# Performance Comparison of Different Multipliers using Booth Algorithm

Snehal R Deshmukh
Dept of E&TC
SSGMCOE
Shegaon, India (MS)

Dinkar L Bhombe
Dept of E&TC
SSGMCOE
Shegaon, India (MS)

***Abstract*— Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. In this paper we try to determine the best solution to this problem by comparing a few multipliers. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does the computations using lesser adders and lesser iterative steps. As a result of which they occupy lessr space as compared to the serial multiplier. This a very important criteria because in the fabrication of chips and high performance system requires components which are as small as possible.**

***Keyswords: multiplier, radix-R, booth algorithm.***

## I. INTRODUCTION

Multipliers play an important role in today's digital signal processing and various other applications. With advancements in technology, many researchers have already tried and are still trying to design multipliers which provides either greater speed, less power consumption, regularity of layout and hence small area or even combination of them in one multiplier which makes them suitable for various increased speed, minimized power and compact VLSI implementation. The usual multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products that needs to be added is the main parameter that defines the performance of the multiplier. In order to minimize the number of partial products to be added, Booth algorithm and Modified Booth algorithm is one of the most popular algorithms [1].

## II. MULTIPLIERS

A Binary multiplier is an electronic hardware circuit that used in digital electronics or a computer or other electronic devices to perform rapid multiplication of two numbers in binary representation. It is obtained using binary adders.
The rules for binary multiplication are as follows

1. If the multiplier digit is a 1, then the product will be same as multiplicand and simply it will be copied down.

2. If the multiplier digit is a 0 the product is also 0.
The multiplication algorithm for an N bit multiplicand by N bit multiplier is shown in fig1.

$Y = Y_{n-1}\ Y_{n-2} \ldots\ldots\ldots\ldots Y_2\ Y_1\ Y_0$ Multiplicand
$X = X_{n-1}\ X_{n-2} \ldots\ldots\ldots\ldots X_2\ X_1\ X_0$ Multiplier

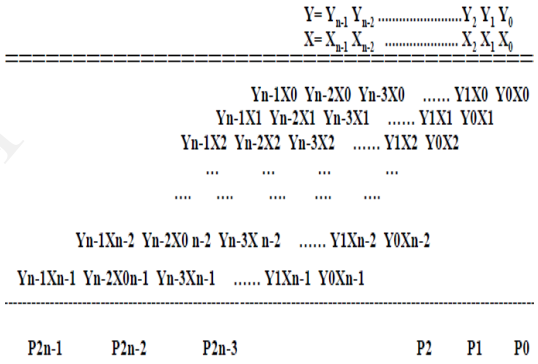



Figure 1. Multiplication algorithm for N*N bit

## III. TYPES OF MULTIPLIER

Basically there are three types of multipliers. They are as follows.

### *a. Serial Multiplier*

Serial multiplier generates partial products sequentially and adds each newly generated product to previously accumulated partial product.

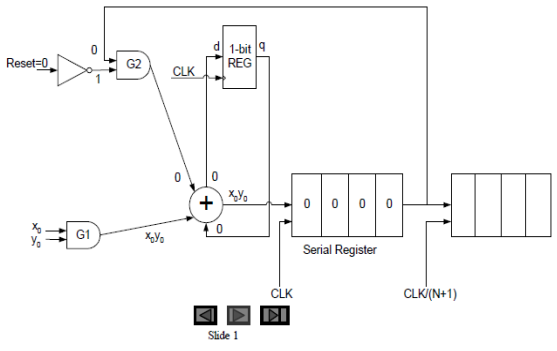



Figure 2.Serial Multiplier

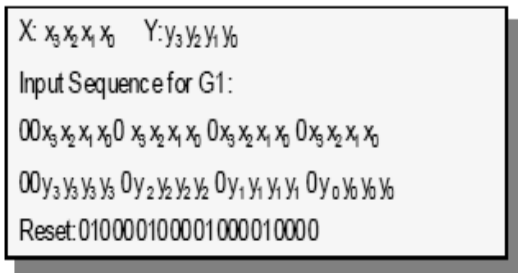



Figure 3

Serial multiplier is used where area and power is important, &delay can be tolerated. Circuit uses one adder to add the m *n partial products. The circuit is shown for m=n=4. Multiplicand and Multiplier inputs have to be arranged in a special manner synchronized with circuit behavior as shown in the fig 2. The inputs could be presented at different rates depending on the length of the multiplicand and the multiplier. Two clocks are used, one to clock the data & one for the reset. A first order approximation of the delay is 0 (m,n). With this circuit arrangement the delay is given as D=[(m+1)n+1] tfa. As shown in fig 3 the individual PP is formed. The addition of the PPs are performed as the intermediate values of PPs, addition are stored in the D Flip Flop, circulated and then added together with the newly formed PP [2].

*Disadvantage*

This approach is not suitable for larger values of M & N.

### *b. Parallel multiplier*

Generates partial products in parallel, accumulates using a fast multi-operand adder.

*Serial/Parallel Multiplier*

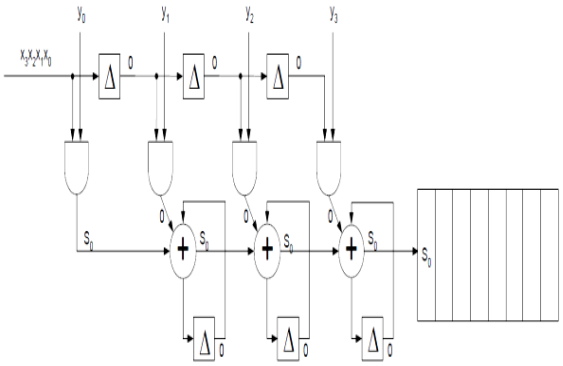


Figure 4.Serial/Parallel Multiplier

One operand is fed to the circuit in parallel while the other is in serial. N partial products are formed for each cycle. On successive cycles, each cycle does the addition of one column of the multiplication table of M*N PPs. The final results are then stored in the output register after completing N+M cycles.

*Disadvantage*

Area required is N-1 for M=N.

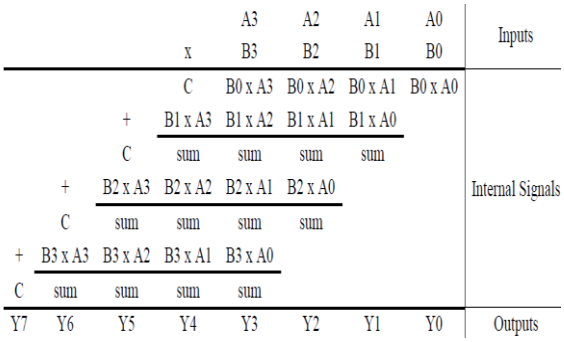



Figure 5. Generation of individual PP and their addition

### *c. Array Multiplier*

Array of identical cells generating new partial products and accumulating them simultaneously is as shown in figue 6. No separate circuits for generation and accumulation is required. This implementation reduces execution time but increases hardware complexity.

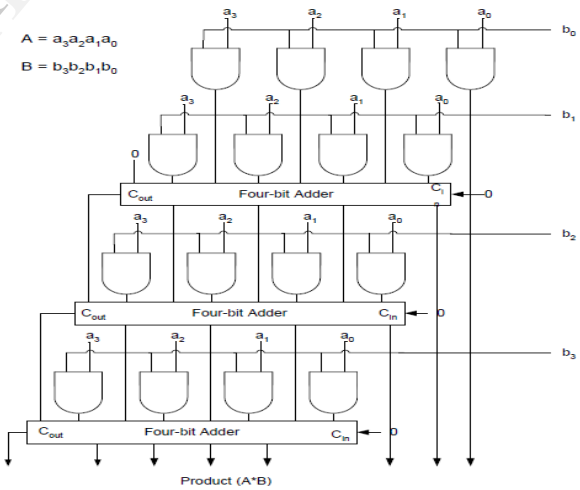



Figure 6. Array Multiplier

Array multiplier is well known due to its regular structure. Multiplier is based on add and shift algorithm. Each and every partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length. Although the method is simple as it can be seen from this example, the addition is done serially as well as in parallel.

*Disadvantage*

Hardware complexity increases with N*M. Now as both multiplicand and multiplier may be positive or negative, 2's complement number system is used to represent them. If the multiplier operand is positive then essentially the same technique can be used but care must be taken for sign bit extension. The reason for dealing with signed number incorrectly is the absence of sign bit expansion in this multiplier.

## IV. MULTIPLICATION ALGORITHM

A circuit that multiplies two unsigned n bit binary numbers, uses a 2 dimensional array of identical subcircuits. Each of which contains a full adder and an ―and‖ gate. For large number of bits this approach may not be appropriate because of the large number of gates needed. Another approach is to use shift register in combination with an adder to implement the traditional method of multiplication. [1]

```
P=0;
For i=0 to n-1 do
If bi=1 then
        P=P+A;
        End if;
        Left shift A;
End for;
```

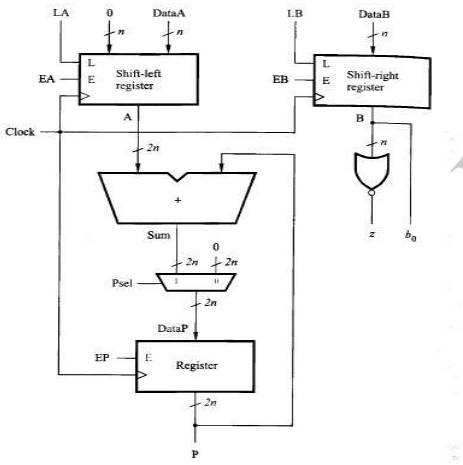



Figure 7. Data circuit of multiplier

## V. BOOTH MULTIPLIERS

This algorithm was invented by Andrew Donald Booth in 1950 while doing study on crystallography. Booth used reception desk calculators that shifts faster than adding and formed the algorithm to increasing the speed. Booth's algorithm is important in the study of computer architecture.[3] . It is a powerful algorithm for signed-number multiplication, which considers both positive and negative numbers uniformly [4]. Booth's Algorithm is a smart move for multiplying signed numbers. It starts with the ability to both add and subtract.[5] An algorithm that uses two's complement notation of signed binary numbers for multiplication.[6]

## VI. MODIFIED BOOTH'S ALGORITHM

Modified Booth's is two times faster than Booth's algorithm. Modified Booth encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive bits in one of the two operands to form the signed multiples. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand. [7]

### *1. Radix-2*

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation. [3]
Illustration of the booth algorithm with example:

Example, $2^{ten} \times (-4)^{ten}$
$0010^{two} * 1100^{two}$
Example, $2^{ten} \times (-4)^{ten}$
$0010^{two} * 1100^{two}$

*Step 1: Making the Booth table [3]*

I. From the above two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.
Therefore, multiplication of 2 x (– 4), where $2^{ten}$ ($0010^{two}$) is the multiplicand and $(-4)^{ten}$ ($1100^{two}$) is the multiplier.
Table 1

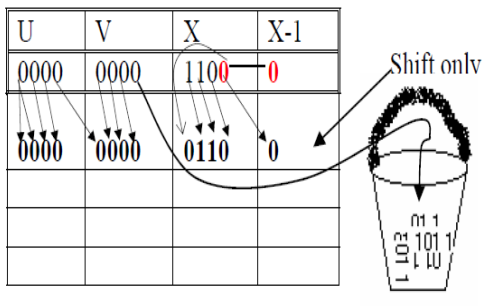
| U | V | X | X-1 | |
|---|---|---|---|---|
| 0000 | 0000 | 1100 | 0 | Load the value |
| | | | | 1st cycle |
| | | | | 2nd cycle |
| | | | | 3rd Cycle |
| | | | | 4th Cycle |

Let X = 1100 (multiplier)
Let Y = 0010 (multiplicand)
2's complement of Y; –Y = 1110
Load the X value in the table.
Load 0 for X-1 value it should be the previous first least significant bit of X .
Load 0 in U and V rows which will have the product of X and Y at the end of operation.
Make four rows for each cycle; this is because we are multiplying four bits numbers.

*Step 2: Booth Algorithm*

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtracted from the partial product, or left unchanged according to the following rules:

Table 2.




Look at the first least significant bits of the multiplier "X", and the previous least significant bits of the multiplier "X - 1".

0 0 Shift only
1 1 Shift only.
0 1 Add Y to U, and shift
1 0 Subtract Y from U, and shift or add (-Y) to U and shift.

Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. Thus a positive number remains positive, and a negative number remains negative. Shift X circular right shift because this will prevent us from using two registers for the X value. Repeat the same steps until the four cycles are completed. [8]

Table 3.

| U | V | X | X-1 |
|---|---|---|---|
| 0000 | 0000 | 1100 | 0 |
| 0000 | 0000 | 0110 | 0 |
| **0000** | **0000** | **0011** | **0** |
| | | | |
| | | | |

← Shift only

*2. Radix-4*

Radix-4 Booth algorithm scans strings of 3 bits with the algorithm given below Append a 0 to the right side of the LSB of the multiplier consider the bits in groups of three, in a way that each group overlaps with the previous group by one bit. Grouping starts from the LSB and the first group only uses 2 bits of the multiplier. According to the value of each vector, Partial Product will be 0, +Y, –Y, +2Y,–2Y. The negative values of y are considered by taking the 2's complement to the Booth recode the multiplier term, we have to consider the bits in groups of three, in a way that each group overlaps with the previous group by one bit. Grouping starts from the LSB and the first group only uses 2 bits of the multiplier. [7] Multiplier is equal to 0 1 0 1 1 10 then a 0 is placed to the right most bit which gives 0 1 0 1 1 10 0 the 3 digits are selected at a time with overlapping left most bit as follows:

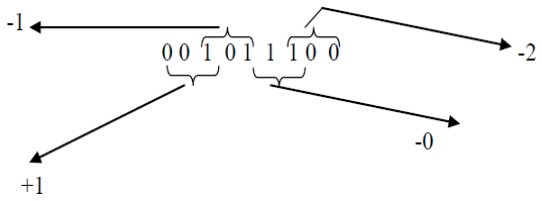
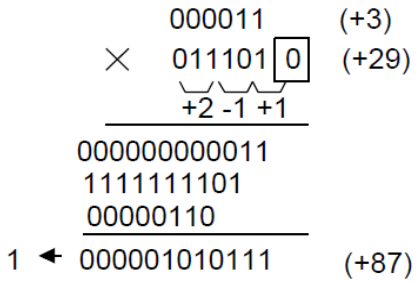



Figure 8 Grouping of three bits

Table 4. Encoding of Radix-4 Booth Multiplier[9] [10] [11]

| Groups | Partial Product |
|---|---|
| 000 | 0 |
| 001 | 1*multiplicand |
| 010 | 1*multiplicand |
| 011 | 2*multiplicand |
| 100 | -2*multiplicand |
| 101 | -1*multiplicand |
| 110 | -1*multiplicand |
| 111 | 0 |




## VII. CONCLUSION

We found that the parallel multipliers are much faster than the serial multiplier. In case of parallel multipliers, the total area is much less than that of serial multipliers. Hence the power consumption is also less. This speeds up the calculation and makes the system faster. While comparing the radix 2 and the radix 4 booth multipliers we found that radix 4 Consumes lesser power than that of radix 2. This is because it uses almost half number of iteration and adders when compared to radix 2. When all the multipliers were compared we found that array multipliers are most power consuming and have the maximum area. This is because it uses a large number of adders. As a result it slows down the system because now the system has to do a lot of calculation. Multipliers are one the most important component of many systems. So we always need to find a better solution in case of multipliers. Our multipliers should

always consume less power and cover less area. In the end we determine that radix 4 modified booth algorithm works the best.

## REFERENCES

[1] Prasanna Raj P, Rao, Ravi, "VLSI Design and Analysis of Multipliers For Low Power", Intelligent Information Hiding and Multimedia Signal
Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.

[2] R.I.Hartle, K.K.Pardhi, Digit-Serial Computation,Kluwer Academic,Boston,MA 1995

[3] Jayashree Taralabenchi, Kavana Hegde, Soumya Hegde, ―Implementation of Binary Multiplication using Booth and Systolic Algorithm on FPGA using VHDL,‖ International Conference & Workshop on Recent Trends in Technology, (TCET) 2012 Proceedings published in International Journal of Computer Applications® (IJCA) .

[4] Louis P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication", Computers, IEEE Transactions,vol.24, pp.: 1014-1015, Oct. 1975

[5] Depth:In More Booth's Algorithm, staff.ustc.edu.cn/~han/CS152CD/Content/COD3e/inmoredepth/IMD3-Booths-Algorithm.pdf - -

[6] Abenet Getahun, ―Booth Multiplication Algorithm,‖ Fall 2003 CSCI 401

[7] Sandeep Shrivastava*, Jaikaran Singh* and Mukesh Tiwari*, ―Implementation of Radix-2 Booth Multiplier and Comparison with Radix-4 Encoder Booth Multiplier,‖ International Journal on Emerging Technologies 2(1): 14-16(2011) ISSN : 0975-8364

[8] Abenet Getahun, ―Booth Multiplication Algorithm,‖ Fall 2003 CSCI 401

[9] Fabrizio Lamberti, Nikolaos Andrikos,Elisardo Antelo, Paolo Montuschi, ―Speeding-up Booth Encoded Multipliers by Reducing the Size of Partial Product Array," Internal Report Dauin/Delen-Politecnico Di Torino and University De Santiago De Compostela, 2009

[10] S. Shafiulla Basha1, Syed. Jahangir Badashah2, ―Design and Implementation of Radix-4 Based High Speed Multiplier For ALU'S Using Minimal Partial Products,International Journal of Advances in Engineering & Technology, July 2012. ©IJAET ISSN: 2231-1963

[11] H. S. Krishnaprasad Puttam1, P. Sivadurga Rao2 & N. V. G. Prasad3, ― Implementation of Low Power and High Speed Multiplier-Accumulator Using SPST Adder and Verilog,International Journal of Modern Engineering Research (IJMER) www.ijmer.com Vol. 2, Issue. 5, Sep.-Oct. 2012 pp-3390-3397 ISSN: 2249-6645.

[12] J.Umamageshwari M.Yeni Saranya M.tech, ―Asic implementation of low power high radix booth encoded multiplier using spst, international journal of communications and engineering volume 05–no.5, issue: 02 march2012 .

[13] Stephen Brown Zvonko Vranesic, Fundamentals of Digital Logic Degin with VHDL, Tata mcgraw-Hill