**: Roll No.:**

**Experiment / assignment / tutorial No.\_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**nature of the Staff In-charge with date**

**Batch: B3 Roll No.: 121**

**Experiment / assignment / tutorial No.\_\_\_2\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** To study and implement Booth’s Multiplication Algorithm. |

**AIM:** Booth’s Algorithm for Multiplication

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment: (Mention CO/CO’s attained here)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

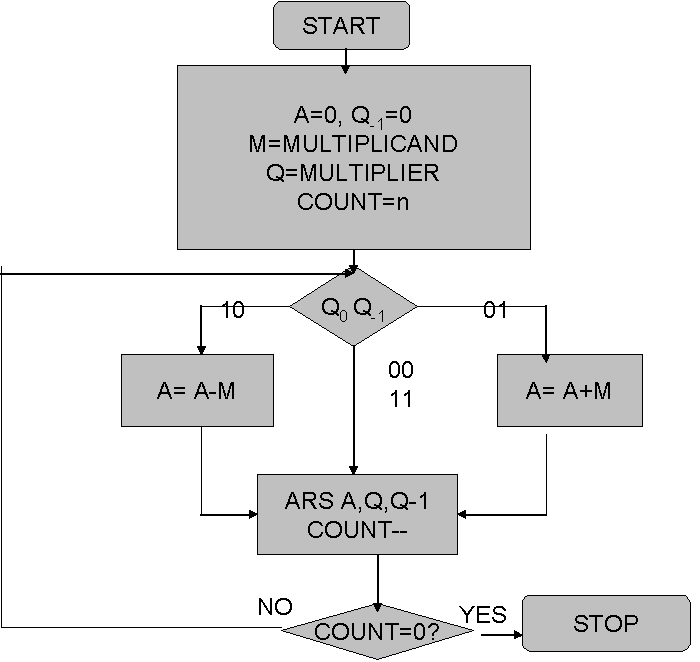
3. Dr. M. Usha, T. S. Srikanth, “Computer System Architecture and Organization”, First Edition, Wiley-India.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

It is a powerful algorithm for signed number multiplication which generates a 2n bit product and treats both positive and negative numbers uniformly. Also the efficiency of the algorithm is good due to the fact that, block of 1’s and 0’s are skipped over and subtraction/addition is only done if pair contains 10 or 01

**Flowchart:**

****

**Design Steps**:

1. Start
2. Get the multiplicand (M) and Multiplier (Q) from the user
3. Initialize A= Q-1 =0
4. Convert M and Q into binar
5. Compare Q0 andQ-1 and perform the respective operation.

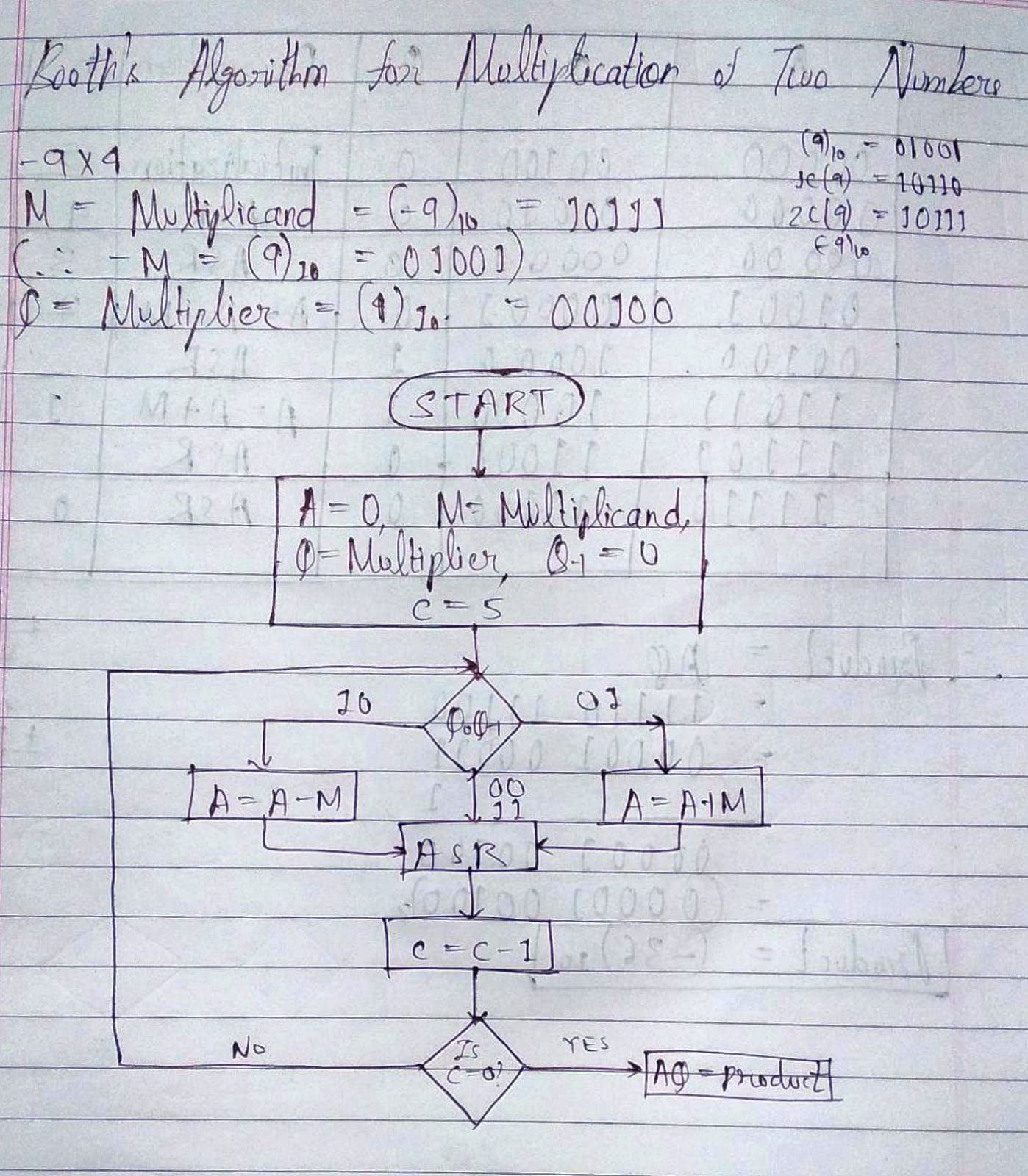
|  |  |
| --- | --- |
| **Q0 Q-1** | **Operation** |
| 00/11 | Arithmetic right shift |
| 01 | A+M and Arithmetic right shift |
| 10 | A-M and Arithmetic right shift |

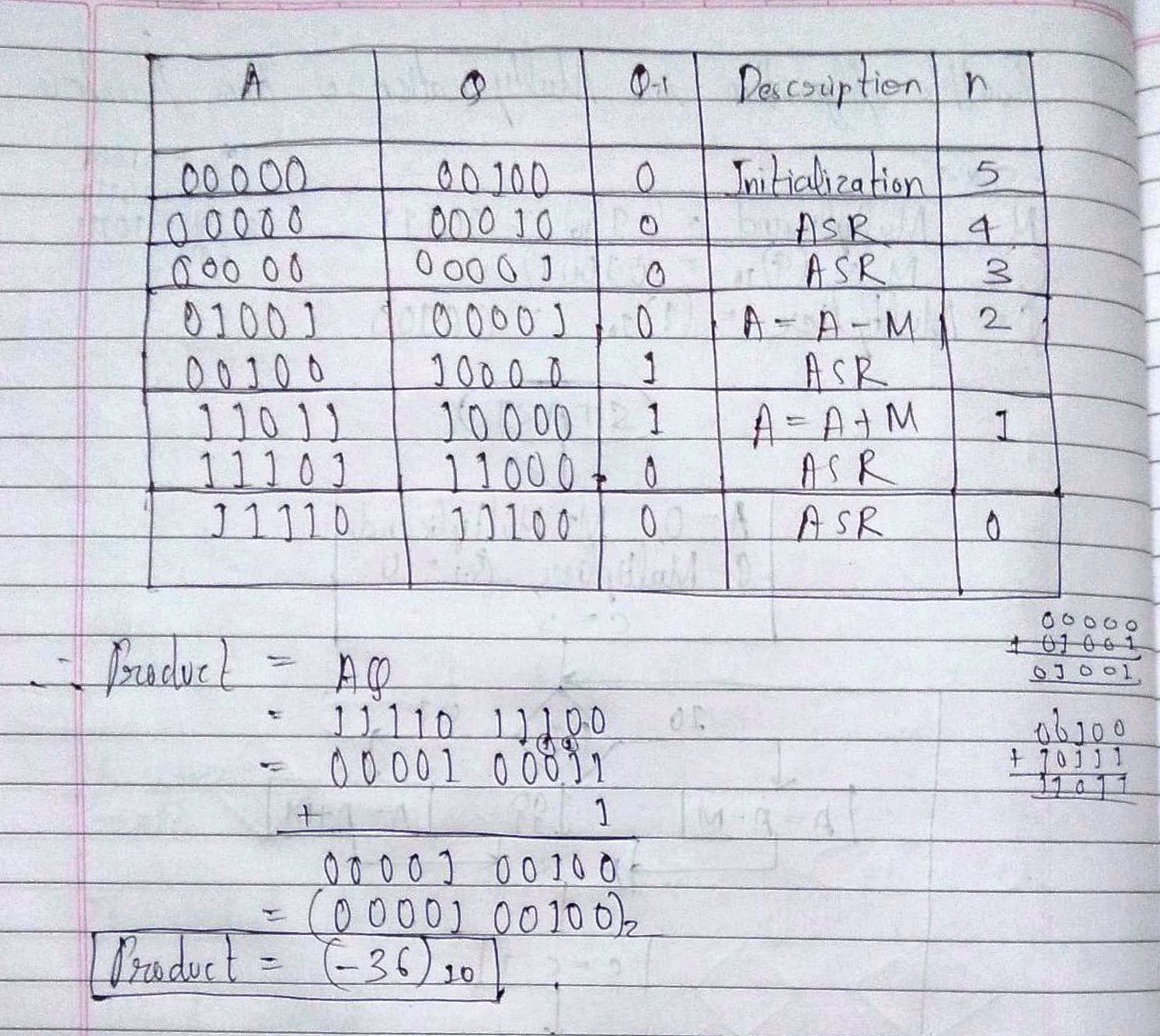
6. Repeat steps 5 till all bits are compared

7. Convert the result to decimal form and display

8. End

Example: (Handwritten solved problem needs to be uploaded)





Code:

#include <stdio.h>

int q = 0,m = 0, c = 0, q1 = 0, m1 = 0, com[10] = { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0};

int Q[10] = {0}, anumcp[10] = {0}, M[10] = {0};

int OCQ[10] = {0}, TCM[10] = {0}, pro[10] = {0}, res[10] = {0};

//q is the Multiplier, m is the multiplicand, c is the carry;

void binary(){

q1 = q;

m1 = m;

int r, r2, i, temp;

for (i = 0; i < 10; i++){

Q[i] = q1 % 2;

anumcp[i] = q1 % 2;

q1 = q1 / 2;

M[i] = m1 % 2;

m1 = m1 / 2;

if(M[i] == 0){

TCM[i] = 1;

}

if(Q[i] == 0){

OCQ[i] =1;

}

}

//part for two's complementing

c = 0;

for ( i = 0; i < 10; i++){

res[i] = com[i]+ TCM[i] + c;

if(res[i] >= 2){

c = 1;

}

else

c = 0;

res[i] = res[i] % 2;

}

for (i = 9; i >= 0; i--){

bcomp[i] = res[i];

}

//in case of negative inputs

if (q < 0){

c = 0;

for (i = 9; i >= 0; i--){

res[i] = 0;

}

for ( i = 0; i < 10; i++){

res[i] = com[i] + OCQ[i] + c;

if (res[i] >= 2){

c = 1;

}

else

c = 0;

res[i] = res[i]%2;

}

for (i = 9; i >= 0; i--){

Q[i] = res[i];

anumcp[i] = res[i];

}

}

if(m < 0){

for (i = 0; i < 10; i++){

temp = M[i];

M[i] = TCM[i];

TCMi] = temp;

}

}

}

void add(int num[]){

int i;

c = 0;

for ( i = 0; i < 10; i++){

res[i] = pro[i] + num[i] + c;

if (res[i] >= 2){

c = 1;

}

else{

c = 0;

}

res[i] = res[i]%2;

}

for (i = 9; i >= 0; i--){

pro[i] = res[i];

printf("%d",pro[i]);

}

printf(":");

for (i = 9; i >= 0; i--){

printf("%d", anumcp[i]);

}

}

void arshift(){//for arithmetic shift right

int temp = pro[9], temp2 = pro[0], i;

for (i = 1; i < 10 ; i++){//shift the MSB of product

pro[i-1] = pro[i];

}

pro[9] = temp;

for (i = 1; i < 10 ; i++){//shift the LSB of product

anumcp[i-1] = anumcp[i];

}

anumcp[9] = temp2;

printf("\nAR-SHIFT: ");//display together

for (i = 9; i >= 0; i--){

printf("%d",pro[i]);

}

printf(":");

for(i = 9; i >= 0; i--){

printf("%d", anumcp[i]);

}

}

void main(){

int i, q\_ = 0;

printf("\nEnter two numbers, each less than 511, to multiply using Booth's Algorithm: ");

//simulating for two numbers each below 16

do{

printf("\nEnter Multiplier Q: ");

scanf("%d",&q);

printf("Enter Multiplicand M: ");

scanf("%d", &m);

}while(q >=511 || m >=511);

printf("\nExpected product = %d", q \* m);

binary();

printf("\n\nBinary Equivalents are: ");

printf("\nQ = ");

for (i = 9; i >= 0; i--){

printf("%d", Q[i]);

}

printf("\nM = ");

for (i = 9; i >= 0; i--){

printf("%d", M[i]);

}

printf("\n2's complement of M = ");//2's Complement of B => B is the Multiplicand as A = A - M => A = A + 2C(M)

for (i = 9; i >= 0; i--){

printf("%d", TCM[i]);

}

printf("\n\n");

for (i = 0;i < 10; i++){

if (Q[i] == q\_){//just shift for 00 or 11

printf("\n-->");

arshift();

q\_ = Q[i];

}

else if(Q[i] == 1 && q\_ == 0){//subtract and shift for 10

printf("\n-->");

printf("\nSUB B: ");

add(TCM);//add two's complement to implement subtraction

arshift();

q\_ = Q[i];

}

else{//add ans shift for 01

printf("\n-->");

printf("\nADD B: ");

add(M);

arshift();

q\_ = Q[i];

}

}

printf("\nProduct is = ");

for (i = 9; i >= 0; i--){

printf("%d", pro[i]);

}

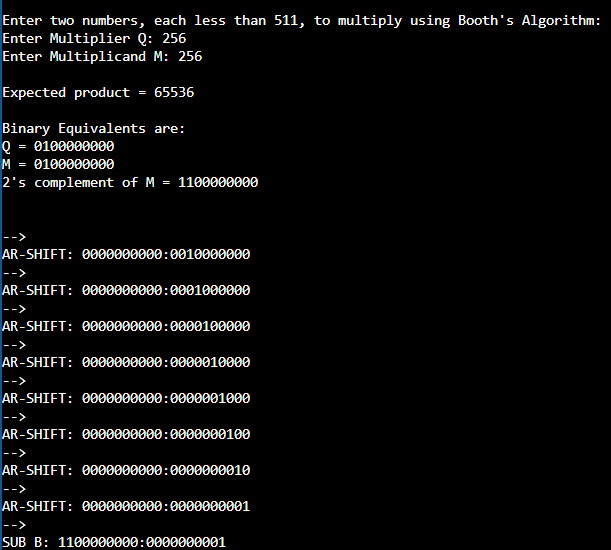
for (i = 9; i >= 0; i--){

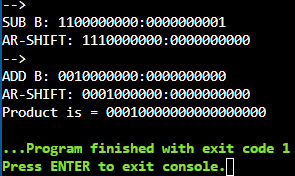
printf("%d", anumcp[i]);

}

}

Output:





**Conclusion:**

Thus, in this experiment, Booth’s Algorithm for multiplication of two signed binary numbers has been implemented. It is using this algorithm that processors perform multiplication of two signed binary numbers. So when a code is written for the multiplication of two variables using the asterisk symbol, then this is the processing which takes place in the microprocessor. It is an efficient algorithm as it performs addition or subtraction only when the Q and Q-1 bits are 01 or 10 respectively. For the rest of the cases including the aforementioned, Arithmetic Shift Right is performed at the end of each cycle.

**Post Lab Descriptive Questions**

1. **Explain advantages and disadvantages of Booth’s algorithm.**

**Ans.** 1. Advantages:

* 1. Easy calculation of multiplication problem.

Both signed and unsigned numbers can be multiplied with good accuracy.

* 1. Consecutive additions will be replaced.

Only if Q0Q-1 are 01 or 10 will addition or subtraction take place. Otherwise, simple ASR takes place (in all steps at the end).

* 1. Less complex and ease scaling.

1. Disadvantages:
   1. This algorithm will not work for isolated 1’s.
   2. It is time consuming.

There are many computational steps.

* 1. If digital gates are more, then the chip area would be large.

1. **Is Booth’s recoding better than Booth’s algorithm? Justify**

**Ans.** Booth’s recoding is derived from Booth’s algorithm. It pairs the multiplier bits and gives one multiplier bit per pair, thus reducing the number of summands by half. This means that the multiplication can be done twice as fast as Booth’s algorithm. Thus, Booth’s recoding is better than Booth’s algorithm.

**Date: \_\_29-09-22\_\_\_ Signature of faculty in-charge**