Generalization of Booth's Algorithm for efficient Multiplication

ABSTRACT: In this paper we summarize the existing work on classical Booth's algorithm of multiplication and propose an improved version of its general form. In Booth's or modified Booth's algorithm two bits or three bits are considered at a time in the multiplier. Here we have shown an approach of using any number of bits of the multiplier. Then we find an optimal value of such bits for which the complexity of the operation will be minimal.

Keywords: Multiplication, Multiplier, Multiplicand, Partial Product, LSB, MSB.

I. INTRODUCTION:

Multiplication is the process of repeated addition one of the earliest development of mathematics. It is commutative and distributed over addition and subtraction. There are several computational approaches for multiplication.

The two numbers in multiplication are called multiplicand and multiplier. Thus the result of multiplication is the number (product) that would be obtained by adding several (multiplier) groups of similar size (multiplicand).

Booth's algorithm multiplies two signed binary numbers in two's complement notation. The algorithm was proposed by A.D Booth in 1951[1]. Booth worked desk calculators that were faster at shifting than adding operation and he employed shift operation to create a fast algorithm for multiplication. This algorithm is of interest in the study of computer architecture.

The Booth's algorithm is based on four stapes. It is performed on binary numbers. Let the multiplier, multiplicand and partial product be M, R and P. In the process an extra 0 is added to the right of the LSB of M and two multiplier bits $M_{n+1}\,M_n$ from LSB is checked and depending on their values R is added of subtracted from P. At end of each step one bit of the multiplier is shifted to the right until it is zero. The basic steps are as follows:

i. if M_{n+1}=0 M_n=0, do nothing.
ii. if M_{n+1}=0 M_n=1, add R to P.
iii. if M_{n+1}=1 M_n=0, subtract R from P.
iv. if M_{n+1}=1 M_n=1do nothing

The modified Booth's algorithm was developed for three bits and is based on eight steps. The operations are performed on binary numbers. The three multiplier bits $M_{n+2}M_{n+1}M_n$ are checked and based on their values the operations are performed. At the beginning of the process one extra 0 is added to the right of LSB of M and the

multiplier is shifted twice to the right at the end of each step until it becomes 0.

1. If $M_{n+2} = 0, M_{n+1} = 0, M_n = 0$, do nothing.
2. If $M_{n+2} = 0, M_{n+1} = 0, M_n = 1$, add R to P.
3. If $M_{n+2} = 0, M_{n+1} = 1, M_n = 0$, add R to P.
4. If $M_{n+2} = 0, M_{n+1} = 1, M_n = 1$, add 2*R to P.
5. If $M_{n+2} = 1, M_{n+1} = 0, M_n = 0$, subtract 2*R from P.
6. If $M_{n+2} = 1, M_{n+1} = 0, M_n = 1$, subtract R from P.
7. If $M_{n+2} = 1, M_{n+1} = 1, M_n = 0$, subtract R from P.
8. If $M_{n+2} = 1, M_{n+1} = 1, M_n = 1$, do nothing

In some articles on modified Booth's algorithm [1] where multiplier developed in higher radix of power of two. But with increasing radix the rate of producing incorrect results increases. Correction on the product needed for accurate results. These changes needs basically for the real numbers to be multiplied.

II. BASIC IDEA:

We have made a generalization of Booth's algorithm [4] as described in section II and III. We have noted that the number of shift of the partial product is one less than the number of bits compared in each step; i.e. if the number of bit of the multiplier is m then the number of shift required to perform on the partial product is m-1. The total numbers of possible bits combinations are2^m. In the Booth's algorithm the total bits combination of the multiplier can be divided into two parts, each part with the number $2^m/2=2^{m-1}$. The all possible operations for one part are the same for the other but opposite; i.e. one part is 1's complement of other. If the total number of bits to be consider is m then the numbers start from m 0s and end with m 1s. So if we divide the total possible combination into two parts then the highest number in the first part is all 1s in the bit combination except the MSB. As an example we can show the following numbers for m bits

0000…00 // m number of 0s, start of the first part
0000…01
0000…10
.
.
.
.
0111…11 // end of the first part
1000…00 // start of the second part
1000…01
1000…10
1000…11
.
.
.
.
1111…11 // m number of 1s, end of the second part

If we have to compute a generalized Booth algorithm for the m bit multiplier then the operations to be performed; i.e. the number of multiplicand to be added to the partial product is as follows:

+0*R // start of the first part
+1*R
+1*R
+2*R
+2*R

.
.
.
+(m-1)*R
+(m-1)*R
+m*R // end of the first part

-m*R // start of the second part
-(m-1)*R
-(m-1)*R
.
.
.
-2*R
-2*R
-1*R
-1*R
-0*R // end of the second part

In the above example the operation of one part is same as other but opposite. The maximum valued multiplicand to be added is m; i.e. the number of multiplier bit to be considered. So, in general we can say that:

For m bits the possible combination will be 0 to 2^m-1, the first part will be from 0 to 2^{m-1}-1 and the second part will be from 2^{m-1} to 2^m-1 and for these numbers the operations to be performed are accordingly for i=1 to 2m, +⌊ j*i ⌋ for first part and i=2m to 1 -⌊ j*i ⌋ for the second part, where j=1/2.

III. ALGORITHMIC FORM:

This process is a generalized form of the classical Booth's multiplication algorithm. In Booth's and the Modified Booth's algorithm the number of bit of the multiplier is two and three respectively. In this process the number of bits is n; n being any number.

Let the multiplicand be R and the multiplier M and the number of bits to be considered be n and the total number of bits of the multiplier be N and the partial product of the multiplication be A. The proposed algorithmic form is as follow:

i. Check the multiplier bits. If the MSB is 1 then take the 1's complement of the multiplier bits, say $\overline{M_n}$. If $\overline{M_n}$ is odd then add 1; i.e. $\overline{M_n}$+1 now subtract ($\overline{M_n}$+1)/2*R from the partial product and if $\overline{M_n}$ is even then subtract $\overline{M}$/2*R from the partial product; i.e. compute

A - ($\overline{M_n}$+1)/2*R if $\overline{Q_n}$ is odd.

A - $\overline{M_n}$/2*R if $\overline{M_n}$ is even.

ii. Check the multiplier bits. If the MSB is 0 and the multiplier bit is even (M_n) then add M_n/2*R to the partial product. If the multiplier bit is odd (M_n) then add (M_n+1)/2*R to the partial product; i.e. compute

A + $\overline{M_n}$/2*R if $\overline{M_n}$ is even.

A + ($\overline{M_n}$+1)/2*R if $\overline{M_n}$ is odd.

IV. CALCULATION OF NUMBER OF PARTIAL PRODUCT:

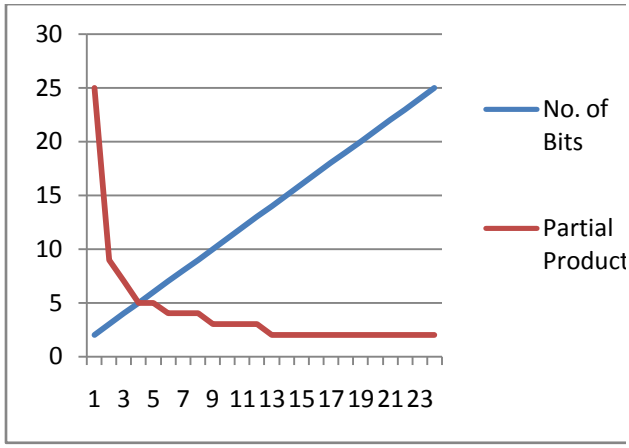
Here we need to mention that if the total number of bits in the multiplier is n and the number of bits to be considered is m then total number of pass P will be:

i. P=n/(-m1) if n is evenly divided by (m-1) otherwise

ii. P=n/(m-1) + 1.

V. GRAPHICAL REPRESENTATION OF ALGORITHM

No. of Bits	Partial Product	No. of Bits	Partial Product
2	25	14	2
3	9	15	2
4	7	16	2
5	5	17	2
6	5	18	2
7	4	19	2
8	4	20	2
9	4	21	2
10	3	22	2
11	3	23	2
12	3	24	2
13	3	25	2



VI. A WORKED OUT EXAMPLE

Let us illustrate our proposed process through an example. In our example we will take a large multiplier and a small multiplicand to show the gain in the complexity of the algorithm. Say multiplier is M and multiplicand is R and initial partial product is P=0.

M=11485=10110011011101

R=5=101

Here n=14 and m=4 then according to our process

```
Pass 1  +1111111111110001          //-3*5
Pass 2  +0000000010100***          //+4*5
Pass 3  +0000001111******          //+3*5
Pass 4  +1110110*********          //-2*5
Pass 5  +1111************          //+3*5
```

=1110000001010001 //11485*5=57425

So, in the above example we can get the correct result. According to the rules P=4.

VII. CONCLUSSION

The process for multiplication of two numbers so far we have described is a general from of the classical Booth's algorithm. It is not concern about the number of bits of the multiplier. The process proved to be efficient over many existing process of multiplication [2][3][4]. We have noted that for a given number of bits of the multiplier the operations are minimum i.e. it is optimized.

VIII. REFERENCES

1. A. Booth, "A signed binary multiplication technique," Q. J. Me& Appl. March., vol. 4, pp. 236-240, 19.51

2. John P. Hayes “Computer architecture and organization”, Second Edition, McGraw-Hill, 1988.
3. M. Morris Mano “Computer System Architecture”, Third Addition, Prentice Hall, 1993.
4. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullmant “The design and analysis of computer algorithm” Pearson, 1974.