number of clock pulses increases with time and the analog input voltage VD is a rising staircase waveform as shown in figure below.

The counting will continue until the DAC output VD, equals and just rises more than unknown analog input voltage VA. Then the comparator output becomes low and this disables the AND gate from passing the clock.

The counting stops at the instance VA< VD, and at that instant the counter stops its progress and the conversion is said to be complete.

49. Design a gray to binary decoder

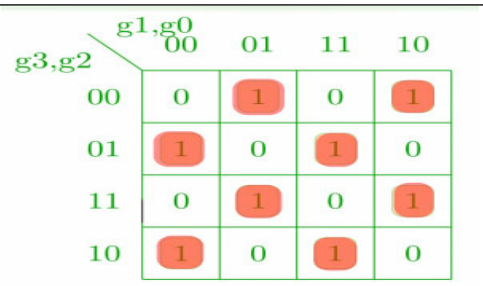
Let b0,b1,b2,and b3 be the bits representing the binary numbers, where b0 is the LSB and b3 is the MSB, and Let g0,g1,g2,and g3 be the bits representing the gray code of the binary numbers, where g0 is the LSB and g3 is the MSB.

Truth table-

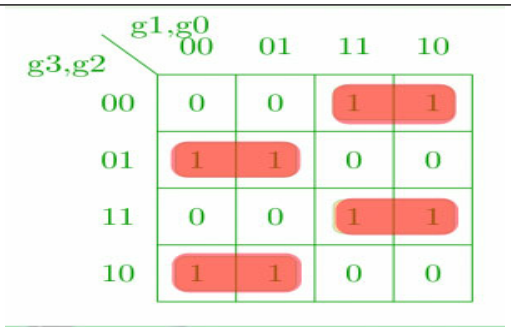
Gray Code				Binary			
g_3	g_2	g_1	g_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

K Map simplification

K-map for b0–



K-map for b1–



K-map for b2–

K-map for b3–

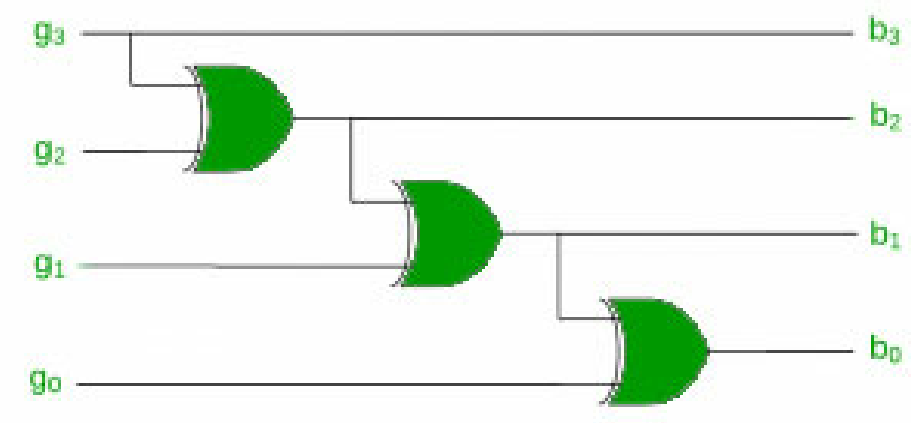
g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Corresponding Boolean expressions –

$$
\begin{aligned}
b_0 &= g_3'g_2'g_1'g_0 + g_3'g_2'g_1g_0' + g_3'g_2g_1'g_0' + g_3'g_2g_1g_0 + g_3g_2'g_1'g_0' + g_3g_2'g_1g_0 \\
&\quad + g_3g_2g_1'g_0 + g_3g_2g_1g_0' \\
&= g_3'g_2'(g_1'g_0 + g_1g_0') + g_3'g_2(g_1'g_0' + g_1g_0) + g_3g_2'(g_1'g_0' + g_1g_0) \\
&\quad + g_3g_2(g_1'g_0 + g_1g_0') \\
&= g_3'g_2'(g_0 \oplus g_1) + g_3'g_2(g_0 \odot g_1) + g_3g_2'(g_0 \odot g_1) + g_3g_2(g_0 \oplus g_1) \\
&= (g_0 \oplus g_1)(g_2 \odot g_3) + (g_0 \odot g_1)(g_2 \oplus g_3) \\
&= g_3 \oplus g_2 \oplus g_1 \oplus g_0 \\
b_1 &= g_3'g_2'g_1 + g_3'g_2g_1' + g_3g_2g_1 + g_3g_2'g_1' \\
&= g_3'(g_2'g_1 + g_2g_1') + g_3(g_2g_1 + g_2'g_1') \\
&= g_3'(g_2 \oplus g_1) + g_3(g_2 \odot g_1) \\
&= g_3 \oplus g_2 \oplus g_1 \\
b_2 &= g_3'g_2 + g_3g_2' \\
&= g_3 \oplus g_2 \\
b_3 &= g_3
\end{aligned}
$$

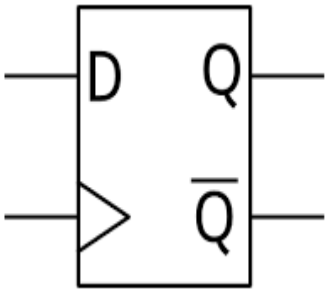
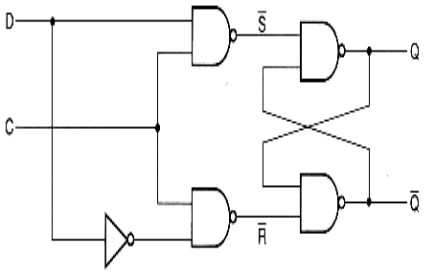
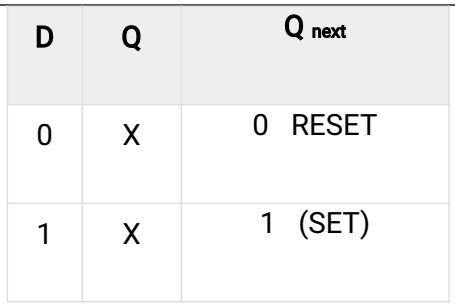
Circuit diagram



50. Construct a D latch using a gated SR latch

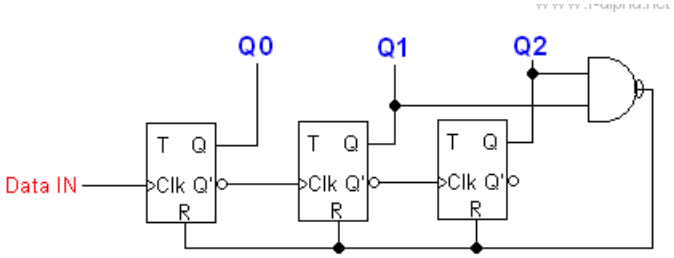
There is one drawback of SR Latch. That is the next state value can't be predicted

when both the inputs S & R are one. So, we can overcome this difficulty by D Latch. It is also called as Data Latch. This circuit has single input D and two outputs Q & Q'. D Latch is obtained from SR Latch by placing an inverter between S & R inputs and connects D input to S. That means we eliminated the combinations of S & R are of same value.

Logic Symbol	Cicuit diagram	Truth table
		

D	Q	Q next
0	X	0 RESET
1	X	1 (SET)

51. Construct a Mod 6 counter using T flip flop

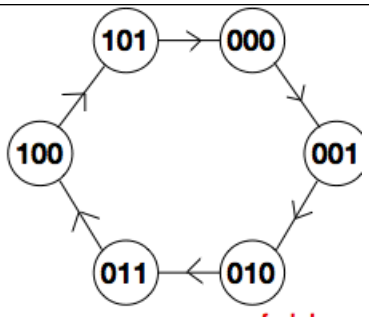


In a **mod-6** counter you are using a **3-bit** counter that actually has 8 states. However, when the counter reaches the seventh state (**Q2 = 1**, **Q1 = 1**, **Q0 = 0**), you force the counter to return to the state (**Q2 = 0**, **Q1 = 0**, **Q0 = 0**)...

For this you use an N**AND** gate, whose output is connected with the **RESET** inputs of the flip-flops...

Truth table

Clock pulses			



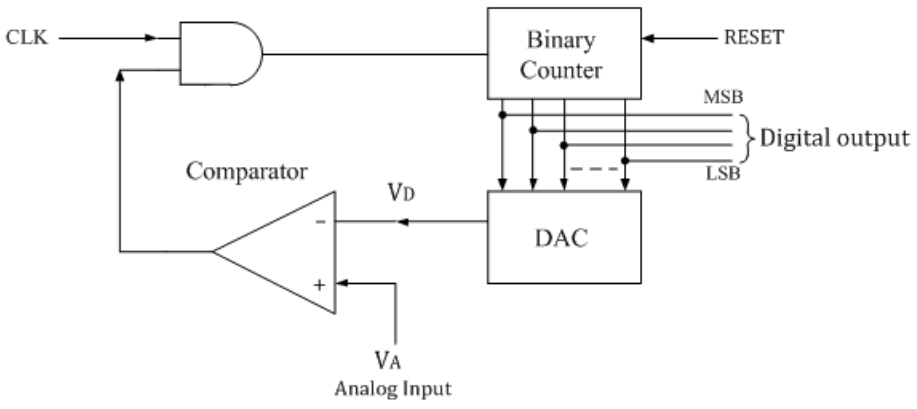
Truth table

Clock pulses	Q2	Q1	Q0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	0	0	0

52. Compare Synchronous and Asynchronous counter

Asynchronous/Ripple Counter	**Synchronous Counter**
• Flip flops are connected in such a way that the o/p of first flip-flop drives the clock of next flip-flop.	• There is no connection between o/p of first flip-flop and clock of next flip-flop.
• Flip-flops are *not clocked* simultaneously.	• Flip-flops are *clocked* simultaneously.
• Circuit is *simple* for more number of states.	• Circuit becomes *complicated* as number of states increases.
• Speed is *slow* as clock is propagated through number of stages.	• Speed is *high* as clock is given at a same time.
• Asynchronous Counter are also known as "Ripple Counter" because of the way the clock pulses or ripples, its way through the flip-flop.	• There is no connection between o/p of first flip-flop and clock of next flip-flop.

53. Explain counter type ADC



Operation:

The n-bit binary counter is initially set to 0 by using reset command. Therefore the digital output is zero and the equivalent voltage VD is also 0V.

When the reset command is removed, the clock pulses are allowed to go through AND gate and are counted by the binary counter.

The D to A converter (DAC) converts the digital output to an analog voltage and applied as the inverting input to the comparator. The output of the comparator enables the AND gate to pass the clock.

The number of clock pulses increases with time and the analog input voltage VD is a rising staircase waveform as shown in figure below.

The counting will continue until the DAC output VD, equals and just rises more than unknown analog input voltage VA. Then the comparator output becomes low and this disables the AND gate from passing the clock.

The counting stops at the instance VA< VD, and at that instant the counter stops its progress and the conversion is said to be complete.

54. State DAC Parameters

- Accuracy :-Accuracy indicates how close the measured value is to the true value. There are many ways of specifying accuracy .The most common are full scale error and linearity error.
    - Full scale error :-It is the maximum deviation of the output value from its expected value expressed in percentage of full scale
    - Linearity error :-It is the maximum deviation in step size from the ideal step size
- Offset voltage:-The output of a DAC will be zero volt when all the binary inputs are 0's.Howeever there will be a small output voltage called offset voltage or offset error
- Monotonicity :-A DAC is monotonic if its output value increases as the binary inputs are incremented from one value to the next
- Resolution(Step size)
    - Resolution of DAC is defined as the smallest change that can occur in the analog output as a result of a change in the digital input The resolution is

always equal to the weight of the LSB and is also known as step size

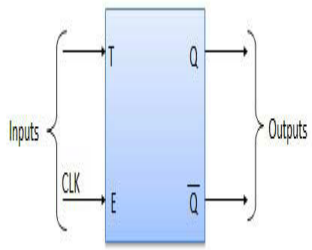
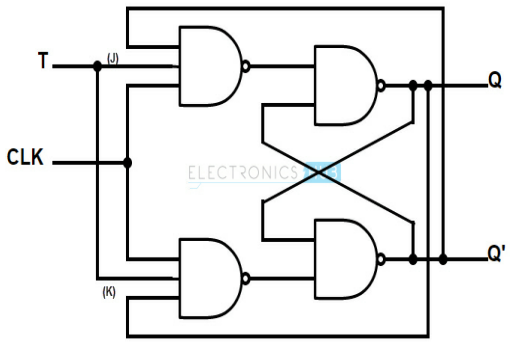
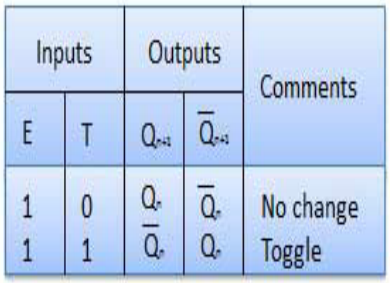
- Percentage of resolution =step size/full scale *100

- Settling time :- The time required for the output of the DAC to settle to within +(1/2) LSB of the final value for a given digital input is known as settling time .
- Speed
- Rate of conversion of a single digital input to its analog equivalent
- Conversion rate depends on
  - clock speed of input signal
  - settling time of converter
  - When the input changes rapidly, the DAC conversion speed must be high
- Linearity

  The difference between the desired analog output and the actual output over the full range of expected values

55. Demonstrate a T Flip flop using a J K Flip flop with Truth table

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together.

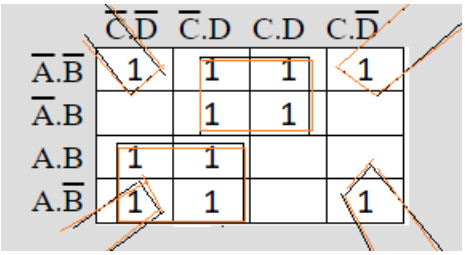
1. Symbol Diagram	2. Block Diagram	3. Truth Table
		

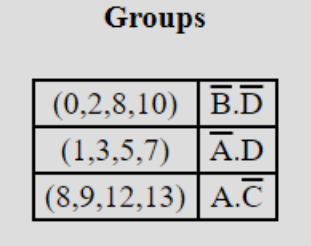
Inputs		Outputs		Comments
E	T	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	Q_n	$\overline{Q}_n$	No change
1	1	$\overline{Q}_n$	Q_n	Toggle

Operation

S.N.	Condition	Operation
1	**T = 0, J = K = 0**	The output Q and Q'won't change
2	**T = 1, J = K = 1**	Output will toggle corresponding to every leading edge of clock signal.

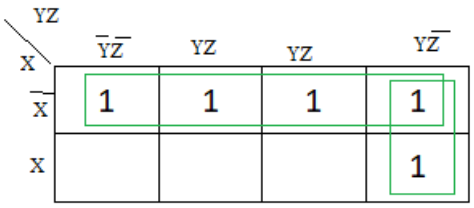
56. Reduce the expression f=∑m(0,1,2,3,5,7,8,9,10,12,13) using K map





y = B'D' + A'D + AC'

57. Reduce the expression F(x,y,z)=∑(0,1,2,3,6)



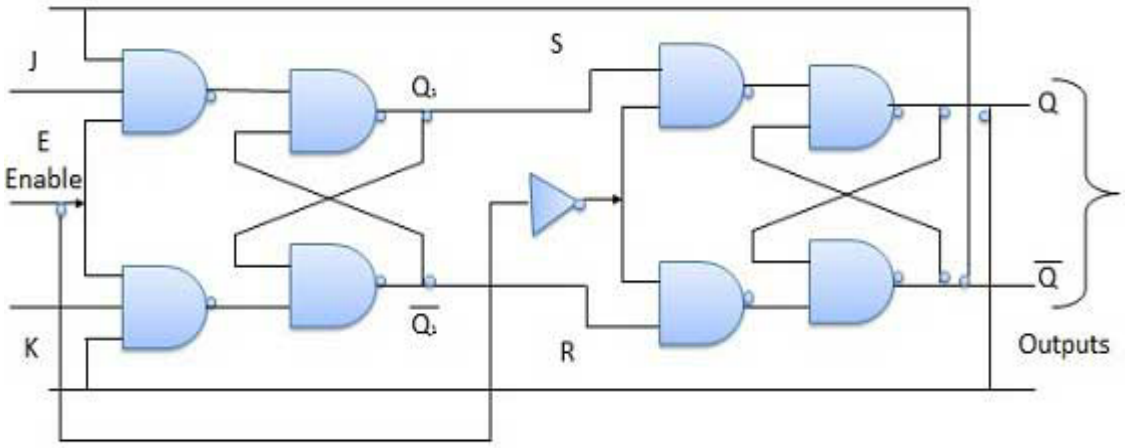
$\overline{X} + Y\overline{Z}$

58. Explain the working of JK Master slave flip flop with logic diagram

Master slave JK FF is a cascade of two S-R FF with feedback from the output of

second to input of first. Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive.

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

Operation

S. N.	Condition	Operation

1	**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.
2	**J = 0 and K = 1 (Reset)**	Clock = 1 Master active, slave inactive. Therefore outputs of the master become Q_1 = 0 and Q_1' = 1. That means S = 0 and R =1. Clock = 0 Slave active, master inactive. Therefore outputs of the slave become Q = 0 and Q'= 1. Again clock = 1 Master active, slave inactive. Therefore even with the changed outputs Q = 0 and Q'= 1 fed back to master, its output will be Q1 = 0 and Q1'= 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'= 1. Thus we get a stable output from the Master slave.
3	**J = 1 and K = 0 (Set)**	Clock = 1 Master active, slave inactive. Therefore outputs of the master become Q_1 = 1 and Q_1' = 0. That means S = 1 and R =0. Clock = 0 Slave active, master inactive. Therefore outputs of the slave become Q = 1 and Q '= 0. Again clock = 1 then it can be shown that the outputs of the slave are stabilized to Q = 1 and Q' = 0.
4	**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.