MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CS3362 C PROGRAMMING AND DATA STRUCTURES LABORATORY

Semester - 03

LABORATORY MANUAL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

1. Practice of C programming using statements, expressions, decision making and iterative statements
2. Practice of C programming using Functions and Arrays
3. Implement C programs using Pointers and Structures
4. Implement C programs using Files
5. Development of real time C applications
6. Array implementation of List ADT
7. Array implementation of Stack and Queue ADTs
8. Linked list implementation of List, Stack and Queue ADTs
9. Applications of List, Stack and Queue ADTs
10. Implementation of Binary Trees and operations of Binary Trees
11. Implementation of Binary Search Trees
12. Implementation of searching techniques
13. Implementation of Sorting algorithms : Insertion Sort, Quick Sort, Merge Sort
14. Implementation of Hashing – any two collision techniques

Ex no:1a **PROGRAM USING I/O STATEMENTS AND EXPRESSIONS**

AIM:

To write a 'C' program to find the area and perimeter of a circle.

ALGORITHM:

Step-1: Start the program.

Step-2: Read radius.

Step-3: Area →PI*radius*radius. (PI=3.1)

Step-4: Perimeter→ 2*PI*radius.

Step-5: Print the area and perimeter.

Step-6: Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define PI 3.14f
void main()
{
float rad, area,perm;
clrscr();
printf("\n Enter the radius of a circle:");
scanf("%f",&rad);
area=PI*rad*rad;
perm=2*PI*rad;
printf("\n Area of a circle: %f",area);
printf("\n Perimeter of a circle: %f",perm);
```

```
getch();
}
```

OUTPUT:

```
Enter the radius of a circle: 2.34
Area of a circle: 17.193384
Perimeter of a circle: 14.695200
```

Result:

Thus, the ‘C’ program for finding the area and perimeter of a circle has been executed successfully.

Ex no:1b. PROGRAM USING DECISION MAKING STATEMENTS

AIM:

To write a 'C' program to find the largest number among three numbers.

ALGORITHM:

Step-1: Start the program.

Step-2: Read three numbers (a, b, & c).

Step-3: If a is greater than b & c, then print "a is greater".

Step-4: ElseIf b is greater than a & b, then print "b is greater".

Step-5: Else print "c is greater".

Step-6: EndIf.

Step-7: Stop the program

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a, b, c;
clrscr();
printf("\n Enter three numbers:");
scanf("%d %d %d",&a,&b,&c);
if((a>b)&&(a>c))
{
printf("\n %d is greater",a);
}
else if((b>a)&&(b>c))
{
printf("\n %d is greater",b);
}
else
{
printf("\n %d is greater",c);
}
getch();
}
```

Output:

```
Enter three numbers: 10 30 20
30 is greater.
```

RESULT:

Thus, the 'C' program for finding the largest number among three numbers has been executed successfully.

Ex no:2 **PROGRAM USING TWO-DIMENSIONAL ARRAY**

AIM:

To write a 'C' program to populate a two dimensional array with height and weight ofpersons and compute the body mass index of the individuals.

ALGORITHM:

Step-1: Start the program.

Step-2: Read the weight, height.

Step-3: Calculate the BMI=weight*weight.

Step-4: Print the BMI.

Step-5: Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int weight[5][5], n, i, j, bmi;
clrscr();
printf("\n Enter the upper limit:");
scanf("%d",&n);
printf("\n Enter Weight and Height:");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&weight[i][j]);
```

```
}
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",weight[i][j]);
bmi=weight[i][j]*weight[i][j];
printf("\n BMI:%d",bmi);
}
}
getch();
}
```

OUTPUT:

```
Enter the upper limit: 2
Enter Weight and Height:
53    5.5
65    5.5
53    5.5
65    5.5
BMI: 18.859083
BMI: 23.129063
```

RESULT:

Thus, the 'C' program to populate a two dimensional array with height and weight ofpersons and compute the body mass index of the individuals has been executed successfully

Ex no:3 **PROGRAM USING STUCTURES AND POINTERS**

AIM:

To write a 'C' program to generate salary slip of employees using structures and pointers.

ALGORITHM:

Step-1: Start the program.

Step-2: Read the employees details e1, empid, name, basic, hra, da, ma, pf, insurance, gross,and net.

Step-3: e1=(struct employee*)malloc(sizeof(struct employee)*num).

Step-4: Print input for every employee.

Step-5: For(i=0;i<num;i++).

Step-6: Calculate gross amount.

Step-7: Calculate net amount.

Step-8: End For.

Step-9: Stop the program.

PROGRAM:

```
#include <stdio.h>
   struct p
   {
      int x;
      char y;
   };
   int main()
   {
            struct p p1[] = {1, 92, 3, 94, 5, 96};
            struct p *ptr1 = p1;
            int x = (sizeof(p1) / 3);
```

```
        if (x == sizeof(int) + sizeof(char))
            printf("%d\n", ptr1->x);
        else
            printf("Falsen");
}
```

OUTPUT:

False n

RESULT:

Thus the program completed and executed successfully

Ex no:4 **Implement C programs using Files**

AIM:

To write a 'C' program to Implement C programs using Files

```
#include <stdio.h>
  int main()
{
  FILE *fptr;
  char name[50];
  int marks, i, num;

  printf("Enter number of students: ");
  scanf("%d", &num);


  fptr = (fopen("C:\\student.txt", "w"));
  if(fptr == NULL)
  {
     printf("Error!");
     exit(1);
  }
 for(i = 0; i < num; ++i)
  {
    printf("For student%d\nEnter name: ", i+1);
    scanf("%s", name);

    printf("Enter marks: ");
    scanf("%d", &marks);

    fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
  }
```

```
    fclose(fptr);
    return 0;
}
```

Output:

```
Enter number of students: 5

For student1
Enter name: manikandan
Enter marks: 76

For student2
Enter name: kameshwaren
Enter marks: 100

For student3
Enter name: ajay raja
Enter marks: 71

For student4
Enter name: gopal
Enter marks: 65

For student5
Enter name: krishna
Enter marks: 89
```

RESULT:

Thus the program completed and executed successfully

Ex no:5 **DEVELOPMENT OF REAL TIME C APPLICATIONS**

AIM:

To write a 'C' program to generate salary slip of employees using structures and pointers.

ALGORITHM:

Step-1: Start the program.

Step-2: Read the employees details e1, empid, name, basic, hra, da, ma, pf, insurance, gross,and net.

Step-3: e1=(struct employee*)malloc(sizeof(struct

employee)*num).Step-4: Print input for every employee.

Step-5:

For(i=0;i<num;i++).

Step-6: Calculate gross

amount.Step-7: Calculate

net amount. Step-8: End

For.

Step-9: Stop the

program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct emp
{
   int empno ;
   char name[10] ;
   int bpay, allow, ded, npay ;
} e[10] ;
```

```
void main()
{
   int i, n ;
   printf("Enter the number of employees : ") ;
   scanf("%d", &n) ;
   for(i = 0 ; i < n ; i++)
   {
      printf("\nEnter the employee number : ") ;
      scanf("%d", &e[i].empno) ;
      printf("\nEnter the name : ") ;
      scanf("%s", e[i].name) ;
      printf("\nEnter the basic pay, allowances & deductions : ") ;
      scanf("%d %d %d", &e[i].bpay, &e[i].allow, &e[i].ded) ;
      e[i].npay = e[i].bpay + e[i].allow - e[i].ded ;
   }
   printf("\nEmp. No. Name \t Bpay \t Allow \t Ded \t Npay \n\n") ;
   for(i = 0 ; i < n ; i++)
   {
      printf("%d \t %s \t %d \t %d \t %d \t %d \n", e[i].empno,
      e[i].name, e[i].bpay, e[i].allow, e[i].ded, e[i].npay) ;
   }
   getch() ;
}
```

OUTPUT:

Enter the number of employees:

2 Enter your input for every

employee:

Employee ID: 9293

Employee Name: Ram

Basic Salary: 250000

HRA: 50000

DA: 4000

Insurance: 6000

Employee ID: 9282

Employee Name: Raj

Basic Salary: 150000

HRA: 25000
DA: 3000

Insurance: 2500

Enter Employee ID to get payslip:

9293Salary Slip of Ram:

Employee ID: 9293

Basic Salary: 250000

House Rent

Allowance: 5000

Dearness Allowance:

6000

Medical Allowance: 5000

Gross Salary: 23900982.00

RupeesDeductions:

Provident Fund: 4000

Insurance: 6000

Net Salary: 23890982.00 Rupees

Do you want to continue(1/0): 1

Enter Employee ID to get

payslip: 9382Salary Slip of Raj:

Employee ID: 9392

Basic Salary: 150000

House Rent Allowance:

25000Dearness

Allowance: 3000

Medical Allowance: 3500

Gross Salary: 4450328.00

RupeesDeductions:

Provident Fund: 3000

Insurance: 2500

Net Salary: 4444828.00 Rupees

Do you want to continue(1/0): 0

RESULT:

Thus, the 'C' program to generate salary slip of employees using structures and pointershas been executed successfully.

Ex. No: 6 **IMPLEMENTATION OF SINGLY LINKED LIST**

AIM:

To write the C program for implementing the linked list.

ALGORITHM:

Step 1: Define a C- struct for each node in the stack. Each node in the list contains data and linkto the next node.

Step 2: To insert the new node, the new pointer will be added as per the position.

Step 3: To delete the existing node in the list, first delete the pointer the particular node andpoints to the next node

Step 4: Display the list

PROGRAM:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>

void main()
{
   struct node
   {
      int num;
      struct node *ptr;
   };
   typedef struct node NODE;

   NODE *head, *first, *temp = 0;
   int count = 0;
   int choice = 1;
   first = 0;
```

```
        while (choice)
        {
           head  = (NODE *)malloc(sizeof(NODE));
           printf("Enter the data item\n");
           scanf("%d", &head-> num);
           if (first != 0)
           {
              temp->ptr = head;
              temp = head;
           }
           else
           {
              first = temp = head;
           }
           fflush(stdin);
           printf("Do you want to continue(Type 0 or 1)?\n");
           scanf("%d", &choice);

        }
        temp->ptr = 0;
        /*  reset temp to the beginning */
        temp = first;
        printf("\n status of the linked list is\n");
        while (temp != 0)
        {
           printf("%d=>", temp->num);
           count++;
           temp = temp -> ptr;
        }
        printf("NULL\n");
        printf("No. of nodes in the list = %d\n", count);
}
```

OUTPUT:

****Main Menu****

Choose one option from the

following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice? 1

Enter value 25

Node inserted

****Main Menu****

hoose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified

location

7.Find

8.Display

9.Exit

Enter your choice? 2

Enter value? 99

Node inserted

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 3

Enter element value55

Enter the location after which you

want to insert 25

can't insert

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 4

Node deleted from the begining ...

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 5

Only node of the list deleted ...

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 6

Enter the location of the node after

which you want to perform deletion

66

can't Delete

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 7

Empty List

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 8

Nothing to print

****Main Menu****

Choose one option from the

following list ...

1.Insert in begining

2.Insert at last

3.Insert in the middle

4.Delete from Beginning

5.Delete from last

6.Delete node after specified location

7.Find

8.Display

9.Exit

Enter your choice? 9

Exit.

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your
choice?1
Enter value

25
Node inserted

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your
choice?2
Ente
r
valu
e?
99
Node inserted

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your
choice?3
Enter element value55

Enter the location after which you want
to insert25
can't insert

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your
choice?4
Node deleted from the begining ...

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice?
5
Only node of the list deleted ...

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice?
6
Enter the location of the node after which you want to perform deletion
66
can't Delete

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice?
7
Empty List

****Main Menu****

Choose one option from the following list ...

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice?
8
Nothing to print

****Main Menu****

Choose one option from the following list ..

1. Insert in begining
2. Insert at last
3. Insert in the middle
4. Delete from Beginning
5. Delete from last
6. Delete node after specified location
7. Find
8. Display
9. Exit

Enter your choice?
9
Exit

RESULT:

Thus the program completed and executed successfully

Ex no:7a **ARRAY IMPLEMENTATION USING STACK ADT**

AIM:

To write the C program for stack using array implementation.

ALGORITHM:

Step1: Define a array which stores stack elements..

Step 2: The operations on the stack are
a. PUSH data into the stack
b. POP data out of stack

Step 3: PUSH DATA INTO STACK
a. Enter the data to be inserted into stack.
b. If TOP is NULL the input data is the first node in stack. The link of the node isNULL. TOP points to that node.
c. If TOP is NOT NULL the link of TOP points to the new node. TOP points to thatnode.

Step 4: POP DATA FROM STACK
a. If TOP is NULL the stack is empty.
b. If TOP is NOT NULL the link of TOP is the current TOP and the pervious TOP ispopped from stack.

Step 5: The stack represented by linked list is traversed to display its content.

PROGRAM:

```
#include <stdio.h>
#define MAXSIZE 5

struct stack
{
   int stk[MAXSIZE];
   int top;
};
typedef struct stack STACK;
STACK s;
```

```
void push(void);
int  pop(void);
void display(void);

void main ()
{
   int choice;
   int option = 1;
   s.top = -1;

   printf ("STACK OPERATION\n");
   while (option)
   {
      printf ("----------------------------------------\n");
      printf ("     1   -->   PUSH            \n");
      printf ("     2   -->   POP           \n");
      printf ("     3   -->   DISPLAY            \n");
      printf ("     4   -->   EXIT         \n");
      printf ("----------------------------------------\n");

      printf ("Enter your choice\n");
      scanf    ("%d", &choice);
      switch (choice)
      {
      case 1:
         push();
         break;
      case 2:
         pop();
         break;
      case 3:
         display();
         break;
      case 4:
         return;
      }
      fflush (stdin);
      printf ("Do you want to continue(Type 0 or 1)?\n");
      scanf    ("%d", &option);
   }
}
/*  Function to add an element to the stack */
void push ()
{
   int num;
   if (s.top == (MAXSIZE - 1))
```

```
    {
        printf ("Stack is Full\n");
        return;
    }
    else
    {
        printf ("Enter the element to be pushed\n");
        scanf ("%d", &num);
        s.top = s.top + 1;
        s.stk[s.top] = num;
    }
    return;
}
/*  Function to delete an element from the stack */
int pop ()
{
    int num;
    if (s.top == - 1)
    {
        printf ("Stack is Empty\n");
        return (s.top);
    }
    else
    {
        num = s.stk[s.top];
        printf ("poped element is = %dn", s.stk[s.top]);
        s.top = s.top - 1;
    }
    return(num);
}
/*  Function to display the status of the stack */
void display ()
{
    int i;
    if (s.top == -1)
    {
        printf ("Stack is empty\n");
        return;
    }
    else
    {
        printf ("\n The status of the stack is \n");
        for (i = s.top; i >= 0; i--)
        {
            printf ("%d\n", s.stk[i]);
        }
```

```
    }
    printf ("\n");
}
```

OUTPUT:

```
STACK OPERATION
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
34
Do you want to continue(Type 0 or 1)?
0
$ a.out
STACK OPERATION
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
34
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
2
poped element is = 34
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
```

```
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
3
Stack is empty
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
50
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
1
Enter the element to be pushed
60
Do you want to continue(Type 0 or 1)?
1
----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
----------------------------------------
Enter your choice
3

The status of the stack is
60
50
```

```
Do you want to continue(Type 0 or 1)?
1
-----------------------------------------
    1   -->   PUSH
    2   -->   POP
    3   -->   DISPLAY
    4   -->   EXIT
-----------------------------------------
Enter your choice

4
```

RESULT:

Thus the program completed and executed successfully

Ex no:7b **ARRAY IMPLEMENTATION USING QUEUE ADT**

AIM:

write a program for Queue using array implementation.

ALGORITHM:

Step1: Define a array which stores queue elements..

Step 2: The operations on the queue are

a. INSERT data into the queue
b. DELETE data out of queue

Step 3: INSERT DATA INTO queue

a. Enter the data to be inserted into queue.
b. If TOP is NULL the input data is the first node in queue. The link of the node isNULL. TOP points to that node.
c. If TOP is NOT NUL the link of TOP points to the new node. TOP points to thatnode.

Step 4: DELETE DATA FROM queue

a. If TOP is NULL the queue is empty .
b. If TOP is NOT NULL the link of TOP is the current TOP. The pervious TOP ispopped from queue.

Step 5: The queue represented by linked list is traversed to display its content.

PROGRAM:

```
#include <stdio.h>
#define MAX 50

void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
```

```
int front = - 1;
main()
{
   int choice;
   while (1)
   {
      printf("1.Insert element to queue \n");
      printf("2.Delete element from queue \n");
      printf("3.Display all elements of queue \n");
      printf("4.Quit \n");
      printf("Enter your choice : ");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         exit(1);
         default:
         printf("Wrong choice \n");
      } /* End of switch */
   } /* End of while */
} /* End of main() */

void insert()
{
   int add_item;
   if (rear == MAX - 1)
```

```
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
} /* End of insert() */

void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
```

```
            for (i = front; i <= rear; i++)
                printf("%d ", queue_array[i]);
            printf("\n");
        }
} /* End of display() */
```

OUTPUT:

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 10

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 15

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 20

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 30
```

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 10

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
15 20 30

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4

RESULT:

Thus the program completed and executed successfull

EX : 8(a) **IMPLEMENTATION OF STACK USING LINKED LIST**

AIM:

To write a program for stack using linked list implementation.

ALGORITHM:

Step1: Define a C-struct for each node in the stack. Each node in the stack contains data andlink to the next node. TOP pointer points to last node inserted in the stack.

Step 2: The operations on the stack are

a. PUSH data into the stack
b. POP data out of stack

Step 3: PUSH DATA INTO STACK

a. Enter the data to be inserted into stack.
b. If TOP is NULL the input data is the first node in stack. the link of the node isNULL. TOP points to that node.
c. If TOP is NOT NULL, the link of TOP points to the new node. TOP points to thatnode.

Step 4: POP DATA FROM STACK

a. If TOP is NULL the stack is empty.
b. If TOP is NOT NULL the link of TOP is the current TOP. The pervious TOP ispopped from stack.

Step 5: The stack represented by linked list is traversed to display its content.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int info;
   struct node *ptr;
}*top,*top1,*temp;

int topelement();
void push(int data);
```

```
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Push");
   printf("\n 2 - Pop");
   printf("\n 3 - Top");
   printf("\n 4 - Empty");
   printf("\n 5 - Exit");
   printf("\n 6 - Dipslay");
   printf("\n 7 - Stack Count");
   printf("\n 8 - Destroy stack");

   create();

   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);

      switch (ch)
      {
      case 1:
         printf("Enter data : ");
         scanf("%d", &no);
         push(no);
         break;
      case 2:
         pop();
```

```
            break;
        case 3:
            if (top == NULL)
                printf("No elements in stack");
            else
            {
                e = topelement();
                printf("\n Top element : %d", e);
            }
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            stack_count();
            break;
        case 8:
            destroy();
            break;
        default :
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create empty stack */
void create()
{
    top = NULL;
}

/* Count stack elements */
```

```
void stack_count()
{
   printf("\n No. of elements in stack : %d", count);
}

/* Push data into stack */
void push(int data)
{
   if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
   count++;
}

/* Display stack elements */
void display()
{
   top1 = top;

   if (top1 == NULL)
   {
      printf("Stack is empty");
      return;
   }

   while (top1 != NULL)
   {
      printf("%d ", top1->info);
```

```
        top1 = top1->ptr;
    }
 }

/* Pop Operation on stack */
void pop()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
    top = top1;
    count--;
}

/* Return top element */
int topelement()
{
    return(top->info);
}

/* Check if stack is empty or not */
void empty()
{
    if (top == NULL)
        printf("\n Stack is empty");
    else
        printf("\n Stack is not empty with %d elements", count);
}

/* Destroy entire stack */
```

```
void destroy()
{
   top1 = top;

   while (top1 != NULL)
   {
      top1 = top->ptr;
      free(top);
      top = top1;
      top1 = top1->ptr;
   }
   free(top1);
   top = NULL;

   printf("\n All stack elements destroyed");
   count = 0;
}
```

OUTPUT:

1 - Push

2 - Pop

3 - Top

4 - Empty

5 - Exit

6 - Dipslay

7 - Stack Count

8 - Destroy stack

Enter choice : 1

Enter data : 56

Enter choice : 1

Enter data : 80

Enter choice : 2

Popped value : 80

Enter choice : 3

Top element : 56

Enter choice : 1

Enter data : 78

Enter choice : 1

Enter data : 90

Enter choice : 6

90 78 56

Enter choice : 7

No. of elements in stack : 3

Enter choice : 8

All stack elements destroyed

Enter choice : 4

Stack is empty

Enter choice : 5

RESULT:

Thus the program completed and executed successfully.

EX.NO :8(b) **IMPLEMENTATION OF QUEUE USING LINKED LIST**

AIM:

To write a program for queue using linked list implementation.

ALGORITHM:

Step1: Define a C-struct for each node in the queue. Each node in the queue contains data and link to the next node. Front and rear pointer points to first and last node inserted in the queue.

Step 2: The operations on the queue are
a. INSERT data into the queue
b. DELETE data out of queue

Step 3: INSERT DATA INTO queue
a. Enter the data to be inserted into queue.
b. If TOP is NULL the input data is the first node in queue. the link of the node isNULL. TOP points to that node.
c. If TOP is NOT NULL the link of TOP points to the new node. TOP points to thatnode.

Step 4: DELETE DATA FROM queue .
a. If TOP is NULL the queue is empty
b. If TOP is NOT NULL.
c. The link of TOP is the current TOP and the pervious TOP is popped from queue.

Step 5: The queue represented by linked list is traversed to display its content.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Enque");
   printf("\n 2 - Deque");
   printf("\n 3 - Front element");
```

```
printf("\n 4 - Empty");
printf("\n 5 - Exit");
printf("\n 6 - Display");
printf("\n 7 - Queue size");
create();
while (1)
{
   printf("\n Enter choice : ");
   scanf("%d", &ch);
   switch (ch)
   {
   case 1:
      printf("Enter data : ");
      scanf("%d", &no);
      enq(no);
      break;
   case 2:
      deq();
      break;
   case 3:
      e = frontelement();
      if (e != 0)
         printf("Front element : %d", e);
      else
         printf("\n No front element in Queue as queue is empty");
      break;
```

```
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create an empty queue */
void create()
{
    front = rear = NULL;
}

/* Returns queue size */
void queuesize()
```

```
{
   printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
void enq(int data)
{
   if (rear == NULL)
   {
      rear = (struct node *)malloc(1*sizeof(struct node));
      rear->ptr = NULL;
      rear->info = data;
      front = rear;
   }
   else
   {
      temp=(struct node *)malloc(1*sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;

      rear = temp;
   }
   count++;
}
```

```
/* Displaying the queue elements */
void display()
{
   front1 = front;

   if ((front1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
      return;
   }
   while (front1 != rear)
   {
      printf("%d ", front1->info);
      front1 = front1->ptr;
   }
   if (front1 == rear)
      printf("%d", front1->info);
}

/* Dequeing the queue */
void deq()
{
   front1 = front;

   if (front1 == NULL)
   {
```

```
        printf("\n Error: Trying to display elements from empty queue");
        return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
        else
        {
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}

/* Returns the front element of queue */
int frontelement()
{
    if ((front != NULL) && (rear != NULL))
        return(front->info);
```

```
    else
        return 0;
}

/* Display if queue is empty or not */
void empty()
{
     if ((front == NULL) && (rear == NULL))
         printf("\n Queue empty");
    else
       printf("Queue not empty");
}
```

OUTPUT:

```
1 - Enque
2 - Deque
3 - Front element
4 - Empty
5 - Exit
6 - Display
7 - Queue size
Enter choice : 1
Enter data : 14

Enter choice : 1
```

Enter data : 85

Enter choice : 1

Enter data : 38

Enter choice : 3

Front element : 14

Enter choice : 6

14 85 38

Enter choice : 7

Queue size : 3

Enter choice : 2

Dequed value : 14

Enter choice : 6

85 38

Enter choice : 7

Queue size : 2

Enter choice : 4

Queue not empty

Enter choice : 5

RESULT:

Thus the program completed and executed successfully.

EX.NO.9 APPLICATIONS OF LIST, STACK AND QUEUE ADTS

AIM:

To write a c program for Applications of List, Stack and Queue ADTs Polynomial additions

ALGORITHM:

1.) [Initialize segment variables][Initialize Counter] Set i=0,j=0,k=0

2.) Repeat step 3 while i<t1 and j<t2

3.) If p1[i].expo=p2[j].expo, then

p3[i].coeff=p1[i].coeff+p2[i].coeff

p3[k].expo=p1[i].exp [Increase counter]

Set i=i+1,j=j+1,k=k+1else if p1[i].expo > p2[j].expo, then

p3[k].coeff=p1[i].coeff p3[k].expo=p1[i].expo [Increase counter] Set i=i+1,k=k+1 else

p3[k].coeff=p2[j].coeff p3[k].expo=p2[j].expo

Set j=j+1,k=k+1

[End of If]

[End of loop]

4.) Repeat while i<t1

p3[k].coeff=p1[i].coeff

p3[k].expo=p1[i].expo Set i=i+1,k=k+1

[End of loop]

5.) Repeat while j<t2

p3[k].coeff=p2[j].coeff

p3[k].expo=p2[j].expo

Set j=j+1,k=k+1

[End of loop]

6.) Return k

7.) Exit

PROGRAM:

```
#include<stdio.h>

/* declare structure for polynomial */
struct poly
{
        int coeff;
        int expo;
};
/* declare three arrays p1, p2, p3 of type structure poly.
* each polynomial can have maximum of ten terms
* addition result of p1 and p2 is stored in p3 */

struct poly p1[10],p2[10],p3[10];

/* function prototypes */
int readPoly(struct poly []);
int addPoly(struct poly [],struct poly [],int ,int ,struct poly []);
void displayPoly( struct poly [],int terms);

int main()
{
        int t1,t2,t3;

        /* read and display first polynomial */
        t1=readPoly(p1);
```

```
        printf(" \n First polynomial : ");
        displayPoly(p1,t1);
        /* read and display second polynomial */
        t2=readPoly(p2);
        printf(" \n Second polynomial : ");
        displayPoly(p2,t2);

        /* add two polynomials and display resultant polynomial */
        t3=addPoly(p1,p2,t1,t2,p3);
        printf(" \n\n Resultant polynomial after addition : ");
        displayPoly(p3,t3);
        printf("\n");

        return 0;
}

int readPoly(struct poly p[10])
{
        int t1,i;

        printf("\n\n Enter the total number of terms in the polynomial:");
        scanf("%d",&t1);

        printf("\n Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER\n");
        for(i=0;i<t1;i++)
        {
```

```
            printf("   Enter the Coefficient(%d): ",i+1);
            scanf("%d",&p[i].coeff);
            printf("      Enter the exponent(%d): ",i+1);
            scanf("%d",&p[i].expo);        /* only statement in loop */
        }
        return(t1);
}

int addPoly(struct poly p1[10],struct poly p2[10],int t1,int t2,struct poly p3[10])
{
        int i,j,k;



        i=0;
        j=0;
        k=0;

        while(i<t1 && j<t2)
        {
            if(p1[i].expo==p2[j].expo)
            {
                p3[k].coeff=p1[i].coeff + p2[j].coeff;
                p3[k].expo=p1[i].expo;


                i++;
                j++;
```

```
                k++;
        }
        else if(p1[i].expo>p2[j].expo)
        {
                p3[k].coeff=p1[i].coeff;
                p3[k].expo=p1[i].expo;
                i++;
                k++;
        }
        else
        {
                p3[k].coeff=p2[j].coeff;
                p3[k].expo=p2[j].expo;
                j++;
                k++;
        }
}

/* for rest over terms of polynomial 1 */
while(i<t1)
{
        p3[k].coeff=p1[i].coeff;
        p3[k].expo=p1[i].expo;
        i++;
        k++;
}
```

```
        /* for rest over terms of polynomial 2 */
        while(j<t2)
        {
                p3[k].coeff=p2[j].coeff;
                p3[k].expo=p2[j].expo;
                j++;
                k++;
        }

        return(k); /* k is number of terms in resultant polynomial*/
}

void displayPoly(struct poly p[10],int term)
{
        int k;

        for(k=0;k<term-1;k++)
        printf("%d(x^%d)+",p[k].coeff,p[k].expo);
        printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);
}
```

OUTPUT:

```
Enter the total number of terms in the polynomial:4
Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER
Enter the Coefficient(1): 3
```

Enter the exponent(1): 4

Enter the Coefficient(2): 7

Enter the exponent(2): 3

Enter the Coefficient(3): 5

Enter the exponent(3): 1

Enter the Coefficient(4): 8

Enter the exponent(4): 0

First polynomial : 3(x^4)+7(x^3)+5(x^1)+8(x^0)

Enter the total number of terms in the polynomial:5

Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER

Enter the Coefficient(1): 7

Enter the exponent(1): 5

Enter the Coefficient(2): 6

Enter the exponent(2): 4

Enter the Coefficient(3): 8

Enter the exponent(3): 2

Enter the Coefficient(4): 9

Enter the exponent(4): 1

Enter the Coefficient(5): 2

Enter the exponent(5): 0

Second polynomial : 7(x^5)+6(x^4)+8(x^2)+9(x^1)+2(x^0)

Resultant polynomial after addition : 7(x^5)+9(x^4)+7(x^3)+8(x^2)+14(x^1)+10(x^0)

RESULT:

Thus the program completed and executed successfully.

EX.N0. 10 IMPLEMENTATION OF BINARY TREES AND OPERATIONS OF BINARY TREES

AIM:

To write a c program for Implementation of Binary Trees and operations of Binary Trees

ALGORITHM:

Step 1: Declare function

add(),search(),findmin().find(),findmax(),Display().Step 2: Create a

structure for a tree contains left pointer and right pointer.

Step 3: Insert an element is by checking the top node and the leaf node and the operation

will be performed.

Step 4: Deleting an element contains searching the tree and deleting the

item.Step 5: Display the Tree elements.

PROGRAM:

```
#include <stdio.h>

#include <stdlib.h>


// structure of a node

struct node

{

   int data;

   struct node *left;
```

```
    struct node *right;
};

// globally initialized root pointer
struct node *root = NULL;

// function prototyping
struct node *create_node(int);
void insert(int);
struct node *delete (struct node *, int);
int search(int);
void inorder(struct node *);
void postorder();
void preorder();
struct node *smallest_node(struct node *);
struct node *largest_node(struct node *);
int get_data();

int main()
{
    int userChoice;
    int userActive = 'Y';
    int data;
```

```
struct node* result = NULL;

while (userActive == 'Y' || userActive == 'y')
{
    printf("\n\n------- Binary Search Tree ------\n");
    printf("\n1. Insert");
    printf("\n2. Delete");
    printf("\n3. Search");
    printf("\n4. Get Larger Node Data");
    printf("\n5. Get smaller Node data");
    printf("\n\n-- Traversals --");
    printf("\n\n6. Inorder ");
    printf("\n7. Post Order ");
    printf("\n8. Pre Oder ");
    printf("\n9. Exit");

    printf("\n\nEnter Your Choice: ");
    scanf("%d", &userChoice);
    printf("\n");

    switch(userChoice)
    {
        case 1:
```

```
    data = get_data();
    insert(data);
    break;

case 2:
    data = get_data();
    root = delete(root, data);
    break;

case 3:
    data = get_data();
    if (search(data) == 1)
    {
        printf("\nData was found!\n");
    }
    else
    {
        printf("\nData does not found!\n");
    }
    break;

case 4:
    result = largest_node(root);
```

```
    if (result != NULL)
    {
        printf("\nLargest Data: %d\n", result->data);
    }
    break;

case 5:
    result = smallest_node(root);
    if (result != NULL)
    {
        printf("\nSmallest Data: %d\n", result->data);
    }
    break;

case 6:
    inorder(root);
    break;

case 7:
    postorder(root);
    break;

case 8:
```

```
            preorder(root);
            break;

        case 9:
            printf("\n\nProgram was terminated\n");
            break;

        default:
            printf("\n\tInvalid Choice\n");
            break;
      }

      printf("\n__________\nDo you want to continue? ");
      fflush(stdin);
      scanf(" %c", &userActive);
   }

   return 0;
}

// creates a new node
struct node *create_node(int data)
{
```

```
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    if (new_node == NULL)
    {
        printf("\nMemory for new node can't be allocated");
        return NULL;
    }

    new_node->data = data;
    new_node->left = NULL;
    new_node->right = NULL;

    return new_node;
}

// inserts the data in the BST
void insert(int data)
{
    struct node *new_node = create_node(data);

    if (new_node != NULL)
    {
        // if the root is empty then make a new node as the root node
```

```
if (root == NULL)
{
    root = new_node;
    printf("\n* node having data %d was inserted\n", data);
    return;
}

struct node *temp = root;
struct node *prev = NULL;

// traverse through the BST to get the correct position for insertion
while (temp != NULL)
{
    prev = temp;
    if (data > temp->data)
    {
        temp = temp->right;
    }
    else
    {
        temp = temp->left;
    }
}
```

```
        // found the last node where the new node should insert
        if (data > prev->data)
        {
            prev->right = new_node;
        }
        else
        {
            prev->left = new_node;
        }

        printf("\n* node having data %d was inserted\n", data);
    }
}

// deletes the given key node from the BST
struct node *delete (struct node *root, int key)
{
    if (root == NULL)
    {
        return root;
    }
    if (key < root->data)
```

```
{
   root->left = delete (root->left, key);
}
else if (key > root->data)
{
   root->right = delete (root->right, key);
}
else
{
   if (root->left == NULL)
   {
      struct node *temp = root->right;
      free(root);
      return temp;
   }
   else if (root->right == NULL)
   {
      struct node *temp = root->left;
      free(root);
      return temp;
   }
   struct node *temp = smallest_node(root->right);
   root->data = temp->data;
```

```
        root->right = delete (root->right, temp->data);
    }
    return root;

}

// search the given key node in BST
int search(int key)
{
    struct node *temp = root;

    while (temp != NULL)
    {
        if (key == temp->data)
        {
            return 1;
        }
        else if (key > temp->data)
        {
            temp = temp->right;
        }
        else
        {
```

```
            temp = temp->left;
        }
    }
    return 0;
}

// finds the node with the smallest value in BST
struct node *smallest_node(struct node *root)
{
    struct node *curr = root;
    while (curr != NULL && curr->left != NULL)
   {
        curr = curr->left;
    }
    return curr;
}

// finds the node with the largest value in BST
struct node *largest_node(struct node *root)
{
    struct node *curr = root;
    while (curr != NULL && curr->right != NULL)
    {
```

```
        curr = curr->right;
    }
    return curr;
}

// inorder traversal of the BST
void inorder(struct node *root)
{
    if (root == NULL)
    {
        return;
    }
    inorder(root->left);
    printf("%d ",  root->data);
    inorder(root->right);
}

// preorder traversal of the BST
void preorder(struct node *root)
{
    if (root == NULL)
    {
        return;
```

```
    }
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}

// postorder travsersal of the BST
void postorder(struct node *root)
{
    if (root == NULL)
    {
        return;
    }
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}

// getting data from the user
int get_data()
{
    int data;
    printf("\nEnter Data: ");
```

```
        scanf("%d", &data);
        return data;
    }
```

OUTPUT:

```
------- Binary Search Tree ------
1. Insert
2. Delete
3. Search
4. Get Larger Node Data
5. Get smaller Node data

-- Traversals --

6. Inorder
7. Post Order
8. Pre Oder
9. Exit

Enter Your Choice: 1
Enter Data: 20
* node having data 20 was inserted
__________
Do you want to continue? y
------- Binary Search Tree ------
1. Insert
2. Delete
```

3. Search

4. Get Larger Node Data

5. Get smaller Node data

-- Traversals --

6. Inorder

7. Post Order

8. Pre Oder

9. Exit

Enter Your Choice: 1

Enter Data: 15

* node having data 15 was inserted

_____

Do you want to continue? y

------- Binary Search Tree ------

1. Insert

2. Delete

3. Search

4. Get Larger Node Data

5. Get smaller Node data

-- Traversals --

6. Inorder

7. Post Order

8. Pre Oder

9. Exit

Enter Your Choice: 1

Enter Data: 25

* node having data 25 was inserted

_____

Do you want to continue? y

------- Binary Search Tree ------

1. Insert

2. Delete

3. Search

4. Get Larger Node Data

5. Get smaller Node data

-- Traversals --

6. Inorder

7. Post Order

8. Pre Oder

9. Exit

Enter Your Choice: 1

Enter Data: 12

* node having data 12 was inserted

_____

Do you want to continue? y

------- Binary Search Tree ------

1. Insert

2. Delete

3. Search

4. Get Larger Node Data

5. Get smaller Node data

-- Traversals --

6. Inorder

7. Post Order

8. Pre Oder

9. Exit

Enter Your Choice: 1

Enter Data: 18

* node having data 18 was inserted

_____

Do you want to continue? y

------- Binary Search Tree ------

1. Insert

2. Delete

3. Search

4. Get Larger Node Data

5. Get smaller Node data

-- Traversals --

6. Inorder

7. Post Order

8. Pre Oder

9. Exit

Enter Your Choice: 1

Enter Data: 65

* node having data 65 was inserted

_____

Do you want to continue? N

RESULT:

Thus the program completed and executed successfully.

EX.N0. 11 **BINARY SEARCH TREE**

AIM:

To write a c program for binary search tree.

ALGORITHM:

Step 1: Declare function add(),search(),findmin().find(),findmax(),Display().

Step 2: Create a structure for a tree contains left pointer and right pointer.

Step 3: Insert an element is by checking the top node and the leaf node and the operation will be performed.

Step 4: Deleting an element contains searching the tree and deleting the

item.Step 5: Display the Tree elements.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.>
#include<conio.h>


struct searchtree
{
    int element;
struct searchtree *left,*right;
}*root;


typedef struct searchtree *node;
```

```
typedef int ElementType;
node insert(ElementType, node);
node delete(ElementType, node);
void makeempty();
node findmin(node);
node findmax(node);
node find(ElementType, node);
void display(node, int);
void main()
{
int ch;
ElementType a;
node temp;
makeempty();
while(1)
{
printf("\n1. Insert\n2. Delete\n3. Find min\n4. Find max\n5. Find\n6. Display\n7. Exit\nEnter Your Choice : ");
scanf("%d",&ch);
switch()
case 1:
printf("Enter an element : ");
scanf("%d", &a);
root = insert(a, root);
```

```
break;
case 2:
printf("\nEnter the element to delete : ");
scanf("%d",&a);
root = delet(a, root);
break;
case 3:
printf("\nEnter the element to search : ");
scanf("%d",&a); temp = find(a, root); if (temp != NULL);
printf("Element found");
break;
case 4:
printf("\nEnter the element to search : ");
scanf("%d",&a); temp = find(a, root);
if (temp != NULL) printf("Element found");
break;
case 5:
temp = findmin (root);
if(temp==NULL)
printf("\nEmpty tree");
else
printf("\nMinimum element : %d", temp->element);
break;
```

```
case 6:
if(root==NULL)
printf("\nEmpty tree");
break;
case 7:
else
display(root, 1);
exit(0) default:
printf(“\nInvalid choice”);
}
}
}
node insert(ElementType x,node t)
{
if(t==NULL)
{
t = (node)malloc(sizeof(node)); t->element = x;
t->left = t->right = NULL;
}
else
{
if(x < t->element)
t->left = insert(x, t->left); else if(x > t->element)
```

```
t->right = insert(x, t->right);
}
return t;
}
node delete(ElementType x,node t)
{
node temp; if(t == NULL)
printf("\nElement not found");
else
{
if(x < t->element)
t->left = delete(x, t->left); else if(x > t->element)
t->right = delet(x, t->right);
else
{
if(t->left && t->right)
{
temp = findmin(t->right);
t->element = temp->element;
t->right = delet(t->element,t->right);
}
else if(t->left == NULL)
t=t->right;
```

```
}
}return t;
}
else
 t=t->left;
 void makeempty()
{
root = NULL;
}
node findmin(node temp)
{
if(temp == NULL || temp->left == NULL) return temp;
return findmin(temp->left);
}
node findmax(node temp)
{
if(temp==NULL || temp->right==NULL) return temp;
return findmin(temp->right);
}
 node find(ElementType x, node t)
{
if(t==NULL) return NULL;
 if(x<t->element) return find(x,t->left); if(x>t->element) return find(x,t->right);
return t
```

```
    }
    void display(node t,int level)
    {
     int i; if(t)
    {
     display(t->right, level+1);
     printf("\n");
    for(i=0;i<level;i++)
    {
    printf(" ");
    }
    printf("%d", t->element);
    display(t->left, level+1);
}
}
```

OUTPUT

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 1 Enter an element : 10

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max

6. Display
7. Exit

Enter your Choice : 1 Enter an element : 20

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 1

Enter an element : 5

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 4 The smallest Number is 5

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 3 Enter an element : 100 Element not Found

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 2 Enter an element : 20

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 6 5
10

1. Insert
2. Delete
3. Find
4. Find Min
5. Find Max
6. Display
7. Exit

Enter your Choice : 7

RESULT:

Thus the program completed and executed successfully.

EX NO :12 IMPLEMENTATION OF LINEAR SEARCH

AIM:

To write the C program for implementing the linear search.

ALGORITHM:

Linear_Search(a, n, val) // 'a' is the given array, 'n' is the size of given array, 'val' is the value to search
Step 1: set pos = -1
Step 2: set i = 1
Step 3: repeat step 4 while i <= n Step 4: if a[i] == val
set pos = i print pos go to step 6
[end of if]
set ii = i + 1 [end of loop]
Step 5: if pos = -1
print "value is not present in the array " [end of if]
Step 6: exit
\

PROGRAM:

```
#include <stdio.h>

void main()
{
    int num;
    int i,  keynum, found = 0;

   printf("Enter the number of elements ");
   scanf("%d", &num);
   int array[num];
   printf("Enter the elements one by one \n");
   for (i = 0; i < num; i++)
   {
      scanf("%d", &array[i]);
   }

   printf("Enter the element to be searched ");
   scanf("%d", &keynum);

   /*  Linear search begins */
```

```
   for (i = 0; i < num ; i++)
   {
      if (keynum == array[i] )
      {
         found = 1;
         break;
      }
   }
   if (found == 1)
      printf("Element is present in the array at position %d",i+1);
   else
      printf("Element is not present in the array\n");
}
```

OUTPUT:

```
Enter the Size of the Array: 6
Enter the Array:
arr[0]=  7
arr[1]=  9
arr[2]=  74
arr[3]=  23
arr[4]=  64
arr[5]=  92
Enter the Element to be Searched: 74
Element is at 2 position from start      {0-based indexing}
Element is at 3 position from start      {1-based indexing}
```

RESULT:

Thus the program completed and executed successfully.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3352 DIGITAL SYSTEMS DESIGN

Semester - 03

LABORATORY MANUAL

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

1. Design of adders and subtractors & code converters.
2. Design of Multiplexers & Demultiplexers.
3. Design of Encoders and Decoders.
4. Design of Magnitude Comparators
5. Design and implementation of counters using flip-flops
6. Design and implementation of shift registers.

EXP NO. : 01 **STUDY OF LOGIC GATES**

DATE :

AIM:

To study about logic gates and verify their truth tables.

APPARATUS REQUIRED:

SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE 2 I/P	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
7.	NAND GATE 3 I/P	IC 7410	1
8.	IC TRAINER KIT	-	1

THEORY:

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output.

OR, AND and NOT are basic gates. NAND and NOR are known as universal gates. Basic gates form these gates.

AND GATE:

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE:

The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE:

The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

NAND GATE:

The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

NOR GATE:

The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

X-OR GATE:

The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

PROCEDURE:

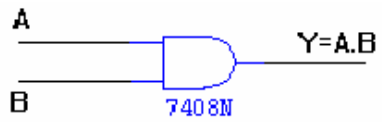
(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

AND GATE:

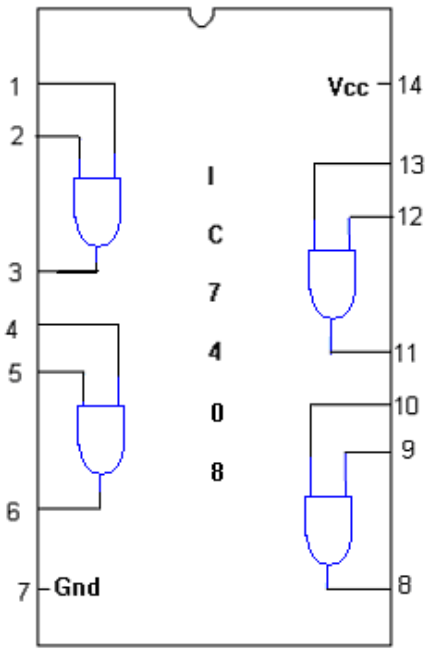
SYMBOL:



TRUTH TABLE

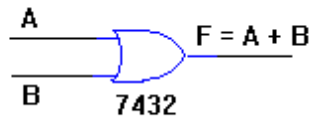
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:

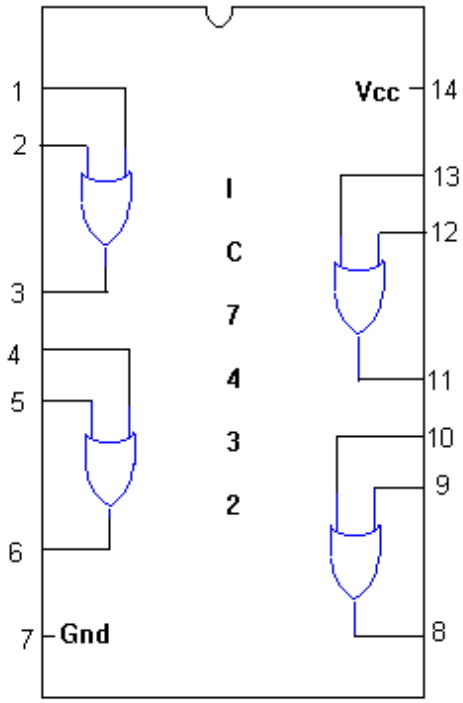


OR GATE:

SYMBOL :



PIN DIAGRAM :

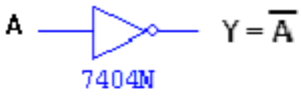


TRUTH TABLE

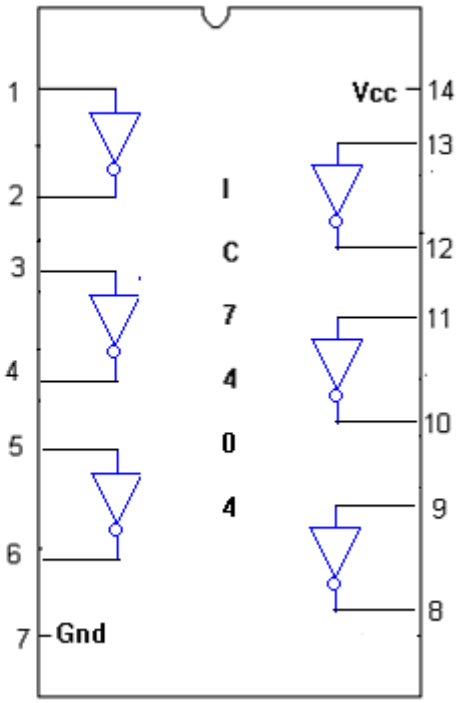
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

SYMBOL:



PIN DIAGRAM:



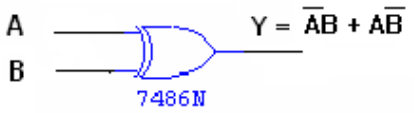
TRUTH TABLE :

A	$\overline{A}$
0	1
1	0

X-OR GATE:

SYMBOL:

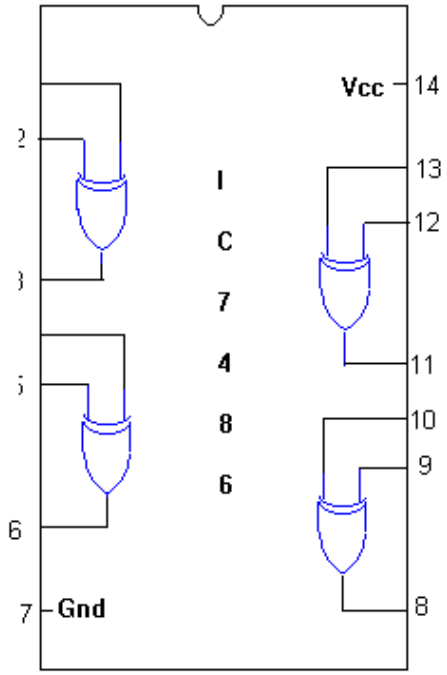
DIAGRAM:



TRUTH TABLE :

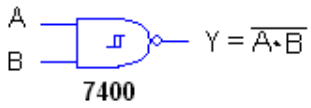
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

PIN



2-INPUT NAND GATE:

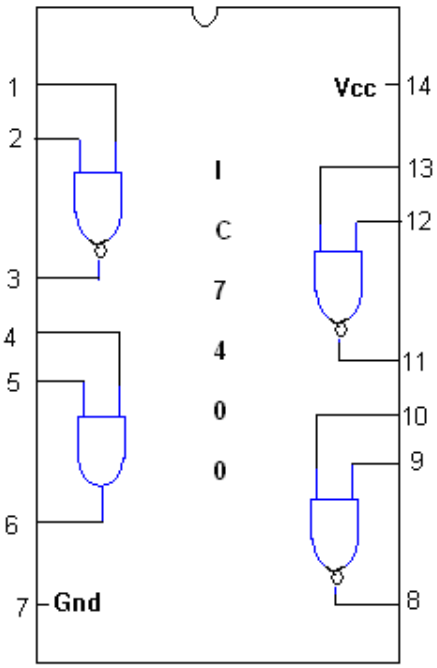
SYMBOL:



TRUTH TABLE

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

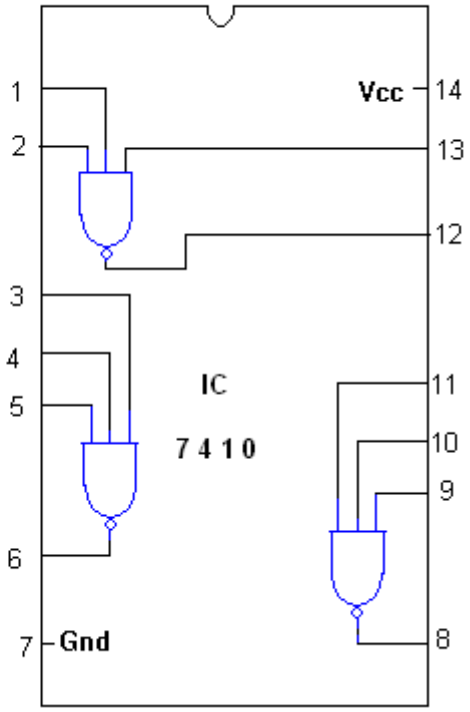
PIN DIAGRAM:



3-INPUT NAND GATE :

SYMBOL :

PIN DIAGRAM :

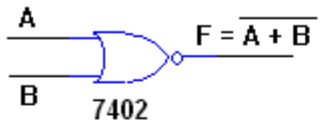


TRUTH TABLE

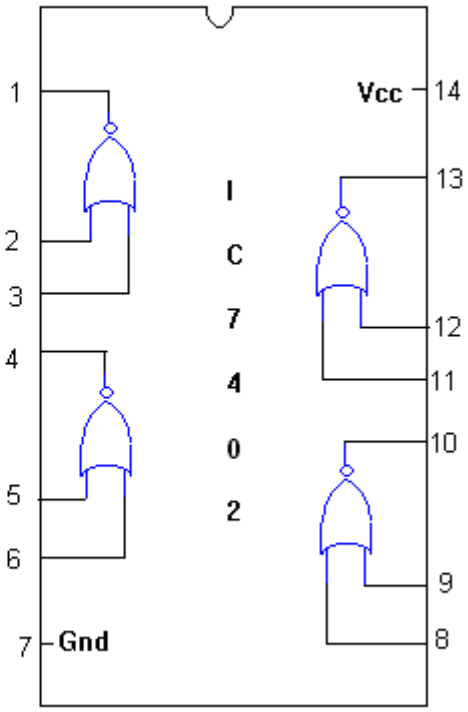
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE:

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

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

RESULT:

The truth tables of all the basic logic gates were verified.

EXP NO. : 02
DATE :

VERIFICATION OF BOOLEAN THEOREMS USING DIGITAL LOGIC GATES

AIM:
To verify the Boolean Theorems using logic gates.

APPARATUS REQUIRED:

SL. NO.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	CONNECTING WIRES	-	As per required

THEORY:

BASIC BOOLEAN LAWS

1. Commutative Law

The binary operator OR, AND is said to be commutative if,

A+B = B+A

A.B=B.A

2. Associative Law

The binary operator OR, AND is said to be associative

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

2. A.(B.C) = (A.B).C

3. Distributive Law

The binary operator OR, AND is said to be distributive

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

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

4. Absorption Law

1. A+AB = A
2. A+AB = A+B

5. Involution (or) Double complement Law

A = A

6. Idempotent Law

1.A+A = A

2.A.A = A

7. Complementary Law

1. A+A' = 1
2. A.A' = 0

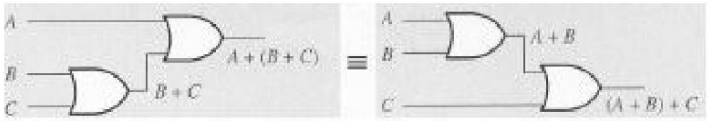
8. De Morgan's Theorem

1. The complement of the sum is equal to the sum of the product of the individual complements. A+B = A.B

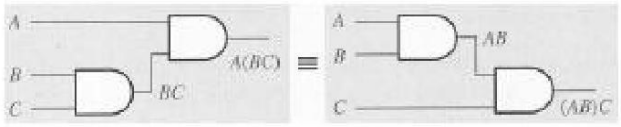
2. The complement of the product is equal to the sum of the individual complements. A.B = A+B

Associative Laws of Boolean Algebra

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



$$A \bullet (B \bullet C) = (A \bullet B) \bullet C$$



Proof of the Associative Property for the OR operation: (A+B)+C = A+(B+C)

A	B	C	(A+B)	(B+C)	A+(B+C)	(A+B)+C
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

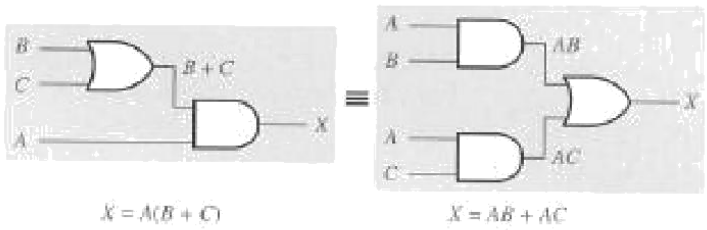
Proof of the Associative Property for the AND operation: (A·B)·C = A·(B·C)

A	B	C	(A·B)	(B·C)	A· (B·C)	(A·B)·C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Distributive Laws of Boolean Algebra

$$A \bullet (B + C) = A \bullet B + A \bullet C$$

$$A\,(B + C) = A\,B + A\,C$$



Proof of Distributive Rule

A	B	C	A·B	A·C	(A·B)+ (A·C)	(B+C)	A·(B+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

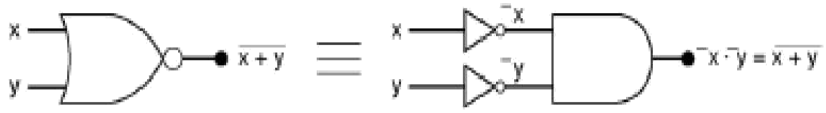
Proof of Distributive Rule

A	B	C	A+B	A+C	(A+B)· (A+C)	(B·C)	A+(B·C)
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1

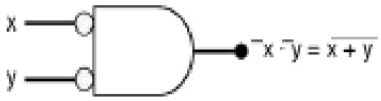
Demorgan's Theorem

a) Proof of equation (1):

Construct the two circuits corresponding to the functions A'. B'and (A+B)' respectively. Show that for all combinations of A and B, the two circuits give identical results. Connect these circuits and verify their operations.



(a)



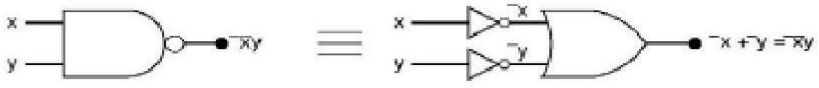
(b)

Proof (via Truth Table) of DeMorgan's Theorem $\overline{A \cdot B} = \overline{A} + \overline{B}$

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

Proof of equation (2)

Construct two circuits corresponding to the functions A'+B'and (A.B)' A.B, respectively. Show that, for all combinations of A and B, the two circuits give identical results. Connect these circuits and verify their operations.



(a)

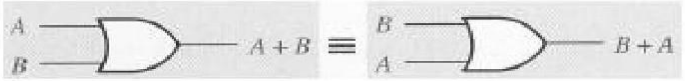
(b)

Proof (via Truth Table) of DeMorgan's Theorem $\overline{A + B} = \overline{A} \cdot \overline{B}$

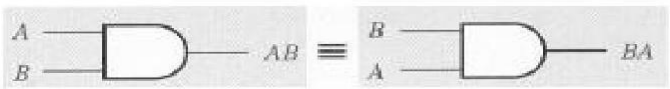
A	B	A+B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Commutative Laws of Boolean Algebra

A + B = B + A



A • B = B • A



We will also use the following set of postulates:

P1: Boolean algebra is closed under the AND, OR, and NOT operations.

P2: The identity element with respect to • is one and + is zero. There is no identityelement with respect to logical NOT.

P3: The • and + operators are commutative.

P4: • and + are distributive with respect to one another. That is,

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

P5: For every value A there exists a value A' such that A•A' = 0 and A+A' = 1. This value is the logical complement (or NOT) of A.

P6: • and + are both associative. That is, (A•B)•C = A•(B•C) and (A+B)+C = A+(B+C).

You can prove all other theorems in boolean algebra using these postulates.

PROCEDURE:

1. Obtain the required IC along with the Digital trainer kit.
2. Connect zero volts to GND pin and +5 volts to Vcc .
3. Apply the inputs to the respective input pins.
4. Verify the output with the truth table.

RESULT:

Thus the above stated Boolean laws are verified.

EXP NO. :03

DATE :

DESIGN AND IMPLEMENTATION OF CODE CONVERTERS

AIM:

To design and implement 4-bit

(i) Binary to gray code converter

(ii) Gray to binary code converter

(iii) BCD to excess-3 code converter

(iv) Excess-3 to BCD code converter

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	X-OR GATE	IC 7486	1
2.	AND GATE	IC 7408	1
3.	OR GATE	IC 7432	1
4.	NOT GATE	IC 7404	1
5.	IC TRAINER KIT	-	1
6.	PATCH CORDS	-	35

THEORY:

The availability of large variety of codes for the same discrete elements of information results in the use of different codes by different systems. A conversion circuit must be inserted between the two systems if each uses different codes for same information. Thus, code converter is a circuit that makes the two systems compatible even though each uses different binary code.

The bit combination assigned to binary code to gray code. Since each code uses four bits to represent a decimal digit. There are four inputs and four outputs. Gray code is a non-weighted code.
The input variable are designated as B3, B2, B1, B0 and the output variables are designated as C3, C2, C1, Co. from the truth table, combinational circuit is designed. The Boolean functions are obtained from K-Map for each output variable.

A code converter is a circuit that makes the two systems compatible even though each uses a different binary code. To convert from binary code to Excess-3 code, the input lines must supply the bit combination of elements as specified by code and the output lines generate the corresponding bit combination of code. Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables.

A two-level logic diagram may be obtained directly from the Boolean expressions derived by the maps. These are various other possibilities for a logic diagram that implements this circuit. Now the OR gate whose output is C+D has been used to implement partially each of three outputs.

PROCEDURE:

(i) Connections were given as per circuit diagram.

(ii) Logical inputs were given as per truth table

(iii) Observe the logical output and verify with the truth tables.

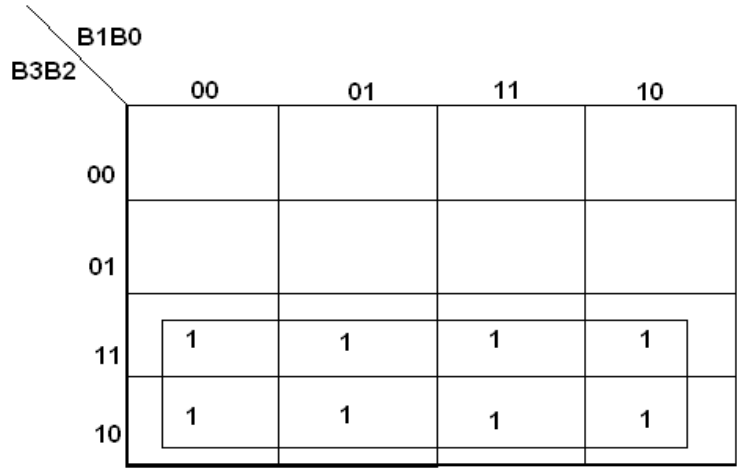
BINARY TO GRAY CODE CONVERTOR

TRUTH TABLE:

| Binary input | Gray code output |

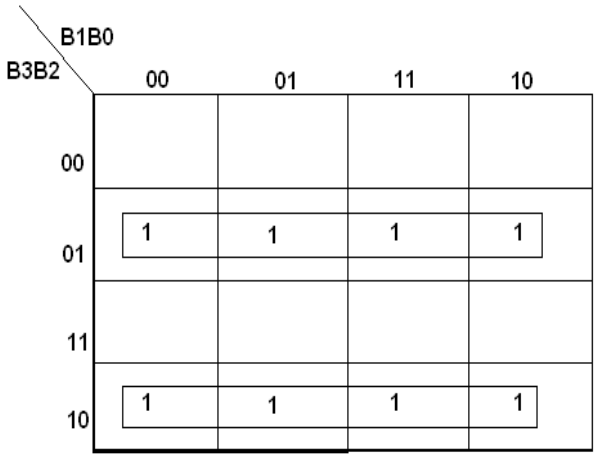
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-Map for G_3:



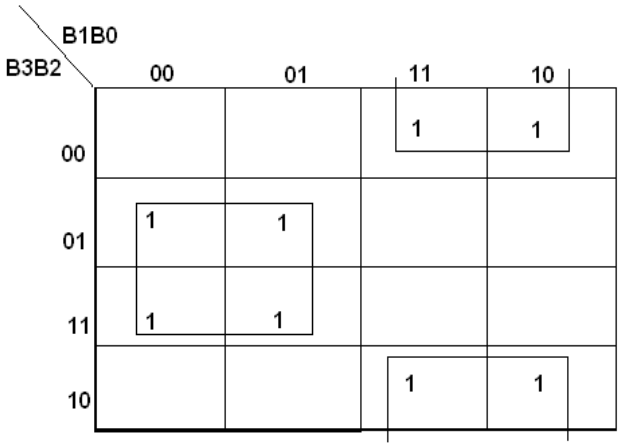
$G_3 = B_3$

K-Map for G_2:



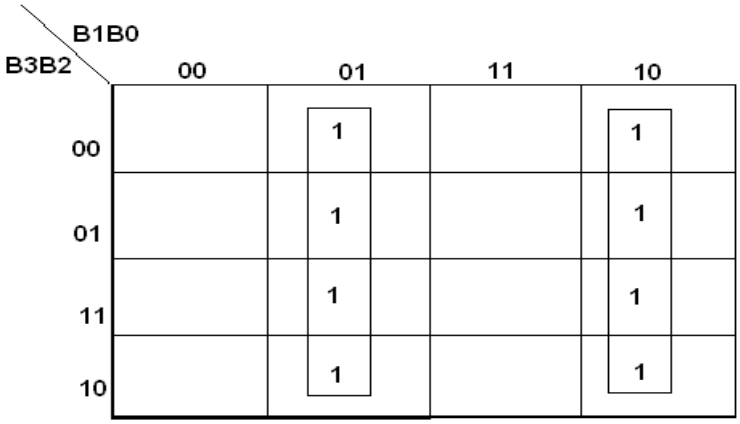
$G2 = B3 \oplus B2$

K-Map for G_1:



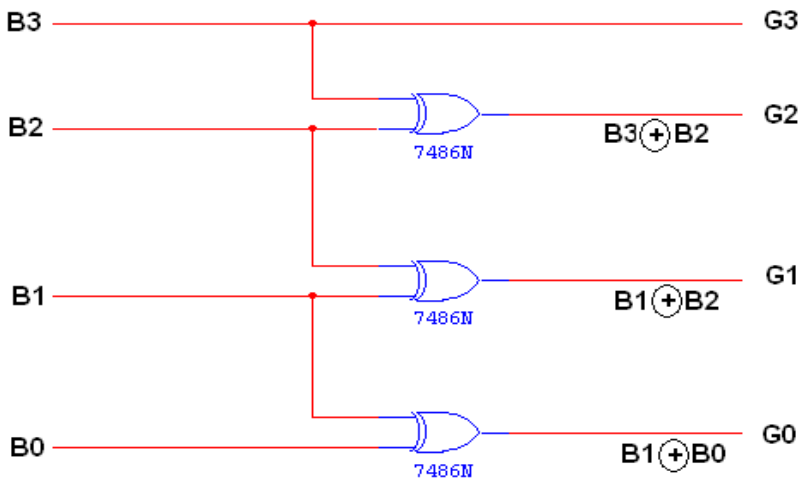
$G1 = B1 \oplus B2$

K-Map for G_0:



$G0 = B1 \oplus B0$

LOGIC DIAGRAM:



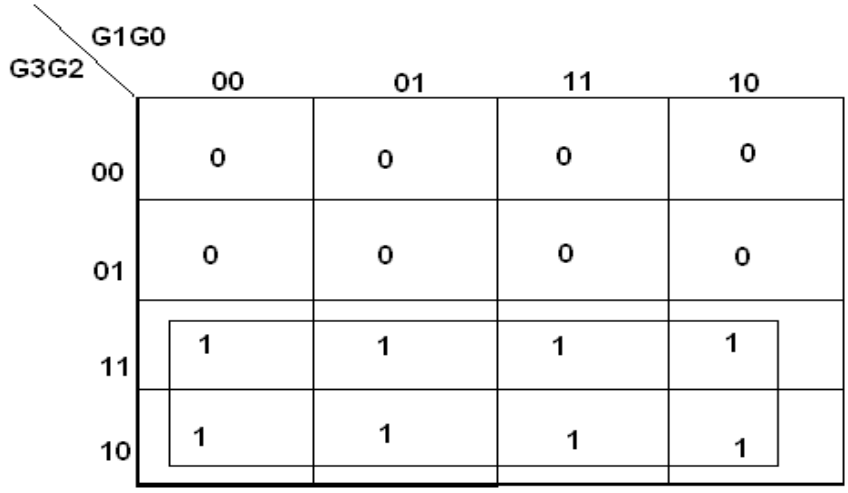
GRAY CODE TO BINARY CONVERTOR

TRUTH TABLE:

| Gray Code | Binary Code |

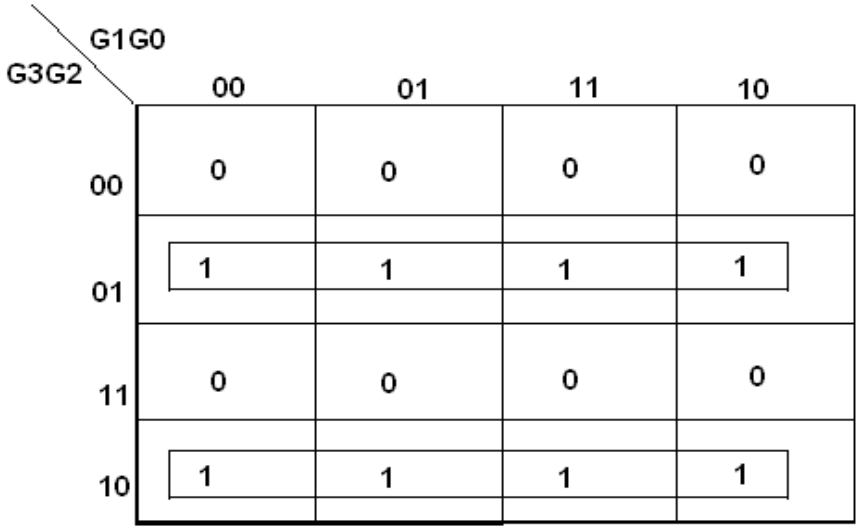
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

K-Map for B_3:



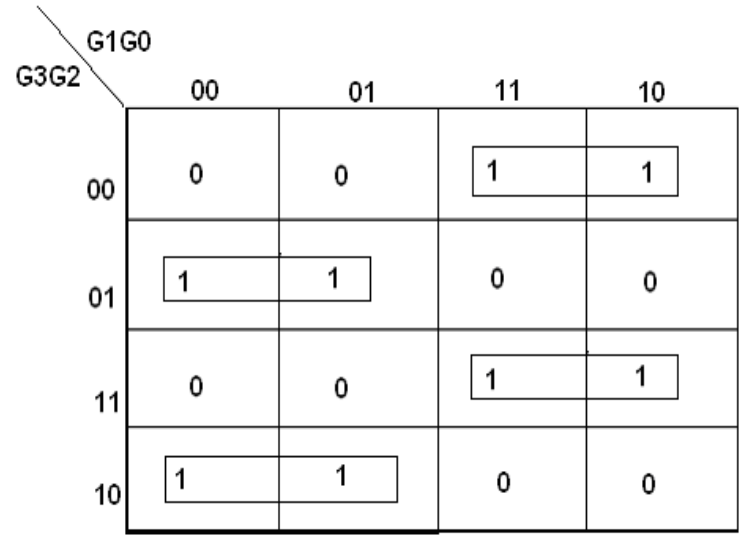
B3 = G3

K-Map for B_2:



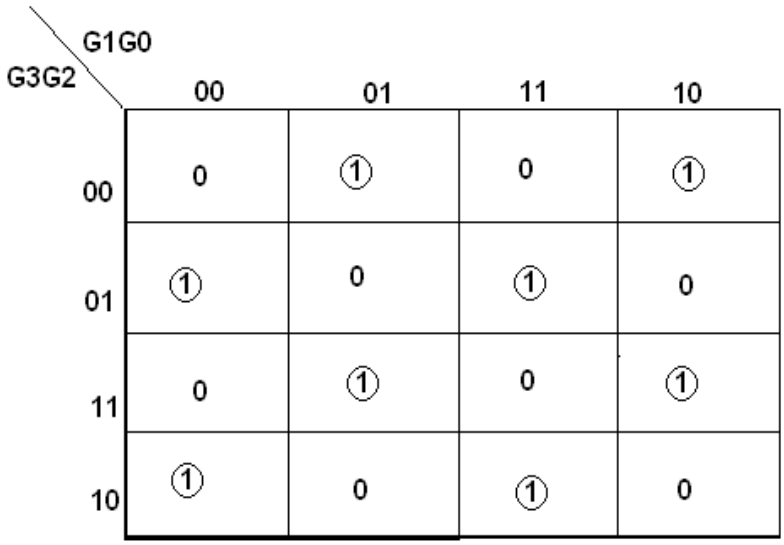
B2 = G3⊕G2

K-Map for B_1:



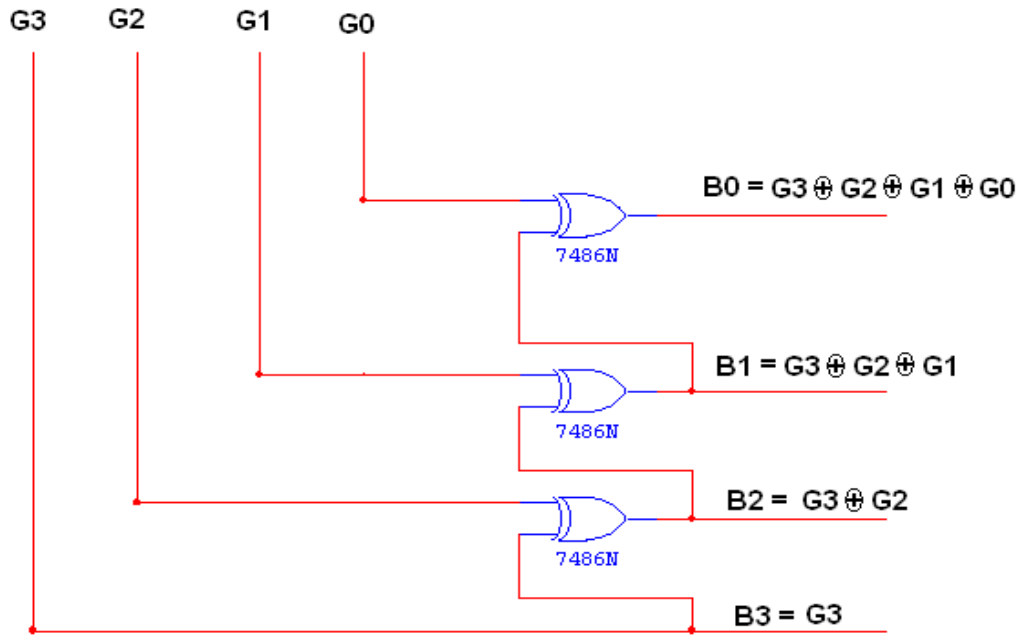
B1 = G3⊕G2⊕G1

K-Map for B_0:



B0 = G3⊕G2⊕G1⊕G0

LOGIC DIAGRAM:

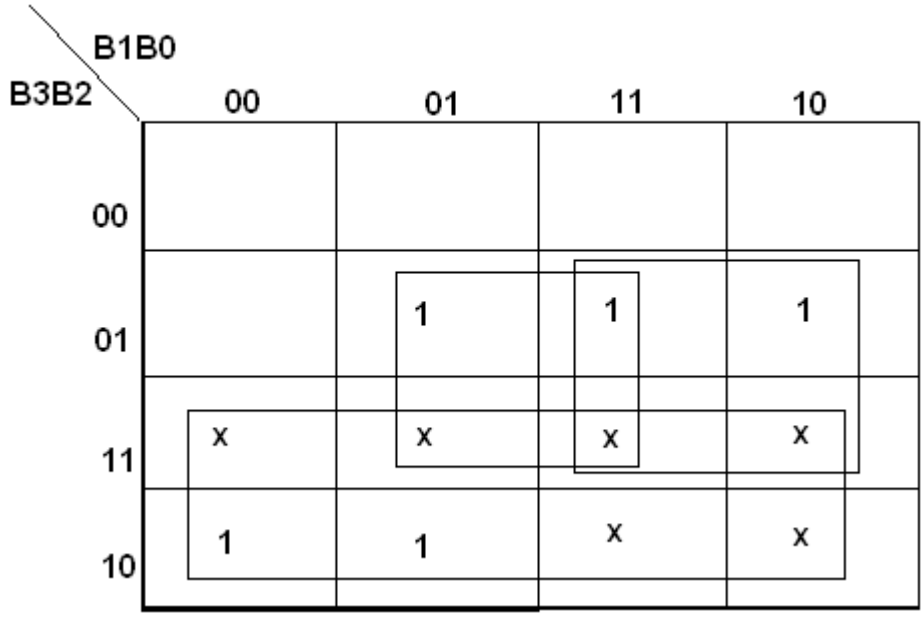


BCD TO EXCESS-3 CONVERTER

TRUTH TABLE:

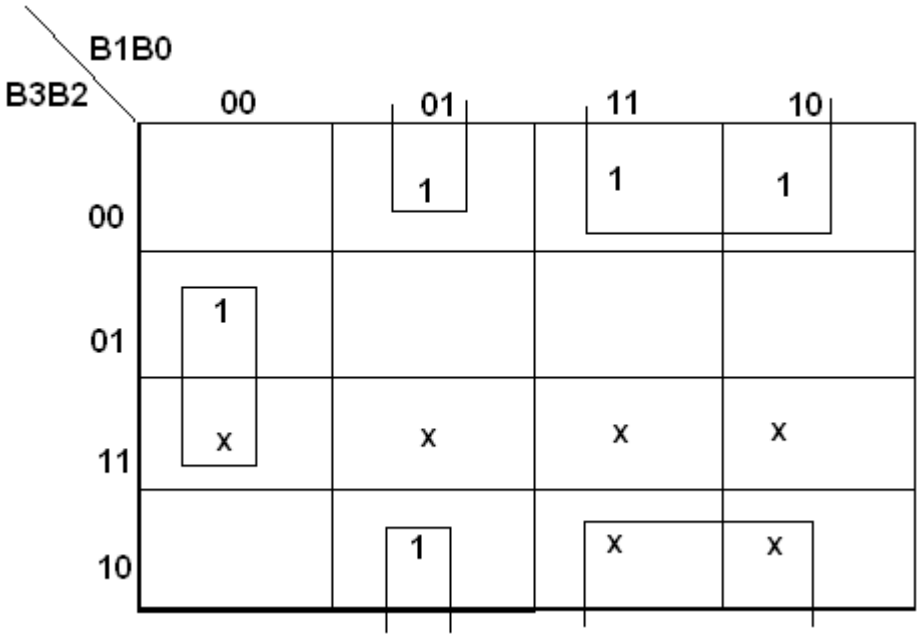
BCD input				Excess – 3 output			
B3	**B2**	**B1**	**B0**	**E3**	**E2**	**E1**	**E0**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

K-Map for E_3:



E3 = B3 + B2 (B0 + B1)

K-Map for E_2:

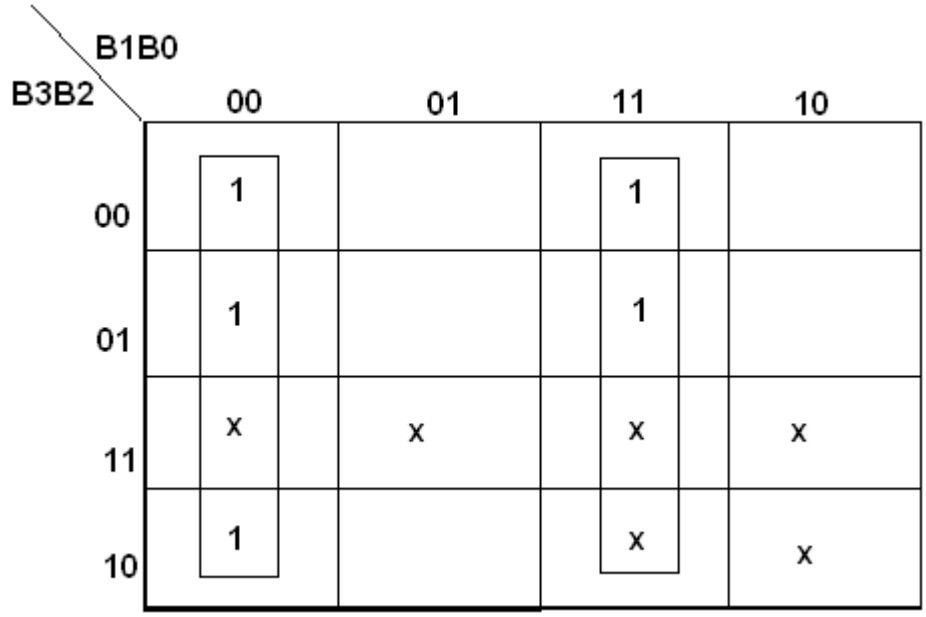


$E2 = B2 \oplus (B1 + B0)$

E2=B2B1+B2B1B0+B2B1B0

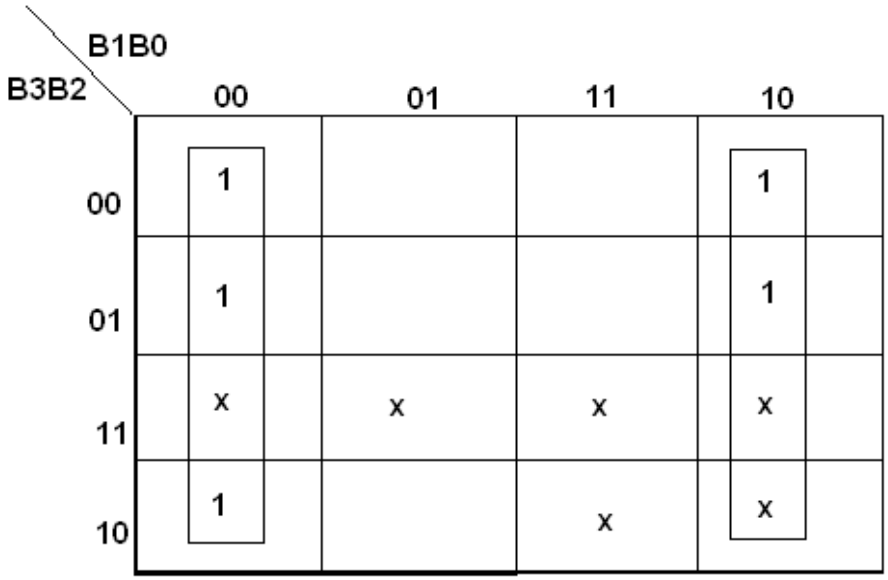
E2=B2B1+B1(B2B0+B2B0) = B2B1+B1(B2 B0)

K-Map for E_1:



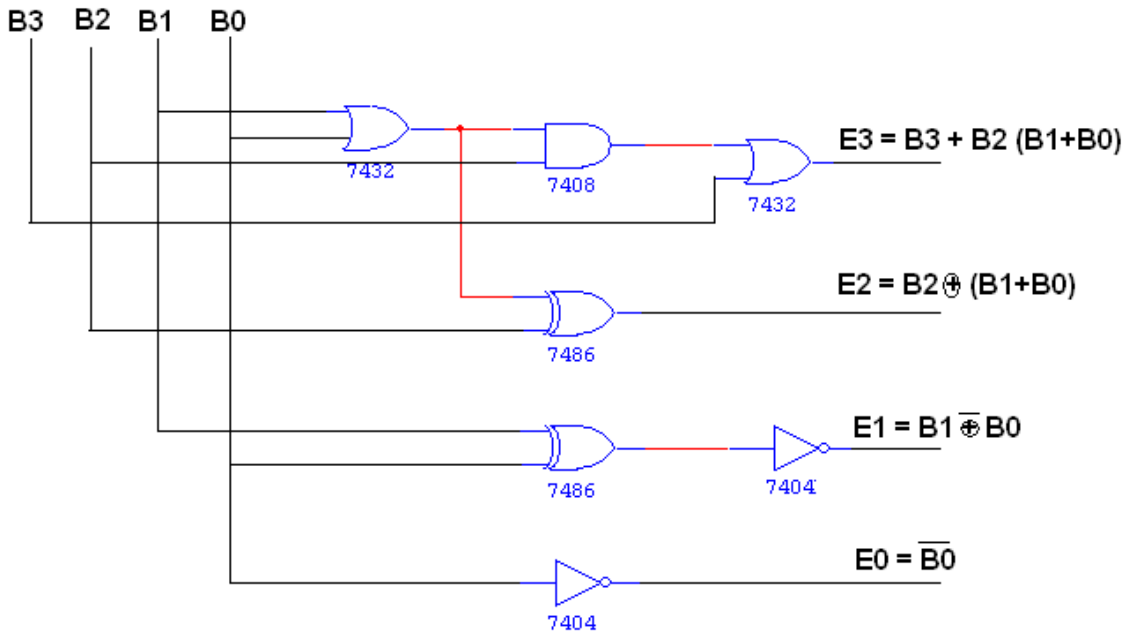
$E1 = B1 \bar{\oplus} B0$ = B1 O B0

K-Map for E_0:



$E0 = \overline{B0}$

LOGIC DIAGRAM:



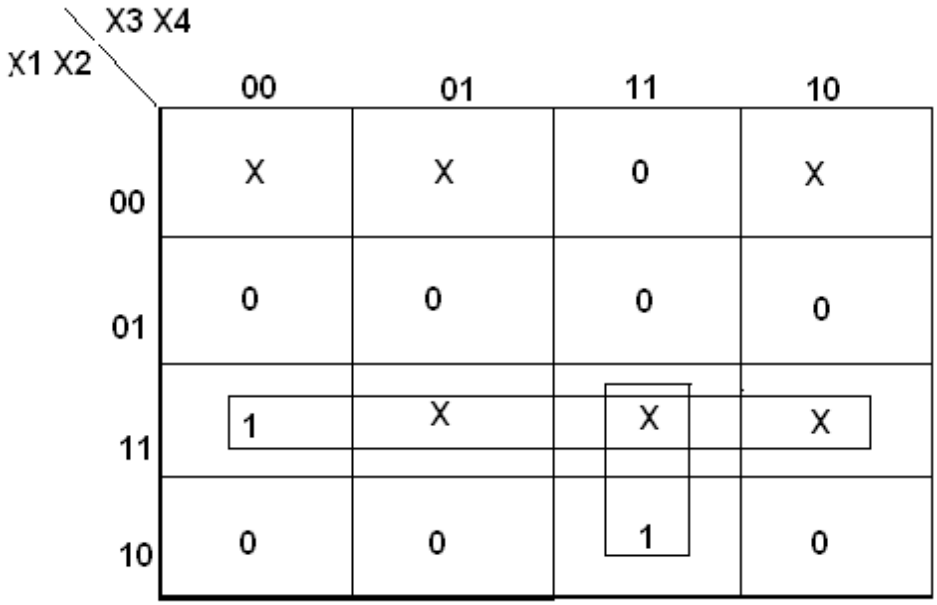
EXCESS-3 TO BCD CONVERTOR

TRUTH TABLE:

| **Excess – 3 Input** | **BCD Output** |

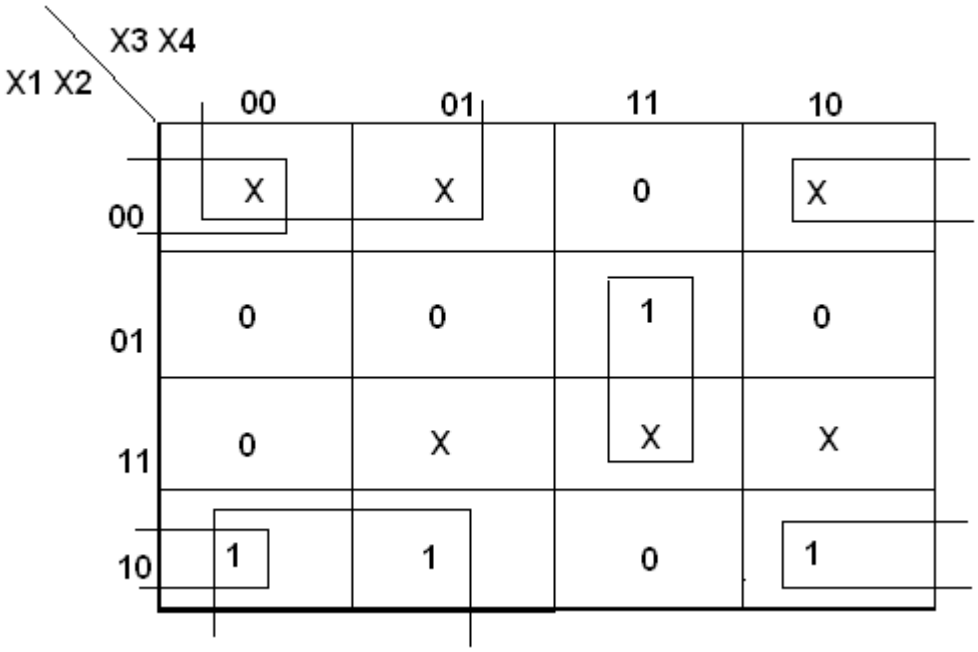
x3	x2	X3	X4	A	B	C	D
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

K-Map for A:



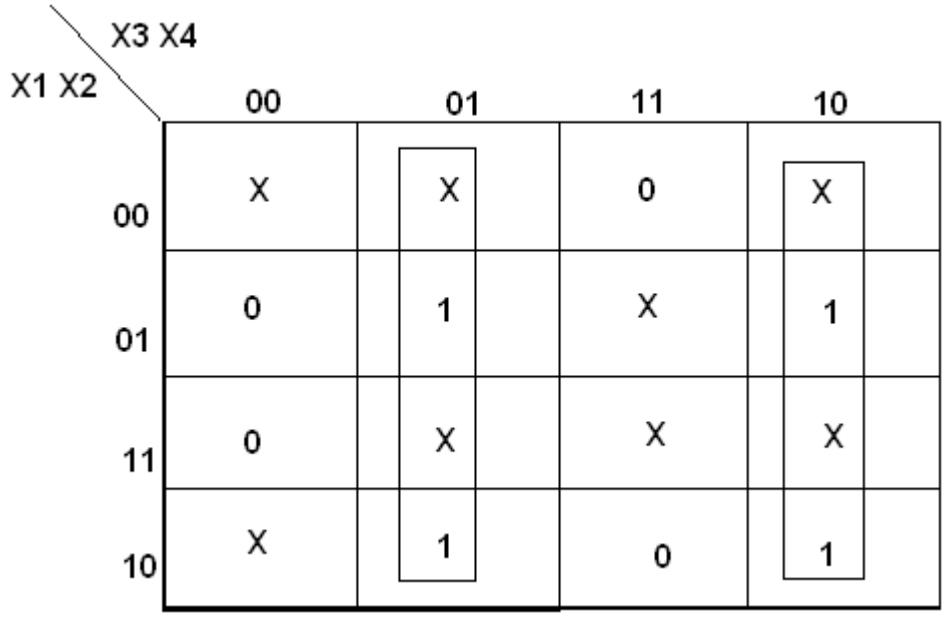
A = X1 X2 + X3 X4 X1

K-Map for B:



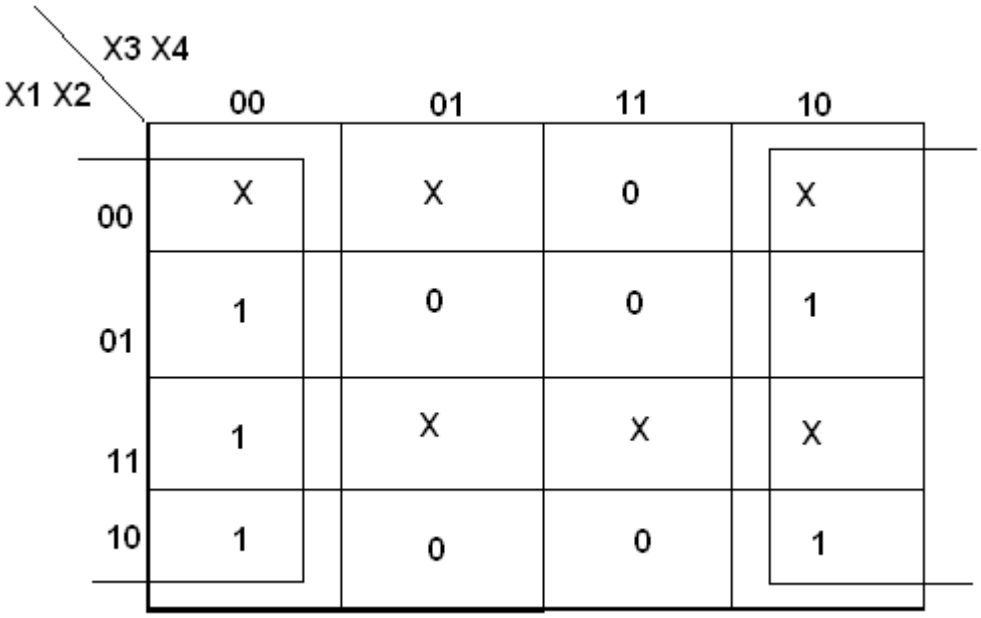
$B = X2 \oplus (\overline{X3} + \overline{X4})$

K-Map for C:



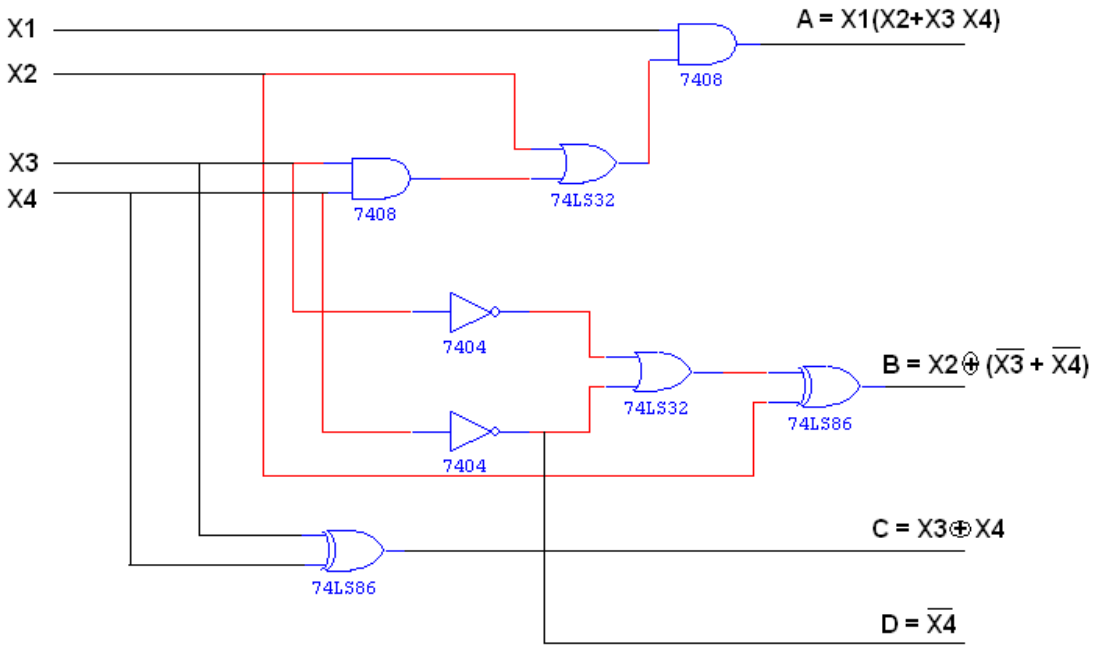
$C = X3 \oplus X4$

K-Map for D:



$D = \overline{X4}$

LOGIC DIAGRAM:



RESULT:

Thus the code converters were designed and verified using the corresponding truth table.

EXP NO. :04 **DESIGN OF 4-BIT ADDER AND SUBTRACTOR**
DATE :

AIM:

To design and implement 4-bit adder and subtractor using IC 7483.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	IC	IC 7483	1
2.	EX-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	40

THEORY:

4 BIT BINARY ADDER:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C_0 and it ripples through the full adder to the output carry C_4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C_0 must be equal to 1 when performing subtraction.

4 BIT BINARY ADDER/SUBTRACTOR:

The addition and subtraction operation can be combined into one circuit with one common binary adder. The mode input M controls the operation. When M=0, the circuit is adder circuit. When M=1, it becomes subtractor.

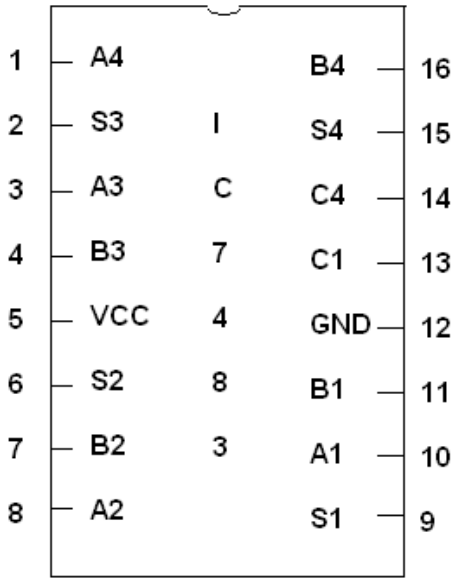
4 BIT BCD ADDER:

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns.

ABCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

PROCEDURE:

1. Connections were given as per circuit diagram.
2. Logical inputs were given as per truth table
3. Observe the logical output and verify with the truth tables.

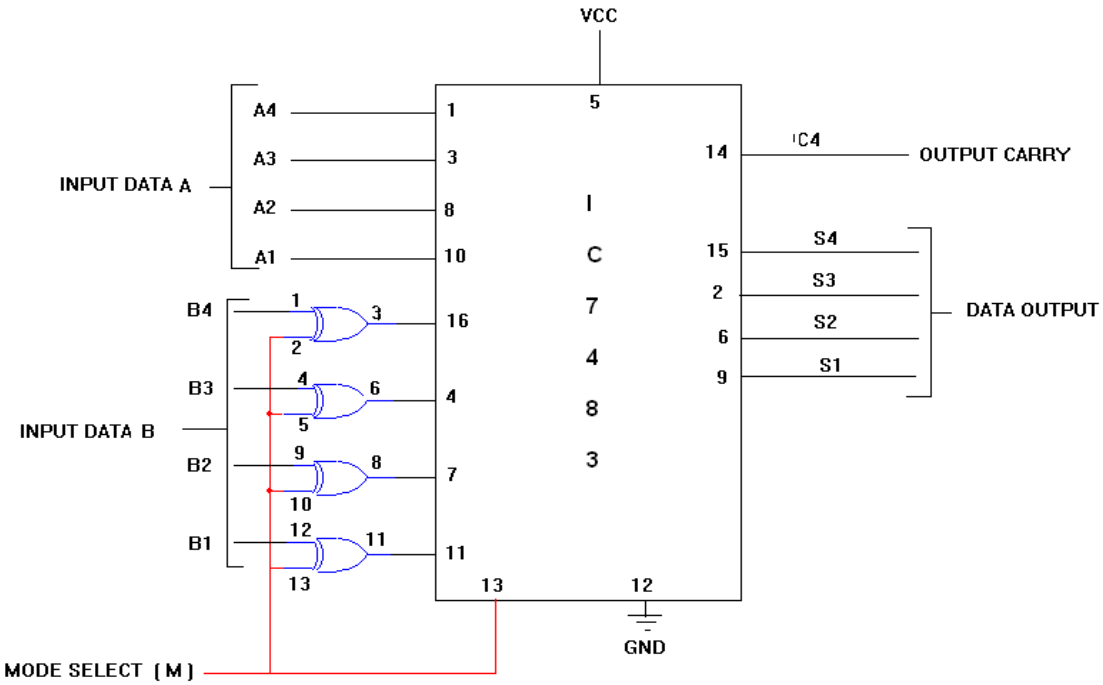


PIN DIAGRAM FOR IC 7483:

Input Data A				**Input Data B**				**Addition**					**Subtraction**				
A4	**A3**	**A2**	**A1**	**B4**	**B3**	**B2**	**B1**	**C**	**S4**	**S3**	**S2**	**S1**	**B**	**D4**	**D3**	**D2**	**D1**
1	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**
0	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
1	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**
1	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
1	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**

4-BIT BINARY ADDER/SUBTRACTOR

LOGIC DIAGRAM:

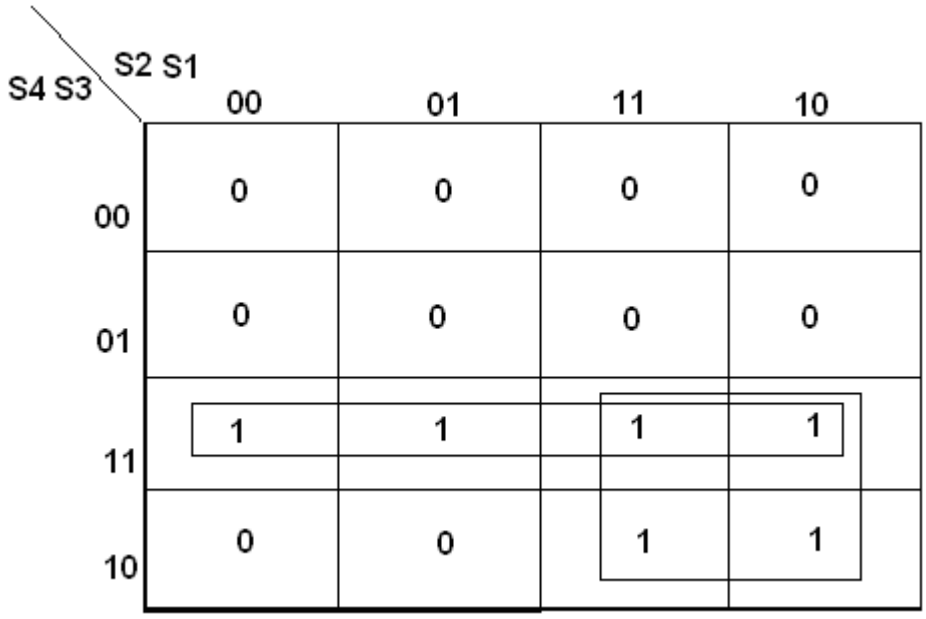


BCD ADDER:

TRUTH TABLE:

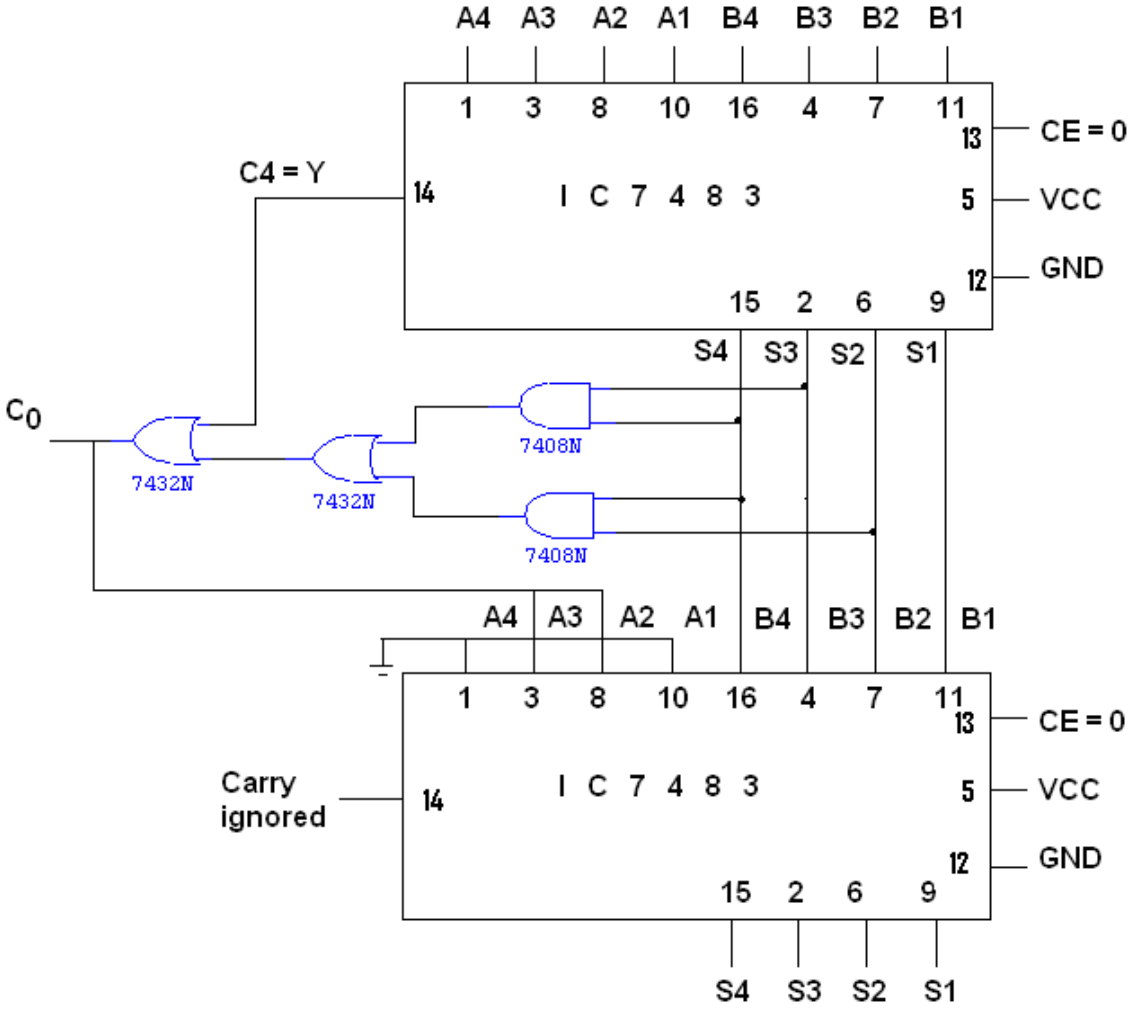
BCD SUM				CARRY
S4	S3	S2	S1	C
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

K MAP



C = S4 (S3 + S2)

LOGIC DIAGRAM:



RESULT:

The design of the 4-bit Binary adder and l subtractor circuit was done and its truth table was verified.

EXP NO. :05
DATE :

DESIGN AND IMPLEMENTATION OF MULTIPLEXER

AIM:

To design and implement multiplexer using logic gates.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P AND GATE	IC 7411	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	32

THEORY:

MULTIPLEXER:

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input line and n selection lines whose bit combination determine which input is selected.

DEMULTIPLEXER:

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer.

In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

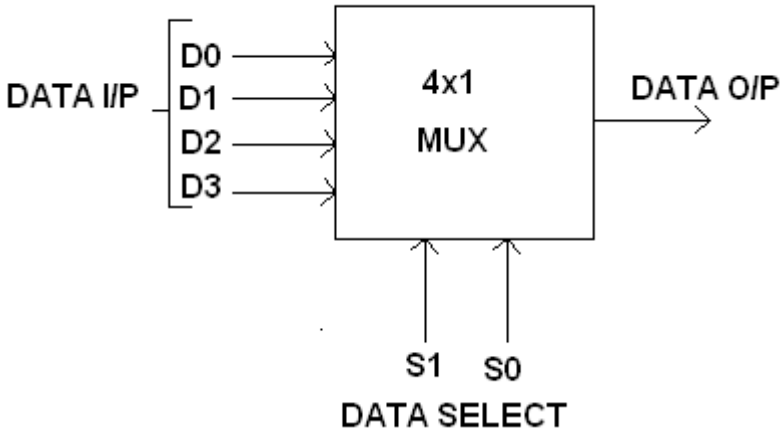
PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:

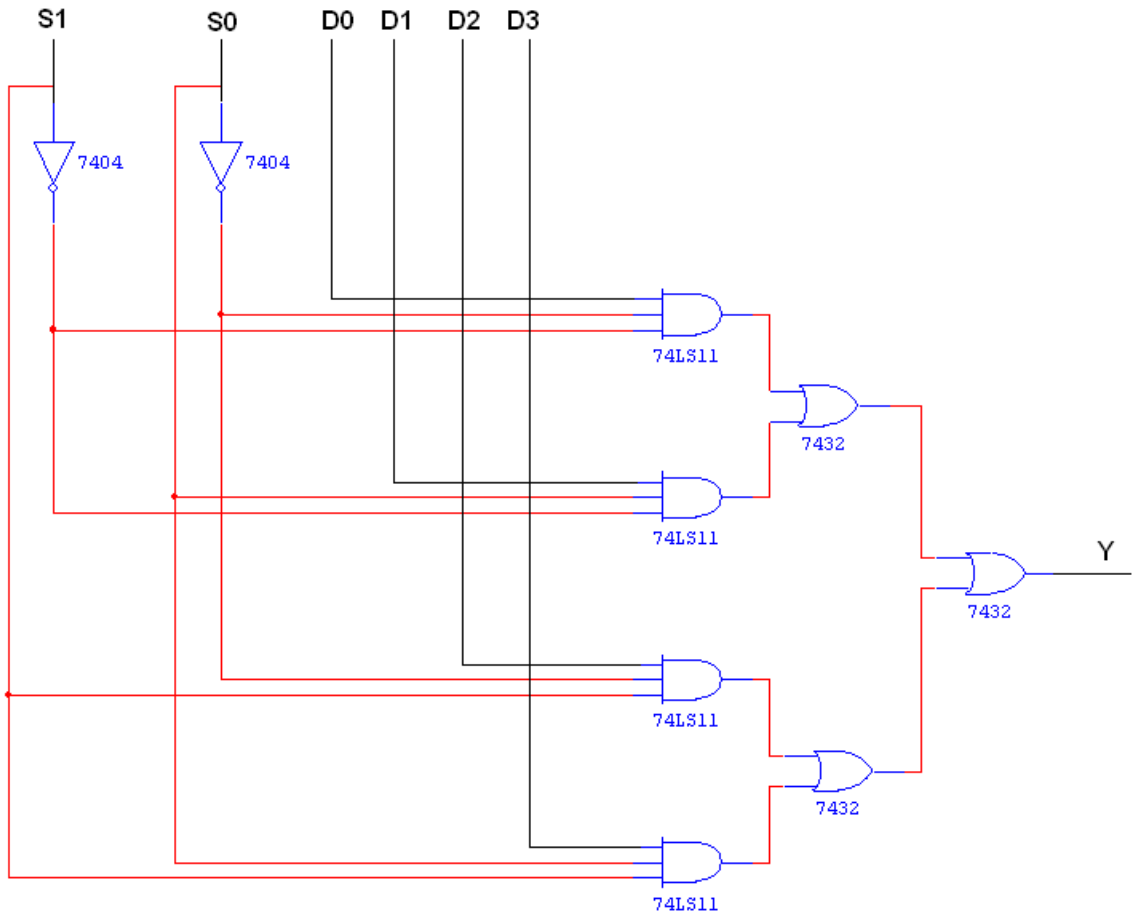


FUNCTION TABLE:

S1	S0	INPUTS Y
0	0	D0 → D0 S1' S0'
0	1	D1 → D1 S1' S0
1	0	D2 → D2 S1 S0'
1	1	D3 → D3 S1 S0

Y = D0 S1' S0' + D1 S1' S0 + D2 S1 S0' + D3 S1 S0

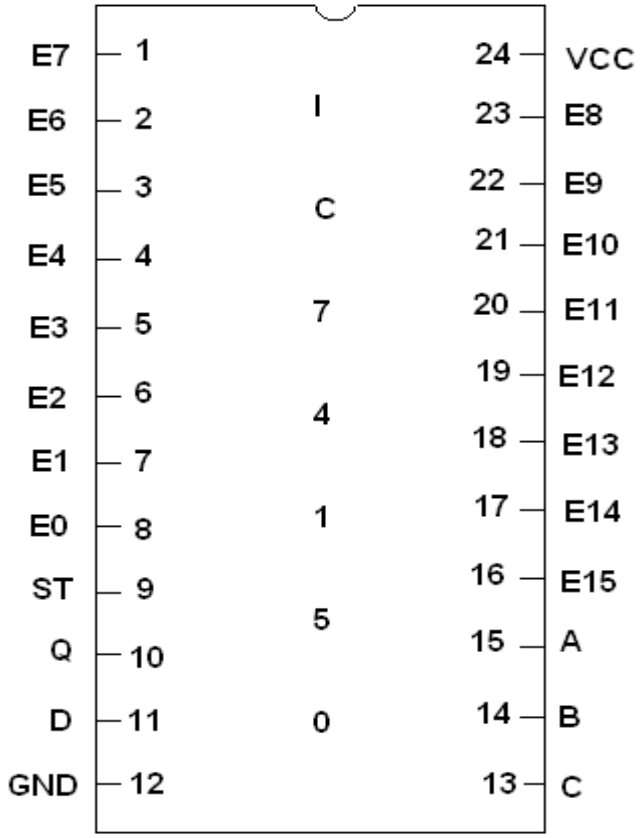
CIRCUIT DIAGRAM FOR MULTIPLEXER:



TRUTH TABLE:

S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

PIN DIAGRAM FOR IC 74150: MUX



RESULT:

The design of the 4x1 Multiplexer circuits was done and their truth tables were verified.

EXP NO. :06

DATE :

DESIGN AND IMPLEMENTATION OF ENCODER AND DECODER

AIM:

To design and implement encoder and decoder using logic gates and study of IC 7445 and IC 74147.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P NAND GATE	IC 7410	2
2.	OR GATE	IC 7432	3
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	27

THEORY:

ENCODER:

An encoder is a digital circuit that perform inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generates the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguila that when all inputs are zero the outputs are zero. The zero outputs can also be generated when D0 = 1.

DECODER:

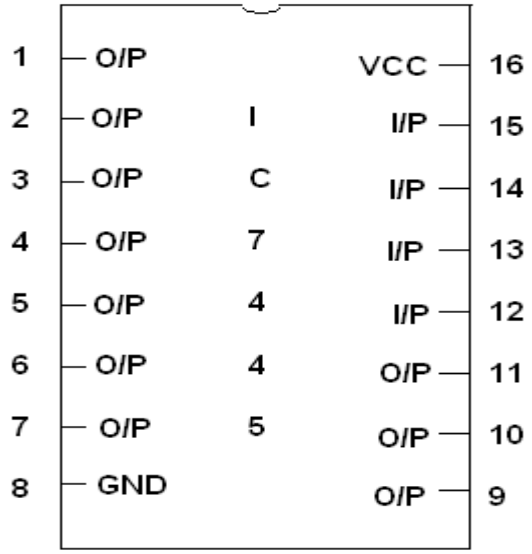
A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing 2^n possible outputs. 2^n output values are from 0 through out $2^n - 1$.

PROCEDURE:

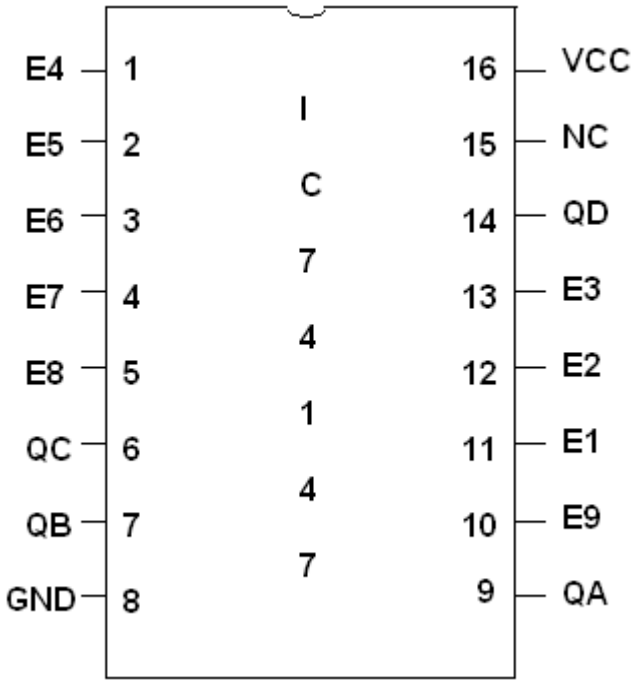
(i) Connections are given as per circuit diagram.
(ii) Logical inputs are given as per circuit diagram.
(iii) Observe the output and verify the truth table.

BCD TO DECIMAL DECODER:

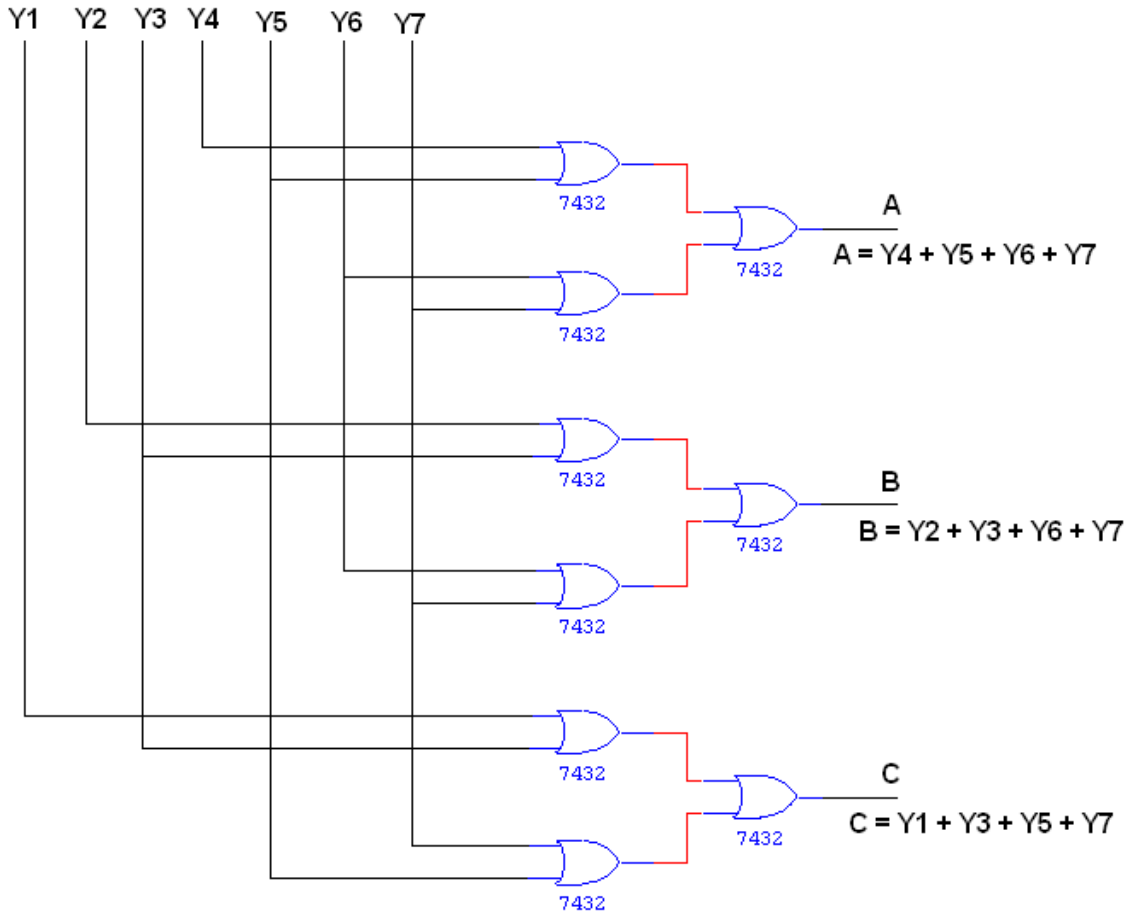
PIN DIAGRAM FOR IC 74155:2x4 Decoder



PIN DIAGRAM FOR IC 74147(Encoder)



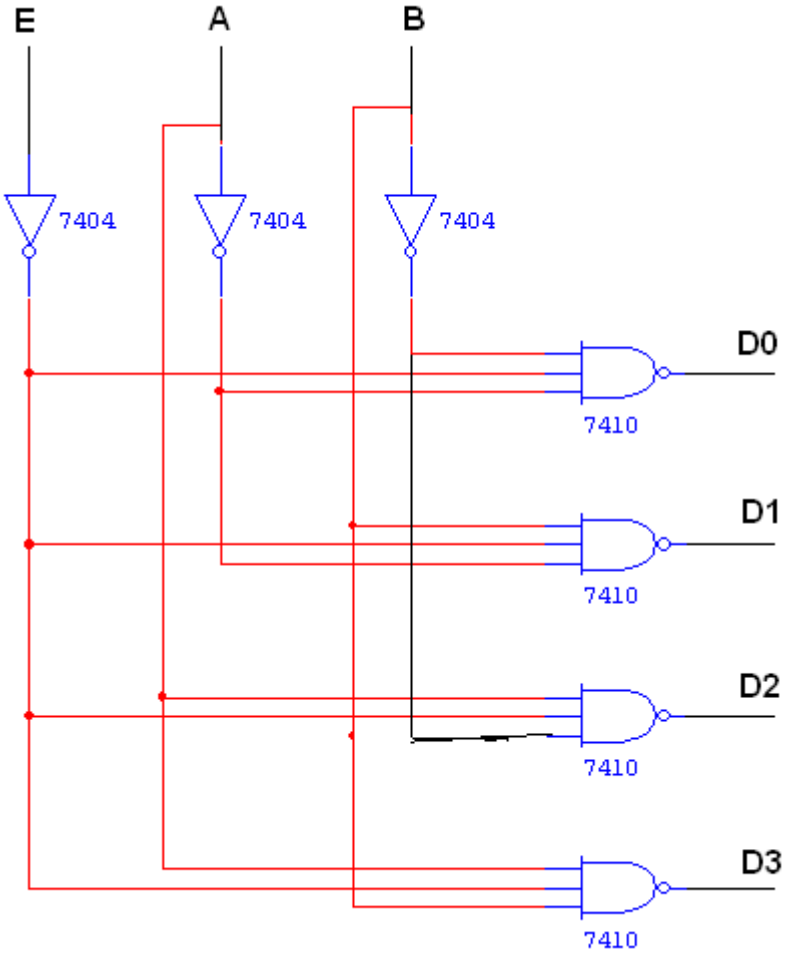
LOGIC DIAGRAM FOR ENCODER:



TRUTH TABLE:

INPUT							OUTPUT		
Y1	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**	**A**	**B**	**C**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

LOGIC DIAGRAM FOR DECODER:



TRUTH TABLE:

INPUT			OUTPUT			
E	**A**	**B**	**D0**	**D1**	**D2**	**D3**
1	**0**	**0**	**1**	**1**	**1**	**1**
0	**0**	**0**	**0**	**1**	**1**	**1**
0	**0**	**1**	**1**	**0**	**1**	**1**
0	**1**	**0**	**1**	**1**	**0**	**1**
0	**1**	**1**	**1**	**1**	**1**	**0**

RESULT:

The design of the Encoder and Decoder circuit was done and the input and output were obtained

EXP NO. :07

DATE :

DESIGN AND IMPLEMENTATION OF 3 BIT SYNCHRONOUS UP/DOWN COUNTER

AIM:

To design and implement 3 bit synchronous up/down counter.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	3 I/P AND GATE	IC 7411	1
3.	OR GATE	IC 7432	1
4.	XOR GATE	IC 7486	1
5.	NOT GATE	IC 7404	1
6.	IC TRAINER KIT	-	1
7.	PATCH CORDS	-	35

THEORY:

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. An up/down counter is one that is capable of progressing in increasing order or decreasing order through a certain sequence. An up/down counter is also called bidirectional counter. Usually up/down operation of the counter is controlled by up/down signal. When this signal is high counter goes through up sequence and when up/down signal is low counter follows reverse sequence.

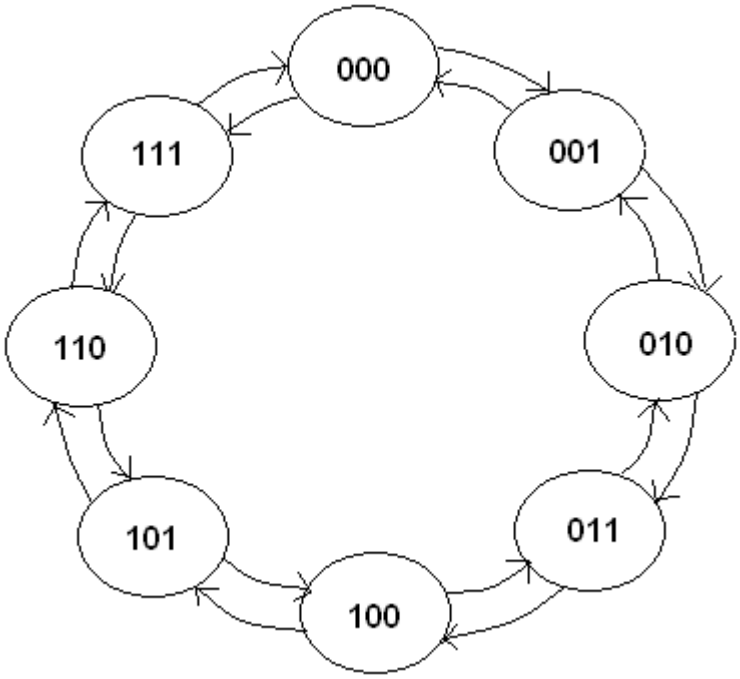
PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

STATE DIAGRAM:



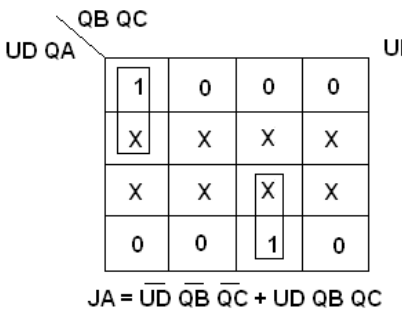
CHARACTERISTIC TABLE:

Q	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

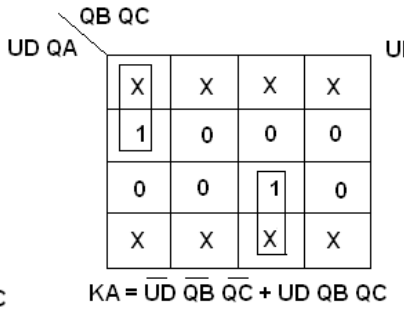
TRUTH TABLE:

Input	Present State			Next State			A		B		C	
Up/Down	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	1	1	X	1	X	1	X
0	1	1	1	1	1	0	X	0	X	0	X	1
0	1	1	0	1	0	1	X	0	X	1	1	X
0	1	0	1	1	0	0	X	0	0	X	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
0	0	1	1	0	1	0	0	X	X	0	X	1
0	0	1	0	0	0	1	0	X	X	1	1	X
0	0	0	1	0	0	0	0	X	0	X	X	1
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	1	0	1	X	0	0	X	1	X
1	1	0	1	1	1	0	X	0	1	X	X	1
1	1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	0	0	X	1	X	1	X	1

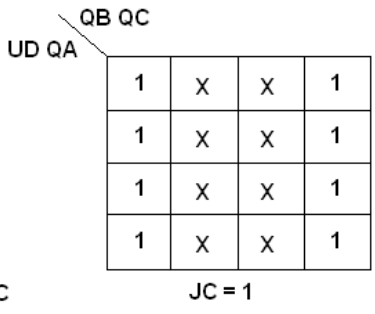
K MAP



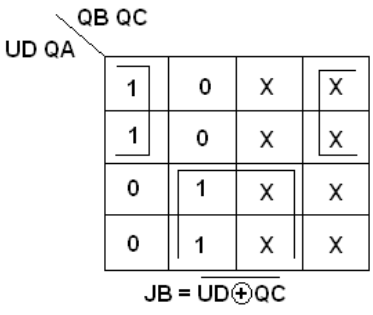
$JA = \overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$



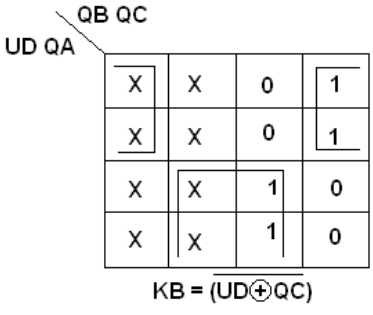
$KA = \overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$



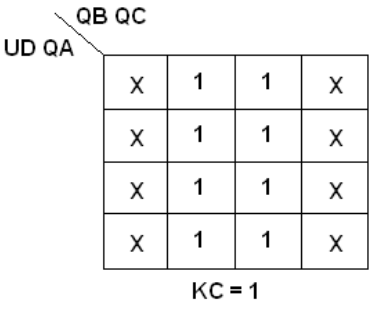
$JC = 1$



$JB = \overline{UD \oplus QC}$

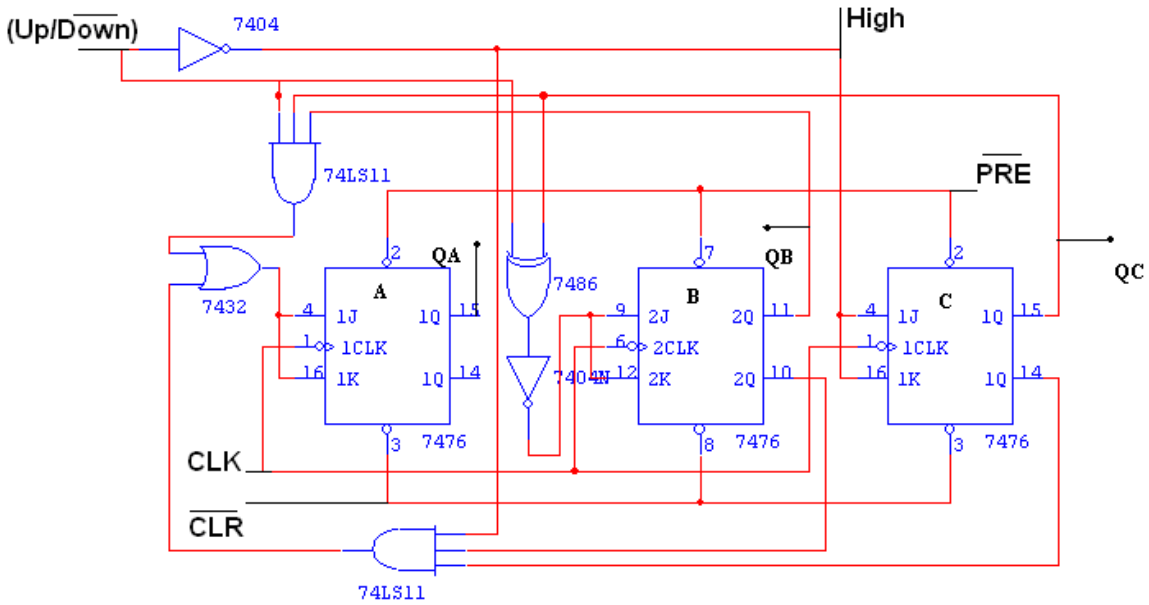


$KB = \overline{(UD \oplus QC)}$



$KC = 1$

LOGIC DIAGRAM:



RESULT:

Thus the 3-bit synchronous up/down counters was implemented successfully.

EXP NO. :08

DATE :

DESIGN AND IMPLEMENTATION OF SHIFT REGISTERS

AIM:

To design and implement

(i) Serial in serial out

(ii) Serial in parallel out

(iii) Parallel in serial out

(iv) Parallel in parallel out

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	35

THEORY:

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

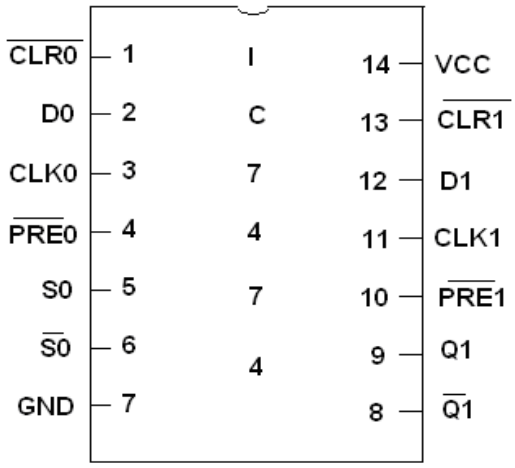
PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

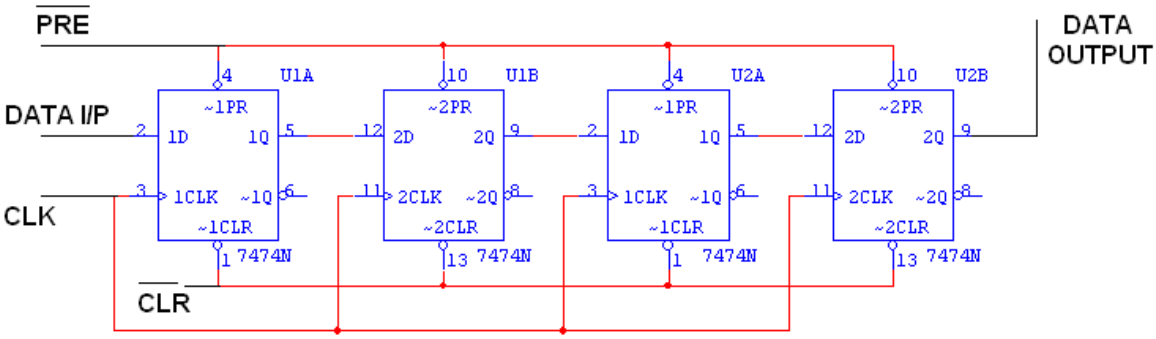
(iii) Observe the output and verify the truth table.

PIN DIAGRAM:



SERIAL IN SERIAL OUT:

LOGIC DIAGRAM:

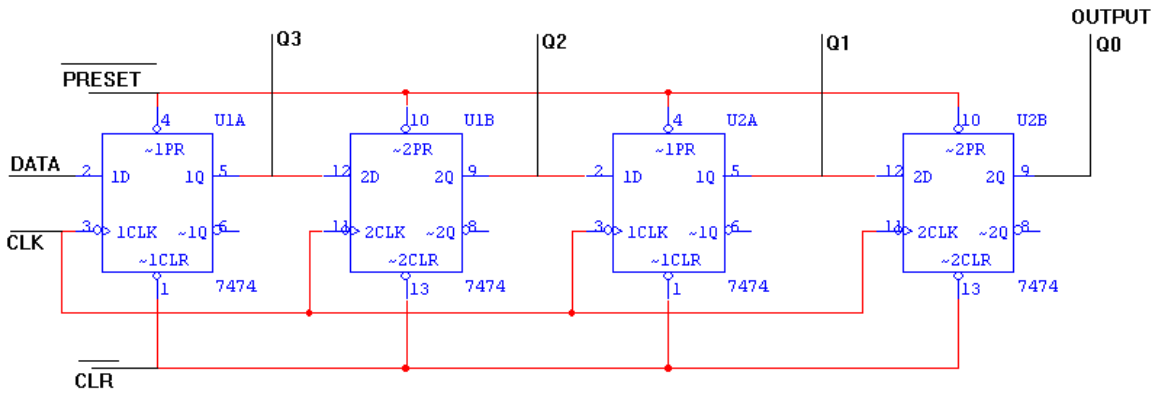


TRUTH TABLE:

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

SERIAL IN PARALLEL OUT:

LOGIC DIAGRAM:

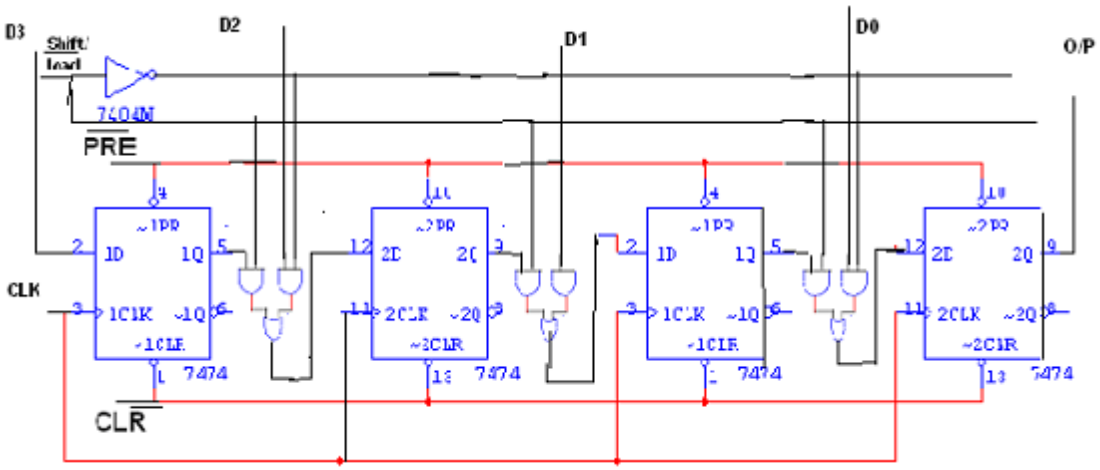


TRUTH TABLE:

CLK	DATA	OUTPUT			
		Q3	Q2	Q1	Q0
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	1	1	0	0	1

PARALLEL IN SERIAL OUT:

LOGIC DIAGRAM:

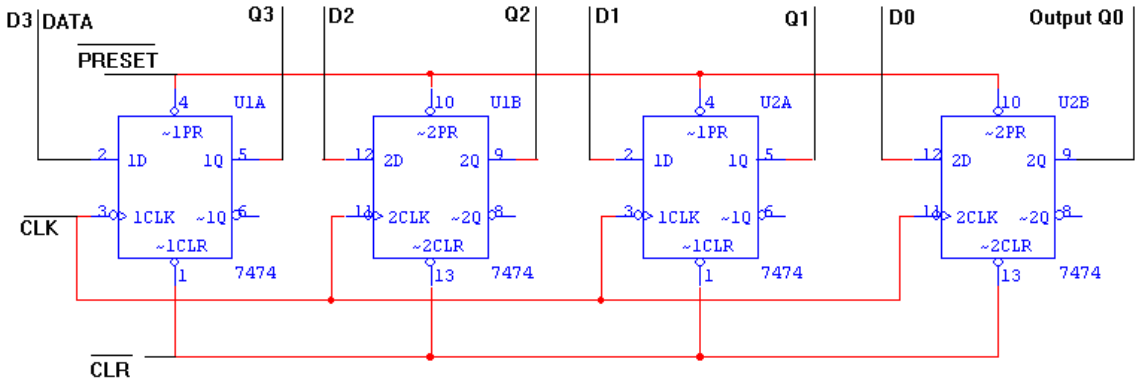


TRUTH TABLE:

CLK	D3	D2	D1	D0	OUTPUT
0	1	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	1

PARALLEL IN PARALLEL OUT:

LOGIC DIAGRAM:



TRUTH TABLE:

CLK	DATA INPUT				OUTPUT			
	D3	D2	D1	D0	Q3	Q2	Q1	Q0
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

RESULT:

Thus the implementation of shift registers using flip flops was completed successfully.

EXP NO. : 09 **DESIGN OF ADDER AND SUBTRACTOR**

DATE :

AIM:

To design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using logic gates.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	23

THEORY:

HALF ADDER:

A half adder has two inputs for the two bits to be added and two outputs one from the sum ‘ S’ and other from the carry ‘ c’ into the higher adder position. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR Gate the carry out from the AND gate.

FULL ADDER:

A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

HALF SUBTRACTOR:

The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.

FULL SUBTRACTOR:

The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor. The expression AB assembles the borrow output of the half subtractor and the second term is the inverted difference output of first X-OR.

PROCEEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

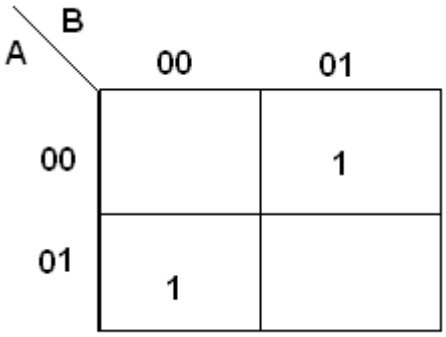
(iii) Observe the output and verify the truth table.

HALF ADDER

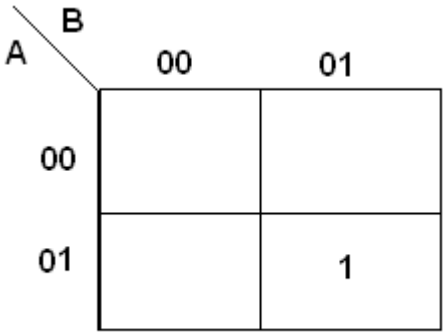
TRUTH TABLE:

A	B	CARRY	SUM
0 0 1 1	0 1 0 1	0 0 0 1	0 1 1 0

K-Map for SUM:



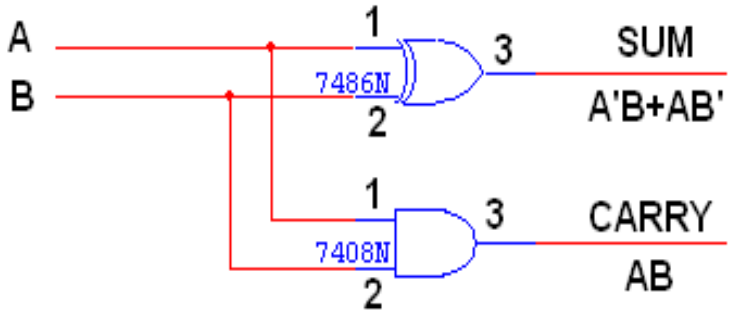
K-Map for CARRY:



SUM = A'B + AB'

CARRY = AB

LOGIC DIAGRAM:

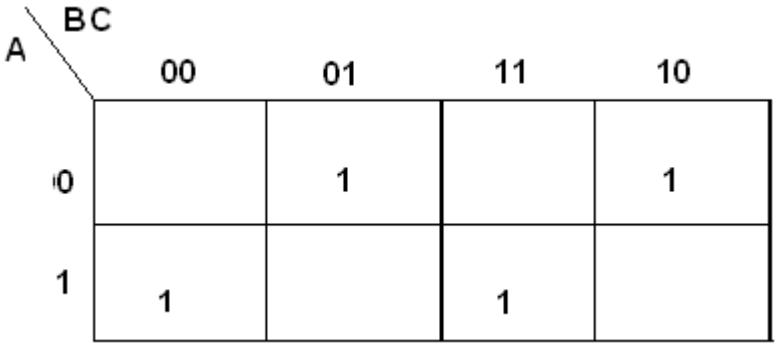


FULL ADDER

TRUTH TABLE:

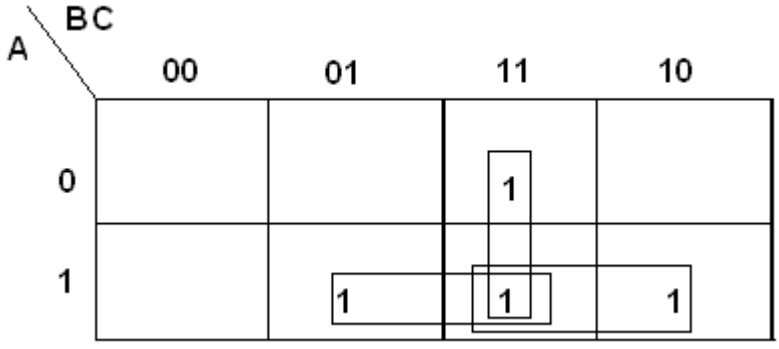
A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-Map for SUM:



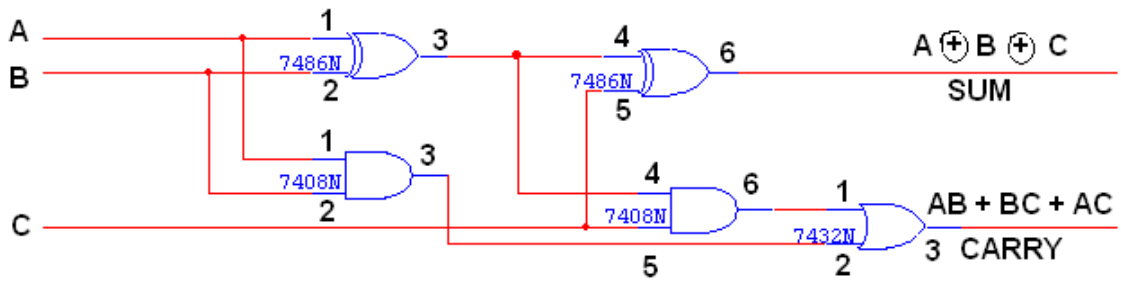
SUM = A'B'C + A'BC' + AB'C' + ABC

K-Map for CARRY:

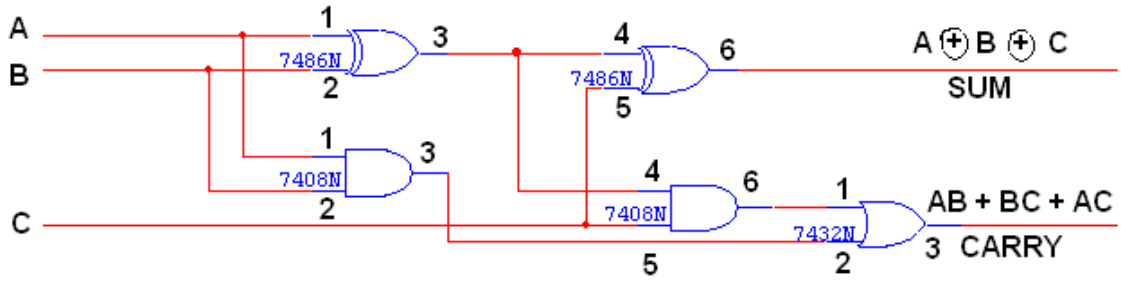


CARRY = AB + BC + AC

LOGIC DIAGRAM:



FULL ADDER USING TWO HALF ADDERS

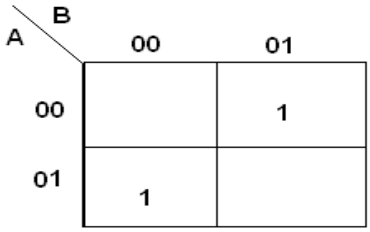


HALF SUBTRACTOR

TRUTH TABLE:

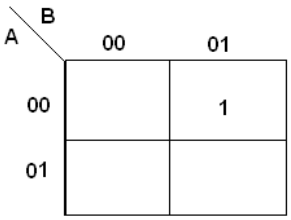
A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K-Map for DIFFERENCE:



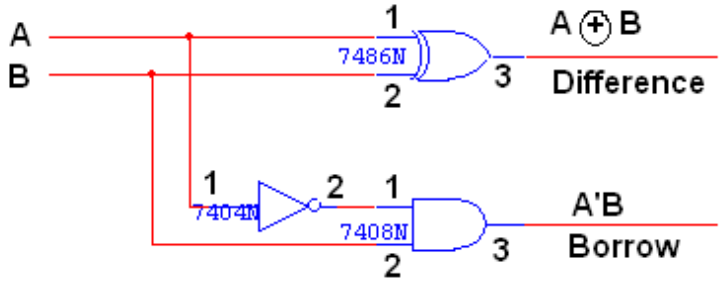
DIFFERENCE = A'B + AB'

K-Map for BORROW:



BORROW = A'B

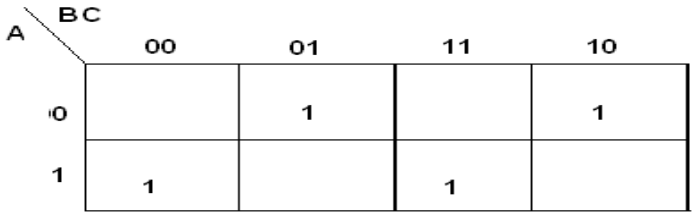
LOGIC DIAGRAM:



FULL SUBTRACTOR

TRUTH TABLE:

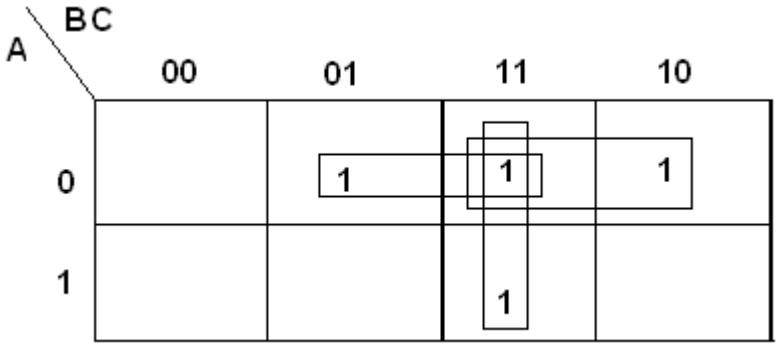
A	B	C	BORROW	DIFFERENCE
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



K-Map for Difference:

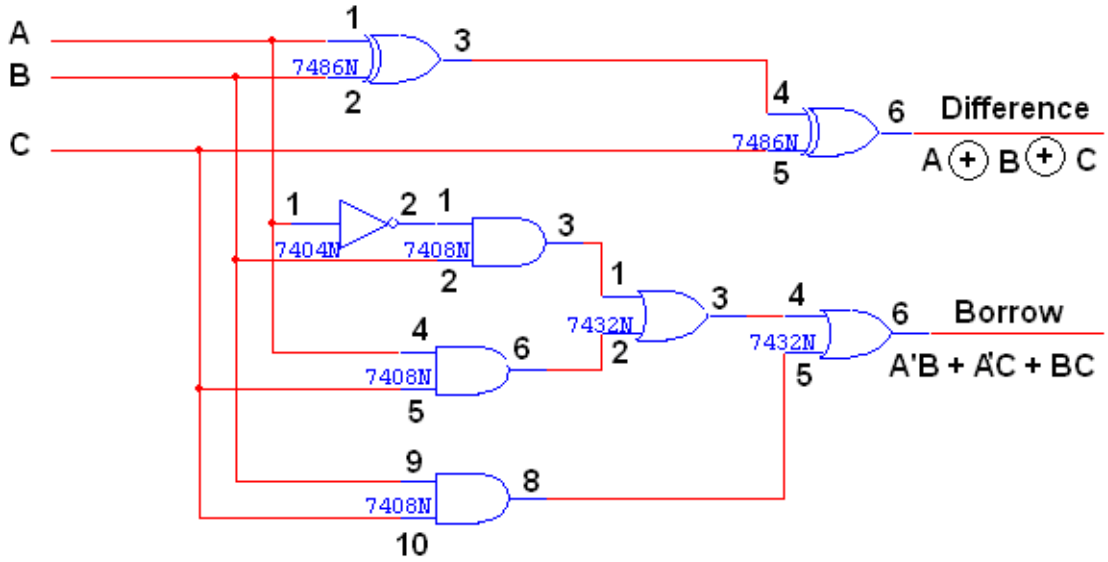
Difference = A'B'C + A'BC' + AB'C' + ABC

K-Map for Borrow:

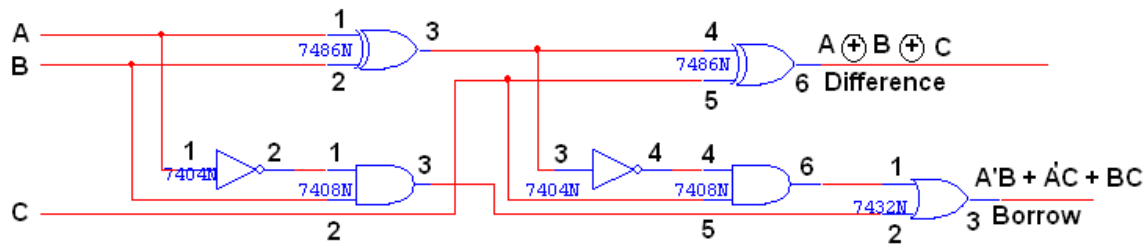


Borrow = A'B + BC + A'C

LOGIC DIAGRAM:



FULL SUBTRACTOR USING TWO HALF SUBTRACTORS:



RESULT:

The design of the half adder, full adder and half subtractor and full subtractor circuits was done and their truth tables were verified.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3361 ELECTRONIC DEVICES AND CIRCUITS LABORATORY

Semester - 03

LABORATORY MANUAL

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

1. Characteristics of PN Junction Diode and Zener diode.
2. Full Wave Rectifier with Filters.
3. Design of Zener diode Regulator.
4. Common Emitter input-output Characteristics.
5. MOSFET Drain current and Transfer Characteristics.
6. Frequency response of CE and CS amplifiers.
7. Frequency response of CB and CC amplifiers.
8. Frequency response of Cascode Amplifier
9. CMRR measurement of Differential Amplifier
10. Class A Transformer Coupled Power Amplifier.

EXPT. NO:

DATE :

CHARACTERISTICS OF PN JUNCTION DIODE

Aim:

To determine the forward and reverse characteristics of the given PN junction diode and to determine cut-in voltage

Apparatus required:

Sl. No.	Description	Range	Quantity
1	Regulated Power Supply	(0-15)V	2
2	Ammeter	0-20m A , 0-200uA	Each 1
3	Voltmeter	(0-20)V	2
4	Diode	IN 4007	2
5	Resistor	1kΩ	2
6	Bread board	---	2
7	Connecting Wires	---	As per required

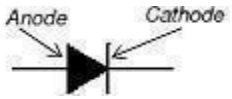
Theory:

Donor impurities (pentavalent) are introduced into one-side and acceptor impurities(trivalent) into the other side of a single crystal of an intrinsic semiconductor to form a PN junction diode with a junction called depletion region (this region is depleted off the charge carriers). This region gives rise to a potential barrier called cut-in Voltage. This is the voltage across the diode at which it starts conducting. The PN junction can conduct beyond this potential. The PN junction supports unidirectional current flow. If positive terminal of the input supply is connected to anode (P-side) and negative terminal of the input supply is connected the cathode Then diode is said to be forward biased.

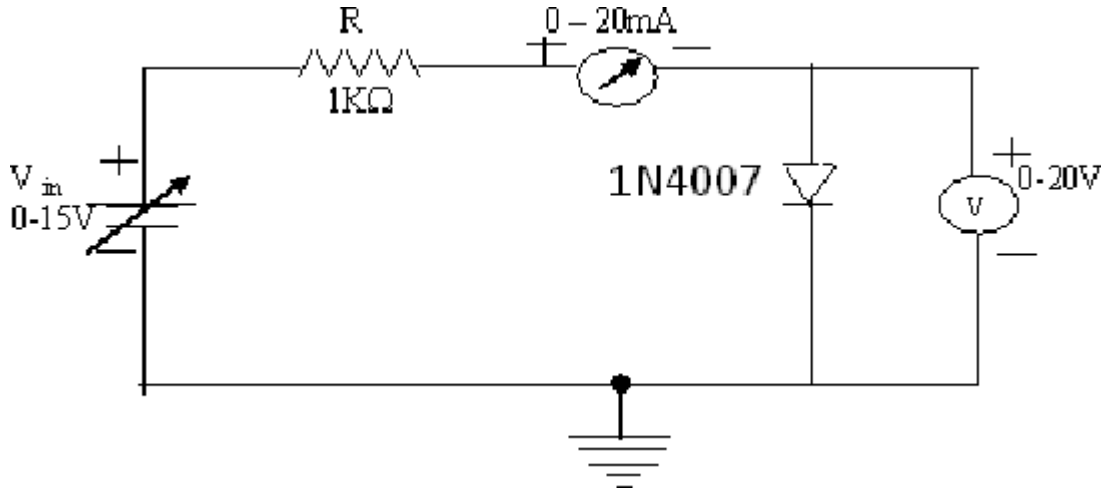
If negative terminal of the input supply is connected to anode (p-side) and positive terminal of the input supply is connected to cathode (n-side) then the diode is said to be reverse biased. On forward biasing, initially no current flows due to barrier potential. As the applied potential exceeds the barrier potential the charge carriers gain sufficient energy to cross the potential barrier and hence enter the other region. On reverse biasing, the majority charge carriers are attracted towards the terminals due to the applied potential resulting in the widening of the depletion region. Since the charge carriers are pushed towards .

Terminals no current flows through the device due to majority charge carriers. There will be some current in the device due to minority carriers. The generation of such carriers is independent of the applied potential and hence the current is constant for all increasing reverse potential. This current is referred to as reverse saturation current (IO) and it increases with temperature.

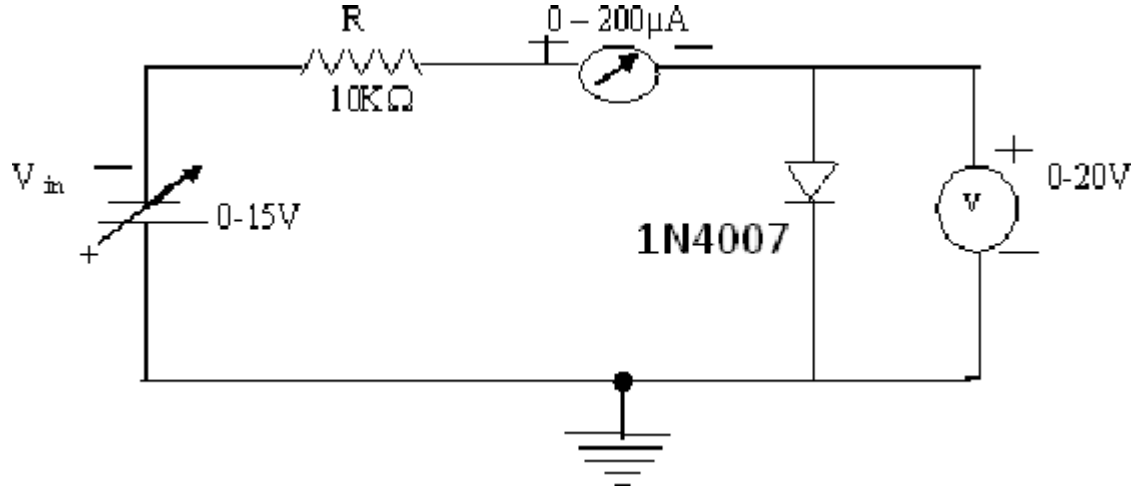
PN Junction Diode Symbol :



Circuit Diagram:Forward Bias



Reverse Bias:



Procedure:

1. Connect the circuit as per the circuit diagram.
2. Switch on the power supply.
3. Vary the power supply voltage step by step from zero volt.
4. Take the voltmeter and ammeter readings for every variation of power supply.
5. Re-Connect the circuit for reverse bias condition as shown in figure.
6. Repeat the step 3 and 4 for reverse bias.
7. Draw the graph for forward bias and reverse bias for PN junction diode.
8. Note down cut-in voltage for PN junction diode.
9. Switch of the power supply.

Tabulation :

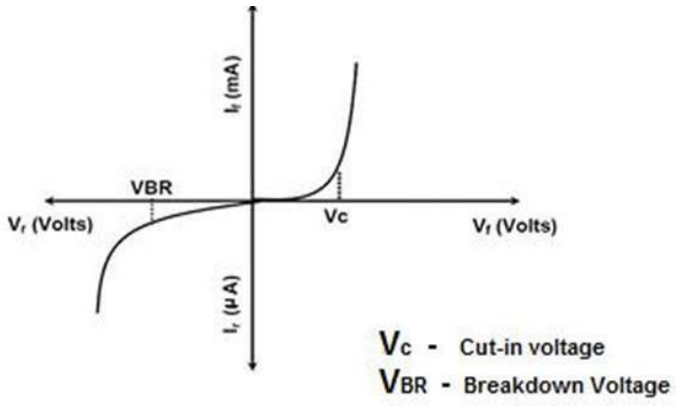
Reverse Bias:

SL.NO	Voltmeter(v)	Current (m A)

Forward bias:

SL.NO	Voltage(v)	Current (m A)

Model Graph:



Result:

Thus the forward and reverse characteristics of the given PN junction diode is determined.

EXPT. NO:
DATE :

CHARACTERISTICS OF ZENER DIODE

Aim:

To determine the breakdown voltage of a given zener diode.

Apparatus Required:

S.No	Description	Range	Quality
1	RPS	(0-15) V	2
2	ammeter	(0-200) mA	2
3	voltmeter	(0-15)V	2
4	Zener Diode	4148	2
5	Connecting wires	-	As per Required
6	Resister	1KΩ	2
7	Bread Board	-	2

Theory:

A properly doped crystal diode, which has a sharp breakdown voltage, is known as zener diode.

1. Forward Bias:

On forward biasing, initially no current flows due to barrier potential. As the applied potential increases, it exceeds the barrier potential at one value and the charge carriers gain sufficient energy to cross the potential barrier and enter the other region. the holes ,which are majority carriers in p-region, become minority carriers on entering the N-regions and electrons, which are the majority carriers in the N-regions become minority carriers on entering the P- region. This injection of minority carriers results current, opposite to the direction of electron movement.

2. Reverse Bias:

When the reverse bias is applied due to majority carriers small amount of current (ie) reverse saturation current flows across the junction. As the reverse bias is increased to breakdown voltage, sudden rise in current takes place due to zener effect.

Zener Effect :

Normally, PN junction of Zener Diode is heavily doped. Due to heavy doping the depletion layer will be narrow. When the reverse bias is increased the potential across the depletion layer is more. This exerts a force on the electrons in the outermost shell. Because of this force the electrons are pulled away from the parent nuclei and become free electrons.

This ionization, which occurs due to electrostatic force of attraction, is known as Zener effect. It results in large number of free carriers, which in turn increases the reverse saturation current

Procedure:

Forward Bias:

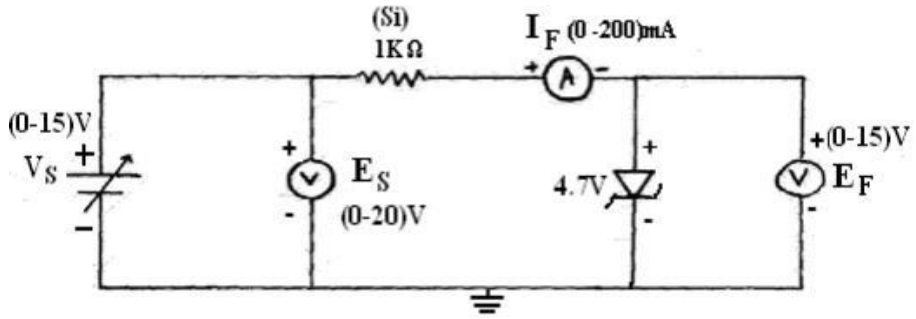
1. Connect the circuit as per the circuit diagram.
2. Vary the power supply in such a way that the readings are taken in steps of 0.1V in the voltmeter till the needle of power supply shows 30V.
3. Note down the corresponding ammeter readings.
4. Plot the graph :V (vs) I.
5. Find the dynamic resistance **r** = $\Delta V / \Delta I$

Reverse Bias:

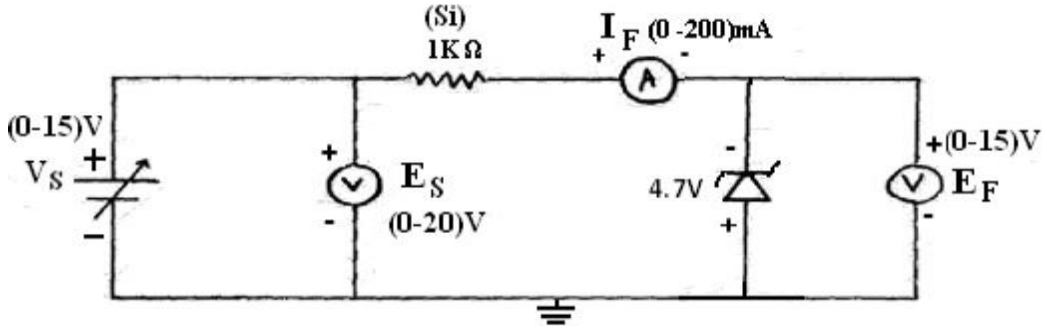
1. Connect the circuit as per the diagram
2. Vary the power supply in such a way that the readings are taken in steps of 0.1V in the voltmeter till the needle of power supply shows 30V.
3. Note down the corresponding Ammeter readings I.
4. Plot a graph between V & I
5. Find the dynamic resistance **r** = $\Delta V / \Delta I$
6. Find the reverse voltage Vr at Iz=20 mA

Circuit Diagram:

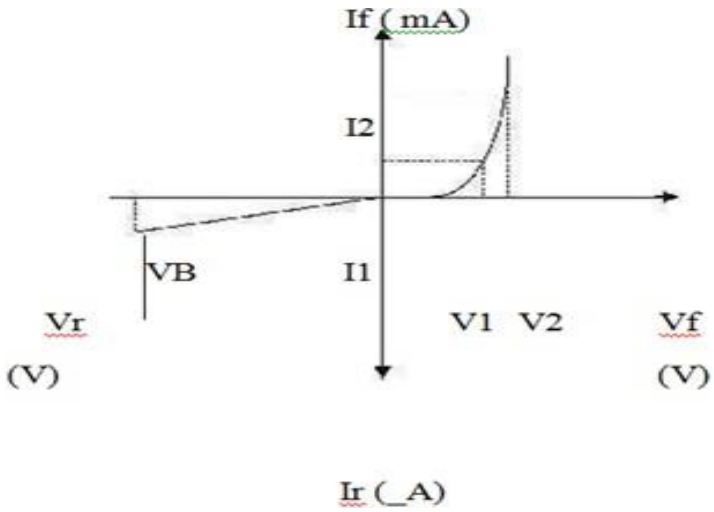
Forward Bias:



Reverse Bias



Model Graph



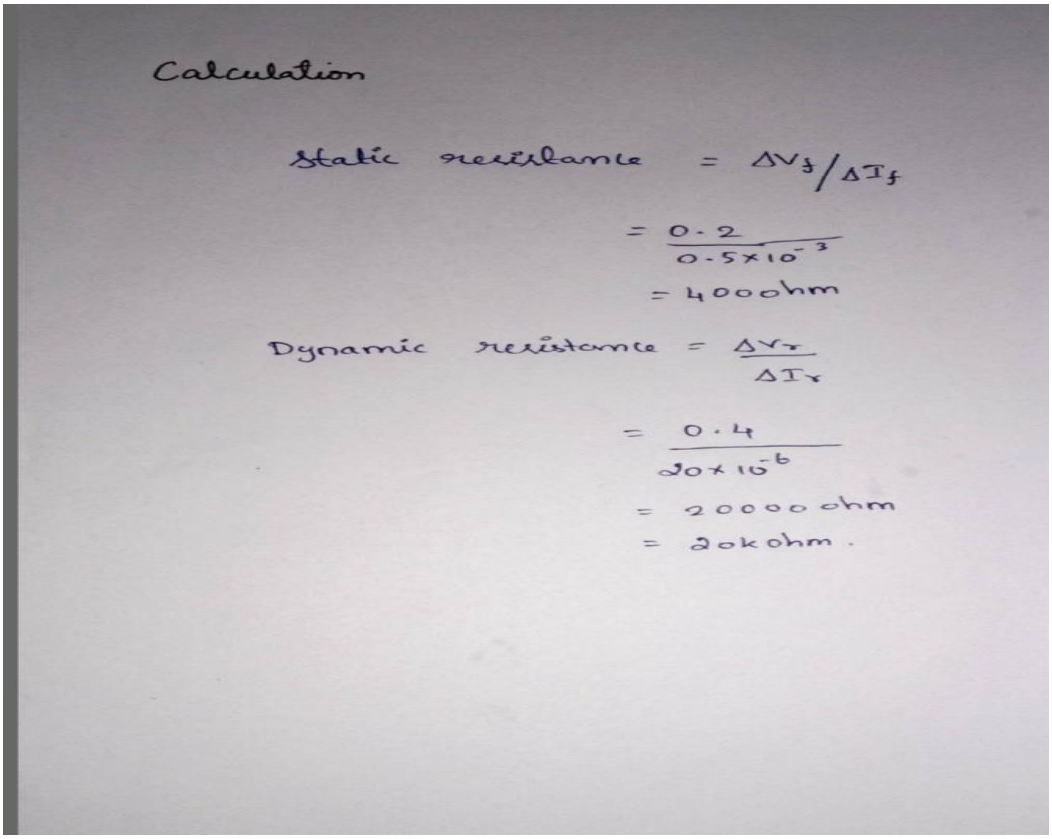
Tabular Column

Forward Bias

S..No.	VOLTAGE (In Volts)	CURRENT (In mA)

Reverse Bias

S..No.	VOLTAGE (In Volts)	CURRENT (In mA)



RESULT:

Forward and Reverse bias characteristics of the zener diode was studied and

i. Forward bias dynamic resistance =400Ω

ii. Reverse bias dynamic resistance = 20kΩ

EXPT. NO:
DATE :

FULL WAVE RECTIFIER

Aim:

To study the operation of Full- Wave Rectifier with and without filter and to find its:

a. Percentage Regulation
b. Ripple Factor
c. Efficiency

Apparatus Required :

S.no	Description	Range	Quantity
1	Diodes	1N4007	2
2	Resistor	1kΩ	1
3	Capacitor	100µf	1
4	Transformer with Center TappedSecondary	(9 - 0 - 9) V	1
5	CRO	30MHz	1
6	Digital Multimeter/Digital Voltmeter	(0-20V)	1
7	Bread Board	-	1
8	Connecting Wires	Single strand	As per required

Operation:

The conversion of AC into pulsating DC is called Rectification. Electronic Devices can convert ACpower into DC power with high efficiency. The full-wave rectifier consists of a center-tapped transformer, which results in equal voltages above and below the center-tap. During the positive half cycle, a positive voltage appears at the anode of D1 while a negative voltage appears at the anode of D2. Due to this diode D1 is forward biased. It results a current Id1 through the load R.

During the negative half cycle, a positive voltage appears at the anode of D2 and hence it is forward biased, resulting a current Id2 through the load. At the same instant a negative voltage appears at the anode of D1, reverse biasing it and hence it doesn't conduct.

Ripple Factor :

Ripple factor is defined as the ratio of the effective value of AC components to the average DC value.It is denoted by the symbol 'γ.

$$\gamma = \frac{V_{ac}}{V_{dc}}, (\gamma = 0.48)$$

Rectification Factor:

The ratio of output DC power to input AC power is defined as efficiency.

$$\eta = \frac{(V_{dc})^2}{(V_{ac})^2}$$

≒ 81% (if R >> R_f, then R_f can be neglected).

Percentage of Regulation:

It is a measure of the variation of DC output voltage as a function of DC output current (i.e., variation in load).

$$\text{Percentage of regulation} = \left(\frac{V_{NL} - V_{FL}}{V_{FL}}\right) * 100 \%$$

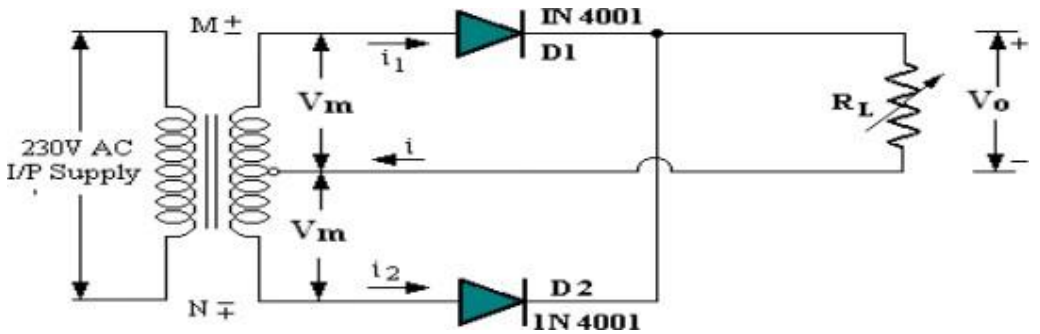
VNL = Voltage across load resistance, when minimum current flows through it. VFL = Voltage across load resistance, when maximum current flows through. For an ideal Full-wave rectifier, the percentage regulation is 0 percent. The percentage of regulation isvery small for a practical full wave rectifier.

Peak- Inverse - Voltage (PIV):

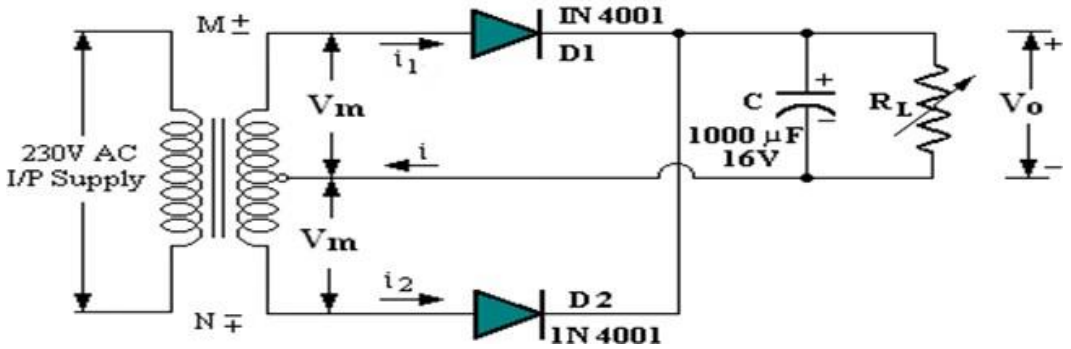
It is the maximum voltage that the diode has to withstand when it is reverse biased.
PIV = 2Vm

Full Wave Rectifier Circuit Diagram

Without Filter



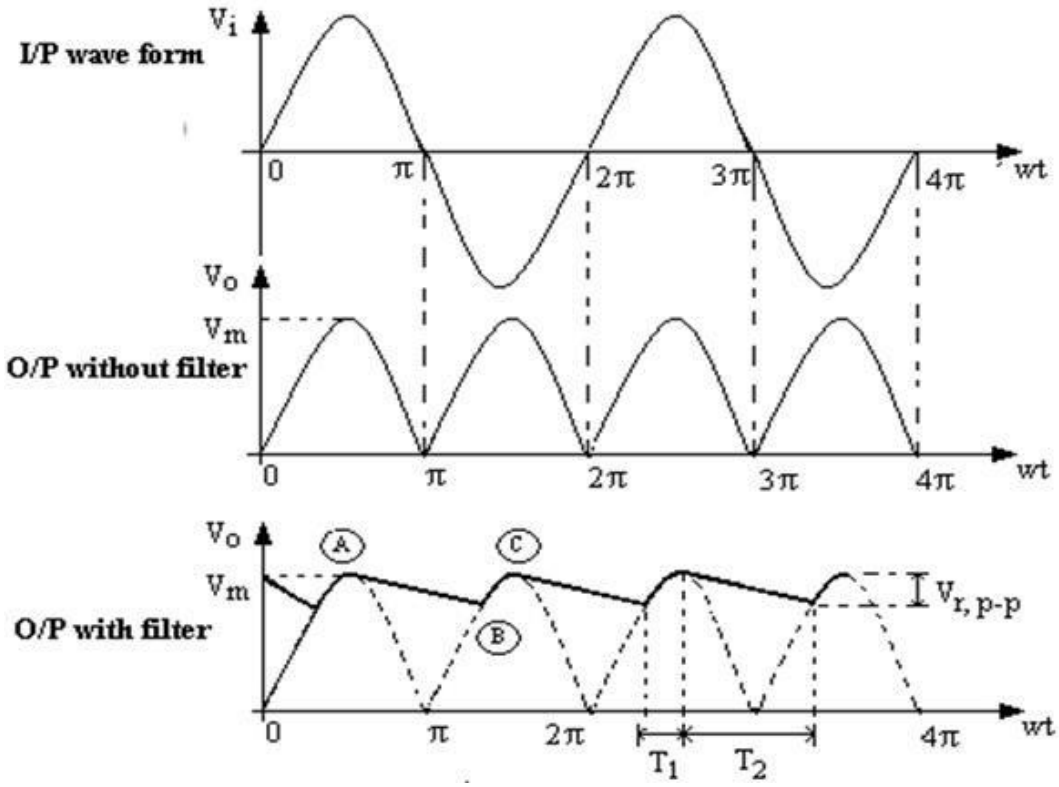
With Filter



Procedure:

1. Connect the circuit as shown in the circuit diagram.
2. Connect the primary side of the transformer to AC mains and the secondary side to rectifierinput.
3. Using a CRO, measure the maximum voltage V_m of the AC input voltage of the rectifier andAC voltage at the output of the rectifier.
4. Using a DC voltmeter, measure the DC voltage at the load resistance.
5. Observe the Waveforms at the secondary windings of transformer and across load resistance for a load of 1KΩ.
6. Calculate the ripple factor ( ), percentage of regulation and efficiency ( ) as per the below given formulae.

Model Graph:



Tabulation
Without Filter:

Vm	Vrms	Vds	Ripple factor	Efficiency

With Filter

Vrms	Vds	Ripple factor	Efficiency

Result :

Thus the characteristics of full wave rectifier were studied.

EXPT. NO:
DATE :

DESIGN OF ZENER DIODE REGULATOR
(ZENER DIODE AS VOLTAGE REGULATOR.)

Aim:

Measurement of percentage regulation by varying load resistor.

Apparatus Required:

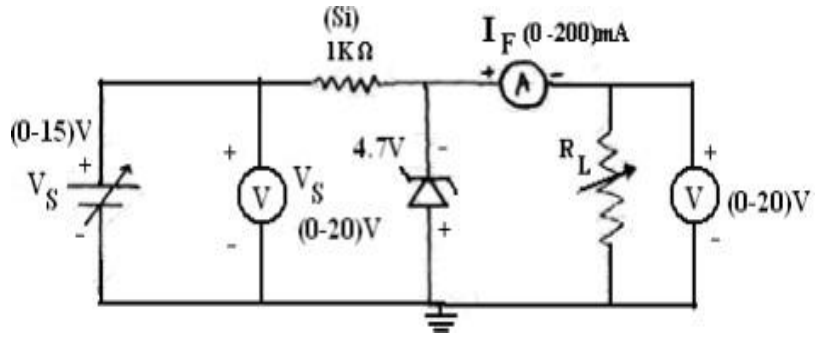
S.No	Description	Range	Quality
1	Resistor	1KΩ	1
2	Voltmeter	0-20V	2
3	Power Supply	0-15 V	1
4	Zener Diode	4148	1
5	Resister Load	-	-

Theory:

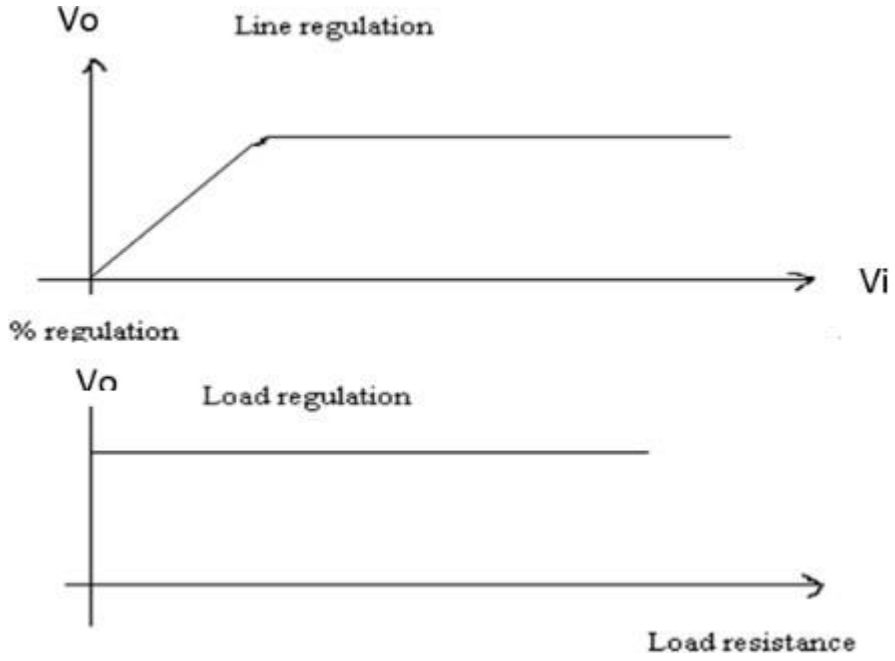
Zener diode is a P-N junction diode specially designed to operate in the reverse biased mode. It is acting as normal diode while forward biasing. It has a particular voltage known as break down voltage, at which the diode break downs while reverse biased. In the case of normal diodes the diode damages at the break down voltage. But Zener diode is specially designed to operate in the reverse breakdown region. The basic principle of Zener diode is the Zener breakdown. When a diode is heavily doped, it's depletion region will be narrow.

When a high reverse voltage is applied across the junction, there will be very strong electric field at the junction. And the electron hole pair generation takes place. Thus heavy current flows. This is known as Zener break down. So a Zener diode, in a forward biased condition acts as a normal diode. In reverse biased mode, after the break down of junction current through diode increases sharply. But the voltage across it remains constant. This principle is used in voltage regulator using Zener diodes.

CIRCUIT DIAGRAM:



Graph:



Procedure:

Input Characteristics:

1. Varying the input voltage keeping load constant: Connect the circuit as showin in fig.1. Keep supply control at minimum.

2. Keep the load RL at 750Ωs for Q-pont. Increase the input voltage VS in step of 1Volt and note V1 and V2. Where V1 is the input and V2 is the output voltage across zener.

3. Plot the curves between input –output at load constant. Find out the δV1 and δV2 from the plot and calculate the line regulation.

Output characteristics:

1. Varying the load keeping input voltage constant: Connect the circuit as showin in fig.1. Keep supply control at minimum.
2. Keep load RL at 3000Ωs. Increase the input voltage V1 to 12 Vdc.
3. Decrease the load and note the voltage V2 with load value.
4. Plot the curves between load and output voltage at input constant. Find out the δV2 and VZ at Q point at set load value from the input plot and calculate load regulation.

Tabulation:

Shunt regulation (Load regulation), Input voltage constant at 2.5 V

S.No	Output Voltage (V)	Load (Ω)

Shunt regulation (Line regulation), load constant at 2Ω

S.No	Input Voltage (V)	Output Voltage (V)

RESULT:

Realization of zener diode by voltage regulator is verified

EXPT. NO:
DATE :

COMMON EMITTER INPUT-OUTPUT CHARACTERISTICS.

Aim:

To obtain the input and output characteristics of the given transistor in common emitter configuration.

Apparatus required:

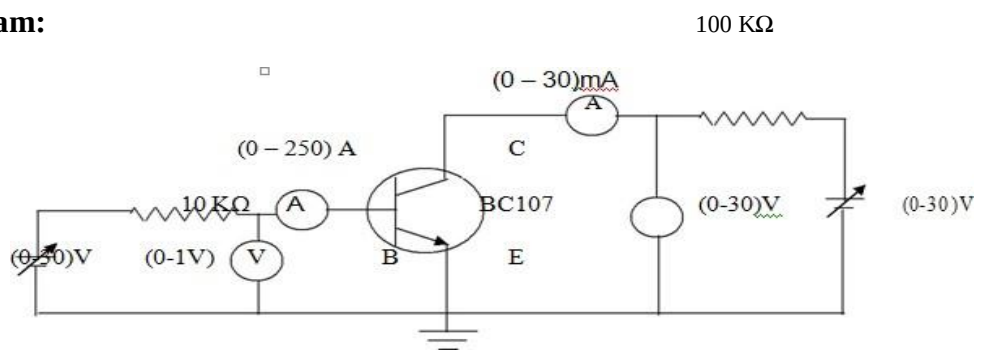
Sl. No.	Description	Range	Quantity
1	Regulated Power Supply	(0-30)V	1
2	Ammeter	(0-30)mA,(0-250)A	Each 1
4	Voltmeter	(0-1)V,(0-30)V	Each 1
5	NPN Transistor	50V,1A,3W	1
6	Resistor	1kΩ	1
7	Connecting Wires	-	As per required

Theory:

A NPN function transistor consist of a silicon (or germanium) crystal in which a layer of p – type silicon is sandwiched between two layers of N – type silicon. The arrow on emitter lead specifies the direction of the current flow when the emitter – base function is forward biased. As the conductivity of the BJT depends on both the majority and minority carriers it is called bipolar device. In CE configuration, Emitter is common to both the Emitter and Base.

A transistor can be in any of the three configurations viz, Common base, Common emitter and Common Collector .The transistor consists of three terminal emitter, collector and base. The emitter layer is the source of the charge carriers and it is heavily doped with a moderate cross sectional area. The collector collects the charge carries and hence has moderate doping and large cross sectional area. The base region acts a path for the movement of the charge carriers. In order to reduce the recombination of holes and electrons the base region is lightly doped and is of hollow cross sectional area. Normally the transistor operates with the emitter base junction forward biased. In transistor, the current is same in both junctions, which indicates that there is a transfer of resistance between the two junctions which is known as transfer resistance of transistor.

Circuit Diagram:



Precautions:

1. While performing the experiment do not exceed the ratings of the transistor. This may lead to the transistor damage.
2. Connect voltmeter and ammeter in correct polarities as shown in the circuit diagram.
3. Do not switch ON the power supply unless you have checked the circuit connections as per the circuit diagram.
4. Make sure while selecting the emitter, base and collector terminals of the transistor.

Procedure:

Input Characteristics:

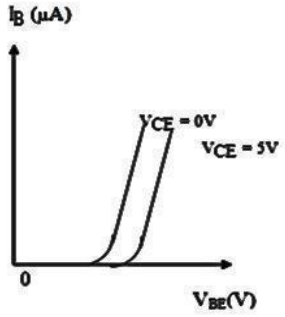
1. Connect the circuit as per the circuit diagram.
2. Set VCE, vary VBE in regular interval of steps and note down the corresponding IB reading.
3. Repeat the above procedure for different values of VCE.
4. Plot the graph: VBE vs. IB for a constant VCE.

Output Characteristics:

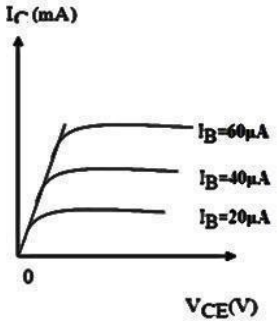
1. Connect the circuit as per the circuit diagram.
2. Set IB, Vary VCE in regular interval of steps and note down the corresponding IC reading.
3. Repeat the above procedure for different values of IB.
4. Plot the graph: VCE vs. IC for a constant IB.
5. Switch of the power supply.
6. Disconnect the components.

Model graph:

Input Characteristics:



Output Characteristics:



Input Characteristics:

V_{ce}= 1V		V_{ce}= 2V	
V_{be}(v)	I_s(m A)	V_{be}(v)	I_s(m A)

Output characteristics:

I_b= 5V		I_b= 10V	
V_{ce}(v)	I_c(mA)	V_{ce}(v)	I_c(m A)

Result:

The transistor characteristics of a Common Emitter (CE) configuration were plotted.

EXPT. NO:
DATE :

MOSFET DRAIN AND TRANSFER CHARACTERISTICS

Aim:

To plot the Transfer and Drain characteristics of MOSFET and determine Transconductance and output Resistance.

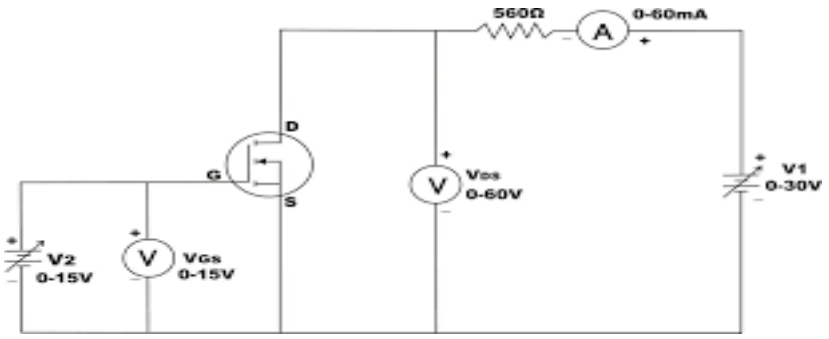
Apparatus Required:

S.No	Description	Range	Quantity
1	MOSFET	IRF 740	1
2	Resistor	560Ω	1
3	Voltmeter	0-15V,0-60V	1
4	RPS	0-15V, 0-30V	1
5	Ammeter	0-60mA	1

Theory:

A MOSFET (Metal oxide semiconductor field effect transistor) has three terminals called Drain, Source and Gate. MOSFET is a voltage controlled device. It has very high input impedance and works at high switching frequency. MOSFET's are of two types 1) Enhancement type 2) Depletion type.

Circuit Diagram:



Tabular Column:

A) Transfer Characteristics:

V_{DS1}= 1 V		V_{DS2}= 2V	
V_{GS} (V)	I_D(mA)	V_{GS} (V)	I_D(mA)

B) Drain Characteristics :

V_{GS} (V)	I_D(mA)

Calculation:

Trans conductance:

$$g_m = \left| \frac{\Delta I_D}{\Delta V_{GS}} \right| = \_\_\_\_\_ \text{mho at constant } V_{DS}$$

Output Resistance:

$$R_0 = \left| \frac{\Delta V_{DS}}{\Delta I_D} \right| = \_\_\_\_\_ \Omega \text{ at constant } V_{GS}$$

Procedure:

Transfer Characteristics:

1. Make the connections as per the circuit diagram.
2. Initially keep V1 and V2 at 0 V.
3. Switch ON the regulated power supplies. By varying V1, set VDS to some constant voltage say 5V.
4. Vary V2 in steps of 0.5V, and at each step note down the corresponding values of VGS and ID. (Note: note down the value of VGS at which ID starts increasing as the threshold voltage).
5. Reduce V1 and V2 to zero.
6. By varying V1, set VDS to some other value say 10V.
7. Repeat step 4.
8. Plot a graph of VGS versus ID for different values of VDS.

Drain or Output Characteristics:

1. Make the connections as per the circuit diagram.

2. Initially keep V1 and V2 at zero volts.

3. By varying V2, set VGS to some constant voltage (must be more than Threshold voltage).

4. By gradually increasing V1, note down the corresponding value of VDS and ID.
(Note: Till the MOSFET jumps to conducting state, the voltmeter which is connected across device as VDS reads approximately zero voltage. Further increase in voltage by V1 source cannot be read by VDS, so connect multi meter to measure the voltage and tabulate the readings in the tabular column).

5. Set VGS to some other value (more than threshold voltage) and repeat step 4.

6. Plot a graph of VDS versus ID for different values of VGS.

Result:

Thus the Transfer and Drain characteristics of MOSFET are plotted

EXPT. NO:
DATE :

FREQUENCY RESPONSE OF COMMON EMITTER AMPLIFIER

Aim:

To design and construct a Common Emitter **A**mplifier using voltage divider bias andto determine the Frequency and bandwidth

Apparatus Required :

S.no	Description	Range	Quantity
1	Transistor	BC 107	1
2	Resistor	61kΩ, 10kΩ, 1kΩ, 4.7kΩ	1,1,1,2
3	Capacitor	10µf, 100µf	2,1
4	Signal Generator	(0-3)MHz	1
5	CRO	30MHz	1
6	Regulated powersupply	(0-30)V	1
7	Bread Board	-	1
8	Connecting Wires	-	As per required

Theory:

A common emitter amplifier is type of BJT amplifier which increases the voltage level of the applied input signal V_{in} at output of collector. The CE amplifier typically has a relatively high input resistance (1 - 10 KΩ) and a fairly high output resistance. Therefore it is generally used to drive medium to high resistance loads. It is typically used in applications where a small voltage signal needs to be amplified to a large voltage signal like radio receivers. The input signal Vin is applied to base emitter junction of the transistor and amplifier output Vo is taken across collector terminal. Transistor is maintained at the active region by using the resistors R1,R2 and Rc. A very small change in base current produces a much larger change in collector current. The output Vo of the common emitter amplifier is 180 degrees out of phase with the applied the input signal V_{in}.

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the CE amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to CE amplifier using AC analysis.
4. Set the input voltage $V_{in}=V_{MSH}/2$ and vary the input signal frequency from 0Hz to 1MHz in incremental steps and note down the corresponding output voltage V_o for at least 20 different values for the considered range.
5. The voltage gain is calculated as $A_V = 20\log (V_0/V_i)$ dB
6. Find the Bandwidth and Gain-Bandwidth Product from Semi-log graph taking

   Frequency on x-axis and gain in dBon y-axis., Bandwidth,

   $BW = f_2-f_1$

Where, f_1 lower cut-off frequency and f_2 upper cut-off frequency

i) Set V_{in} = 0 by reducing the amplitude of the input signal from signal generator

ii) Open circuit the capacitors since it blocks DC voltage

iii) Set V_{CC}= +10v and measure the voltage drop across the Resistor V_{RC}, voltage across Collector- Emitter Junction V_{CE} and Voltage drop across base emitter junction. V_{BE}

iv) Find the Q-point of the transistor and draw the DC load line.

Maximum signal handling capacity :

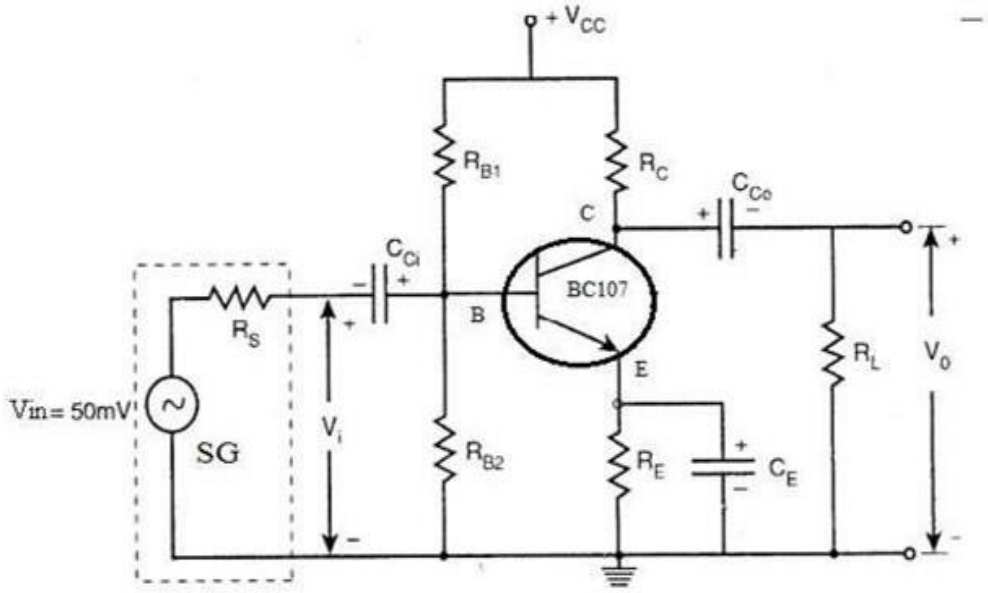
It is the process to find the maximum input voltage that can be handled by the amplifier, so that it amplifies the input signal without anydistortion.

Procedure:

Apply input signal Vin = 20 mV of 1Khz frequency to the amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across RL.

By increasing the amplitude of the input signal find maximum inputvoltage V_{MSH} across V_{BE} at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistor for efficient operating of transistor.

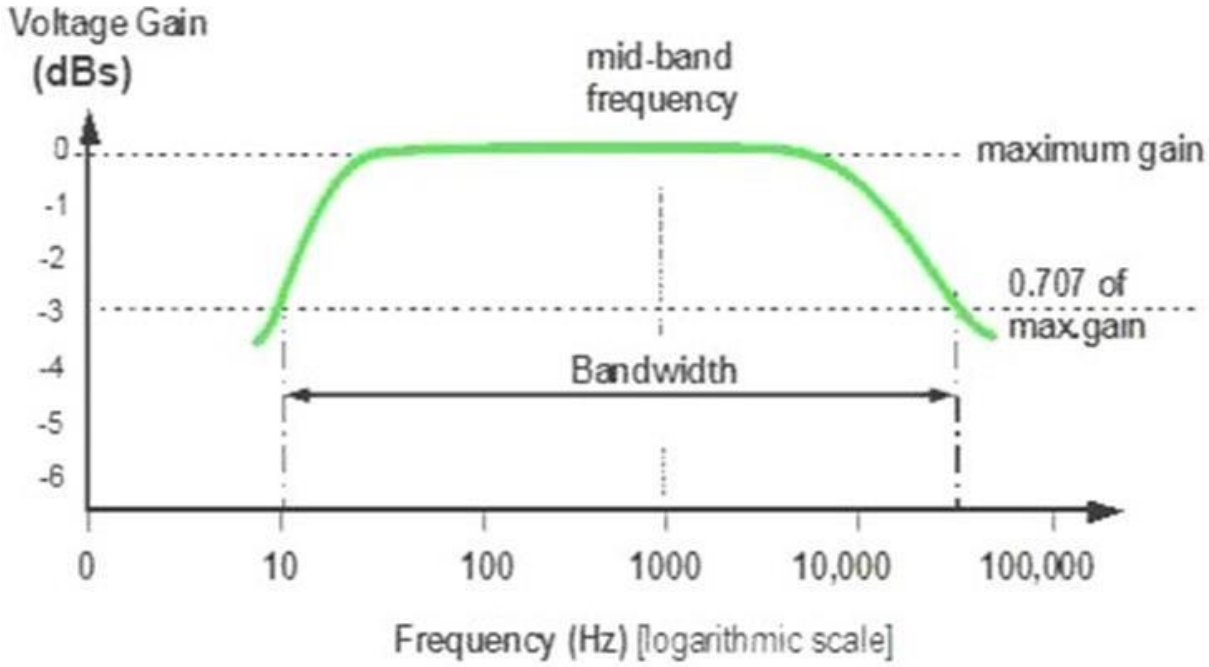
CE Amplifier without Feedback:



Tabulation [Without Feedback] :

S. NO	FREQUENCY [Hz]	OUTPUT VOLTAGE [VO] in Volts	GAIN = Vo/ Vi	GAIN= 20 log Vo/Vi dB

Model Graph:


Voltage Gain
(dBs)
mid-band
frequency
maximum gain
0.707 of
max.gain
Bandwidth
0
-1
-2
-3
-4
-5
-6
0
10
100
1000
10,000
100,000
Frequency (Hz) [logarithmic scale]

RESULT:

Thus the voltage and frequency response of a common emitter amplifier was constructed and measured.

EXPT. NO:
DATE :

FREQUENCY RESPONSE OF COMMON SOURCE AMPLIFIER

Aim:

To determine frequency response and bandwidth using common source amplifier

Apparatus Required:

S.No.	**Description**	**Range**	**Quantity**
1	Transistor	BFW10	1
2	Resistor	2kΩ,10kΩ,1kΩ	Each 1
3	Capacitor	0.1 µf,10 µf,1 µf	Each 1
4	signal Generator	(0-3) MHz	1
5	CRO	30MHz	1
6	Regulated power supply	(0-30) V	1
7	Bread Board	-	1
8	Connecting Wires	Single strand	As per required

Theory :

There are three basic types of FET amplifier or FET transistor namely common source amplifier, common gate amplifier and source follower amplifier. The common-source (CS) amplifier may be viewed as a transconductance amplifier or as a voltage amplifier.

i) As a transconductance amplifier, the input voltage is seen as modulating the current going to the load.

ii) As a voltage amplifier, input voltage modulates the amount of current flowing through the FET, changing the voltage across the output resistance according to Ω's law.

However, the FET device's output resistance typically is not high enough for a reasonable transconductance amplifier (ideally infinite), nor low enough for a decent voltage amplifier (ideally zero). Another major drawback is the amplifier's limited high-frequency response.

Therefore, in practice the output often is routed through either a voltage follower (common-drain or CD stage), or a current follower (common-gate or CG stage), to obtain more favorable output and frequency characteristics

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the CS amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to CE amplifier using AC analysis.
4. Set the input voltage Vin=V MSH /2 and vary the input signal frequency from 0Hz to 1MHz in incremental steps and note down the corresponding output voltage Vo for atleast 20 different values for the considered range.
5. The voltage gain is calculated as Av = 20log (V0/Vi)
6. Find the Bandwidth and Gain-Bandwidth Product from Semi-log graph taking frequency on x-axis and gain in dB on y-axis., Bandwidth, BW = f2-f1

   where f1 - lower cut-off frequency and f2 - upper cut-off frequency

a. Dc Analysis:

It is the procedure to find the operating region of transistor

Steps:

i) Set Vin = 0 by reducing the amplitude of the input signal from signal generator

ii) Open circuit the capacitors since it blocks DC voltage

iii) Set VCC= +10v and measure the voltage drop across the Resistor VRC, voltage across Collector-Emitter Junction VCE and Voltage drop across base emitter junction. VBE

iv) Find the Q-point of the transistor and draw the DC load line.

Maximum signal handling capacity:

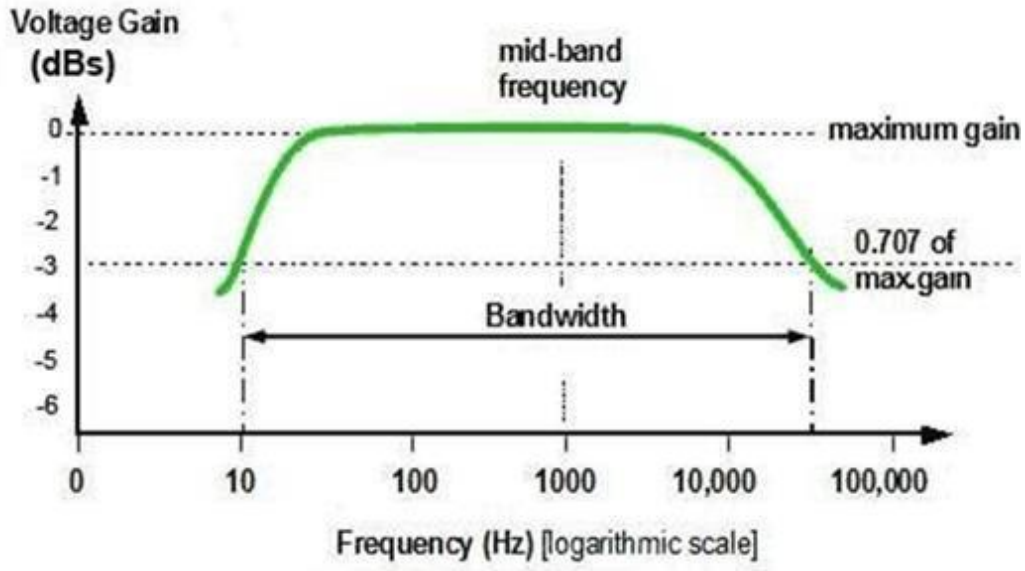
It is the process to find the maximum input voltage that can be handled by the amplifier, so that it amplifies the input signal without any distortion.

Procedure:

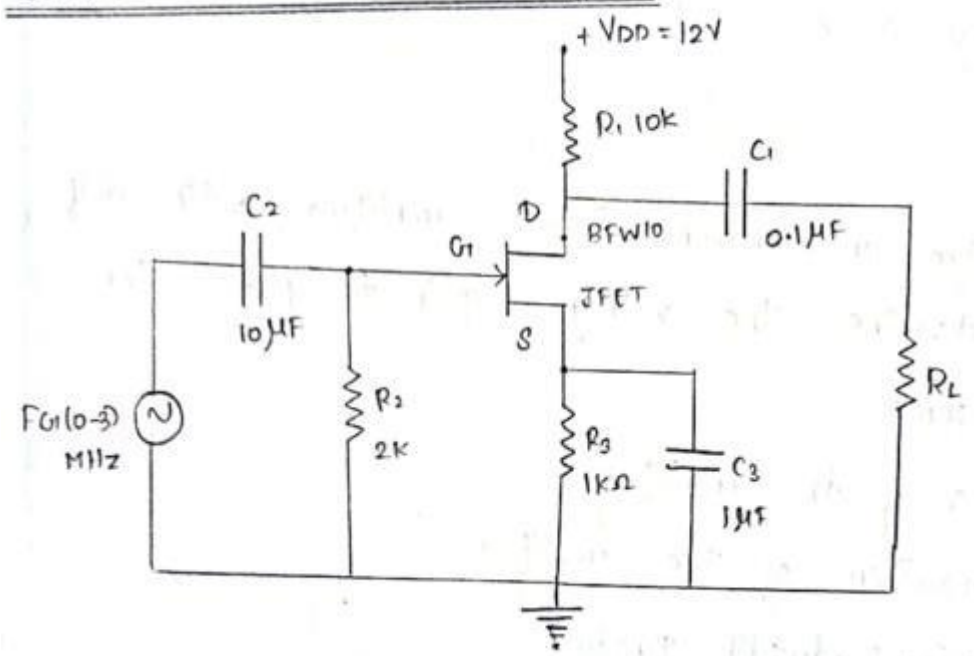
i. Apply input signal Vin = 1 V of 1Khz frequency to the CS amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across RL.

ii. By increasing the amplitude of the input signal find maximum input voltage V MSH across VBE at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistor for efficient operating of transistor.

Model graph :



Common Source Circuit Diagram :-



TABULATION

Input voltage (Vin) = 50mV

S. NO	**Frequency [Hz]**	**Output Voltage [VO] in Volts**	**Gain= 20 log Vo/Vi dB**

`RESULT:

The common Source amplifier was constructed and input resistance and gain were determined.

EXPT. NO:
DATE :

FREQUENCY RESPONSE OF COMMON BASE AMPLIFIER

Aim:

To determine the frequency response and bandwidth using Common Base amplifier

Apparatus Required:

S.No.	Description	Range	Quantity
1	Transistor	BC 107	1
2	Resistor	47KΩ,68KΩ,4.7KΩ, 2.2KΩ,10KΩ	Each 1
3	Capacitor	10μF	3
4	signal Generator	(0-3) MHz	1
5	CRO	30 MHz	1
6	Regulated power supply	(0-30) V	1
7	Bread Board	-	1
8	Connecting Wires	Single strand	as per required

Theory:

A common base amplifier is type of BJT amplifier which increases the voltage levelof the applied input signal Vin at output of collector. The Common base amplifier typically has good voltage gain and relatively high output impedance. But the Common base amplifier unlike CE amplifier has very low input impedance which makes it unsuitable for most voltage amplifier. It is typically used used as an active load for a cascode amplifier and also as a current follower circuit.

Circuit Operation:

A Positive-Going Signal Voltage At The Input Of A CB Pushes The Transistor Emitter InA Positive Direction While The Base Voltage Remains Fixed, Hence Vbe Reduces. The Reduction In VBE Results In Reduction In VRC, Consequently VCE Increases. The Rise In CollectorVoltage Effectively Rises The Output Voltage.

The Positive Going Pulse At The Input Produces A Positive-Going Output, Hence The There Is No Phase Shift From Input To Output In CB Circuit. InThe Same Way The Negative-Going Input Produces A Negative-Going Output.

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the CB amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to CEamplifier using AC analysis.
4. Set the input voltage $V_{in}=V_{MSH}/2$ and vary the input signal frequency from 0Hz to1MHz in incremental steps and note down the corresponding output voltage V_o for at least 20 different values for the considered range.
5. The voltage gain is calculated as $A_V = 20\log (V0/Vi)$
6. Find the Bandwidth and Gain-Bandwidth Product from Semi-log graph taking frequency on x-axis and gain in dB on y-axis., Bandwidth,

$$BW = f_2 - f_1$$

Where , f_1 lower cut-off frequencyand f_2 upper cut-off frequency

DC Analysis:

It is the procedure to find the operating region of transistor

Steps:

i) Set Vin = 0 by reducing the amplitude of the input signalfrom signal generator

ii) Open circuit the capacitors since it blocks DC voltage

iii) Set VCC= +10v and measure the voltage drop across the Resistor V_{RC}, voltage across Collector- Emitter Junction V_{CE} and Voltagedrop across base emitter junction. V_{BE}

iv) Find the Q-point of the transistor and draw the DC load line.

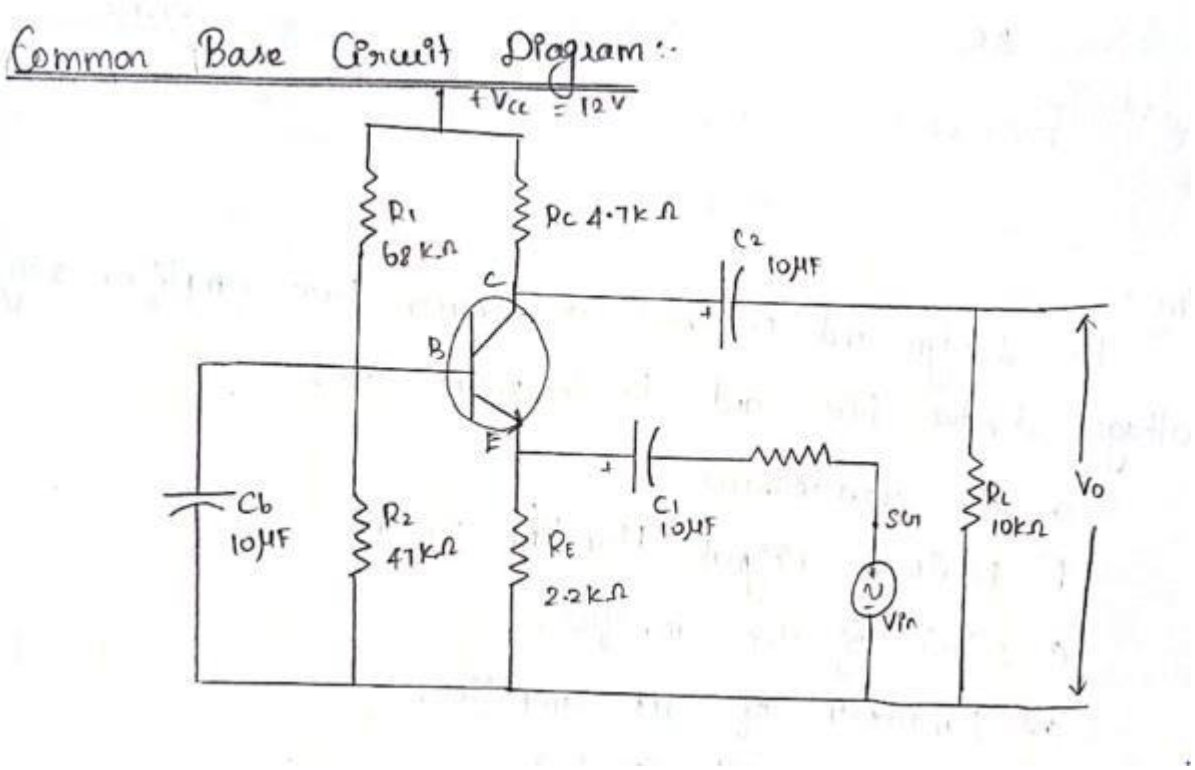
Maximum signal handling capacity:

It is the process to find the maximum input voltage that can be handled by theamplifier, so that it amplifies the input signal without any distortion

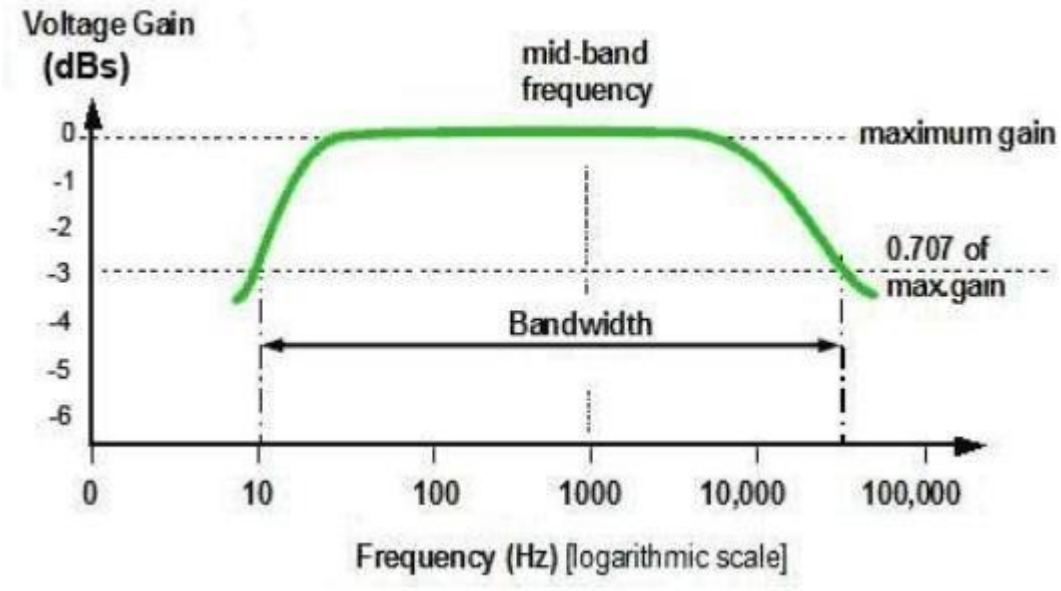
Procedure:

i. Apply input signal Vin = 20 mV of 1Khz frequency to the amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across RL.

ii. By increasing the amplitude of the input signal find maximum input voltage V MSH across VBE at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistor for efficient operating of transistor.



Model Graph:



Tabulation:

S. NO	FREQUENCY [Hz]	OUTPUT VOLTAGE [VO] in Volts	GAIN= 20 log V_o/V_i dB

RESULT:

The Common base amplifier was constructed and input resistance and gain were determined.

EXPT.NO:
DATE:

FREQUENCY RESPONSE OF COMMON COLLECTOR AMPLIFIER

Aim:

To Design and Construct a Common collector Amplifier and to determine its Frequency response and bandwidth

Apparatus Required:

S.No	Description	Range	Quantity
1	Transistor	BC 107	1
2	Resistor	10k Ω,1k Ω	2,1
3	Capacitor	10µF	2
4	Signal Generator	0-3MHz	1
5	CRO	0-30MHz	1
6	Regulated power supply	0-30 V	1
7	Bread Board	-	1
8	Connecting Wires	Single strand	as required

Theory:

A common collector amplifier is a unity gain BJT amplifier used for impedance matching and as a buffer amplifier.

Circuit Operation:

When a positive half-cycle of the input signal is applied to Base emitter junction of transistorthe forward bias voltage V_{be} is increased, which in turn increases the base current I_b of transistor. Since emitter current Ie is directly proportional to I_b the voltage drop across the Emitter Ve= IeRe is increased, hence, output voltage Vo is increased, thus, we get positive half-cycle of the output. It means that a positive-going input signal results in a positive going output signal and, consequently, the input and output signals are in phase with each other. Similarly the negative half cycle of input signal produces negative going output signal.

Characteristics of a CC Amplifier

1. high input impedance (20-500 K Ω)
2. low output impedance (50-1000 Ω)
3. high current gain of (1 + β) i.e. 50 – 500
4. voltage gain of less than 1 (unity)
5. power gain of 10 to 20 dB
6. no phase reversal of the input signal

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the CE amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to CE amplifier using AC analysis.
4. Set the input voltage $V_{in}=V_{MSH}/2$ and vary the input signal frequency from 0Hz to 1MHz in incremental steps and note down the corresponding output voltage V_o for at least 15 different values for the considered range.
5. The voltage gain is calculated as $A_v = 20\log (V_0/V_{in})$
7. Find the Bandwidth and Gain-Bandwidth Product from Semi-log graph taking frequency on x-axis and gain in dB

   on y-axis., Bandwidth,

   $BW = f_2-f_1$

   Where f_1 - lower cut-off frequency f_2 - upper cut-off frequency

a. DC Analysis:

It is the procedure to find the operating region of transistor

Steps:

i) Set Vin = 0 by reducing the amplitude of the input signal from signal generator

ii) Open circuit the capacitors since it blocks DC voltage

iii) Set VCC= +10v and measure the voltage drop across the Resistor V_{RC}, voltage across Collector- Emitter Junction V_{CE} and Voltage drop across base emitter junction. V_{BE}

iv) Find the Q-point of the transistor and draw the DC load line.

Q point analysis:

It is the procedure to choose the operating point of transistor

b. Maximum signal handling capacity :

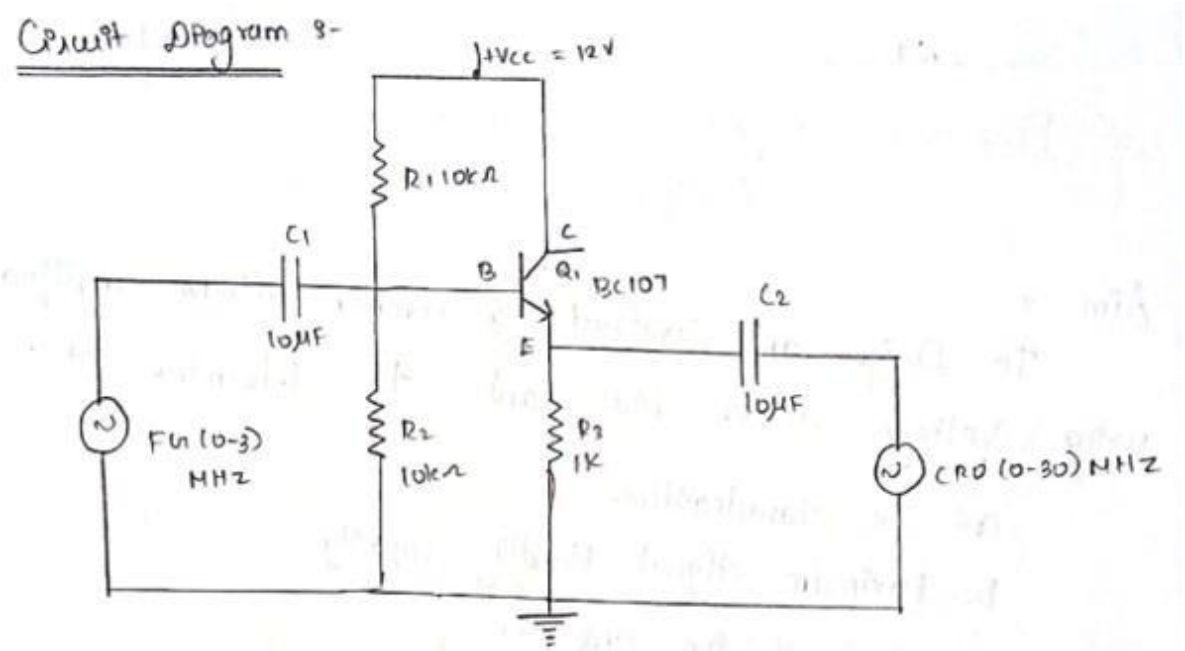
It is the process to find the maximum input voltage that can be handled by the amplifier, so that it amplifies the input signal without anydistortion.

Procedure:

i. Apply input signal Vin = 1 V of 1Khz frequency to the CC amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across R_L.

ii. By increasing the amplitude of the input signal find maximum input voltage V_{MSH} across V_{BE} at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistor for efficient operating of transistor.

Circuit Diagram:



TABULATION

S. NO	FREQUENCY [Hz]	OUTPUT VOLTAGE [V_O] in Volts	GAIN= 20 log V_o/V_i dB

RESULT:

The common collector amplifier was constructed and input resistance and gain were determined.

EXPT.NO:
DATE:

FREQUENCY RESPONSE OF CASCODE AMPLIFIER

Aim:

To Design and Construct a Cascode Amplifier and to determine its:

a. DC Characteristics
b. Maximum Signal Handling Capacity
c. Gain of the amplifier
d. Bandwidth of the amplifier
e. Gain -Bandwidth Product

Apparatus Required:

S.no	Description	Range	Quantity
1	Transistor	BC 107	1
2	Resistor	120 KΩ,270KΩ,25KΩ, 1KΩ,	1,1,1,3
3	Capacitor	10mf,0.01µf	1,2
4	Signal Generator	(0-3) MHz	1
5	CRO	30MHz	1
6	Regulated powersupply	(0-30) V	1
7	Bread Board	-	1
8	Connecting Wires	Single strand	As per required

Theory:

The cascode configuration has one of two configurations of multistage amplifier. In each case the collector of the leading transistor is connected to the emitter of the following transistor. The arrangement of the two transistors is shown in the circuit diagram. The cascode amplifier consists of CE stage connected in series with CB stage. The arrangement provides a relatively high input impedance with low voltage gain for the first stage to ensure the input miller capacitance is at a minimum, whereas the following CB stage provides an excellent high frequency response.

Features:

1. It provides high voltage gain and has high input impedance.
2. It provides high stability and has high output impedance

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the CE amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to CE amplifier using AC analysis.
4. Set the input voltage $V_{in}=V_{MSH}/2$ and vary the input signal frequency from 0Hz to 1MHz in incremental steps and note down the corresponding output voltage V_o for at least 20 different valuesfor the considered range.
5. The voltage gain is calculated as $Av = 20\log (V_0/V_i)$
6. Find the Bandwidth and Gain-Bandwidth Product from Semi-log graph taking frequency on x-axisand gain in dB on y-axis., Bandwidth, $BW = f_2-f_1$

   Where f_1 - lower cut-off frequencyf_2 - upper cut-off frequency

Dc Analysis:

It is the procedure to find the operating region of transistor

Steps:

i) Set Vin = 0 by reducing the amplitude of the input signal from signal generator
ii) Open circuit the capacitors since it blocks DC voltage
iii) Set VCC= +10v and measure the voltage drop across the Resistor V_{RC}, voltage acrossCollector- Emitter Junction V_{CE} and Voltage drop across base emitter junction. V_{BE}
iv) Find the Q-point of the transistor and draw the DC load line.

Q point analysis:

It is the procedure to choose the operating point of transistor

a. Maximum signal handling capacity:

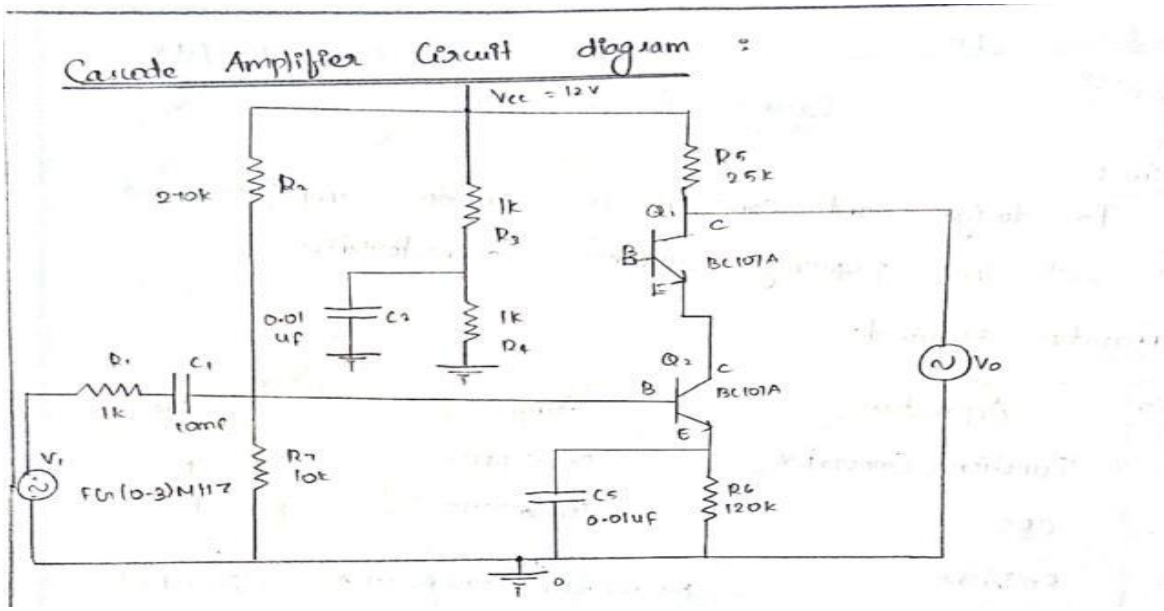
It is the process to find the maximum input voltage that can be handled by the amplifier, so thatit amplifies the input signal without any distortion.

Procedure:

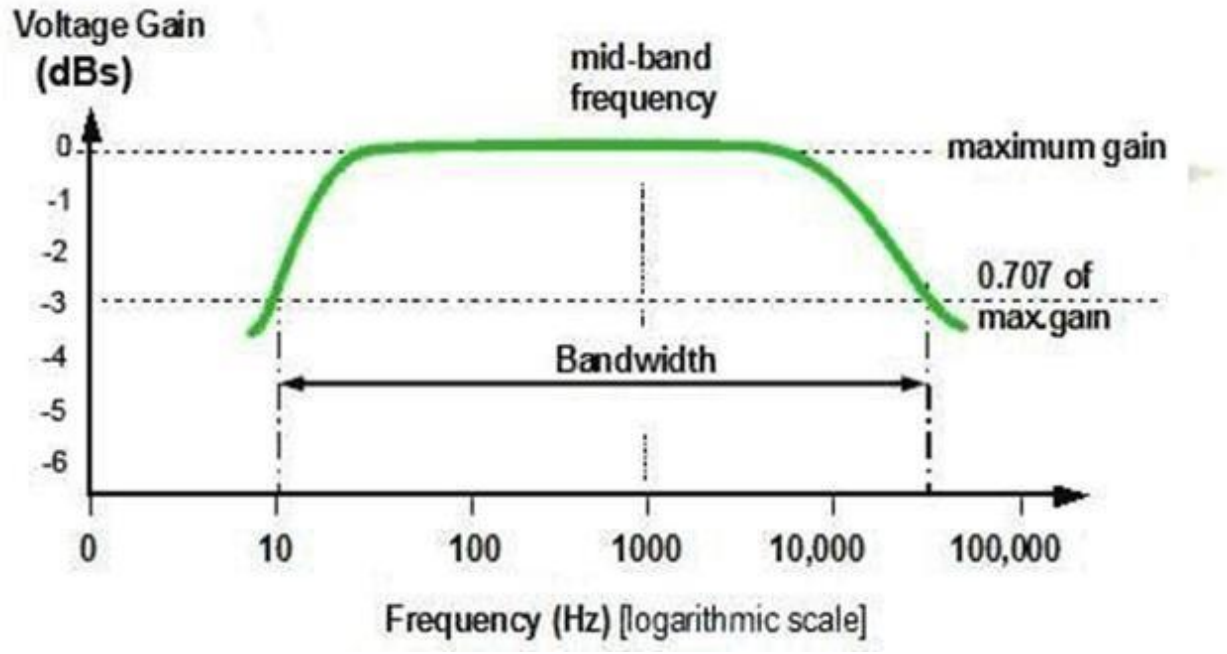
i) Apply input signal Vin = 20 mV of 1Khz frequency to the amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across RL.

ii) By increasing the amplitude of the input signal find maximum input voltage V_{MSH} across V_{BE} at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistorfor efficient operating of transistor.

Circuit Diagram:



Model Graph:



Tabulation

S. NO	FREQUENCY [Hz]	OUTPUT VOLTAGE [VO] in Volts	GAIN= 20 log Vo/Vi dB

RESULT:

The Cascode amplifier was constructed and input resistance and gain were determined.

EXPT.NO:
DATE:

CMRR measurement of Differential Amplifier

Aim:

To Design and Construct a Differential **A**mplifier using BJT and to determine its:

a. Transfer Characteristics
b. Gain of the amplifier in common mode
c. Gain of the amplifier in differential mode
d. CMRR (Common Mode Rejection Ratio)

Apparatus Required:

S.no	Description	Range	Quantity
1	Transistor	BC 107	2
2	Resistor	1kΩ,470Ω	2,1
3	Signal Generator	(0-3)MHz	1
4	CRO	30MHz	1
5	Regulated Power supply	(0-30)V	1
6	Bread Board	-	1
7	Connecting Wires	Single strand	as required

Theory:

A differential amplifier is a type of electronic amplifier that amplifies the differencebetween two voltages but does not amplify the particular voltages. The need for differential amplifier arises in many physical measurements where response from D.C to many MHZ isrequired. It is also used in input stage of integrated amplifier. The output signal in differential amplifier is proportional to the difference between the twoinput signals. $V_o = A_d (V_1 – V_2)$.

Where V1, V2 are the input voltages and Ad is the differential gain. If V1 = V2, then output voltage is zero. A non zero output voltage is obtained if V1 and V2are not equal.

i) The difference mode input voltage is defined as Vd = (V1-V2)

ii) The common mode input voltage is defined as the V_{CM}= (V1+V2)/2

iii) The CMRR is defined as the ratio of the differential gain Ad to common mode gain Ac andis generally expressed in dB. CMRR= 20 $\log_{10}$ (A_d / A_C)

Procedure:

1. Connect the circuit as per the circuit diagram
2. Determine the Q-point of the Differential amplifier using DC analysis.
3. Determine Maximum input voltage that can be applied to amplifier using AC analysis.
4. Determine the Transfer characteristics of Differential amplifier by plotting the graph for normalized differential input voltage [(V_{b1} – V_{b2}) / V_T] vs. Normalized collector current [Ic / Io].
5. Calculate the voltage gain of differential amplifier for differential modeas
   A_d = 20log (V_0/V_i) , Where Vi = V1 – V2
6. Calculate the voltage gain of differential amplifier for Common mode as
   A_C = 20log (V_0/V_i), Where Vi = (V1+ V2 / 2)
7. Find the Common mode rejection ratio of differential amplifier using the formula given below. CMRR= 20 $\log_{10}$ (Ad/Ac)

Where Ad- Differential mode gain in dB ,Ac – Common Mode gain in dB

DC Analysis:

It is the procedure to find the operating region of transistor

Steps

i) Set Vin = 0 by reducing the amplitude of the input signal from signal generator
ii) Open circuit the capacitors since it blocks DC voltage
iii) Set VCC= +10v and measure the voltage drop across the Resistor V_{RC}, voltage across Collector- Emitter Junction V_{CE} and Voltage drop across base emitter junction. V_{BE}
iv) Find the Q-point of the transistor and draw the DC load line.

Q point analysis:

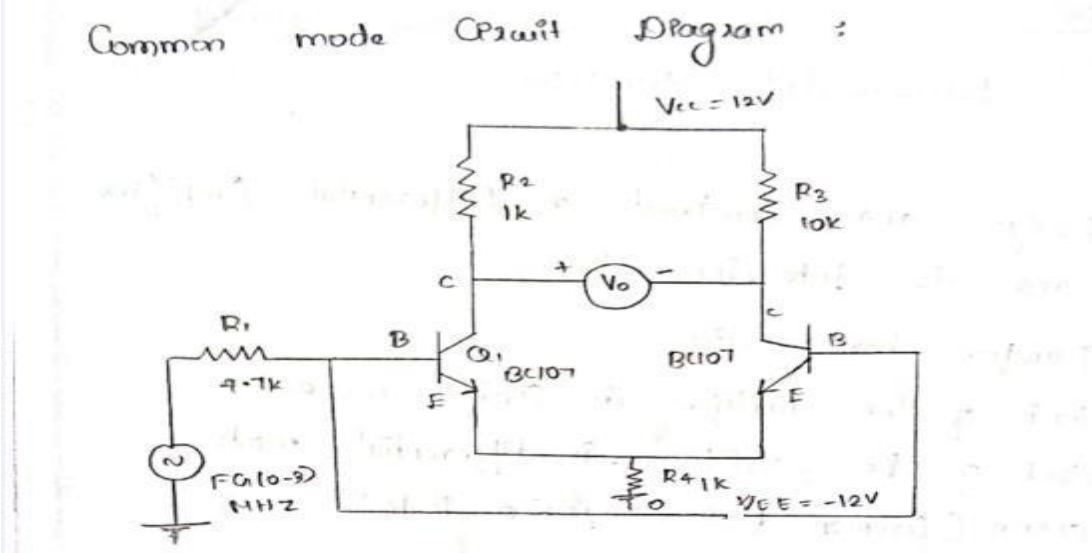
It is the procedure to choose the operating point of transistor

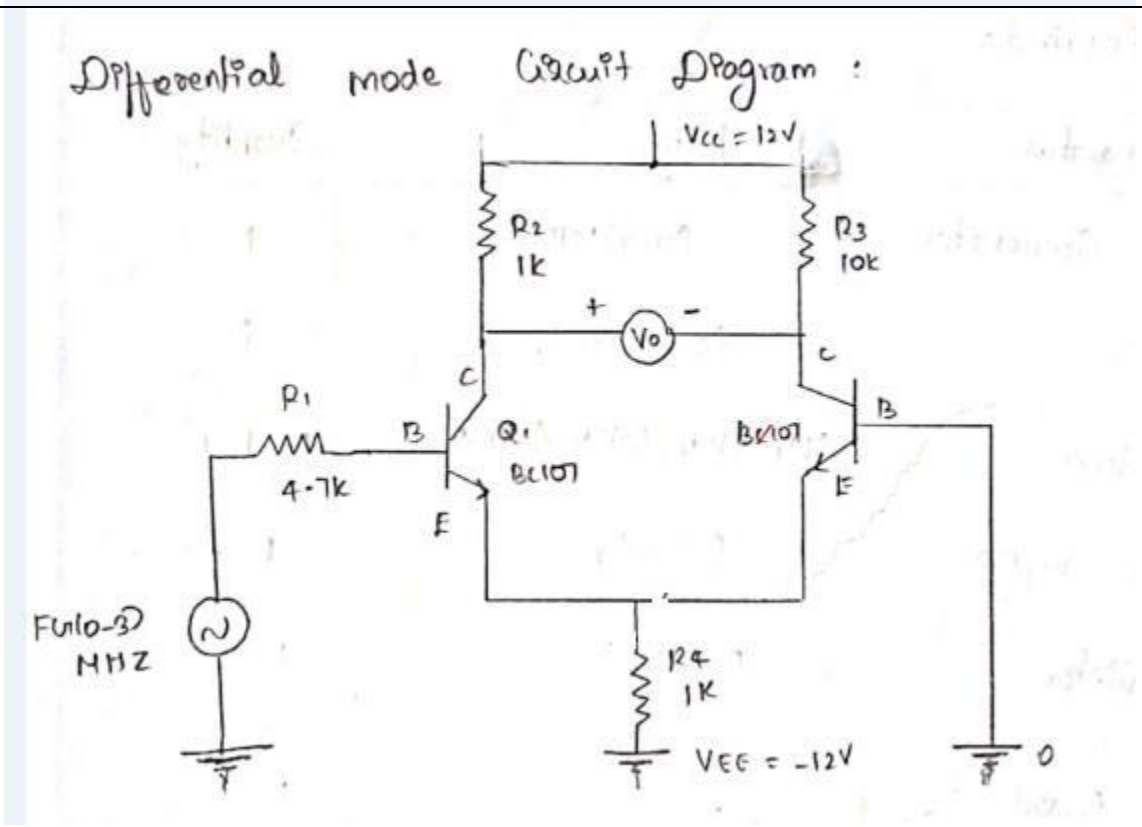
a. Maximum signal handling capacity:

It is the process to find the maximum input voltage that can be handled by the amplifier, so that itamplifies the input signal without any distortion.

Procedure:

Apply input signal Vin = 20 mV of 1Khz frequency to the amplifier using the signal generator between base emitter junction of the transistor. Find the sinusoidal output using CRO across RL. By increasing the amplitude of the input signal find maximum input voltage VMSH across VBE at which the sinusoidal signal gets distorted during the process which can be seen in the CRO. The amplitude obtained at this point is maximum voltage that can be applied to the transistor for efficient operating of transistor.





Tabulation

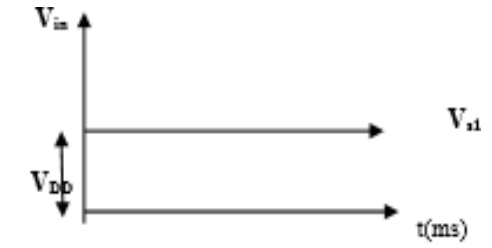
Differential mode voltage = 0.5 V
Common mode voltage = 5V

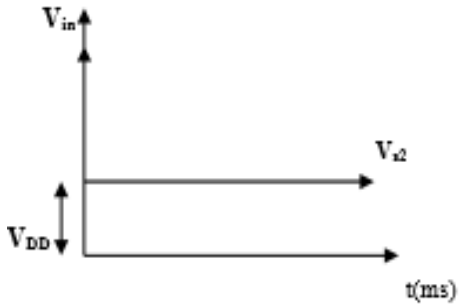
S.No	Frequency	Output Voltage		Gain dB		CMRR
		D.M	C.M	D.M	C.M	

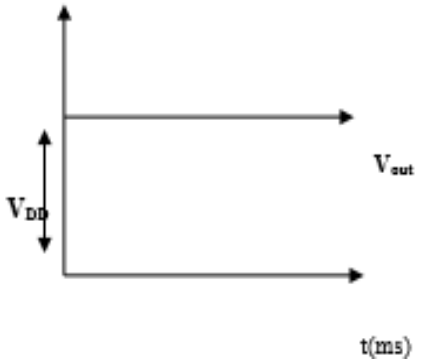
DC Transfer Characteristics:

S.No	Input Voltage	Output Current	
	Vin = (V1-V2) (V)	I_1 (mA)	I_2(mA)

Model Graph:







RESULT:

The Differential amplifier was constructed and input resistance and gain were determined.

EXPT.NO:
DATE:

CLASS A TRANSFORMER COUPLED POWER AMPLIFIER

Aim:

To construct a class A power amplifier and to determine the efficiency from its output waveform.

Apparatus Required:

S.No	Description	Range	Quantity
1	Transistor	SL100	1
2	Resistor	100 Ω , 10 Ω, 560 Ω	2,1,1
3	Capacitor	2.2 µf, 5 µf	1,1
4	Transformer withCenter Tapped Secondary	(9 - 0 - 9) V	1
5	CRO	30MHz	1
6	Function Generator	-	1
7	Regulated PowerSupply	0-30V	1
8	Digital Multimeter/Digital Voltmeter	(0-20V)	1
9	Bread Board	-	1
10	Connecting Wires	Single strand	As per required

Theory:

Transistor power amplifiers handle large signals. If the current flows at all times duringthe full cycle of the signal, the power amplifier is known as class A amplifier.

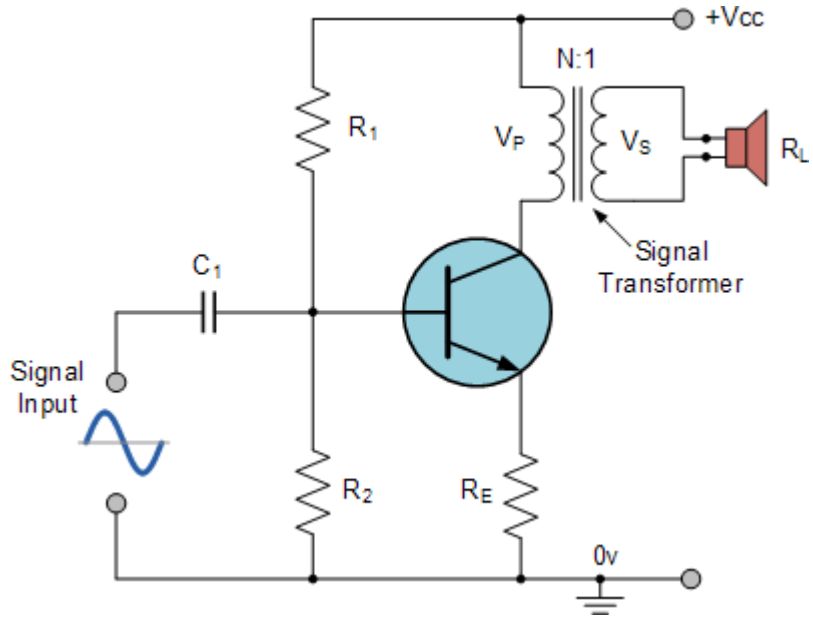
Obviously, for this to happen the power amplifier must be biased in such a way that no part of the signal is cut off. The efficiency of class A amplifier is only 50%. But it provides less power dissipation. Also there is no distortion in class A power amplifier.

Procedure:

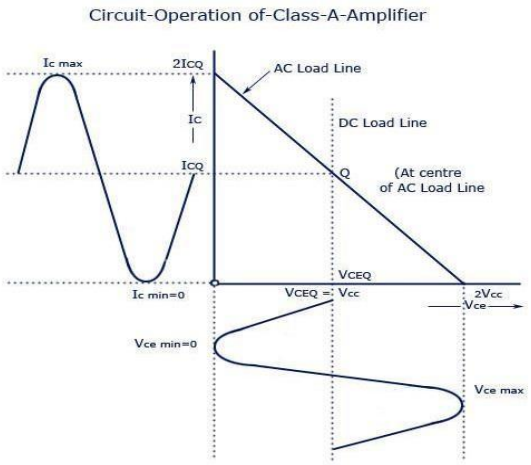
1. Connect the circuit as per the circuit diagram.
2. Set the input and apply the input signal from the FG and observe the output.
3. Note down the collector current using ammeter.
4. Draw the input and output waveform.

5. By using specified formula, AC output power, DC input power and efficiency are calculated.

Circuit Diagram Class A Power Amplifier



Model Graph



Tabulation:

	Voltage(V)	Time(ms)	Current(mA)
Input waveform	2	2ms	2
Output waveform	2	2ms	3

RESULT:

Thus the class A amplifier was designed and constructed also its efficiency was calculated.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3401 NETWORKS AND SECURITY

Semester - 04

LABORATORY MANUAL

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

Experiments using C

1. Implement the Data Link Layer framing methods,

   (i)Bit stuffing, (ii) Character stuffing
2. Implementation of Error Detection / Correction Techniques

   (i)LRC, (ii) CRC, (iii) Hamming code
3. Implementation of Stop and Wait, and Sliding Window Protocols
4. Implementation of Go back-N and Selective Repeat Protocols.
5. Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford).
6. Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's).
7. Data encryption and decryption using Data Encryption Standard algorithm.
8. Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm.
9. Implement Client Server model using FTP protocol.

Experiments using Tool Command Language

1. Implement and realize the Network Topology - Star, Bus and Ring using NS2.
2. Implement and perform the operation of CSMA/CD and CSMA/CA using NS2.

Program to implement Bit Stuffing

Aim:

To write a C program to implement Bit Stuffing.

```
#include<stdio.h>
#include<string.h>
int main()
{
   int a[20],b[30],i,j,k,count,n;
   printf("Enter frame size (Example: 8):");
   scanf("%d",&n);
   printf("Enter the frame in the form of 0 and 1 :");
   for(i=0; i<n; i++)
      scanf("%d",&a[i]);
   i=0;
   count=1;
   j=0;
   while(i<n)
   {
      if(a[i]==1)
      {
         b[j]=a[i];
         for(k=i+1; a[k]==1 && k<n && count<5; k++)
         {
            j++;
            b[j]=a[k];
            count++;
            if(count==5)
            {
               j++;
               b[j]=0;
            }
            i=k;
         }
      }
      else
      {
         b[j]=a[i];
      }
      i++;
      j++;
   }
   printf("After Bit Stuffing :");
   for(i=0; i<j; i++)
      printf("%d",b[i]);
   return 0;
```

}

OUTPUT for BIT STUFFING:

```
Enter frame size (Example: 8):12
Enter the frame in the form of 0 and 1 :0 1 0 1 1 1 1 1 1 0 0 1
After Bit Stuffing :0101111101001
```

Program to implement Character Stuffing

Aim:

To write a C program to implement Character Stuffing.

Program to implement Character Stuffing

```
#include<stdio.h>
#include<string.h>
main()
{
      char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3];
      int i, j, p = 0, q = 0;
      clrscr();
      printf("Enter characters to be stuffed:");
      scanf("%s", a);
      printf("\nEnter a character that represents starting delimiter:");
      scanf(" %c", &sd);
      printf("\nEnter a character that represents ending delimiter:");
      scanf(" %c", &ed);
      x[0] = s[0] = s[1] = sd;
      x[1] = s[2] = '\0';
      y[0] = d[0] = d[1] = ed;
      d[2] = y[1] = '\0';
      strcat(fs, x);
      for(i = 0; i < strlen(a); i++)
      {
         t[0] = a[i];
         t[1] = '\0';
         if(t[0] == sd)
            strcat(fs, s);
         else if(t[0] == ed)
            strcat(fs, d);
         else
            strcat(fs, t);
      }
      strcat(fs, y);
      printf("\n After stuffing:%s", fs);
      getch();
   }
```

Output:-

```
Enter characters to be stuffed: goodday
Enter a character that represents starting delimiter: d
Enter a character that represents ending delimiter: g
After stuffing:  dggooddddayg.
```

IMPLEMENTATION OF ERROR DETECTION AND CORRECTION TECHNIQUE

Aim:

To write a C program to implement Error Detection and Correction Techniques.

Algorithm:

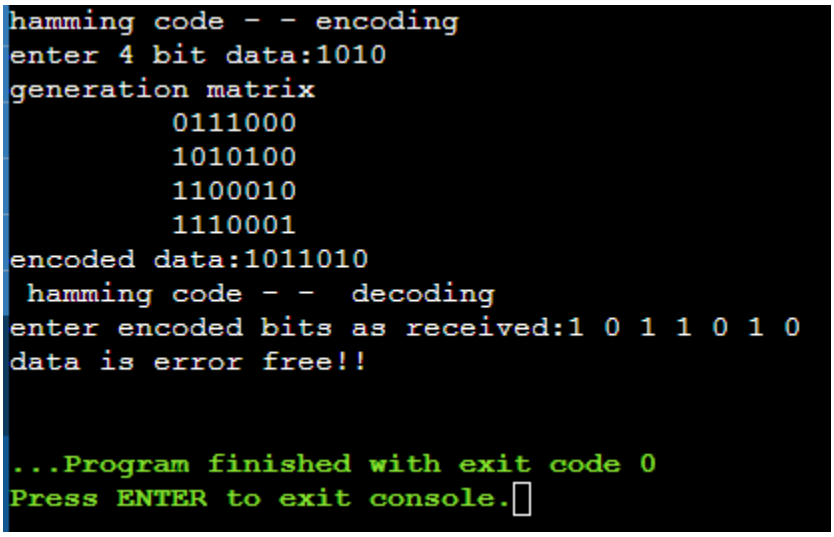
1. Get 4 bit data.
2. Generate the Generates Matrix.
3. Encode the data.
4. Create Parity check Matrix.
5. Enter the error data.
6. Calculate syndrome using eH^T.
7. Decode the error data.

Program:

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
char data[5];
int encoded[8],edata[7],syndrome[3];
int hmatrix[3][7]={1,0,0,0,1,1,1,0,1,0,1,0,1,1,0,0,1,1,1,0,1};
char gmatrix[4][8]={"0111000","1010100","1100010","1110001"};
int main()
{
int i,j;
printf("hamming code - - encoding \n");
printf("enter 4 bit data:");
scanf("%s",data);
printf("generation matrix \n");
for(i=0;i<4;i++)
printf("\t %s \n",gmatrix[i]);
printf("encoded data:");
for(i=0;i<7;i++)
{
for(j=0;j<4;j++)
encoded[i]+=((data[j]-'0')*(gmatrix[j][i]-'0'));
encoded[i]=encoded[i]%2;
printf("%d",encoded[i]);
}
printf("\n hamming code - - decoding \n");
printf("enter encoded bits as received:");
for(i=0;i<7;i++)
scanf("%d",&edata[i]);
for(i=0;i<3;i++)
{
```

```
for(j=0;j<7;j++)
syndrome[i]=syndrome[i]+(edata[j]*hmatrix[i][j]);
syndrome[i]=syndrome[i]%2;
}
for(j=0;j<7;j++)
if((syndrome[0]==hmatrix[0][j])&&(syndrome[1]==hmatrix[1][j])&&
(syndrome[2]==hmatrix[2][j]))
break;
if(j==7)
printf("data is error free!!\n");
else
{
printf("error received a bit number %d of data \n",j+1);
edata[j]=!edata[j];
printf("the correct data should be :");
for(i=0;i<7;i++)
printf("%d",edata[i]);
}
}
```

OUTPUT:



Result:

Thus Error Detection and Correction Technique has been implemented.

Hamming code Implementation in C/C++
Pre-requisite: Hamming Code
Given a message bit in the form of an array **msgBit[]**, the task is to find the Hamming Code of the given message bit.
Examples:
Input: *S = "0101"*
Output:
Generated codeword:
r1 r2 m1 r4 m2 m3 m4
0 1 0 0 1 0 1
Explanation:
Initially r1, r2, r4 is set to '0'.
r1 = Bitwise XOR of all bits position that has '1' in its 0th-bit position.

r2 = Bitwise XOR of all bits that has '1' in its 1st-bit position.
r3 = Bitwise XOR of all bits that has '1' in its 2nd-bit position.
Input: *S = "0111"*
Output:
Generated codeword:
r1 r2 m1 r4 m2 m3 m4
0 0 0 1 1 1 1

Approach: The idea is to first find the number of redundant bits which can be found by initializing **r** with **1** and then incrementing it by **1** each time while 2^r is smaller than **(m + r + 1)** where **m** is the number of bits in the input message. Follow the below steps to solve the problem:

- Initialize **r** by **1** and increment it by **1** until 2^r is smaller than **m+r+1**.
- Initialize a vector **hammingCode** of size **r + m** which will be the length of the output message.
- Initialize all the positions of redundant bits with **-1** by traversing from **i = 0 to r – 1** and setting **hammingCode [2^i – 1] = -1**. Then place the input message bits in all the positions where **hammingCode[j]** is not **-1** in order where **0 <= j < (r + m)**.
- Initialize a variable **one_count** with **0** to store the number of ones and then traverse from **i = 0** to **(r + m – 1)**.
- If the current bit i.e., **hammingCode[i]** is not **-1** then find the message bit containing set bit at $\log_2(i+1)^{th}$ position by traversing from **j = i+2 to r+m** by incrementing **one_count** by **1** if **(j & (1<<x))** is not **0** and **hammingCode[j – 1]** is **1**.
- If for index **i**, **one_count** is even, set **hammingCode[i] = 0** otherwise set **hammingCode[i] = 1**.
- After traversing, print the **hammingCode[]** vector as the output message.

Below is the implementation of the above approach:

// C program for the above approach

#include <math.h>

```
#include <stdio.h>

// Store input bits
int input[32];

// Store hamming code
int code[32];

int ham_calc(int, int);
void solve(int input[], int);

// Function to calculate bit for
// ith position
int ham_calc(int position, int c_l)
{
    int count = 0, i, j;
    i = position - 1;

    // Traverse to store Hamming Code
    while (i < c_l) {

        for (j = i; j < i + position; j++) {

            // If current boit is 1
            if (code[j] == 1)
                count++;
        }

        // Update i
        i = i + 2 * position;
    }

    if (count % 2 == 0)
        return 0;
    else
        return 1;
}

// Function to calculate hamming code
void solve(int input[], int n)
{
    int i, p_n = 0, c_l, j, k;
    i = 0;

    // Find msg bits having set bit
    // at x'th position of number
    while (n > (int)pow(2, i) - (i + 1)) {
        p_n++;
```

```
        i++;
    }

    c_l = p_n + n;

    j = k = 0;

    // Traverse the msgBits
    for (i = 0; i < c_l; i++) {

        // Update the code
        if (i == ((int)pow(2, k) - 1)) {
            code[i] = 0;
            k++;
        }

        // Update the code[i] to the
        // input character at index j
        else {
            code[i] = input[j];
            j++;
        }
    }

    // Traverse and update the
    // hamming code
    for (i = 0; i < p_n; i++) {

        // Find current position
        int position = (int)pow(2, i);

        // Find value at current position
        int value = ham_calc(position, c_l);

        // Update the code
        code[position - 1] = value;
    }

    // Print the Hamming Code
    printf("\nThe generated Code Word is: ");
    for (i = 0; i < c_l; i++) {
        printf("%d", code[i]);
    }
}

// Driver Code
void main()
{
```

```
    // Given input message Bit
    input[0] = 0;
    input[1] = 1;
    input[2] = 1;
    input[3] = 1;

    int N = 4;

    // Function Call
    solve(input, N);
}
```

Output:

The generated Code Word is: 0001111

IMPLEMENTATION OF STOP AND WAIT PROTOCOL USING C PROGRAM

AIM:

To write a 'C' program to simulate a Stop and Wait protocol.

ALGORITHM:

Sender

1. Create a server socket using socket system call
2. If the receiver is ready, initialize sender's frame sequence.
3. Get data from user.
4. Send it to the receiver along with sequence number
5. Wait for acknowledgements.
6. If an acknowledgement is received, send the next frame of data.
7. Stop

Receiver

1. Indicate to sender .the readiness to accept frames.
2. Initialize receiver's expected frame sequence.
3. Accept the incoming frame
4. Send an acknowledgement.
5. Repeat step 3 and 4 until all frames are received in sequence.
6. Discard frame, thereby force the sender to retransmit
7. Stop.

PROGRAM:

SERVER:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<stdlib.h>
#include<netinet/in.h>
```

```
#include<unistd.h>
#include<stdio.h>
#include<string.h>
void main()
{
struct sockaddr_in cli,serv;
```

```
int sockfd,connfd,len,i,n,flag,x;
char ms[10][50],ack[40];
len=sizeof(struct sockaddr);
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv.sin_family=AF_INET;
serv.sin_port=9960;
serv.sin_addr.s_addr=INADDR_ANY;
bind(sockfd,(struct sockaddr*)&serv,sizeof(struct sockaddr));
listen(sockfd,5);
connfd=accept(sockfd,(struct sockaddr *)&cli,&len);
printf("Enter the limit\n");scanf("%d",&n);
printf("Enter the data\n");for(i=1;i<=n;i++)
{printf("%d",i);
scanf("%s",ms[i]);}//printf("Enter the num");//scanf("%d",x);
printf("End of Data");
for(i=1;i<=n;i++)
	printf("%s",ms[i]);
flag=1;
for(i=1;i<=n;i++)
{
if(flag==1)
{
send(connfd,ms[i],sizeof(ms[i]),0);
flag=0;
}
recv(connfd,ack,sizeof(ack),0);
printf("%s",ack);
if((strcmp(ack,"yes"))==0)
	flag=1;
else
{printf("Acknowlegment is not given\n");
	flag=1;
	continue;
}}
close(sockfd);
}
```

CLIENT:

```
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
  int a[3];
struct sockaddr_in cli,serv;
int sockfd,c,i,n;
char msg[50][50],ac[10];
sockfd=socket(AF_INET,SOCK_STREAM,0);
cli.sin_family=AF_INET; cli.sin_port=9960; cli.sin_addr.s_addr=INADDR_ANY;
connect(sockfd,(struct sockaddr*)&cli,sizeof(struct sockaddr));
printf("Enter the limit\n");
scanf("%d",&n);
printf("The received data\n");
for(i=1;i<=n;i++) {recv(sockfd,msg[i],sizeof(msg[i]),0);
printf("%s",msg[i]);
printf("Enter the ack\n");
scanf("%s",ac);
send(sockfd,ac,sizeof(ac),0);
}
close(sockfd);
```
} **OUTPUT:**

SERVER:

```
sam@boss[share]$vi sandws.c
sam@boss[share]$cc sandws.c
sam@boss[share]$cc sandws.c
sam@boss[share]$./a.out
Enter the limit  4
Enter the data
1a
2b
3c
4d
End of Data
```

abcd
yes
yes
yes
yes

CLIENT:

Enter the limit4
The received data a
Enter the ack yes b
Enter the ack yes c
Enter the ack yes d
Enter the ack yes

RESULT:

Thus the Stop & Wait Protocol program was executed successfully**.**

IMPLEMENTATION OF SLIDING WINDOW PROTOCOL USING C PROGRAMAIM:

To write a program to simulate a sliding window protocol that uses Selective repeat and Go back-ARQ.

ALGORITHM:

SERVER:

1) Create a server socket using socket function

socket(int family,int type,int protocol);

2) Assume the sending window size as 6 (m=3)

3) If the receiver is ready, initialize sender's frame sequence to 0

4) Get data from user.

5) Send it to the receiver along with sequence number

6) Increment sequence number by 1.

7) Repeat step 4-6 until all frames have been sent.

8) Wait for acknowledgement

9) If all acknowledgements have arrived then go to step 11.

10) Set sequence number to earliest outstanding frame for which there is no ACK . Go to step 4.

11) Stop

CLIENT:

1) Create a socket for client using socket function

int socket(int family,int type,int protocol);

2) Indicate to sender, the readiness to accept frames.

3) Initialize receiver's expected frame sequence to 0.

4) Accept the incoming frame

5) If frame's sequence == receiver's sequence then go to step 6

6) Send an acknowledgement

7) Repeat step 3-6 until all frames are received in sequence and go to step 8

8) Discard frame, thereby force the sender to retransmit. Go to step 3.

9) Stop

PROGRAM:

SERVER:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
main()
{
int len,sockfd,connfd,i,n,c;
char rdata[10][100],rmsg[10][100],win[10],los[20],temp[10][100],ch[10],recmsg[10];
struct sockaddr_in serv,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv.sin_family=AF_INET;
serv.sin_port=9960;
serv.sin_addr.s_addr=INADDR_ANY;
bind(sockfd,(struct sockaddr*)&serv,sizeof(struct sockaddr));
listen(sockfd,5);
len=sizeof(struct sockaddr);
connfd=accept(sockfd,(struct sockaddr*)&cli,&len);
read(connfd,win,sizeof(win));
sscanf(win,"%d",&n);
printf("%d",n);
for(i=1;i<=n;i++)
{
recv(connfd,rdata[i],sizeof(rdata[i]),0);
printf("the %d data send by the client is %s \n",i,rdata[i]);
```

```
}
printf("enter the lost data");
scanf("%s",&los);
write(connfd,los,sizeof(ch));
sscanf(ch,"%d",&c);
if(c==1)
{
recv(connfd,recmsg,sizeof(recmsg),0);
printf("the data received using selective repeat is %s",recmsg);
}
else
{
for(i=1;i<=n;i++)
{
recv(connfd,recmsg[i],sizeof(recmsg[i]),0);
strcpy(temp[i],rmsg[i]);
printf("the data recv is %s \n",temp[i]);
}
}
close(connfd);
close(sockfd);
}
```

CLIENT:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
main()
{
int sockfd,amount,i,w,a,ch;
char data[10][100],lost[20],st[10],si[10],s,g,choice[10];
struct sockaddr_in serv,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
cli.sin_family=AF_INET;
cli.sin_port=9960;
cli.sin_addr.s_addr=INADDR_ANY;
connect(sockfd,(struct sockaddr *)&cli,sizeof(struct sockaddr));
printf("enter the total amount of data to be send \n");
```

```
scanf("%d",&amount);
for(i=1;i<=amount;i++)
{
printf("enter the %d data to be send \n",i);
scanf("%s",data[i]);
}
printf("enter the window size");
scanf("%s",si);
send(sockfd,si,sizeof(si),0);
sscanf(si,"%d",&w);
for(i=0;i<w;i++)
{
send(sockfd,data[i],sizeof(data[i]),0);
}
read(sockfd,lost,sizeof(lost));
printf("lost frame is %s \n",lost);
sscanf(lost,"%d",&a);
printf("enter the choice \1.selective repeat \n 2.go back ARQ \n");
scanf("%s",choice);
send(sockfd,choice,sizeof(choice),0);
sscanf(choice,"%d",&ch);
switch(ch)
{
case 1:
send(sockfd,data[a],sizeof(data[a]),0);
break;
case 2:
for(i=a;i<=(w+(a-1));i++)
{
send(sockfd,data[i],sizeof(data[i]),0);
}
printf("the lost data is sent again \n");
break;
}
close(sockfd);
}
```

OUTPUT:

‘

SERVER

sam@boss[sha]$./a.out
the 1 data sent by client is shali
the 2 data sent by the client is akshaya
the 3 data sent by the client is raji
enter the lost data 2
the data recv is shali
the data recv is akshaya
the data recv is raji

RESULT:

Thus Sliding Window Protocol is simulated using Selective repeat and Go back-ARQ.

{

IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM

AIM:

To simulate a link failure and to observe distance vector routing protocol in action.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
int dmat[20][20];
int n,i,j,k,count=0,dist;
clrscr();
printf("\n enter the number of nodes:");
scanf("%d",&n);
printf("enter the cost matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
scanf("%d",&dmat[i][j]);
dmat[i][i]=0;
rt[i].dist[j]=dmat[i][j];
rt[i].from[j]=j;
}
do
{
count=0;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
for(k=0;k<n;k++)
if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
{
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
}
}
while(count!=0);
```

```
{
for(i=0;i<n;i++)
{
printf("\n state value for router %d is \n",i+1);
for(j=0;j<n;j++)
{
printf("\n node %d via %d distance %d",i+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n");
getch();
return 0;
}
```

RESULT:

Thus the Distance Vector Routing program was executed successfully.

‘

AIM:

To simulate a link failure and to observe Link state routing protocol in action.

PROGRAM:

```
#include<stdio.h>
//#include<conio.h>
//#include<process.h>
 #include<stdlib.h>
#include<string.h>
#include<math.h>
        #define IN 99
        #define N 6
        int dijkstra(int
        cost[][N],int source,int
        target);char
        interface[6][6][20]={{"0","0","0","0","0","0"},{"0","0","192.1.1.1","0","198.1.1.1"
        ,"0"},{"0","192
        .
        1.1.2","0","0","0","200.1.1.1"},{"0","0","0","0","198.1.1.2","0"},{"0","192.
        1.1.3","0","198.1.1.3", "
        0","200.1.1.2"},{"0","0","200.1.1.3","0","200.1.1.4","0"}};
        char *strrev(char *str)

char *p1,*p2; if(! str || ! *str) return str;
for(p1=str,p2=str+strlen(str)-1;p2>p1;++p1,--p2)
{
*p1^=*p2;
*p2^=*p1;
*p1^=*p2;
}
return str;

}
int main()
{
int cost[N][N],i,j,w,ch,co; char ip[10];
int source,target,x,y;
printf("\tThe shortest path algorithm (DIJKSTRA'S ALGORITHM in C \n\n");
for(i=1;i<N;i++)
for(j=1;j<N;j++) cost[i][j]=IN; for(x=1;x<N;x++)
{
for(y=x+1;y<N;y++)
{
printf("\nEnter the weight of the path between nodes %d and %d: ",x,y);
scanf("%d",&w); cost[x][y]=cost[y][x]=w;
}
printf("\n");
}
```

```
{
for(x=1;x<N;x++)
{
for(y=1;y<N;y++)
{ printf("%s: \t",interface[x][y]);
}
printf("\n");
}
printf("\nEnter the source: "); scanf("%d",&source); printf("\nEnter the target ");
scanf("%d",&target); co=dijsktra(cost,source,target);
printf("\nThe shortest path : %d\n",co); return 0;
}
int dijsktra(int cost[][N],int source,int target)
{
int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j,x,y; char path[N];
int path1[N]; for(i=1;i<N;i++)
{
dist[i]=IN; prev[i]=-1;
}
start=source; selected[start]=1; dist[start]=0; while(selected[target]==0)
{
min=IN; m=0;
for(i=1;i<N;i++)
{
d=dist[start]+cost[start][i]; if(d<dist[i]&&selected[i]==0)
{
dist[i]=d; prev[i]=start;
}
if(min>=dist[i]&&selected[i]==0)
{
min=dist[i]; m=i;

}
}
start=m; selected[start]=1;
}
start=target; j=0;
while(start!=-1)
{
path1[j++]=start; start=prev[start];

}
path[j]='\0'; strrev(path); printf("%s",path);
printf("\n");
for(j=j-1;j>=0;j--)
{
printf("%d\t",path1[j]); if(j>0)
{
x=path1[j]; y=path1[j-1];
printf("%s\t%s\n",interface[x][y],interface[y][x]);
}
```

```
{
}
return dist[target];
}
```

RESULT:

Thus the Link State Routing program was executed successfully**.**

STUDY OF DATA ENCRYPTION AND DECRYPTION USING C PROGRAM

Experiment No: 10b Date:

Aim:

To write a C program to implement Encryption and Decryption

ALGORITHM:

1. Initialize the variables.
2. Enter the data to be encrypted.
3. Using Encrypt function, encrypt data and print it.
4. Using Decrypt function, decrypt data and print it.

{

Program: #include<stdio.h> #include<conio.h> #include<string.h>

```
void encrypt(char password[],int key)
{
unsigned int i; for(i=0;i<strlen(password);++i)
{
password[i]=password[i]-key;
}
}
void decrypt(char password[],int key)
{
unsigned int i; for(i=0;i<strlen(password);++i)
{
password[i]=password[i]+key;
}
}
int main()
{
char password[20]; printf("Enter the password:\n"); scanf("%s",password);
printf("Password=%s\n",password); encrypt(password, 0XFACA); printf("Encrypted value=%s\n",password); decrypt(password, 0XFACA); printf("Decrypted value=%s\n",password);

return 0;
} OUTPUT:
```

{

Another pgm :

```
#include<stdio.h> #include<conio.h> void main() {

int data[7],rec[7],i,c1,c2,c3,c;

printf ("this works for message of 4bits in size \n enter message bit one by one: "); scanf ("%d %d %d %d",& data[0],&data[1],&data[2],&data[4]); data[6]=data[0]^data[2]^data[4];

data[5]=data[0]^data[1]^data[4]; data[3]=data[0]^data[1]^data[2];

printf("\n the encoded bits are given below: \n"); for (i=0;i<7;i++) {

printf("%d ",data[i]);

}

printf("\n enter the received data bits one by one: "); for (i=0;i<7;i++) {

scanf("%d",& rec[i]);

}

c1=rec[6]^rec[4]^rec[2]^rec[0]; c2=rec[5]^rec[4]^rec[1]^rec[0];
c3=rec[3]^rec[2]^rec[1]^rec[0]; c=c3*4+c2*2+c1 ;

if(c==0) {

printf ("\n congratulations there is no error: ");

} else {

printf("\n error on the position: %d\n the correct message is \n",c); if(rec[7-c]==0)

rec[7-c]=1; else rec[7-c]=0;

for (i=0;i<7;i++) { printf("%d ",rec[i]);

}

}
```

{

getch();

}

Output with error

Output without error

Result:

.

‘

Implement Client Server model using FTP protocol

The entire process can be broken down into following steps:

TCP Server –

1. using create(), Create TCP socket.
2. using bind(), Bind the socket to server address.
3. using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection
4. using accept(), At this point, connection is established between client and server, and they are ready to transfer data.
5. Go back to Step 3.

TCP Client –

1. Create TCP socket.
2. connect newly created client socket to server.

TCP Server:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
void func(int connfd)
{
    char buff[MAX];
    int n;
    // infinite loop for chat
    for (;;) {
        bzero(buff, MAX);

        // read the message from client and copy it in buffer
        read(connfd, buff, sizeof(buff));
        // print buffer which contains the client contents
        printf("From client: %s\t To client : ", buff);
        bzero(buff, MAX);
        n = 0;
        // copy server message in the buffer
        while ((buff[n++] = getchar()) != '\n')
            ;
```

```
{

        // and send that buffer to client
        write(connfd, buff, sizeof(buff));

        // if msg contains "Exit" then server exit and chat ended.
        if (strncmp("exit", buff, 4) == 0) {
            printf("Server Exit...\n");
            break;
        }
    }
}

// Driver function
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

    // Binding newly created socket to given IP and verification
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");

    // Now server is ready to listen and verification
    if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
    }
    else
        printf("Server listening..\n");
    len = sizeof(cli);
```

```
{
    // Accept the data packet from client and verification
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("server accept failed...\n");
        exit(0);
    }
    else
        printf("server accept the client...\n");

    // Function for chatting between client and server
    func(connfd);

    // After chatting close the socket
    close(sockfd);
}
```

TCP Client:

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
```

```
{
    }
}

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
        != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    // function for chat
    func(sockfd);

    // close the socket
    close(sockfd);
}
```

Compilation –
Server side:
gcc server.c -o server
./server
Client side:
gcc client.c -o client
./client

Output –
Server side:

‘

Socket successfully created..

Socket successfully binded..

Server listening..

server accept the client...

From client: hi

To client : hello

From client: exit

To client : exit

Server Exit...

Client side:

Socket successfully created..

connected to the server..

Enter the string : hi

From Server : hello

Enter the string : exit

From Server : exit

Client Exit...

Implement and realize the Network Topology - Star, Bus and Ring using NS2

NS2 programs are generally written using two major languages i.e C++ and TCL. NS2 supports both the types of Network i.e. wired and wireless networks. In wired networks, nodes use wired channel where as in wireless channel, node transmission takes place through

wireless channels. Major aspect of any simulation is the creation of Nodes and topology. We have focused here on all the types of topology and its respective codes for students to get an idea about NS2 simulation. It's just a one way of our teaching to make you know few basics of NS2. You can approach us for any particular concept or code in NS2.

Example 1: Formation of Bus Topology

#Creating five nodes
set node1 [$ns node]
setnode2 [$ns node]
set node3 [$ns node]
setnode4 [$ns node]
set node5 [$ns node]
#Creating Lan connection between the nodes
set lan0 [$ns newLan "$node1 $node2$node3 $node4 $node5" 0.7Mb 20ms LL Queue/FQ M AC/Csma/Cd Channel]
#Creating a TCP agent and attaching it to node 1
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $node1 $tcp0
#Creating a TCP Sink agent for TCP and attaching it to node 3
set sink0 [new Agent/TCPSink]
$ns attach-agent $node3 $sink0
#Connecting the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
 # Creating a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$ cbr0 set interval_ 0.05
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start time"
$ ns at 5.5 "$cbr0 stop time"
#Here we call the finish procedure after 10 seconds of simulation time
$ns at 10.0 "End"
#Finally run the simulation
$ns run

Here, we have created four nodes and they are linked together using Bus topology. We have given the overall code for all the parameters used in the construction of bus topology.

Example 2: Formation of Ring Topology

#Creating six nodes
setnode1 [$ns node]
set node2 [$ns node]
setnode3 [$ns node]
set node4 [$ns node]
setnode5 [$ns node]
set node6 [$ns node]
#Creating links between the nodes
$ ns duplex-link $node1 $node2 1Mb 15ms FQ
$ns duplex-link $node2 $node3 1Mb 15ms FQ

‘

$ ns duplex-link $node3 $node4 1Mb 15ms FQ
$ns duplex-link $node4 $node5 1Mb 15ms FQ
$ ns duplex-link $node5 $node6 1Mb 15ms FQ
$ns duplex-link $node6 $node1 1Mb 15ms FQ\”**Forms Ring Topology”**

Here, we have created six nodes and are connected using Ring topology. The code may look similar to the early one, but it shows the difference in terms of its connectivity. All the nodes are connected in a ring manner, in order to have high data transmission rates.

Example 3: Formation of Mesh Topology

#Creating four nodes
set node1 [$ns node]
setnode2 [$ns node]
set node3 [$ns node]
setnode4 [$ns node]
#Creating links between the nodes
$ ns duplex-link $node1 $node2 1Mb 20ms FQ
$ns duplex-link $node1 $node3 1Mb 20ms FQ
$ ns duplex-link $node1 $node4 1Mb 20ms FQ
$ns duplex-link $node2 $node3 1Mb 20ms FQ
$ ns duplex-link $node2 $node4 1Mb 20ms FQ
$ns duplex-link $node3 $node4 1Mb 20ms FQ**“Forms Mesh Topology”**

Implement and perform the operation of CSMA/CD and CSMA/CA using NS2.

#Lan simulation

```
set ns [new Simulator]
#define color for data flows
$ns color 1 Blue
$ns color 2 Red
 #open tracefiles
set tracefile1 [open out_node60.tr w]
set winfile [open winfile w]
$ns trace-all $tracefile1
#open nam file
set namfile [open out_node60.nam w]
$ns namtrace-all $namfile
#define the finish procedure
proc finish {} {
global ns tracefile1 namfile
$ns flush-trace
close $tracefile1
close $namfile
```

```
{
exec nam out_node60.nam &
exit 0
}
#create sixty nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
set n10 [$ns node]
set n11 [$ns node]
set n12 [$ns node]
set n13 [$ns node]
set n14 [$ns node]
set n15 [$ns node]
set n16 [$ns node]
set n17 [$ns node]
set n18 [$ns node]
set n19 [$ns node]
set n20 [$ns node]
set n21 [$ns node]
set n22 [$ns node]
set n23 [$ns node]
set n24 [$ns node]
set n25 [$ns node]
set n26 [$ns node]
set n27 [$ns node]
set n28 [$ns node]
set n29 [$ns node]
set n30 [$ns node]
set n31 [$ns node]
set n32 [$ns node]
set n33 [$ns node]
set n34 [$ns node]
set n35 [$ns node]
set n36 [$ns node]
set n37 [$ns node]
set n38 [$ns node]
set n39 [$ns node]
set n40 [$ns node]
set n41 [$ns node]
```

{

```
set n42 [$ns node]
set n43 [$ns node]
set n44 [$ns node]
set n45 [$ns node]
set n46 [$ns node]
set n47 [$ns node]
set n48 [$ns node]
set n49 [$ns node]
set n50 [$ns node]
set n51 [$ns node]
set n52 [$ns node]
set n53 [$ns node]
set n54 [$ns node]
set n55 [$ns node]
set n56 [$ns node]
set n57 [$ns node]
set n58 [$ns node]
set n59 [$ns node]
set n60 [$ns node]
$n1 color Red
$n1 shape box
#create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
$ns duplex-link $n10 $n21 2Mb 10ms DropTail
$ns duplex-link $n30 $n22 2Mb 10ms DropTail
$ns duplex-link $n8 $n21 2Mb 10ms DropTail
$ns duplex-link $n9 $n23 2Mb 10ms DropTail
$ns simplex-link $n20 $n30 0.3Mb 100ms DropTail
$ns simplex-link $n20 $n31 0.3Mb 100ms DropTail
$ns simplex-link $n26 $n32 0.3Mb 100ms DropTail
$ns simplex-link $n23 $n33 0.3Mb 100ms DropTail
$ns duplex-link $n7 $n24 2Mb 10ms DropTail
$ns duplex-link $n6 $n25 2Mb 10ms DropTail
$ns duplex-link $n4 $n26 2Mb 10ms DropTail
$ns duplex-link $n47 $n27 2Mb 10ms DropTail
$ns simplex-link $n21 $n34 0.3Mb 100ms DropTail
$ns simplex-link $n21 $n35 0.3Mb 100ms DropTail
$ns simplex-link $n21 $n36 0.3Mb 100ms DropTail
$ns simplex-link $n22 $n37 0.3Mb 100ms DropTail
$ns duplex-link $n45 $n28 2Mb 10ms DropTail
$ns duplex-link $n36 $n28 2Mb 10ms DropTail
$ns duplex-link $n35 $n29 2Mb 10ms DropTail
$ns duplex-link $n53 $n30 2Mb 10ms DropTail
```

{

```
$ns simplex-link $n21 $n39 0.3Mb 100ms DropTail
$ns simplex-link $n30 $n31 0.3Mb 100ms DropTail
$ns simplex-link $n42 $n34 0.3Mb 100ms DropTail
$ns simplex-link $n53 $n43 0.3Mb 100ms DropTail
$ns duplex-link $n54 $n31 2Mb 10ms DropTail
$ns duplex-link $n55 $n32 2Mb 10ms DropTail
$ns duplex-link $n59 $n33 2Mb 10ms DropTail
$ns duplex-link $n44 $n34 2Mb 10ms DropTail
$ns simplex-link $n52 $n53 0.3Mb 100ms DropTail
$ns simplex-link $n22 $n13 0.3Mb 100ms DropTail
$ns simplex-link $n12 $n23 0.3Mb 100ms DropTail
$ns simplex-link $n2 $n44 0.3Mb 100ms DropTail
$ns duplex-link $n26 $n35 2Mb 10ms DropTail
$ns duplex-link $n14 $n36 2Mb 10ms DropTail
$ns duplex-link $n12 $n37 2Mb 10ms DropTail
$ns duplex-link $n11 $n45 2Mb 10ms DropTail
$ns simplex-link $n2 $n47 0.3Mb 100ms DropTail
$ns simplex-link $n2 $n49 0.3Mb 100ms DropTail
$ns simplex-link $n2 $n56 0.3Mb 100ms DropTail
$ns simplex-link $n2 $n50 0.3Mb 100ms DropTail
```

#set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL
Queue/DropTail MAC/Csma/Cd Channel]

```
set lan [$ns newLan "$n0
$n1 $n2 $n3 $n4 $n5 $n7
$n8 $n9 $n10 $n11 $n12
$n13 $n14 $n15 $n16
$n18 $n19 $n20 $n21
$n22 $n23 $n25 $n26
$n27 $n28 $n29 $n30
$n31 $n32 $n33 $n34
$n35 $n36 $n37 " 0.5Mb
40ms LL
Queue/DropTail
MAC/Csma/Cd Channel]
#Give node position
#$ns duplex-link-op $n0
$n2 orient right-down
#$ns duplex-link-op $n1
$n2 orient right-up
#$ns simplex-link-op $n2
$n3 orient right
#$ns simplex-link-op $n3
$n2 orient left
```

{

```
$ns duplex-link-op $n0
$n2 orient right-down
$ns duplex-link-op $n1
$n2 orient right-down
$ns simplex-link-op $n2
$n3 orient right-down
$ns simplex-link-op $n3
$n2 orient right-down
$ns duplex-link-op $n10
$n21 orient right-down
$ns duplex-link-op $n30
$n22 orient right-down
$ns duplex-link-op $n8
$n21 orient right-down
$ns duplex-link-op $n9
$n23 orient right-down
$ns simplex-link-op $n20
$n30 orient right-down
$ns simplex-link-op $n20
$n31 orient right-down
$ns simplex-link-op $n26
$n32 orient right-down
$ns simplex-link-op $n23
$n33 orient right-down
$ns duplex-link-op $n7
$n24 orient right-up
$ns duplex-link-op $n6
$n25 orient right-up
$ns duplex-link-op $n4
$n26 orient right-up
$ns duplex-link-op $n47
$n27 orient right-up
$ns simplex-link-op $n21
$n34 orient right-up
$ns simplex-link-op $n21
$n35 orient right-up
$ns simplex-link-op $n21
$n36 orient right-up
$ns simplex-link-op $n22
$n37 orient right-up
$ns duplex-link-op $n45
$n28 orient right
$ns duplex-link-op $n36
$n28 orient right
$ns duplex-link-op $n35
$n29 orient right
```

{

```
$ns duplex-link-op $n53
$n30 orient right
$ns simplex-link-op $n21
$n39 orient right
$ns simplex-link-op $n30
$n31 orient right
$ns simplex-link-op $n42
$n34 orient right
$ns simplex-link-op $n53
$n43 orient right
$ns duplex-link-op $n54
$n31 orient left
$ns duplex-link-op $n55
$n32 orient left
$ns duplex-link-op $n59
$n33 orient left
$ns duplex-link-op $n44
$n34 orient left
$ns simplex-link-op $n52
$n53 orient left
$ns simplex-link-op $n22
$n13 orient left
$ns simplex-link-op $n12
$n23 orient left
$ns simplex-link-op $n2
$n44 orient left
#set queue size of
link(n2-n3) to 20
$ns queue-limit $n2 $n3
20
 #setup TCP connection
set tcp [new
Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new
Agent/TCPSink/DelAck]
$ns attach-agent $n4
$sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set packet_size_
1000
#set ftp over tcp
connection
set ftp [new
Application/FTP]
```

{

```
$ftp attach-agent $tcp
#setup a UDP connection
set udp [new
Agent/UDP]
$ns attach-agent $n1
$udp
set null [new Agent/Null]
$ns attach-agent $n5
$null
$ns connect $udp $null
$udp set fid_ 2
#setup a CBR over UDP
connection
set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_
1000
$cbr set rate_ 0.01Mb
$cbr set random_ false
#scheduling the events
$ns at 0.1 "$cbr start"
$ns at 0.3 "$ftp start"
$ns at 90.0 "$ftp stop"
$ns at 100.0 "$cbr stop"
proc plotWindow
{tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set
cwnd_]
puts $file "$now $cwnd"
 $ns at [expr
$now+$time]
"plotWindow $tcpSource
$file"
}
$ns at 0.1 "plotWindow
$tcp $winfile"
 $ns at 100.0 "finish"
 $ns run
```

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3461 COMMUNICATION SYSTEMS LABORATORY

Semester - 04

LABORATORY MANUAL

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

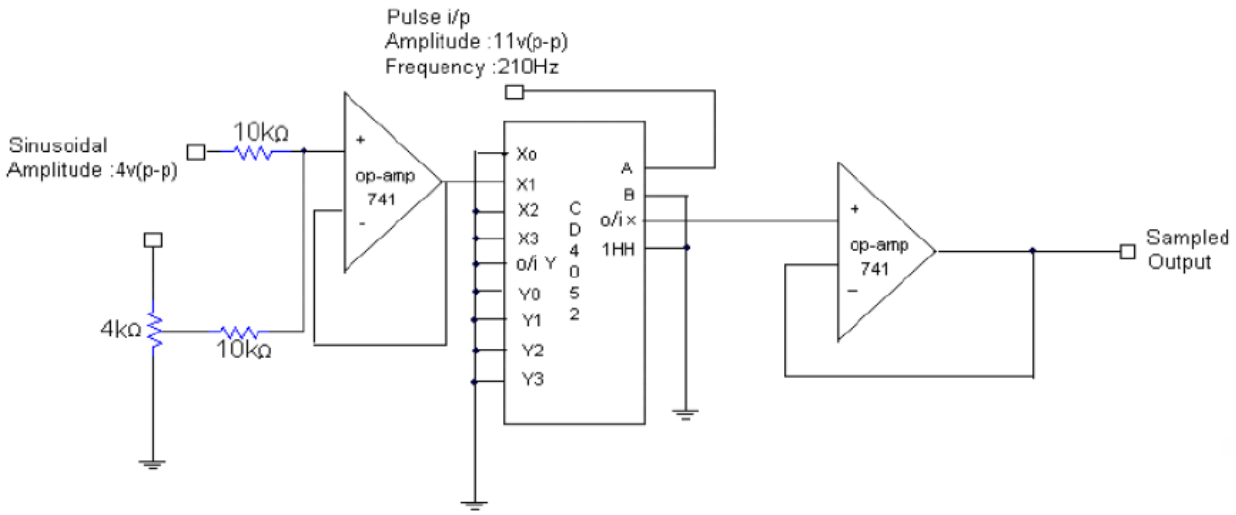
PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

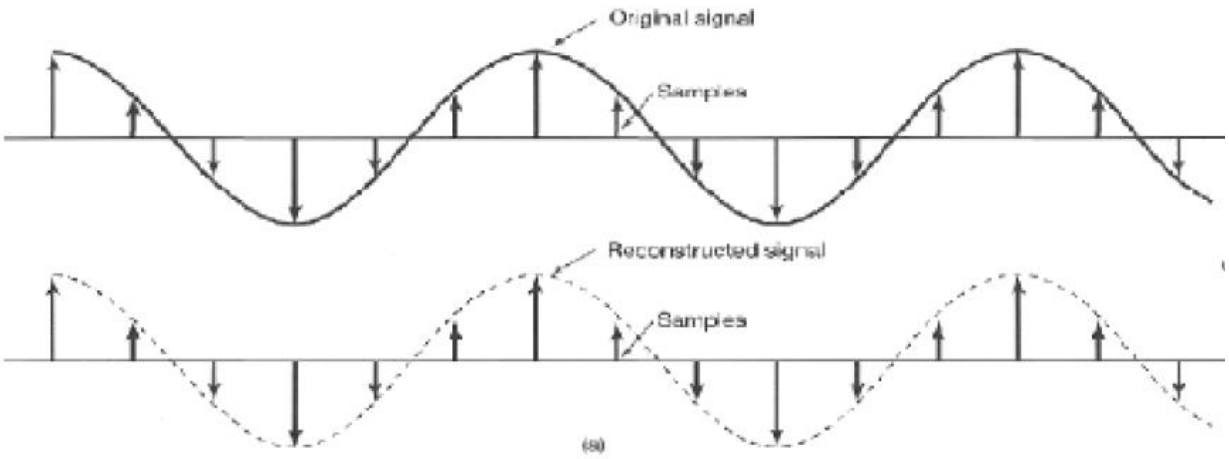
1. AM- Modulator and Demodulator
2. FM - Modulator and Demodulator
3. Pre-Emphasis and De-Emphasis.
4. Signal sampling and TDM.
5. Pulse Code Modulation and Demodulation.
6. Pulse Amplitude Modulation and Demodulation.
7. Pulse Position Modulation and Demodulation and Pulse Width Modulation and Demodulation.
8. Digital Modulation – ASK, PSK, FSK.
9. Delta Modulation and Demodulation.
10. Simulation of ASK, FSK, and BPSK Generation and Detection Schemes.
11. Simulation of DPSK, QPSK and QAM Generation and Detection Schemes.
12. Simulation of Linear Block and Cyclic Error Control coding Schemes.

CIRCUIT DIAGRAM:



Sampling Circuit

OUTPUT:



EX.NO :	SIGNAL SAMPLING AND RECONSTRUCTION
DATE :	

AIM:

To obtain the sampling of a signal and reconstruct the original signal from the samples.

APPARATUS REQUIRED:

Qualite Technologies Sampling Trainer kit - 1

Dual Trace Cathode Ray Oscilloscope - 1

CRO Probe - 2

THEORY:

Sampling of the signal is fundamental operation in signal processing. A continuous time signal is first converted to discrete time signal by sampling process. The sufficient no. of samples of the signal must be taken so that the original signal is represented in its samples completely. It should be possible to recover or reconstruct the original signal completely from its samples. The no. of samples to be taken depends on maximum signal frequency present in the signal. Sampling theorem gives the complete idea about sampling and reconstruction

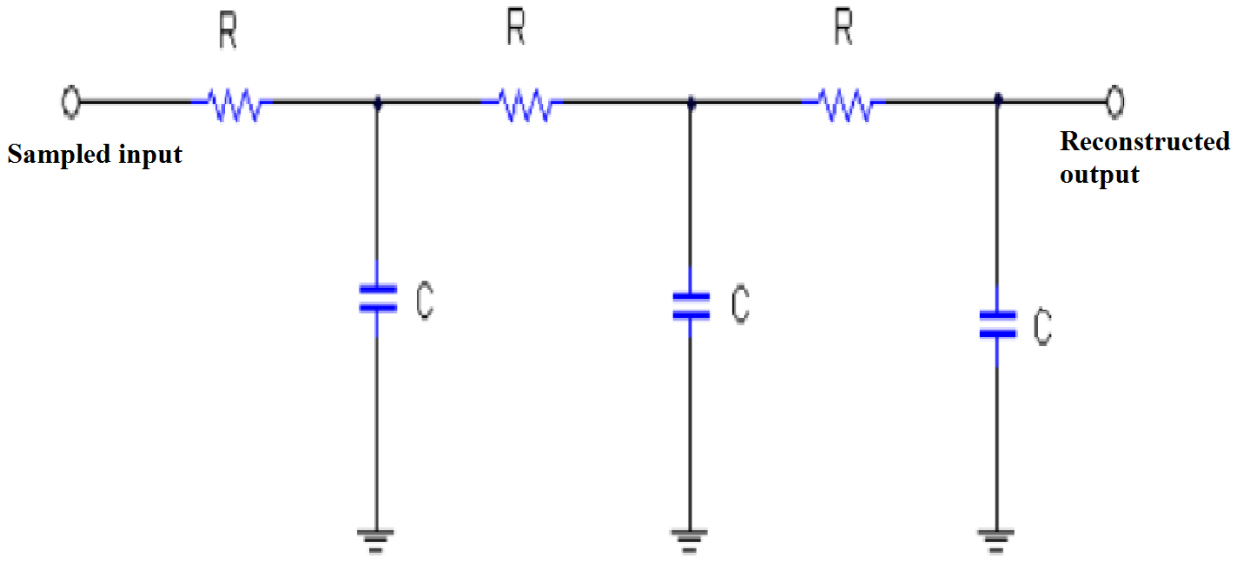


Fig. Reconstruction circuit

TABULATION:

	AMPLITUDE	TIME PERIOD
INPUT SIGNAL		
SAMPLING SIGNAL		
SAMPLED OUTPUT		
RECONSTRUCTED OUTPUT		

PROCEDURE:

1. Turn on the power to the trainer kit.

2. Apply the clock signal and the analog signal to the sampling circuit. And observe the output signal in the CRO.

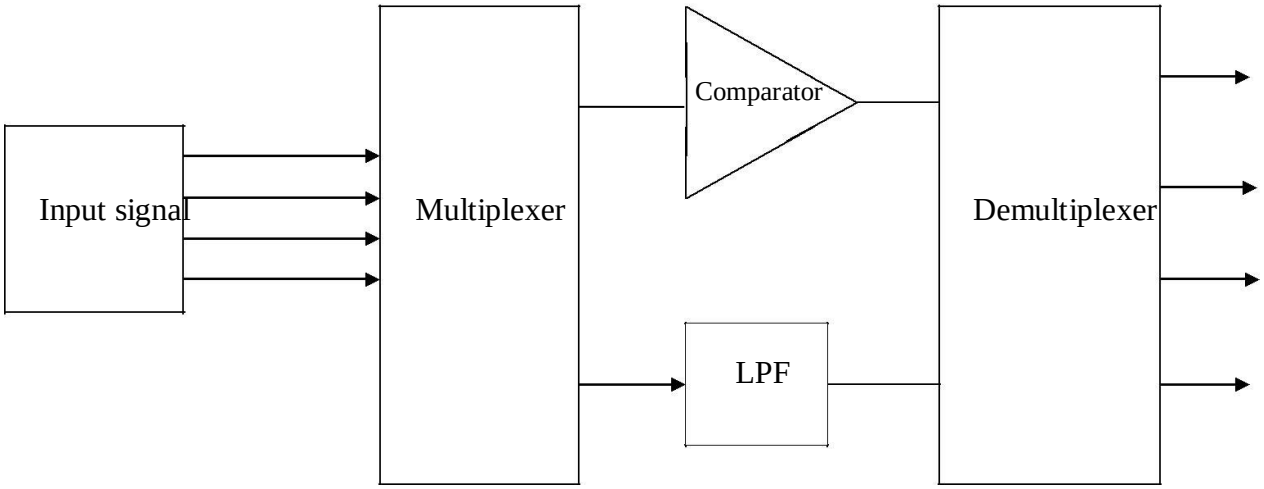
3. Apply the sampled output to the reconstructing circuit. Observe the output on the CRO.

4. Design the reconstructing circuit. Depending on sampling frequency, R & C values are calculated using the relations Fs = 1/Ts, Ts = RC. Choosing an appropriate value for C, R can be found using the relation R=Ts/C

RESULT:

Thus the sampling and reconstruction of the signal is obtained .

BLOCK DIAGRAM:



TABULATION:

		Amplitude	Time
TDM INPUT	X0		
	X1		
	X2		
	X3		
TDM OUTPUT	Y0		
	Y1		
	Y2		
	Y3		

EX.NO :	TIME DIVISION MULTIPLEXING AND DEMULTIPLEXING
DATE :	

AIM

To Study the Operation of Time Division Multiplexing and Demultiplexing using kit.

APPARATUS REQUIRED

TDM Trainer Kit-1.
Dual Trace CRO-1.
CRO Probe-2.

THEORY

Time-division multiplexing (TDM) is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line so that each signal appears on the line only a fraction of time in an alternating pattern.

TDM is used primarily for digital signals,but may be applied in analog multiplication in which two or more signals or bit streams are transferred appearing simultaneously as sub-channels in one communication channel, but are physically taking turns on the channel. The time domain is divided into several recurrent *time slots* of fixed length, one for each sub-channel. A sample byte or data block of sub-channel 1 is transmitted during time slot 1, sub-channel 2 during time slot 2, etc. One TDM frame consists of one time slot per sub-channel plus a synchronization channel and sometimes error correction channel before the synchronization. After the last sub-channel, error correction, and synchronization, the cycle starts all over again with a new frame, starting with the second sample, byte or data block from sub-channel 1.

Demultiplexer performs the reverse process of Multiplexing and routes the separated signals to their corresponding Receivers or Destinations.

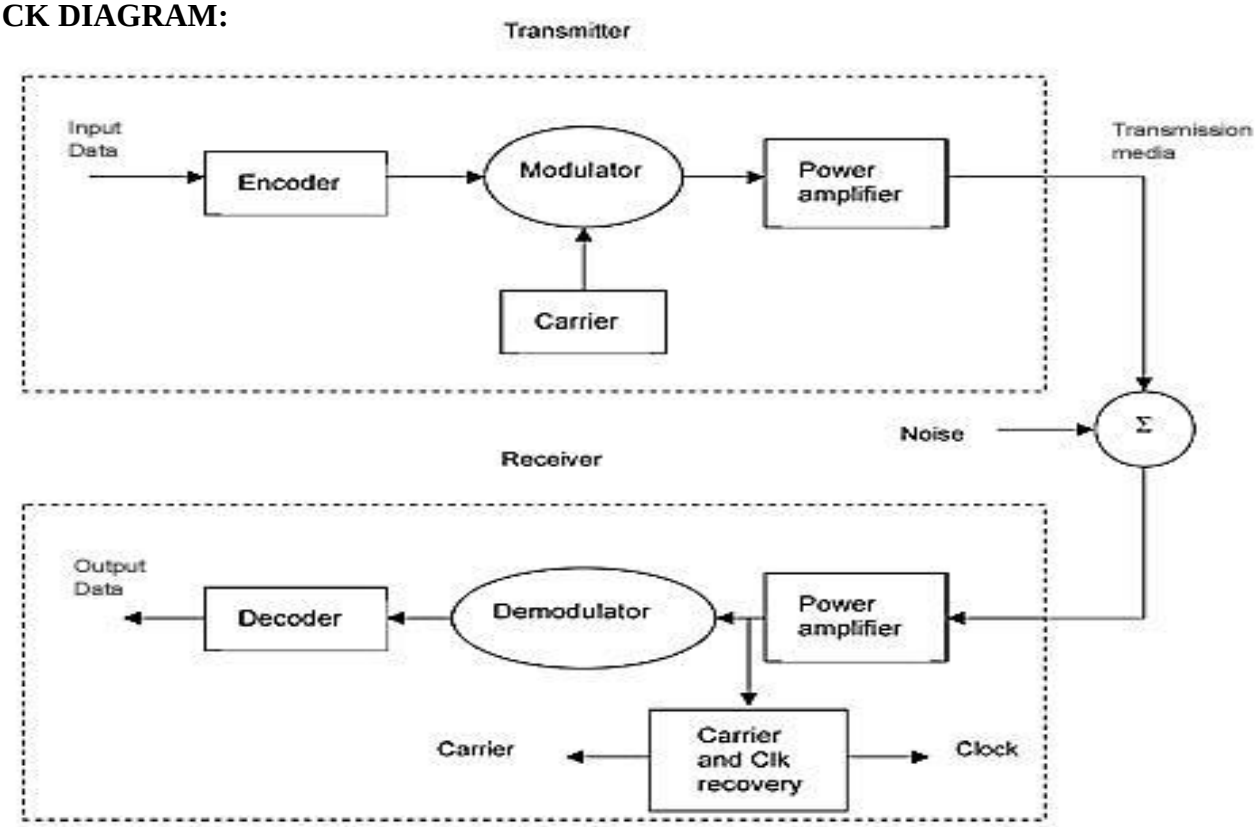
PROCEDURE:

1. Turn on the TDM Trainer Kit.
2. To give the Four input signals to the Multiplexer using Signal Generator.
3. Observe the Multiplexer (TDM) Output
4. Connect the TDM output to the Comparator. Observe the Comparator output. TDM pulses are now converted in PWM pulses.
5. Connect the PWM pulses to TDM input of demultiplexer and note the frequency & Amplitude. The PWM pulses corresponding to each channel are separated as four streams.
6. Take one output and connect it to the low pass filter(LPF) and amplifier block. Observe the output of the corresponding input.
7. Repeat this process for all inputs. This is the output of Demodulated TDM.

RESULT

Thus the Time division Multiplexing and Demultiplexing was studied and output wave forms are obtained.

BLOCK DIAGRAM:



Tabulation

	Amplitude	**Time**
INPUT SIGNAL		
CARRIER SIGNAL		
MODULATED SIGNAL		
DEMODULATED SIGNAL		

EX.NO :	AMPLITUDE MODULATION & DEMODULATION
DATE :	

AIM

To generate amplitude modulation and demodulation signals.

APPARATUS REQUIRED

Amplitude modulation trainer Kit
Dual Trace CRO-1.
CRO Probe-2.

THEORY

Amplitude Modulation is defined as a process in which the amplitude of the carrier wave c (t) is Varied linearly with the instantaneous amplitude of the message signal m (t).The standard form of an Amplitude modulated (AM) wave is defined by

$$s(t) = A_c\,[1 + K_a\, m(t) \cos(2\pi f_c\, t)]$$

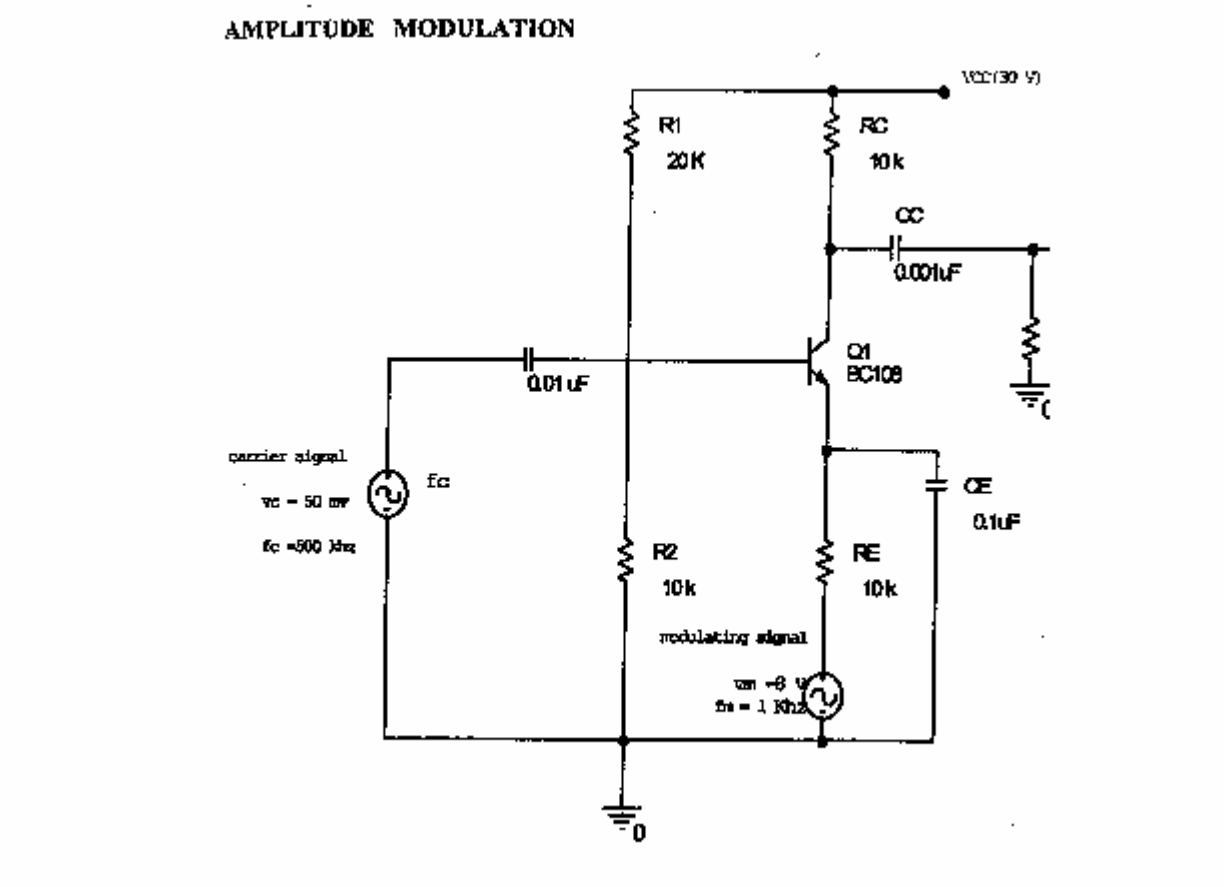
Where a K is a constant called the amplitude sensitivity of the modulator.

The demodulation circuit is used to recover the message signal from the incoming AM wave at the receiver. An envelope detector is a simple and yet highly effective device that is well suited for the demodulation of AM wave, for which the percentage modulation is less than 100%.Ideally, an envelop detector produces an output signal that follows the envelop of the input signal wave form exactly; hence, the name. Some version of this circuit is used in almost all commercial AM radio receivers.

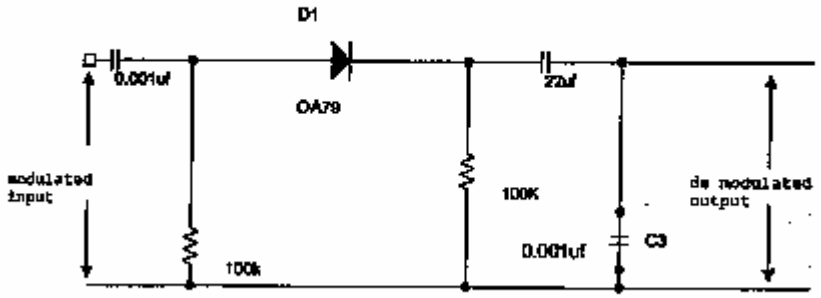
The Modulation Index is defined as, m = $\dfrac{(E_{\max} - E_{\min})}{(E_{\max} + E_{\min})}$

Where Emax and Emin are the maximum and minimum amplitudes of the modulated wave.

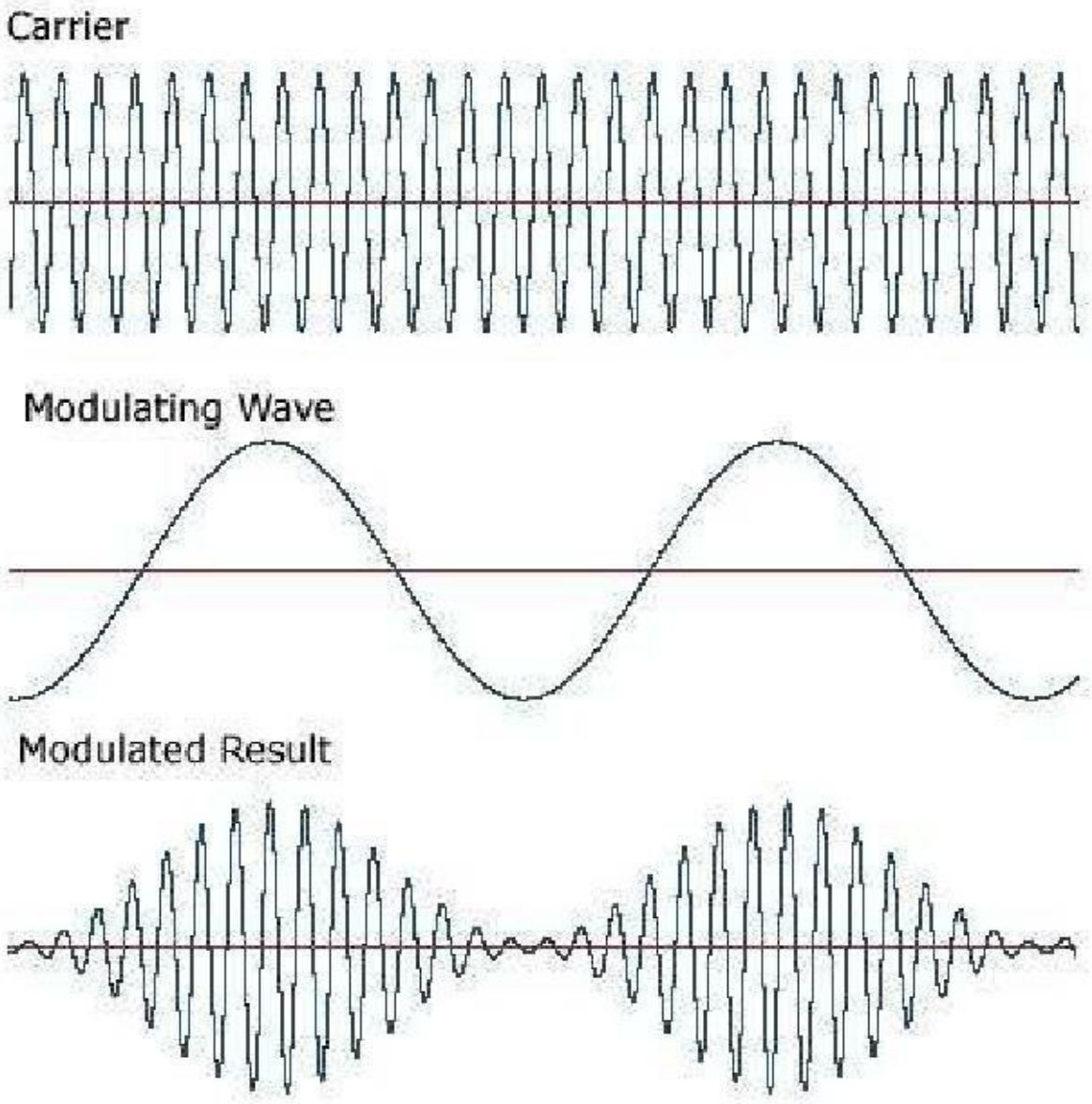
MODULATION CIRCUIT:



AM DETECTOR



OUTPUT WAVEFORM:



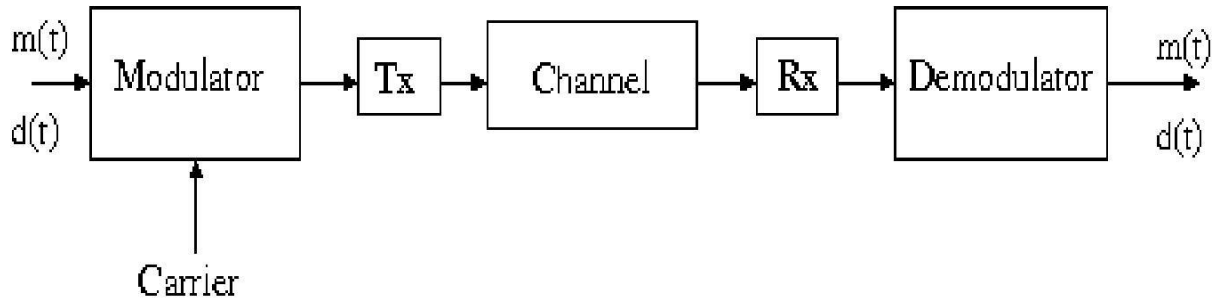
TABULATION

	AMPLITUDE	**TIME**
INPUT SIGNAL		
CARRIER SIGNAL		
MODULATED SIGNAL		
DEMODULATED SIGNAL		

RESULT

Thus the generation of Amplitude Modulation and demodulation was generated and output waveforms are obtained.

BLOCK DIAGRAM:



TABULATION:

	AMPLITUDE	**TIME**
INPUT SIGNAL		
CARRIER SIGNAL		
MODULATED SIGNAL		
DEMODULATED SIGNAL		

EX.NO :	FREQUENCY MODULATION & DEMODULATION
DATE :	

AIM

To generate Frequency modulation and demodulation signals.

APPARATUS REQUIRED

Frequency modulation trainer Kit
Dual Trace CRO-1.
CRO Probe-2.

THEORY

The process, in which the frequency of the carrier is varied in accordance with the instantaneous amplitude of the modulating signal, is called "Frequency Modulation". The FM signal is expressed as

$$s(t) = A_c \cos(2\pi f_c + \beta \sin(2\pi f_m t)$$

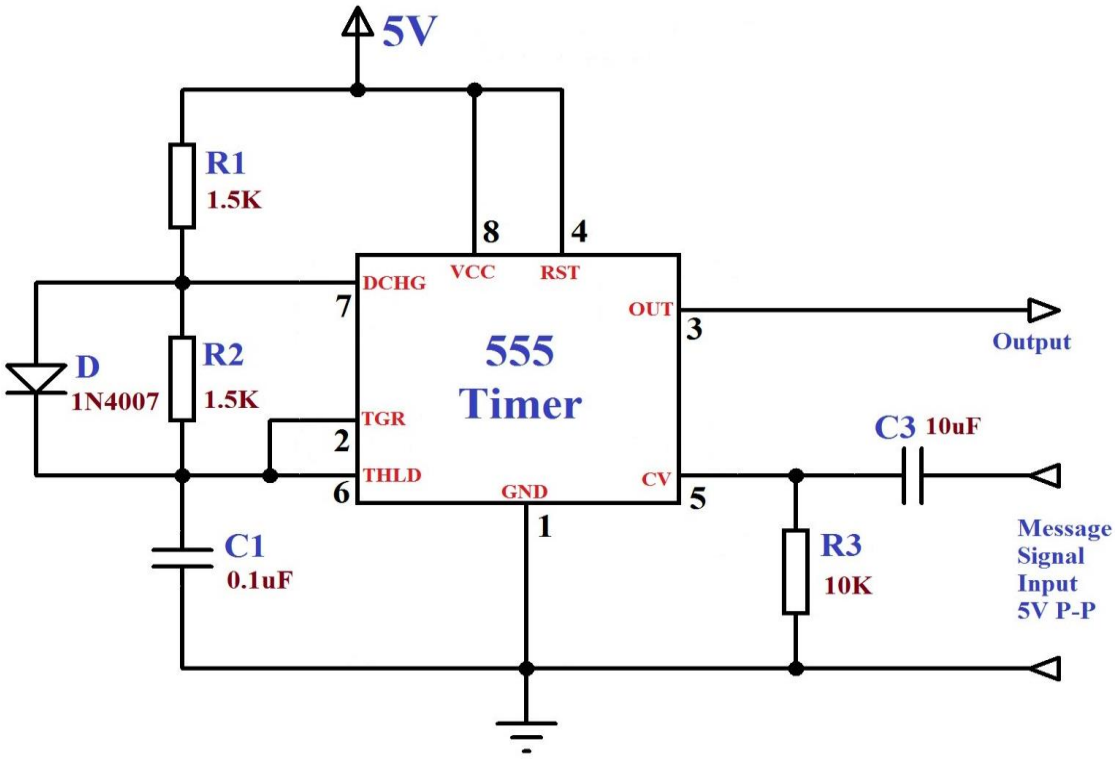
Where

A_C is amplitude of the carrier signal,

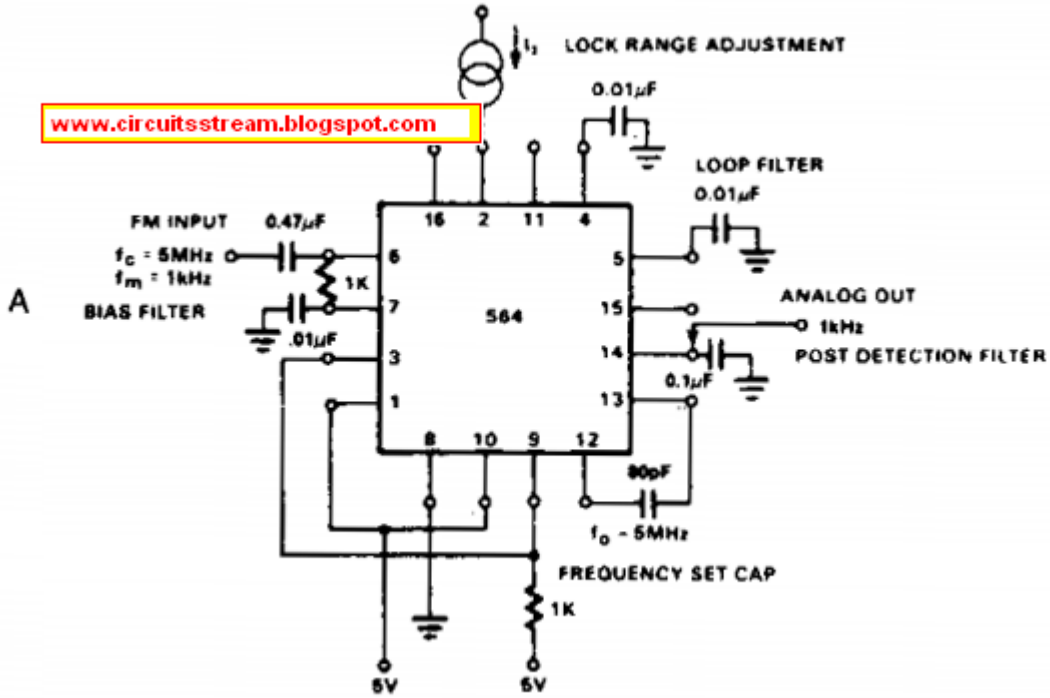
fc is the carrier frequency

β is the modulation index of the FM wave

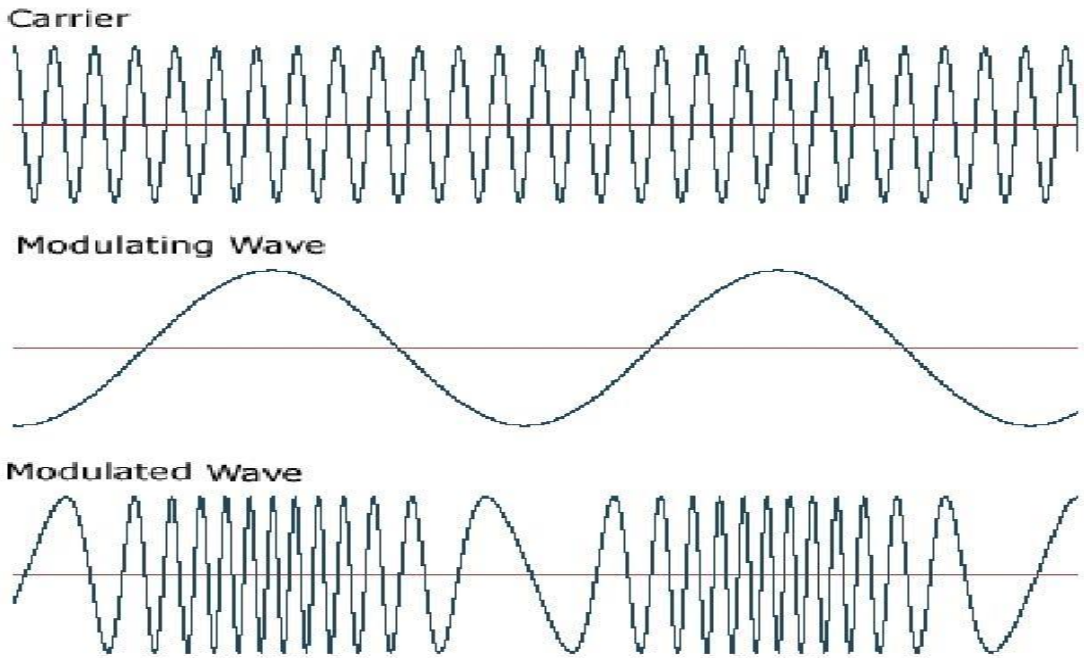
FM MODULATION CIRCUIT:



FM DEMODULATION CIRCUIT



Output Waveform



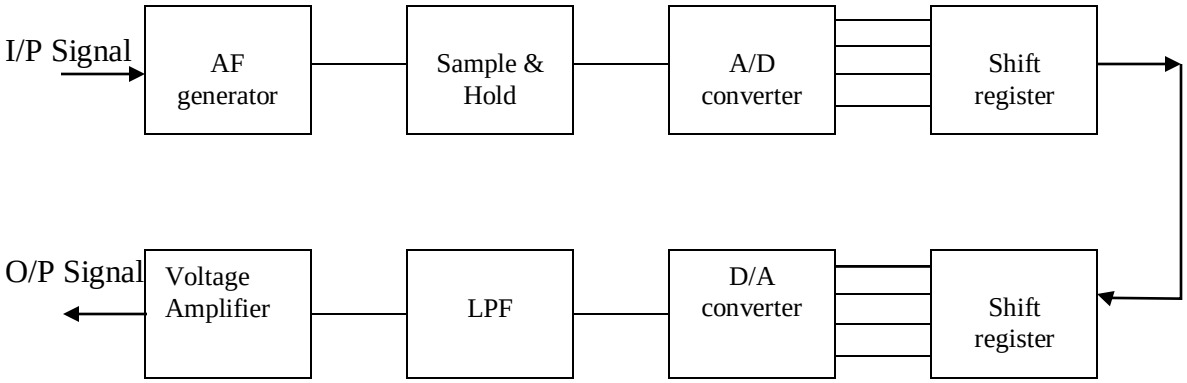
Tabulation

SIGNAL	AMPLITUDE	TIME
INPUT SIGNAL CARRIER SIGNAL MODULATED SIGNAL DEMODULATED SIGNAL		

RESULT

Thus the generation of frequency Modulation and demodulation was generated and output waveforms are obtained.

Block Diagram



TABULATION

	AMPLITUDE	**TIME**
ANALOG OUTPUT		
SERIAL DATA		
DAC OUTPUT		
FILTER OUTPUT		

EX.NO :	PULSE CODE MODULATION
DATE :	

AIM

To generate Pulse Code Modulated Signal and Demodulates to get the original signal.

APPARATUS REQUIRED

PCM Trainer Kit-1.
Dual Trace CRO-1.
CRO Probe.

THEORY

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. It is the standard form of digital audio in computers, Compact Discs, digital telephony and other digital audio applications. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest value within a range of digital steps.

A PCM stream has two basic properties that determine the stream's fidelity to the original analog signal: the sampling rate, which is the number of times per second that samples are taken; and the bit depth, which determines the number of possible digital values that can be used to represent each sample.

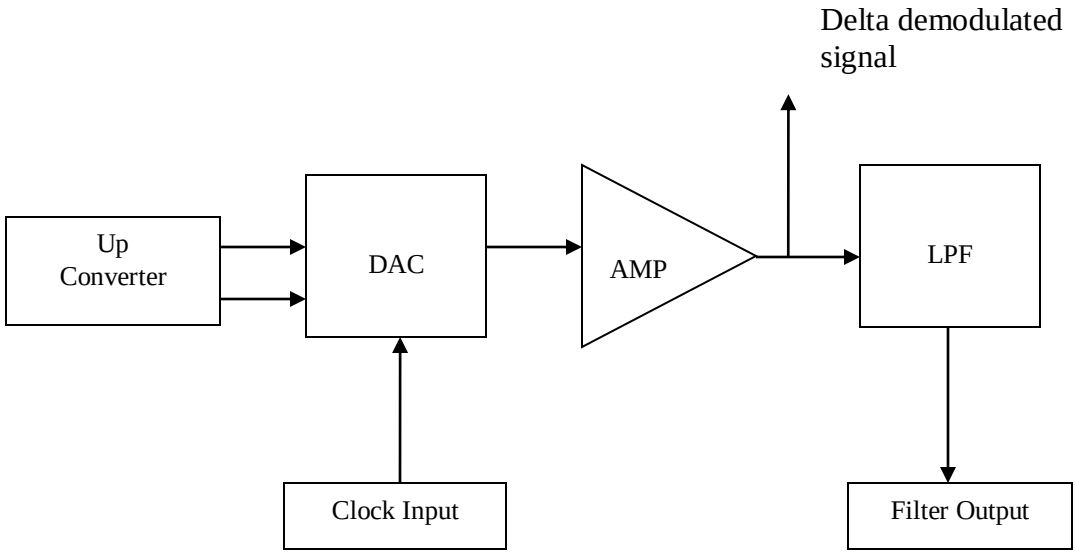
PROCEDURE

1. Turn on the trainer kit.
2. Get the modulated signal using trainer kit. Apply the modulated signal at the input
3. Observe the serial data and D/A output.
4. Note the demodulated output.
5. Compare the input and output signals

RESULT

Thus the pulse code modulationof the signal was generated and demodulates the original signal and the output wave form obtained and verified

Block Diagram



Tabulation

	Amplitude	**Time**
ANALOG OUTPUT		
SERIAL DATA		
DAC OUTPUT		
FILTER OUTPUT		

EX.NO :	DELTA MODULATION AND DEMODULATION
DATE :	

AIM

To Study the Operation of Delta Modulation and Demodulation.

APPARATUS REQUIRED

Delta modulation trainer Kit
Dual Trace CRO-1.
CRO Probe-2.

THEORY

A Delta modulation (DM or -modulation) is an analog-to- digital and digital-to- analog signal conversion technique used for transmission of voice information where quality is not of primary importance. DM is the simplest form of (DPCM) where the differences between successive samples are encoded into n-bit data streams. In delta modulation, the transmitted data are reduced to a 1-bit data stream. Its main features are:

- The analog signal is approximated with a series of segments
- Each segment of the approximated signal is compared to the original analog wave to determine the increase or decrease in relative amplitude
- The decision process for establishing the state of successive bits is determined by this comparison
- Only the change is sent, that is, only an increase or decrease of the signal amplitude from the previous sample is sent whereas a no-change condition causes the modulated signal to remain at the same 0 or 1 state of the previous sample.

To achieve delta modulation must use techniques, that is, the analog signal is sampled at a rate several times higher than the.

RESULT

Thus the Delta modulation and Demodulation was studied and output wave forms are obtained.

EX.NO :	SIMULATION OF BPSK
DATE :	

AIM

To Simulate the BPSK using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY

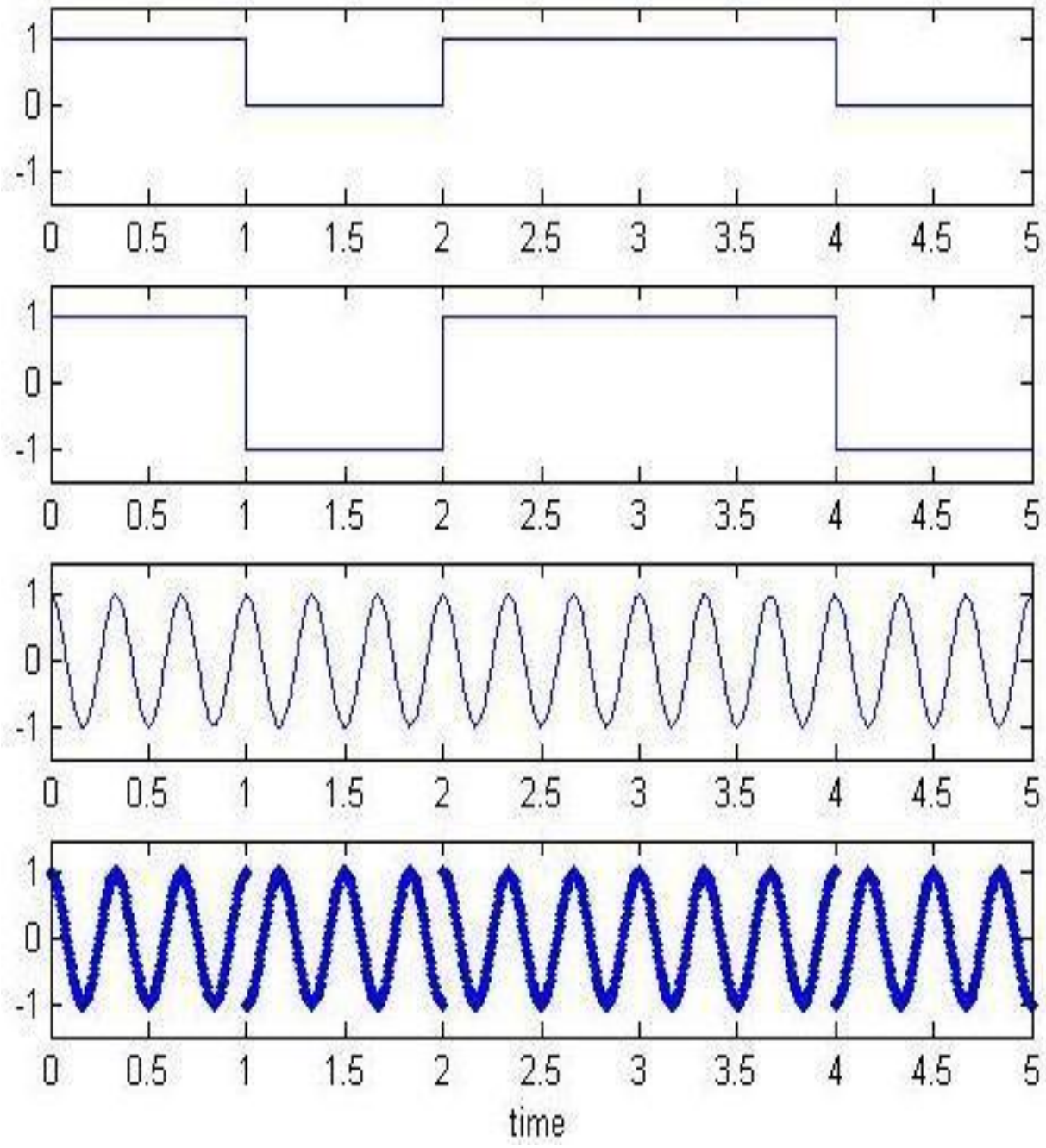
BPSK (also sometimes called PRK, phase reversal keying, or 2PSK) is the simplest form of phase shift keying (PSK). It uses two phases which are separated by 180° and so can also be termed 2-PSK. It does not particularly matter exactly where the constellation points are positioned, and in this figure they are shown on the real axis, at 0° and 180°. This modulation is the most robust of all the PSKs since it takes the highest level of noise or distortion to make the demodulator reach an incorrect decision. It is, however, only able to modulate at 1 bit/symbol (as seen in the figure) and so is unsuitable for high data-rate applications.In the presence of an arbitrary phase-shift introduced by the communications channel, the demodulator is unable to tell which constellation point is which. As a result, the data is often differentially encoded prior to modulation.BPSK is functionally equivalent to 2-QAM modulation.

Coding

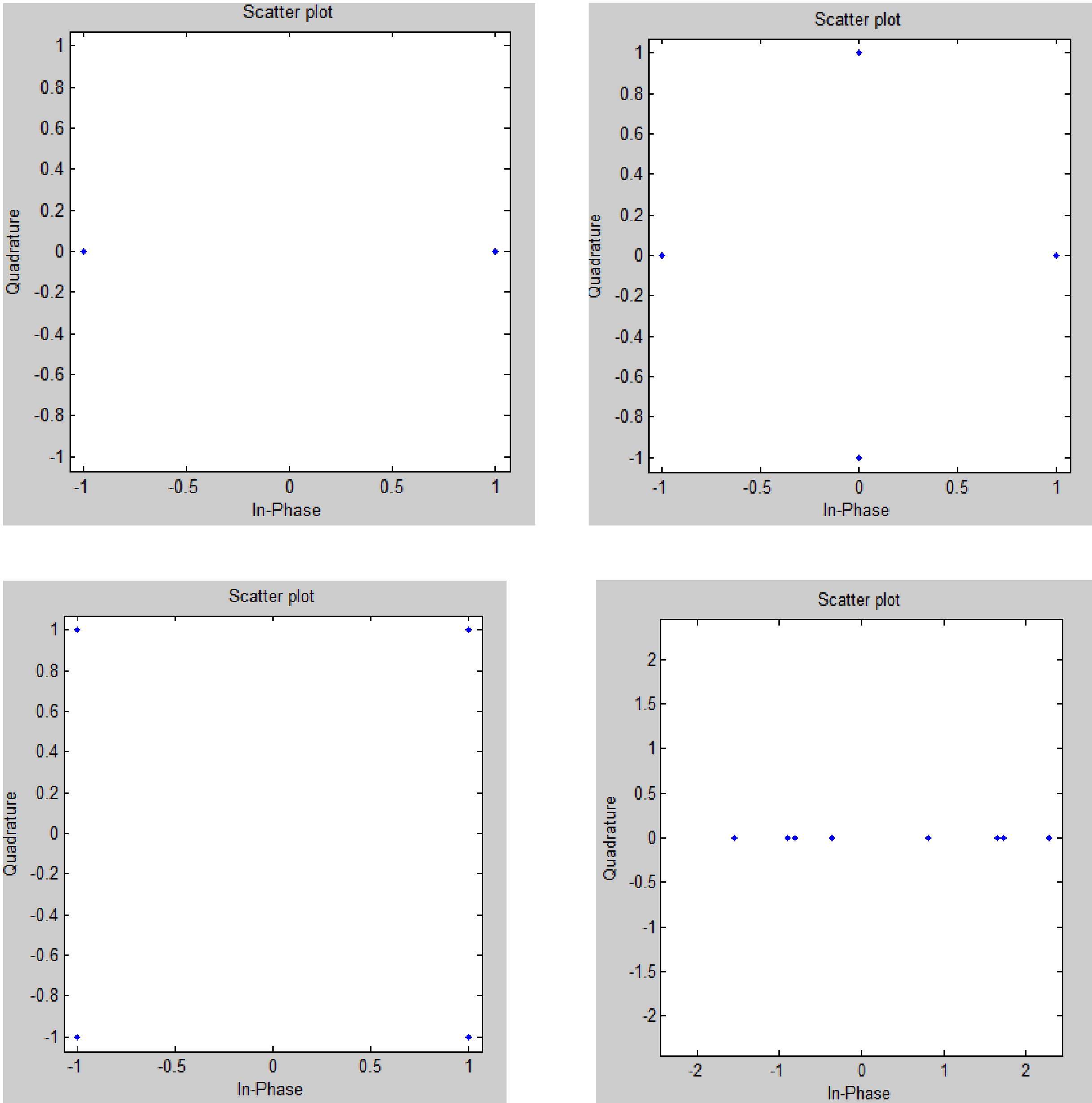
```
clc;
clear all;
close all;
d=[1 0 1 1 0];                  % Data sequence
b=2*d-1;                        % Convert unipolar to bipolar
T=1;                            % Bit duration
Eb=T/2;                         % This will result in unit amplitude waveforms
fc=3/T;                         % Carrier frequency
t=linspace(0,5,1000);           % discrete time sequence between 0 and 5*T
                                (1000 samples)
N=length(t);                    % Number of samples
Nsb=N/length(d);                % Number of samples per bit
dd=repmat(d',1,Nsb);            % replicate each bit Nsb times
bb=repmat(b',1,Nsb); dw=dd';    % Transpose the rows and columns
dw=dw(:)';                       % Convert dw to a column vector (colum by column)
                                and convert to a row vector

bw=bb';
bw=bw(:)';                      % Data sequence samples
w=sqrt(2*Eb/T)*cos(2*pi*fc*t);  % carrier waveform
bpsk_w=bw.*w;                    % modulated waveform

% plotting commands follow

subplot(4,1,1);
plot(t,dw); axis([0 5 -1.5 1.5])

subplot(4,1,2);
plot(t,bw); axis([0 5 -1.5 1.5])

subplot(4,1,3);
plot(t,w); axis([0 5 -1.5 1.5])

subplot(4,1,4);
plot(t,bpsk_w,'.'); axis([0 5 -1.5 1.5])
xlabel('time')

% Constellation plot for BPSK, QPSK and QAM
 clear all
 close all
 N = 8;
 BPSK_Const = [ 1 -1];
 QPSK_Const = [ 1 j -1 -j];
 QAM_Const = [1+j, 1-j, -1-j,-1+j];
  BPSK_Symb = randsrc(1,N,BPSK_Const);
```

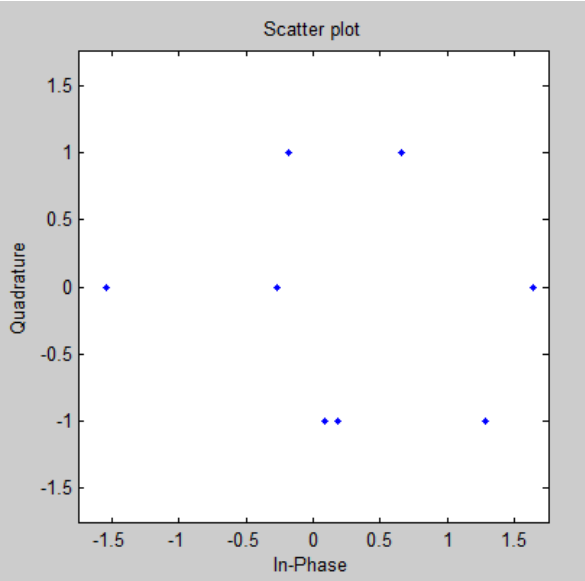
```
 QPSK_Symb = randsrc(1,N,QPSK_Const);
 QAM_Symb = randsrc(1,N,QAM_Const);
  %without Noise
 scatterplot(BPSK_Symb)
figure
 scatterplot(QPSK_Symb)
figure
 scatterplot(QAM_Symb)
figure
% with noise
Noise = randn(1,N)
 BPSK_Symb = randsrc(1,N,BPSK_Const)+Noise;
 QPSK_Symb = randsrc(1,N,QPSK_Const)+Noise;
 QAM_Symb = randsrc(1,N,QAM_Const)+Noise;
   %without Noise
 scatterplot(BPSK_Symb)
figure
 scatterplot(QPSK_Symb)
figure
 scatterplot(QAM_Symb)
figure
```

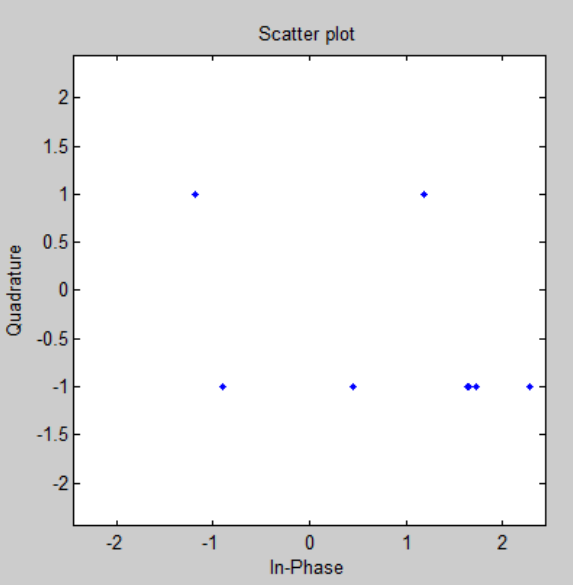
Output



CONSTELLATION DIAGRAM:




Scatter plot
Quadrature
In-Phase
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5


Scatter plot
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2

RESULT

Thus the **B**PSK was simulated using MATLAB and output waveforms are obtained output verified.

EX.NO :	SIMULATION OF QPSK
DATE :	

AIM

To Simulate the QPSK using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY

QPSK uses four points on the constellation diagram, equispaced around a circle. With four phases, QPSK can encode two bits per symbol, shown in the diagram with Gray coding to minimize the bit error rate (BER) — sometimes misperceived as twice the BER of BPSK.The mathematical analysis shows that QPSK can be used either to double the data rate compared with a BPSK system while maintaining thesame bandwidth of the signal, or to maintain the data-rate of BPSK but halving the bandwidth needed. In this latter case, the BER of QPSK isexactly the same as the BER of BPSK - and deciding differently is a common confusion when considering or describing QPSK. The transmitted carrier can undergo numbers of phase changes.

Given that radio communication channels are allocated by agencies such as the Federal Communication Commission giving a prescribed (maximum) bandwidth, the advantage of QPSK over BPSK becomes evident: QPSK transmits twice the data rate in a given bandwidth compared to BPSK - at the same BER. The engineering penalty that is paid is that QPSK transmitters and receivers are more complicated than the ones for BPSK. However, with modern electronics technology, the penalty in cost is very moderate.

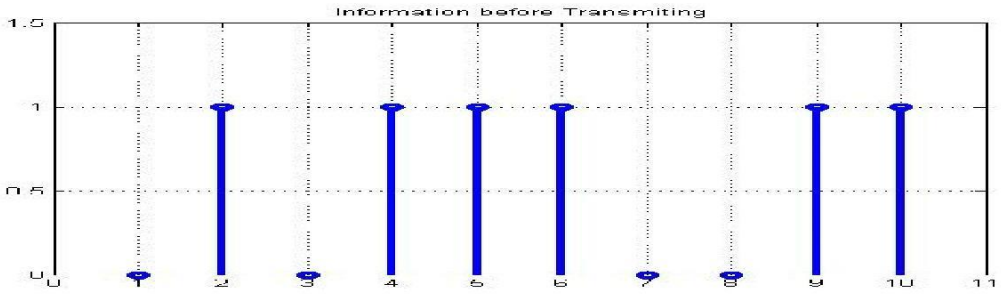
As with BPSK, there are phase ambiguity problems at the receiving end, and differentially encoded QPSK is often used in practice.

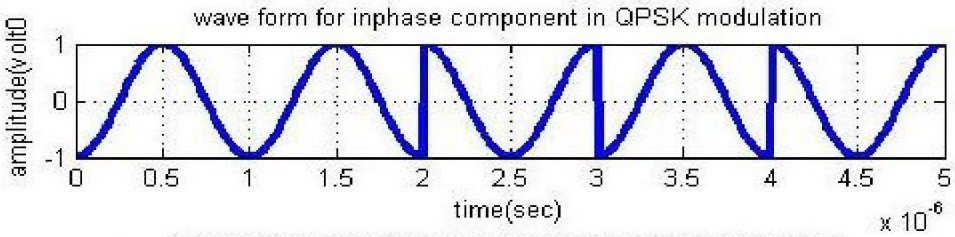
Coding

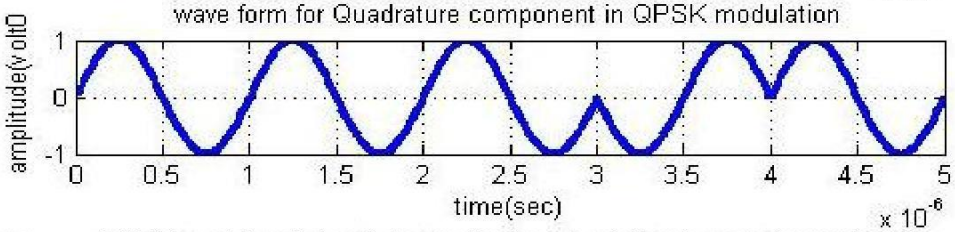
```
% QPSK Modulation and Demodulation
clc;
clear all;
close all;
data=[0 1 0 1 1 1 0 0 1 1];    %Number_of_bit=1024;
                               %data=randint(Number_of_bit,1);
figure(1)
stem(data, 'linewidth',3), grid on;
title(' Information before Transmiting ');
axis([ 0 11 0 1.5]);
 data_NZR=2*data-1;                          % Data Represented at NZR form for QPSK
                                             modulation
s_p_data=reshape(data_NZR,2,length(data)/2);  % S/P convertion of data
 br=10.^6;                                   %Let us transmission bit rate 1000000
f=br;                                        % minimum carrier frequency
T=1/br;                                      % bit duration
t=T/99:T/99:T;                               % Time vector for one bit information

% QPSK modulation

 y=[];
y_in=[];
y_qd=[];
for(i=1:length(data)/2)
   y1=s_p_data(1,i)*cos(2*pi*f*t);           % inphase component
   y2=s_p_data(2,i)*sin(2*pi*f*t) ;          % Quadrature component
   y_in=[y_in y1];                           % inphase signal vector
   y_qd=[y_qd y2];                           %quadrature signal vector
   y=[y y1+y2];                              % modulated signal vector
end
Tx_sig=y;                                    % transmitting signal after modulation
tt=T/99:T/99:(T*length(data))/2;
 figure(2)
 subplot(3,1,1); plot(tt,y_in,'linewidth',3),
grid on;
title(' wave form for inphase component in QPSK modulation ');
xlabel('time(sec)');
ylabel(' amplitude(volt0');
```

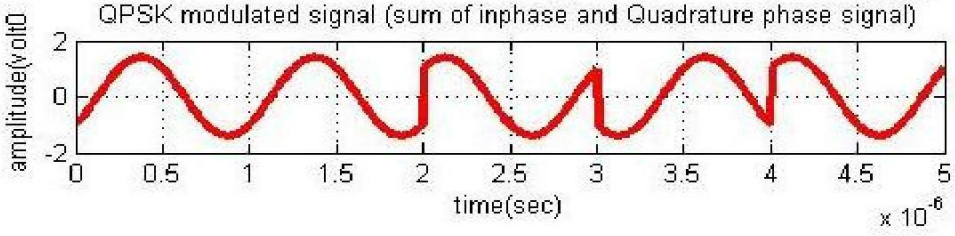
```
subplot(3,1,2); plot(tt,y_qd,'linewidth',3), grid on;
title(' wave form for Quadrature component in QPSK modulation '); xlabel('time(sec)');
ylabel(' amplitude(volt0'); subplot(3,1,3);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('QPSK modulated signal (sum of inphase and Quadrature phase signal)'); xlabel('time(sec)');
ylabel(' amplitude(volt0');

% QPSK demodulation

Rx_data=[];
Rx_sig=Tx_sig; % Received signal for(i=1:1:length(data)/2)

% inphase coherent dector Z_in=Rx_sig((i-
 1)*length(t)+1:i*length(t)).*cos(2*pi*f*t);
% above line indicat multiplication of received & inphase carred signal
Z_in_intg=(trapz(t,Z_in))*(2/T);% integration using trapizodial rull
   if(Z_in_intg>0) % Decession Maker Rx_in_data=1;
   else Rx_in_data=0;
   end

   % Quadrature coherent dector Z_qd=Rx_sig((i-
   1)*length(t)+1:i*length(t)).*sin(2*pi*f*t);
   %above line indicat multiplication ofreceived & Quadphase carred signal

   Z_qd_intg=(trapz(t,Z_qd))*(2/T);%integration using trapizodial rull if (Z_qd_intg>0)%
      Decession Maker
      Rx_qd_data=1; else Rx_qd_data=0;
     end

 Rx_data=[Rx_data Rx_in_data Rx_qd_data]; % Received Data vector end

figure(3) stem(Rx_data,'linewidth',3) title('Information
after Receiveing '); axis([ 0 11 0 1.5]), grid on;
```

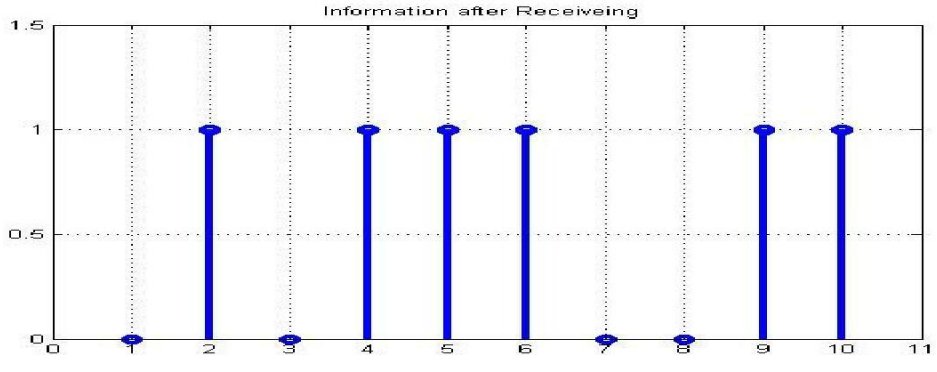
Output


Information before Transmiting


wave form for inphase component in QPSK modulation
amplitude(volt0
time(sec)
x 10^-6


wave form for Quadrature component in QPSK modulation
amplitude(volt0
time(sec)
x 10^-6


QPSK modulated signal (sum of inphase and Quadrature phase signal)
amplitude(volt0
time(sec)
x 10^-6


Information after Receiveing

RESULT

Thus the QPSK was simulated using MATLAB and output waveforms are obtained output verified.

EX.NO :	PERFORMANCE EVALUATION OF QAM MODULATION
DATE:	

AIM

To Simulate the QAM using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY

QAM is both an analog and a digital modulation scheme. It conveys two analog message signals, or two digital bit streams, by changing (modulating) the amplitudes of two carrier waves, using the amplitude-shift keying (ASK) digital modulation scheme or amplitude modulation (AM) analog modulation scheme. The two carrier waves, usually sinusoids, are out of phase with each other by 90 ° and are thus called quadrature carriers or quadrature components — hence the name of the scheme. The modulated waves are summed, and the resulting waveform is a combination of both phase-shift keying (PSK) and amplitude-shift keying (ASK), or (in the analog case) of phase modulation (PM) and amplitude modulation. In the digital QAM case, a finite number of at least two phases and at least two amplitudes are used. PSK modulators are often designed using the QAM principle, but are not considered as QAM since the amplitude of the modulated carrier signal is constant.

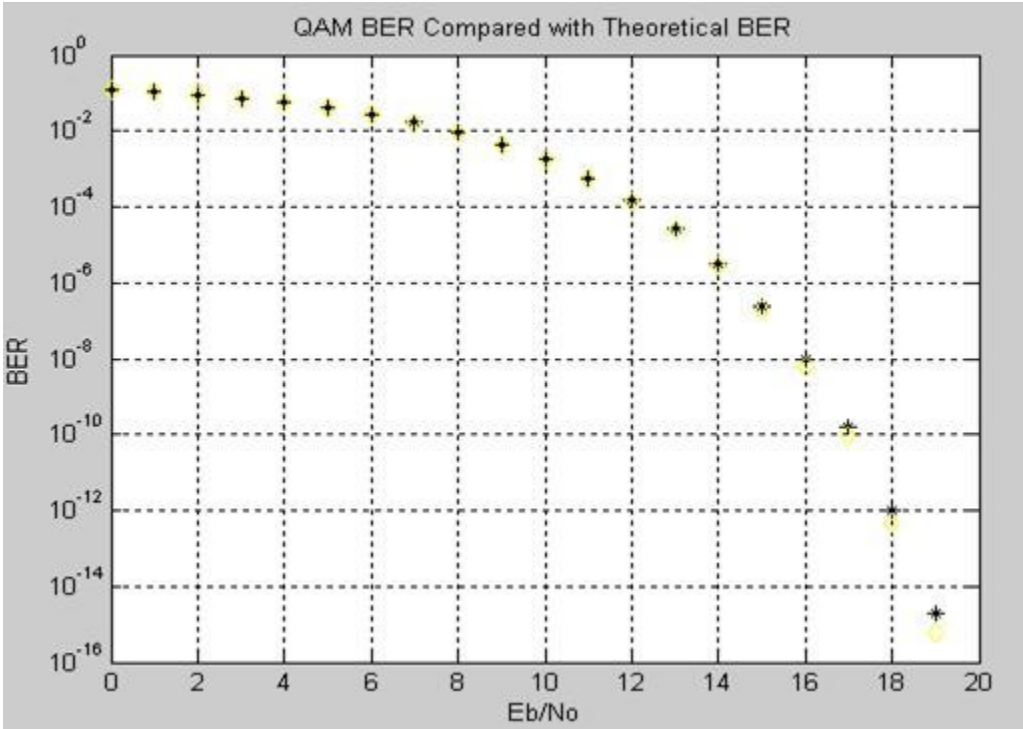
Coding:

```
M=input('Enter the value of M : '); Nsamp=input('Enter the value of Nsamp :
'); msg=[0:M-1 0]
modsig=qammod(msg,M);
txsig=rectpulse(modsig,Nsamp);
rxsig=txsig*exp(j*pi/180)
num=ones(Nsamp,1)/Nsamp;den=1;EbNo=[0:20]
ber=semianalytic(txsig,rxsig,'qam',M,Nsamp,num,den,EbNo);
bertheory=berawgn(EbNo,'qam',M);
figure;semilogy(EbNo,ber,'k*'); hold on;
semilogy(EbNo,bertheory,'yd'); xlabel('Eb/No');
ylabel('BER');
grid;
title('QAM BER Compared with Theoretical BER'); hold off;
```

OUTPUT OF QAM:

Enter the value of M: 16

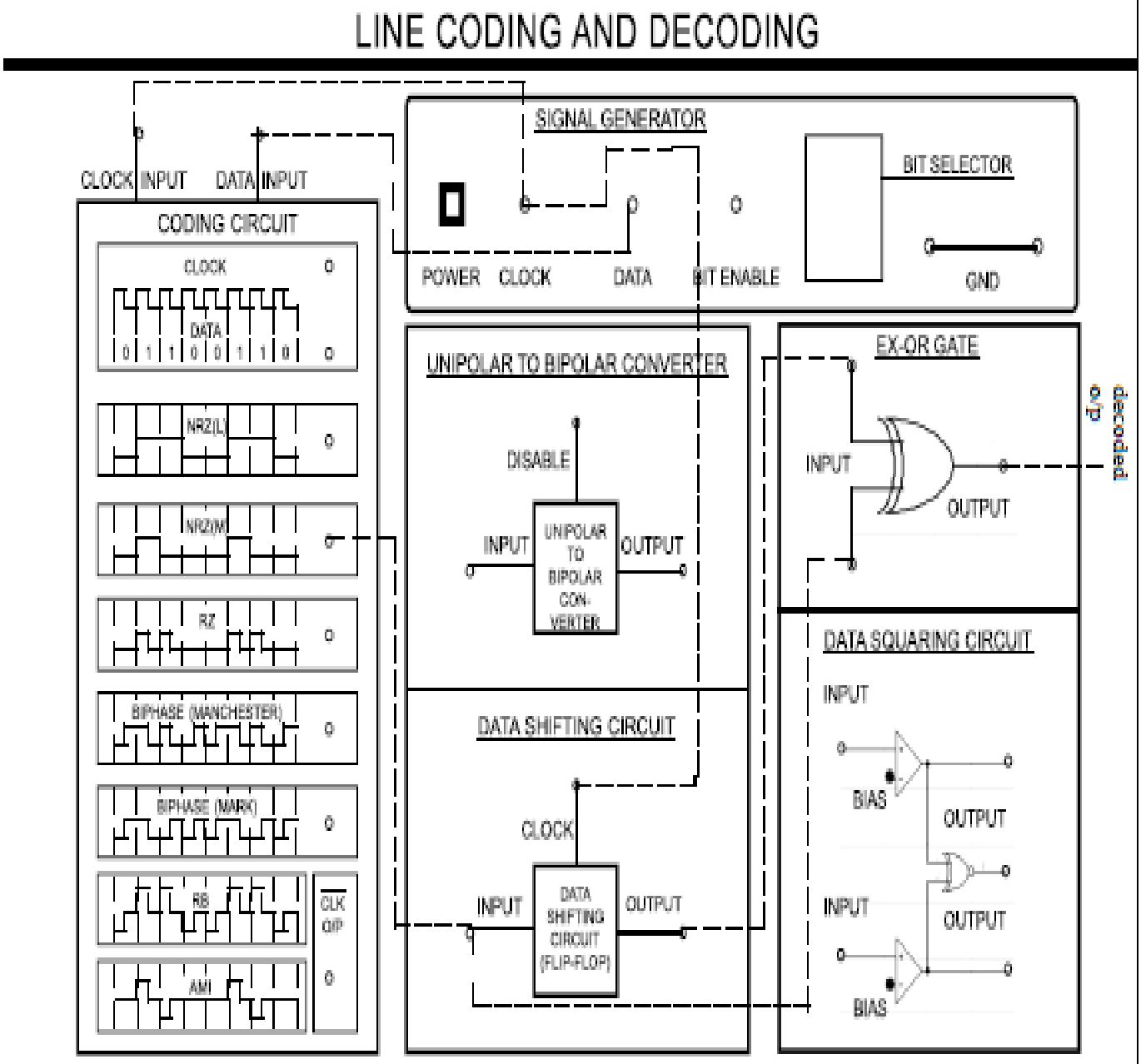
Enter the value of Nsamp : 16



RESULT

Thus the performance evaluation of QAM was simulated using MATLAB and output waveforms are obtained output verified.

NRZ(M) CONNECTION DIAGRAM:



EX.NO :	LINE CODING TECHNIQUES
DATE :	

AIM

To study and verify the operation of NRZ (M) coding and decoding.

APPARATUS REQUIRED

Line coding and decoding trainer kit, CRO and connecting probes

THEORY

In digital systems, the electrical waveforms are coded representations of the original information. If the original information is an analog signal, this must be converted to a series of discrete values that can be transmitted digitally. The process of converting the original information into a data sequence is referred to as source coding.

The line coding is the process of converting source coded signals into standard digital codes for the purpose of transmission over the channel. There are many possible ways of assigning the waveforms into the digital data. Simplest form of coding is ON-OFF, where a '1' is transmitted by a pulse and a '0' is transmitted by no pulse.

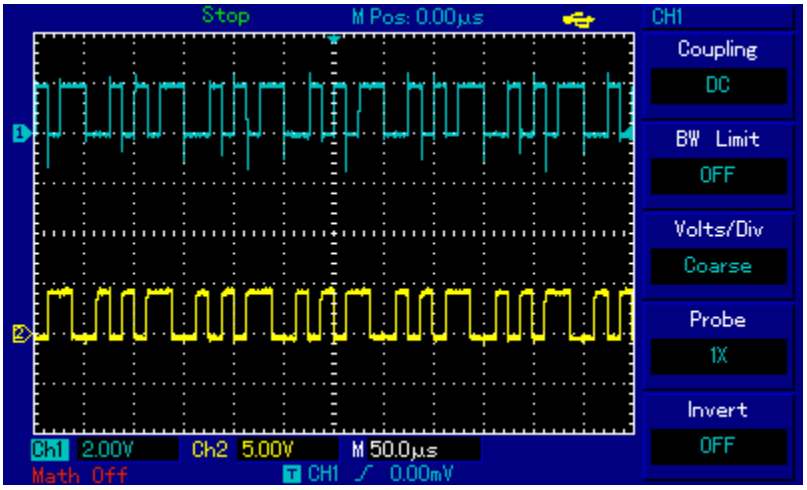
Generally the line coding is used in transmitter section while decoding in receiver section. The line decoding is the process of converting standard digital codes into source coded waveforms. Various line coding formats are

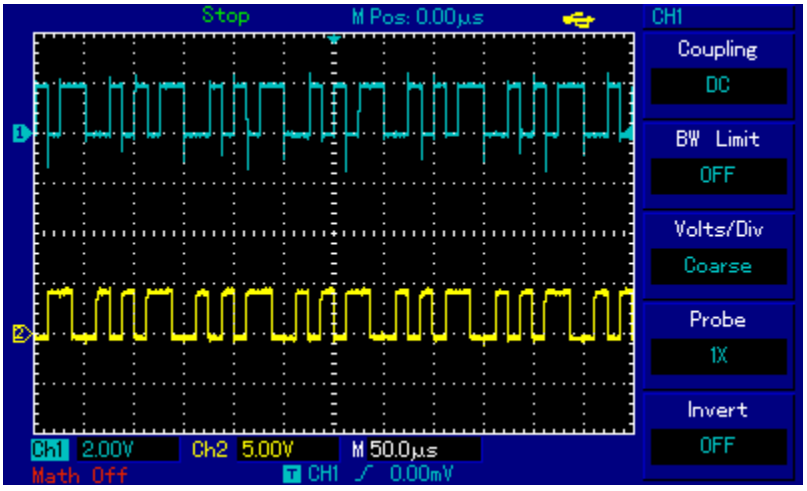
1. Unipolar RZ
2. Polar RZ
3. Polar NRZ
4. Bipolar NRZ
5. Bipolar RZ
6. Manchester coding

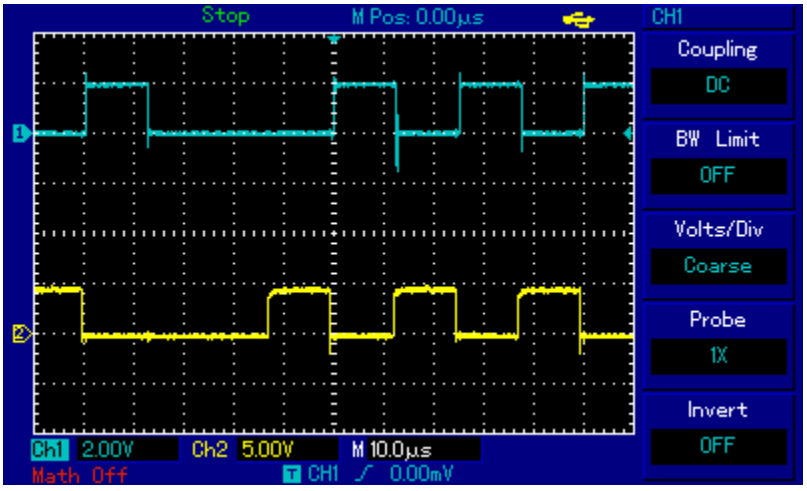
PROCEDURE:

1. Turn on the power to the trainer kit. Check the clock signal from the signal generator section.
2. Switch OFF the trainer and patch the circuit as per the circuit diagram.
3. Switch ON the trainer, to set the bit pattern generator, press the thumb wheel switch and set any decimal value.That corresponding BCD value of digital data will appear on the data output terminal when the bit enable switch is energized. Check this BCD output and clock by using CRO.
4. Patch the data output to the input of coding circuit.
5. Patch the clock pulse output to the input of coding circuit.

OUTPUT:


Stop
M Pos: 0.00µs
CH1
Coupling
DC
BW Limit
OFF
Volts/Div
Coarse
Probe
1X
Invert
OFF
Ch1 2.00V
Ch2 5.00V
M 50.0µs
Math Off
CH1 0.00mV


Stop
M Pos: 0.00µs
CH1
Coupling
DC
BW Limit
OFF
Volts/Div
Coarse
Probe
1X
Invert
OFF
Ch1 2.00V
Ch2 5.00V
M 50.0µs
Math Off
CH1 0.00mV


Stop
M Pos: 0.00µs
CH1
Coupling
DC
BW Limit
OFF
Volts/Div
Coarse
Probe
1X
Invert
OFF
Ch1 2.00V
Ch2 5.00V
M 10.0µs
Math Off
CH1 0.00mV

6. Connect the CRO CH1 at data input and CH2 at NRZ(M).
7. Observe the two chennal output is same as we studied theory about NRZ(M).
8. Connect the NRZ(M) data to the input of data shifting circuit connect CH2 to the output of shifting circuit and verify the shifted output.
9. Then connect NRZ (M) data and shifted data to the input of EX-OR Gate. This EXOR gives the decoded output.
10. Connect the CRO CH1 at the input data and CH2 at the output of EX-OR gate and verify the input and output is same by changing the bit pattern.

RESULT

Thus the Line coding schemes are obtained and the output was verified.

EX.NO :	SIMULATION OF ASK, PSK & FSK
DATE :	

AIM

To Simulate the ASK,PSK.FSK using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY

PSK is a digital modulation scheme which is analogues to phase modulation. In binary phase shift keying two output phases are possible for a single carrier frequency one out of phase represent logic 1 and logic 0. As the input digital binary signal change state the phase of output carrier shift two angles that are 180^{o} out of phase.

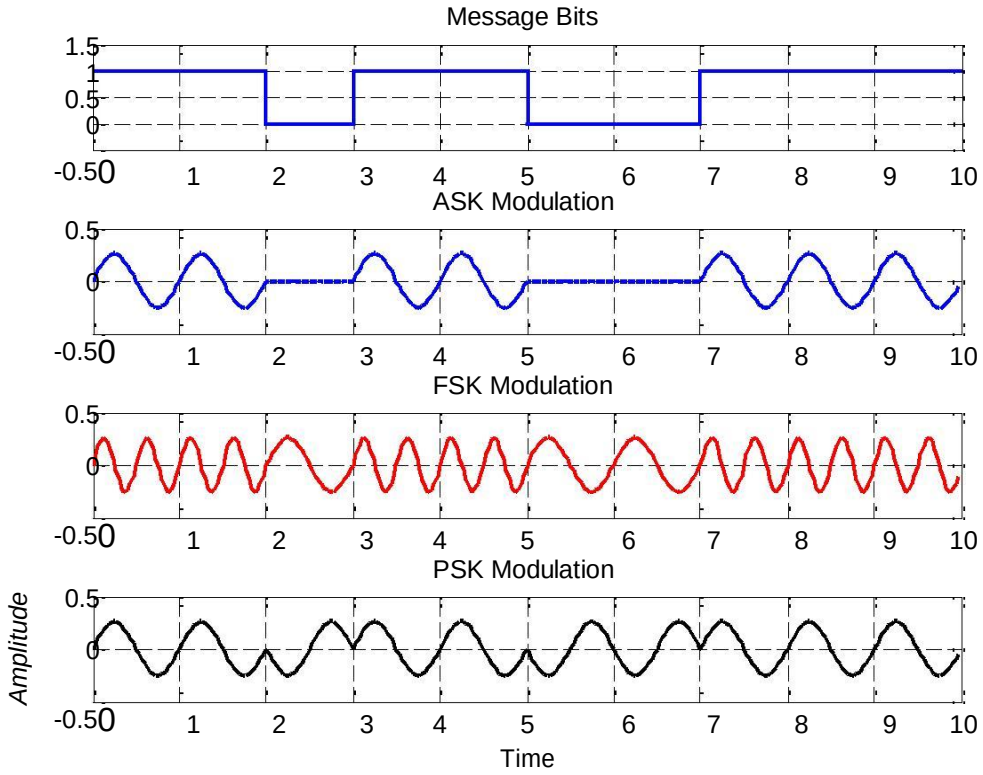
ASK or ON-OFF key is the simplest digital modulation technique. In this method there is only one unit energy carrier it is switched ON/OFF depending upon the input binary sequence to transmit symbol 0 & 1. No pulse is transmitted output contains some complete no of cycle of carrier frequency.

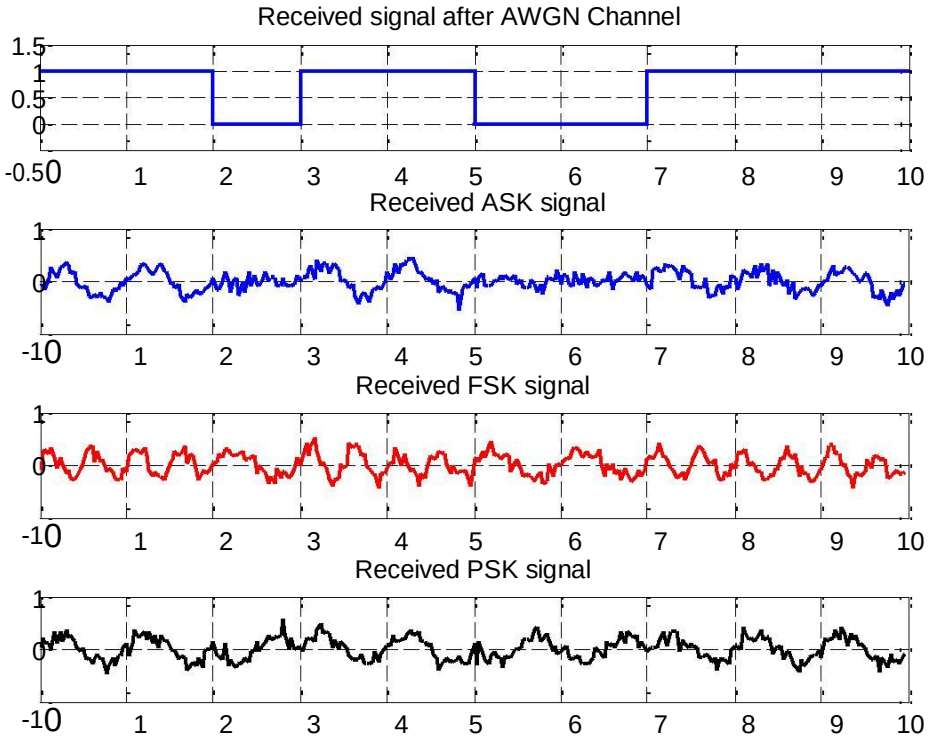
In digital data communication, binary code is transmitted by shifting a carrier frequency between two preset frequencies. This type of transmission is called frequency shift keying technique.

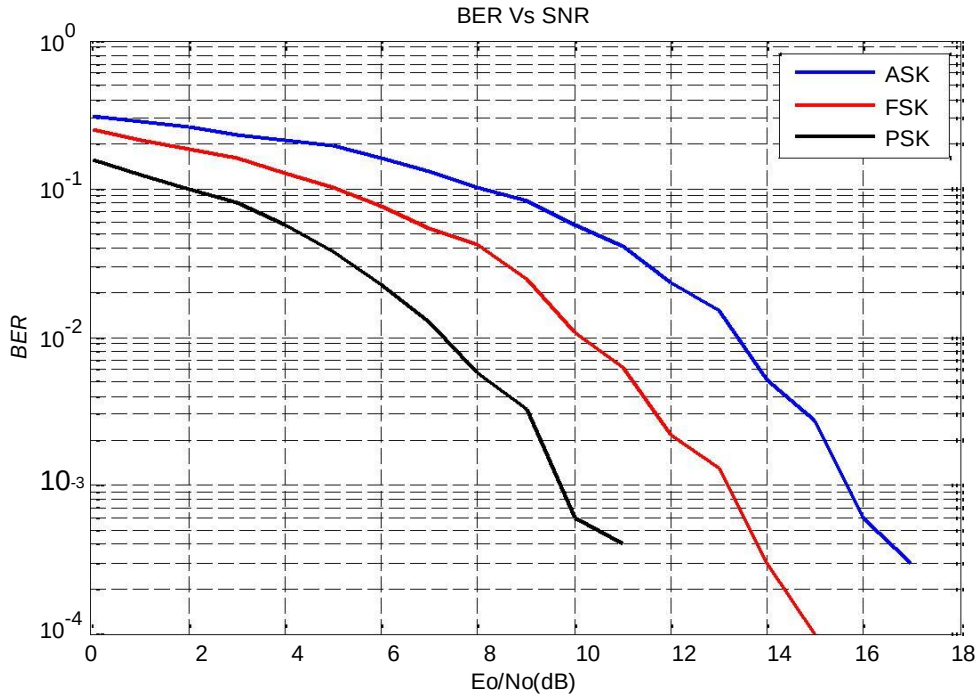
Coding:

```
clc; clear all; close all;
n=10000;
b=randint(1,n);
f1=1;f2=2; t=0:1/30:1-1/30; %ASK sa1=sin(2*pi*f1*t);
E1=sum(sa1.^2);
sa1=sa1/sqrt(E1); %unit energy sa0=0*sin(2*pi*f1*t);
%FSK
sf0=sin(2*pi*f1*t);
E=sum(sf0.^2);
sf0=sf0/sqrt(E);
sf1=sin(2*pi*f2*t);
E=sum(sf1.^2);
sf1=sf1/sqrt(E);
%PSK sp0=-sin(2*pi*f1*t)/sqrt(E1); sp1=sin(2*pi*f1*t)/sqrt(E1);

%MODULATION
ask=[];psk=[];fsk=[]; for i=1:n
   if b(i)==1 ask=[ask sa1]; psk=[psk sp1]; fsk=[fsk sf1];
   else
      ask=[ask sa0]; psk=[psk sp0]; fsk=[fsk sf0];
   end end figure(1)
subplot(411)
stairs(0:10,[b(1:10) b(10)],'linewidth',1.5) axis([0 10 -0.5 1.5])
title('Message Bits');grid on subplot(412) tb=0:1/30:10-1/30;
plot(tb, ask(1:10*30),'b','linewidth',1.5) title('ASK Modulation');grid on

subplot(413)
plot(tb, fsk(1:10*30),'r','linewidth',1.5) title('FSK
Modulation');grid on subplot(414)
plot(tb, psk(1:10*30),'k','linewidth',1.5) title('PSK
Modulation');grid on xlabel('Time');ylabel('Amplitude')
%AWGN
for snr=0:20 askn=awgn(ask,snr);
   pskn=awgn(psk,snr); fskn=awgn(fsk,snr);

   %DETECTION
   A=[];F=[];P=[]; for i=1:n
      %ASK Detection
      if sum(sa1.*askn(1+30*(i-1):30*i))>0.5 A=[A 1];
      else
         A=[A 0]; end
      %FSK Detection
```

```
        if sum(sf1.*fskn(1+30*(i-1):30*i))>sum(sf0.*fskn(1+30*(i-1):30*i)) F=[F 1];
        else
            F=[F 0]; end
        %PSK Detection
        if sum(sp1.*pskn(1+30*(i-1):30*i))>0 P=[P 1];
        else
            P=[P 0]; end
    end

    %BER
    errA=0;errF=0; errP=0; for i=1:n
        if A(i)==b(i) errA=errA;
        else errA=errA+1;
        end
        if F(i)==b(i) errF=errF;
else
            errF=errF+1; end
        if P(i)==b(i) errP=errP;
        else errP=errP+1;
        end end
    BER_A(snr+1)=errA/n; BER_F(snr+1)=errF/n; BER_P(snr+1)=errP/n;
end

figure(2)
subplot(411)
stairs(0:10,[b(1:10) b(10)],'linewidth',1.5) axis([0 10 -0.5 1.5]);grid on
title('Received signal after AWGN Channel') subplot(412)
tb=0:1/30:10-1/30;
plot(tb, askn(1:10*30),'b','linewidth',1.5) title('Received ASK signal');grid on subplot(413)
plot(tb, fskn(1:10*30),'r','linewidth',1.5) title('Received FSK signal');grid on
subplot(414)
plot(tb, pskn(1:10*30),'k','linewidth',1.5) title('Received PSK signal');grid on figure(3)
semilogy(0:20,BER_A, 'b','linewidth',2) title('BER Vs SNR')
grid on; hold on
semilogy(0:20,BER_F,'r','linewidth',2) semilogy(0:20,BER_P, 'k','linewidth',2)
xlabel('Eo/No(dB)')
ylabel('BER') hold off
legend('ASK','FSK','PSK');
```


Message Bits
ASK Modulation
FSK Modulation
PSK Modulation
Amplitude
Time


Received signal after AWGN Channel
Received ASK signal
Received FSK signal
Received PSK signal



RESULT

Thus the ASK,FSK& PSK was simulated using MATLAB and output waveforms are obtained output verified.

EX.NO :	IMPLEMENTATION OF LINEAR BLOCK CODES
DATE :	

AIM

To Simulate the Linear Block Codes using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY:

In coding theory, a linear code is an error-correcting code for which any linear combination of codewords is also a codeword. Linear codes are traditionally partitioned into block codes and convolutional codes, although Turbo codes can be seen as a hybrid of these two types. Linear codes allow for more efficient encoding and decoding algorithms than other codes

Linear codes are used in forward error correction and are applied in methods for transmitting symbols (e.g., bits) on a communications channel so that, if errors occur in the communication, some errors can be corrected or detected by the recipient of a message block. The codewords in a linear block code are blocks of symbols which are encoded using more symbols than the original value to be sent. A linear code of length n transmits blocks containing n symbols. For example, the [7,4,3] Hamming code is a linear binary code which represents 4-bit messages using 7-bit codewords. Two distinct codewords differ in at least three bits. As a consequence, up to two errors per codeword can be detected and a single error can be corrected. This code contains 2^4=16 codewords.

Coding

```
clc;
clear all;
%input generator matrix g=input('Enter the generator matrix:')
disp('g=')
disp('The order of linear block code for given generator matrix is :') [n,k]=size(transpose(g))
for i=1:2^k for j=k:-1:1
    if rem(i-1,2^(-j+k+1))>=2^(-j+k) u(i,j)=1;
    else u(i,j)=0;
    end end
end u;
disp('The possible code words are :') c=rem(u*g,2)
disp('The minimum hamming distance dm in for given block code is :')
d_min=min(sum((c(2:2^k,:))'))
%Cord word
r=input('Enter the received code word:') p=[g(:,n-k+2:n)];
h=[transpose(p),eye(n-k)]; disp('Hamming Code')
ht=transpose(h)
disp('Syndrome of a given code word is:') s=rem(r*ht,2)
for i=1:1:size(ht) if (ht(i,1:3)==s)
    r(i)=1-r(i); break;
  end end
disp('The error is in bit :') i
disp('The corrected code word is :') r
```

OUTPUT OF LINEAR BLOCK CODE :

Enter the generator matrix:[1 0 0 0 1 0 1;0 1 0 0 1 1 1;0 0 1 0 1 1 0;0 0 0 1 0 1 1]

g =

1	0	0	0	1	0	1
0	1	0	0	1	1	1
0	0	1	0	1	1	0
0	0	0	1	0	1	1

g=

The order of linear block code for given generator matrix is :

n = 7

k = 4

The possible code words are :

c =

0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1

The minimum hamming distance dm in for given block code is : d_min = 3

Enter the recevied code word:[1 0 0 0 1 0 0]

r = 1 0 0 0 1 0 0

Hamming Code

ht =

1	0	1
1	1	1
1	1	0
0	1	1
1	0	0
0	1	0
0	0	1

Syndrome of a given code word is:

s =0 0 1

The error is in bit :

i = 7

The corrected code word is :

r = 1 0 0 0 1 0 1

RESULT

Thus the Linear Block Codes was Implemented & simulated by using MATLAB and output waveforms are obtained output verified.

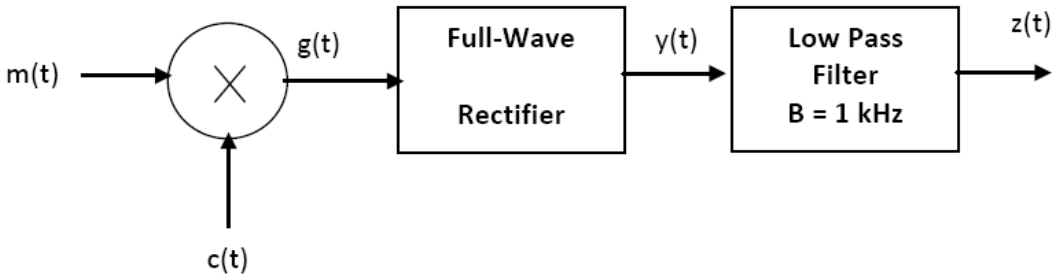


Fig.11.1 COMMUNICATION LINK BLOCK DIAGRAM

EX.NO :	COMMUNICATION LINK SIMULATION
DATE :	

Aim :

The main AIM is to learn the basic tools and concepts for simulating communication systems using MATLAB.

Software Used:

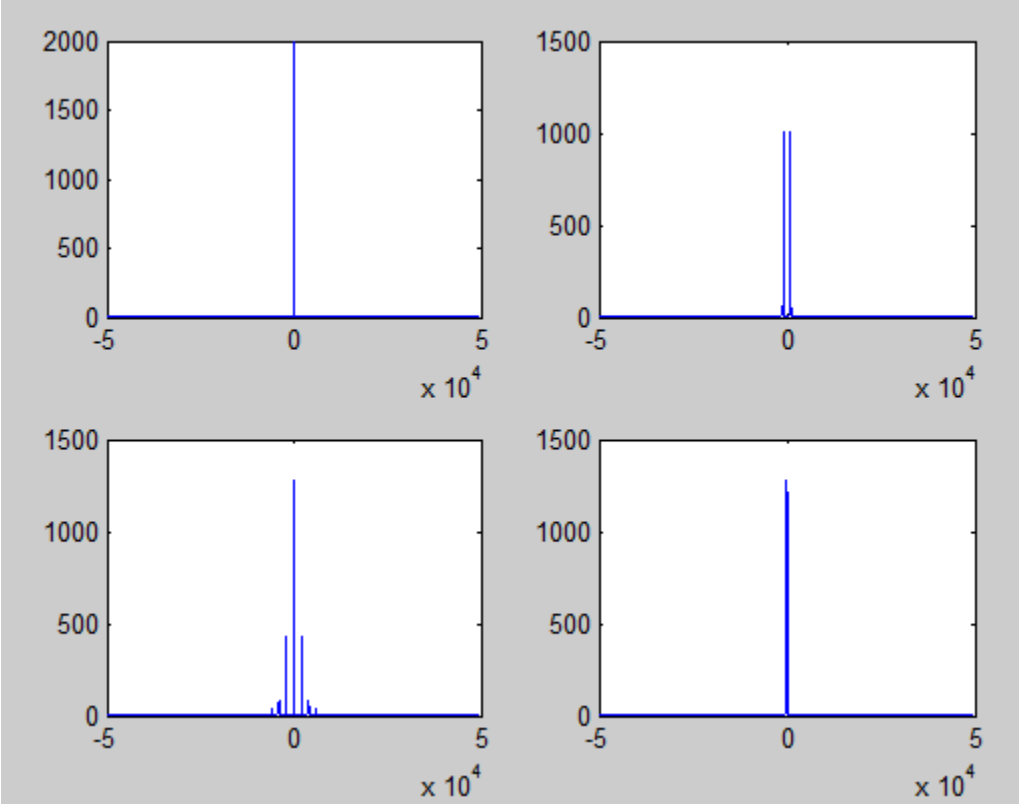
MATLAB 7.0

Program :

```
% Define the time interval
ts=0.00001; t= -0.1:ts:0.1;
% Define the functions m(t) and c(t)
m=exp(-100*abs(t)); c=cos(2*pi*1000*t);
% Performe the multiplication
g=m.*c;
% Perform full-wave rectification
y=abs(g);
% Create the filter
cutoff=1000; [a b]=butter(5,2*cutoff*ts);
% Get the output after the filter;
z=filter(a,b,y);
% Plot the input and output on the same graph
figure (1)
plot(t,m,t,z);
legend('Input Signal','Output Signal')
xlabel ('time') ;ylabel('amplitude')
title ('Case Study')
% Finding the FT of the signals
M=abs(fftshift(fft(m))); G=abs(fftshift(fft(g))); Y=abs(fftshift(fft(y)));
```

```
Z=abs(fftshift(fft(z)));
% Creating the vector for the frequency axis
f=[-length(t)/2:length(t)/2-1]/(length(t)*ts);
% Plotting all FT on one sheet, in a 2x2 matrix format
figure (2)
subplot (221)
plot(f,M)
subplot(222)
plot(f,G)
subplot (223)
plot(f,Y)
subplot(224)
plot(f,Z)
```

OUTPUT:



Result:

Thus the communication link was simulated and the outputs were generated.

EX.NO :	ZERO FORCING EQUALIZER
DATE :	

Aim:

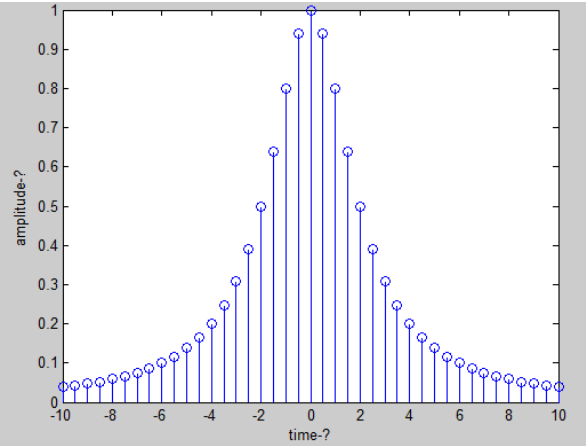
To design the least mean square equalizer using MATLAB.

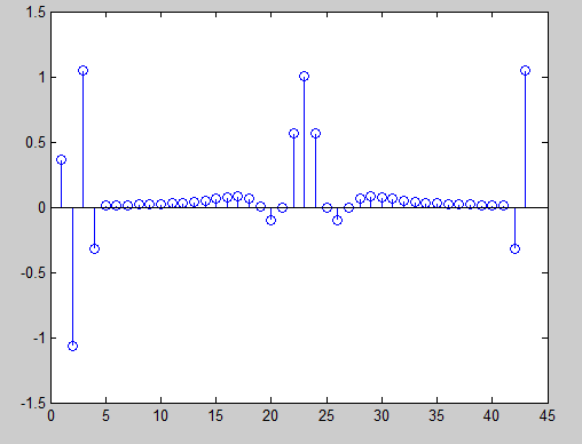
Apparatus Required:

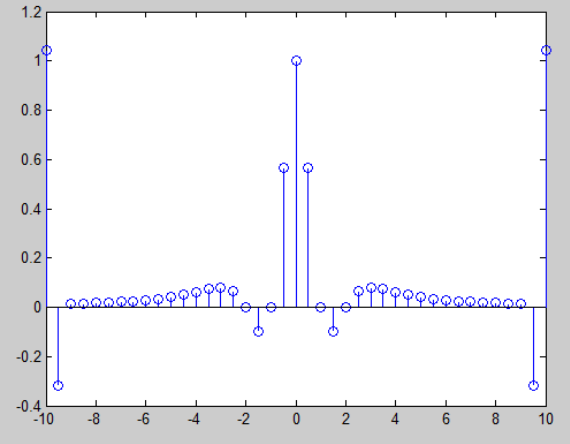
Matlab Version 6.5 and above.

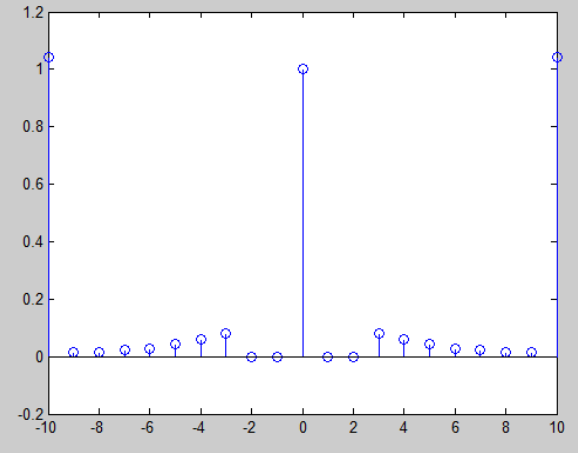
Theory:

Zero Forcing Equalizer refers to a form of linear equalization algorithm used in communication systems which applies the inverse of the frequency response of the channel. The Zero-Forcing Equalizer applies the inverse of the channel frequency response to the received signal, to restore the signal after the channel. It has many useful applications. For example, (MIMO) where knowing the channel allows recovery of the two or more streams which will be received on top of each other on each antenna. The name Zero Forcing corresponds to bringing down the intersymbol interference (ISI) to zero in a noise free case. This will be useful when ISI is significant compared to noise.


amplitude-?
time-?







CODING:

```
clear all;close all;echo on
T=1;N0=1;Fs=2/T;Ts=1/Fs;
t=-10*T:T/2:10*T;
x=1./(1+((0.5/T)*t).^2);
stem(t,x);
title('equalizer output');
xlabel('time-→');
ylabel('amplitude-→');
X=[];
for m=-2:1:2
   for n=-2:1:2
      X(m+3,n+3)=1./(1+((0.5/T)*(m*T-n*T/2)).^2);
   end
end
%D=D+N0/2*eye(5)
K=inv(X)
c_opt=K*[0 0 1 0 0]'
equalized_x=filter(c_opt,1,[x 0 0])
figure
stem(equalized_x);equalized_x=equalized_x(3:length(equalized_x));
figure
stem(t,equalized_x);
for i=1:2:length(equalized_x);
downsampled_equalizer_output((i+1)/2)=equalized_x(i);
end;
figure
s=size(downsampled_equalizer_output)
t=linspace(-10*T,10*T,s(2))
stem(t,downsampled_equalizer_output)
```

RESULT:

Thus the Zero Forcing equalizer was designed and simulated using MATLAB, various performances were obtained.

EX.NO :	EQUALIZATION-LMS ALGORITHM
DATE :	

AIM

To Simulate the LMS Equalization Algorithm using MATLAB coding.

APPARATUS REQUIRED

MATLAB V13

THEORY

The realization of the causal Wiener filter looks a lot like the solution to the least squares estimate, except in the signal processing domain. The least squares solution, for input matrix $\mathbf{x}$ and output vector $\mathbf{y}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X^T X})^{-1}\mathbf{X^T}\boldsymbol{y}.$$

The FIR least mean squares filter is related to the Wiener filter, but minimizing the error criterion of the former does not rely on cross-correlations or auto-correlations. Its solution converges to the Wiener filter solution. Most linear adaptive filtering problems can be formulated using the block diagram above. That is, an unknown system $\mathbf{h}(n)$ is to be identified

and the adaptive filter attempts to adapt the filter $\hat{\mathbf{h}}(n)$ to make it as close as possible to $\mathbf{h}(n)$, while using only observable signals $x(n)$, $d(n)$ and $e(n)$; but $y(n)$, $v(n)$ and $h(n)$ are not directly observable. Its solution is closely related to the Wiener filter.

Definition of symbols[

n is the number of the current input sample

p is the number of filter taps

$\{\cdot\}^H$ (Hermitian transpose or conjugate transpose)

$\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-p+1)]^T$

$\mathbf{h}(n) = [h_0(n), h_1(n), \ldots, h_{p-1}(n)]^T, \quad \mathbf{h}(n) \in \mathbb{C}^p$

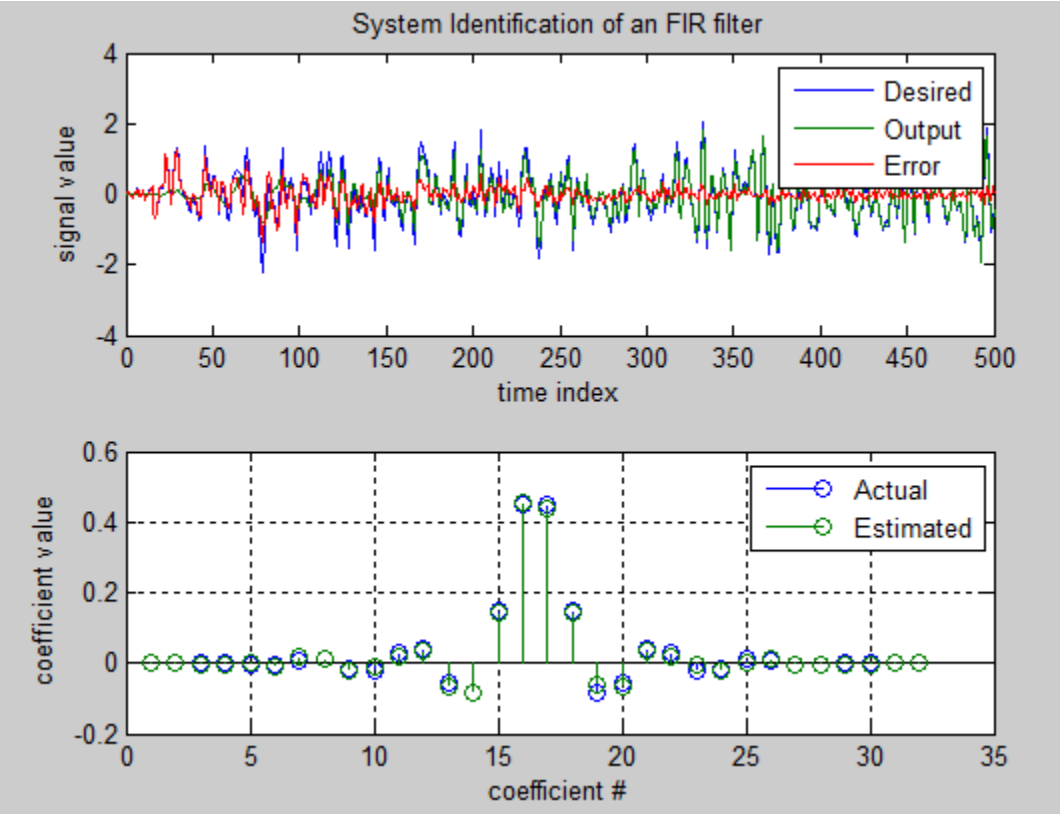
$y(n) = \mathbf{h}^H(n) \cdot \mathbf{x}(n)$

$d(n) = y(n) + \nu(n)$

$\hat{\mathbf{h}}(n)$ estimated filter; interpret as the estimation of the filter coefficients after n samples

$e(n) = d(n) - \hat{y}(n) = d(n) - \hat{\mathbf{h}}^H(n) \cdot \mathbf{x}(n)$

OUTPUT:


System Identification of an FIR filter
signal value
time index
Desired
Output
Error
coefficient value
coefficient #
Actual
Estimated

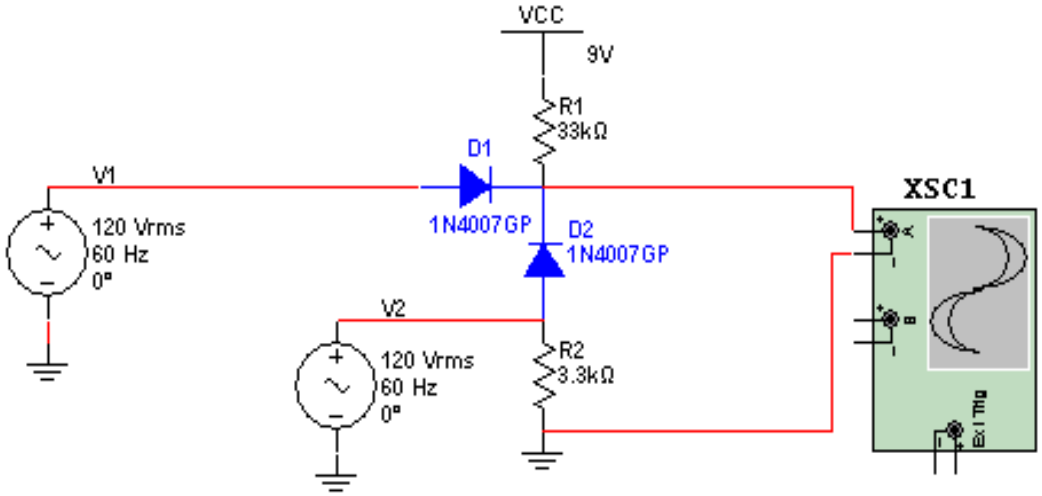
Coding

```
clear all;
%       s1
      [s1 s2]= RandStream.create('mrg32k3a','NumStreams',2);
      x  = randn(s1,1,500);     % Input to the filter
       b  = fir1(31,0.5);     % FIR system to be identified
       n  = 0.1*randn(s2,1,500); % Observation noise signal
       d  = filter(b,1,x)+n;  % Desired signal
       mu = 0.008;            % LMS step size
       h = adaptfilt.lms(32,mu);
       [y,e] = filter(h,x,d);
       subplot(2,1,1); plot(1:500,[d;y;e]);
       title('System Identification of an FIR filter');
       legend('Desired','Output','Error');
       xlabel('time index'); ylabel('signal value');
       subplot(2,1,2); stem([b.',h.Coefficients.']);
       legend('Actual','Estimated');
       xlabel('coefficient #'); ylabel('coefficient value'); grid on;
```

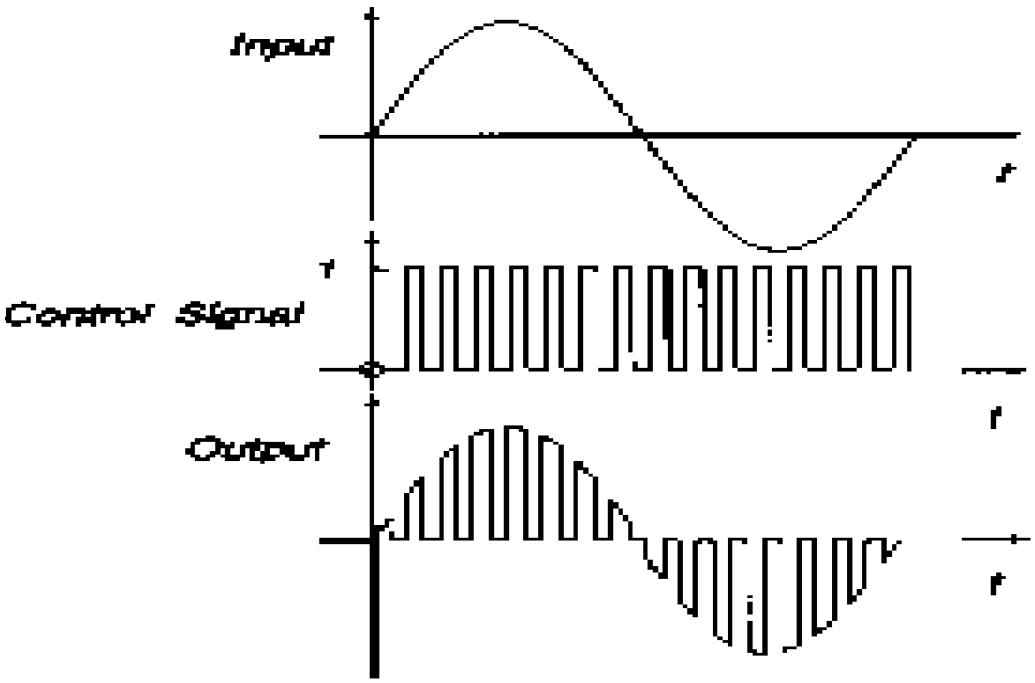
RESULT

Thus the LMS Equalization Algorithm was simulated using MATLAB and output waveforms are obtained output verified.

CIRCUIT DIAGRM:



MODELGRAPH



EX.NO :	PULSE AMPLITUDE MODULATION
DATE :	

AIM:

To generate PAM to observe the output waveform.

APPARATUS REQUIRED:

Sl. No	Name of the Apparatus	Range	Quantity
1	Function generator		1
2	Diode	1N4007	2
3	Resistors,	3.3K ohms,	2
4	Bread Board,		1
5	Power supply		1
6	CRO	0-30MHz	1
7	Probe		2
8	Connecting Wires		10

PROCEDURE:

1. Measure the amplitude of each pulse, no. of pulses in one cycle, frequency of pulses.
2. Vary the modulating input signal and observe its effect on the output.
3. Observe and compare the output waveform with input waveform/
4. See the input and output that are same in frequency, amplitude and phase and Draw the waveform of the modulating signal, sampling signal, pulse amplitude a modulated signal.

TABULATION:

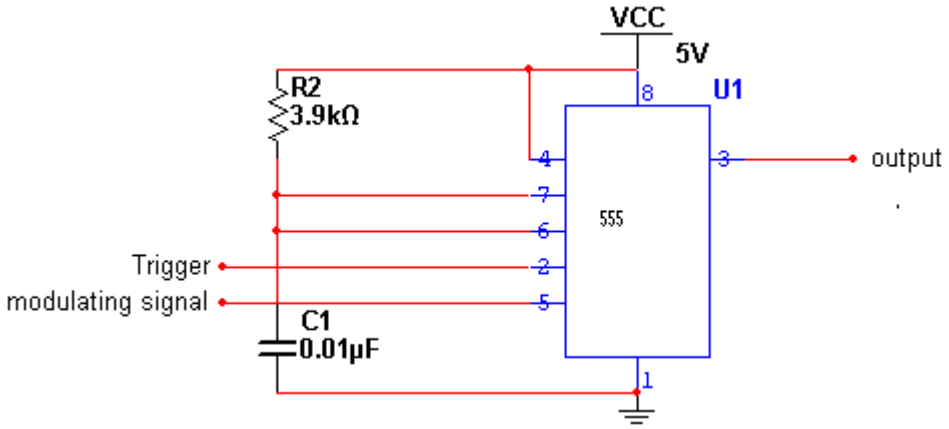
Before modulation			After modulation	
Frequency of sampling signal	Frequency of modulating signal	Amplitude of modulating signal	No. of pulses in each cycle	Amplitude of pulses

RESULT:

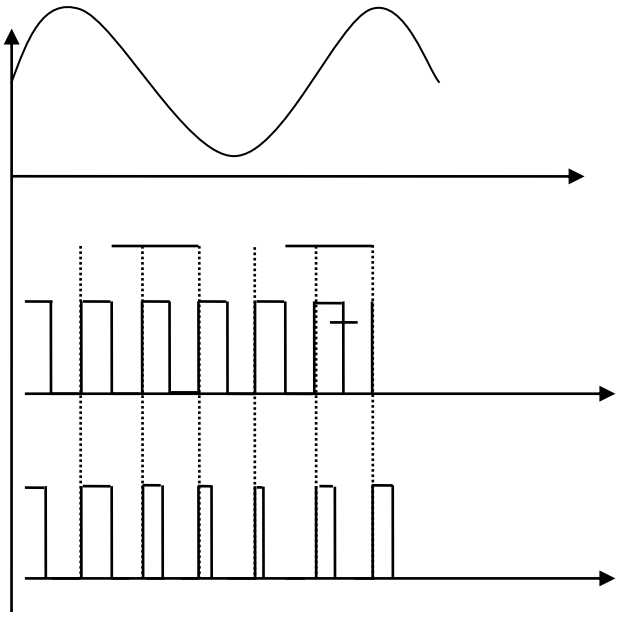
In this technique , the signal is sampled at regular intervals, and each sample is made proportional to the **amplitude** of the signal at the instant of sampling

In PAM generation the **amplitude of the carrier pulse varies.**

CIRCUIT DIAGRAM :



MODEL GRAPH



EX.NO :	PULSE WIDTH MODULATION
DATE :	

AIM:

To generate PWM to observe the modulated output.

APPARATUS REQUIRED

Sl. No	Name of the Apparatus	Range	Quantity
1	IC 555		1
2	Capacitors,	0.01 micro Farad.	1
3	Resistors,	3.9K ohms,	1
4	Bread Board,		1
5	Power supply		1
6	CRO	0-30MHz	1
7	Probe		2
8	Connecting Wires		10
9	Function generator		1

Procedure:

1. Circuit connections are given as per circuit diagram.
2. The input signal of frequency 1KHz and amplitude 1Vpp is fed to the circuit from function generator.
3. The carrier signal's amplitude and frequency are measured at OUTPUT 1
4. The PWM wave is observed on the CRO.
5. The width of each pulse and the number of pulses are measured.

Tabulation

Modulating signal	Amplitude	Frequency
Carrier signal	Amplitude	Frequency
PWM signal	No. of pulses per cycle	
	Maximum width of pulses	
	Minimum width of pulses	
	Amplitude of pulses	

Result:

Thus in PWM the width of the carrier signal changes in accordance with the instantaneous amplitude of the modulating signal.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3462 LINEAR INTEGRATED CIRCUITS LABORATORY

Semester - 04

LABORATORY MANUAL

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

DESIGN AND ANALYSIS OF THE FOLLOWING CIRCUITS

1. Series and Shunt feedback amplifiers-Frequency response, Input and output impedance
2. RC Phase shift oscillator and Wien Bridge Oscillator
3. Hartley Oscillator and Colpitts Oscillator
4. RC Integrator and Differentiator circuits using Op-Amp
5. Clippers and Clampers
6. Instrumentation amplifier
7. Active low-pass, High pass & Band pass filters
8. PLL Characteristics and its use as frequency multiplier, clock synchronization
9. R-2R ladder type D-A converter using Op-Amp

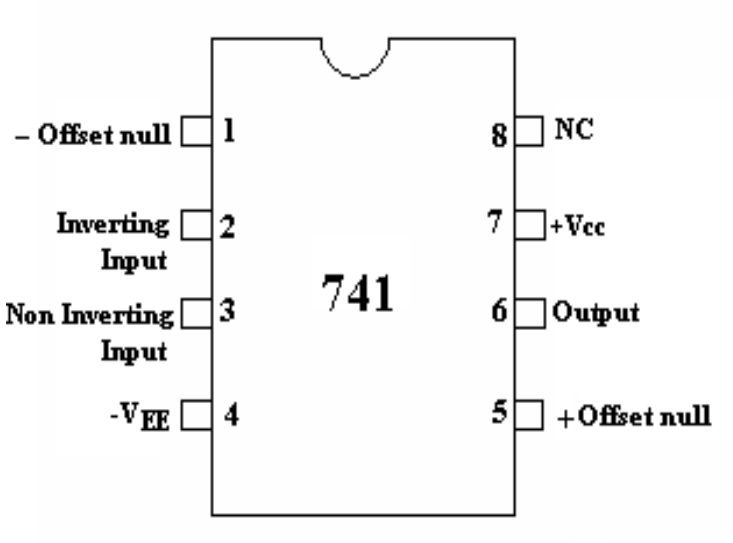
SIMULATION USING SPICE (Using Transistor):

1. Tuned Collector Oscillator
2. Twin -T Oscillator / Wein Bridge Oscillator
3. Double and Stagger tuned Amplifiers
4. Bistable Multivibrator
5. Schmitt Trigger circuit with Predictable hysteresis
6. Analysis of power amplifier

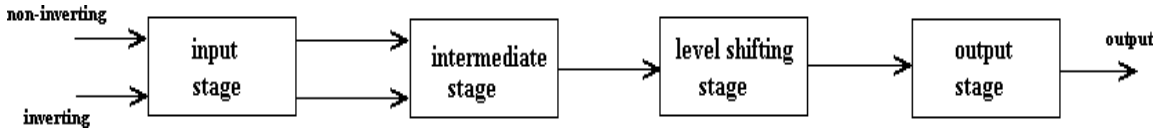
IC 741 - General Description:

The IC 741 is a high performance monolithic operational amplifier constructed using the planar epitaxial process. High common mode voltage range and absence of latch-up tendencies make the IC 741 ideal for use as voltage follower. The high gain and wide range of operating voltage provides superior performance in integrator, summing amplifier and general feed back applications.

Pin Configuration:



Block Diagram of Op-Amp:



Features:

1. No frequency compensation required.
2. Short circuit protection
3. Offset voltage null capability
4. Large common mode and differential voltage ranges
5. Low power consumption
6. No latch-up

SPECIFICATIONS:-

1. Voltage gain A = ∞ typically 2,00,000
2. Input resistance $R_L = \infty$ Ω, practically 2MΩ
3. Output resistance R =0, practically 75Ω
4. Bandwidth = ∞ Hz. It can be operated at any frequency
5. Common mode rejection ratio = ∞
   (Ability of op amp to reject noise voltage)
6. Slew rate + ∞ V/μsec
   (Rate of change of O/P voltage with respect to applied I/P)
7. When $V_1 = V_2$, V_D=0
8. Input offset voltage (Rs ≤ 10KΩ) max 6 mv
9. Input offset current = max 200nA
10. Input bias current : 500nA
11. Input capacitance : typical value 1.4pF
12. Offset voltage adjustment range : ± 15mV
13. Input voltage range : ± 13V
14. Supply voltage rejection ratio : 150 μV/V
15. Output voltage swing: + 13V and – 13V for R_L > 2KΩ
16. Output short-circuit current: 25mA
17. supply current: 28mA
18. Power consumption: 85mW
19. Transient response: rise time= 0.3 μs Overshoot= 5%

APPLICATIONS:-

1. AC and DC amplifiers.
2. Active filters.
3. Oscillators.
4. Comparators.
5. Regulators., etc.,

EXP.NO:	RC PHASE SHIFT AND WIEN BRIDGE OSCILLATOR USING OP-AMP

AIM:

To design RC Phase Shift and Wien Bridge Oscillator using Op-amp IC 741 and to test its performance.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741	- - -	01
2.	RESISTORS	1.5KΩ 3.3KΩ, 33KΩ,	EACH 03
		10 KΩ 22 KΩ, 1MΩ,	EACH 01
3.	CAPACITORS	0.1μf	03
4.	DIGITAL TRAINER KIT	- - -	01
5.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
6.	CONNECTING WIRES	- - -	FEW

DESIGN PROCEDURE:

Design a RC phase shift oscillator to oscillate at 200Hz.

1. Select f_o = 200Hz.

2. Assume C = 0.1μf & determine R from f_o.

$$f_o = \frac{1}{2\pi\sqrt{6}RC} = R = \frac{1}{2\pi\sqrt{6}\,f_{oc}} = 3.3K.$$

3. To prevent the loading of amp because it is necessary that R1>>10R.

   Therefore R1=10R=33K.

4. At this frequency the gain must be atleast 29 (i.e.) $R_f / R_1 = 29$.

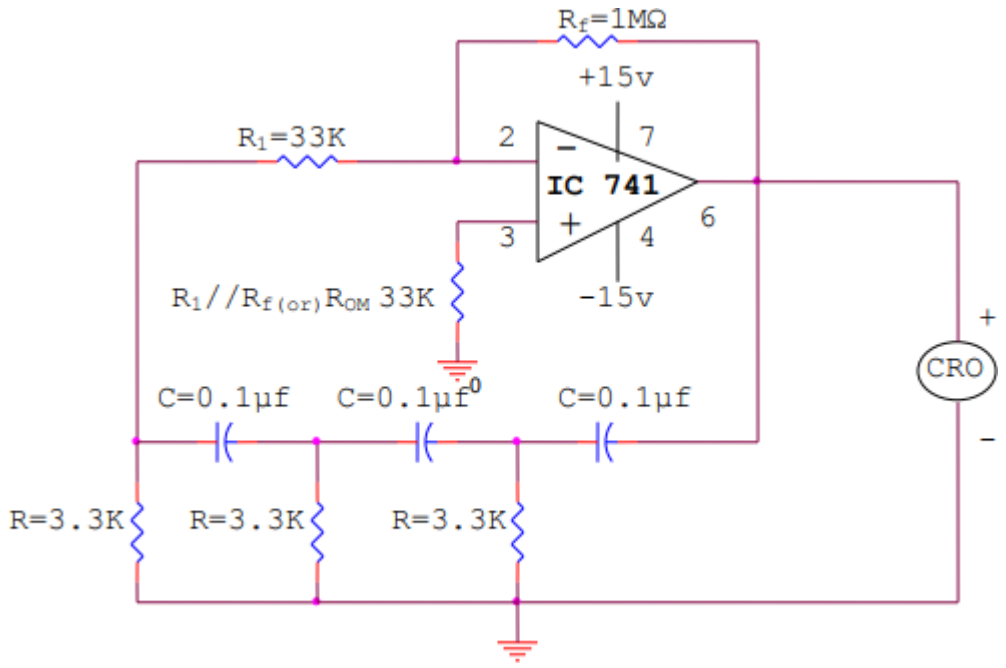
   Therefore $R_f = 29R_1$.

   $R_f = 29\ (33K) = 957K\Omega$.

   Therefore use $R_f = 1M\Omega$.

RC PHASE SHIFT OSCILLATOR:-

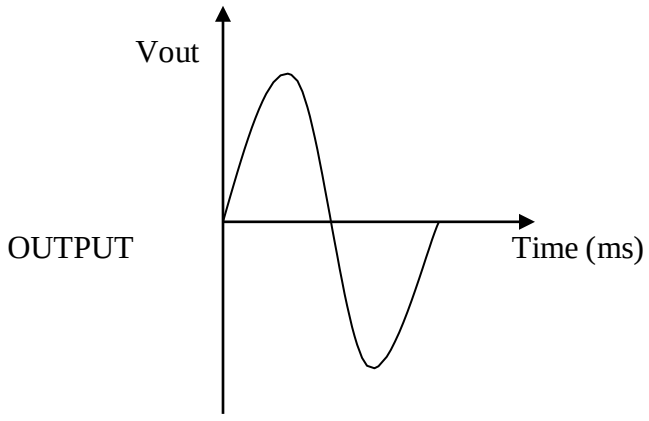
CIRCUIT DIAGRAM:-



TABULATION:.

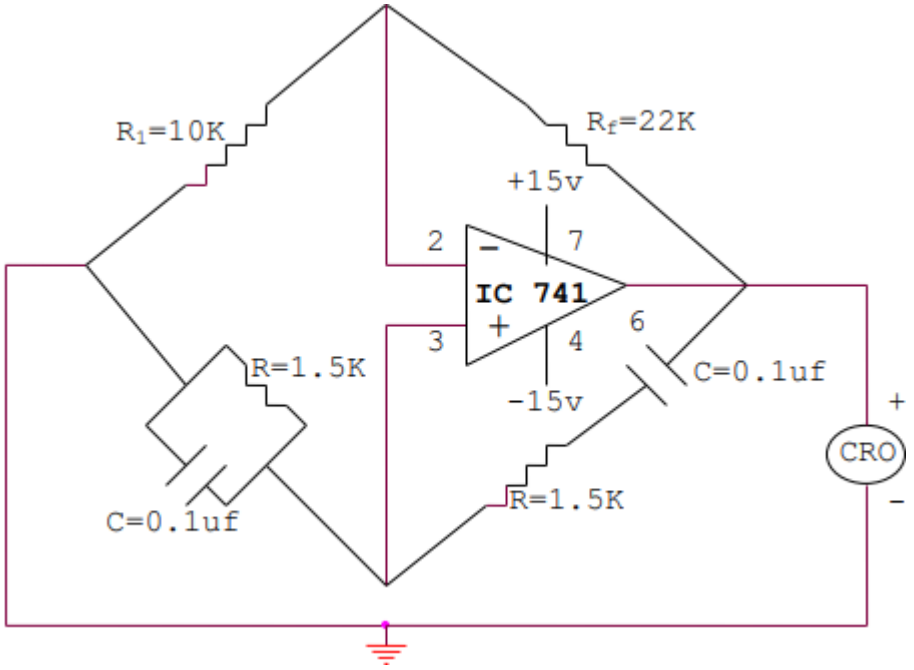
OBSERVED OUTPUT WAVEFORM			Design Frequency (Hz)
Amplitude (volts)	**Time period (ms)**	**Frequency (Hz)**	

MODEL GRAPH:



WIEN BRIDGE OSCILLATOR:-

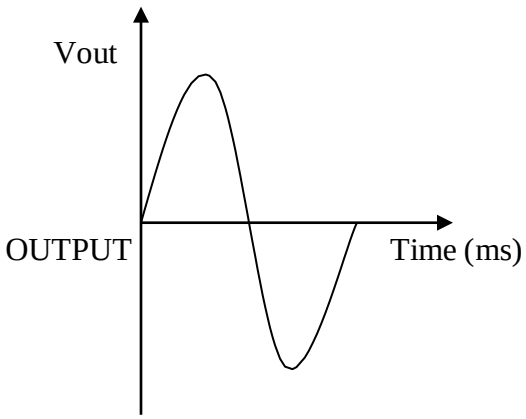
CIRCUIT DIAGRAM:-



TABULATION:

OBSERVED OUTPUT WAVEFORM			Design Frequency (Hz)
Amplitude (volts)	Time period (ms)	Frequency (Hz)	

MODEL GRAPH:



PROCEDURE- (RC PHASE SHIFT):-

1. Select the given frequency of oscillation f_0 = 200Hz.
2. Assume either R or C to find out the other using formula $f_0 = \frac{1}{2\pi\sqrt{6}RC}$.
3. The gain is selected such that R_f / R_1 = 29K. Assume R_f or R_1 to find the other.
4. Connect the circuit as per as the circuit diagram.
5. Measure the amplitude frequency of the output signal plot the graph.

DESIGN PROCEDURE:

(i) Select frequency f_0 = 1 KHz.

(ii) Use $f_0 = \frac{1}{2\pi RC}$, A = 1+ (R_f / R_1) = 3. To find R & R_f.

(iii) Therefore $R_f = 2R_1$ & assume C = 0.1μf & find R from

$$R = \frac{1}{2\pi f_o C} = 1.59K\Omega.$$

(iv) Assume R_1 = 10K & find R_f from $R_f = 2R_1$

Therefore R_f = 20K ≅ 22KΩ

PROCEDURE:

1. Select the given frequency of oscillation f_0 = 1 KHz.
2. Assume either R or C to find out the other using formula $\frac{1}{2\pi RC}$. Also determine the value of other components as given in design procedure.
3. Connect the circuit as per as the circuit diagram.
4. Measure the amplitude and frequency of the output signal to plot the graph.

RESULT:

Thus RC Phase Shift and Wien Bridge Oscillator were designed and tested usingop-amp IC 741.

EXP.NO:	HARTLEY OSCILLATOR AND COLPITTS OSCILLATOR USING OP-AMP

AIM:

To design Hartley Oscillator and Colpitts Oscillator using Op-amp IC 741 and to test its performance.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741	- - -	01
2.	RESISTORS	1KΩ 10KΩ, 33KΩ,	EACH 03
		10 KΩ 22 KΩ, 1MΩ,	EACH 01
3.	CAPACITORS	100 pF, 400pF	03
4.	POWERSUPPLY	0-30V	01
5.	BREAD BOARD	- - -	01
6.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
7	CONNECTING WIRES	- - -	FEW

PROCEDURE- (HARTLEY):-

1. Assume either C or L to find out the other using formula $f_0 = \frac{1}{2\pi\sqrt{L_T C}}$.
2. The gain is selected such that $R_1 / R_2 = 10$. Assume R_2 or R_1 to find the other.
3. Connect the circuit as per as the circuit diagram.
4. Find the frequency of oscillation f_0 using above value and practical verification.
5. Measure the amplitude frequency of the output signal plot the graph.

DESIGN PROCEDURE: (HARTLEY):-

Gain of the Amplifier $G = \frac{R_2}{R} = \frac{100 \times 10^3}{R}$

$$R = \frac{100 \times 10^3}{10} = 10K\Omega$$

For Hartley oscillator, the frequency of oscillations is given by

$$f_o = 1/ (2\pi \sqrt{(Leq\ C)})$$

Where Leq = L1 + L2

$$Leq = 1.0 \times 10^{-6} + 0.1 \times 10^{-6}$$

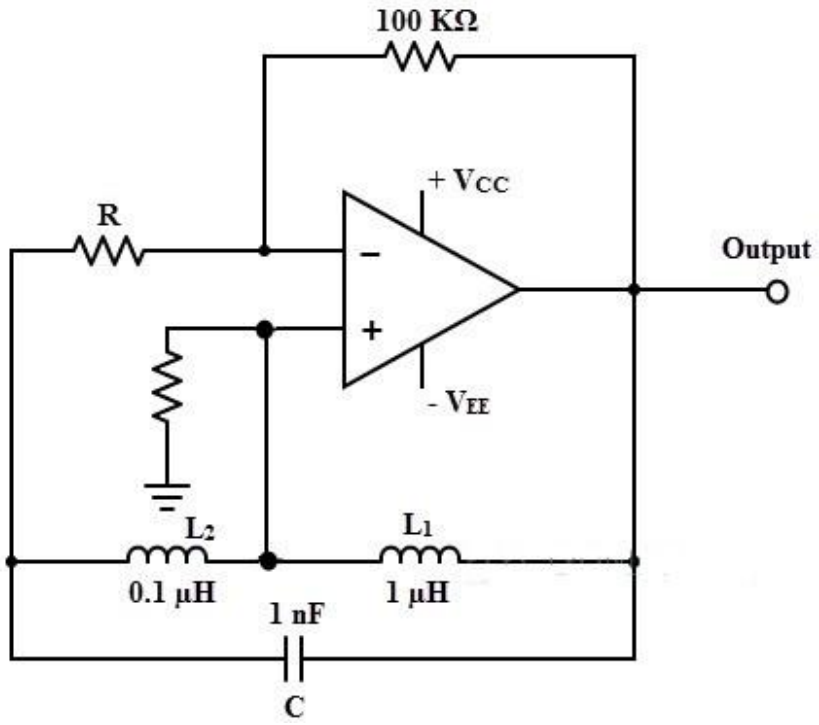
$$Leq = 1.1 \times 10^{-6}$$

The given capacitor value is $C = 1 \times 10^{-9}$ F

$$\text{Therefore, } fo = 1/ (2\pi \sqrt{(1.1 \times 10^{-6} \times 1 \times 10^{-9})}$$

$$= 4.799 \text{ MHz.}$$

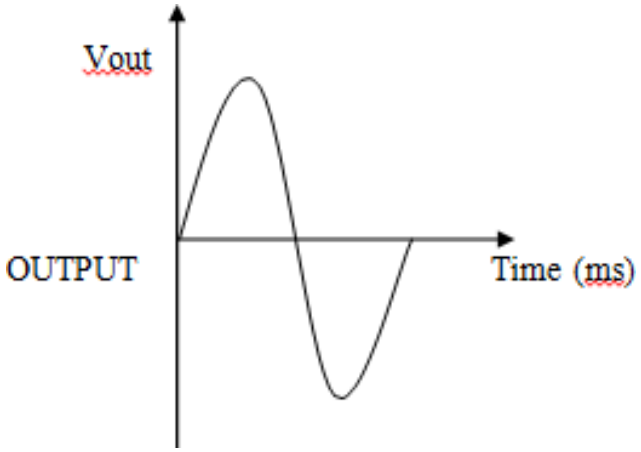
HARTLEY OSCILLATOR OSCILLATOR:-CIRCUIT DIAGRAM:-



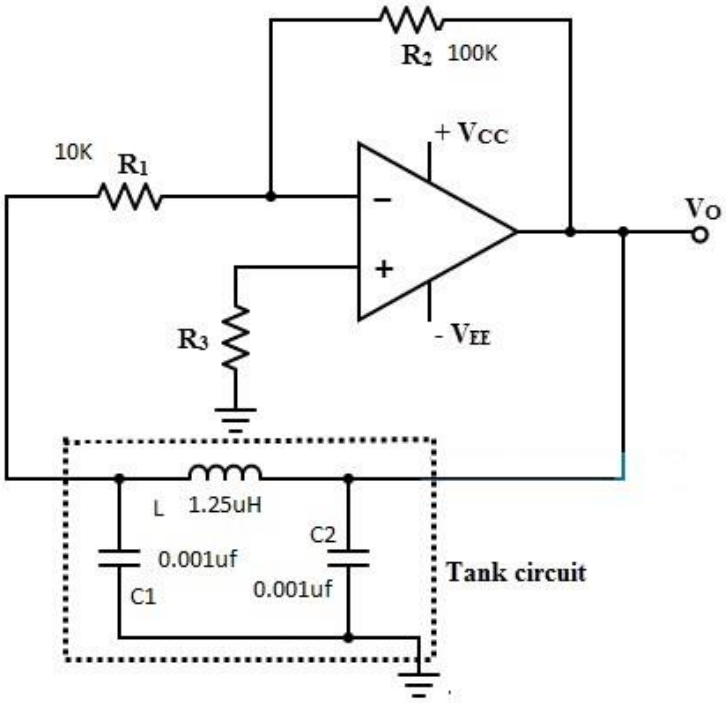
TABULATION:.

OBSERVED OUTPUT WAVEFORM			Design Frequency (Hz)
Amplitude (volts)	Time period(ms)	Frequency(Hz)	

MODEL GRAPH:



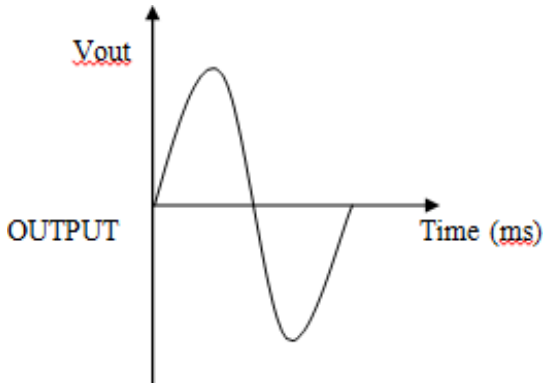
COLPITTS OSCILLATOR OSCILLATOR:-CIRCUIT DIAGRAM:-



TABULATION:.

OBSERVED OUTPUT WAVEFORM			Design Frequency (Hz)
Amplitude (volts)	Time period(ms)	Frequency(Hz)	

MODEL GRAPH:



PROCEDURE:

1. Assume either C or L to find out the other using formula $f_0 = \frac{1}{2\pi\sqrt{LCeq}}$.
2. The gain is selected such that $R_1 / R_2 = 10$. Assume R_2 or R_1 to find the other.
3. Connect the circuit as per as the circuit diagram.
4. Find the frequency of oscillation f_0 using above value and practical verification.
5. Measure the amplitude frequency of the output signal plot the graph.

DESIGN PROCEDURE: (HARTLEY):-

$f = 1/ (2\pi\sqrt{} (LCeq))$

Where Ceq = C1 C2 / (C1 + C2)

But in given data, C1= C2 = 0.001µF

Therefore, Ceq = (0.001× 10-6 × 0.001× 10-6) / (C0.001× 10-6 + 0.001× 10-6)

= 5 × 10-10 F

$f = 1/ (2\pi\sqrt{} (5 \times 10\text{-}6 \times 5 \times 10\text{-}10))$
$= 3.183$ MHz

The new value of frequency, $f = 2 \times 3.183$

$= 6.366$ MHz

Therefore the operating frequency equation becomes

$6.366 \times 106 = 1/ (2\pi\sqrt{} (L \times 5 \times 10\text{-}10))$

$L = 1.25$ µH

RESULT:

Thus Hartley Oscillator and Colpitts Oscillator were designed and tested usingop-amp IC 741.

EXP.NO:	INTEGRATOR AND DIFFERENTIATOR USING OP-AMP

AIM:

To design an Integrator and Differentiator using op-amp IC 741 and to test their characteristics & Performance.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741	- - -	01
2.	RESISTORS	100 Ω, 1.5KΩ	Each 02
		10KΩ, 15KΩ	Each 01
3.	CAPACITOR	0.1μf, 0.01μf	Each 01
		0.001μf,	05
4.	DIGITAL TRAINER KIT	- - -	01
5.	SIGNAL GENERATOR	(0-3)MHz	01
6.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
7.	CONNECTING WIRES	- - -	FEW

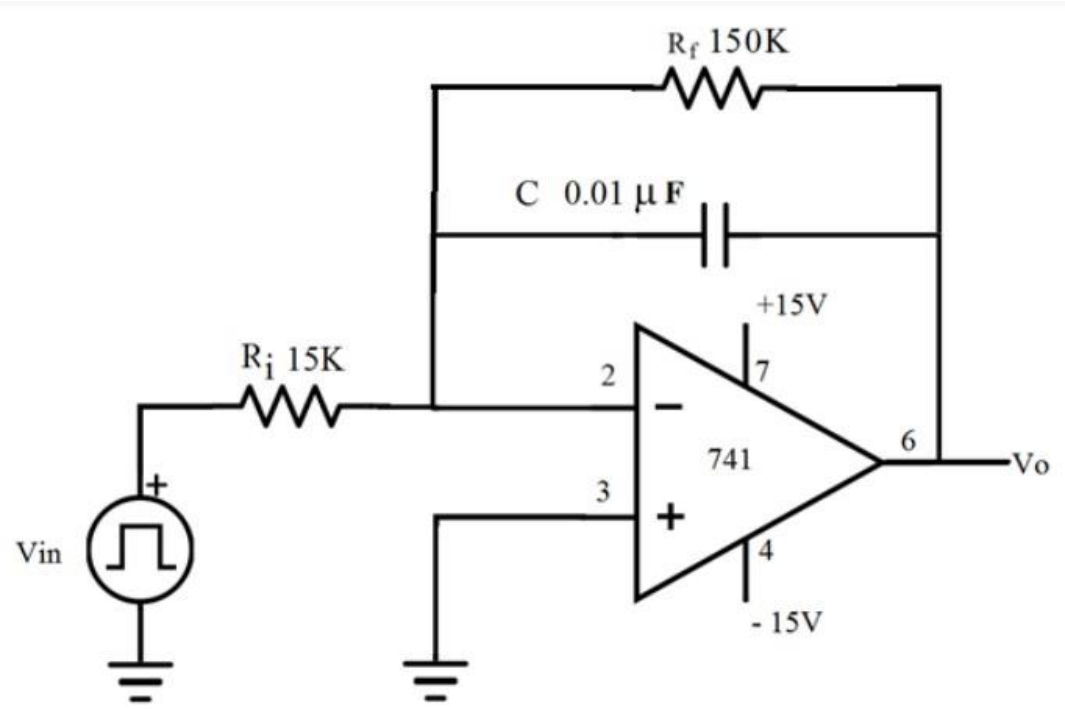
PROCEDURE:

1. From the given frequency f_a & f_b, the values of R_f, C_f, R_1 & R_{comp} are calculated as givenin the design procedure.
2. Connect the circuit as shown in the circuit diagram.
3. Apply the sinusoidal input as the constant amplitude to the inverting terminal of op-amp.
4. Gradually increase the frequency & observe the output amplitude.
5. Calculate the gain with respect to frequency & plot its graph.

PROCEDURE: DIFFERENTIATOR

1. Select f_a equal to the highest frequency of the input signal to be differentiated. Calculatethe component values of C_1 & R_f.
2. Choose $f_b = 20f_a$ & calculate the values of R_1 & C_f, so that $R_1C_1=R_f C_f$.
3. Connect the components as shown in the circuit diagram.
4. Apply a sinusoidal & square wave input to the inverting terminal of op-amp through R_1C_1.
5. Observe the shape of the output signal for the given input in CRO.
6. Note down the reading and plot the graph of input versus output wave for both cases.

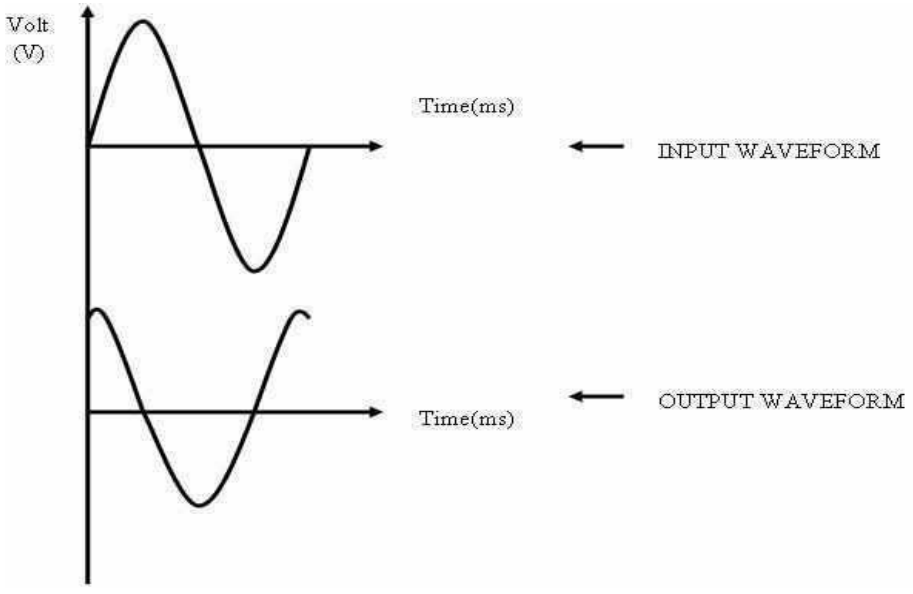
INTEGRATOR:- CIRCUIT DIAGRAM:-



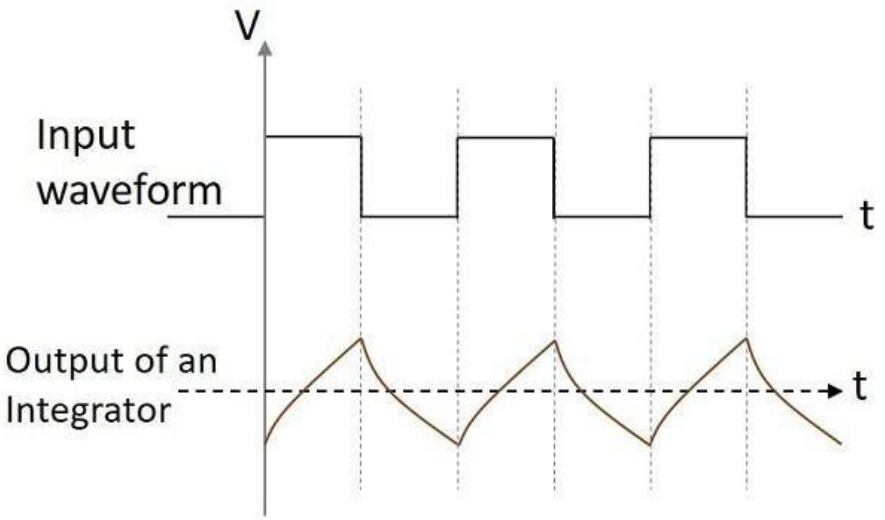
TABULATION:

	Input	Output
Amplitude		
Time Period		

MODELGRAPH: SINE WAVEFORM

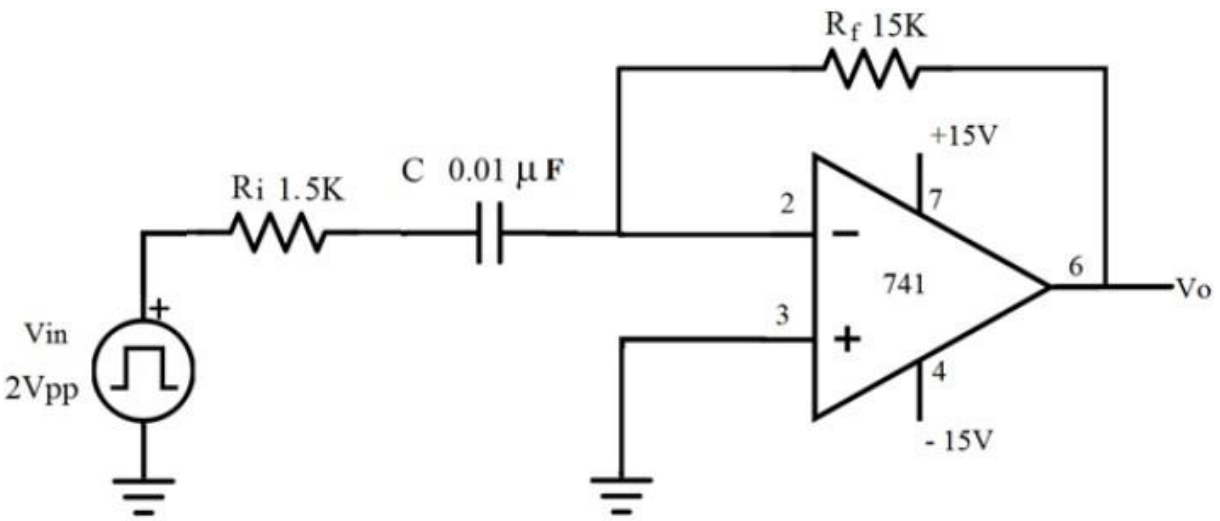


MODELGRAPH: SQUARE WAVEFORM



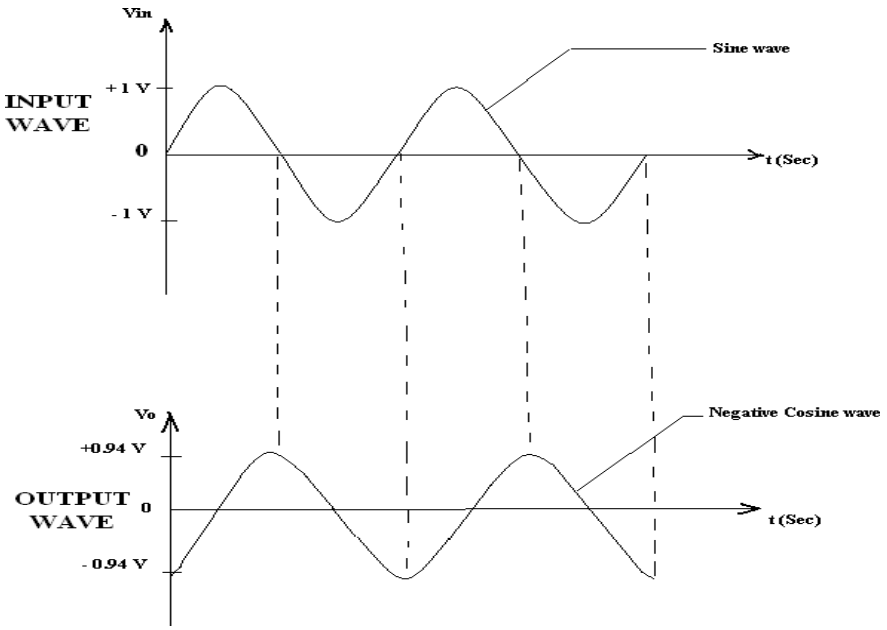
	Input	Output
Amplitude		
Time period		

DIFFERENTIATOR:-CIRCUIT DIAGRAM:



MODEL GRAPH:

(ii)FOR SINE WAVE INPUT



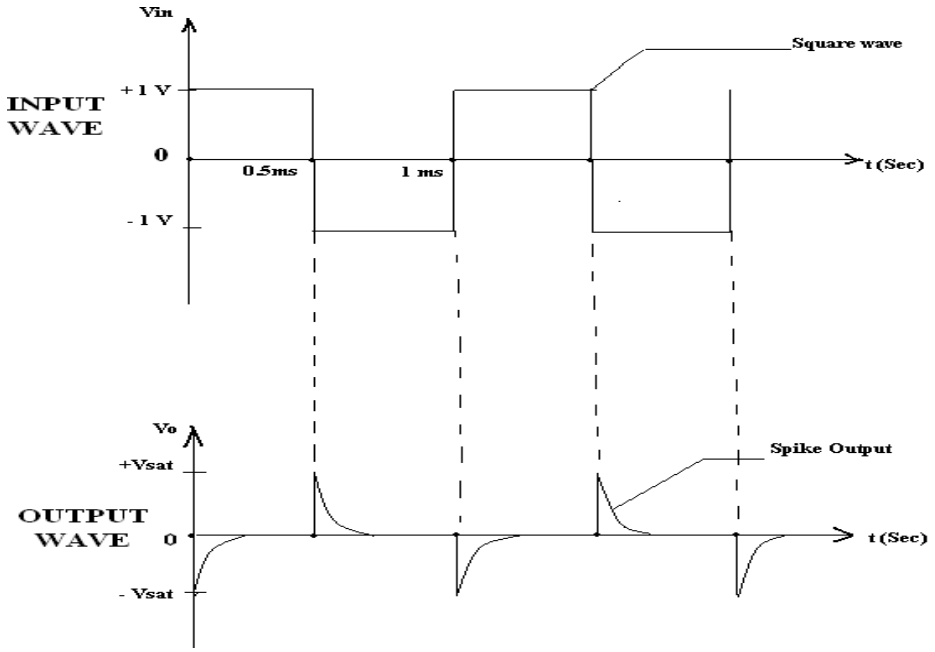
TABULATION:

	Input	Output
Amplitude		
Time period		

TABULATION:

	Input	Output
Amplitude		
Time period		

MODEL GRAPH: SQUARE WAVEFORM



DESIGN PROCEDURE-(INTEGRATOR):-

Design of integrator to integrate at cut-off frequency 1 KHz.

Take fa = $\frac{1}{2\pi R_f C_f}$

= 1 KHz.

Always take $C_f < 1$ µf and

Let $\boxed{C_f = 0.01\mu f}$

$R_f = \frac{1}{2\pi C_f f_a}$

$R_f = 15.9K\Omega$

$\boxed{R_f = 15\ K\Omega}$

Take $f_b = \frac{1}{2\pi R_1 C_f} = 10$ KHz.

$R_1 = \frac{1}{2\pi f_b C_f} = 1.59\ K\Omega.$

$\boxed{R_1 \approx 1.5K\Omega}$

$R_{comp} = R_1 // R_f = \frac{R_1 R_f}{R_1 + R_f} \approx R_1$, Assume $R_L = 10K\Omega$

$\boxed{R_{comp} = 1.5K\Omega}$

DESIGN PROCEDURE-(DIFFERENTIATOR):-

Design a differentiator to differentiate an input signal that varies in frequency from 10Hz to 1KHz. Apply a sine wave & square wave of 2Vp-p & 1 KHz frequency & observe the output.

To find R_f & C_1

Given: $f_a = 1$ KHz.

$$f_a = \frac{1}{2\pi R_f C1}$$

$f_a = 1$KHz.

Assume $C_1 = 0.1\mu f$

$$R_f = 1.59K\Omega \approx 1.5K\Omega$$

To find R_1 & C_f

Select $f_b = 20f_a$ with $R_1C_1 = R_f C_f$

$$f_b = 20KHz = \frac{1}{2\pi R_1 C_1}$$

$$R_1 = 79.5\Omega \approx 100\Omega$$

$$C_f = \frac{R_1 C_1}{R_f} = \frac{82X0.1X10^{-6}}{1.5K\Omega}$$

$$C_f = 0.005\mu f.$$

$$R_{OM} \approx R_1 // R_f = 100\Omega$$

RESULT:

Thus an Integrator and Differentiator using op-amp are designed and their performance was successfully tested using op-amp IC 741.

EXP.NO:	CLIPPER AND CLAMPER

AIM:

To design a Clipper and Clamper using op-amp IC 741 and to test their characteristics & Performance.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741	-	01
2.	RESISTORS	10KΩ.	06
		22KΩ	01
3.	DIGITAL TRAINER KIT	- - -	01
4.	SIGNAL GENERATOR	(0-3)MHz	02
5.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
6.	CONNECTING WIRES	- - -	FEW

PROCEDURE:

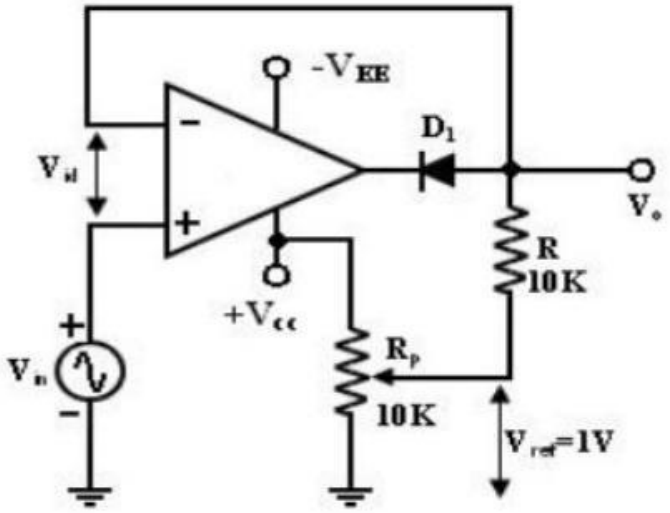
Clipper:

1. Connect the clipper circuit, set the function generator to 20 Vp-p, 1 kHz, sine wave
2. Connect the oscilloscope to Vo.
3. Vary the Vbias according to these values (2, 3, 5, 7, 9)
4. Plot the output waveform seen in the oscilloscope for each value.

Clamper:

1. Connect the clamper circuit, set the function generator to 20 Vp-p, 1 kHz, sine wave.
2. Connect the oscilloscope to Vo.
3. Vary the Vbias according to these values (2, 3, 5, 7, 9).
4. Plot the output waveform seen in the oscilloscope for each value.

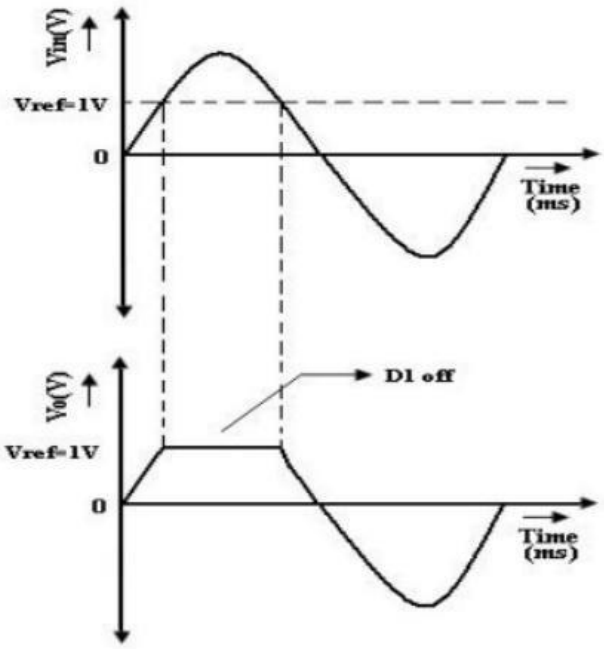
CIRCUIT DIAGRAM:POSITIVE CLIPPER



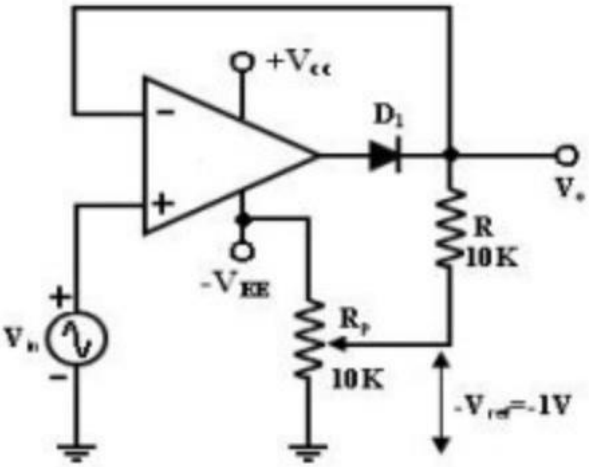
TABULATION:

	Input	Output
Amplitude		
Time period		

MODEL GRAPH: POSITIVE CLIPPER



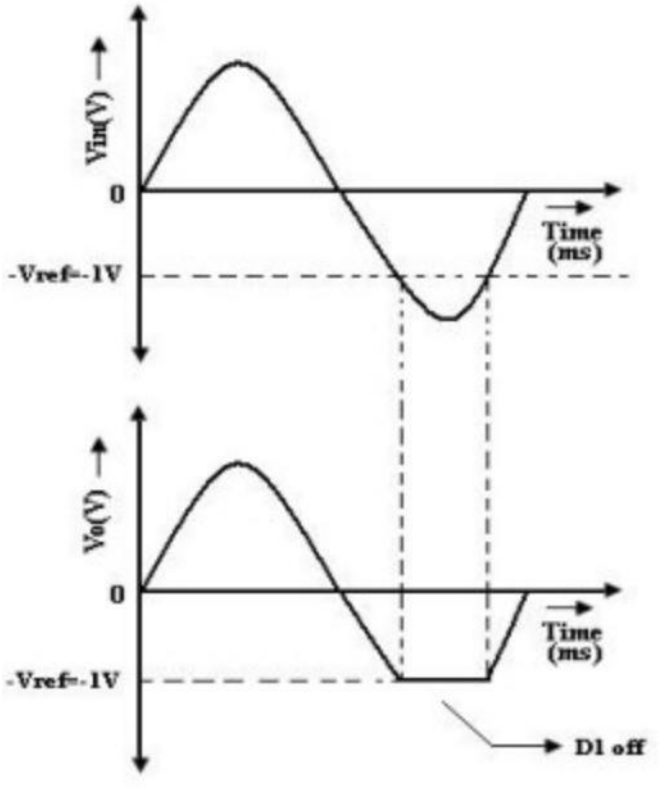
CIRCUIT DIAGRAM: NEGATIVE CLIPPER



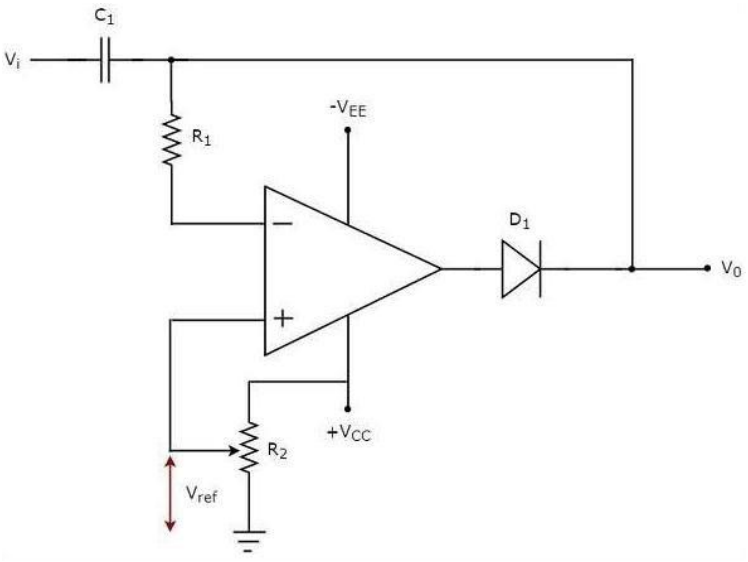
TABULATION:

	Input	Output
Amplitude		
Time period		

MODEL GRAPH: NEGATIVE CLIPPER



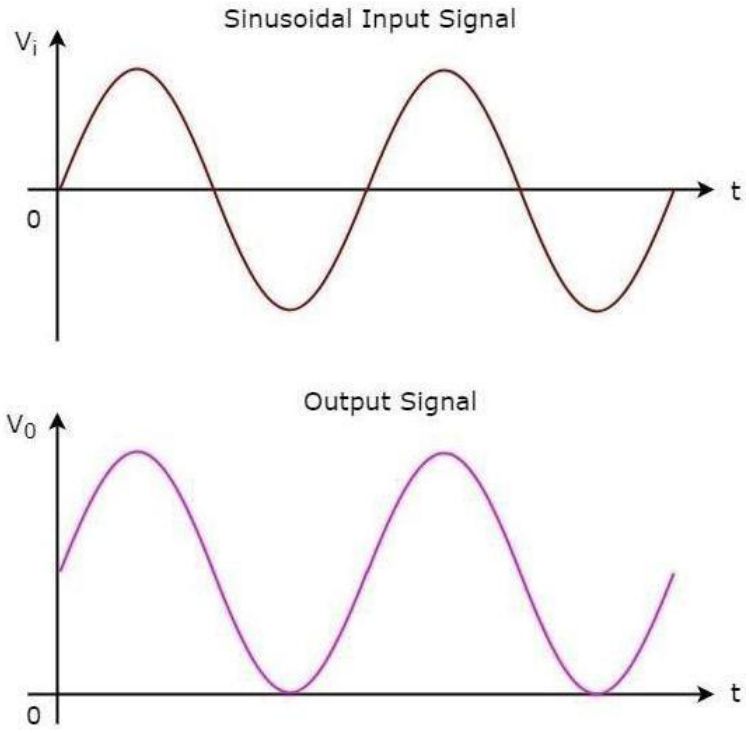
CIRCUIT DIAGRAM:POSITIVE CLAMPER



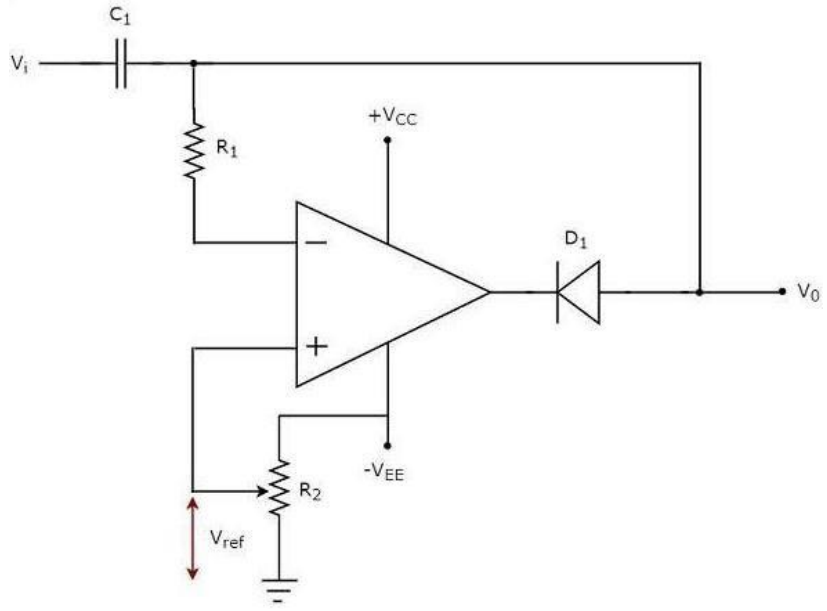
TABULATION:

	Input	Output
Amplitude		
Time period		

MODEL GRAPH: POSITIVE CLAMPER



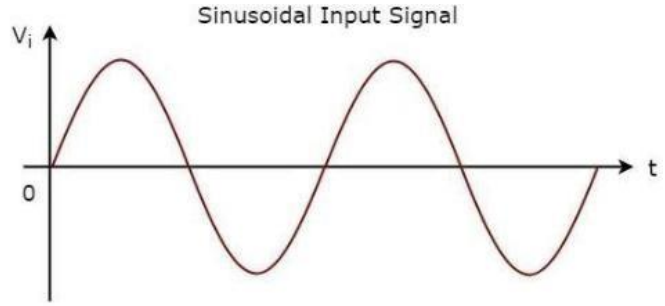
CIRCUIT DIAGRAM: NEGATIVE CLIPPER

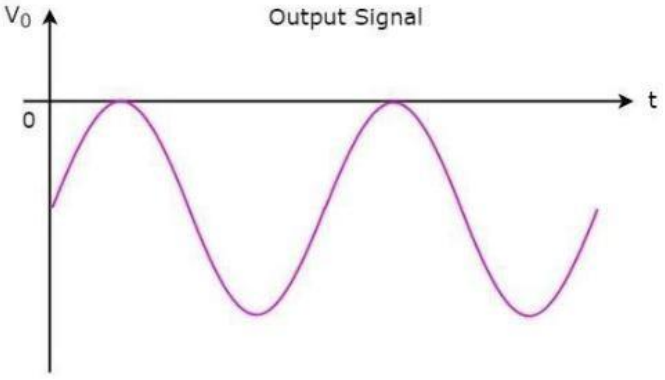


TABULATION:

	Input	Output
Amplitude		
Time period		

MODEL GRAPH: NEGATIVE CLIPPER





RESULT:

Thus a Clipper and Clamper was constructed and tested using op-amp IC 741.

EXP.NO:	INSTRUMENTATION AMPLIFIER

AIM:

To construct and test the CMRR (Common Mode Rejection Ratio) of a 3 op-amp instrumentation amplifier using op-amp IC741.

APPARATUS REQUIRED:

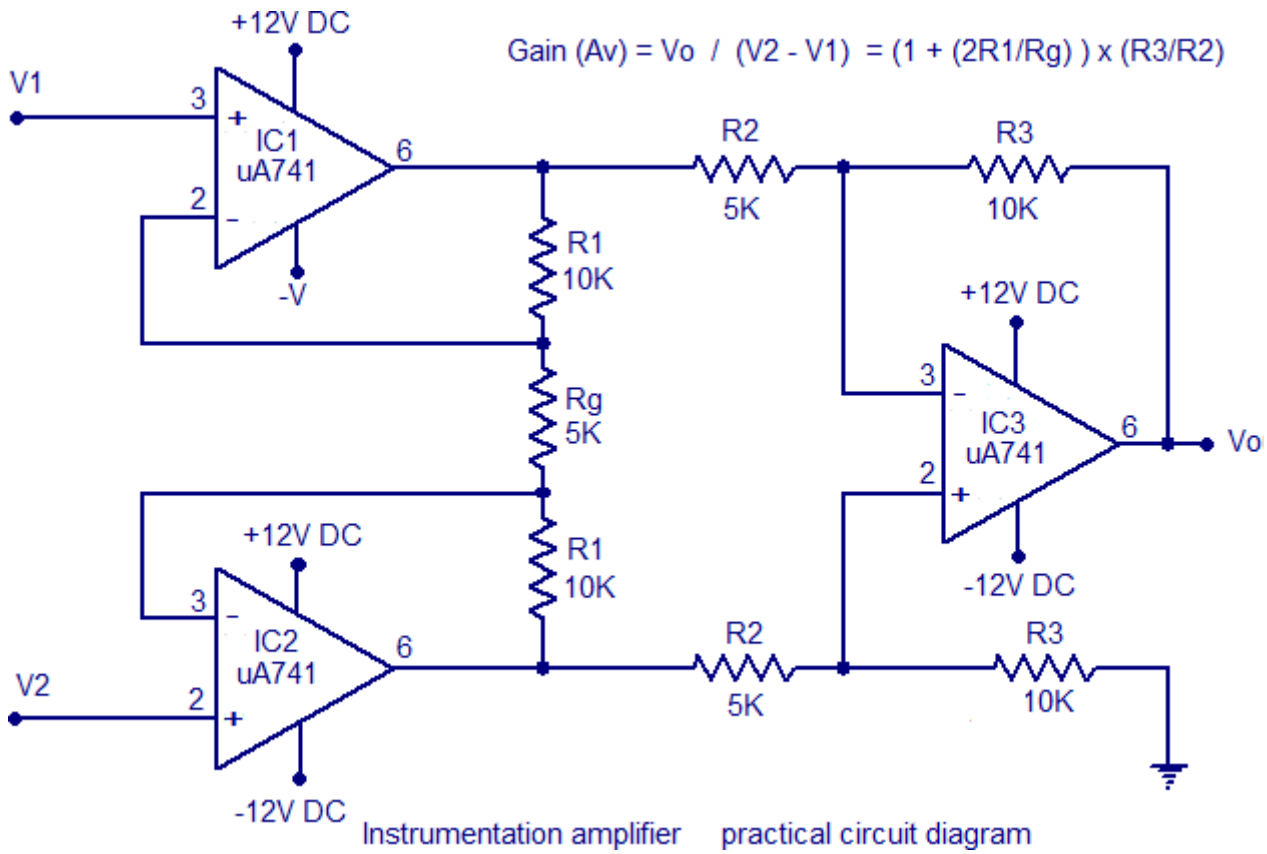
S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741		03
2.	RESISTORS	1KΩ.	06
		22KΩ	01
3.	DIGITAL TRAINER KIT	- - -	01
4.	SIGNAL GENERATOR	(0-3)MHz	02
5.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
6.	CONNECTING WIRES	- - -	FEW

PROCEDURE:

1. Select the entire resistor with same value of resistance R. Let R_G be the gain varyingresistor with different values of resistance. For simplicity, let R_G be a constant value.
2. Connect the circuit as shown in the circuit diagram.
3. Give the input V_1 & V_2 to the non-inverting terminals of first & secondOp-amp respectively.
4. By varying the value of R_G, measure the output voltage for common mode and differential mode operation. Since R_G is selected as constant value, provide different inputvalue of V_1 & V_2.
5. Calculate the differential mode gain A_d and common mode gain A_c to calculate the

CMRR as $$\text{CMRR}=20\log\left|\frac{A_d}{A_c}\right|$$

CIRCUIT DIAGRAM:



TABULATION:

COMMON MODE GAIN CALCULATION - A_C

S.No	R_G (KΩ)	V_1 (Volts)	V_2 (Volts)	V_o (Volts)	$A_c = \frac{V_o}{\frac{V_1 + V_2}{2}}$
1.					
2.					
3.					
4.					
5.					

DIFFERENTIAL MODE GAIN - A_D & CMRR CALCULATION.

S.No	R_G (KΩ)	V_1 (Volts)	V_2 (Volts)	V_o (Volts)	$A_d = \frac{V_o}{V_1 - V_2}$	CMRR = $20 \log \left(A_d / A_c \right)$(dB)
1.						
2.						
3.						
4.						
5.						

RESULT:

Thus a 3 op-amp instrumentation amplifier was constructed and CMRR is tested using op-amp IC 741.

EXP.NO:	ACTIVE LOW PASS, HIGH PASS AND BAND PASS FILTER USING OP-AMP

AIM:

To design an Active Low Pass, High Pass and Band Pass Filter using op-amp and to test their performance

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741	- - -	02
2.	RESISTORS	1.5 KΩ	02
		10KΩ	01
		22KΩ	04
3.	CAPACITORS	0.1μf, 0.01μf	01
4.	DIGITAL TRAINER KIT	- - -	01
5.	SIGNAL GENERATOR	(0-3)MHz	01
6.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
7.	CONNECTING WIRES	- - -	FEW

DESIGN PROCEDURE: (ACTIVE HPF):

Design a HPF at cutoff frequency f_L of 1KHZ & P.B gain of 2. Follow the same procedureas LPF & interchange the R & C position with capacitor first & resistor in parallel to capacitor where the other end connected to ground.

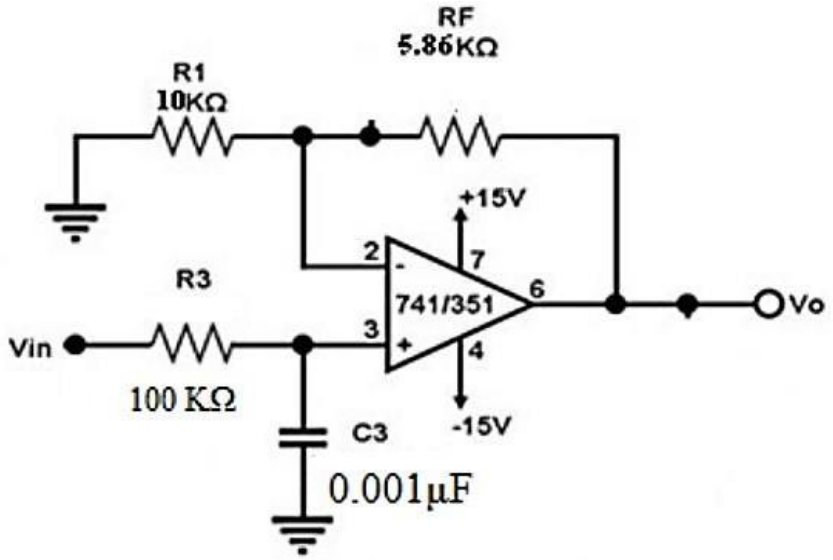
In high pass filter Theoretical gain is given as $\left|\frac{Vo}{Vin}\right| = \frac{Af(f/f_L)}{\sqrt{1+(f/f_H)^2}}$

PROCEDURE - (LPF & HPF):

1. Connect the circuit as shown in the circuit diagram.
2. Select the corresponding cut-off frequency (higher or lower) and determine the value ofC & R. select the value of R_1 & R_f depending on desired passband gain $A_{f..}$
3. Apply a constant voltage input sinusoidal signal to the non-inverting terminal of op-amp.
4. Tabulate the output voltage V_o with respect to different values of input frequency.
5. Calculate passband gain and plot the graph of frequency versus voltage gain & check the graph to get approximately the same characteristic as shown in the model graph.

LOWPASS FILTER:-

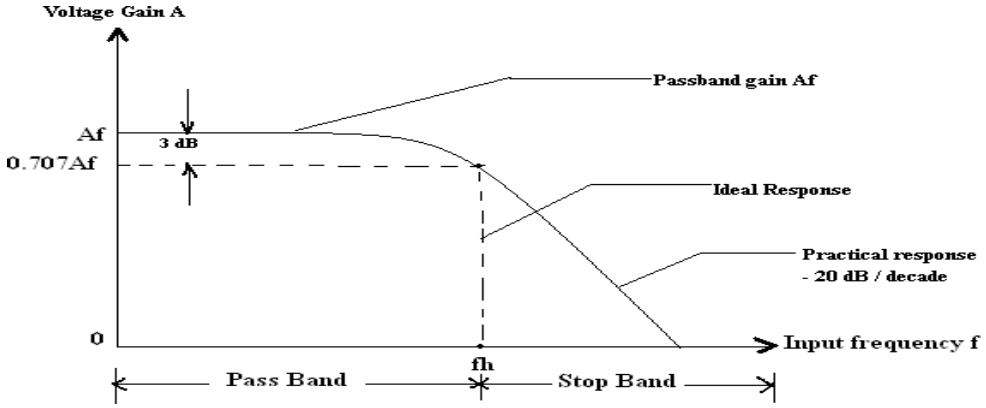
CIRCUIT DIAGRAM:-



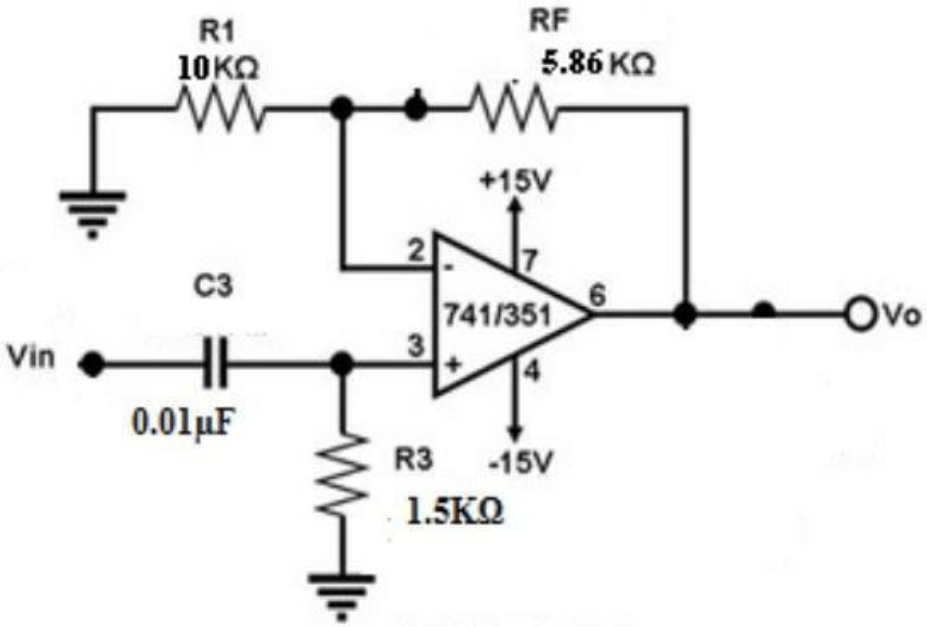
TABULATION:

S.No	Frequency(Hz)	Output Voltage (Volts)	Gain = 20 log (V_0 /V_{in}) (dB)
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

MODEL GRAPH:



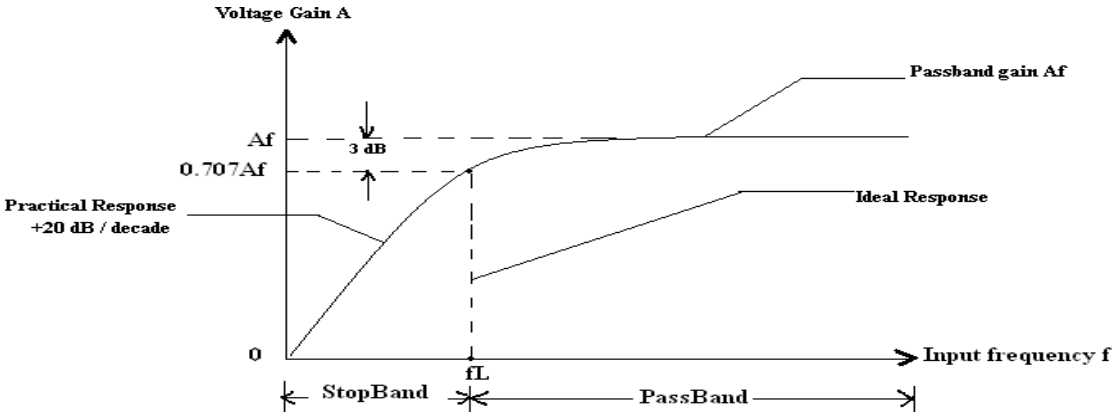
CIRCUIT DIAGRAM - (HIGH PASS FILTER):-



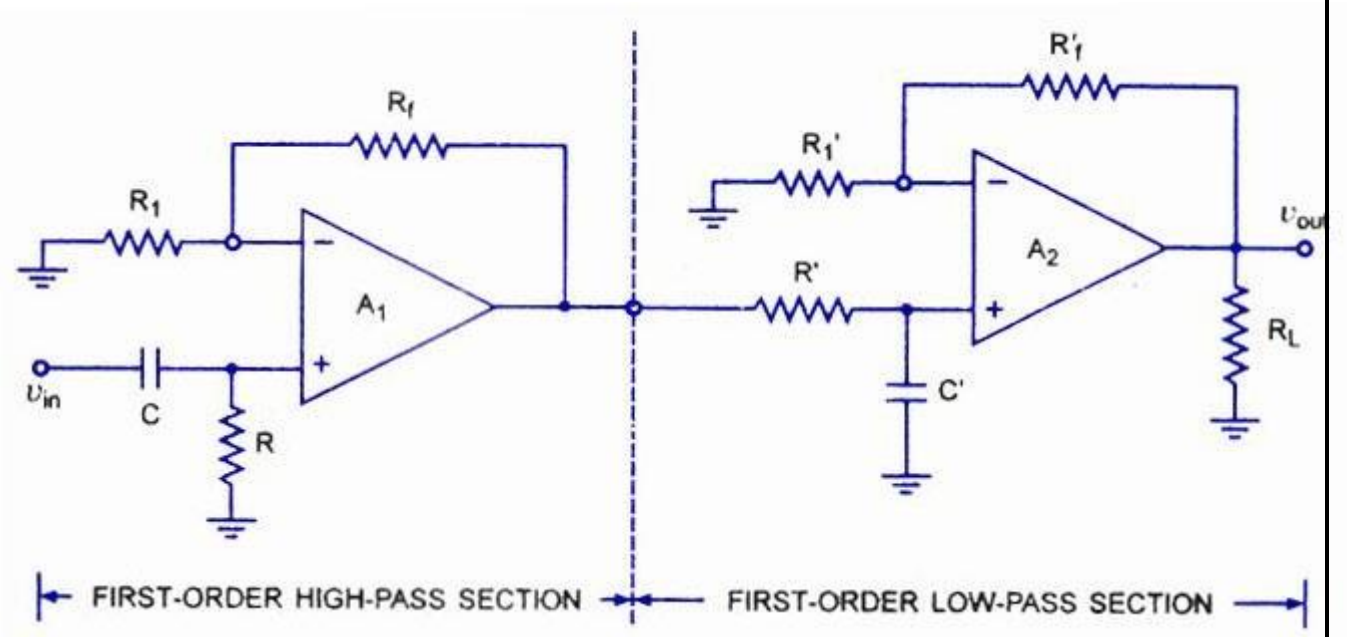
TABULATION:

S.No	Frequency(Hz)	Output Voltage (Volts)	Gain = 20 log (V_0 /V_{in}) (dB)
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

MODEL GRAPH:



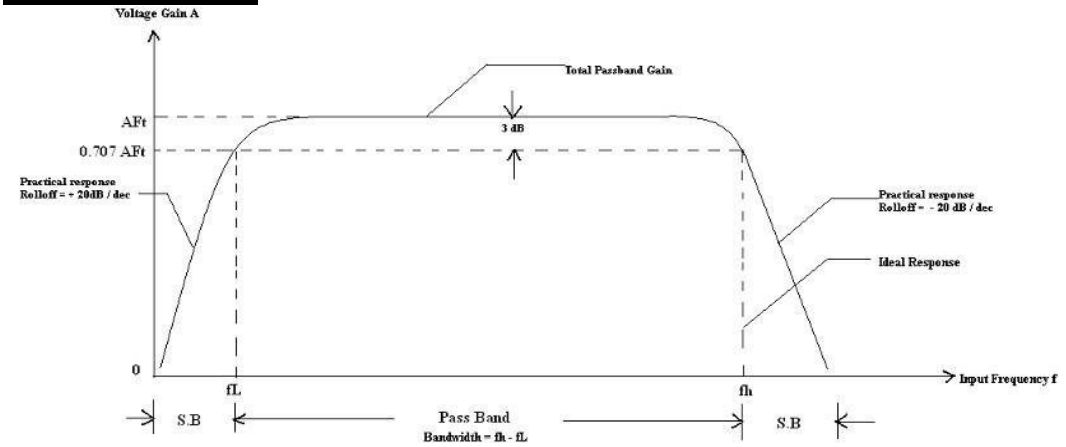
CIRCUIT DIAGRAM: (BANDPASS FILTER)



TABULATION:

S.No	Frequency(Hz)	Output Voltage (Volts)	Gain = 20 log (V_0 /V_{in}) (dB)
1			
2			
3			
4			
5			
6			
7			
8			

MODEL GRAPH:



PROCEDURE:

1. Select the lower and higher cut-off frequency and calculate the value of R & C for thegiven frequencies.
2. Design for LPF & HPF separately and then combine the circuit by first placing the HPF followed by a LPF (i.e.) HPF in series with LPF.
3. Connect the circuit as shown in the circuit diagram.
4. Apply a constant voltage input sinusoidal signal to the non-inverting terminal of op-amp.
5. Tabulate the output voltage V_o with respect to different values of input frequency.
6. Calculate passband gain and plot the graph of frequency versus voltage gain & check the graph to get approximately the same characteristic as shown in the model graph.

DESIGN PROCEDURE - (ACTIVE BPF):-

Design a BPF to pass a band of 1 KHz to 10 KHz with a passband gain of 4.

1. Select the highest cut-off frequency of LPF as f_H = 10 KHz and the lowest cut-off frequency of HPF as f_L = 1 KHz.
2. Design the HPF first by taking f_L = 1KHz. Assume the value of C < 1µf.
   Let C = 0.1µf.
3. Calculate R from the expression.

$$f_L = \frac{1}{2\pi RC}; \qquad \text{Therefore } R_1 = \frac{1}{2\pi f_L C}$$

$$R = \frac{1}{2\pi(1KHz)(0.1X10^{-6})};$$

$$\boxed{R = 1.59K\Omega \approx R=1.5K\Omega}$$

4. Then design the LPF by taking f_H = 10KHz. Assume the value of C < 1μf. Let C = 0.01μf.

5. Calculate R from the expression $f_H = \frac{1}{2\pi RC}$; Therefore $R = \frac{1}{2\pi f_H C}$

$$R = \frac{1}{2\pi(10KHz)(0.01X10^{-6})};$$

$$\boxed{R = 1.59K\Omega \approx R=1.5K\Omega}$$

6. Calculate the values of R_f & R_1 with the use of pass band gain.

   Overall P.B gain of $\boxed{BPF = 4 = 2\ (HPF)\ X\ 2\ (LPF)}$

DESIGN PROCEDURE (ACTIVE LPF):

Design a LPF at cutoff frequency f_H of 1KHz with a passband gain of 2.

1. Choose the given value of f_H = 1KHz.

2. Select the value of C < 1μf

3. Assume C = 0.1μf. Calculate R from $f_H = \frac{1}{2\pi RC}$

$$R = \frac{1}{2\pi f_H C}$$

$$R = \frac{1}{2\pi X 1 X 10^3 X 0.1\mu f} = 1.5K\Omega$$

R = 1.5KΩ C = 0.1μf

4. Determine the value of R_1 & R_f from pass band gain of the filter.

$$A_f = 1 + \frac{Rf}{R1} = 2.$$

Therefore $R_f = R_1$ to select $A_f = 2$.

Assume $R_f = R_1 = 22K\Omega$ & Assume $R_L = 10K\Omega$

Therefore for both HPF & LPF the value of $R_f = R_1$ to obtain a individual P.B gain of 2.

$$A_f = (1+\frac{Rf}{R1}) = 2 \text{ (for HPF)}$$

$$A_f = (1+\frac{Rf}{R1}) = 2 \text{ (for LPF)}$$

Let $R_f = R_1 = 22K\Omega$.

7. Q of the filters is calculated as $\frac{f_c}{B.W} = \frac{f_c}{f_H - f_L}$

Where $f_C = \sqrt{f_H f_L}$ is the center frequency.

8. Cascade HPF & then LPF to form BPF.

Calculate the practical gain dB using 20 log (Vo/Vin)

And Theoretical gain is $\left|\frac{Vo}{Vin}\right| = \frac{Af_T(\frac{f}{f_L})}{\sqrt{[1+(\frac{f}{f_H})^2][1+(\frac{f}{f_L})^2]}}$

5. Calculate the practical gain in dB using Gain (dB)=20log (Vo/Vin);

Theoretical gain is given as $\left|\frac{Vo}{Vin}\right| = \frac{Af}{\sqrt{1+(f/f_H)^2}}$

A_f – P.B gain.

f – Input frequency.

f_H – Higher cut-off frequency of LPF.

RESULT:

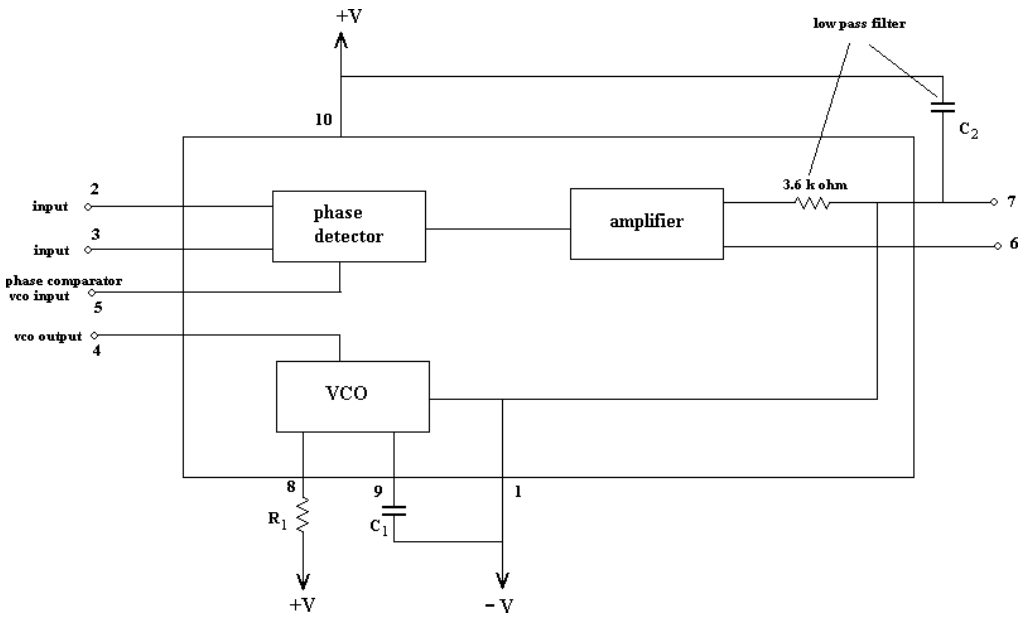
Thus an Active Lowpass, High pass and Band Pass Filters are designed and tested usingop-amp IC 741.

PLL IC 565

THEORY

The Signetics SE/NE 560 series is monolithic phase locked loops. The SE/NE 560, 561, 562, 564, 565, & 567 differ mainly in operating frequency range, power supply requirements and frequency and bandwidth adjustment ranges. The device is available as 14 Pin DIP package and as 10-pin metal can package. Phase comparator or phase detector compare the frequency of input signal f_s with frequency of VCO output f_o and it generates a signal which is function of difference between the phase of input signal and phase of feedback signal which is basically a d.c voltage mixed with high frequency noise. LPF remove high frequency noise voltage. Output is error voltage. If control voltage of VCO is 0, then frequency is center frequency (f_o) and mode is free running mode. Application of control voltage shifts the output frequency of VCO from f_o to f. On application of error voltage, difference between f_s & f tends to decrease and VCO is said to be locked. While in locked condition, the PLL tracks the changes of frequency of input signal.

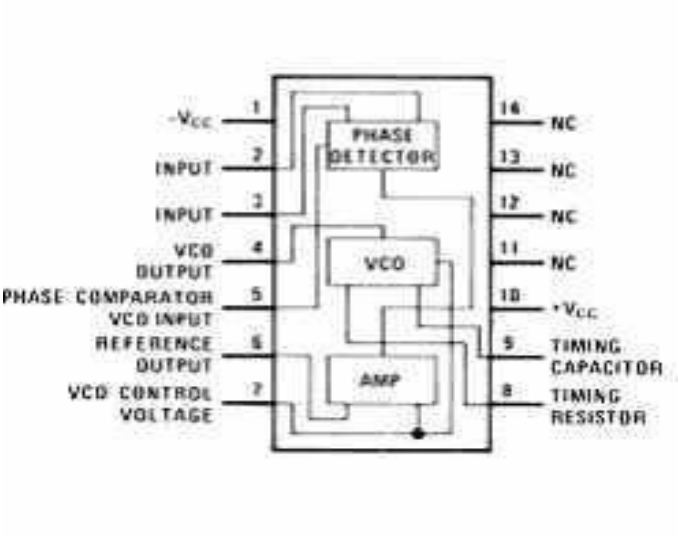
Block Diagram of IC 565



PROCEDURE:

1. Determine the component values using the design procedure given here.
2. Connect the components as shown in the circuit diagram.
3. Note down the readings of output waveform with respect to input signal.

Pin Configuration:



Specifications:

1.	Operating frequency range	:	0.001 Hz to 500 KHz
2.	Operating voltage range	:	±6 to ±12V
3.	Inputs level required for tracking	:	10mV rms minimum to 3v (p-p) max.
4.	Input impedance	:	10 KΩ typically
5.	Output sink current	:	1mA typically
6.	Drift in VCO center frequency	:	300 PPM/^{o}C typically(f_{out}) with temperature
7.	Drif in VCO centre frequency with	:	1.5%/V maximumsupply voltage
8.	Triangle wave amplitude	:	typically 2.4 V_{PP} at ± 6V
9.	Square wave amplitude	:	typically 5.4 V_{PP} at ± 6V
10.	Output source current	:	10mA typically
11.	Bandwidth adjustment range	:	<±1 to >± 60%Center frequency f_{out} =

$1.2/4R_1C_1$ Hz

= free running frequency$F_L = \pm 8\ f_{out}/V$ Hz

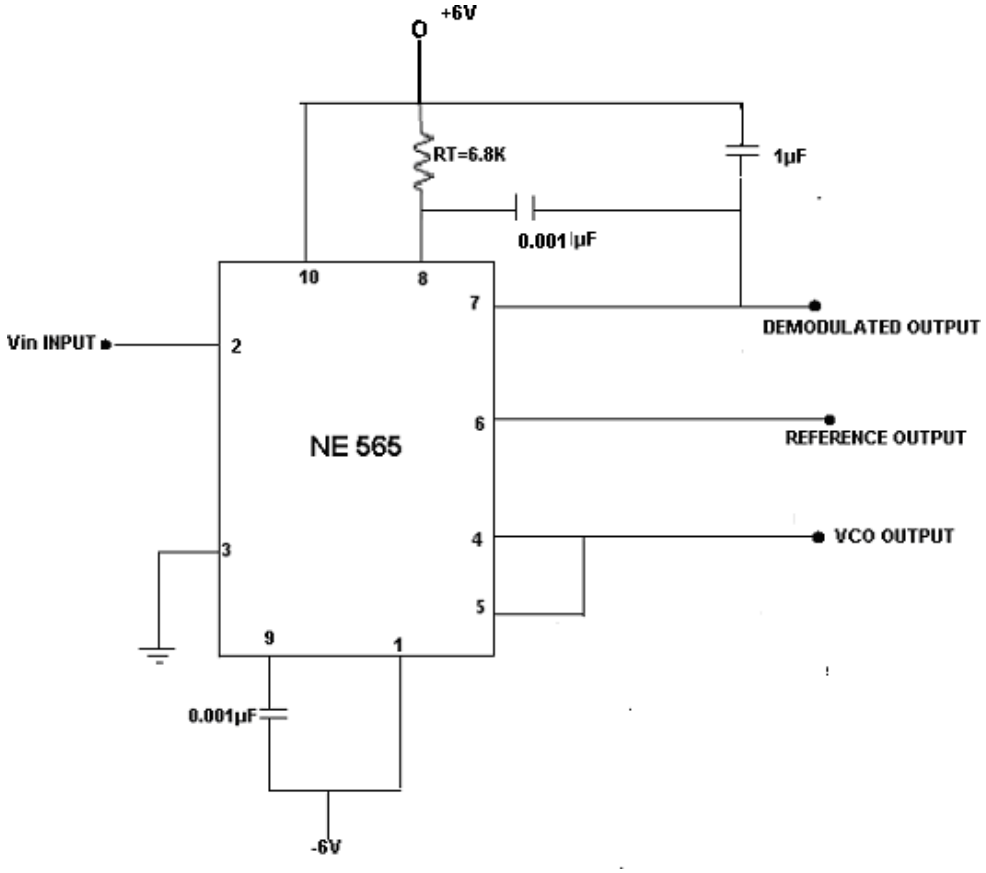
$V = (+V) - (-V)$

$$f_c = \pm \left[\frac{f_3}{2\Pi(3.6)x10\ xC2} \right]^{1/2}$$

Applications:

1. Frequency multiplier
2. Frequency shift keying (FSK) demodulator
3. FM detector

CIRCUITDIAGRAM



NE 565 PLL connection diagram

DESIGN PROCEDURE:-

If C= 0.01µF and the frequency of input trigger signal is 2KHz, output pulse width of 555 in Monostable mode is given by

$1.1R_AC = 1.2T = 1.2/f$ $R_A = 1.2/(1.1Cf) = 54.5K\Omega$

$f_{IN} = f_{OUT}/N$ Under locked conditions,

$f_{OUT} = Nf_{IN} = 2f_{IN} = 4KHz$

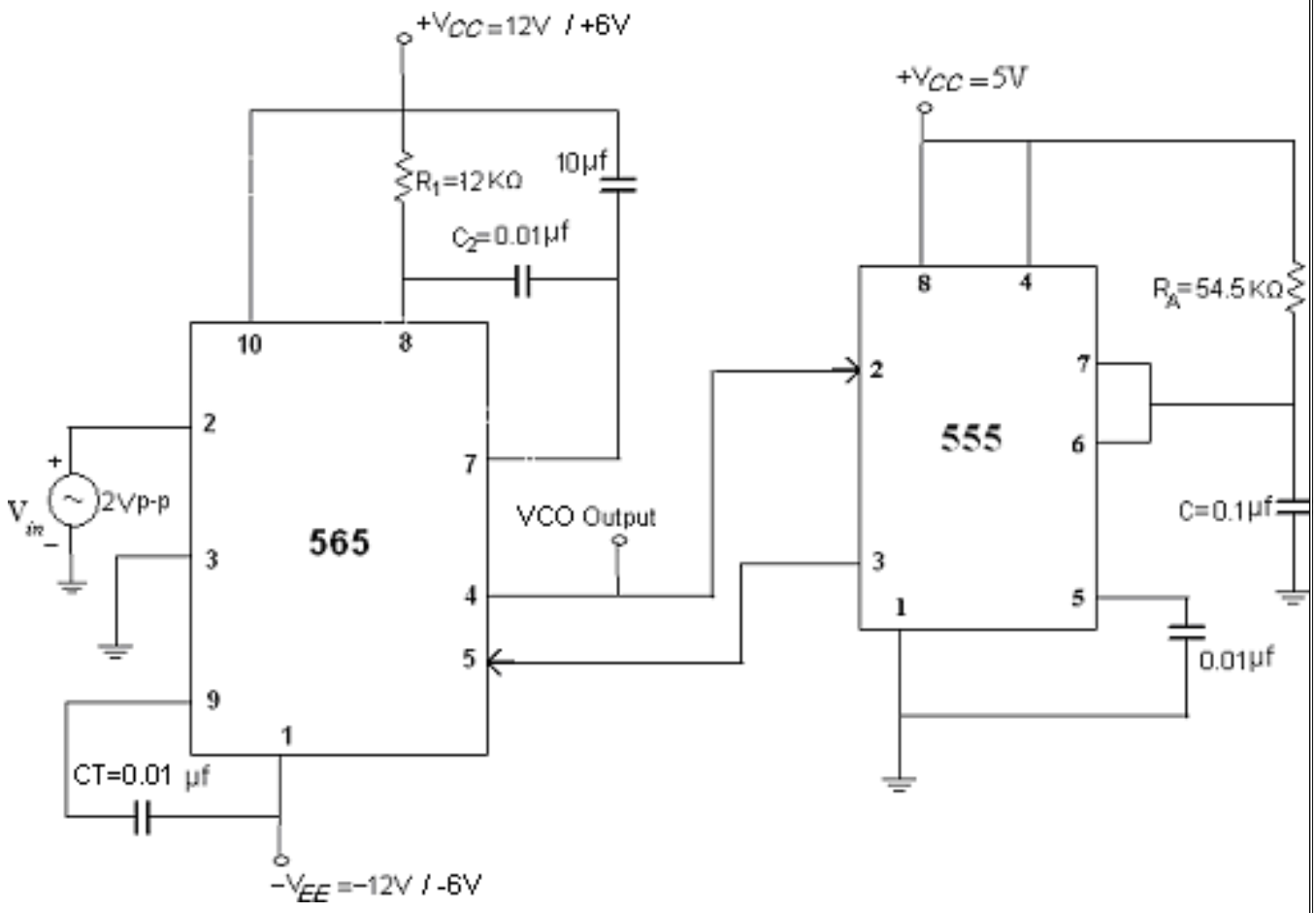
EXP.NO:	PLL CHARACTERISTICS AND FREQUENCY MULTIPLIER USING PLL

AIM:

To design & test the characteristics of PLL and to construct and test frequency multiplier using PLL IC565.

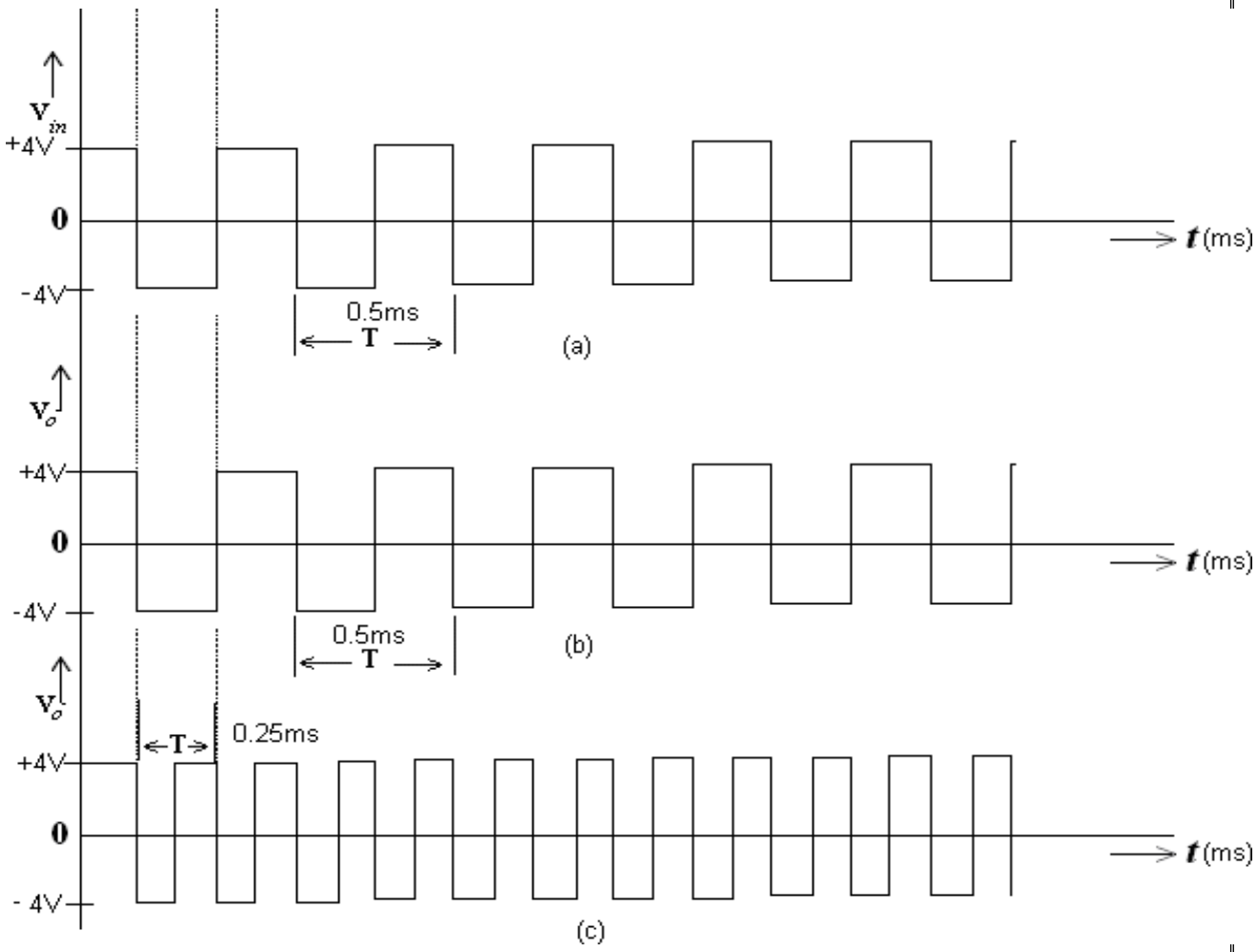
APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	VALUE	QUANTITY
1	IC 565	- - -	01
2	IC 555	- - -	01
3	RESISTORS	12KΩ, 54.5 KΩ, 6.8K	Each one
4	CAPACITORS	0.01μF	4
		0.1 μf, 10μf, 1 μf	EACH 01
5	DIGITAL TRAINER KIT	- - -	01
6	REGULATED POWER SUPPLY	(0 -30V), 1A	1
7	CATHODE RAY OSCILLOSCOPE	(0 – 30MHz)	1
8	CONNECTING WIRES	- - -	FEW



THEORY:

The frequency divider is inserted between the VCO and the phase comparator of PLL. Since the output of the divider is locked to the input frequency f_{IN}, the VCO is actually running at a multiple of the input frequency .The desired amount of multiplication can be obtained by selecting a proper divide– by – N network ,where N is an integer. To obtain the output frequency $f_{OUT}=2f_{IN}$, N = 2 is chosen. One must determine the input frequency range and then adjust the free running frequency f_{OUT} of the VCO by means of R_1 and C_1 so that the output frequency of the divider is midway within the predetermined input frequency range. The output of the VCO now should be $2f_{IN}$. The output of the VCO should be adjusted by varying potentiometer R_1. A small capacitor is connected between pin7 and pin8 to eliminate possible oscillations. Also, capacitor C_2 should be large enough to stabilize the VCO frequency



PLL as Frequency Multiplier

(a) : Input

(b) : PLL output under locked conditions without 555

(c) : Output at pin4 of 565 with 555 connected in the feedback

SAMPLE READINGS:

PARAMETER	INPUT	OUTPUT
Amplitude (V_{p-p})		
Frequency (KHz)		

PROCEDURE:-

1. The circuit is connected as per the circuit diagram.
2. Apply a square wave input to the pin2 of the 565
3. Observe the output at pin4 of 565 under locked condition.
4. Give the output of 565 to the pin2 of 555 IC.
5. Observe the output of 555 at pin3.
6. Now give the output of 555 as feedback to the pin5 of the 565.
7. Observe the frequency of output signal fo at pin4 of 565 IC.
8. Plot the waveforms in graph.

RESULT:

Thus the PLL characteristics are designed and tested and Frequency multiplier using IC565 is constructed and tested.

EXP.NO:	R-2R LADDER TYPE D- A CONVERTER USING OP-AMP

AIM:

To design a 4-bit R-2R ladder type DAC using OP-AMP.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	VALUE	QUANTITY
1	IC 741	- - -	01
2	RESISTOR	22K, 10K	05, 04
3	DIGITAL MULTIMETER METER		
4	DIGITAL TRAINER KIT	- - -	01
5	REGULATED POWER SUPPLY	(0 -30V), 1A	1
6	CATHODE RAY OSCILLOSCOPE	(0 – 30MHz)	1
7	CONNECTING WIRES	- - -	FEW

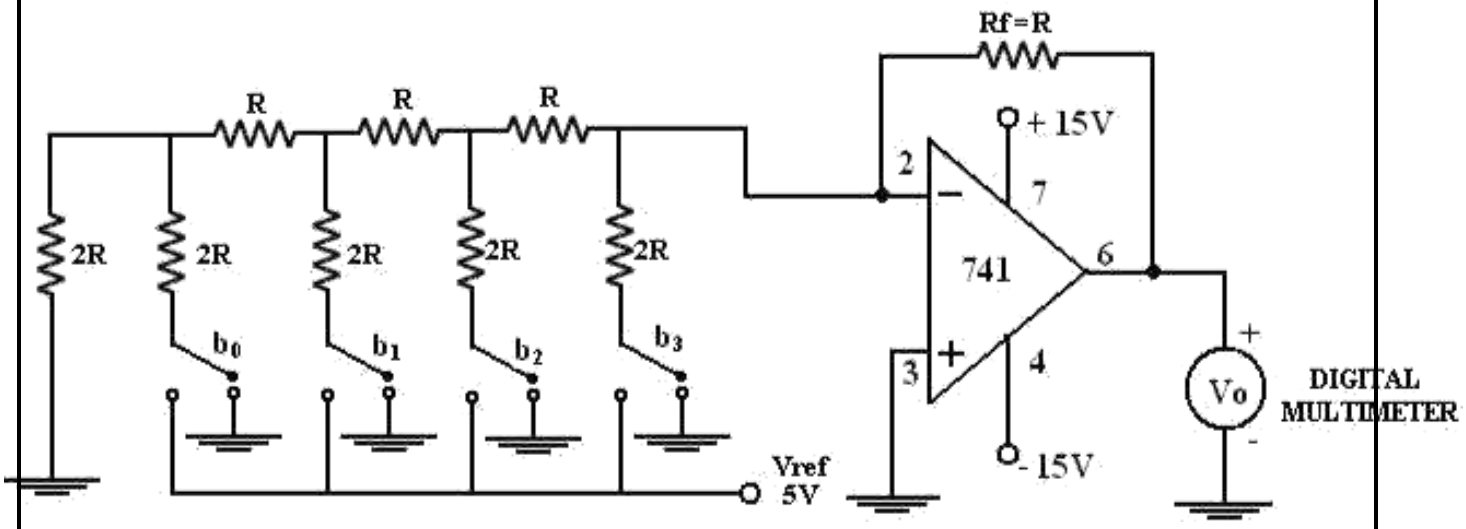
THEORY:

In R-2R ladder network only two values of resistors are required. Consider 4bit DAC, where switch position d1,d2,d3,d4 corresponding to binary words.

PROCEDURE:

1. Connections are made as per the circuit diagram.
2. The inputs are given through b0,b1,b2,b3.
3. The inputs are given from (0-15)V and observe the outputs in voltmeter.
4. The graph is drawn.

CIRCUIT DIAGRAM:



TABULATION:

Decimal	Equivalent Binary				Practical Voltage (V)	Theoretical Voltage (V)
	b3	b2	b1	b0		

DESIGN PROCEDURE:

$V_o = -R_f (b_3/2R + b_2/4R + b_1/8R + b_0/16R) V_{ref} R_f = R$

$V_o = -V_{ref} (b_3/2 + b_2/4 + b_1/8 + b_0/16)$ Assume R = 10K

2R = 22K

RESULT:

Thus the 4-bit R-2R ladder type DAC is designed and its outputs are verified.

CONTENT BEYOND THE SYLLABUS

EXP.NO:	INVERTING, NON-INVERTING AND DIFFERENTIAL AMPLIFIERS USING OP-AMP

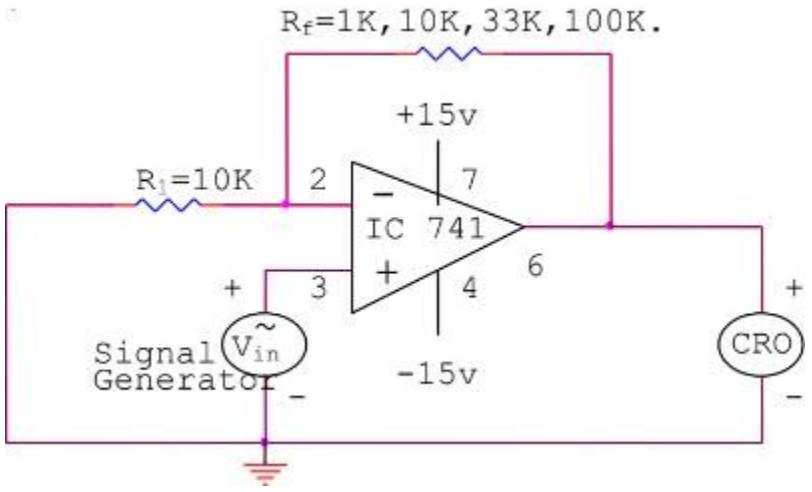
AIM:

To design the Inverting, Non-Inverting and Differential Amplifiers using Op-amp IC741 and test their performance.

APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	IC 741		01
2.	RESISTORS	1KΩ, 33KΩ	EACH 01
		10KΩ, 100 KΩ.	EACH 02
3.	DIGITAL TRAINER KIT	- - -	01
4.	SIGNAL GENERATOR	(0-3)MHz	01
5.	CATHODE RAY OSCILLOSCOPE	(0-30)MHz	01
6.	CONNECTING WIRES	- - -	FEW

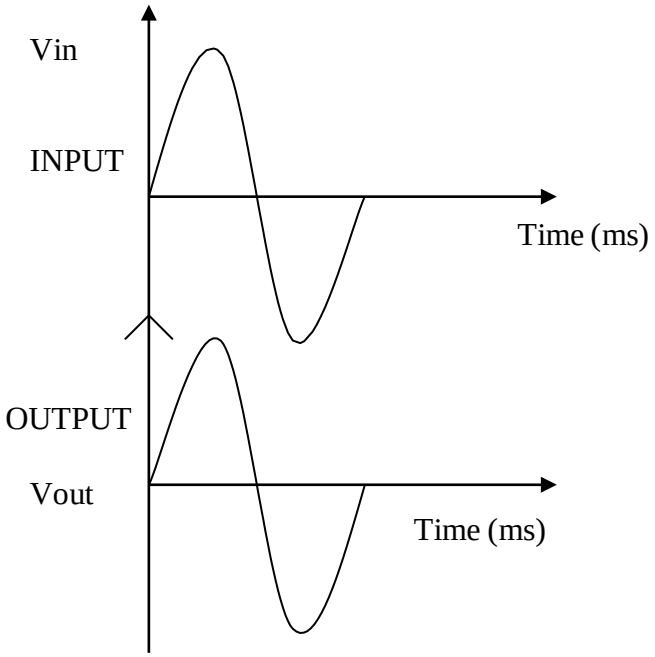
NON-INVERTING AMPLIFER:-CIRCUIT DIAGRAM:-



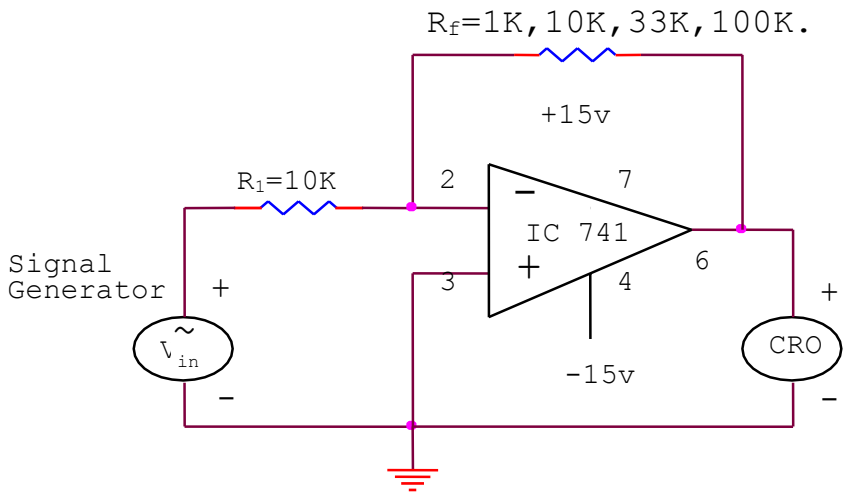
TABULATION:

	Input	Output
Amplitude		
Time Period		

MODEL GRAPH:



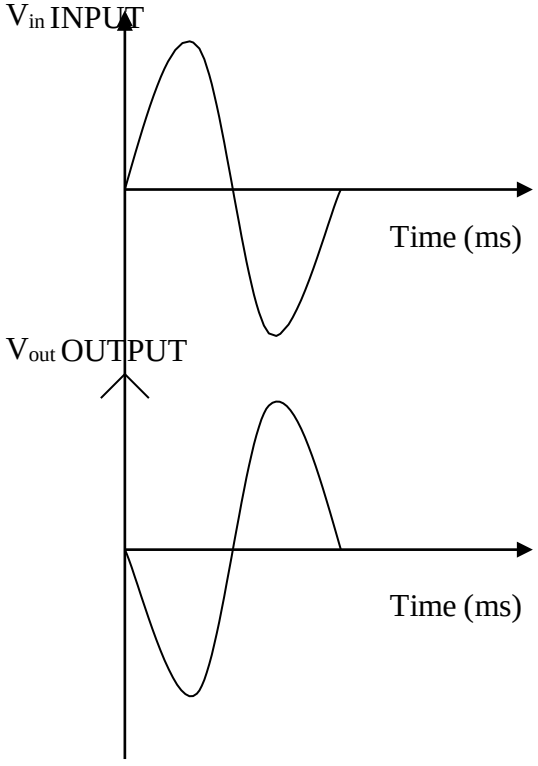
INVERTING AMPLIFIER:-CIRCUIT DIAGRAM:-



TABULATION:

	Input	Output
Amplitude		
Time Period		

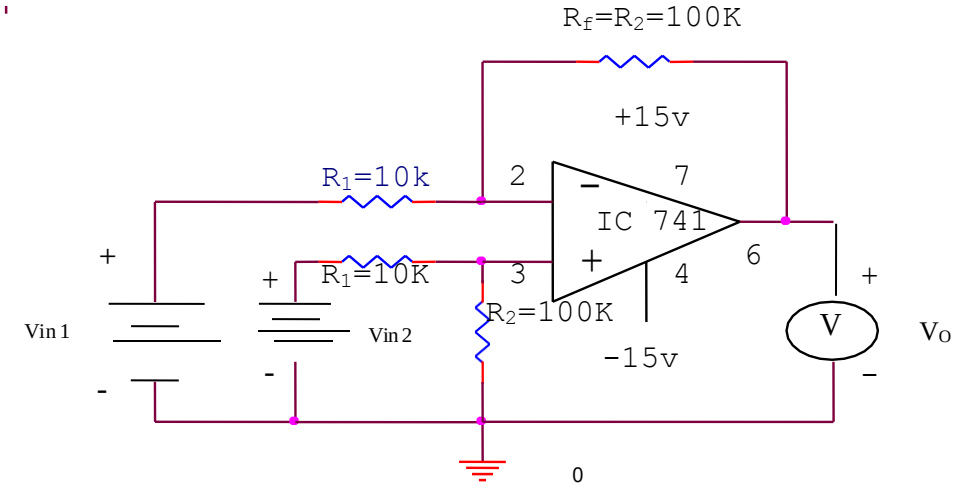
MODEL GRAPH:



PROCEDURE-(INVERTING & NON-INVERTING AMPLIFIER):-

1. Select R_1 as a constant value and choose a value of R_f.
2. Connect the circuit as per as the circuit diagram.
3. Apply the constant amplitude input voltage to the circuit.
4. Measure the output voltage amplitude for different value of R_f from CRO.
5. Calculate the practical gain for different value of R_f & compare it with theoretical gain.
6. Practical gain & theoretical gain should be approximately equal.
7. Plot the graph of the input wave versus output wave for any one practical case.

DIFFERENTIAL AMPLIFIER:-CIRCUIT DIAGRAM:-



TABULATION:

S.No	V_{in1} (Volts)	V_{in2} (Volts)	$V_{in2} - V_{in1}$ (Volts)	V_0 (Volts)	Theoretical Gain $A = -R_f / R_1$	Practical Gain $A=V_0 / (V_{in2} - V_{in1})$

PROCEDURE:

1. Select the value of R_1, R_2, R_3 & R_f such that $R_1=R_2$ and $R_3=R_f$.
2. Connect the circuit as per as the circuit diagram.
3. Provide constant input voltage V_{in1} to Non-inverting terminal of op-amp through R_1 & constant input voltage V_{in2} to inverting terminal of op-amp through R_2.
4. Measure the output voltage using CRO.
5. Calculate the theoretical gain and compare it with practical gain.
6. Practical gain & theoretical gain should be approximately equal.
7. Plot the graph of the input wave versus output wave for any one practical case.

RESULT:

Thus the Inverting, Non-Inverting and Differential Amplifiers are designed and their performance was successfully tested using op-amp IC 741.

EXP.NO:	DC POWER SUPPLY USING LM317 AND LM723

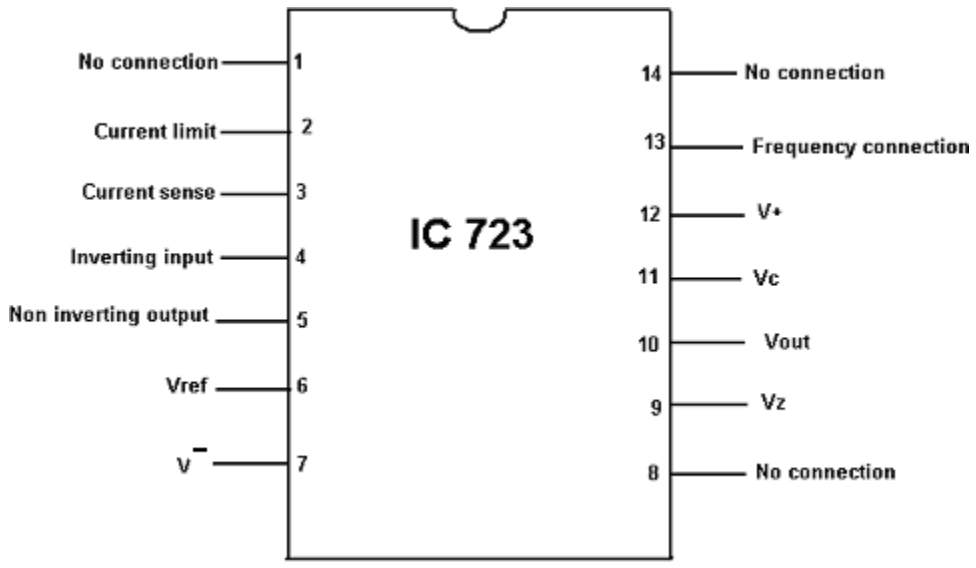
AIM:

To design and test the power supply voltage regulator using LM317 and LM723 ICs.

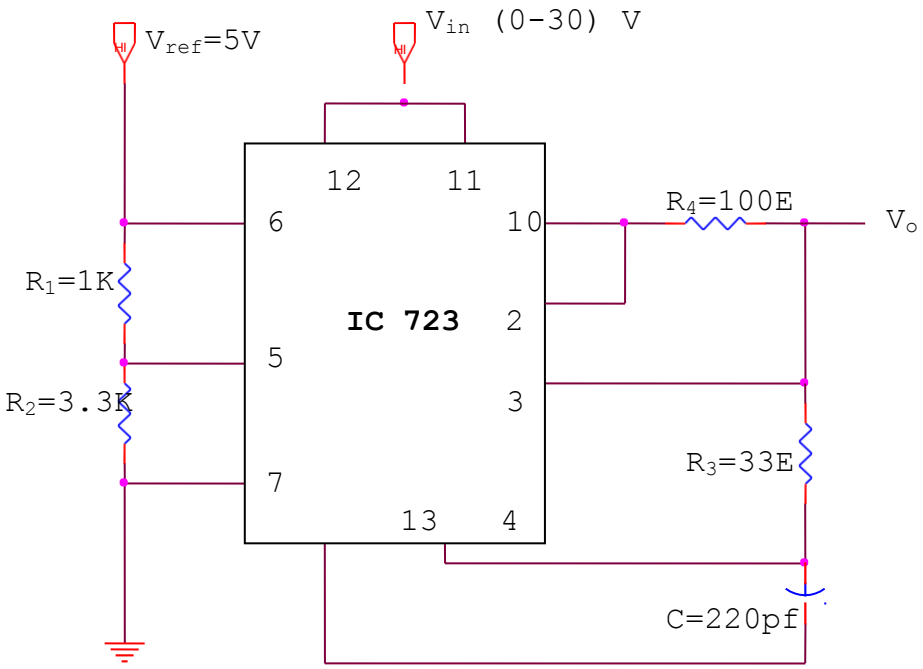
APPARATUS REQUIRED:

S.NO	COMPONENTS / EQUIPMENT	RANGE	QUANTITY
1.	LM317 and LM723	- - -	EACH 01
2.	RESISTORS	30Ω, 100Ω, 1KΩ, 3.3KΩ, 220Ω,	EACH 01
3.	DIGITAL TRAINER KIT	- - -	01
4.	ANALOG VOLTMETER	(0-10)V	01
5.	DUAL POWER SUPPLY	(0-30)V	01

PIN DIAGRAM:



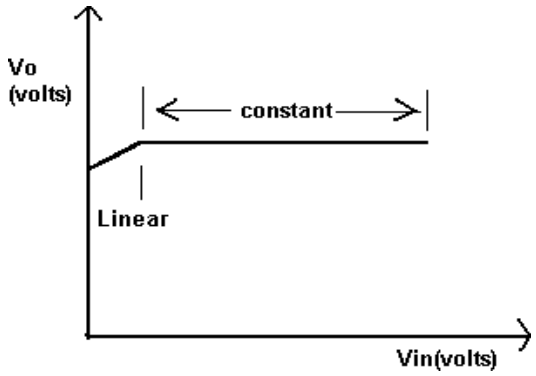
CIRCUIT DIAGRAM - (LM723):



TABULATION:

S.NO	INPUT VOLTAGE (Volts)	OUTPUT VOLTAGE (Volts)
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

MODEL GRAPH:



CHARACTERISTICS OF THE LM317HVK:

The LM317HVK will provide a regulated output current of upto 1.5A,Provided that if is not subjected to a power dissipation of more than about 15W.This means it should be electrically isolated from, and fastened to, a large heat sink such as the metal chassis of the power supply.

The LM317 requires a minimum "dropout" voltage of 3v across its input and output terminals or it will drop out of regulation. Thus the upper limit of Vo is 3V below the minimum input voltage from the unregulated supply.

It is good practice to connect bypass capacitors .This reduces the ripple voltage from the rectifier.

The LM317HVK protects itself against over heating, too much internal power dissipation and too much current. When the chip temperature reaches 175 degrees, the 317 shuts down. If the product of output current and input-to-output voltage exceeds 15 to 20W, or if currents greater than about 1.5A are required the LM317 also shuts down. When the overload condition isremoved the Operation is resumed. All these features are made possible by the remarkable internal circuitry of LM317.

Along with the simple 3 pin fixed regulators; a number of adjustable or programmable devices are available. Some devices also include features such as programmable current limiting. It is also possible to configure multiple regulators so that they track or follow each other.

DESIGN PROCEDURE: (IC723)

Given :

Vref = 5vVo = 4v

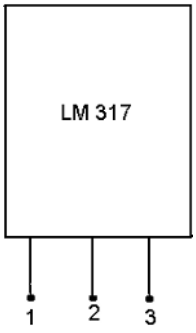
We know that

Vo = Vref (R2/ (R1+R2))$4R_1+4R_2=5R_2$

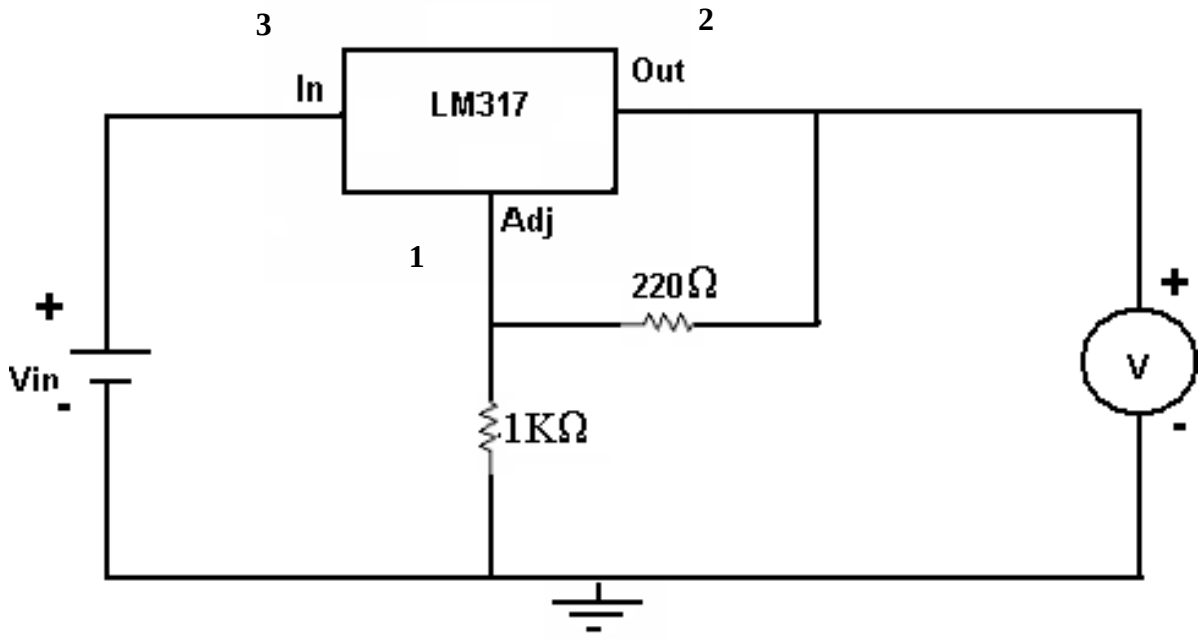
$R_2=4R_1$

Let $R_1=1k\Omega$ $R_2=4K\Omega \approx 3.3K\Omega$

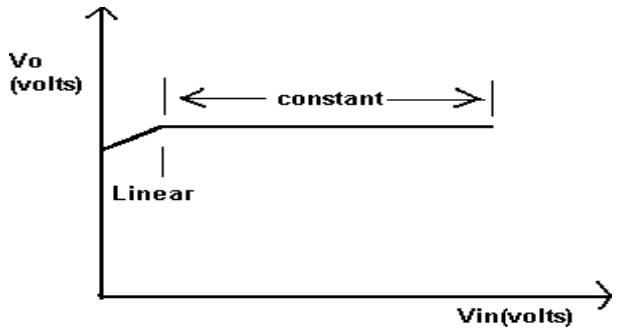
PIN DIAGRAM:



CIRCUIT DIAGRAM - (LM317):



MODEL GRAPH:



TABULATION:

S.NO	INPUT VOLTAGE (Volts)	OUTPUT VOLTAGE (Volts)

PROCEDURE:

1) Connections are made as per the circuit diagram.

2) The reference voltage of 5v is set and the input voltage is varied between (0-30) v

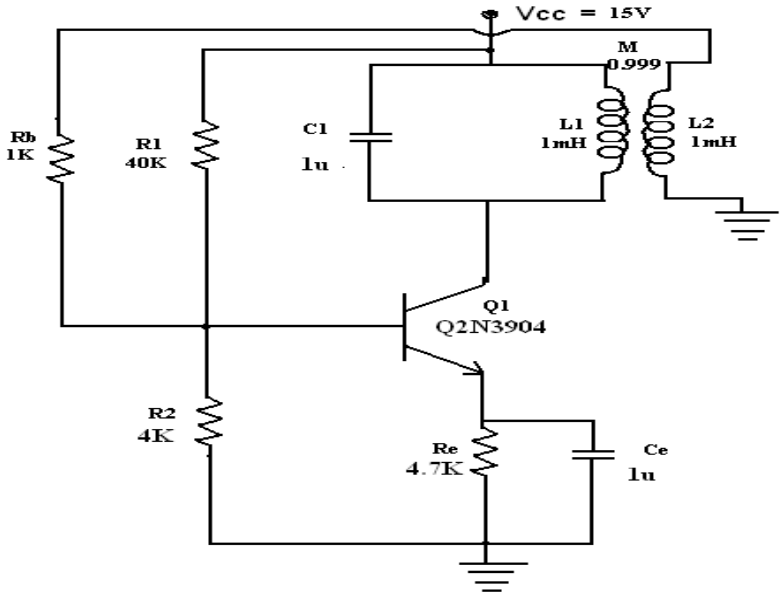
3) The corresponding output is taken using voltmeter.

4) The readings are tabulated and the graph is plotted.

RESULT:

The 723 & 317 IC voltage regulators are designed and the regulation of supply voltage was tested.

Circuit diagram:




```
vcc 1 0 dc 10v
R1 1 3 40k
R2 3 0 4k
R3 4 0 4.7k
R4 5 3 1k
C1 4 0 1u
C2 1 2 1u
L1 1 2 1mh
L2 0 5 1mh
K L1 L2 0.9999
Q1 2 3 4 Q2N2222
.tran 0 1ms
.op
.model Q2N2222 NPN(Is=3.108f Xti=3 Eg=1.11 Vaf=131.5 Bf=217.5 Ne=1.541
+ Ise=190.7f Ikf=1.296 Xtb=1.5 Br=6.18 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=14.57p Vjc=.75 Mjc=.3333 Fc=.5 Cje=26.08p Vje=.75
+ Mje=.3333 Tr=51.35n Tf=451p Itf=.1 Vtf=10 Xtf=2 Rb=10)
.probe
.end
```

EXP.NO:	TUNED COLLECTOR OSCILLATOR USING PSPICE

AIM: To simulate and analyze the Tuned Collector Oscillator using PSPICE.

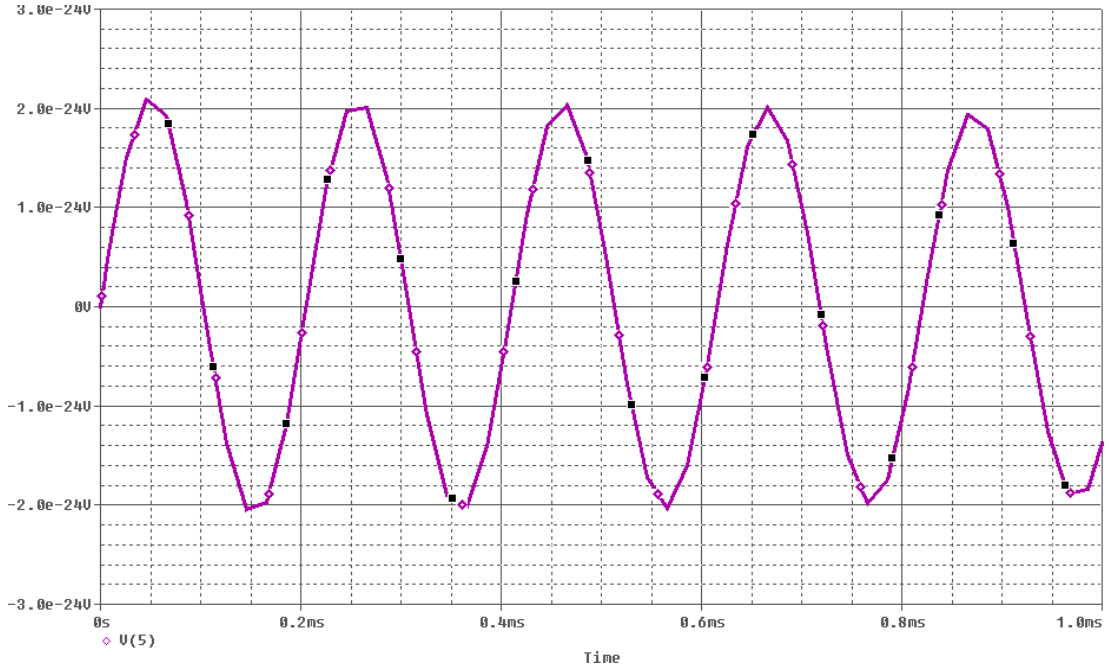
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

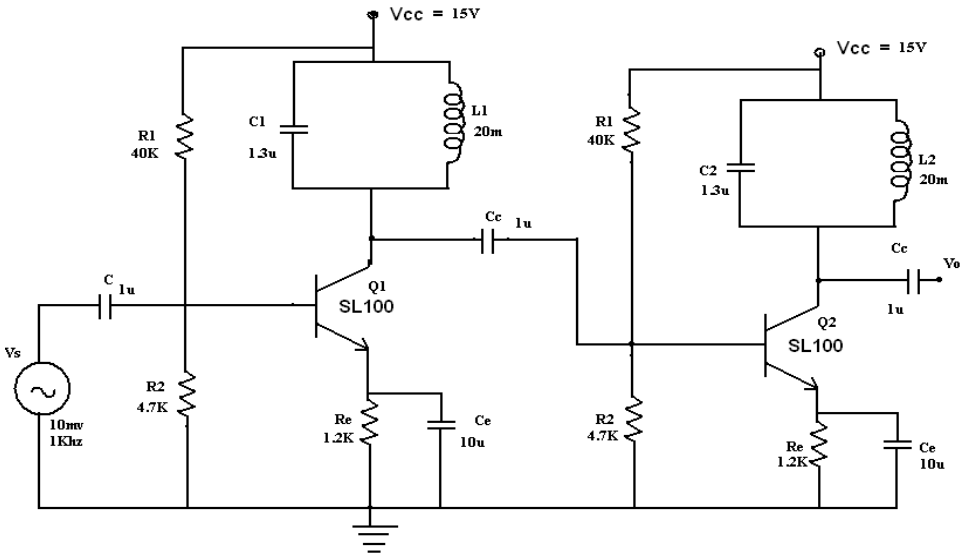
Output waveform:



Result

Tuned Collector Oscillator has been simulated and analyzed by Pspice.

Circuit diagram




```
.lib eval lib
Vcc 1 0 dc 15v
vin 2 0 ac 20mv
c1 2 3 1u
r1 1 3 40k
c5 1 4 1.3u
l1 1 4 20m
r3 3 0 4.7k
r4 5 0 1.2k
ce 5 0 10u
c2 4 6 10u
r5 6 0 4.7k
r6 1 6 40k
r7 8 0 1.2k
c3 8 0 10u
c4 7 9 1u
r8 9 0 1k
c6 1 7 1.3u
l2 1 7 20m
q1 4 3 5 Q2N2222
q2 7 6 8 Q2N2222
.ac dec 10 10 10khz
.op
.probe
.model Q2N2222 NPN(Is=3.108f Xti=3 Eg=1.11 Vaf=131.5 Bf=217.5 Ne=1.541
+ Ise=190.7f Ikf=1.296 Xtb=1.5 Br=6.18 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=14.57p Vjc=.75 Mjc=.3333 Fc=.5 Cje=26.08p Vje=.75
+ Mje=.3333 Tr=51.35n Tf=451p Itf=.1 Vtf=10 Xtf=2 Rb=10)
.end
```

EXP.NO:	STAGGER TUNED AMPLIFIER USING PSPICE

AIM: To simulate and analyze the Stagger Tuned Amplifier using PSPICE.

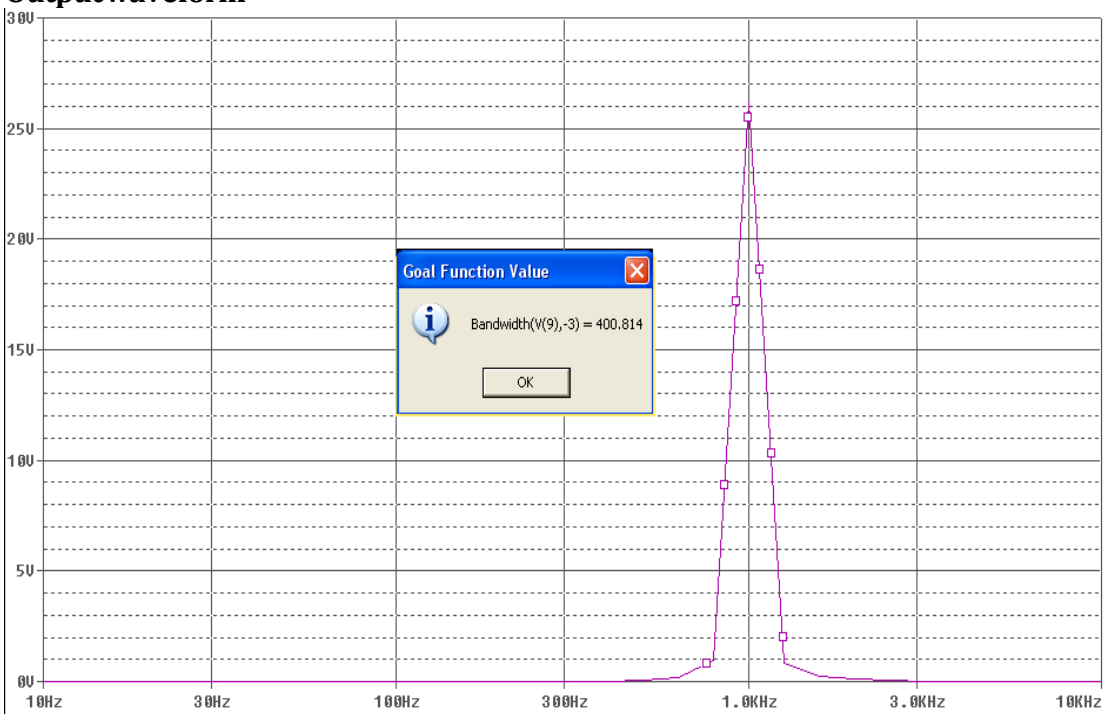
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

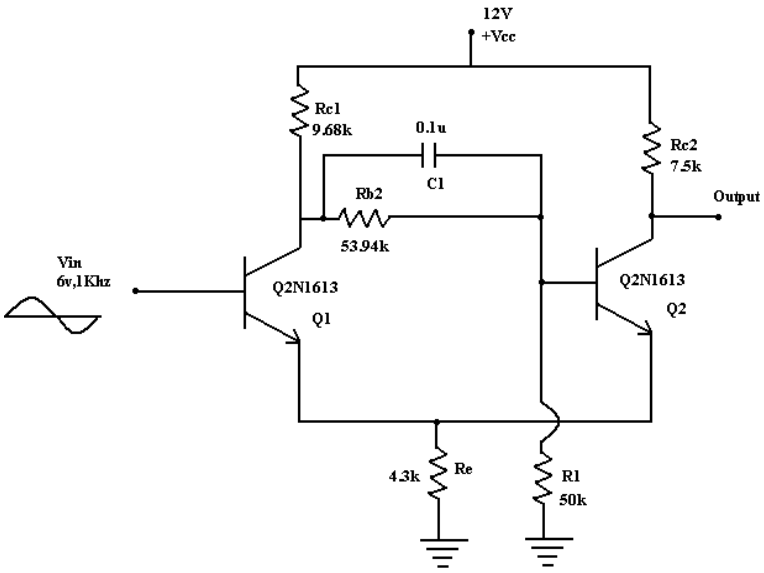
Output waveform



Result

Stagger Tuned Oscillator has been simulated and analyzed by Pspice.

Circuit diagram




```
.lib eval.lib
vcc 6 0 dc 12v
vin 1 0 sin(0 6v 1khz)
rc1 6 3 9.68k
rc2 6 5 7.5k
rb2 3 4 53.94k
r1 4 0 50k
re 2 0 4.3k
c1 3 4 0.1u
q1 3 1 2 Q2N2222
q2 5 4 2 Q2N2222
.op
.tran 0 5ms
.model Q2N2222 NPN(Is=3.108f Xti=3 Eg=1.11 Vaf=131.5 Bf=217.5 Ne=1.541
+ Ise=190.7f Ikf=1.296 Xtb=1.5 Br=6.18 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=14.57p Vjc=.75 Mjc=.3333 Fc=.5 Cje=26.08p Vje=.75
+ Mje=.3333 Tr=51.35n Tf=451p Itf=.1 Vtf=10 Xtf=2 Rb=10)
.probe
.end
```

EXP.NO:	SCHMITT TRIGGER USING PSPICE

AIM: To simulate and analyze the Schmitt Trigger using PSPICE.

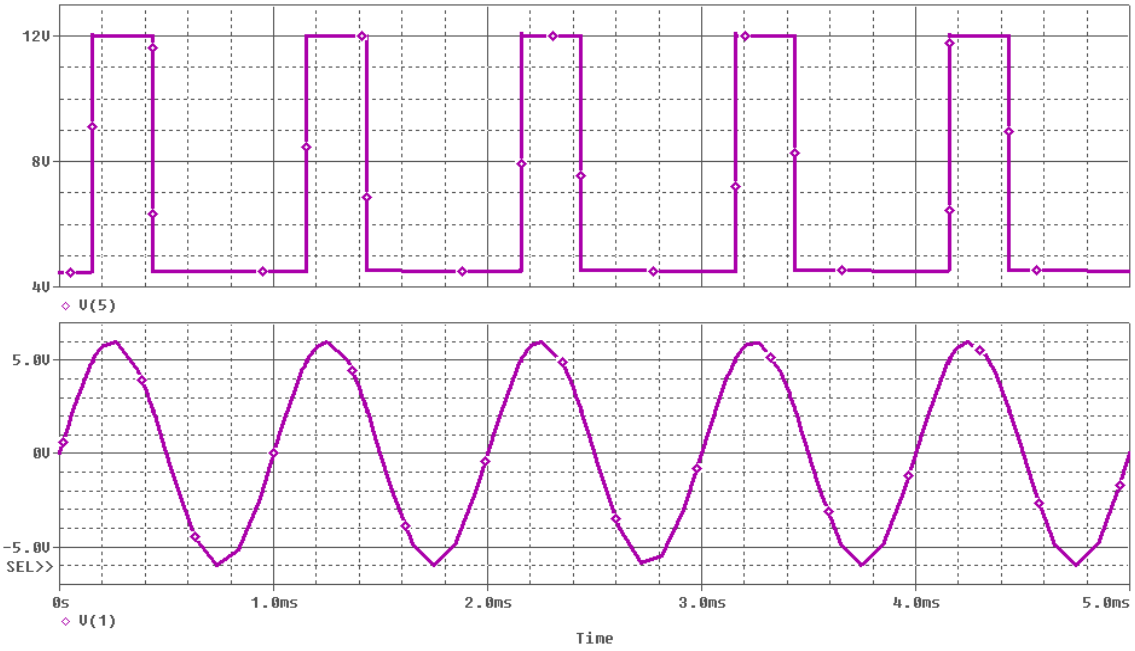
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

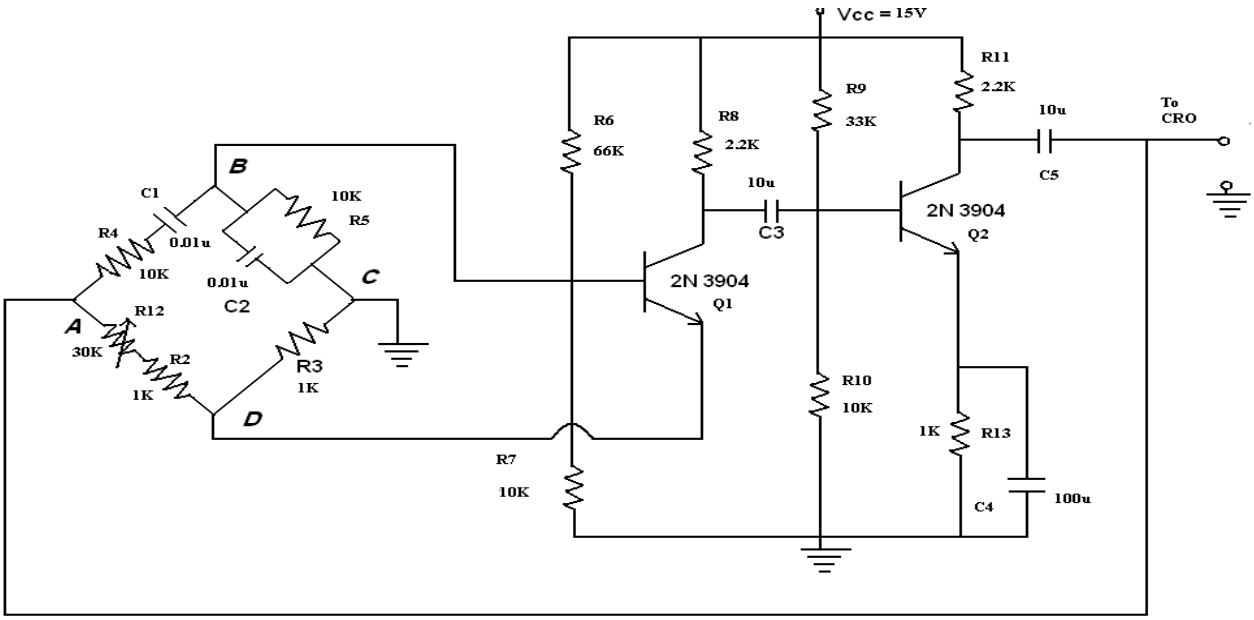
Output waveform



Result

Schmitt Trigger has been simulated and analyzed by Pspice.

Circuit diagram




```
.lib eval.lib
vcc 1 0 dc 15v
IS 6 0 PWL(0US 0MA 10US 0.1MA 20US 0.1MA 50US 0.1MA 10MS 0MA) r12 4 3 30k
r2 3 2 1k
r3 2 0 1k
r4 4 5 10k
c1 5 6 0.01u
r5 6 0 10k
c2 6 0 0.01u
r6 1 6 66k
r7 6 0 10k
r8 1 7 2.2k
c3 7 8 10u
r9 1 8 33k
r10 8 0 10k
r11 1 9 2.2k
r13 10 0 1k
c4 10 0 100u
c5 9 4 10u
q1 7 6 2 Q2N2222
q2 9 8 10 Q2N2222
.tran 0 1
.op
.model Q2N2222 NPN(Is=3.108f Xti=3 Eg=1.11 Vaf=131.5 Bf=217.5 Ne=1.541
+ Ise=190.7f Ikf=1.296 Xtb=1.5 Br=6.18 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=14.57p Vjc=.75 Mjc=.3333 Fc=.5 Cje=26.08p Vje=.75
+ Mje=.3333 Tr=51.35n Tf=451p Itf=.1 Vtf=10 Xtf=2 Rb=10)
.probe
.end
```

EXP.NO:	WEIN BRIDGE OSCILLATOR USING PSPICE

AIM: To simulate and analyze the Wein Bridge Oscillator using PSPICE.

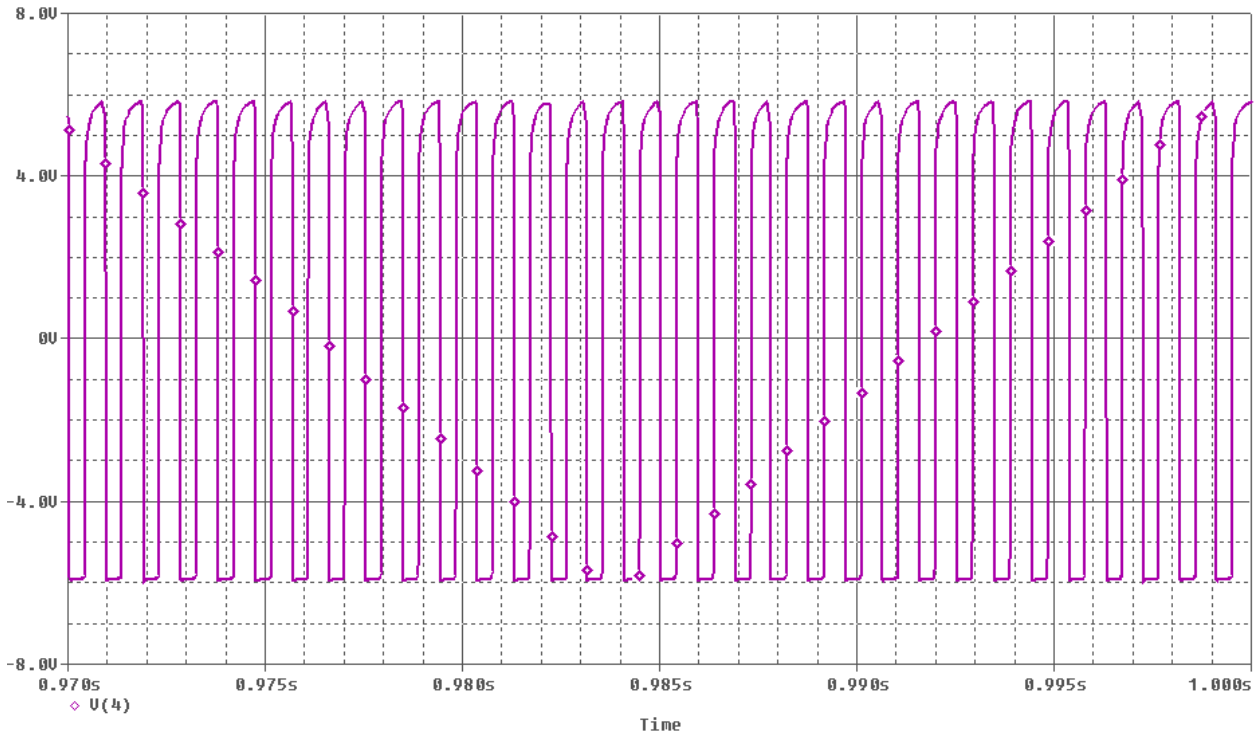
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

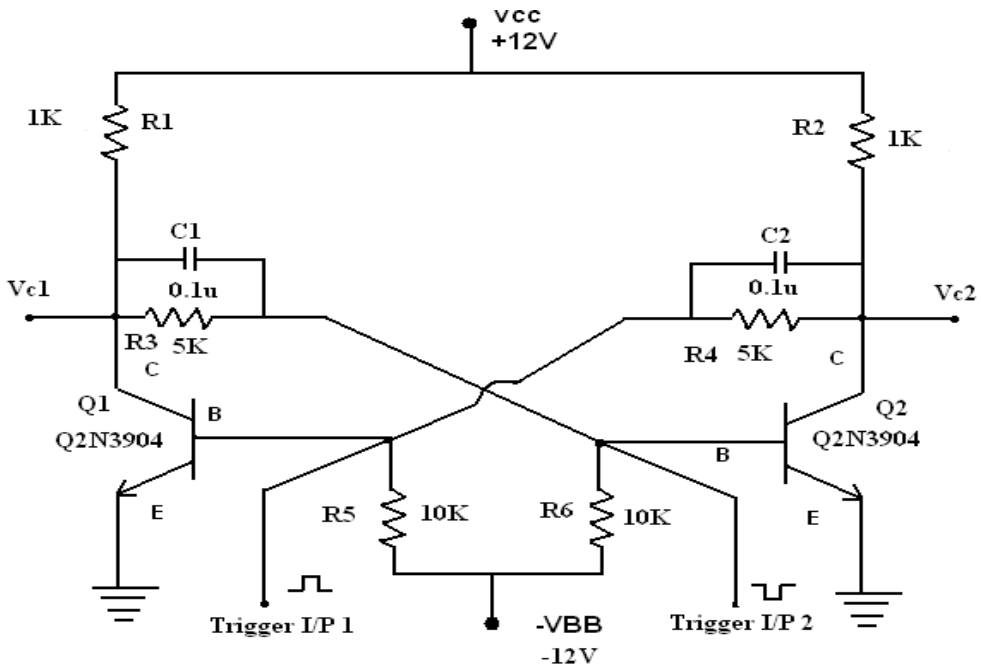
Output waveform



Result

Wein Bridge Oscillator has been simulated and analyzed by Pspice.

Circuit diagram




```
.lib eval.lib
vcc 1 0 dc 12v
vbb 0 2 dc 12v
v1 4 0 pulse (0 5v 0 0.001ms 0.001ms 0.5ms 1ms)
v2 6 0 pulse (5 0v 0 0.001ms 0.001ms 0.5ms 1ms)
r1 1 3 1k
r2 1 5 1k
r3 3 6 5k
r4 5 4 5k
r5 4 2 10k
r6 6 2 10k
c1 3 6 .01u
c2 5 4 .01u
q1 3 4 0 Q2N2222
q2 5 6 0 Q2N2222
.tran 0.1ms 10ms
.op
.probe
.model Q2N2222 NPN(Is=3.108f Xti=3 Eg=1.11 Vaf=131.5 Bf=217.5 Ne=1.541
+ Ise=190.7f Ikf=1.296 Xtb=1.5 Br=6.18 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=14.57p Vjc=.75 Mjc=.3333 Fc=.5 Cje=26.08p Vje=.75
+ Mje=.3333 Tr=51.35n Tf=451p Itf=.1 Vtf=10 Xtf=2 Rb=10)
.end
```

EXP.NO:	BISTABLE MULTIVIBRATOR USING PSPICE

AIM: To simulate and analyze the Bistable Multivibrator using PSPICE.

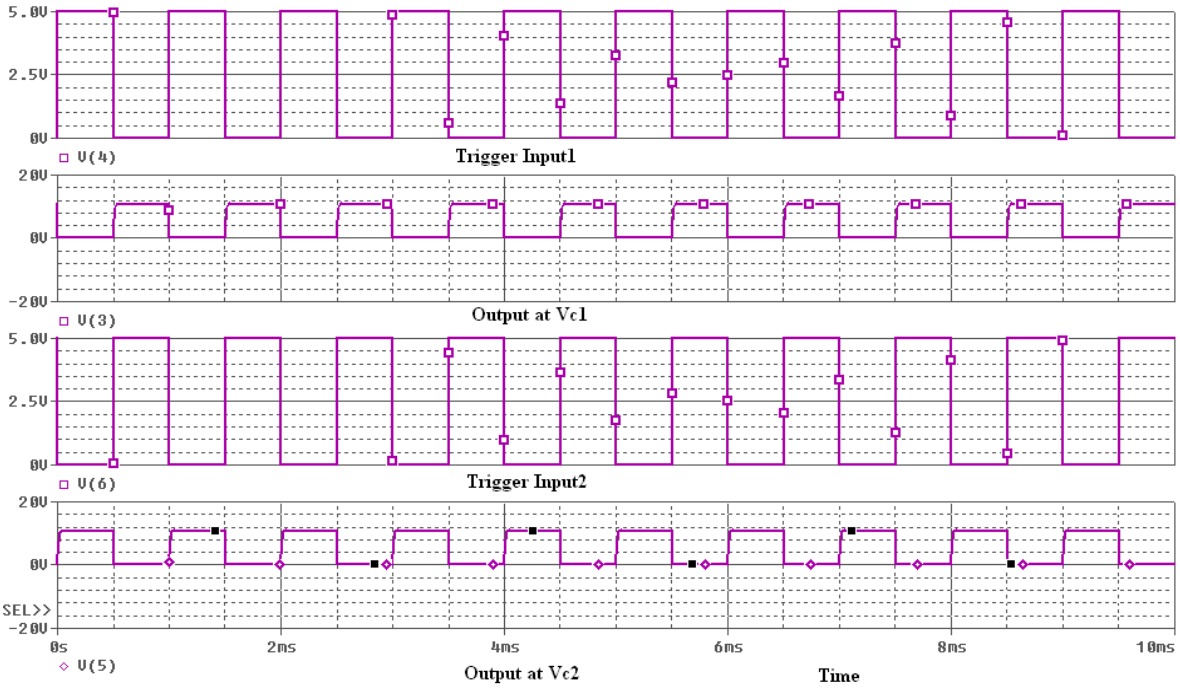
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

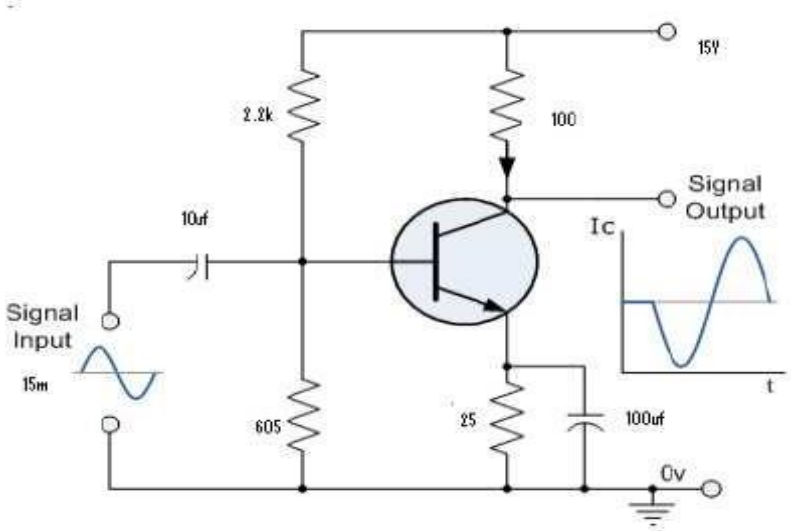
Output waveform



Result

Bistable Multivibrator has been simulated and analyzed by Pspice.

Circuit Diagram




```
VS 1 0 SIN (0 5MV 10KHZ)
VCC 5 0 15V
CB 1 2 10UF
CC 3 6 10UF
CE 4 0 100UF
R1 5 2 2.7K
R2 2 0 605
RC 5 3 100
RE 4 0 25
RL 6 0 47
Q1 3 2 4 SL100
.MODEL SL100 NPN
.TRAN 0.1MS 0.5MS
.PROBE
.END
```

EXP.NO:	CLASS A POWER AMPLIFIER USING PSPICE

AIM: To simulate and analyze the Class A Power Amplifier using PSPICE.

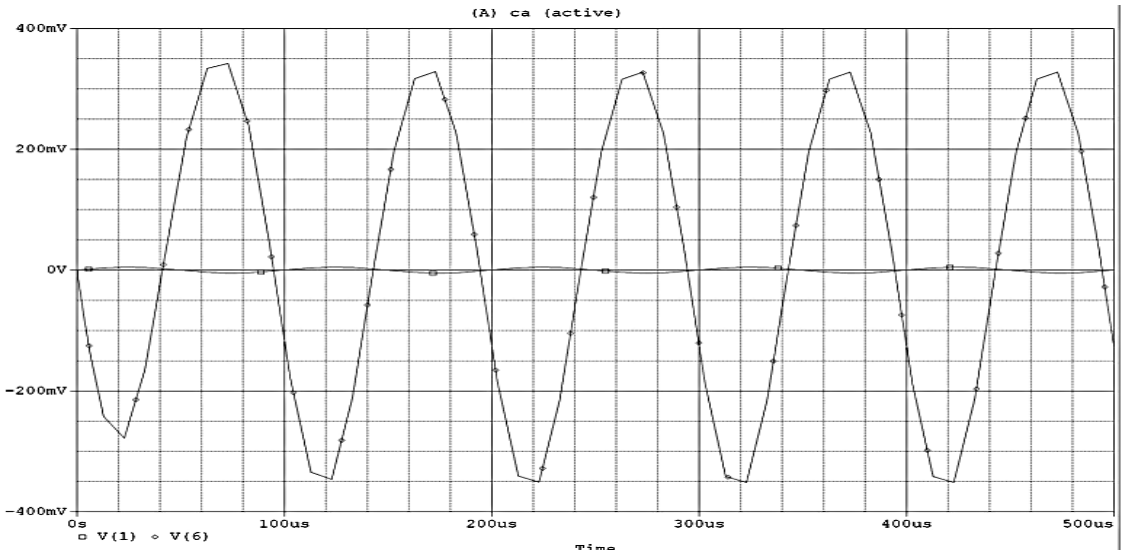
SOFTWARE REQUIRED:

OrCAD software.

PROCEDURE:

1. Open PSPICE A/D windows
2. Create a new circuit file
3. Enter the program representing the nodal interconnections of various components
4. Run the program
5. Observe the response through all the elements in the output file
6. Observe the required outputs (Graphs) in output window.

Output waveform



Result

Class A Power Amplifier has been simulated and analyzed by Pspice.

MADHA
Expertise | Empathy | Excellence
ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EC3492 DIGITAL SIGNAL PROCESSING

Semester - 04

LABORATORY MANUAL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

To excel in providing value based education in the field of Electronics and Communication Engineering, keeping in pace with the latest technical developments through commendable research, to raise the intellectual competence to match global standards and to make significant contributions to the society upholding the ethical standards.

Mission

- ✓ To deliver Quality Technical Education, with an equal emphasis on theoretical and practical aspects.
- ✓ To provide state of the art infrastructure for the students and faculty to upgrade their skills and knowledge.
- ✓ To create an open and conducive environment for faculty and students to carry out research and excel in their field of specialization.
- ✓ To focus especially on innovation and development of technologies that is sustainable and inclusive, and thus benefits all sections of the society.
- ✓ To establish a strong Industry Academic Collaboration for teaching and research, that could foster entrepreneurship and innovation in knowledge exchange.
- ✓ To produce quality Engineers who uphold and advance the integrity, honour and dignity of the engineering.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide the students with a strong foundation in the required sciences in order to pursue studies in Electronics and Communication Engineering.
2. To gain adequate knowledge to become good professional in electronic and communication engineering associated industries, higher education and research.
3. To develop attitude in lifelong learning, applying and adapting new ideas and technologies as their field evolves.
4. To prepare students to critically analyze existing literature in an area of specialization and ethically develop innovative and research oriented methodologies to solve the problems identified.
5. To inculcate in the students a professional and ethical attitude and an ability to visualize the engineering issues in a broader social context.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Design, develop and analyze electronic systems through application of relevant electronics, mathematics and engineering principles.

PSO2: Design, develop and analyze communication systems through application of fundamentals from communication principles, signal processing, and RF System Design & Electromagnetics.

PSO3: Adapt to emerging electronics and communication technologies and develop innovative solutions for existing and newer problems.

LIST OF EXPERIMENTS:

1. Generation of elementary Discrete-Time sequences
2. Linear and Circular convolutions
3. Auto correlation and Cross Correlation
4. Frequency Analysis using DFT
5. Design of FIR filters (LPF/HPF/BPF/BSF) and demonstrates the filtering operation
6. Design of Butterworth and Chebyshev IIR filters (LPF/HPF/BPF/BSF) and demonstrate the filtering operations
7. Study of architecture of Digital Signal Processor
8. Perform MAC operation using various addressing modes
9. Generation of various signals and random noise
10. Design and demonstration of FIR Filter for Low pass, High pass, Band pass and Band stop filtering
11. Design and demonstration of Butter worth and Chebyshev IIR Filters for Low pass, High pass, Band pass and Band stop filtering
12. Implement an Up-sampling and Down-sampling operation in DSP Processor

EXP. NO: 1 GENERATION OF ELEMENTARY DISCRETE-TIME SEQUENCES

DATE:

AIM:

To generate basic sequences such as Unit impulse, Unit step, Ramp, Exponential, Sine sequence & Cosine Sequence using MATLAB Programs.

APPARATUS REQUIRED:

1. Personal Computer
2. MATLAB Software

ALGORITHM:

1. Start the program.
2. Get the N point values from the user.
3. Assign the range of the time axis.
4. Give the title for the x axis and y axis for the program.
5. Plot the data sequence as a discrete values or continuous as per our requirements.

THEORY:

i. Unit Step Sequence:

The Unit Step Sequence is designed as unit step means that the amplitude of U(t) = 1

$U(n) = 1 \; ; n \geq 0$

$= 0 \; ; n < 0$

ii. Ramp Sequence:

The ramp sequence is defined as

$Ur(n) = n \; ; n \geq 0$

$= 0 \; ; n < 0$

iii .Exponential Sequence:

Exponential sequence is defined as

$g(n) = a^n \; ; n \geq 0$

$0 \; ; n < 0$

When the values of a > 1 the sequence grows exponentially and when the value is 0 < a < 1 the sequence

decay‟s exponentially. Note also that a < 0 the discrete time exponential signal takes attenuating signal.

iv. Cosine signal:

A discrete cosine signal is given by

$$X(n) = A \cos (\omega_o n + \phi)$$

Where ωo is the frequency and ϕ is the phase using bulers identity

we can write $A\cos (\omega_o n + \phi) = A/2\ e^{j\phi}\ e^{j\omega on} + A/2\ e^{-j\phi}\ e^{-j\omega on}$ ·Since $| e^{j\omega on}|^2 = 1^{st}$ the energy signal is infinite and the average power of the signal is 1

v. Sinusoidal signal:

A continuous time sinusoidal signal is given by

$$x(t) = A \sin (rt + \theta)$$

Where „A‟ is amplitude & r is the frequency in rad/sec and θ are the phase angle radians. The analog sinusoidal signal has the following properties

i) The signal is periodic satisfy the condition

$$x(t+T) = x(t)$$

ii) For different value of frequencies the continuous time sinusoidal signals are themselves different.

PROCEDURE:

- Start the MATLAB software and create new M-file.
- Type the program in the file.
- Save and compile the program.
- Give the input data.
- Observe the output waveform.
- Thus the graph is to be plotted.

PROGRAM:

Discrete Time Signals:

%Unit step sequence:

```
n=input ('Enter the n value');
t=0:1: n-1;
```

```
y=ones (1,n);
subplot (2,2,1);
stem (t,y);
xlabel ('n-->');
ylabel ('Amplitude');
title ('Unit step signal');
```

%Generation of sine sequence:

```
n=0:0.01:pi;
y=sin (2*pi*n);
subplot (2,2,2)
stem (n,y);
xlabel ('n->');
ylabel ('Amplitude');
title ('Sine sequence');
```

%Generation of cosine sequence:

```
n=0:0.01: pi;
y=cos (2*pi*n);
subplot (2,2,3);
stem (n,y);
xlabel ('x(n)');
ylabel ('Amplitude');
title ('Cosine sequence');
```

%Exponential sequence:

```
n=input ('Enter the length');
t=0: n;
a=input ('Enter the a value');
y=exp (a*t);
subplot (2,2,4);
stem (t,y);
xlabel ('Time');
ylabel('Amplitude');
title('Exponential sequence');
```

%Ramp signal:

```
clc;
l=input('Enter the length of the sequence');
n=0:l;
r=n;
stem(n,r);
disp(r);
title('Ramp sequence');
xlabel('Time');
ylabel('Amplitude');
```

%Impulse signal:

```
n=input ('Enter the n value');
t=0:1: n-1;
y= (1, zeros(1,n-1));
subplot (2,2,5);
stem (t,y);
xlabel ('n-->');
ylabel ('Amplitude');
title ('Impulse signal');
```

Continuous Time Signals:

%Unit step sequence:

```
n=input ('Enter the n value');
t=0:1: n-1;
y=ones (1,n);
subplot (2,2,1);
plot (t,y);
xlabel ('n-->');
ylabel ('Amplitude');
title ('Unit step signal');
```

%Generation of sine sequence:

```
t=0:0.01:pi;
y=sin (2*pi*t);
subplot (2,2,2)
```

```
plot (t,y);
xlabel ('n->');
ylabel ('Amplitude');
title ('Sine sequence');
```

%Generation of cosine sequence:

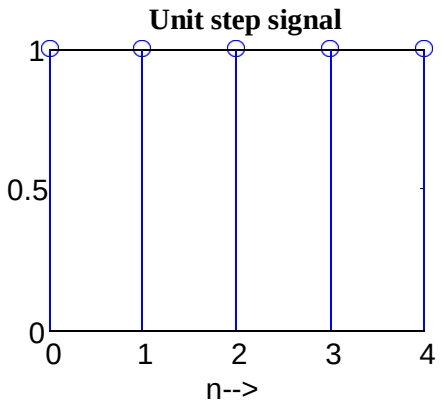
```
t=0:0.01: pi;
y=cos (2*pi*t);
subplot (2,2,3);
plot (t,y);
xlabel ('x(n)');
ylabel ('Amplitude');
title ('Cosine sequence');
```

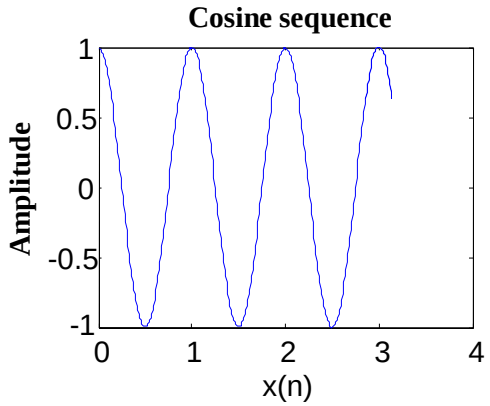
%Exponential sequence:

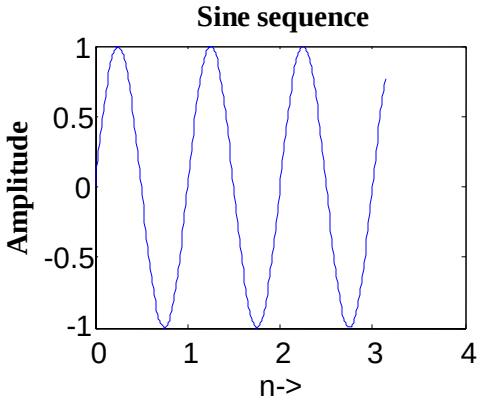
```
n=input ('Enter the length');
t=0: n;
a=input ('Enter the a value');
y=exp (a*t);
subplot (2,2,4);
plot (t,y);
xlabel ('Time');
ylabel('Amplitude');
title('Exponential sequence');
```

OUTPUT:

Enter the n value = 5



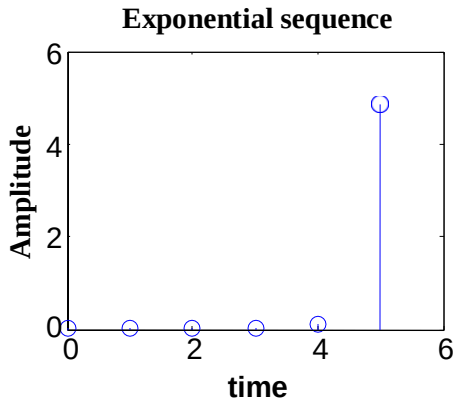


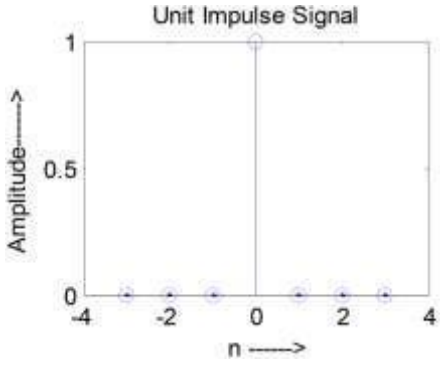


OUTPUT:

Enter the length 5

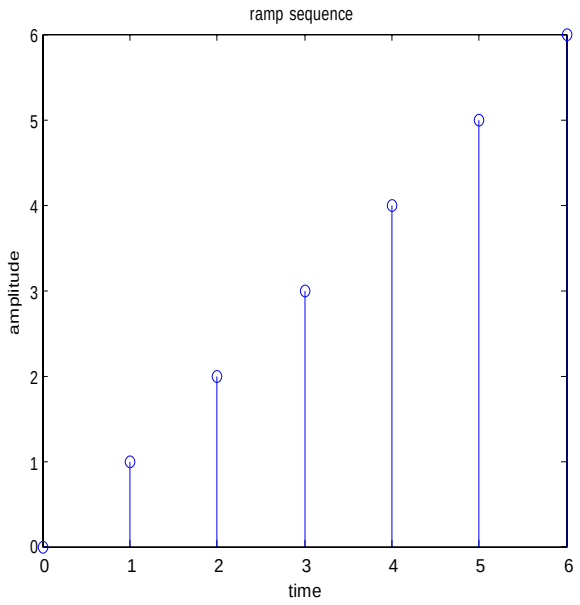
Enter the a value 5





OUTPUT:

Enter the length of the sequence 6



RESULT:

Thus the basic sequences of Unit impulse, Unit step, Ramp, Exponential and Sine Sequence & Cosine Sequence are generated using MATLAB Programs.

EXP. NO: 2 LINEAR CONVOLUTION AND CIRCULAR CONVOLUTION
DATE:

AIM:

To perform Linear Convolution and Circular Convolution of two discrete sequences using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get the input values from the user.
3. Assign the range of the time axis.
4. Give the title for the x axis and y axis for the program.
5. Plot the data sequence as a discrete values or continuous as per our requirements.

THEORY:

The linear convolution of two sequence x(n) of L no of samples and h(n) of M no of samples produce a result y(n) which contains N= L + M – 1 . If is a sequence which is periodic with N samples. Linear convolution can be used to find the response of a filter.

In the case of circular convolution if x(n) contains L no of samples and h(n) has N no of samples and that L > M, then we perform circular convolution between the two using N = Max (L, M) by adding L, M no of zero samples to the sequence h(n) . So that both sequence are periodic with N. Circular cannot be used to find the response of a linear filter without zero padding.

PROCEDURE:

- Start the MATLAB software and create new M-file.
- Type the program in the file.
- Save & compile the program.
- Give the input data.
- Observe the output waveform.
- Thus the graph is to be plotted.

PROGRAM:

%Linear convolution:

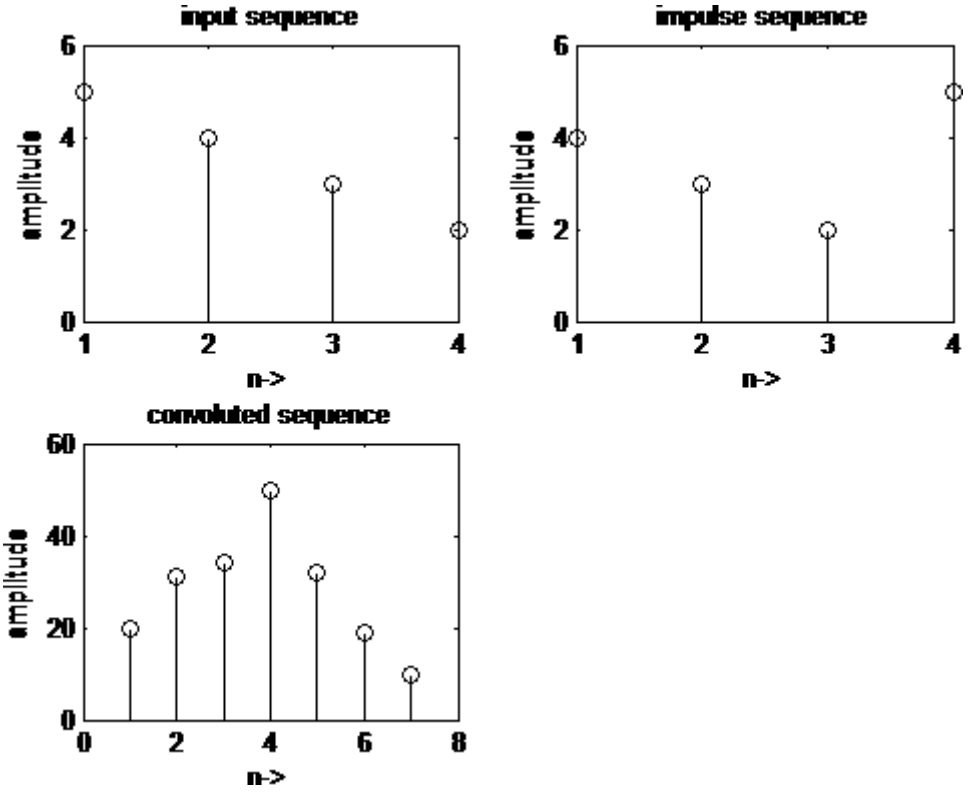
```
clc;
x=input ('Enter the input sequence');
n1=length(x);
subplot (2,2,1);
Stem(x);
title ( ' Input sequence');
xlabel ('n->');
ylabel ('Amplitude');
h=input ('Enter the impulse sequence');
n2=length (h);
subplot (2,2,2);
stem (h);
title (' Impulse sequence');
xlabel ('n->');
ylabel ('Amplitude');
y=conv(x, h);
n=1:n1+n2-1;
subplot (2,2,3);
stem (n,y);
Disp(y);
title ('Convoluted sequence');
xlabel ('n->');
ylabel ('Amplitude');
```

OUTPUT

Enter the input sequence [5 4 3 2]

Enter the impulse sequence [4 3 2 5]

Output is 20 31 34 50 32 19 10



PROGRAM

%Circular convolution

```
clc;
X1=input ('Enter the first sample');
X2=input ('Enter the second sample');
l1=length(x1);
l2=length(x2);
x1s=fft(x1);
Disp (x1s);
x2s=fft(x2);
Disp (x2s);
x3s=x1s.*x2s;
y=ifft (x3s);
Disp(y);
Subplot (3, 1, 1);
n=0:1:l1-1;
Stem (n, x1);
Title ('First input sample');
```

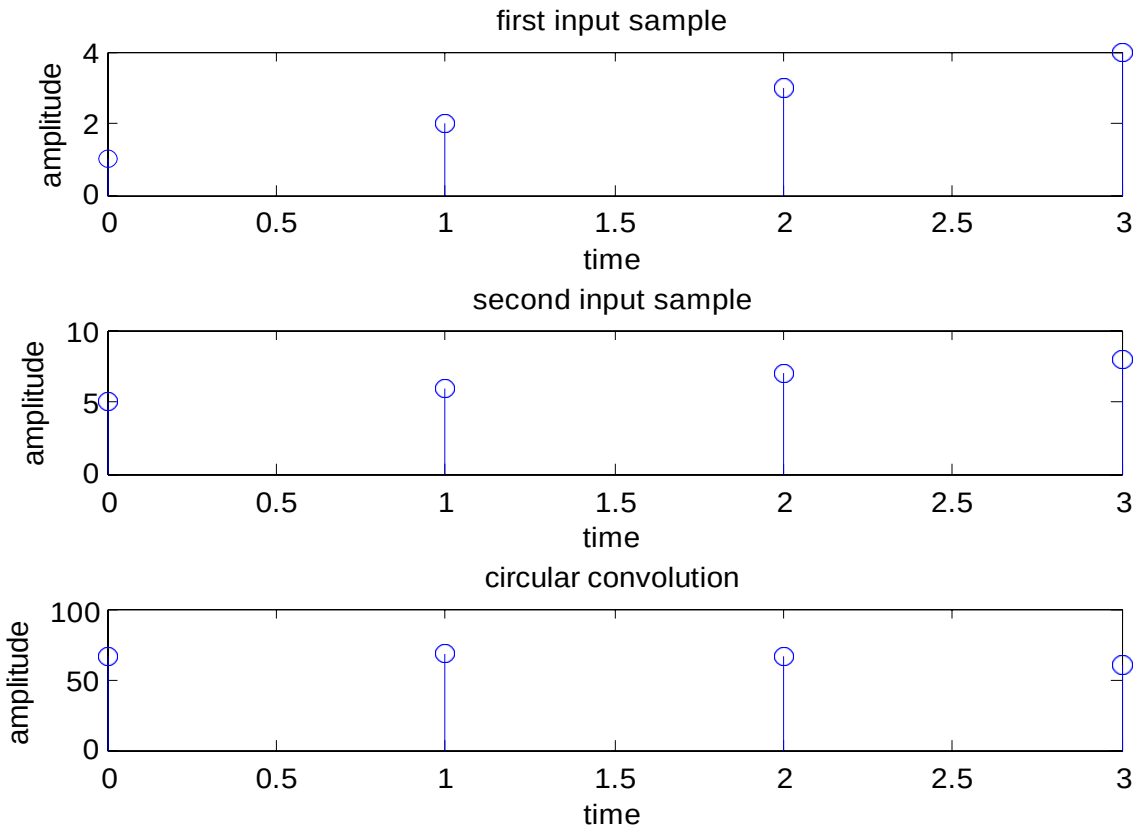
```
Xlabel ('Time');
Ylabel ('Amplitude');
Subplot (3, 1, 2);
n=0:1:l2-1;
Stem (n, x2);
Title (Second input sample');
Xlabel ('Time');
Ylabel ('Amplitude');
Subplot (3, 1, 3);
Stem (n, y);
Title ('Circular convolution');
Xlabel ('Time');
Ylabel ('Amplitude');
```

OUTPUT:

Enter the first sample [1 2 3 4]

Enter the second sample [5 6 7 8]

Output is 66 68 66 60



RESULT:

Thus the linear convolution and circular convolution programs were performed and verified using MATLAB.

EXP.NO:3 AUTO CORRELATION AND CROSS CORRELATION

DATE:

AIM:

To implement auto-correlation and cross-correlation functions using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get the input values from the user.
3. Assign the range of the time axis.
4. Give the title for the x axis and y axis for the program.
5. Plot the data sequence as a discrete values or continuous as per our requirements.

PROCEDURE:

- Start the MATLAB software and create new M-file.
- Type the program in the file.
- Save and compile the program.
- Give the input data.
- Observe the output waveform.
- Thus the graph is to be plotted.

PROGRAM:

% Cross correlation

```
clc;
clear all; close all;
x=input ('Enter the first sequence');
h=input('Enter the second sequence');
y=xcorr(x,h);
subplot(3,1,1);
stem(x);
xlabel('x');
ylabel('Amplitude');
title('x sequence');
subplot(3,1,2);
```

```
stem(h);
xlabel('h');
ylabel('Amplitude');
title('h sequence');
subplot(3,1,3);
stem(y);
xlabel('y');
ylabel('amplitude');
title(' y sequence');
```

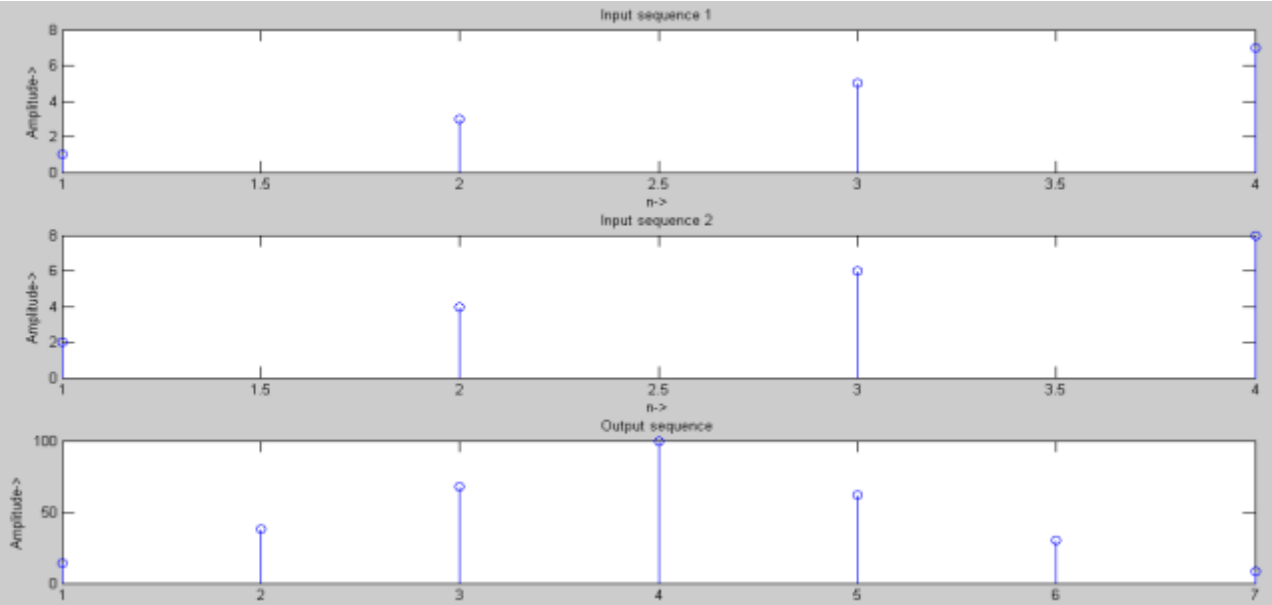
OUTPUT (Cross Correlation):

Enter the first sequence [1 2 3 4]

Enter the second sequence [4 3 2 1]

The resultant signal is

Y= 1.0 4.0 10.0 20.0 25.0 24.00 16.00



%Auto correlation

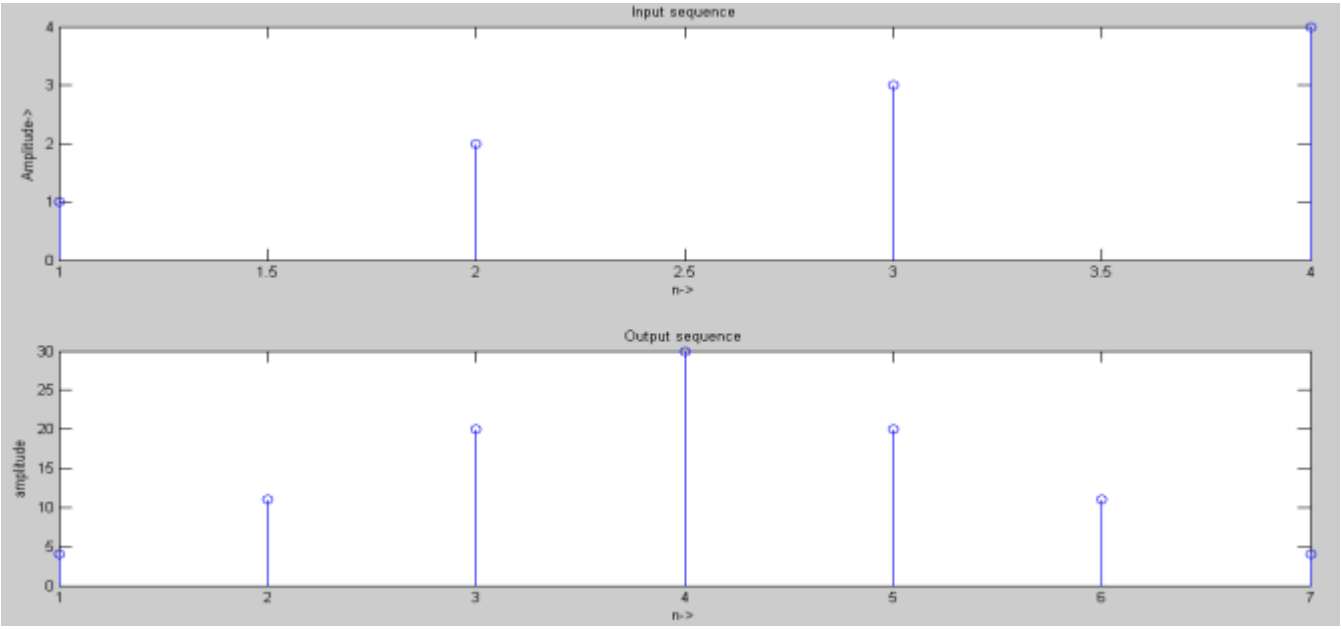
```
clc;
clear all; close all;
x=input ('Enter the t sequence');
y=xcorr(x,x);
```

```
subplot(2,1,1);
stem(x);
xlabel('x');
ylabel('Amplitude');
title('x sequence');
subplot(2,1,2);
stem(y);
xlabel('y');
ylabel('Amplitude');
title(' y sequence');
disp('The resultant signal is');
```

OUTPUT (Auto Correlation):

Enter the sequence = [1 2 3 4]

The resultant signal is Y= 4 11 20 30 20 11 4



RESULT:

Thus the auto-correlation and cross-correlation functions are generated using MATLAB.

EXP. NO: 4 FREQUENCY ANALYSIS USING DFT
DATE:

AIM

To write a program for frequency analysis using DFT.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get the input values from the user.
3. Assign the range of the time axis.
4. Give the title for the x axis and y axis for the program.
5. Plot the data sequence as a discrete values or continuous as per our requirements.

THEORY:

DFT is used for analyzing discrete-time finite-duration signals in the frequency domain

Let x[n]be a finite-duration sequence of length N such that x[n]=0, 0<n<N-1outside.

The DFT pair of is:

$$X[k] = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-1} x(n) \, W_N^{kn}, 0 \le k \le N-1 \end{cases}$$

$$X[k] = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-1} x(n) \, W_N^{kn}, 0 \le k \le N-1 \end{cases}$$

PROGRAM:

```
N=input('type length of DFT= ');
T=input('type sampling period= ');
freq=input('type the sinusoidal freq= ');
k=0:N-1;
f=sin(2*pi*freq*1/T*k);
F=fft(f);
stem(k,abs(F));
```

```
grid on;
xlabel('k');
ylabel('X(k)');
```

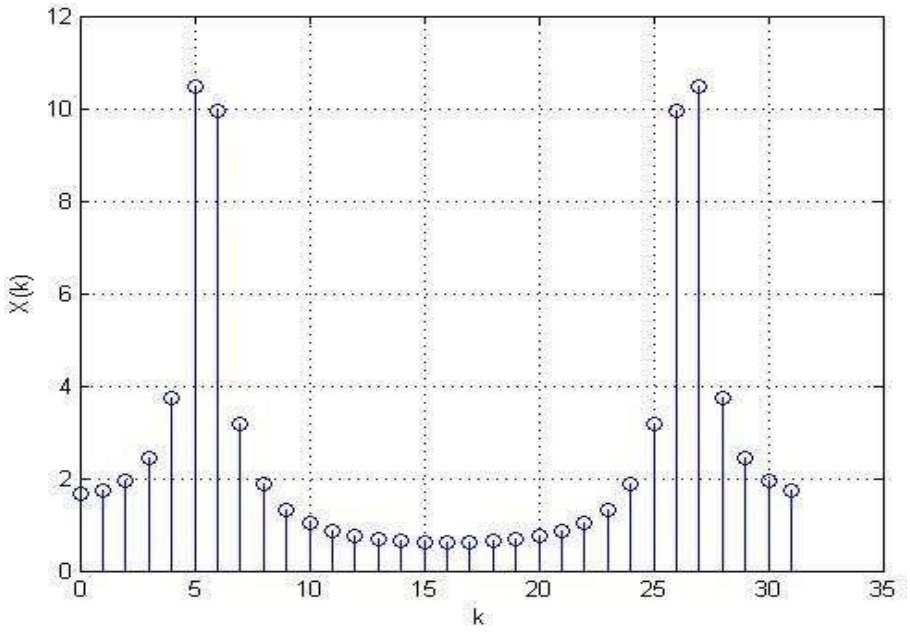
INPUT:

type length of DFT=32

type sampling period=64

type the sinusoidal freq=11

OUTPUT:



RESULT

Thus the Frequency Analysis of the signal using DFT is obtained using MATLAB.

EXP.NO: 5A DESIGN OF FIR FILTER USING RECTANGULAR WINDOW
DATE:

AIM

To design the FIR low pass, High pass, Band pass and Band stop filters using rectangular window and find out the response of the filter by using MATLAB.

APPARATUS REQUIRED

Personal Computer

MATLAB Software

ALGORITHM

1. Start the program.
2. Get pass band ripple and stop band ripple.
3. Get Pass band and stop band frequency.
4. Get sampling frequency.
5. Calculate the order of the filter.
6. Find the window Coefficient.
7. Plot the magnitude and phase response.

THEORY

The rectangular window (sometimes known as the **boxcar** or **Dirichlet window**) is the simplest window, equivalent to replacing all but *N* values of a data sequence by zeros, making it appear as though the waveform suddenly turns on and off:

W(n)= 1.

Other windows are designed to moderate these sudden changes because discontinuities have undesirable effects on the discrete-time Fourier transform (DTFT) and/or the algorithms that produce samples of the DTFT.

The rectangular window is the 1st order *B*-spline window as well as the 0th power cosine window.

PROCEDURE

- Start the MATLAB software and create new M-file.
- Type the program in the file.

- Save & compile the program.
- Give the input data.
- Observe the output waveform.
- Thus the graph is to be plotted.

PROGRAM:

```
clc;
clear all;
close all;
rp=input('Pass band ripple=');
rs=input('Stop band ripple=');
fs=input('Stop band frequency in rad/sec=');
fp=input('Pass band frequency in rad/sec=');
f=input('Sampling frequency in rad/sec=');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem)
n1=n+1;
if(rem(n,2)~=0);
n1=n;
n=n-1;
end
y=boxcar(n1);
                                        %LOW PASS FILTER
b=fir1(n,wp,'low',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(o/pi,m);
ylabel('Gain in db >');
xlabel('Normalized frequency --->');
```

```
title('LOW PASS FILTER')
                                        %HIGH PASS FILTER
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(o/pi,m);
ylabel('Gain in db >');
xlabel('Normalized frequency --->');
title('HIGH PASS FILTER')
                                            %BAND PASS FILTER
wn=[wp,ws];
b=fir1(n,wp,'band',y);
[h,o]=freqz(b,1,256);
subplot(2,2,3);

plot(o/pi,m);
ylabel('Gain in db --->');
xlabel('Normalized frequency --->');
title('BAND PASS FILTER')
                                        %BAND STOP FILTER
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(o/pi,m);
ylabel('Gain in db >');
xlabel('Normalized frequency --->');
title('BAND STOP FILTER')
```

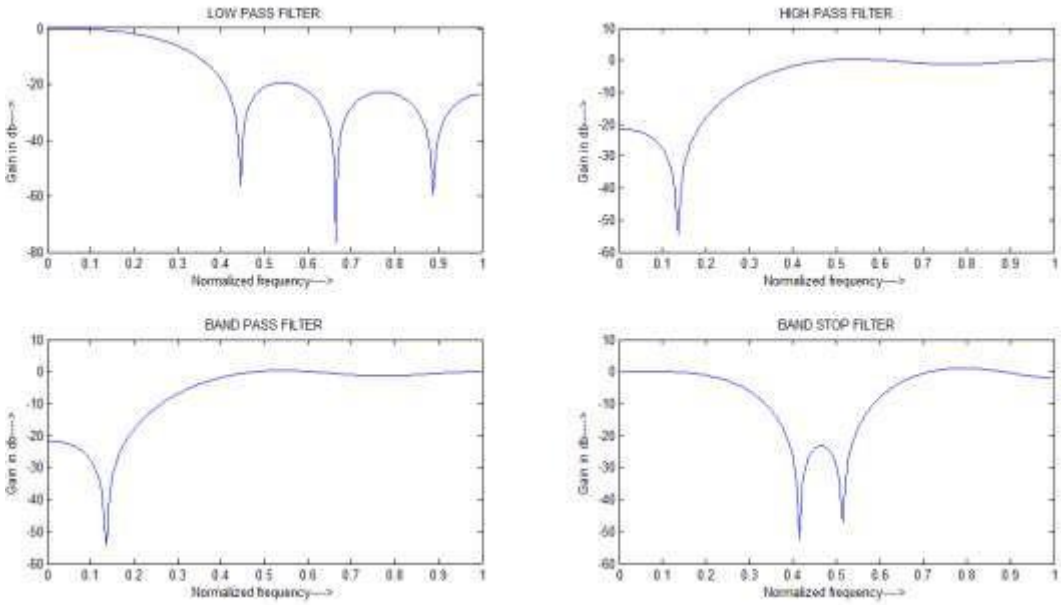
OUTPUT

```
Pass band ripple=0.03
Stop band ripple=0.04
```

Stop band frequency in rad/sec=1200

Pass band frequency in rad/sec=600

Sampling frequency in rad/sec=4000



RESULT

Thus the design of FIR low pass, high pass, band pass and band stop filters were obtained for Rectangular Window using MATLAB.

EXP.NO: 5B DESIGN OF FIR FILTER USING HAMMING WINDOW
DATE:

AIM:

To design the FIR low pass, High pass, Band pass and Band stop filters for Hamming window by using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get pass band ripple and stop band ripple.
3. Get Pass band and stop band frequency.
4. Get sampling frequency.
5. Calculate the order of the filter.
6. Find the window Coefficient.
7. Plot the magnitude and phase response.

THEORY:

HAMMING WINDOW:

The filters response can be obtained by

$$W_H(n) = 0.54+0.46\cos 2\pi n/N-1 \quad ; \quad -(N-1)/2 \le n \le N-1/2$$

$$= 0 \quad ; \quad 0$$

The frequency response is

$$W_H(e^{jw}) = 0.54\sin wn/2/\sin w/2+0.23\sin(wn/2-\pi n/n-1)/\sin(w/2-\pi/n-1)+0.23\sin(wn/2+n/n-1)/\sin(w/2-\pi/n-1)$$

PROCEDURE:

- Start the MATLAB software and create new M-file.
- Type the program in the file.
- Save & compile the program.
- Give the input data.
- Observe the output waveform.

- Thus the graph is to be plotted.

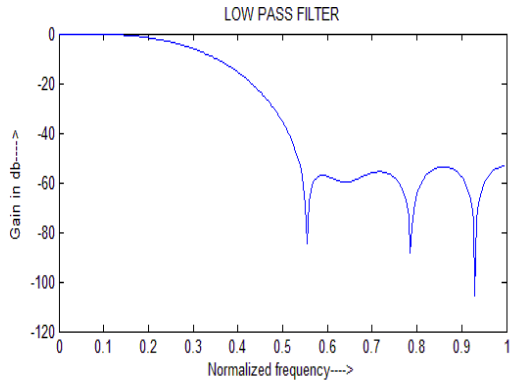
PROGRAM:

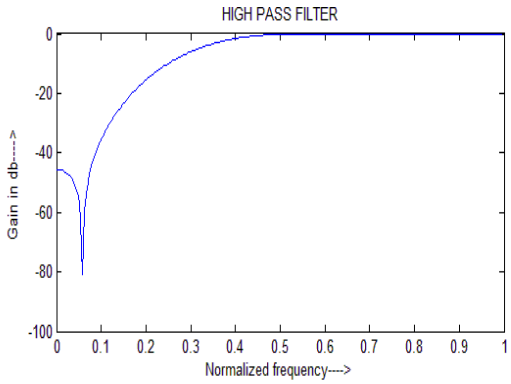
```
clc;
clear all;
close all;
rp=input('Pass band ripple=');
rs=input('Stop band ripple=');
fs=input('Stop band frequency in rad/sec=');
fp=input('Pass band frequency in rad/sec=');

f=input('Sampling frequency in rad/sec=');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem)
n1=n+1;
if(rem(n,2)~=0);
n1=n;
n=n-1;
end
y=hamming(n1);
                                %LOW PASS FILTER
b=fir1(n,wp,'low',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(o/pi,m);
ylabel('Gain in db >');
xlabel('Normalized frequency --->');
title('LOW PASS FILTER')
                                %HIGH PASS FILTER
b=fir1(n,wp,'high',y);
```

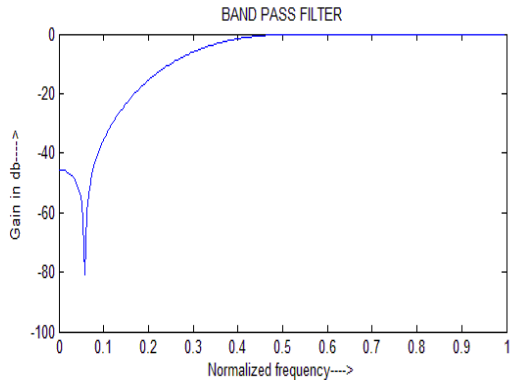
```
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(o/pi,m);
ylabel('Gain in db --->');
xlabel('Normalized frequency --->');
title('HIGH PASS FILTER')
                                %BAND PASS FILTER
wn=[wp,ws];
b=fir1(n,wp,'band',y);
[h,o]=freqz(b,1,256);
subplot(2,2,3);
plot(o/pi,m);
ylabel('Gain in db --->');
xlabel('Normalized frequency --->');
title('BAND PASS FILTER')

                                %BAND STOP FILTER
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);

m=20*log10(abs(h));
subplot(2,2,4);
plot(o/pi,m);
ylabel('Gain in db --->');
xlabel('Normalized frequency --->');
title('BAND STOP FILTER')
```

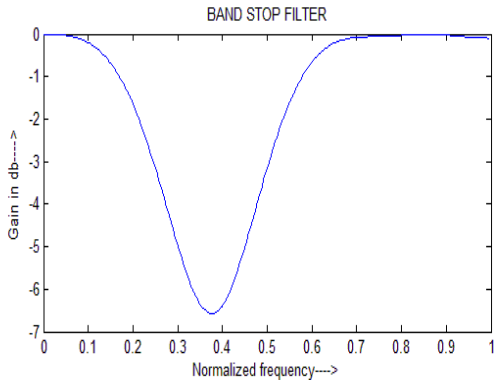
OUTPUT:

Pass band ripple	=0.03
Stop band ripple	=0.04
Stop band frequency in rad/sec	=1200
Pass band frequency in rad/sec	=600
Sampling frequency in rad/sec	=4000









RESULT:

Thus the design of FIR low pass, high pass, bands pass and band stop filters for Hamming Window was obtained using MATLAB.

EXP. NO: 6A IIR FILTER USING BUTTERWORTH FILTER APPROXIMATION DATE:

AIM:

To design the IIR low pass, High pass, Band pass and Band stop filters using Butterworth approximation and find out the response of the filter by using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get pass band ripple and stop band ripple.
3. Get Pass band and stop band frequency.
4. Calculate the order of the filter.
5. Plot the band pass and band stop filter.

THEORY:

Butterworth Filter:

- ✓ The magnitude response of butter worth filter decreases maintain as the frequency increases 0 to $\propto$.
- ✓ The filter transition band is more in butter worth filter.
- ✓ The poles on the butter worth filter are lie on a circle.
- ✓ For the same specification the no. of files create disadvantage.

PROCEDURE:

- ➢ Start the MATLAB software and create new M-file.
- ➢ Type the program in the file.
- ➢ Save & compile the program.
- ➢ Give the input data.
- ➢ Observe the output waveform.
- ➢ Thus the graph is to be plotted.

PROGRAM:

```
clear all;
clc;
close all;
format long
rp=input('enter the pass band ripple');
rs=input('enter the stop band ripple');
wp=input('enter the pass band frequency ');
ws=input('enter the stop band frequency ');
[n1,w1]=buttord(wp,ws,rp,rs);
[num,den]=butter(n1,w1);
[num1,den1]=butter(n,w1,'stop');

[g,w]=freqz(num,den);
[g1,w1]=freqz(num1,den1);
m=20*log10(abs(g));
m1=20*log10(abs(g1));
title('Gain response of Butterworth filter');
Subplot(2,1,1);
Plot(w/pi,m);
Ylabel('gain in db?>');
Xlabel('omega/pi');
title('Bandpass filter');
Subplot(2,1,2);
Plot(w/pi,m1);
Ylabel('gain in db?>');
Xlabel('omega/pi');
title('Bandreject filter');
```

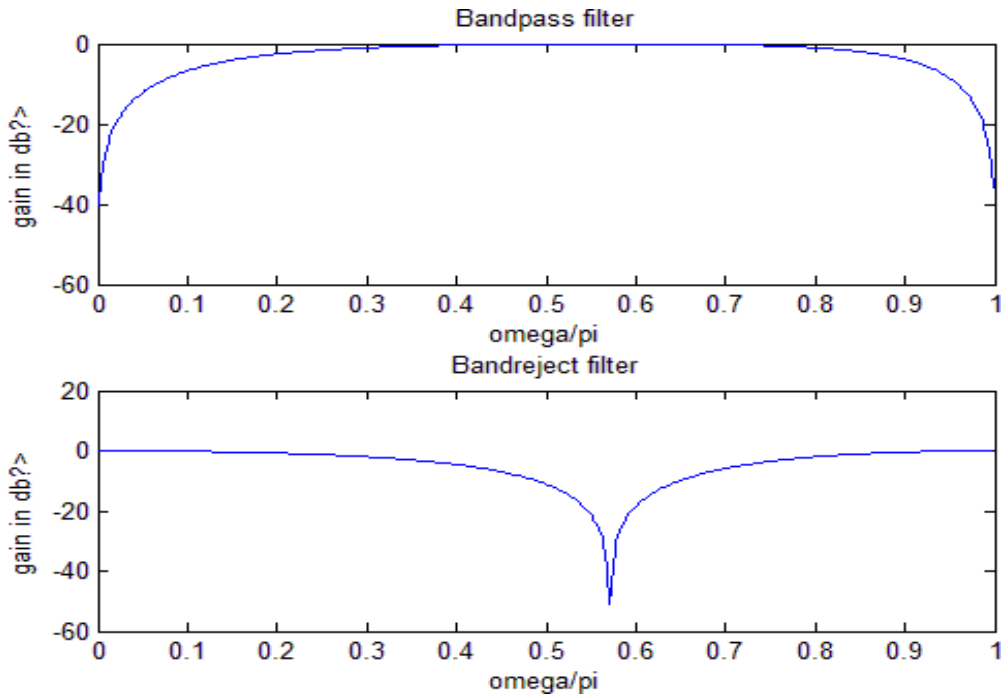
OUTPUT:

enter the pass band ripple 10

enter the stop band ripple 30

enter the pass band frequency [0.2 0.8]

enter the stop band frequency [0.4 0.7]



RESULT:

Thus the design of IIR band pass and band stop filters using Butterworth method was executed using MATLAB.

EXP. NO: 6B IIR FILTER USING CHEBYSHEV FILTER APPROXIMATION
DATE:

AIM:

To design the IIR Band pass and Band stop filters using Chebyshev approximation and find out the response of the filter by using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get pass band ripple and stop band ripple.
3. Get Pass band and stop band frequency.
4. Calculate the order of the filter.
5. Plot the band pass and band stop filter.

THEORY:

Chebyshev Filter:

- ✓ The magnitude response of the chebyshev filter exhibits ripple in the pass band or stop band according to the type.
- ✓ The transition band is less as compare to butter worth filter.
- ✓ The poles on a chebyshev filter are lie on ellipse.
- ✓ The order of chebyshev filter is less than that of butter worth.

PROCEDURE:

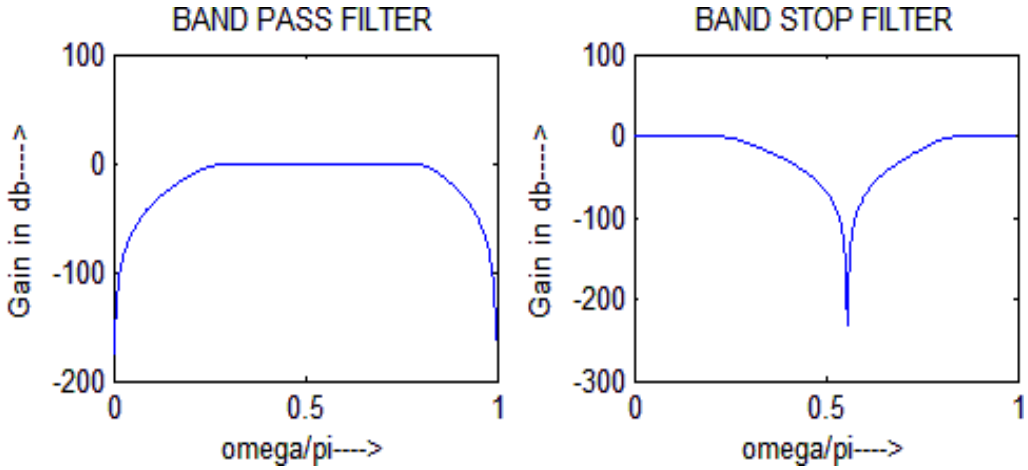
- ➢ Start the MATLAB software and create new M-file.
- ➢ Type the program in the file.
- ➢ Save & compile the program.
- ➢ Give the input data.
- ➢ Observe the output waveform.
- ➢ Thus the graph is to be plotted.

PROGRAM:

```
clear all;
clc;
close all;
format long
rp=input('enter the pass band ripple');
rs=input('enter the stop band ripple');
wp=input('enter the pass band frequency ');
ws=input('enter the stop band frequency ');
[n1,w1]=cheb1ord(wp,ws,rp,rs);
[num,den]=cheby1(n1,rp,w1);
[num1,den1]=cheby1(n1,rs,w1,'stop');
[g,w]=freqz(num,den);
[g1,w1]=freqz(num1,den1);
m=20*log10(abs(g));
m1=20*log10(abs(g1));
title('Gain response of chebyshev filter');
Subplot(2,1,1);
plot(w/pi,m);
Ylabel('gain in db?>');
Xlabel('omega/pi');
title('Bandpass filter');
Subplot(2,1,2);
plot(w/pi,m1);
Ylabel('gain in db?>');
Xlabel('omega/pi');
title('Bandreject filter');
```

OUTPUT:

```
enter the pass band ripple      1
enter the stop band ripple      2
enter the pass band frequency          [0.3 0.8]
enter the stop band frequency          [0.2 0.9]
```



RESULT:

Thus the MATLAB program for IIR filter using Chebyshev approximation for the sequence was performed.

EXP.NO:7 STUDY OF ARCHITECTURE OF DIGITAL SIGNAL PROCESSOR

DATE:

AIM:

To study the various architecture of digital signal processor TMS320C50 Kit.

Introduction:

The hardware experiments in the DSP lab are carried out on the Texas Instruments TMS320C6713 DSP Starter Kit (DSK), based on the TMS320C6713 floating point DSP running at 225 MHz The basic clock cycle instruction time is 1/(225 MHz)= 4.44 nanoseconds. During each clock cycle, up to eight instructions can be carried out in parallel, achieving up to 8×225 = 1800 million instructions per second (MIPS).

The DSK board includes a 16MB SDRAM memory and a 512KB Flash ROM. It has an on-board 16-bit audio stereo codec (the Texas Instruments AIC23B) that serves both as an A/D and a D/A converter. There are four 3.5 mm audio jacks for microphone and stereo line input, and speaker and head-phone outputs. The AIC23 codec can be programmed to sample audio inputs at the following sampling rates: f_s = 8, 16, 24, 32, 44.1, 48, 96 kHz

The ADC part of the codec is implemented as a multi-bit third-order noise-shaping delta-sigma converter that allows a variety of oversampling ratios that can realize the above choices of fs. The corresponding oversampling decimation filters act as anti-aliasing pre-filters that limit the spectrum of the input analog signals effectively to the Nyquist interval [−fs/2, fs/2]. The DAC part is similarly implemented as a multi-bit second-order noise-shaping delta-sigma converter whose oversampling interpolation filters act as almost ideal reconstruction filters with the Nyquist interval as their pass band.

The DSK also has four user-programmable DIP switches and four LEDs that can be used to control and monitor programs running on the DSP. All features of the DSK are managed by the Code Composer Studio (CCS). The CCS is a complete integrated development environment (IDE) that includes an optimizing C/C++ compiler, assembler, linker, debugger, and program loader.

The CCS communicates with the DSK via a USB connection to a PC. In addition to facilitating all programming aspects of the C6713 DSP, the CCS can also read signals stored On the DSP memory, or the SDRAM, and plot them the following block diagram depicts the overall operations involved in all of the hardware experiments in the DSP lab. Processing is interrupt-driven at the sampling rate fs, as explained below.

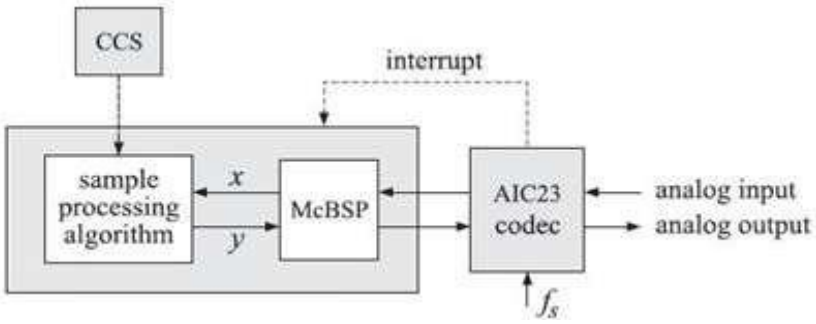


Fig: Interrupt processing

The AIC23 codec is configured (through CCS) to operate at one of the above sampling rates fs. Each collected sample is converted to a 16-bit two‟s complement **short** data type in integer C). The codec actually samples the audio input in stereo, that is, it collects two samples for the left and right channels.

Architecture

The ‟54x DSPs use an advanced, modified Harvard architecture that maximizes processing power by maintaining one program memory bus and three data memory buses. These processors also provide an arithmetic logic unit (ALU) that has a high degree of parallelism, application-specific hardware logic, on-chip memory, and additional on-chip peripherals.

These DSP families also provide a highly specialized instruction set, which is the basis of the operational flexibility and speed of these DSPs. Separate program and data spaces allow simultaneous access to program instructions and data, providing the high degree of parallelism. Two reads and one write operation can be performed in a single cycle.

Instructions with parallel store and application-specific instructions can fully utilize this architecture. In addition, data can be transferred between data and program spaces. Such parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. Also included are the control mechanisms to manage interrupts, repeated operations, and function calls.

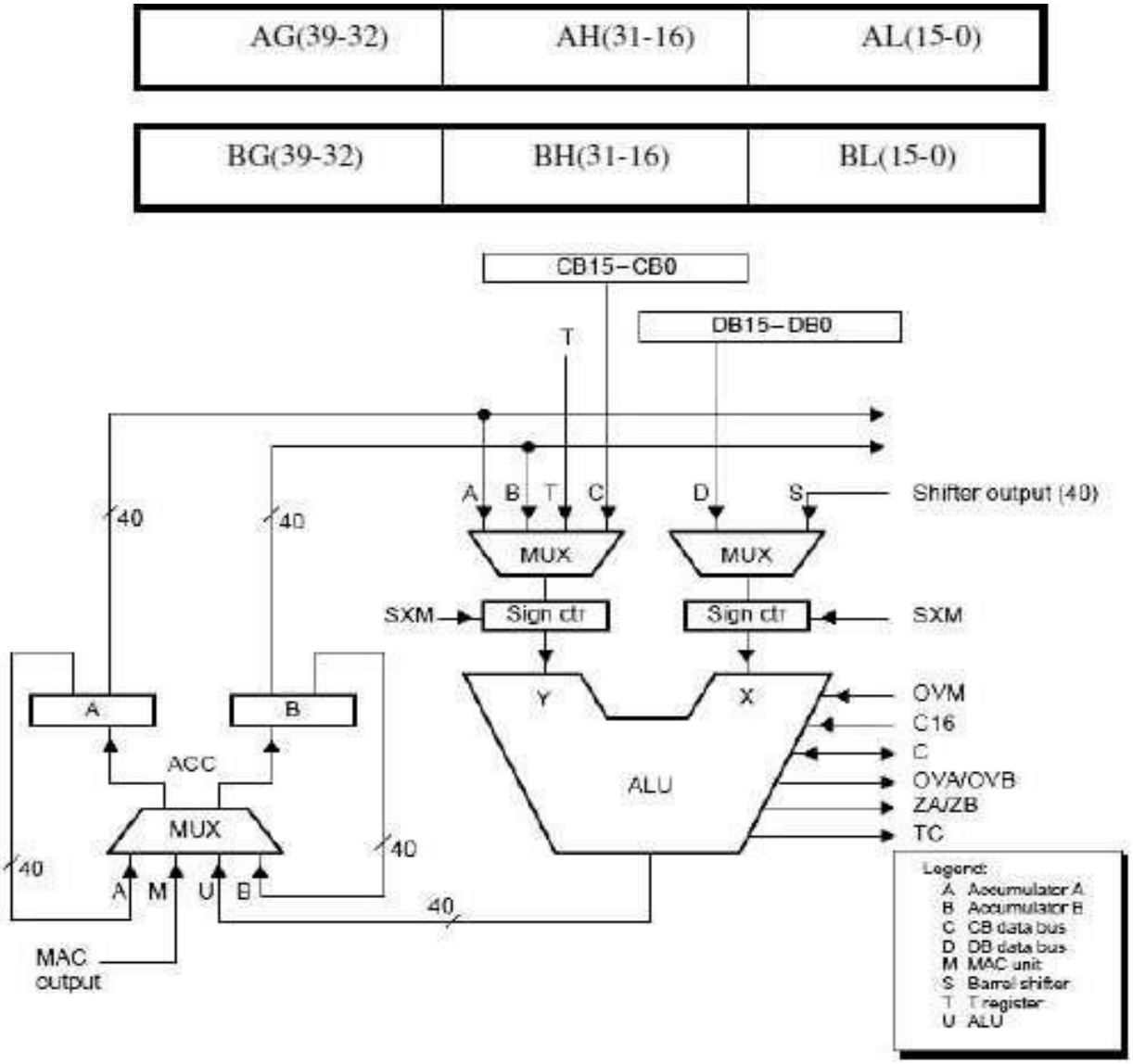


Figure Functional diagram of the central processing unit of the TMS320C54xx processors.

1. Central Processing Unit (CPU)

The CPU of the "54x devices contains:

- A 40-bit arithmetic logic unit (ALU)
- Two 40-bit accumulators
- A barrel shifter
- A 17-bit multiplier/adder
- A compare, select, and store unit (CSSU)

2 Arithmetic Logic Unit (ALU)

The “54x devices perform 2s-complement arithmetic using a 40-bit ALU and two 40-bit accumulators (ACCA and ACCB). The ALU also can perform Boolean operations. The ALU can function as two 16-bit ALUs and perform two 16-bit operations simultaneously when the C16 bit in status register 1 (ST1) is set.

3 Accumulators

The accumulators, ACCA and ACCB, store the output from the ALU or the multiplier / adder block; the accumulators can also provide a second input to the ALU or the multiplier / adder. The bits in each accumulator are grouped as follows:

- Guard bits (bits 32–39)
- A high-order word (bits 16–31)
- A low-order word (bits 0–15)

Instructions are provided for storing the guard bits, the high-order and the low-order accumulator words in data memory, and for manipulating 32-bit accumulator words in or out of data memory. Also, any of the accumulators can be used as temporary storage for the other.

4 Barrel Shifter

The “54x“s barrel shifter has a 40-bit input connected to the accumulator or data memory (CB, DB) and a 40-bit output connected to the ALU or data memory (EB). The barrel shifter produces a left shift of 0 to 31 bits and a right shift of 0 to 16 bits on the input data. The shift requirements are defined in the shift-count field (ASM) of ST1 or defined in the temporary register (TREG), which is designated as a shift-count register.

This shifter and the exponent detector normalize the values in an accumulator in a single cycle. The least significant bits (LSBs) of the output are filled with 0s and the most significant bits (MSBs) can be either zero-filled or sign-extended, depending on the state of the sign-extended mode bit (SXM) of ST1. Additional shift capabilities enable the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations

5 Multiplier/Adder

The multiplier / adder performs 17 □ □ 17bit 2s-complement multiplication with a 40-bit accumulation in a single instruction cycle. The multiplier / adder block consists of several elements: a multiplier, adder, signed/unsigned input control, fractional control, a zero detector, a rounder (2s-complement), overflow/saturation logic, and TREG. The multiplier has two inputs: one input is selected from the TREG, a data-memory operand, or an accumulator; the other is selected from the program memory, the data memory, an accumulator, or an immediate value.

The fast on-chip multiplier allows the ‘‘54x to perform operations such as convolution, correlation, and filtering efficiently. In addition, the multiplier and ALU together execute multiply/accumulate (MAC) computations and ALU operations in parallel in a single instruction cycle. This function is used in determining the Euclid distance, and in implementing symmetrical and least mean square (LMS) filters, which are required for complex DSP algorithms.

6 Compare, Select, and Store Unit (CSSU)

The compare, select, and store unit (CSSU) performs maximum comparisons between the accumulator‘‘s high and low words, allows the test/control (TC) flag bit of status register 0 (ST0) and the transition (TRN) register to keep their transition histories, and selects the larger word in the accumulator to be stored in data memory. The CSSU also accelerates Viterbi-type butterfly computation with optimized on-chip hardware.

7 Program Control

Program control is provided by several hardware and software mechanisms: The program controller decodes instructions, manages the pipeline, stores the status of operations, and decodes conditional operations. Some of the hardware elements included in the program controller are the program counter, the status and control register, the stack, and the address-generation logic. Some of the software mechanisms used for program control include branches, calls, conditional instructions, a repeat instruction, reset, and interrupts.

The ‘‘54x supports both the use of hardware and software interrupts for program control. Interrupt service routines are vectored through a reloadable interrupt vector table. Interrupts can be globally enabled/disabled and can be individually masked through the interrupt mask register (IMR). Pending interrupts are indicated in the interrupt flag register (IFR). For detailed information on the structure of the interrupt vector table, the IMR and the IFR, see the device-specific data sheets.

8 Status Registers (ST0, ST1)

The status registers, ST0 and ST1, contain the status of the various conditions and modes for the ‘‘54x devices. ST0 contains the flags (OV, C, and TC) produced by arithmetic operations and bit manipulations in addition to the data page pointer (DP) and the auxiliary register pointer (ARP) fields. ST1 contains the various modes and instructions that the processor operates on and executes.

9 Auxiliary Registers (AR0–AR7)

The eight 16-bit auxiliary registers (AR0–AR7) can be accessed by the central arithmetic logic unit (CALU) and modified by the auxiliary register arithmetic units (ARAUs). The primary function of the

auxiliary registers is generating 16-bit addresses for data space. However, these registers also can act as general-purpose registers or counters.

10 Temporary Register (TREG)

The TREG is used to hold one of the multiplicands for multiply and multiply/accumulate instructions. It can hold a dynamic (execution-time programmable) shift count for instructions with a shift operation such as ADD, LD, and SUB. It also can hold a dynamic bit address for the BITT instruction. The EXP instruction stores the exponent value computed into the TREG, while the NORM instruction uses the TREG value to normalize the number. For ACS operation of Viterbi decoding, TREG holds branch metrics used by the DADST and DSADT instructions.

11 Transition Register (TRN)

The TRN is a 16-bit register that is used to hold the transition decision for the path to new metrics to perform the Viterbi algorithm. The CMPS (compare, select, max, and store) instruction updates the contents of the TRN based on the comparison between the accumulator high word and the accumulator low word.

12 Stack-Pointer Register (SP)

The SP is a 16-bit register that contains the address at the top of the system stack. The SP always points to the last element pushed onto the stack. The stack is manipulated by interrupts, traps, calls, returns, and the PUSHD,PSHM, POPD, and POPM instructions. Pushes and pops of the stack predecrement and post increment, respectively, all 16 bits of the SP.

13 Circular-Buffer-Size Register (BK)

The 16-bit BK is used by the ARAUs in circular addressing to specify the data block size.

14 Block-Repeat Registers (BRC, RSA, REA)

The block-repeat counter (BRC) is a 16-bit register used to specify the number of times a block of code is to be repeated when performing a block repeat. The block-repeat start address (RSA) is a 16-bit register containing the starting address of the block of program memory to be repeated when operating in the repeat mode. The 16-bit block-repeat end address (REA) contains the ending address if the block of program memory is to be repeated when operating in the repeat mode.

15 Interrupt Registers (IMR, IFR)

The interrupt-mask register (IMR) is used to mask off specific interrupts individually at required times. The interrupt-flag register (IFR) indicates the current status of the interrupts.

16 Processor-Mode Status Register (PMST)

The processor-mode status register (PMST) controls memory configurations of the ‟54x devices.

17 Power-Down Modes

There are three power-down modes, activated by the IDLE1, IDLE2, and IDLE3 instructions. In these modes, the ‟54x devices enter a dormant state and dissipate considerably less power than in normal operation. The IDLE1 instruction is used to shut down the CPU.

The IDLE2 instruction is used to shut down the CPU and on-chip peripherals. The IDLE3 instruction is used to shut down the ‟54x processor completely. This instruction stops the PLL circuitry as well as the CPU and peripherals.

RESULT:

Thus, the architecture of DSP processor TMS320C50 was studied.

EXP. NO: 8 MAC OPERATIONS USING VARIOUS ADDRESSING MODES
DATE:

AIM:

To write an assembly language program for MAC operations using various addressing modes of Processor.

TOOLS REQURIED:

- DSP hardware.
- TMS320C5X-starter kit.
- RS232 cable.

PROCEDURE:

- ✓ Start the Process.
- ✓ In the C50 debugger software.
    - Project-> New project → Save project (.dbj)
    - File → New file
    - Type the program → Save file (.asm)
    - Project → add file to project (asm file)
    - Project → add file to project (micro 50)
    - Project → build
    - Serial → Port settings → Auto detect
    - Serial →Load program → filename.asc
    - Serial→Communication window
    - Reset kit.
    - SD input address→ender
    - Give the input data
    - Reset kit
    - Go C000→ enter
    - Reset kit
    - SD output address→ enter
- ✓ Stop the Process.

PROGRAM

ADDITION

```
INP1   .SET   0H
INP2   .SET   1H
OUT    .SET   2H
       .MMREGS
       .TEXT
START:
       LD     #140H,DP
       RSBX   CPL
       NOP
       NOP
       NOP
       NOP
       LD     INP1, A
       ADD    INP2, A
       STL    A, OUT
H:     B      H
```

INPUT:

A000H 0004H

A001H 0004H

OUTPUT:

A002H 0008H

SUBTRACTION:

```
INP1   .SET   0H
INP2   .SET   1H
OUT    .SET   2H
       .MMREGS
       .TEXT
START:
       LD     #140H,DP
```

```
        RSBX  CPL
        NOP
        NOP
        NOP
        NOP
        LD    INP1, A
        SUB   INP2, A
        STL   A, OUT
H:      B     H
```

INPUT:

A000H 0004H

A001H 0002H

OUTPUT:

A002H 0002H

MULTIPLICATION

```
INP1  .SET  0H
INP2  .SET  1H
OUT   .SET  2H
        .MMREGS
        .TEXT
START:
        LD    #140H, DP
        RSBX  CPL
        RSBX  FRCT
        NOP
        NOP
        NOP
        NOP
        LD    #00H, A
        LD   00H,   T
        MPY 01H, A
```

```
        STL     A, 02H
        STH     A, 03H
H:      B       H
```

INPUT:

A000H 0004H

A001H 0002H

OUTPUT:

A002H 0008H

DIVISION

```
DIVID       .SET    0H
DIVIS       .SET    1H
OUT         .SET    2H
        .MMREGS
        .TEXT
START:
        STM     #140H, ST0
        RSBX    CPL
        RSBX    FRCT
        NOP
        NOP
        NOP
        NOP
        LD      DIVID, A
        RPT     #0FH
        SUBC    DIVIS, A
        STL     A, OUT
H:      B       H
```

INPUT:

A000H 000AH

A001H 0002H

OUTPUT:

A002H 0005H

RESULT

Thus, the MAC operations of various addressing modes were performed by using DSP processor.

EXP. NO: 9 FIR FILTER IMPLEMENTATION

DATE:

AIM:

To implement the FIR Filter using DSP Processor.

TOOLS REQURIED:

- DSP hardware.
- TMS320C5X-starter kit.
- RS232 cable.

ALGORITHM:

- ✓ Get the sum of terms to design the filter.
- ✓ Specify the value of angular frequency from 0 to π and plots divided into 0.01 division.
- ✓ Using FIR function get the design of filter.
- ✓ Finally adjust the value and execute the program.

THEORY:

- ➢ In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time.
- ➢ Infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).
- ➢ The impulse response of N^{th}-order discrete-time FIR filter lasts for $N + 1$ samples, and then settles to zero.
    - o FIR is non recursive structure without response of FIR filter depends only on present and past input samples.

PROCEDURE:

- ✓ Start the program by clicking view click the workspace.
- ✓ Click the serial go to the port settings.
- ✓ Before auto detect reset the kit& click ok to continue.
- ✓ Click project ⟶ new. Save the file &all files will be DSP project.
- ✓ Click assembly file & save the file as ".asm".
- ✓ In left hand side, right click project & add file to project.
- ✓ In left hand side, right click command file & add file to project.
- ✓ Click built in project for the compile of program. Click ok to continue.

- ✓ In serial select the load program, download file will open and browse it. Click ok to continue.
- ✓ In serial communication window type „*sd'* space starting address, enter the input value.
- ✓ After entering the data execute the program.
- ✓ Click enter to verify the output

%FIR LOW PASS FILTER:

```
.mmregs
        .text
        B       START
CTABLE:
        .word   0196H
        .word   017EH
        .word   0EBH
        .word   00H
        .word   0FEFFH
        .word   0FE37H
        .word   0FDEDH
        .word   0FE44H
        .word   0FF35H
        .word   083H
        .word   01D3H
        .word   02B9H
        .word   02DEH
        .word   0218H
        .word   07EH
        .word   0FE6CH
        .word   0FC72H
        .word   0FB36H
        .word   0FB4CH
        .word   0FD0FH
        .word   084H
        .word   0552H
        .word   0ACBH
```

```
        .word   0100CH
        .word   0142FH
        .word   01675H
        .word   01675H
        .word   0142FH
        .word   0100CH
        .word   0ACBH
        .word   0552H
        .word   084H
        .word   0FD0FH
        .word   0FB4CH
        .word   0FB36H
        .word   0FC72H
        .word   0FE6CH
        .word   07EH
        .word   0218H
        .word   02DEH
        .word   02B9H
        .word   01D3H
        .word   083H
        .word   0FF35H
        .word   0FE44H
        .word   0FDEDH
        .word   0FE37H
        .word   0FEFFH
        .word   00H
        .word   0EBH
        .word   017EH
        .word   0196H
        Move the Filter coefficients
        from program memory to data memory
START:
        LAR    AR0,#0200H
```

```
        MAR    *,AR0
        RPT    #33H
        BLKP CTABLE,*+
        SETC CNF
 Input data and perform convolution
ISR:  LDP #0AH
        IN      0,6H
      IN  0,4H
        NOP
        NOP
        NOP
        NOP
        LAR    AR1,#0300H
        LACC 0
        AND    #0FFFH
        SUB    #800H
      MAR    *,AR1
        SACL  *
        LAR    AR1,#333H
        ZAP
        RPT    #33H
        MACD          0FF00H,*-
        APAC
        LAR    AR1,#0300H
        SACH *              ;give as sach *,1 incase of overflow
        LACC *
        ADD    #800H
        SFR                       ;remove if o/p is less amplitude
        SACL  *
        OUT    *,4
        NOP
        B       ISR
        .end
```

%FIR HIGH PASS FILTER:

* Filter type: high pass filter

* Filter Order: 52

* Cutoff frequency in KHz = 3.000000

```
        .mmregs
        .text
        B       START
TABLE:
        .word   0FCD3H
        .word   05H
        .word   0FCB9H
        .word   087H
        .word   0FD3FH
        .word   01ADH
        .word   0FE3DH
        .word   0333H
        .word   0FF52H
        .word   04ABH
        .word   0FFF8H
        .word   0595H
        .word   0FFACH
        .word   0590H
        .word   0FE11H
        .word   047CH
        .word   0FB0BH
        .word   029DH
        .word   0F6BAH
        .word   0AEH
        .word   0F147H
        .word   01CH
        .word   0E9FDH
        .word   04C5H
        .word   0D882H
```

```
        .word   044BCH
        .word   044BCH
        .word   0D882H
        .word   04C5H
        .word   0E9FDH
        .word   01CH
        .word   0F147H
        .word   0AEH
        .word   0F6BAH
        .word   029DH
        .word   0FB0BH
        .word   047CH
        .word   0FE11H
        .word   0590H
        .word   0FFACH
        .word   0595H
        .word   0FFF8H
        .word   04ABH
        .word   0FF52H
        .word   0333H
        .word   0FE3DH
        .word   01ADH
        .word   0FD3FH
        .word   087H
        .word   0FCB9H
        .word   05H
        .word   0FCD3H
*       Move the Filter coefficients
*       from program memory to data memory
START:
        MAR  *,AR0
        LAR    AR0,#0200H
        RPT    #33H
```

```
        BLKP CTABLE,*+
        SETC CNF
* Input data and perform convolution
ISR:    LDP    #0AH
        LACC #0
        SACL  0
        OUT    0,05             ;pulse to find sampling frequency
        IN       0,06H
        LAR    AR7,#0           ;change value to modify sampling freq.
        MAR  *,AR7
BACK:           BANZ BACK,*-
        IN  0,4
        NOP
        NOP
        NOP
        NOP
        MAR  *,AR1
        LAR    AR1,#0300H
        LACC 0
        AND   #0FFFH
        SUB    #800H
        SACL  *
        LAR    AR1,#333H
        MPY   #0
        ZAC
        RPT    #33H
        MACD          0FF00H,*-
        APAC
        LAR    AR1,#0300H
        SACH *              ;give as sach *,1 incase of overflow
        LACC *
        ADD   #800H
        SACL  *
```

```
        OUT   *,4

        LACC #0FFH
        SACL  0
        OUT   0,05
        NOP
        B       ISR
.end
```

% FIR BAND PASS FILTER:

```
* Filter type: bandpass filter
* Filter Order: 52
*  lower Cutoff frequency in KHz = 3.000000Hz
*  upper Cutoff frequency in KHz = 5.000000Hz
.mmregs
.text
B       START
CTABLE:
.word  024AH
        .word  010FH
        .word  0FH
        .word  0FFECH
        .word  0C6H
        .word  0220H
        .word  0312H

.word  02D3H
        .word  012FH
        .word  0FEBDH
        .word  0FC97H
        .word  0FBCBH
        .word  0FCB0H
        .word  0FE9EH
```

```
.word  029H
.word  0FFDCH
.word  0FD11H
.word  0F884H
.word  0F436H
.word  0F2A0H
.word  0F58AH
.word  0FD12H
.word  075FH
.word  01135H
.word  01732H
.word  01732H
.word  01135H
.word  075FH
.word  0FD12H
.word  0F58AH
.word  0F2A0H
.word  0F436H
.word  0F884H
.word  0FD11H
.word  0FFDCH
.word  029H
.word  0FE9EH
.word  0FCB0H
.word  0FBCBH
.word  0FC97H
.word  0FEBDH
.word  012FH
.word  02D3H
.word  0312H
.word  0220H
.word  0C6H
.word  0FFECH
```

```
        .word  0FH
        .word   010FH
        .word   024AH
*        Move the Filter coefficients
*        from program memory to data memory
START:
MAR  *,AR0
LAR    AR0,#0200H
RPT    #33H
BLKP  CTABLE,*+
SETC CNF
ISR:    LDP    #0AH
LACC #0
SACL  0
OUT    0,05              ;pulse to find sampling frequency
IN       0,06H
LAR    AR7,#0            ;change value to modify sampling freq.
MAR  *,AR7
BACK:          BANZ BACK,*-
IN       0,4
NOP
NOP
NOP
NOP
MAR  *,AR1
LAR    AR1,#0300H
LACC 0
AND   #0FFFH
SUB    #800H
SACL  *
LAR    AR1,#333H
MPY   #0
ZAC
RPT    #33H
```

```
MACD        0FF00H,*-
APAC
LAR   AR1,#0300H
SACH *          ;give as sach *,1 incase of overflow
LACC *
ADD   #800H
SACL   *
OUT   *,4
LACC #0FFH
SACL  0
OUT   0,05
NOP
B       ISR
.end
```

% FIR BAND REJECT FILTER:

```
* Filter Order: 52
*  lower Cutoff frequency in KHz = .000000Hz
*  upper Cutoff frequency in KHz = .000000Hz
        .mmregs
        .text
        B       START
CTABLE:
        .word   0FEB9H
        .word   14EH
        .word   0FDA1H
        .word   155H
        .word   0FE1BH
        .word   282H
        .word   0FEAFH
        .word   2ACH
        .word   0FD35H
        .word   8DH
```

```
.word   0F9D9H
.word   0FE07H
.word   0F7CCH
.word   0FEE2H
.word   0FA2FH
.word   4BAH
.word   1AH
.word   25CH
.word   420H
.word   1008H
.word   89H
.word   0D61H
.word   0F3F2H
.word   0AF9H
.word   0DB7EH
.word   045DFH
.word   045DFH
.word   0DB7EH
.word   0AF9H
.word   0F3F2H
.word   0D61H
.word   89H
.word   1008H
.word   420H
.word   25CH
.word   1AH
.word   4BAH
.word   0FA2FH
.word   0FEE2H
.word   0F7CCH
.word   0FE07H
.word   0F9D9H
.word   8DH
```

```
        .word   0FD35H
        .word   2ACH
        .word   0FEAFH
        .word   282H
        .word   0FE1BH
        .word   155H
        .word   0FDA1H
        .word   14EH
        .word   0FEB9H
START:
        MAR   *,AR0
        LAR    AR0,#0200H
        RPT    #33H
        BLKP  CTABLE,*+
        SETC  CNF
ISR:    LDP    #0AH
        LACC #0
        SACL  0
        OUT    0,05             ;pulse to find sampling frequency
        IN       0,06H
        LAR    AR7,#0           ;change value to modify sampling freq.
        MAR  *,AR7
BACK:          BANZ BACK,*-
        IN  0,4
        NOP
        NOP
        NOP
        NOP
        MAR  *,AR1
        LAR    AR1,#0300H
        LACC 0
        AND   #0FFFH
        SUB    #800H
```

```
SACL  *
LAR    AR1,#333H
MPY   #0
ZAC
RPT    #33H
MACD          0FF00H,*-
APAC
LAR    AR1,#0300H
SACH *              ;give as sach *,1 incase of overflow
LACC *
ADD   #800H
SACL   *
OUT   *,4
LACC #0FFH
SACL  0
OUT   0,05
NOP
B        ISR
.end
```

RESULT:

Thus the FIR Filter was implemented using DSP Processor.

EXP. NO: 10 IIR FILTER IMPLEMENTATION

DATE:

AIM:

To implement the IIR Filter using DSP Processor.

TOOLS REQURIED:

- DSP hardware.
- TMS320C5X-starter kit.
- RS232 cable.

ALGORITHM:

- ✓ Get the sum of terms to design the filter.
- ✓ Specify the value of angular frequency from 0 to π and plots divided into 0.01 division.
- ✓ Using IIR function gets the design of filter.
- ✓ Finally adjust the value and execute the program.

THEORY:

- ➢ In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time.
- ➢ Infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).
- ➢ The impulse response of N^{th}-order discrete-time FIR filter lasts for $N + 1$ samples, and then settles to zero.
    - o FIR is non recursive structure without response of FIR filter depends only on present and past input samples.

PROCEDURE:

- ✓ Start the program by clicking view click the workspace.
- ✓ Click the serial go to the port settings.
- ✓ Before auto detect reset the kit& click ok to continue.
- ✓ Click project⟶ new. Save the file &all files will be DSP project.
- ✓ Click assembly file & save the file as ".asm".
- ✓ In left hand side, right click project & add file to project.
- ✓ In left hand side, right click command file & add file to project.

- ✓ Click built in project for the compile of program. Click ok to continue.
- ✓ In serial select the load program, download file will open and browse it. Click ok to continue.

- ✓ In serial communication window type „*sd'* space starting address, enter the input value.
- ✓ After entering the data execute the program.
- ✓ Click -enter- to verify the output.

PROGRAM FOR IIR FILTER:

%Low Pass Filter

```
.MMREGS
      .TEXT
TEMP   .SET   0
INPUT  .SET   1
T1     .SET   2
T2     .SET   3
T3     .SET   4
;
K      .SET   315eh
M      .SET   4e9fh

;cut-off freq is 1Khz. = Fc
;sampling frequency is 100 æs (ie) 0.1ms.
; a = 2 * (355/113) * 1000 = 6283.18/1000 = 6.28 ;; divide by 1000 for secs
; K = aT/(1+aT) = 6.28*0.1 / (6.28*0.1+1) = 0.3857
; M = 1/(1+aT) = 1 / (6.28*0.1+1) = 0.61425
;convert to Q15 format
; K = K * 32767 = 12638.23 = 315Eh
; M = M * 32767 = 20127.12 = 4E9Fh
;Sampling Rate is 100 æs & Cut off Frequency is 1 Khz
      LDP    #100H
      LACC   #0
      SACL   T1
      SACL   T2
```

```
      SACL    TEMP
      OUT     TEMP,4          ;CLEAR DAC BEFORE START TO WORK
LOOP:
      LACC    #0
      SACL    TEMP
      OUT     TEMP,5          ;OUTPUT LOW TO DAC2 TO CALCULATE TIMING;
      IN      TEMP,06           ;SOC;
      LAR     AR7,#30h        ;CHANGE VALUE TO MODIFY SAMPLING FREQ
                              ;sampling rate 100ms.
       MAR     *,AR7
         BACK:          BANZ BACK,*-
;
      IN      INPUT,4         ;INPUT DATA FROM ADC1
         NOP
         NOP
;
      LACC    INPUT
      AND     #0FFFH
      SUB     #800h
      SACL    INPUT
;
      LT      INPUT
      MPY     #K
      PAC
      SACH    T1,1
;;;CALL MULT ----MULTIPLICATION TO BE DONE WITH K
;;RESULT OF MULT IN T1
;
      LT      T2      ;PREVIOUS RESULT IN T2
      MPY     #M
      PAC
      SACH    T3 ,1
;;;CALL MULT ----MULTIPLICATION TO BE DONE WITH M
```

```
;;RESULT OF MULT IN T3+
      LACC   T1
      ADD    T3

      SACL   T2
      ADD    #800h
      SACL   TEMP
      OUT    TEMP,4          ;OUTPUT FILTER DATA TO DAC1
      LACC   #0FFH
      SACL   TEMP
      OUT    TEMP,5          ;OUTPUT HIGH TO DAC2 TO CALCULATE TIMING
      B      LOOP
```

%High Pass Filter:

```
        .MMREGS
        .TEXT
START:
        LDP #100H
        LACC #00H
        SACL 00H
        SACL 01H
        SACL 02H
        SACL 03H
        SACL 04H
        SACL 05H
LOOP:
        LACC #00H
        SACL 00H
        IN 0,06H
        LAR AR7,#30H
        MAR *,AR7
BACK:       BANZ BACK,*-
;       LT 01H
;       MPY #0FFFFB5DEH
```

```
;       PAC
;       SACH 05H,1
        IN 0,04H
        NOPS

        NOP
        NOP
        NOP
        LT 01H
;       MPY #0FFFFB5DEH
        MPY #4A22H
;       MPY #315EH
        PAC
        SACH 05H,1
        LACC 00H
        AND #0FFFH
        XOR #800H
        SUB #800H
        SACL 00H
        SACL 01H
        ZAP
        LT 00H
;       DMOV 00H
;       LTD 00H
        MPY #4A22H
;       MPY #315EH
        PAC
        SACH 02H,1
        LT 03H
        MPY #1446H
;       MPY #4E9FH
        PAC
        SACH 04H,1
```

```
        LACC 02H
        ADD 04H
        SUB 05H
        SACL 03H
        ADD #800H

        SACL 00H
;       OUT 00H,1AH
        OUT 0,04H
        B LOOP
        NOP
        NOP
H:      B H
```

%Band Pass Filter:

```
        .MMREGS
.TEXT
START:
        LDP #100H
        NOP
        NOP
        NOP
        LACC #00H
        LAR AR0,#00FFH
        LAR AR1,#8000H
        MAR *,AR1
LOOP:
        SACL *+,AR0
        BANZ LOOP,AR1

LOOP1:
        LACC #00H
        SACL 00H
```

```
        IN 0,06H
        LAR AR7,#30H
        MAR *,AR7
BACK:   BANZ BACK,*-
        IN 0,04H
        NOP

        NOP
        NOP
        NOP

        LT 04H
        MPY #2FC4H
;       MPY #05F8H
        PAC
        SACH 24H,0
;       SACH 24H,4

        LT 03H
        MPY #99B2H
;       MPY #1336H
        PAC
        SACH 23H,0
;       SACH 23H,4
        LT 02H
        MPY #0DB29H
;       MPY #1B65H
        PAC
        SACH 22H,0
;       SACH 22H,4
        LT 01H
        MPY #99B2H
;       MPY #1336H
```

```
        PAC
        SACH 21H,0
;       SACH 21H,4
        LACC 03H
        SACL 04H
        LACC 02H
        SACL 03H

        LACC 01H
        LACC 02H

        LACC 00H
        AND #0FFFH
        XOR #800H
        SUB #800H
        SACL 00H
        SACL 01H
        ZAP
;       DMOV 03H
;       DMOV 02H
;       DMOV 01H
        LT 00H
        MPY #2FC4H
;       MPY #05F8H
        PAC
        SACH 20H,0
;       SACH 20H,4
;
        LT 73H
        MPY #2A22H
;       MPY #0544H
        PAC
        SACH 63H,0
```

```
;       SACH 63H,4
        LT 72H
        MPY #6761H
;       MPY #0CECH
        PAC
        SACH 62H,0
;       SACH 62H,4
        LT 71H

        MPY #0B6E8H
;       MPY #16DDH
        PAC
        SACH 61H,0
;       SACH 61H,4

        LACC 72H
        SACL  73H
        LACC 71H
        SACL  72H
        LACC 70H
        SACL 71H
;       DMOV 72H
;       DMOV 71H
;       LTD 70H
        LT 70H
        MPY #0F184H
;       MPY #1E30H
        PAC
        SACH 60H,0
;       SACH 60H,4
        LACC 20H
        SUB 21H
        ADD 22H
```

```
        SUB 23H
        ADD 24H
        ADD 60H
        SUB 61H
        ADD 62H
        SUB 63H
        SACL 70H
        ADD #800H
        SACL 00H

        OUT 00H,1AH
        IN 0,04H
        B LOOP1
        NOP
        NOP
H:      B       H
```

% Band Reject Filter:

```
;IIR BANDREJECT FILTER
;.MMREGS
.TEXT
START:
        LDP #100H
        LACC #00H
        LAR AR0,#00FFH
        LAR AR1,#8000H
        MAR *,AR1
LOOP:
        SACL *+,AR0
        BANZ LOOP,AR1
LOOP1:
        LACC #00H
        SACL 00H
```

```
        IN 0,06H
        LAR AR7,#30H
        MAR *,AR7
BACK:   BANZ BACK,*-
        IN 0,04H
        NOP
        NOP
        NOP
        NOP
        LT 04H

        MPY #003BH

        PAC
;       SACH 24H,0
;       RPT #0BH
;       SFR
        SACH 24H,4
        LT 03H
        MPY #0000H
        PAC
;       RPT #0BH
;       SFR
;       SACH 23H,0
        SACH 23H,4
        LT 02H
        MPY #0077H
        PAC
;       RPT #0BH
;       SFR
;       SACH 22H,0
        SACH 22H,4
        LT 01H
```

```
        MPY #0000H
        PAC
;       RPT #0BH
;       SFR
;       SACH 21H,0
        SACH 21H,4
        LACC 03H
        SACL 04H
        LACC 02H
        SACL 03H

        LACC 01H
        LACC 02H

        LACC 00H
        AND #0FFFH
        XOR #800H
        SUB #800H
        SACL 00H
        SACL 01H
        ZAP
        LT 00H
        MPY #003BH
        PAC
;       RPT #0BH
;       SFR
;       SACH 20H,0
        SACH 20H,4
        LT 73H
        MPY #0B04H
        PAC
;       RPT #0BH
;       SFR
```

```
;       SACH 63H,0
        SACH 63H,4
        LT 72H
        MPY #1226H
        PAC
;       RPT #0BH
;       SFR
;       SACH 62H,0
        SACH 62H,4

LT 71H
        MPY #21A3H
        PAC
;       RPT #0BH
;       SFR
;       SACH 61H,0
        SACH 61H,4
        LACC 72H
        SACL 73H
        LACC 71H
        SACL 72H
        LACC 70H
        SACL 71H
        LT 70H
        MPY #15E9H
        PAC
;       RPT #0BH
;       SFR
;       SACH 60H,0
        SACH 60H,4
        LACC 20H
        ADD 21H
        SUB 22H
```

```
        ADD 23H
        ADD 24H
        ADD 60H
        SUB 61H
        ADD 62H
        SUB 63H
        SACL 70H
        ADD #800H
        SACL 00H
;       OUT 00H,1AH

        OUT 0,04H
        B LOOP1
        NOP
        NOP
H:      B  H
```

RESULT:

Thus the IIR Filter was implemented using DSP Processor.

EXP. NO: 11 IMPLEMENT UP-SAMPLING AND DOWN-SAMPLING

DATE:

AIM

To write a program to implement Up-sampling and Down-sampling operation using MATLAB.

APPARATUS REQUIRED:

Personal Computer

MATLAB Software

ALGORITHM:

1. Start the program.
2. Get the number of samples.
3. Calculate the output using the interpolator and decimator formula.
4. Stop the process.

THEORY:

Decimation is the process of reducing the sampling rate. In practice, this usually implies low pass-filtering a signal, then throwing away some of its samples. "Down sampling" is a more specific term which refers to just the process of throwing away samples, without the low pass filtering operation.

The decimation factor is simply the ratio of the input rate to the output rate. It is usually symbolized by "D", so input rate / output rate D.A signal can be down sampled (without doing any filtering) whenever it is "oversampled", that is, when a sampling rate was used that was greater than the Nyquist criteria required.

Specifically, the signal's highest frequency must be less than half the post-decimation sampling rate.

Interpolation is the process of increasing the sampling rate. In practice, this usually implies low pass-filtering a signal, then throwing away some of its samples. "Up sampling" is a more specific term which refers to just the process of throwing away samples, without the low pass filtering operation. The interpolation factor is simply the ratio of the input rate to the output rate. It is usually symbolized by "M", so input rate / output rate=M.

A signal can be up sampled (without doing any filtering) whenever it is "under sampled", that is, when a sampling rate was used that was greater than the Nyquist criteria required. Specifically, the signal's highest frequency must be double the post-interpolation sampling rate.

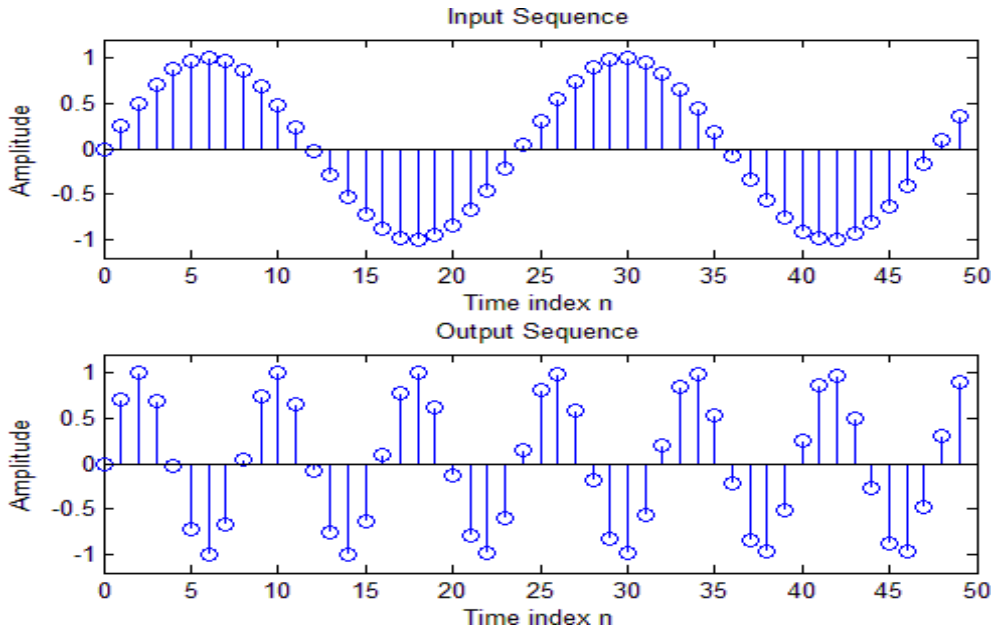
PROCEDURE:

- Start the MATLAB software and create new M-file.
- Type the program in the file.
- Save & compile the program.
- Give the input data.
- Observe the output waveform.
- Thus the graph is to be plotted

PROGRAM:

```
%down sampler
        clf;
        n = 0: 49;
        m = 0: 50*3 - 1;
        x = sin(2*pi*0.042*m);
        y = x([1: 3: length(x)]);
        subplot(2,1,1)
        stem(n, x(1:50)); axis([0 50 -1.2 1.2]);
        title("Input Sequence");
        xlabel("Time index n");
        ylabel("Amplitude");
        subplot(2,1,2)
        stem(n, y); axis([0 50 -1.2 1.2]);
        title("Output Sequence");
        xlabel("Time index n");
        ylabel("Amplitude");
```

OUTPUT:



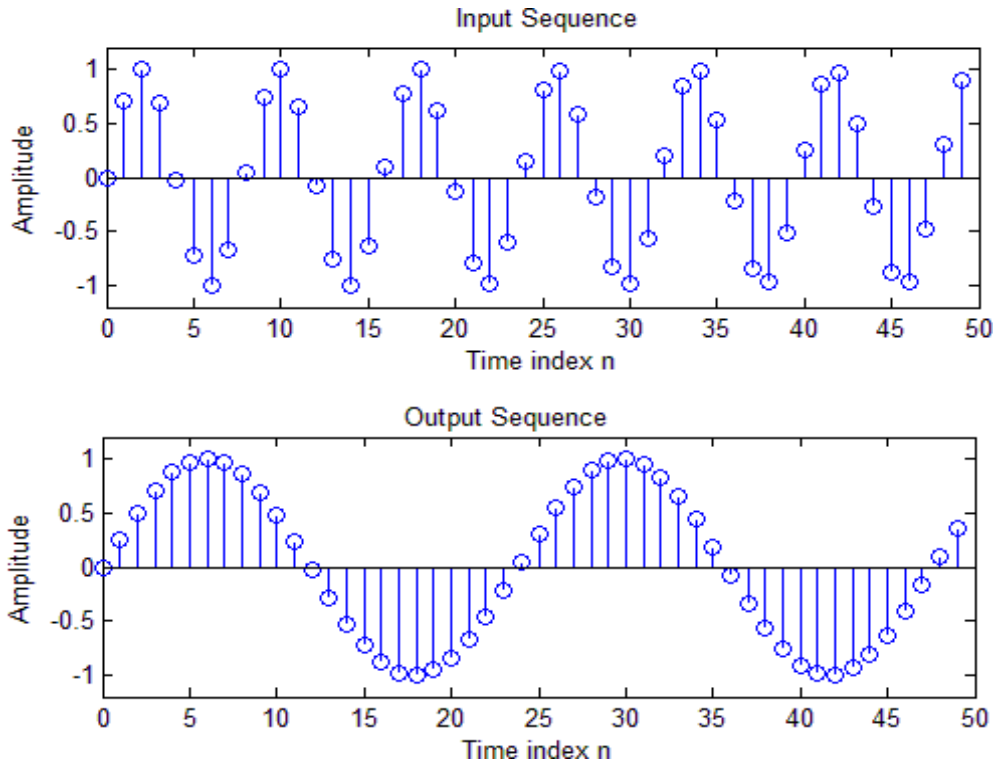
PROGRAM

%UP SAMPLING

```
clc;
clear all;
close all;
N=125;
N= 0 : 1: N -1 ;
X= sin (2* pi * n/15);
L=2;
figure (1)
stem (n,x);
grid on;
xlabel (´No. of samples´);
ylabel (´Amplitude´);
title (´ Original sequence´);
x1 =[zeros (1, L*N)];
n1 = 1:1: L*N;
j= 1:L: L*N;
x1 (j) =x;
figure(2)
stem (n1-1, x1);
```

```
grid on;
xlabel (´No. of samples´);
ylabel (´Amplitude´);
title (´ Unsampled sequence´);
a=1;
b=fir1(5, 0.5, ´low´);
y=filter (b,a, x1);
figure (3)
stem (n1-1, y);
grid on;
xlabel(´ No. of samples´);
ylabel (´Amplitude´);
title (´ Interpolated sequence´);
```

OUTPUT:



RESULT:

Thus the Up sampling and down-sampling operations were implemented using MATLAB.