# 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 WORD MULTIPLIER**

* **Objective:** The purpose of this project is to model and simulate hardware multiplication unit using Booth algorithm multiplication algorithm.
* **Introduction**

A **binary multiplier** is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers. 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.

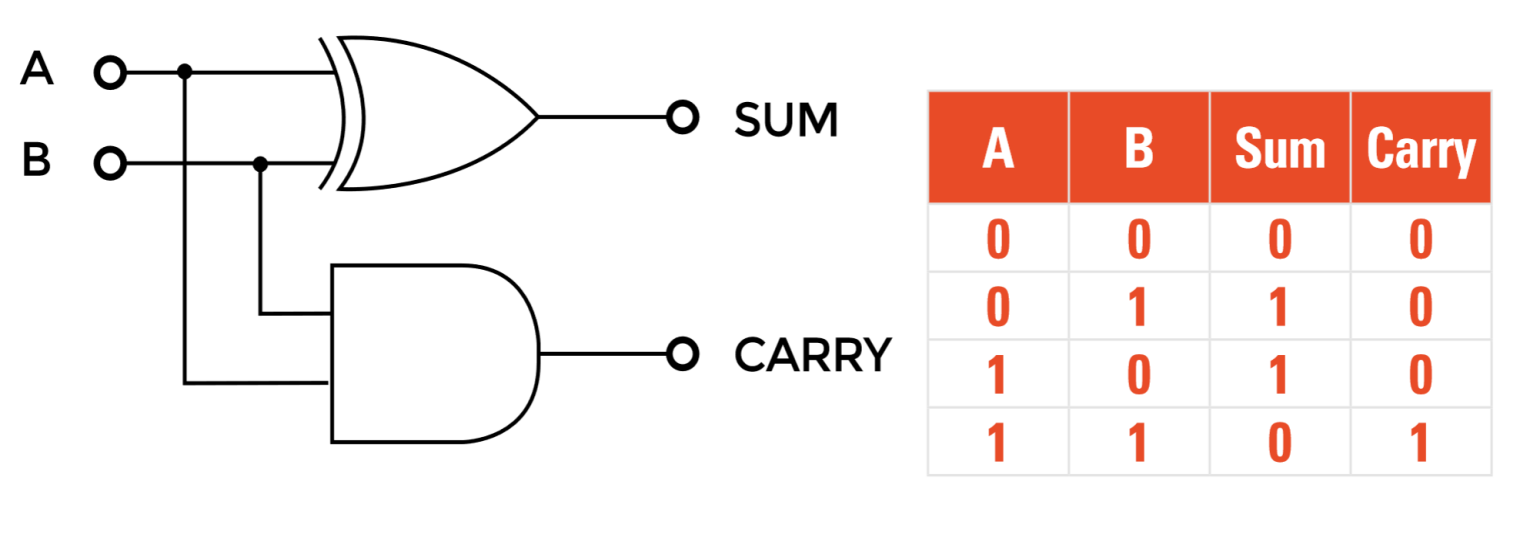
Mutipliers can be classified as hardware multipliers and software multipliers. In older digital systems, there was no hardware multiplier and multiplication was implemented with a micro program. The micro program needed many micro instruction cycles to complete the multiplication process, which make 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

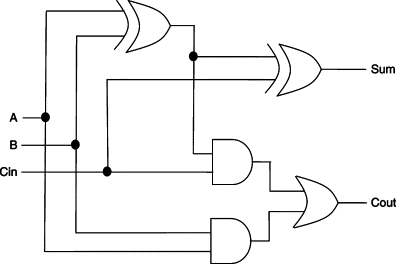
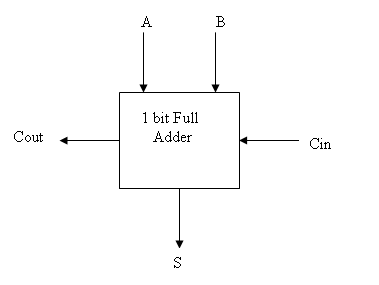
* HALF ADDER 

1. The half adder adds two single binary digits A and B. 
2. It has two outputs, sum (S) and carry (C).



* FULL ADDER 

1. A full adder adds three one-bit numbers, often written as A, B, and Cin. 
2. A and B are the operands, and Cin is a bit carried in from the previous less significant stage.

v 

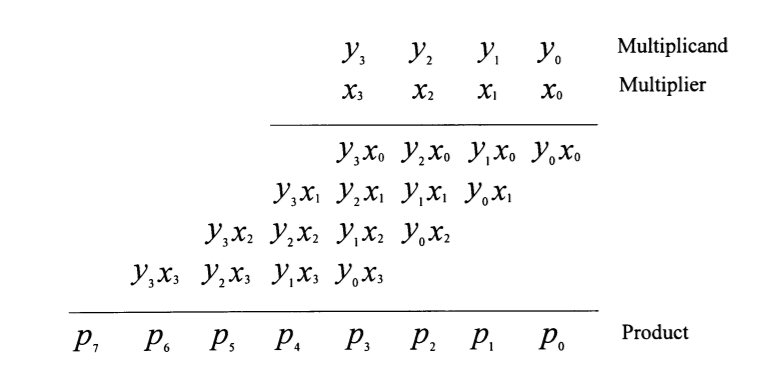
* **MULTIPLICATION ALGORITHM**

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)**

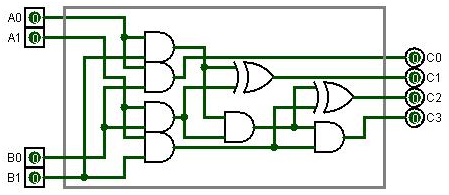
**0 0 0 x x                 ; 110 X 0 (Shifted one position left)**

------------------------------

**1 0 0 1 0 = 18**

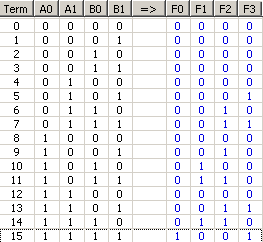
**2 bit multiplier :**  
A 2-bit multiplier circuit that performs multiplication through a series of additions. For example, suppose we want to multiply 2 \* 1.

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.



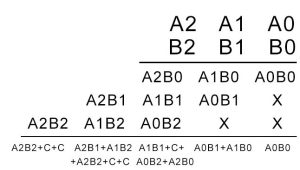
|  |  |  |  |
| --- | --- | --- | --- |
| ****Multiplier Bits**** | | ****Multiple of Multiplicand**** | |
| ****Yi+1**** | ****Y1**** | ****Multiples**** | ****Implementation**** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | x |
| 1 | 0 | 2 | Shift left X by 1 |
| 1 | 1 | 3 | (Shift left X by 1) + X |

**Truth Table for 2 Bit Multiplier**



**3 bit multiplier :**  
3 bit multiplier works in a similar way.

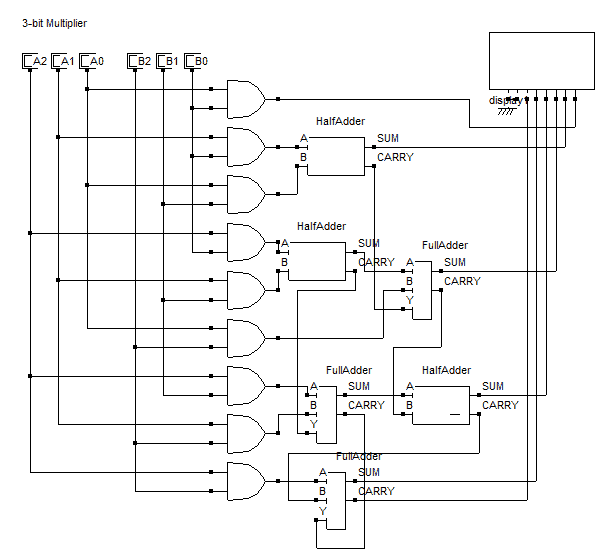
Consider two general 3-bit binary numbers A2A1A0 and B2B1B0. Multiplying the two numbers with each other using [standard binary arithmetic rules](https://technobyte.org/binary-arithmetic-rules/" \t "https://technobyte.org/multiplier-2-bit-3-bit-digital/_blank), we get the following equation.

[](https://i1.wp.com/technobyte.org/wp-content/uploads/2018/09/3-bit-multiplier-digital-e1538309492370.jpg?ssl=1)

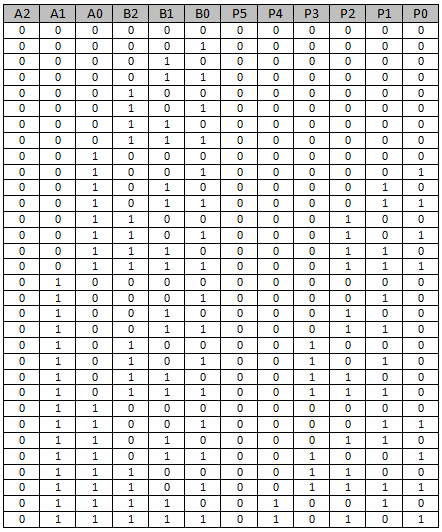
Adding A2B0 and A1B1 will give rise to one carry, adding the sum obtained from that, and the carry obtained from adding A1B0 and A0B1 to A0B2 will give rise to another carry. Thus, two carries are generated and are carried over to the addition between A2B1 and A1B2, where two more carries are created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

Hence the resulting circuit will contain nine AND gates, three half adders, and three [full adders](https://technobyte.org/2018/10/half-adder-full-adder-half-subtractor-full-subtractor" \l "How_to_design_a_Full_Adder_circuit" \t "https://technobyte.org/multiplier-2-bit-3-bit-digital/_blank).

**Diagram:**



**Truth table :**



* **CODE:**

#include<iostream>

using namespace std;

void add(int a[], int x[], int q);

void complement(int a[], int n) {

int i;

int x[8] = {0};

x[0] = 1;

for (i = 0; i < n; i++) {

a[i] = (a[i] + 1) % 2;

}

add(a, x, n);

}

void add(int ac[], int x[], int q) {

int i, c = 0;

for (i = 0; i < q; i++) {

ac[i] = ac[i] + x[i] + c;

if (ac[i] > 1) {

ac[i] = ac[i] % 2;

c = 1;

}else

c = 0;

}

}

void ashr(int ac[], int qr[], int &qn, int q) {

int temp, i;

temp = ac[0];

qn = qr[0];

cout << "\t\tashr\t\t";

for (i = 0; i < q - 1; i++) {

ac[i] = ac[i + 1];

qr[i] = qr[i + 1];

}

qr[q - 1] = temp;

}

void display(int ac[], int qr[], int qrn) {

int i;

for (i = qrn - 1; i >= 0; i--)

cout << ac[i];

cout << " ";

for (i = qrn - 1; i >= 0; i--)

cout << qr[i];

}

int main(int argc, char \*\*argv) {

int mt[10], br[10], qr[10], sc, ac[10] = { 0 };

int brn, qrn, i, qn, temp;

cout<<"\n\n\t-----------------------------------------------------------------------------------------";

cout << "\n\t\t Enter the multiplicand and multipier in signed 2's complement form if negative";

cout<<"\n\t\t -----------------------------------------------------------------------------------------";

cout << "\n\n\t\t Number of multiplicand bit = ";

cin >> brn;

cout << "\n\t\t multiplicand = ";

for (i = brn - 1; i >= 0; i--)

cin >> br[i]; //multiplicand

for (i = brn - 1; i >= 0; i--)

mt[i] = br[i];

complement(mt, brn);

cout << "\n\t\t No. of multiplier bit = ";

cin >> qrn;

sc = qrn;

cout << "\n\t\t Multiplier = ";

for (i = qrn - 1; i >= 0; i--)

cin >> qr[i];

qn = 0;

temp = 0;

cout << "\nqn\tq[n+1]\t\tBR\t\tAC\tQR\t\tsc\n";

cout << "\t\t\tinitial\t\t";

display(ac, qr, qrn);

cout << "\t\t" << sc << "\n";

while (sc != 0) {

cout << qr[0] << "\t" << qn;

if ((qn + qr[0]) == 1) {

if (temp == 0) {

add(ac, mt, qrn);

cout << "\t\tsubtracting BR\t";

for (i = qrn - 1; i >= 0; i--)

cout << ac[i];

temp = 1;

}

else if (temp == 1) {

add(ac, br, qrn);

cout << "\t\tadding BR\t";

for (i = qrn - 1; i >= 0; i--)

cout << ac[i];

temp = 0;

}

cout << "\n\t";

ashr(ac, qr, qn, qrn);

}

else if (qn - qr[0] == 0)

ashr(ac, qr, qn, qrn);

display(ac, qr, qrn);

cout << "\t";

sc--;

cout << "\t" << sc << "\n";

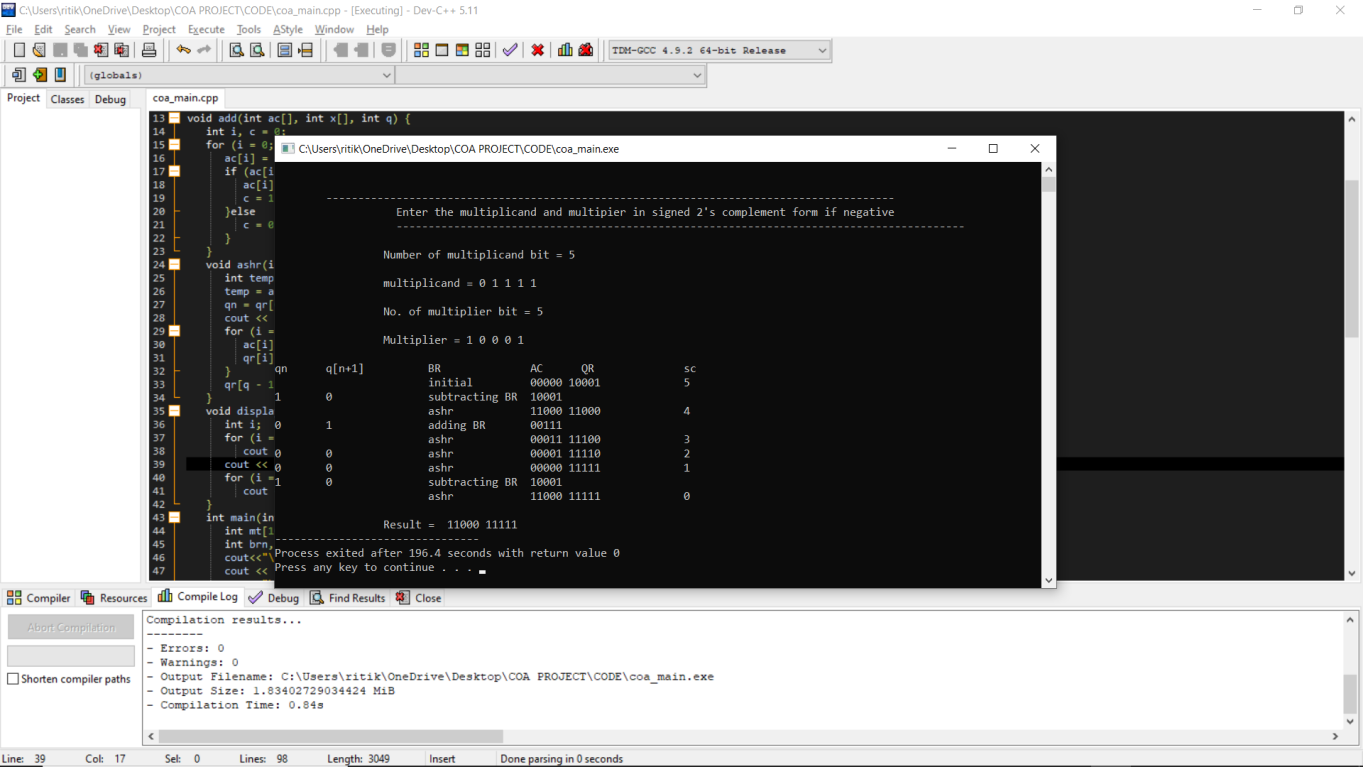
}

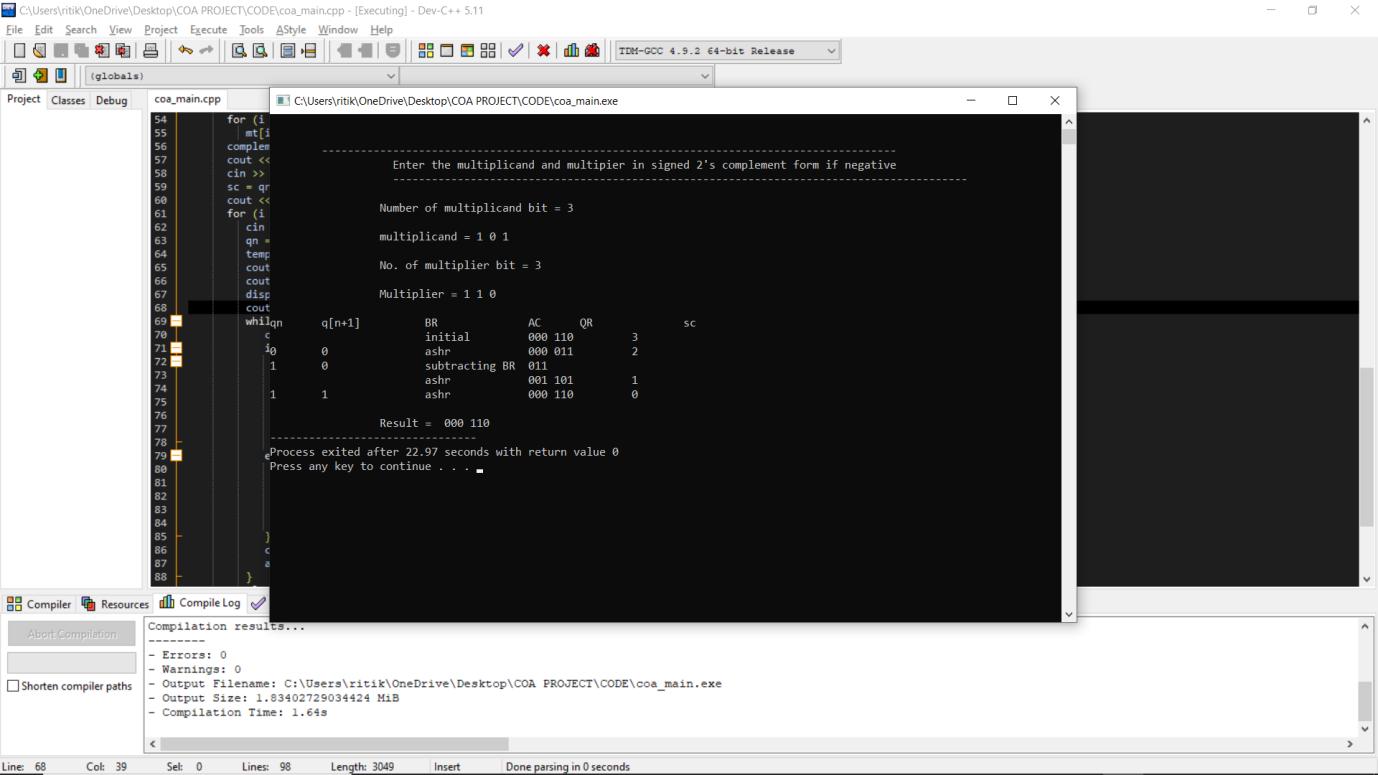
cout << "\n\t\t Result = ";

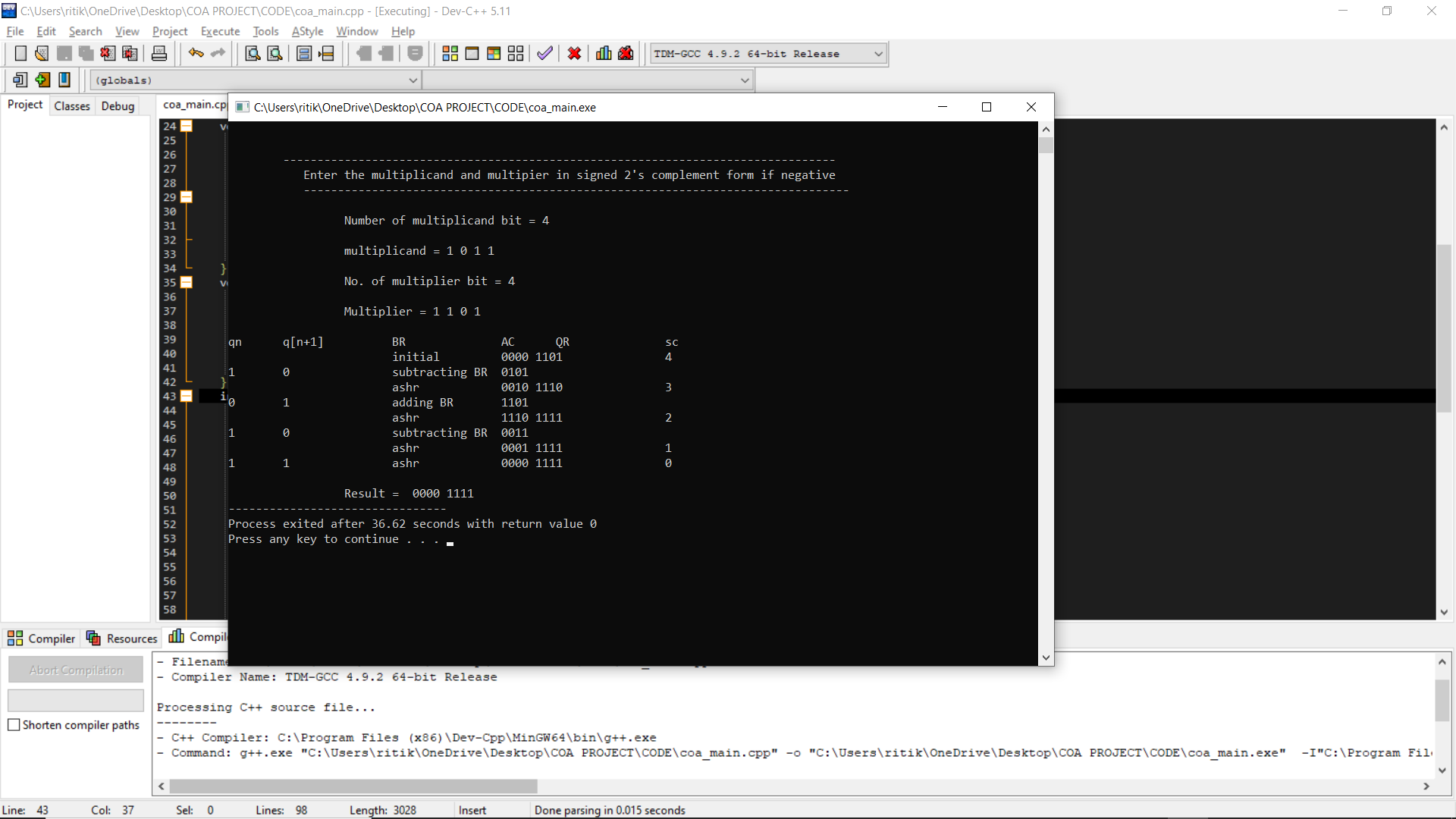
display(ac, qr, qrn);

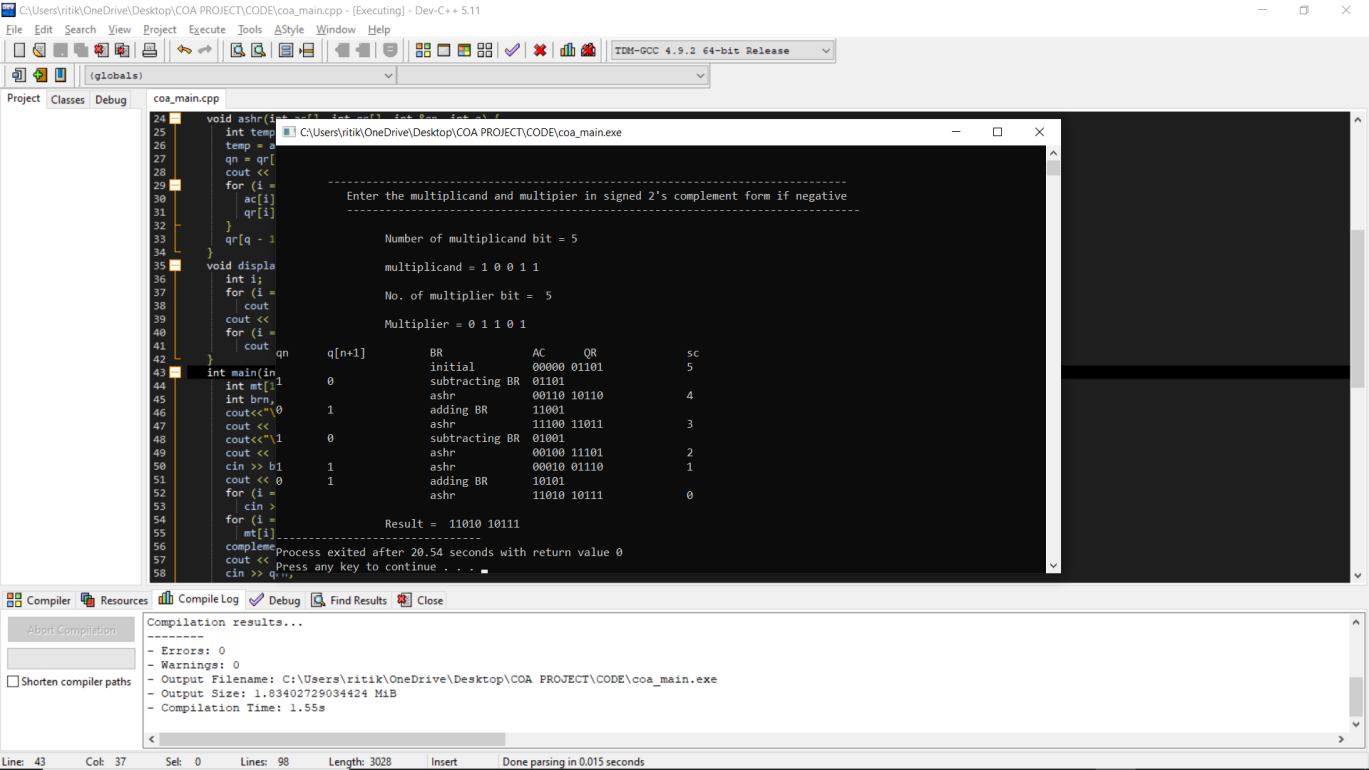
}

**OUTPUT:**

****

****

****

****

* **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>