# Different Multipliers & its performance analysis In VLSI using VHDL

Sonal Prajapati
*Assistant Professor, Department of Electrical Engineering, VIER, Kotambi, Gujarat*

***Abstract*- Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Array multiplier is the basic one. However, they suffer from more propagation delay. Hence, where regularity, high performance and low power are primary concerns, Booth multipliers tend to be the primary choice. Booth multipliers allow the operation on signed operands in 2's-complement which are derived from array multipliers where each bit in a partial product line an encoding scheme is used to determine whether the bit is positive, negative or zero. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In this algorithm each partial product line operates on 2 bits at a time, thereby reducing the total number of the partial products. This is particularly true for operands using 16 bits or more.**

***Index Terms*- Array multiplier, parallel multiplier, propagation delay, VHDL, LUT, DSP block, utilization and twiddle f-actors.**

## 1. INTRODUCTION

Performance is recognized as a one of the critical parameter in digital system design field, especially in Digital signal processing (DSP) applications. Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in DSP as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, Subtraction, and shift operations. Multiplication operation is the series of repeated additions, number to be added is the multiplicand and the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length is determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. In digit, serial multipliers have single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers have been plagued by complicated switching systems or irregularities in design. The parallel Multipliers provide high performance compared to serial multipliers and avoiding irregularities. In this paper, evaluating performance of 3 different parallel multipliers in terms of power, delay and area and find out which one is providing high performance of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is twofold, evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive Addition's of the columns of the shifted bit of the 'multiplicand'. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix.

They are then added to form the product bit for the particular form .Multiplication is there for multi operand operation. To extend the multiplication to both signed and unsigned numbers a convenient. Number system would be the representation of numbers in two's complement format. Multiplication is an important fundamental function in arithmetic function.

## 2. MULTIPLIERS

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.

Characteristics of multiplier:

An efficient multiplier should have following characteristics:

Speed: Multiplier should perform operation at high speed.

Area: A multiplier should occupy less number of slices and LUTs.

Power: Multiplier should consume less power.

Multiplication process has three main steps:

1. Partial product generation.
2. Partial product reduction.
3. Final addition.

For the multiplication of an n-bit multiplicand with an m bit multiplier, m partial products are generated and product formed is n + m bits long.

Different types of multipliers:

Here we discuss different types of multipliers which are

1. Binary multiplier
2. Array multiplier
3. Booth multiplier
4. Modified booth multiplier

2.1 Binary multiplier:

A binary multiplier is an electronic hardware device used in digital electronics or a computer or other electronic device to perform rapid multiplication of two numbers in binary representation. It is built using binary adders. The rules for multiplication can be stated as

1. If the multiplier digit is 0 the product is also 0.
2. If the multiplier digit is 1 then the multiplicand is simply copied down and represents the product.
3. It should be capable of shifting left partial products.
4. It should be able to add all the partial products to give the products as sum of partial products.
5. It should be examine the sign bits. If they are alike the sign of the product will be a positive, if the sign bits are opposite product will b negative. The sign bit of the product stored with above criteria should be displayed along with the product.

If binary multiplier is implemented using full adder the delay is reduced from C (in) to C (out) because the carry chain is the critical delay path in adders. Instead carry save adders are normally implemented to minimize delay from each input to the outputs.

2.2 Array multiplier:

It is well known due to its regular structure. Multipliers circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit adders and then added.

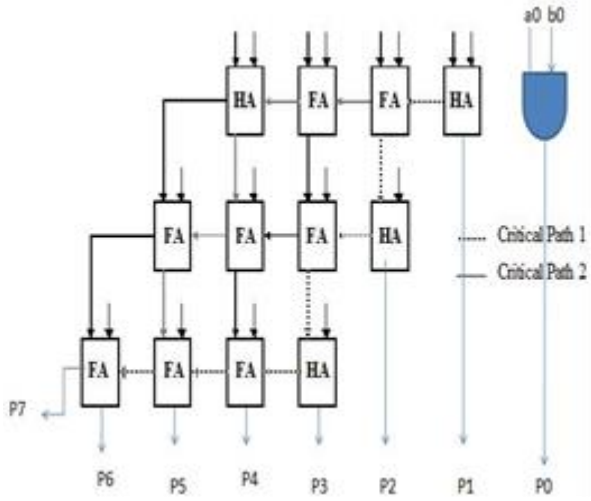



Fig.1 Architecture of array multiplier boao is generated by using AND gate For m*n Array multiplier Number of AND Gates are m*n.

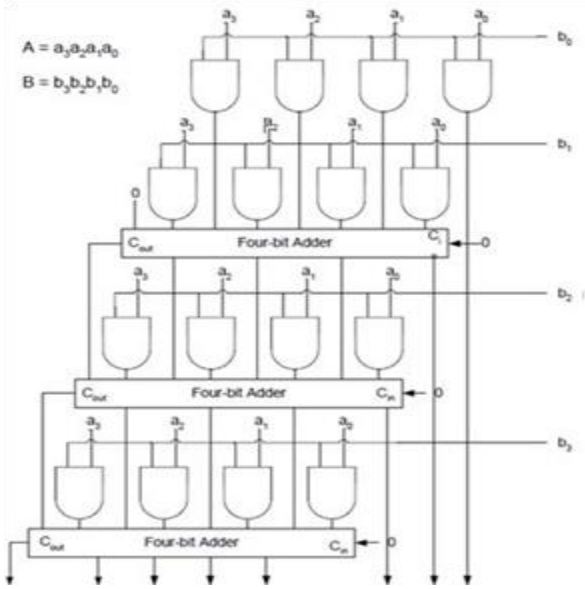
The addition can be performed with normal carry propagate adder.

N-1adders are required where N is the multiplier length Numbers of Half Adders are n.

Numbers of Full Adders are n (m-2) Total numbers of adders are n (m-1)

```
          A     a3 a2 a1 a0(multiplicand)
          B     b3 b2 b1 b0(multiplier)
     ---------------------------------
                b0a3 b0a2 b0a1 b0a0
           b1a3 b1a2 b1a1 b1a0
      b2a3 b2a2 b2a1 b2a0
 b3a3 b3a2 b3a1 b3a0
-------------------------------------------------
   P7   P6   P5    P4   P3  P1  P0
(Carry)
```

An example of 4-bit multiplication method is shown below:




Product (a*b)

Fig.2 Example of 4-bit multiplication

Although the method is simple as it can be seen from this example, the addition is done serially as well as in parallel. To improve on the delay and area the CRAs are replaced with Carry Save Adders, in which every carry and sum signal is passed to the adders of the next stage. Final product is obtained in a final adder by any fast adder (usually carry ripple adder). In array multiplication we need to add, as many partial products as there are multiplier bits. This arrangements is shown in the figure below

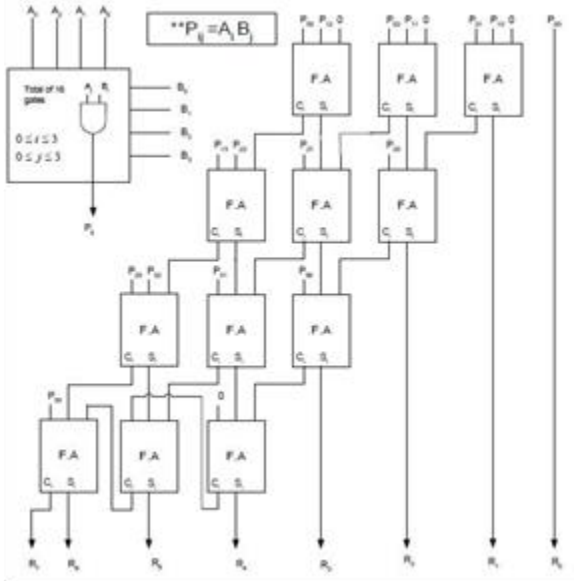



Fig .3 Array multiplier using carry save adder

Problem in Array Multiplier:

Carry should be travelled up to P7 then only output generates

Tc—Carry propagation time

Ts---Sum Propagation time of an adder

Ta---And gate time

If Tc>>Ts

Neglect Ts(since Ts is local and Tc is Global phenomenon)

Total multiplier Time is Ta+6Tc or Ta+[(n-1)+(m-1)]Tc

If Tc<Ts,

Total multiplication time is Ta+3Tc+3Ts orTa+(n-1)Ts+(m-1)Tc

2.3 Booth multiplier:

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. In booth multiplication process arithmetic shift and circular shift operations are performed.

Circular shift:

In combinational mathematics, a circular shift is the operation of rearranging the entries in a tuple, either by moving the final entry to the first position, while shifting all other entries to the next position, or by performing the inverse operation. A circular shift is a special kind of cyclic permutation, which in turn is a special kind of permutation.

The results of repeatedly applying circular shifts to a given tuple are also called the circular shifts of the tuple. For example, repeatedly applying circular shifts to the four- tuple (a, b, c, d) successively gives.

- (d, a, b, c),
- (c, d, a, b),
- (b, c, d, a),
- (a, b, c, d) (the original four-tuple),

and then the sequence repeats; this four-tuple therefore has four distinct circular shifts. However, not all n-tuples have n distinct circular shifts. For instance, the 4-tuple (a, b, a, b) only has 2 distinct circular shifts. In general the number of circular shifts of an n-tuple could be any divisor of n, depending on the entries of the tuple.

In computer programming a circular shift (or bitwise rotation) is a shift operator that shifts all bits of its operand. Unlike an arithmetic shift a circular shift

does not preserve a number's sign bit or distinguish a number's exponent from its mantissa. Unlike a logical shift, the vacant bit positions are not filled in with zeros but are filled in with the bits that are shifted out of the sequence.

If the bit sequence 0001 0111 were subjected to a circular shift of one bit position

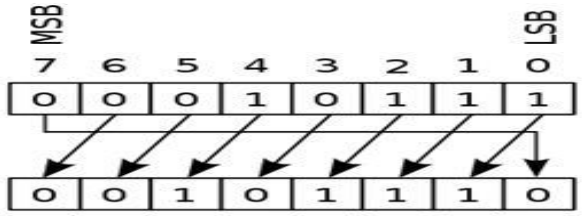



Fig.4 Left circular shift operation

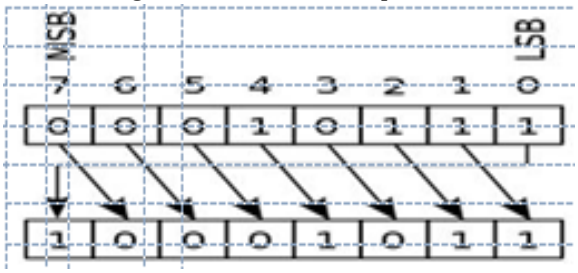



Fig.5 Right circular shift operation

Steps are for implementing the booth algorithm:

Let X and Y are two binary numbers and having m and n numbers of bits (m and n are equal) respectively.

Step 1: Making booth table: In booth table we will take four columns one column for multiplier second for previous first LSB of multiplier and other two (U and V) for partial product accumulator.

1. From two numbers, choose multiplier (X) and multiplicand (Y).
2. Take 2's complement of multiplicand (Y).
3. Load Xi value in the table.
4. Load 0 for Xi-1 value.
5. Load 0 in U and V which will have product of X & Y at the end of the operation.
6. Make n rows for each cycle because we are multiplying m and n bits numbers.

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

Table3.2 Booth recoding table

| | Xi | | Operation to be performed |
|---|---|---|---|
| 0 | 0 | | Shift only |
| 1 | 1 | | Shift only |
| 0 | | 1 | Add Y to U and shift |

Table3.2 Booth recoding table

2. Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. So, positive numbers and negative numbers remains positive and negative respectively.
3. Circularly right shift X because this will prevent us from using two registers for the X value. Repeat the same steps until n no. of cycles are completed. In the end we get the product of X and Y.

Let us consider an example for Booth multiplication

Consider 1101(x)*1011(y)

| | U | V | X | X-1 | |
|---|---|---|---|---|---|
| | 0000 | 0000 | 1101 | 0 | |
| + | 0101 | | | | |
| | 0101 | | | | |
| RSA | 0010 | 1000 | 1110 | 1 | (RSC) |
| + | 1011 | | | | |
| | 1101 | | | | |
| RSA | 1110 | 1100 | 0111 | 0 | (RSC) |
| (-) | 0101 | | | | |
| | 0011 | | | | |
| RSA | 0001 | 1110 | 1011 | 1 | (RSC) |
| | 0000 | 1111 | 1101 | 1 | (STOP) |

The original version of the Booth algorithm (Radix-2) had two drawbacks. They are:

(i) The number of add subtract operations and the number of shift operations becomes variable and become inconvenient in designing parallel multipliers.

(ii) The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix 4.

2.4 Modified booth multiplier:

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In

this case the delay of multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of the additions, the performance will get better.


Fig.6 Radix-4 multiplication in dot notation

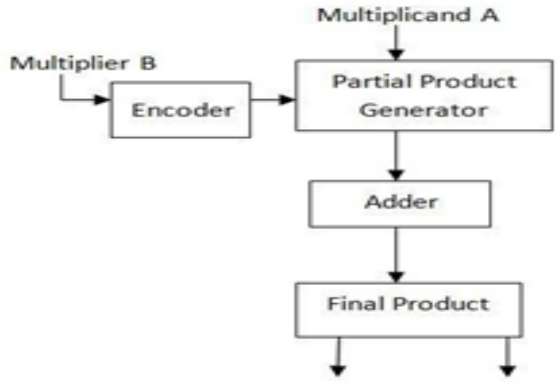



Fig.7 Block Diagram of Modified Booth Algorithm

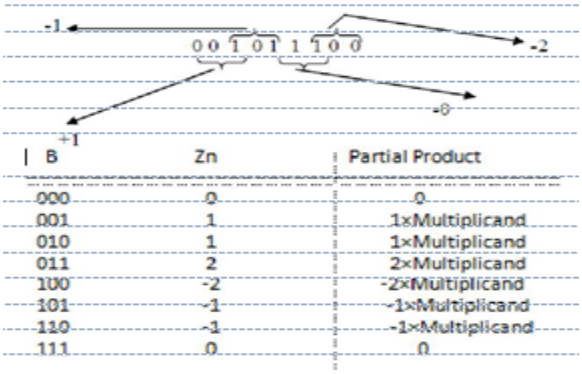
Table 3.3 below is used to convert a binary number to radix-4 number. Initially, a ―0‖ is placed to the right most bit of the multiplier. Then 3 bits of the multiplicand is recoded according to table below or according to the following equation:

$Zi = -2xi+1 + xi + xi-1$

Example:

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

Table3.3 Binary to Radix-4 conversion table




| B | Zn | Partial Product |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 1 | 1×Multiplicand |
| 010 | 1 | 1×Multiplicand |
| 011 | 2 | 2×Multiplicand |
| 100 | -2 | -2×Multiplicand |
| 101 | -1 | -1×Multiplicand |
| 110 | -1 | -1×Multiplicand |
| 111 | 0 | 0 |

For example, an unsigned number can be converted into a signed-digit number radix 4:

(10 01 11 01 10 10 11 10)2 = ( –2 2 –1 2 –1 –1 0 –2)4

Here –2*multiplicand is actually the 2s complement of the multiplicand with an equivalent left shift of one bit position. Also, +2 *multiplicandis the multiplicand shifted left one bit position which is equivalent to multiplying by 2.

To enter 2*multiplicand into the adder, an (n+1)-bit adder is required. In this case, the multiplicand is offset one bit to the left to enter into the adder while for the low-order multiplicand position a 0 is added. Each time the partial product is shifted two bit positions to the right and the sign is extended to the left.

During each add-shift cycle, different versions of the multiplicand are added to the new partial product depends on the equation derived from the bit-pair recoding table above.

## 3. FINAL RESULT

For comparison we have implemented different multipliers in terms of delay (ns), area and power. These Multipliers were modelled in VHDL and synthesized by using Xilinx design suite14.4

PERFORMANCE COMPARISON OF ARRAY, BOOTH AND MODIFIED BOOTH MULTIPLIERS

| | Array | Booth | Modified Booth |
|---|---|---|---|
| DELAY(ns) | 23.856 | 30.798 | 7.818 |
| POWER(mw) | 0.081 | 0.014 | 0.014 |
| BELLS(AREA) | 123 | 458 | 22 |
| I/O BUFFERS | 32 (16+16) | 33 (17+16) | 27 (11+16) |

The above table shows the performance comparison of different multipliers. In array Multiplier more area and high Power consumption. In comparison use of booth multiplier Occupies more area, low power consumption and speed is almost same as array multiplier.The Modified Booth Multiplier is Best

from the above multipliers because of area occupied is Less and more fastest multiplier in comparison to array and booth multipliers .it is consuming low power also.

## 4. CONCLUSION

We analyzed the area occupied, power consumed and the time delay consumed by different multipliers and found out best among them. After comparing all we came to a conclusion that Modified booth multiplier is best suited for high speed and low power applications.

## 5. FUTURE WORK

There are several future research directions are possible for further reduction of power and time delay consumption. One of the possible directions is radix higher-than-4 recoding. We have only considered radix-4 recoding as it is a simple and popular choice. Higher-radix recoding further reduces the number of PPs and thus has the potential of power saving.

## REFERENCES


[1] S. Shafiulla Basha, Syed. Jahangir Badashah. ‘Design and implementation of Radix-4 based high speed multiplier for ALU’s using minimal partial product‘ International Journal of Advances in Engineering & Technology, July 2012 ISSN: 2231-1963.

[2] Dhanya Geethanjali Sasidharan, AarathyIyer ,Comparison of Multipliers Based on Modified Booth Algorithm‘ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, January February 2013, pp.1513-1516

[3] S. Jagadeesh, S. Venkata Chary, Design of Parallel Multiplier– Accumulator Based on Radix-4 Modified Booth Algorithm with SPST‘ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, issue 5, September-October 2012, pp.425-431

[4] A. R. Cooper, ―Parallel architecture modified Booth multiplier,‖Proc. Inst. Electr. Eng. G, vol. 135, pp. 125–128, 1988 M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[5] Marc Hunger and Daniel Marienfeld, ‚New self-checking booth multiplier‘, Int. J. Appl. Math, Comput. Sci., Vol.18, No.3, 319– 328,2008.

[6] S. Waser and M. J. Flynn, Introduction to Arithmetic for Digital Systems Designers. New York: Holt, Rinehart and Winston, 1982.

[7] A. R. Omondi, Computer Arithmetic Systems. Englewood Cliffs, NJ:Prentice-Hall, 1994.

[8] Prof. Vojin G. Oklobdzija ‚High-Speed VLSI Arithmetic Units: Adders and Multipliers‘ September 13, 1999

[9] Kai Hwang ―Computer Arithmetic: Principles, Architecture, and Design‖ John Wiley & Sons 1979

[10] S. D. Pezaris "A 40-ns 17-Bit by 17-Bit Array Multiplier", IEEE Trans. on Computers, pp. 442-447,.Abr. 1971

[11] Priyastalin, anuradha, k ranjithkumar, n vaishnav, d vigneswara, s t santhosh ―high speed multiplier with pipelining‖ InternationalJournal of VLSI and Embedded Systems-IJVES .

[12] NavdeepKaur, Rajeev Kumar Patial ―Implementation of Modified Booth Multiplier using Pipeline Technique on FPGA‖ International Journal of Computer Applications (0975 – 8887).

[13] K. Srishylam, Prof. Syed Amjad Ali, M.Praveena ―Implementation of Hybrid CSA, Modified Booth Algorithm and Transient power Minimization techniques in DSP/Multimedia Applications‖International Journal of Engineering Research and Applications (IJERA)

[14] Shanthala S, S. Y. Kulkarni ―VLSI Design and Implementation of Low Power MAC Unit with Block Enabling Technique‖ European Journal of Scientific Research

[15] Iffat Fatima ―Analysis of Multipliers in VLSI‖ Journal of Global Research in Computer Science.