Mini Project

Group Members:Harish Shinde(53),Shubham Bhor(57),Rhuturaj Takle(63)

Class:SE COMPS B3 **Grade:** **Sign:**

Problem:-Simulation for Booths Algorithm.

Abstract: Booth's Algorithm is used for Multiplication of two numbers. The main purpose is to produce a multiplier with simulation and to maximize the speed in which the multiplier performs the calculation.The simulation algorithm can be analysed by using algorithm analysis measures. In the field of Digital Signal Processing and graphics applications, multiplication is an important and computationally intensive operation. The efficiency of the multiplier has always been a critical issue and, therefore, the subject of many research projects and papers. Booth algorithm is a crucial improvement in the design of signed binary multiplication.This simulation of Booth's algorithm using python tkinter can multiply signed integers and provides each steps as well as the answer in decimal as well as binary.

Program:

```
from tkinter import *

import tkinter.messagebox

import time

from PIL import Image, ImageTk

def convert_to_binary(n, z):

    return ("{0:0%db}" %int(z)).format(n)


def convert_to_integer(b):

    return int(b,2)




def bit_flip(a,z):

    a = convert_to_binary(a,z)

    b = ""

    for w in a:
```

```
        if w == '1':
            w = '0'
        else:
            w = '1'
        b = b + w
    return b


def get_twos_complement(b,z):
    w = bit_flip(b,z)
    one = convert_to_binary(1,z)
    a = add_binary(w, one,z)
    a = a[-z:]
    return a

def add_binary(a, b, z):
    return convert_to_binary(int(a, 2) + int(b, 2),z)

def getRMB(a):
    return a[len(a)-1:]


def right_shift(A,z):
    A = convert_to_integer(A)
    b = convert_to_binary(A >> 1, z*2)
    return b


def calculate():
            def leftclick(event):
```

```
            a=t1.get()
            b=t2.get()
            z=t3.get()
            a=int(a)
            b=int(b)
            z=int(z)
            A = convert_to_binary(0,z)
            B = ""
            q = ""
            B_Comp = ""

            if(a < 0):  #For Negative Number
                a *= -1
                B = get_twos_complement(a,z)
                B_Comp = convert_to_binary(a.z)
            else:
                B = convert_to_binary(a,z)
                B_Comp = get_twos_complement(a,z)

            if(b<0):  #For Negative Number
                b *= -1
                q = get_twos_complement(b,z)
            else:
                q = convert_to_binary(b,z)
            q1 = '0'

            result="************** Binary Multiplication using Booth's
Algorithm*****************"
            Label(root,text=result,fg='orange',font='times 20').grid(row=5,sticky=W)
            result="A="+A+"="+str(convert_to_integer(A))+"                    M =
"+B+"="+str(convert_to_integer(B))
```

```
            Label(root,text=result).grid(row=6,sticky=W)
            result="M_Comp = "+B_Comp+"="+str(convert_to_integer(B_Comp))+"
q = "+q+"="+str(convert_to_integer(q))
            Label(root,text=result).grid(row=7,sticky=W)
            count = z
            print(count)
            step = 1
            i=9
            result="    A                                    q
q-1                          M                              OPERATIONS\n"
            Label(root,text=result).place(x=20,y=90)
            result=A+"          "+q+"                  "+q1+"          "+B+"
Initialize\n"
            T.insert(END, result)
            while(count>0):
                q0 = getRMB(q)
                #For the cases where q0 and q-1 are both 0 or 1
                if ((q0 == '0') and (q1 == '0')) or ((q0 == '1') and (q1 == '1')):
                    result=A+"          "+q+"                  "+q1+"
opcode="+q0+q1+"\n"
                    T.insert(END, result)
                    i=i+1
                    c = A + q + q1
                    c = right_shift(c,z)
                    c = A[0] + c
                    A = c[:z]
                    q = c[z:2*z]
                    q1 = c[len(c)-1]
                    result=A+"          "+q+"                  "+q1+"
Arithmetic Right Shift\n"
                    T.insert(END, result)
```

```
                    result="------------------------------------------------------------------
-------------------------------------------------------------------------------\n"
                    T.insert(END, result)
                #For the cases where q0 is 0 and q-1 is 1
                elif (q0 == '0') and (q1 == '1'):
                    result=A+"          "+q+"               "+q1+"
opcode="+q0+q1+"\n"
                    T.insert(END, result)
                    i=i+1
                    A = add_binary(A, B, z)
                    A = A[-z:]
                    c = A + q + q1
                    result=A+"          "+q+"               "+q1+"
A=A+M\n"
                    T.insert(END, result)
                    c = right_shift(c,z)
                    c = A[0] + c
                    A = c[:z]
                    q = c[z:2*z]
                    q1 = c[len(c)-1]
                    result=A+"          "+q+"               "+q1+"
Arithmetic Right Shift \n"
                    T.insert(END, result)
                    result="------------------------------------------------------------------
-------------------------------------------------------------------------------\n"
                    T.insert(END, result)
                    #For the cases where q0 is 1 and q-1 is 0
                elif (q0 == '1') and (q1 == '0'):
                    result=A+"          "+q+"               "+q1+"
opcode="+q0+q1+"\n"
                    T.insert(END, result)
                    i=i+1
```

```
                A = add_binary(A, B_Comp, z)
                A = A[-z:]
                c = A + q + q1
                result=A+"                "+q+"                    "+q1+"                    A=A-M\n"
                T.insert(END, result)
                c = right_shift(c,z)
                c = A[0] + c
                A = c[:z]
                q = c[z:2*z]
                q1 = c[len(c)-1]
                result=A+"                "+q+"                    "+q1+"                    Arithmetic Right Shift\n"
                T.insert(END, result)
                result="-----------------------------------------------------------------------------------------------------------------------------------------------------\n"
                T.insert(END, result)
            count-=1
            step+=1
            result="                                    The Result in binary form is "+(A+q)+"\n"
            Label(root,text=result,fg="red",font='times 20').place(x=0,y=560)
            res = ""
            a=A+q
            if a[0] == '1':
                i = convert_to_integer(a)
                b = bit_flip(i,z)
                one = convert_to_binary(1,z)
                res = add_binary(b, one,z)
            else:
                res = a
```

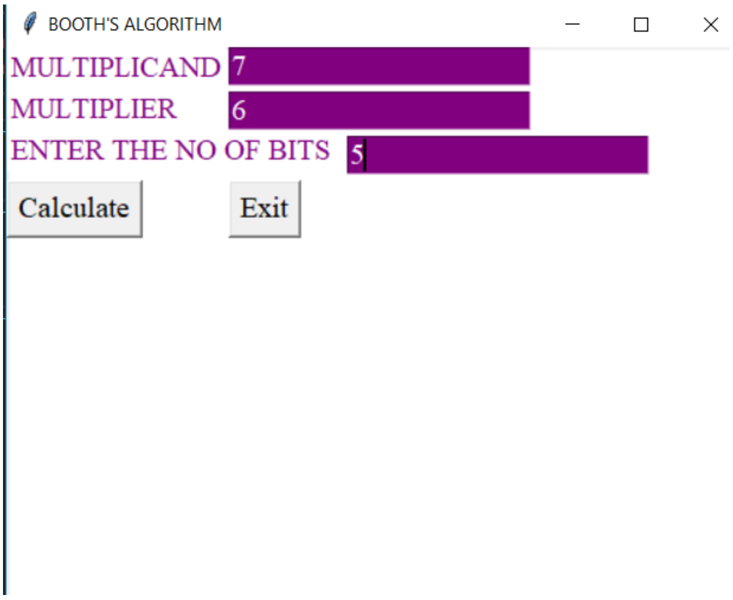
```
                    value = convert_to_integer(res)
                    if a[0] == '1':
                        value*=-1

                    result="                    The Result in Decimal form is "+str(value)+"\n"
                    Label(root,text=result,fg="red",font='times 20').place(x=0,y=600)
        root=Tk()
        root.title("BOOTH'S ALGORITHM")
        root.geometry('1000x900')
        T = Text(root, height=25, width=150)
        T.place(x=10,y=130)
        root.bind("<Button-1>",leftclick)

def quits():
        answer=tkinter.messagebox.askquestion("Quit","Do You Want To Quit")
        if answer=='yes':
                root.quit()
root=Tk()
root.title("BOOTH'S ALGORITHM")
root.geometry('500x500')
image = Image.open("booths.png")
photo = ImageTk.PhotoImage(image)
label = Label(image=photo)
label.image = photo # keep a reference!
label.place(x=0,y=0)
l1=Label(root,text="MULTIPLICAND",font='times 15',fg='purple',bg='white')
l2=Label(root,text="MULTIPLIER",font='times 15',fg='purple',bg='white')
l3=Label(root,text="ENTER THE NO OF BITS",font='times 15',fg='purple',bg='white')
t1=Entry(root,bg='purple',fg='white',font='times 15')
t2=Entry(root,bg='purple',fg='white',font='times 15')
```

```
t3=Entry(root,bg='purple',fg='white',font='times 15')
b3=Button(root,text="Calculate",font='times 15',command=calculate)
b2=Button(root,text="Exit",font='times 15',command=quits)
l1.grid(row=1,sticky=W)
l2.grid(row=2,sticky=W)
l3.grid(row=3,sticky=W)
t1.place(x=150,y=0)
t2.place(x=150,y=30)
t3.place(x=230,y=60)
b2.place(x=150,y=90)
b3.place(x=0,y=90)
root.mainloop()
```

Output:-



BOOTH'S ALGORITHM

************* Binary Multiplication using Booth's Algorithm***************

A=00000=0 M = 00111=7

M_Comp = 11001=25 q = 00110=6

A	q	q-1	M	OPERATIONS
00000	00110	0	00111	Initialize
00000	00110	0		opcode=00
00000	00011	0		Arithmetic Right Shift
00000	00011	0		opcode=10
11001	00011	0		A=A-M
11100	10001	1		Arithmetic Right Shift
11100	10001	1		opcode=11
11110	01000	1		Arithmetic Right Shift
11110	01000	1		opcode=01
00101	01000	1		A=A+M
00010	10100	0		Arithmetic Right Shift
00010	10100	0		opcode=00
00001	01010	0		Arithmetic Right Shift

The Result in binary form is 0000101010

The Result in Decimal form is 42