Don Bosco Institute of Technology,Kurla

EXPERIMENT NO:2

Title: A program for binary multiplication

Name: Hayden Cordeiro **Class : SE Comps** **Roll No: 05**

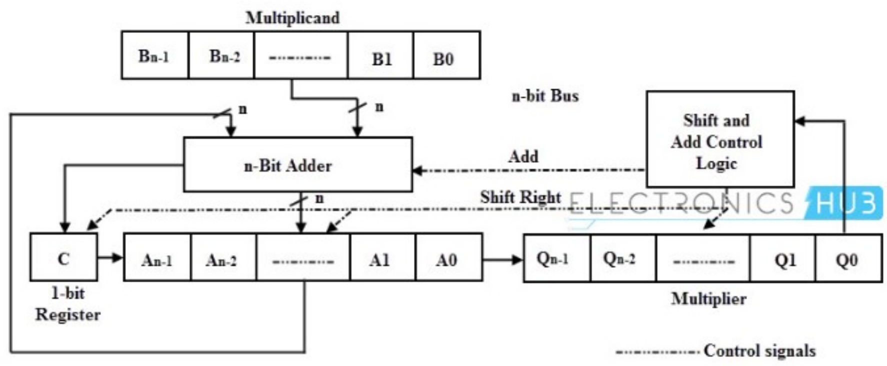
Class:S.E **Lecturer:Sejal.Ch**

Subject: PA

EXPERIMENT NO:2

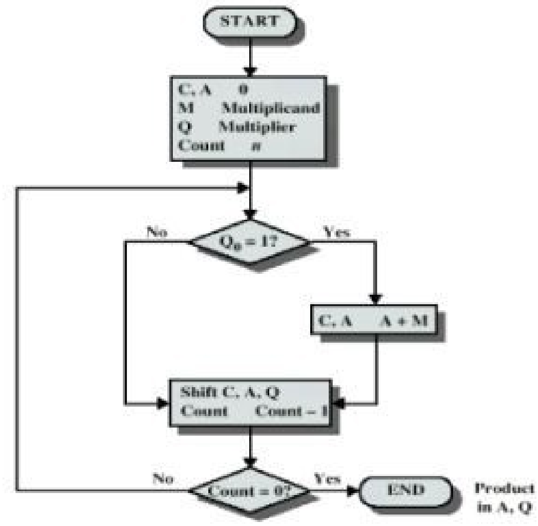
Simulate binary multiplication

AIM	Write a program to simulate binary multiplication
LEARNING OBJECTIVE	To implement the operation of the arithmetic unit including the implementation of fixedpoint multiplication for unsigned numbers.
LEARNING OUTCOME	Students will be able to write a higher level language code for simulating hardware operation for binary multiplication.
LAB OUTCOME	CSL 403.1: Ability to compile a code for computer operations.
PROGRAM OUTCOME	PO11, PO52, PO83, PO93, PO122, PSO12
BLOOM'S TAXONOMY LEVEL	Remember, Understand
THEORY	As in binary number system there are only 0 and 1 present as digits so we have to know the fundamental interrelation between these two digits during multiplication. Like in case of binary addition and binary multiplication there are also four steps to be followed during a bigger multiplication or we can say these fundamental steps as well. 1 X 1 = 1 1 X 0 = 0 0 X 1 = 0 0 X 0 = 0 As we can see that if we can compare these rules of binary multiplication with that of decimal multiplication we will not have any difference at all. So it is a comparatively easy method than the previously discussed two operations.

	 In this, the 4 bit multiplier is stored in Q register, the 4 bit multiplicand is stored in register B and the register A is initially cleared to zero. The multiplication process starts with checking of the least significant bit of B whether it is 0 or 1. If the B0 = 1, the number in the multiplicand (B) is added with the least significant bits of the A register and all bits of C, A and Q registers are shifted to the right one bit. If the bit B0 = 0, the combined C and Q registers are shifted to the right by one bit without performing any addition. This process is repeated for n times for n bit numbers. This method of binary multiplication is called as parallel multiplier. Consider the below figure in which the multiplier and multiplicand values are given as 1011 and 1101 which are loaded into the Q and A registers respectively. Initially the register C is zero and hence the A register is zero, which stores the carry in addition. Since the B0 =1, then the number in the B is added to the bits of A and produce the addition result as 1101, and the Q and A register are shifted their values one bit right so the new values during the first cycle are 0110 and 1101 respectively. This process has to be repeated four times to perform the 4 bit multiplication. The final multiplication result will be available in the A and Q registers as 10001111 as shown in the figure.
SOFTWARE USED	C/C++/Java/Python
STEPS TO EXECUTE THE PROGRAM	1. Take two 4 bit binary numbers from the user (M=Multiplicand & Q=Multiplier). 2. Intialise the counter with the count of number of bits . 3. Intialise A and C to zero ,where A is accumulator which stores the MSB of the result and C is the carry which stores

	the required value after partial addition. Multiplier Q stores the LSB of the result. 4. Check LSB bit of Q,If 1 change C and A=A+M and shift C,A,Q towards right. 5. Check LSB bit of Q,If 0 shift C,A,Q towards right. 6. Check the counter,if it is 1,move to step 4,otherwise store the result in A and Q.

FLOWCHART



EXAMPLE

M
1011

Counter	C	A	Q	Operations	
4	0	0000	1101	Initial Values	
3	0	1011	1101	Add	First Cycle
	0	0101	1110	Shift	
2	0	0010	1111	Shift	Second Cycle
1	0	1101	1111	Add	Third Cycle
	0	0110	1111	Shift	
0	1	0001	1111	Add	Fourth Cycle
	0	1000	1111	Shift	

CODE

Part A:

```
import sys
m=input();q=input();counter=len(m);acc='0000';carry='0'
print("----------------------------------------------")
print("counter"+" | "+"carry"+" | "+"accu"+"   | "+"   q"+"  |
"+"operartion |")
print("----------------------------------------------")
print(counter,"      |   ",carry," | ",acc," | ",q,"| Initial    |")
print("----------------------------------------------")
counter-=1;ask=input('enter 1 to continue');sys.stdout.write("\
```

```
033[F") ,sys.stdout.write("\033[K")
while(ask=='1' and counter!=-1):
    if(q[len(q)-1]=='1'):
        #sum
        sum=str(bin(int(m,2) + int(acc,2))[2:]).zfill(5)
        carry=sum[0]
        acc=sum[1:]
        print(counter,"      |   ",carry," | ",acc," | ",q,"| add
|")
    else:
        carry=acc[0]
    ##rs
    q=(acc[3]+q)[:4]
    acc=(carry+acc)[:4]
    print(counter,"      |   ",carry," | ",acc," | ",q,"| shift      |")
    print("---------------------------------------------")
    counter-=1;ask=input('enter 1 to
continue');sys.stdout.write("\033[F") ;sys.stdout.write("\
033[K")
```

```
0111
0010
-------------------------------------------------
counter | carry | accu    |     q   | operartion |
-------------------------------------------------
4       |    0  |  0000   |  0010 | Initial    |
-------------------------------------------------
3       |    0  |  0000   |  0001 | shift      |
-------------------------------------------------
2       |    0  |  0111   |  0001 | add        |
2       |    0  |  0011   |  1000 | shift      |
-------------------------------------------------
1       |    0  |  0001   |  1100 | shift      |
-------------------------------------------------
0       |    0  |  0000   |  1110 | shift      |
-------------------------------------------------
```

Part B:

```
import sys
m=input();q=input();counter=len(m);acc='0000';carry='0'
print("---------------------------------------------")
print("counter"+" | "+"carry"+" | "+"accu"+"    | "+"    q"+"   |
"+"operartion |")
print("---------------------------------------------")
print(counter,"      |   ",carry," | ",acc," | ",q,"| Initial    |")
print("---------------------------------------------")
counter-=1;ask=input('enter 1 to continue');sys.stdout.write("\
033[F") ,sys.stdout.write("\033[K")
while(ask=='1' and counter!=-1):
    if(q[len(q)-1]=='1'):
        #sum
```

```
                sum=str(bin(int(m,2) + int(acc,2))[2:]).zfill(5)
                carry=sum[0]
                acc=sum[1:]
                print(counter,"       |    ",carry," | ",acc," | ",q,"| add
|")
        else:
                carry=acc[0]
        ##rs
        q=(acc[3]+q)[:4]
        acc=(carry+acc)[:4]
        print(counter,"       |    ",carry," | ",acc," | ",q,"| shift       |")
        print("----------------------------------------------------")
        counter-=1;ask=input('enter 1 to
continue');sys.stdout.write("\033[F") ;sys.stdout.write("\
033[K")
```

case 1:

```
111
10
-------------------------------------------------
counter | carry | accu    |    q   | operartion |
-------------------------------------------------
2       |    0  |  0000   |  10 | Initial    |
-------------------------------------------------
1       |    0  |  0000   |  01 | shift      |
-------------------------------------------------
0       |    0  |  0111   |  01 | add        |
0       |    0  |  0011   |  10 | shift      |
-------------------------------------------------
```

case 2:

```
10
111
-------------------------------------------------
counter | carry | accu    |    q   | operartion |
-------------------------------------------------
3       |    0  |  0000   |  111 | Initial    |
-------------------------------------------------
2       |    0  |  0010   |  111 | add        |
2       |    0  |  0001   |  011 | shift      |
-------------------------------------------------
1       |    0  |  0011   |  011 | add        |
1       |    0  |  0001   |  101 | shift      |
-------------------------------------------------
0       |    0  |  0011   |  101 | add        |
0       |    0  |  0001   |  110 | shift      |
-------------------------------------------------
```

CONCLUSION	We have successfully implemented fixed point multiplication for unsigned numbers . The binary inputs were taken from the user and the steps of the process were displayed in a tabular format. The additional scope of the experiment was to multiply binary number of variable length.
REFERENCES	1. William Stallings, “Computer Organization and Architecture: Designing for Performance”,Pearson Publication, 10 th Edition, 2013 2. B. Govindarajulu, “Computer Architecture and Organization: Design Principles and Applications”, Second Edition, McGrawHill (India)