MULTIPLICATION ALGORITHM

A report

Submitted in partial fulfillment for the

Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

Submitted by

Aditya Chaurasia (202151006)

Under the guidance of

Dr Bhanu Murthy

Indian Institute of information Technology Vadodara
(2021-2022)

➢ Multiplication Algorithm

A multiplication algorithm is an algorithm (or method) to multiply two numbers. Depending on the size of the numbers, different algorithms are used. Efficient multiplication algorithms have existed since the advent of the decimal system.
Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier.

- Throughout this report, we are discussing the binary multiplication with different bit size. Based on all these criteria the multipliers generally classified as signed and unsigned multipliers. Base on the architecture the multiplier can be divided into three, Truncated Multiplier, Booth Multiplier, Karatsuba multipliers and Vedic multiplier are very somein that categories.

➢ Booth Multiplication Algorithm

The Booth multiplication algorithm defines a multiplication algorithm that can multiply two signed binary numbers in two's complement.

Booth's algorithm contains the addition of one of two predetermined values (A and S) to a product (P) continually and then implementing a rightward arithmetic shift on the product (P). Let us consider the predetermined values to be A and S, and the product to be P. Consider that the multiplicand and multiplier are m and r respectively. Let the number of bits in m and r be x and y respectively.

- **The Booth's multiplication algorithm involves the following steps −**

Step 1 − The values of A and S and the initial value of P are determined. These values should have a length that is equal to (x + y + 1).

- For A, the MSB is filled with the value of m, and the remaining (y+1) bits are filled with zeros.
- For S, the MSB is filled with the value of (-m) in two's complement notations, and the remaining (y + 1) bits are filled with zeros.
- For P, the MSB for x is filled with zeros. To the right of this value, the value of r is appended. Then, the LSB is filled with a zero.

Step 2 − The LSBs of P are determined.

- In case they are 01, find the value of P + A, and ignore the overflow or carry if any.
- In case they are 10, find the value of P + S, and ignore the overflow or carry if any.
- In case they are 00, use P directly in the next step.
- In case they are 11, use P directly in the next step.

Step 3 − The value obtained in the second step is arithmetically shifted by one place to the right. P is now assigned the new value.

Step 4 − Step 2 and Step 3 are repeated for y number of times. Step 5: The LSB is dropped from P, which gives the product of m and r

Example − Find the product of 3 x (-4), where m = 3, r = -4, x = 4 and y = -4.

A = 001100001

S = 110100000

P = 000011000

The loop has to be performed four times since y = 4.

P = 000011000

Here, the last two bits are 00.

Therefore, P = 000001100 after performing the arithmetic right shift.

P = 000001100

Here, the last two bits are 00.

Therefore, P = 000000110 after performing the arithmetic right shift.

P = 000000110

Here, the last two bits are 10.

Therefore, P = P + S, which is 110100110.

P = 111010011 after performing the arithmetic right shift.

P = 111010011

Here, the last two bits are 11.

Therefore, P = 111101001 after performing the arithmetic right shift.

The product is 11110100 after dropping the LSB from P.

11110100 is the binary representation of -12.

➢ Karatsuba Multiplier Algorithm

The **Karatsuba algorithm** is a fast multiplication algorithm that uses a **divide and conquer approach** to multiply two numbers. It was discovered by Anatoly Karatsuba in 1960 and published in 1962.

Basically Karatsuba stated that if we have to multiply two n-digit numbers x and y, this can be done with the following operations, assuming that B is the base of m and m < n (for instance: m = n/2)

- First both numbers **x** and **y** can be represented as **x1,x2** and **y1,y2** with the following formula.

$$x = x_1 * B^m + x_2 x$$
$$y = y_1 * B^m + y_2 y$$

The product x * y becomes the following product:

$$xy = (x_1 * B^m + x_2)(y_1 * B^m + y_2)$$

$$xy = x_1 * y_1 * B^{2m} + x_1 * y_2 * B^m + x_2 * y_1 * B^m + x_2 * y_2$$

```
Consider the following multiplication: 47 x 78

x = 47
x = 4 * 10 + 7

x1 = 4
x2 = 7

y = 78
y = 7 * 10 + 8

y1 = 7
y2 = 8

a = x1 * y1 = 4 * 7 = 28
c = x2 * y2 = 7 * 8 = 56
b = (x1 + x2)(y1 + y2) - a - c = 11 * 15 - 28 - 56

11 * 15 can in turn be multiplied using Karatsuba Algorithm
```

➢ Vedic multiplication Algorithm

The multiplication of two operands using VEDIC multiplier is achieved by multiplication by Vertically and Crosswise and then adding all the results. This multiplication algorithm can be understood using two operands 46 and 33. The operand 33 can be represented as 33 = (3×10 + 3) and 46 can be represented as 46 = (4×10 + 6). The multiplication (46×33) can be represented as (3×6 + 40×3 + 30×6 + 30×40). This multiplication is shown in Figure 1

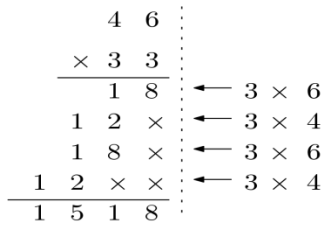

Figure 1: VEDIC Multiplication

- Similar way, this multiplication algorithm can be adopted to implement faster binary multiplier. A 4-bit binary multiplication is shown below in Figure 2

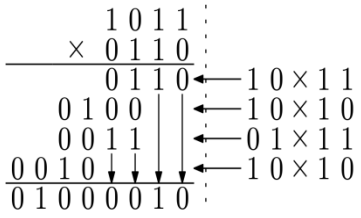

Figure 2: VEDIC multiplication for 4-bit data width.

- **Comparison in terms of Addition and Multiplication**

The number of addition and multiplication decide the speed of operation and the delay. The numbers of these blocks are also capable to increase or decrease the area power etc of the entire design. Comparing other multiplier, Vedic multiplier requires less Multiplication and addition steps.

Multiplier	8x8 bit		16x16 bit		32x32 bit	
	Multiplication	Addition	Multiplication	Addition	Multiplication	Addition
Booth Multiplier	40	26	96	72	288	243
Karatsuba Multiplier	15	10	40	24	198	75
Vedic Multiplier	8	4	16	8	32	16

(Comparison in terms of Addition and Multiplication)

- **Comparisons in Terms of LUT and Delay:-**

	2x2	4x4	8x8
Total no. of LUT	-	24000	28800
LUT used	-	6	20
Delay in (ns)	-	12.85	20.69

(Results of Booth Multiplier)

	2x2	4x4	8x8
Total no of LUT	9112	9312	9582
LUT used	5	14	22
Delay in (ns)	5.658	12.35	17.76

(Results of Karatsuba Multiplier)

	2x2	4x4	8x8
Total no. of LUT	9112	9312	9582
LUT used	4	11	18
Delay in (ns)	5.505	11.35	16.85

(Results of Vedic Multiplier)

LUT - Look Up Table

- **Conclusion**

On doing the hardware implementation of the three multipliers I could conclude that the Vedic multiplier is best. If we consider the number of LUTs used by the Vedic multiplier is 22.2 % less than the Karatsuba multiplier and 11.1% less than the booth multiplier. If we consider the delay of the Vedic multiplier is 30.56% less than the Karatsuba and 22.83% less than the booth multiplier.

- ***Reference*** *:-* *https://digitalsystemdesign.in/vedic-multiplier/*
  https://en.wikipedia.org/wiki/Karatsuba_algorithm
  https://www.tutorialspoint.com/what-is-booth-multiplication-algorithm-in-computer-architecture