DELHI TECHNOLOGICAL UNIVERSITY

INNOVATIVE PROJECT REPORT

COA (CO 4)

Submitted to – Dr. Pawan Singh Mehra
Submitted by – Ritik Singh (2K19/CO/319)

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

SIMULATION OF BINARY MULTIPLIER

Objective: The purpose of this project is about simulating a binary multiplier which are used for reducing partial products and computing result using adders.

➢ Introduction

A **binary multiplier** is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers.The two numbers are more specifically known as multiplicand and multiplier and the result is known as a product.

The multiplicand & multiplier can of various bit size. The product's bit size depends on the bit size of the multiplicand & multiplier. The bit size of the product is equal to the sum of bit size of multiplier& multiplicand.

It is built using binary adders. A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together.

In binary encoding each long number is multiplied by one digit (either 0 or 1),as the product by 0 or 1 is just 0 or the same number. Therefore, the multiplication of two binary numbers comes down to calculating partial products (which are 0 or the first number), shifting them left, and then adding them together.

Mutipliers can be classified as hardware multipliers and software multipliers. In older digital systems, there was no hardware multiplier and multiplication was implemented with a micro program. The micro program needed many micro instruction cycles to complete the multiplication process, which make the microprogrammed multipliers slow. For high speed digital systems, hardware multipliers are usually used. In modem microprocessors and ASIC processors, most arithmetic logic units (ALU) contain a hardware multiplier. High speed hardware multipliers have been of interest for sometime. More sophisticated approaches for multiplier designs can be implemented today due to the increase density of integrated circuits.

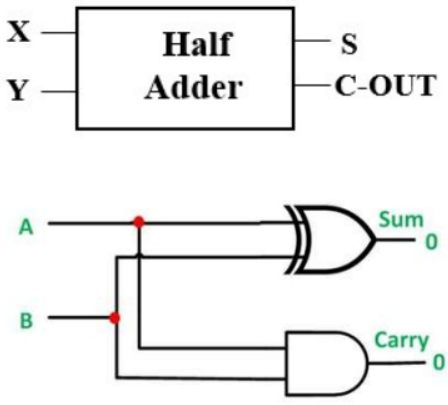
ADDERS

In electronics, an adder is a digital circuit that performs addition of two or more numbers. Adders can be constructed for many numerical representations, such as Binary-coded decimal or excess-3. Adders are different types in generally

➢ HALF ADDER

a) The half adder adds two single binary digits A and B.

b) It has two outputs, sum (S) and carry (C).

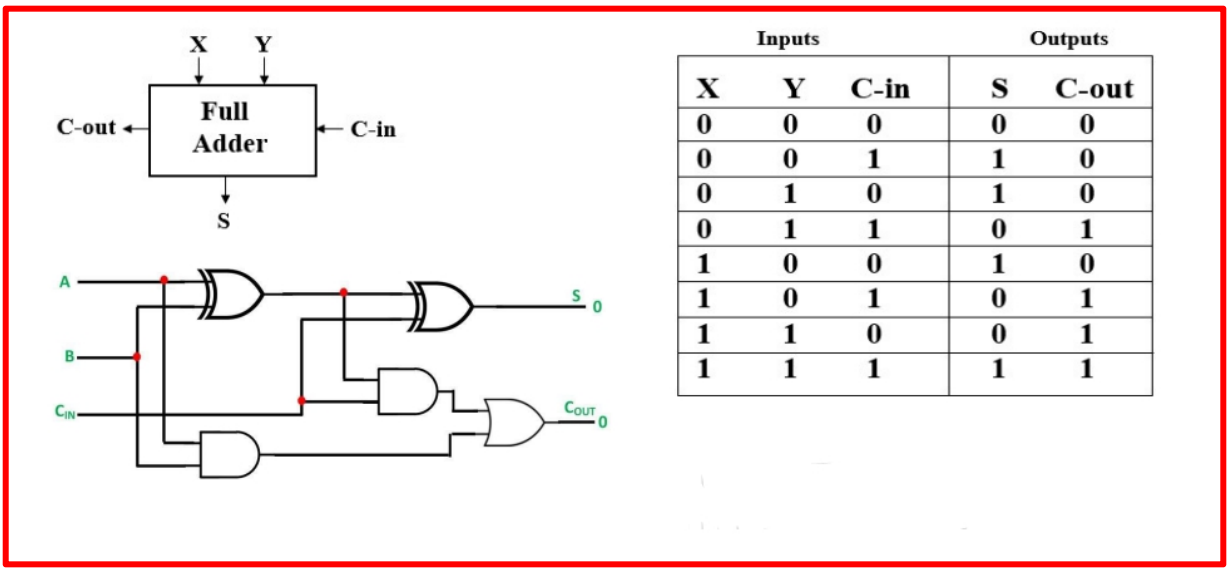


Inputs		Outputs	
X	Y	S	C-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

➢ FULL ADDER

a) Adding two single-bit binary values, X, Y along with a carry input bit C-in and

b) produces a sum bit S and a carry out C-out bit.



Inputs			Outputs	
X	Y	C-in	S	C-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

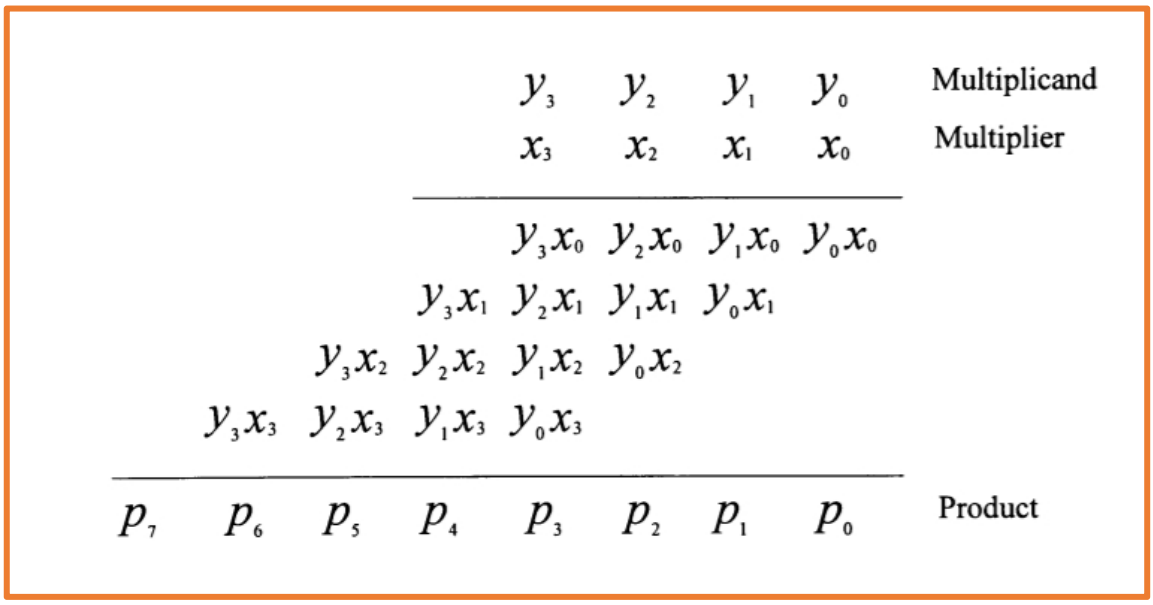
➢ MULTIPLICATION ALGORITHM

There are a number of algorithms used for multiplication . The 3-bit recoding algorithm is one of the most well known . It is used in the design of many kinds of hardware and software multipliers. This algorithm is used to reduce the number of partial product rows by about half, so, the speed of multiplication increases significantly 10 and the chip area is reduced. The 3-bit recoding algorithm is also called the Modified Booth's Algorithm and was developed from Booth's algorithm . A number of other multiple-bit recoding algorithms for multiplication have been developed.

Booth Algorithm

1) Set the Multiplicand and Multiplier binary bits as M and Q, respectively.
2) Initially, we set the AC and Qn + 1 registers value to 0.
3) SC represents the number of Multiplier bits (Q), and it is a sequence counter that is continuously decremented till equal to the number of bits (n) or reached to 0.
4) A Qn represents the last bit of the Q, and the Qn+1 shows the incremented bit of Qn by 1.
5) On each cycle of the booth algorithm, Qn and Qn + 1 bits will be checked on the following parameters as follows:
6) When two bits Qn and Qn + 1 are 00 or 11, we simply perform the arithmetic shift right operation (ashr) to the partial product AC. And the bits of Qn and Qn + 1 is incremented by 1 bit.
7) If the bits of Qn and Qn + 1 is shows to 01, the multiplicand bits (M) will be added to the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.
8) If the bits of Qn and Qn + 1 is shows to 10, the multiplicand bits (M) will be subtracted from the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.
9) The operation continuously works till we reached n - 1 bit in the booth algorithm.
10) Results of the Multiplication binary bits will be stored in the AC and QR registers.

The process of digital multiplication is based on addition, and many of the techniques useful in addition carry over to multiplication



Example:

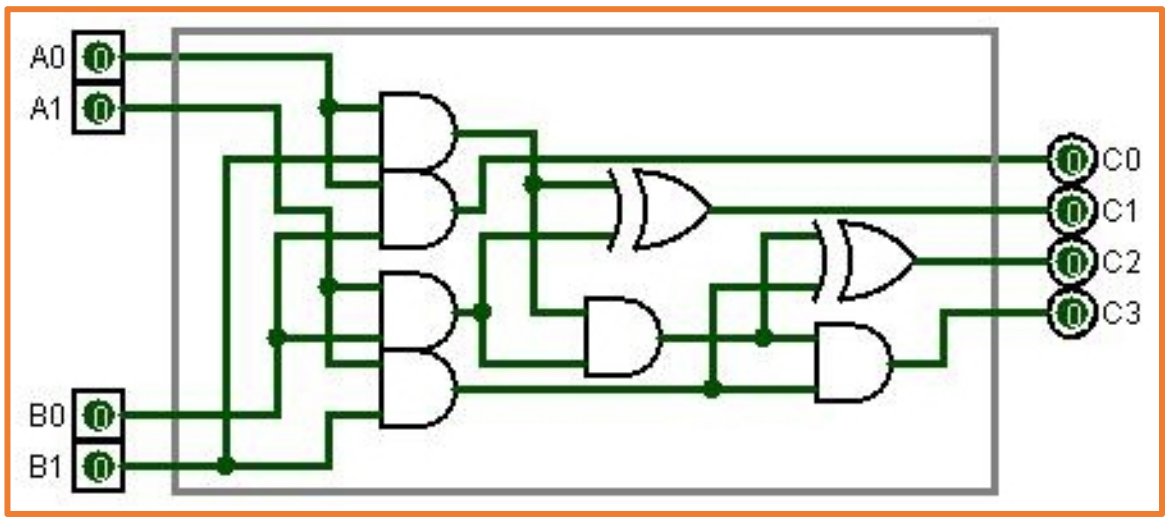
Let us take an example of multiplying two binary numbers as follows. The process is similar to multiplying two decimal numbers, with a difference that the resulting numbers are all binary.

110 = 6
X 011 = 3
---
1 1 0 ; 110 X 1 (Shifted one position left)
1 1 0 x ; 110 X 1 (Shifted one position left)
0 0 0 x x ; 110 X 0 (Shifted one position left)
---
1 0 0 1 0 = 18

2 bit multiplier :

A 2-bit multiplier circuit that performs multiplication through a series of additions. For example, suppose we want to multiply 2 * 1.

Instead of building a multiplier circuit, we can instead use an adder and perform 2 * 1 by adding 1 + 1. The first number indicates how many times the second number is added to itself.



Multiplier Bits		Multiple of Multiplicand	
Yi+1	**Y1**	**Multiples**	**Implementation**
0	0	0	0
0	1	1	x
1	0	2	Shift left X by 1
1	1	3	(Shift left X by 1) + X

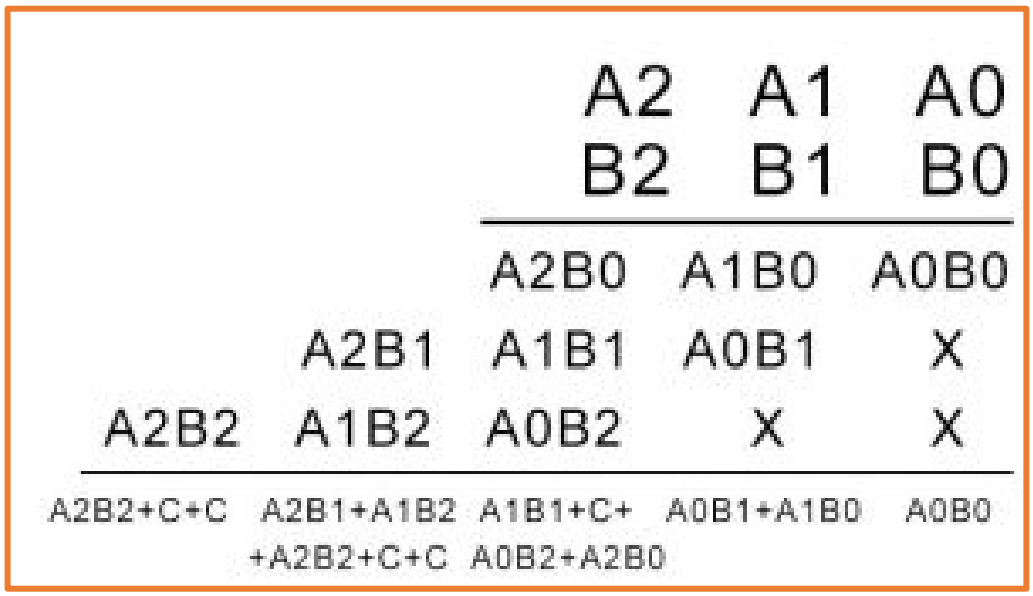
Truth Table for 2 Bit Multiplier

Term	A0	A1	B0	B1	=>	F0	F1	F2	F3
0	0	0	0	0		0	0	0	0
1	0	0	0	1		0	0	0	0
2	0	0	1	0		0	0	0	0
3	0	0	1	1		0	0	0	0
4	0	1	0	0		0	0	0	0
5	0	1	0	1		0	0	0	1
6	0	1	1	0		0	0	1	0
7	0	1	1	1		0	0	1	1
8	1	0	0	0		0	0	0	0
9	1	0	0	1		0	0	1	0
10	1	0	1	0		0	1	0	0
11	1	0	1	1		0	1	1	0
12	1	1	0	0		0	0	0	0
13	1	1	0	1		0	0	1	1
14	1	1	1	0		0	1	1	0
15	1	1	1	1		1	0	0	1

3 bit multiplier :

3 bit multiplier works in a similar way.

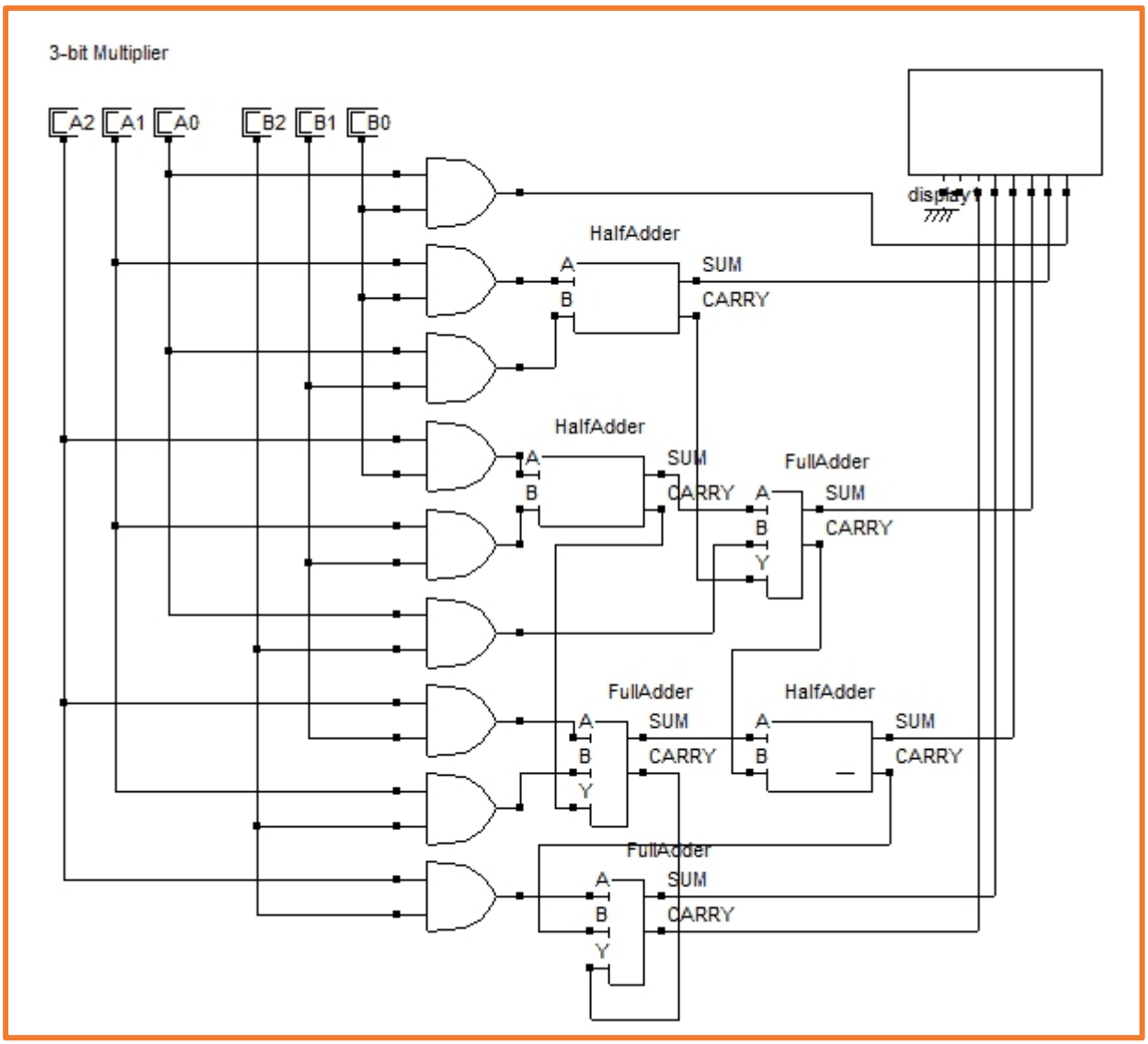
Consider two general 3-bit binary numbers A2A1A0 and B2B1B0. Multiplying the two numbers with each other using standard binary arithmetic rules, we get the following equation.



Adding A2B0 and A1B1 will give rise to one carry, adding the sum obtained from that, and the carry obtained from adding A1B0 and A0B1 to A0B2 will give rise to another carry. Thus, two carries are generated and are carried over to the addition between A2B1 and A1B2, where two more carries are created similarly.

Hence the resulting circuit will contain nine AND gates, three half adders, and three full adders.

Diagram:


3-bit Multiplier
A2
A1
A0
B2
B1
B0
display
HalfAdder
A
B
SUM
CARRY
HalfAdder
A
B
SUM
CARRY
FullAdder
A
B
Y
SUM
CARRY
FullAdder
A
B
Y
SUM
CARRY
HalfAdder
A
B
SUM
CARRY
FullAdder
A
B
Y
SUM
CARRY

Truth table :

A2	A1	A0	B2	B1	B0	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	1
0	0	1	0	1	0	0	0	0	0	1	0
0	0	1	0	1	1	0	0	0	0	1	1
0	0	1	1	0	0	0	0	0	1	0	0
0	0	1	1	0	1	0	0	0	1	0	1
0	0	1	1	1	0	0	0	0	1	1	0
0	0	1	1	1	1	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	0	1	0	0
0	1	0	0	1	1	0	0	0	1	1	0
0	1	0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	1	0	0	1	0	1	0
0	1	0	1	1	0	0	0	1	1	0	0
0	1	0	1	1	1	0	0	1	1	1	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	1	1
0	1	1	0	1	0	0	0	0	1	1	0
0	1	1	0	1	1	0	0	1	0	0	1
0	1	1	1	0	0	0	0	1	1	0	0
0	1	1	1	0	1	0	0	1	1	1	1
0	1	1	1	1	0	0	1	0	0	1	0
0	1	1	1	1	1	0	1	0	1	0	1

➢ Different Types of Multipliers

The different types of multipliers are,

Booth Multiplier

The function of the booth's multiplier is, to multiply 2 signed binary numbers which are represented in 2's complement form. The advantages of booths multipliers are Minimum complex, Multiplication is speeded up. The disadvantages of booths multipliers are Power consumption is high.

Array Multiplier

The multiplier circuit is based on the add shift algorithm. The main advantage of the array multiplier is it's simple in design and regular in shape. The disadvantage of an array multiplier is the delay is high and high power consumption.

Combinational Multiplier

The combinational multiplier performs multiplication of two unsigned binary numbers. The advantage of a combinational multiplier is that it can easily generate intermediate products. The main disadvantage of the combinational multiplier is it occupies large areas.

Sequential Multiplier

Multiplication is divided into the sequence of steps, where the partial product generated is added to the accumulator partial sum now is shifted to the next step. The advantage of this is it occupies less area. The disadvantage os a sequential multiplier is it is a slow process.

Wallace tree Multiplier

It reduces the number of partial products and uses carry select adder for the addition of partial products. The advantage of the Wallace tree multiplier is a high speed and medium complex design. The main disadvantage of Wallace tree multiplier is the layout design is irregular and occupies a larger area.

Shift and Add Multiplier

It is similar to the normal multiplication process, which we do in mathematics, from array multiplier flow chat where X = Multiplicand; Y = Multiplier; A = Accumulator, Q = Quotient. Firstly Q is checked if it's 1 or no if it is 1 then add A and B and shift A_Q arithmetic right, else if it is not 1 directly shift A_Q arithmetic right and decrement N by 1, in the next step check if N is 0 or no. If N not 0 repeats from Q=0 step else terminate the process.

ARRAY MULTIPLIERS

An **array multiplier** is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders. This array is used for the nearly simultaneous addition of the various product terms involved. To form the various product terms, an array of AND gates is used before the Adder array.

The design structure of the array Multiplier is regular, it is based on the add shift algorithm principle.

Partial product = the multiplicand * multiplier bit…….(2)

where AND gates are used for the product, the summation is done using Full Adders and Half Adders where the partial product is shifted according to their bit orders.

➢ APPLICATIONS

These are most commonly used in various applications especially in the field of digital signal processing to perform the various algorithms.

Commercial applications like computers, mobiles, high speed calculators and some general purpose processors require binary multipliers.

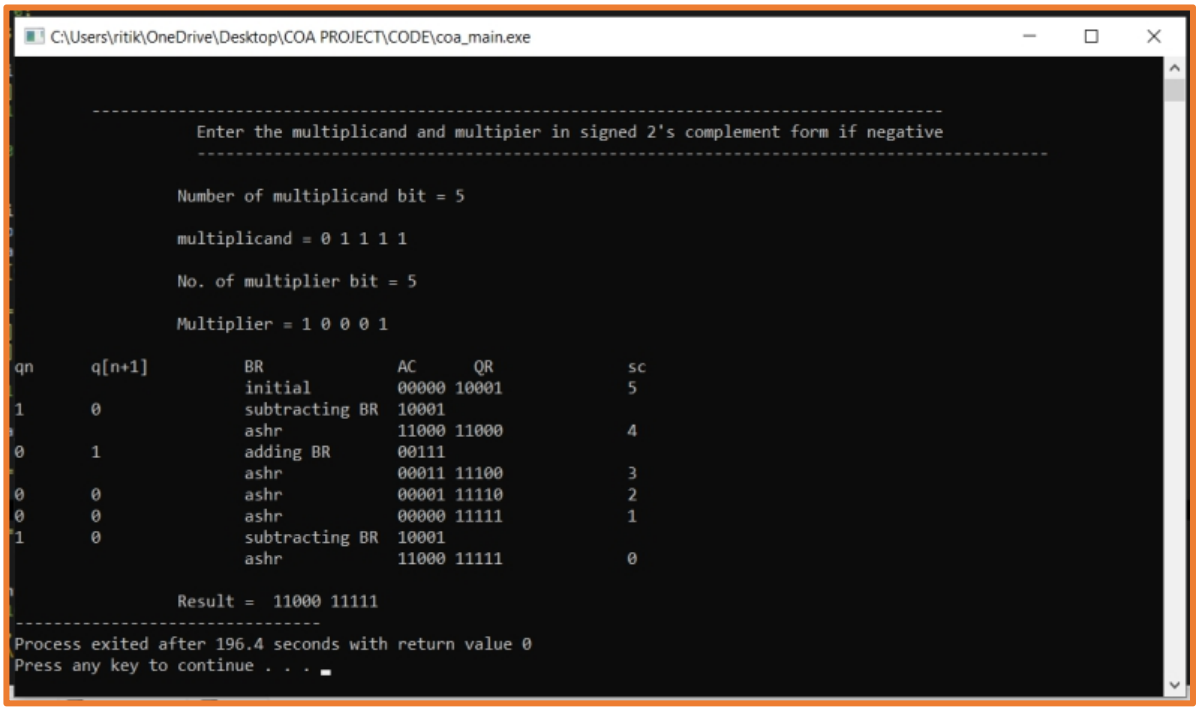
➢ CODE:

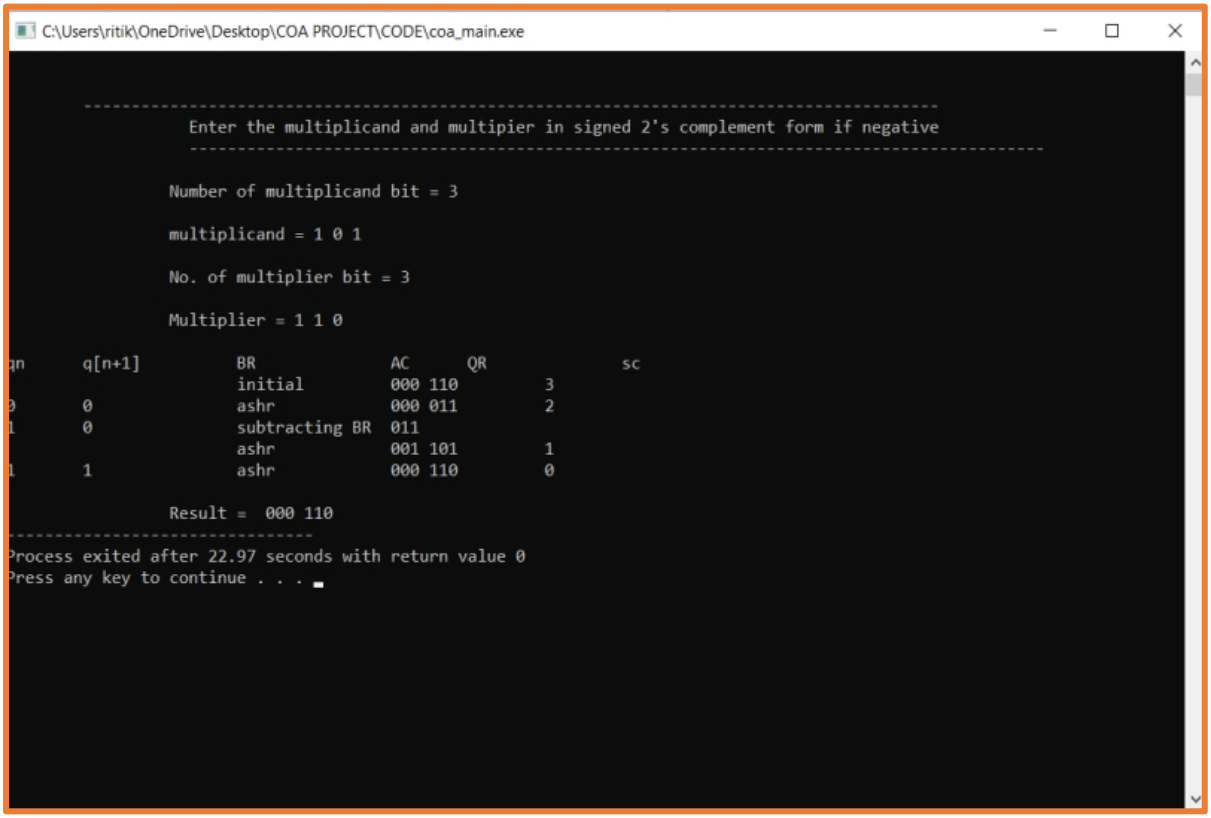
```
#include<iostream>
using namespace std;
void add(int m1[], int m2[], int m3);
void complement(int m1[], int a1) {
  int i;
  int m2[8] = {0};
  m2[0] = 1;
  for (i = 0; i < a1; i++) {
    m1[i] = (m1[i] + 1) % 2;
  }
  add(m1, m2, a1);
}
void add(int z1[], int m2[], int m3) {
  int i, v1 = 0;
  for (i = 0; i < m3; i++) {
    z1[i] = z1[i] + m2[i] + v1;
    if (z1[i] > 1) {
      z1[i] = z1[i] % 2;
      v1 = 1;
    }else
      v1 = 0;
    }
  }
  void ashr(int z1[], int z2[], int &z3, int m3) {
    int temp, i;
    temp = z1[0];
    z3 = z2[0];
    cout << "\t\tashr\t\t";
    for (i = 0; i < m3 - 1; i++) {
      z1[i] = z1[i + 1];
      z2[i] = z2[i + 1];
```

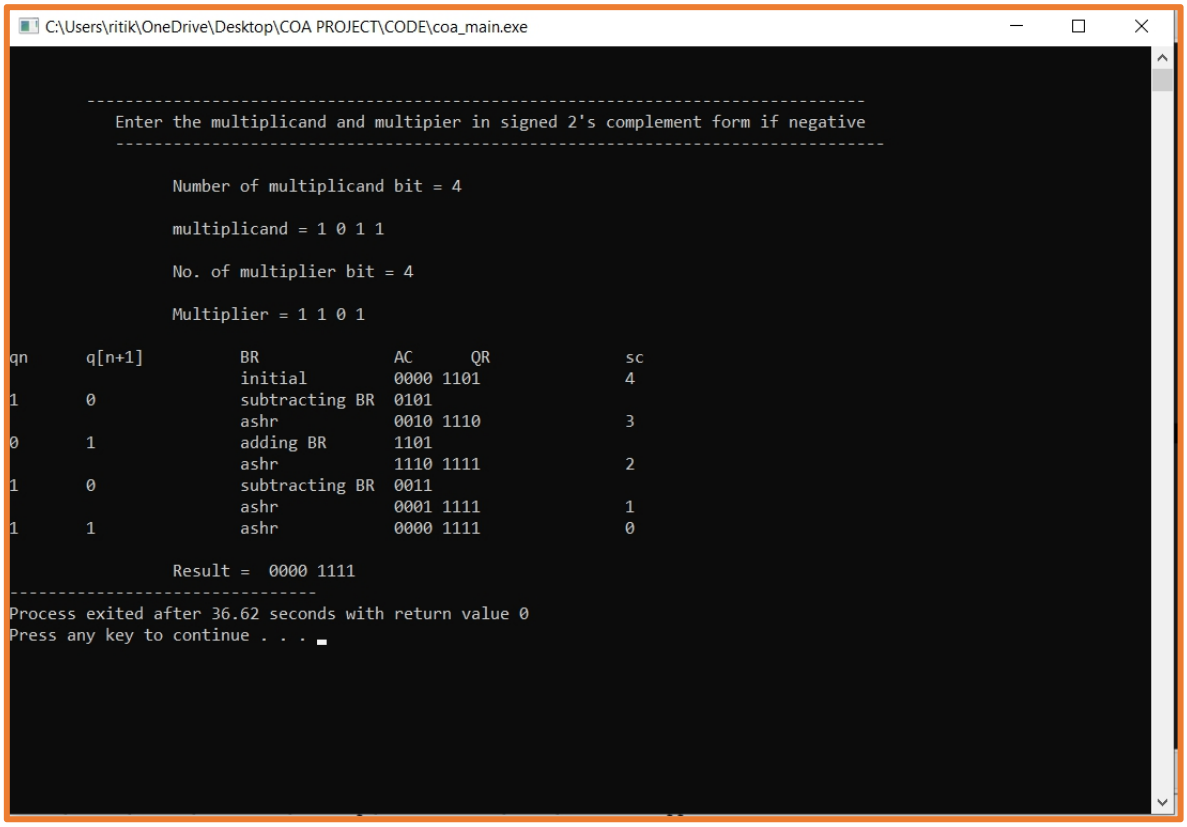
```
    }
    z2[m3 - 1] = temp;
  }
  void display(int z1[], int z2[], int v3) {
    int i;
    for (i = v3 - 1; i >= 0; i--)
      cout << z1[i];
    cout << " ";
    for (i = v3 - 1; i >= 0; i--)
      cout << z2[i];
  }
  int main(int argc, char **argv) {
    int f1[10], ff[10], z2[10], yz, z1[10] = { 0 };
    int o, v3, i, z3, temp;
    cout<<"\n\n\t--------------------------------------------------------------------------------";
    cout << "\n\t   Enter the multiplicand and multipier in signed 2's complement form if
negative";
    cout<<"\n\t   --------------------------------------------------------------------------------";
          cout << "\n\n\t\t Number of multiplicand bit = ";
    cin >> o;
    cout << "\n\t\t multiplicand = ";
    for (i = o - 1; i >= 0; i--)
      cin >> ff[i]; //multiplicand
    for (i = o - 1; i >= 0; i--)
      f1[i] = ff[i];
    complement(f1, o);
    cout << "\n\t\t No. of multiplier bit = ";
    cin >> v3;
    yz = v3;
    cout << "\n\t\t Multiplier = ";
    for (i = v3 - 1; i >= 0; i--)
      cin >> z2[i];
      z3 = 0;
      temp = 0;
      cout << "\nqn\tq[n+1]\t\tBR\t\tAC\tQR\t\tsc\n";
      cout << "\t\t\tinitial\t\t";
```

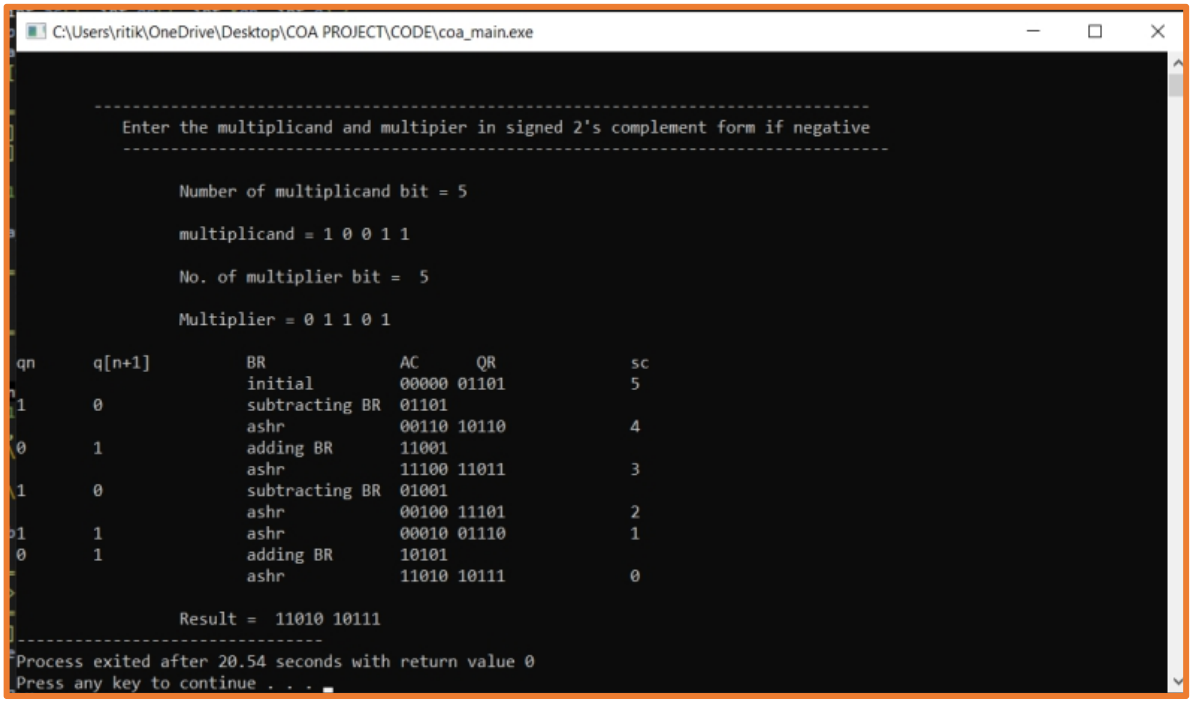
```
    display(z1, z2, v3);
    cout << "\t\t" << yz << "\n";
    while (yz != 0) {
      cout << z2[0] << "\t" << z3;
      if ((z3 + z2[0]) == 1) {
        if (temp == 0) {
          add(z1, f1, v3);
          cout << "\t\tsubtracting BR\t";
          for (i = v3 - 1; i >= 0; i--)
            cout << z1[i];
          temp = 1;
        }
      else if (temp == 1) {
        add(z1, ff, v3);
        cout << "\t\tadding BR\t";
        for (i = v3 - 1; i >= 0; i--)
          cout << z1[i];
          temp = 0;
      }
      cout << "\n\t";
      ashr(z1, z2, z3, v3);
    }
    else if (z3 - z2[0] == 0)
      ashr(z1, z2, z3, v3);
      display(z1, z2, v3);
      cout << "\t";
      yz--;
      cout << "\t" << yz << "\n";
}
cout << "\n\t\t Result =  ";
display(z1, z2, v3);
}
```

➢ OUTPUT:


C:\Users\ritik\OneDrive\Desktop\COA PROJECT\CODE\coa_main.exe
Enter the multiplicand and multipier in signed 2's complement form if negative
Number of multiplicand bit = 5
multiplicand = 0 1 1 1 1
No. of multiplier bit = 5
Multiplier = 1 0 0 0 1
qn q[n+1] BR AC QR sc
initial 00000 10001 5
1 0 subtracting BR 10001
ashr 11000 11000 4
0 1 adding BR 00111
ashr 00011 11100 3
0 0 ashr 00001 11110 2
0 0 ashr 00000 11111 1
1 0 subtracting BR 10001
ashr 11000 11111 0
Result = 11000 11111
Process exited after 196.4 seconds with return value 0
Press any key to continue . . .


C:\Users\ritik\OneDrive\Desktop\COA PROJECT\CODE\coa_main.exe
Enter the multiplicand and multipier in signed 2's complement form if negative
Number of multiplicand bit = 3
multiplicand = 1 0 1
No. of multiplier bit = 3
Multiplier = 1 1 0
qn q[n+1] BR AC QR sc
initial 000 110 3
0 0 ashr 000 011 2
1 0 subtracting BR 011
ashr 001 101 1
1 1 ashr 000 110 0
Result = 000 110
Process exited after 22.97 seconds with return value 0
Press any key to continue . . .


C:\Users\ritik\OneDrive\Desktop\COA PROJECT\CODE\coa_main.exe
Enter the multiplicand and multipier in signed 2's complement form if negative
Number of multiplicand bit = 4
multiplicand = 1 0 1 1
No. of multiplier bit = 4
Multiplier = 1 1 0 1
qn q[n+1] BR AC QR sc
initial 0000 1101 4
1 0 subtracting BR 0101
ashr 0010 1110 3
0 1 adding BR 1101
ashr 1110 1111 2
1 0 subtracting BR 0011
ashr 0001 1111 1
1 1 ashr 0000 1111 0
Result = 0000 1111
Process exited after 36.62 seconds with return value 0
Press any key to continue . . .


C:\Users\ritik\OneDrive\Desktop\COA PROJECT\CODE\coa_main.exe
Enter the multiplicand and multipier in signed 2's complement form if negative
Number of multiplicand bit = 5
multiplicand = 1 0 0 1 1
No. of multiplier bit = 5
Multiplier = 0 1 1 0 1
qn q[n+1] BR AC QR sc
initial 00000 01101 5
1 0 subtracting BR 01101
ashr 00110 10110 4
0 1 adding BR 11001
ashr 11100 11011 3
1 0 subtracting BR 01001
ashr 00100 11101 2
1 1 ashr 00010 01110 1
0 1 adding BR 10101
ashr 11010 10111 0
Result = 11010 10111
Process exited after 20.54 seconds with return value 0
Press any key to continue . . .

➢ Bibliography:

- https://en.wikipedia.org/wiki/Binary_multiplier#:~:text=A%20binary%20multiplier%20is%20an,to%20implement%20a%20digital%20multiplier.
- https://en.wikipedia.org/wiki/Multiplication_algorithm
- https://vlsiuniverse.blogspot.com/2013/05/binary-multiplier.html#:~:text=Binary%20multiplication%20process%3A%20A%20Binary,provide%20the%20result%20as%20output.&text=The%20two%20numbers%20A1A0%20and,a%204%2Dbit%20output%20P3P2P1P0.
- https://www.cs.columbia.edu/~martha/courses/3827/sp11/slides/2bit_multiplier_soln.pdf
- https://www.electricaltechnology.org/2018/05/binary-multiplier-types-binary-multiplication-calculator.html
- https://technobyte.org/multiplier-2-bit-3-bit-digital/
- https://electronics.stackexchange.com/questions/99813/3-bit-multipliers-how-do-they-work/99837
- https://inst.eecs.berkeley.edu/~eecs151/sp18/files/Lecture21.pdf
- https://www.electronicshub.org/binary-multiplication/
- https://www.sciencedirect.com/topics/engineering/binary-multiplication
- https://en.wikipedia.org/wiki/Binary_multiplier