Name: SUYOG DHAKAL 075BCT092

LAB3: MULTIPLICATION OF TWO UNSIGNED INTEGER BINARY NUMBERS BY PARTIAL-PRODUCT METHOD.

Objective: To simulate binary multiplication by partial product method.

Theory:

The program for multiplying two numbers is based on the procedure we use to multiply number with paper and pencil. Multiplication process consists of checking the bits of the multiplier B and adding the multiplicand A, as many times as there are 1's in B, provided that the value of A is shifted left from one line to the next. As the computer can add only two numbers at a time, we reserve a memory location, P (say) to store intermediates sums. The intermediate sum is called partial products as they hold a partial product until all numbers are added. This is the reason why the method named partial product method.Partial product is initially started with the zero. The multiplicand A is added to the content of P for each bit of the multiplier B that is 1. The Value of A is shifted left after checking each bit of the multiplier. The final value in P gives the products of the two unsigned integer binary number.
For 4-bit numbers, when multiplied, the product contains eight significant bits.

Code:

AND.m

```
function var = AND(a,b)
  if(a==1 && b==1)
   var =1;
   else
    var=0;
```

```
  endif
endfunction
```

OR.m

```
function var = OR(a,b)
  if(a==0 && b==0)
   var = 0;
  else
   var = 1;
  endif
 endfunction
```

XOR.m

```
function var = XOR(a,b)
 if(a!=b)
   var = 1;
 else
   var = 0;
  endif
endfunction
```

fulladder.m

```
function [sum,carry] = fulladder(a,b,c)
 sum = XOR(XOR(a,b),c);
 carry = OR(AND(a,b),AND(XOR(a,b),c));
endfunction
```

adder.m

```
function [sum,carry] = adder(num1,num2,sub=0)
 if(nargin<2)
   error("few arguments");
 endif
 if(!isvector(num1) && !isvector(num2))
   error("Requires vector arguments");
 endif
 i=length(num1);
 j=length(num2);
 if(i>j)
   gt=i;
```

```
 else
  gt=j;
 endif

 sum = zeros(1,gt);
 carry = sub;

 while(i>=1 && j>=1)
  [sum(gt), carry] = fulladder(num1(i--), XOR(num2(j--),sub),carry);
  --gt;
 endwhile

 while(i>=1)
  [sum(i),carry] = fulladder(num1(i--),XOR(0,sub),carry);
 endwhile

 while(j>=1)
  [sum(j), carry] = fulladder(0, XOR(num2(j--),sub),carry);
 endwhile

endfunction
```

multiplier.m

```
function [product carry] = multiplier(mpc, mpl)

        lenmpc = length(mpc);
        lenmpl = length(mpl);

        res = 2 * lenmpc;
        partialproduct = zeros(lenmpl, res);

        for i = 1 : lenmpl
                if (mpl(lenmpl + 1 - i) == 1)
                        startbit = (res + 2 - lenmpc - i);
                        endbit = res + 1 - i;
                        partialproduct(i, startbit:endbit) = mpc(1, :);
                endif
        endfor

        product = zeros(1, size(partialproduct, 2));
```

```
    for i = 1 : size(partialproduct, 1)
        [product carry] = adder(partialproduct(i, :), product);
    endfor

    partialproduct

endfunction
```

OUTPUT:

```
>>  multiplier([1 1],[1 1])
partialproduct =

   0   0   1   1
   0   1   1   0

ans =

   1   0   0   1
```

Discussion and conclusion:

For implementation of multiplication of two unsigned integer binary numbers in a digital computer, partial product method is frequently used. The intermediate sum is called partial products as they hold a partial product until all numbers are added. This is the reason why the method is named partial product method. In this lab, the implementation of this method was understood.