iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** IV **Month of publication:** April 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Implementation of Modified Booth Algorithm for Parallel MAC

Stephen[1], Ravikumar. M[2]
[1]PG Scholar, ME (VLSI DESIGN), [2]Assistant Professor, Department ECE
Mahendra Engineering College, Namakkal, Tamilnadu, India.

Abstract --This paper presents the methods required to implement a high speed and high performance parallel complex number multiplier. The designs are structured using Radix-4 Modified Booth Algorithm and Wallace tree. These two techniques are employed to speed up the multiplication process as their capability to reduce partial products generation and compress partial product term by a ratio of 3:2. Despite that, carry save-adders (CSA) is used to enhance the speed of addition process for the system. The system has been designed efficiently using VHDL codes for 8x8-bit signed numbers and successfully simulated and synthesized using Xilinx [16].
Keywords: Multiplier and accumulator (MAC), Carry save adder (CSA), Radix-4 Modified Booth algorithm, Digital Signal Processing (DSP).

I. INTRODUCTION

The speed of multiplication operation is of great importance in digital signal processing as well as in the general purpose processors today. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, 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 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 i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate 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 therefore a 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. The MAC (Multiplier and Accumulator Unit) is used for image processing and digital signal processing (DSP) in a DSP processor. Algorithm of MAC is Booth's radix-4 algorithm, Modified Booth Multiplier; Wallace tree improves speed and reduces the power [9].

A. Basics of Multiplier

Multiplication is a mathematical operation that at its simplest is an abbreviated process of adding an integer to itself a specified number of times [2]. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result (product).

In elementary school, students learn to multiply by placing the multiplicand on top of the multiplier. The multiplicand is then multiplied by each digit of the multiplier beginning with the rightmost, Least Significant Digit (LSD). Intermediate results (partial products) are placed one atop the other, offset by one digit to align digits of the same weight. The final product is determined by summation of all the partial-products. Although most people think of multiplication only in base 10, this technique applies equally to any base, including binary. Figure.1 shows the data flow for the basic multiplication technique just described. Each black dot represents a single digit.

Here, we assume that MSB represent the sign of the digit. The operation of multiplication is rather simple in digital electronics. It has its origin from the classical algorithm for the product of two binary numbers. This algorithm uses addition and shift left operations to calculate the product of two numbers. Based upon the above procedure, we can deduce an algorithm for any kind of multiplication which is shown in Figure.2. We can check at the initial stage also that whether the product will be positive or negative or after getting the whole result, MSB of the results tells the sign of the product.

©IJRASET: All Rights are Reserved

Volume 4 Issue IV, April 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

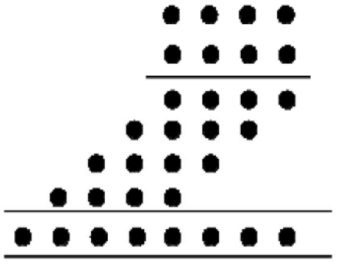


Figure.1 Basic Multiplication

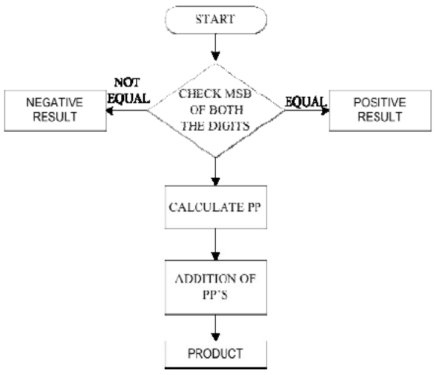


Figure.2 Signed Multiplication Algorithm

B. Binary Multiplication

In the binary number system the digits, called bits, are limited to the set [0, 1]. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial-products simple and efficient. Summing these partial-products is the time consuming task for binary multipliers. One logical approach is to form the partial-products one at a time and sum them as they are generated. Often implemented by software on processors that do not have a hardware multiplier, this technique works fine, but is slow because at least one machine cycle is required to sum each additional partial-product. For applications where this approach does not provide enough performance, multipliers can be implemented directly in hardware. The two main categories of binary multiplication include signed and unsigned numbers. Digit multiplication is a series of bit shifts and series of bit additions, where the two numbers, the multiplicand and the multiplier are combined into the result. Considering the bit representation of the multiplicand x = xn- 1…..x1 x0 and the multiplier y = yn-1…..y1y0 in order to form the product up to n shifted copies of the multiplicand are to be added for unsigned multiplication [2].

C. Multiplication Process

The simplest multiplication operation is to directly calculate the product of two numbers by hand.This procedure can be divided into three steps: partial product generation, partial product reduction and the final addition. To further specify the operation process, let us calculate the product of 2 two's complement numbers, for example, 11012 (−310) and 01012 (510), when computing the product by hand, which can be described according to Figure.3. The first operand is called the multiplicand and the second the multiplier. The intermediate products are called partial products and the final result is called the product. However, the multiplication process, when this method is directly mapped to hardware, is shown in Figure.2. As can been seen in the Figures, the multiplication operation in hardware consists of PP generation, PP reduction and final addition steps. The two rows before the product are called sum and carry bits. The operation of this method is to take one of the multiplier bits at a time from right to left, multiplying the multiplicand by the single bit of the multiplier and shifting the intermediate product one position to the left of the earlier intermediate products. All the bits of the partial products in each column are added to obtain two bits: sum and carry. Finally, the sum and carry bits in each column have to be summed. Similarly, for the multiplication of an *n*-bit multiplicand and an *m*-bit multiplier, a product with *n* + *m* bits long and *m* partial products can be generated. The method shown in Figure.3 is also called a non-Booth encoding scheme [7].

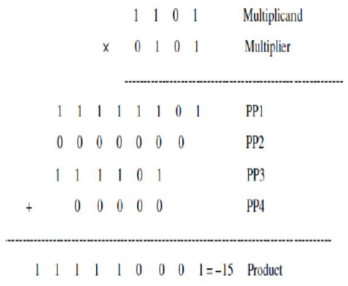


Figure.3 Multiplication calculations by hand

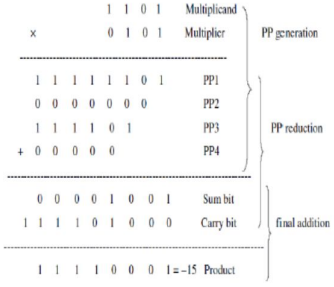


Figure.4 Multiplication Operation in hardware

This paper is organize as follows, section 2 discusses about multiplier & accumulator, section 3 design of MAC and its importance with specifications of operations, section 4 simulation results and discussions, section 5 advantages of this method. Conclusion has been summarized end section 6.

II. A MULTIPLIER AND ACCUMULATOR

A. Overview of MAC

A multiplier can be divided into three operational steps. The first is radix-4 Booth encoding in whicha partial product is generated from the multiplicand X and the multiplier Y. The second is adder array or partial product compression to add all partial products. The last is the final addition in which the process to accumulate the multiplied results is included.The general hardware architecture of this MAC is shown in Figure.2. It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y. This is added to the previous multiplication result Z as the accumulation step.The N-bit 2's complement binary number can be expressed as

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \qquad x_i \in 0, 1. \quad \text{…….. (1)}$$

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm. Each of the two terms on the right-hand side of (5) is calculated independently and the final result is produced by adding the two results. The MAC architecture implemented by (5) is called the standard design [6]. If bit data are multiplied, the number of the generated partial products is proportional to N. In order to add them serially, the execution time is also proportional to N. The architecture of a multiplier, which is the fastest, uses radix-4 Booth encoding that generates partial products. If radix-4 Booth encoding is used, the number of partial products, is reduced to half, resulting in the decrease in Addition of Partial Products step. In addition, the signed multiplication based on 2's complement numbers is also possible. Due to these reasons, most current used multipliers adopt the Booth encoding.

B. Multiplier and Accumulator Unit

MAC is composed of an adder, multiplier and an accumulator. Usually adders implemented are Carry- Select or Carry-Save adders, as speed is of utmost importance in DSP (Chandrakasan, Sheng, & Brodersen, 1992 and Weste & Harris, 3rd Ed). One implementation of the multiplier could be as a parallel array multiplier. The inputs for the MAC are to be fetched from memory location and fed to the multiplier block of the MAC, which will perform multiplication and give the result to adder which will accumulate the result and then will store the result into a memory location. This entire process is to be achieved in a single clock cycle (Weste & Harris, 3rd Ed). The architecture of the MAC unit which had been designed in this work consists of one 16-bit register, one 16-bit Modified Booth Multiplier, 32-bit accumulator. To multiply the values of A and B, Modified Booth multiplier is used instead of conventional multiplier because Modified Booth multiplier can increase the MAC unit design speed and reduce multiplication complexity. SPST Adder is used for the addition of partial products and a register is used for accumulation. The operation of the designed MAC unit is as in equation (6). The product of Ai x Bi is always fed back into the 32-bit accumulator and then added again with the next product Ai x Bi. This MAC unit is capable of multiplying and adding with previous product consecutively up to as many as times.

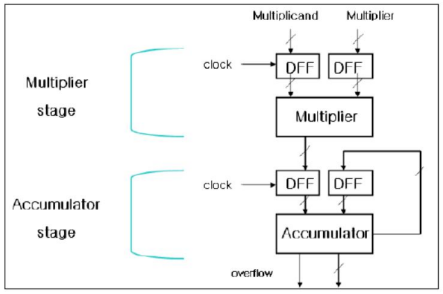


Figure.5 Simple Multiplier and Accumulator Architecture

III. DESIGN OF MAC

In the majority of digital signal processing (DSP) applications the critical operations usually involve many multiplications and/or accumulations. For real-time signal processing, a high speed and high throughput Multiplier-Accumulator (MAC) is always a key to achieve a high performance digital signal processing system. In the last few years, the main consideration of MAC design is to enhance its speed. This is because; speed and throughput rate is always the concern of digital signal processing system. But for the epoch of personal communication, low power design also becomes another main design

consideration.

This is because; battery energy available for these portable products limits the power consumption of the system. Therefore, the main motivation of this work is to investigate various Pipelined multiplier/accumulator architectures and circuit design techniques which are suitable for implementing high throughput signal processing algorithms and at the same time achieve low power consumption. A conventional MAC unit consists of (fast multiplier) multiplier and an accumulator that contains the sum of the previous consecutive products. The function of the MAC unit is given by the following equation [5]:

$$F = _ AiBi \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The main goal of a DSP processor design is to enhance the speed of the MAC unit, and at the same time limit the power consumption. In a pipelined MAC circuit, the delay of pipeline stage is the delay of a 1-bit full adder. Estimating this delay will assist in identifying the overall delay of the pipelined MAC. In this work, 1-bit full adder is designed. Area, power and delay are calculated for the full adder, based on which the pipelined MAC unit is designed for low power.

A. High-Speed Booth Encoded Parallel Multiplier Design

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing 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 can be considered as a series of repeated additions. The number to be added is the multiplicand, 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 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 [5].

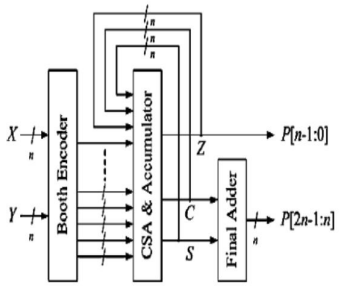


Figure.6 Hardware architecture of the proposed MAC.

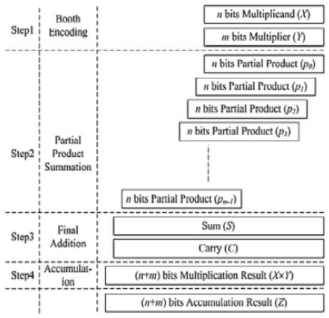


Figure.7 Basic arithmetic steps of multiplication and accumulation.

The basic multiplication principle is twofold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate 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 therefore a multi operand operation. To extend the multiplication to both signed and unsigned.

B. Modified Booth Encoder

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

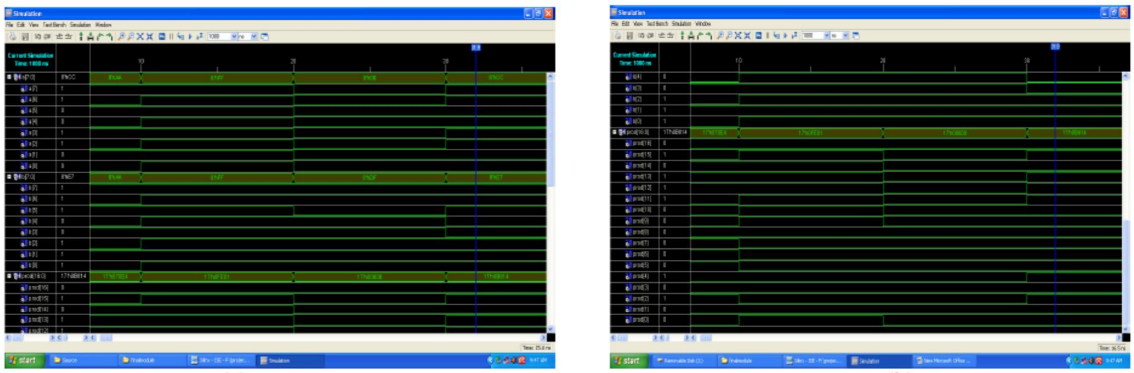
Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied [12]. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the

multiplier. Figure.3 shows the grouping of bits from the multiplier term for use in modified booth encoding. For the partial product generation, we adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits. Figure.11 shows a computing example of Booth multiplying two numbers "2AC9" and "006A". The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals.

IV. ADVANTAGES OF THIS METHOD

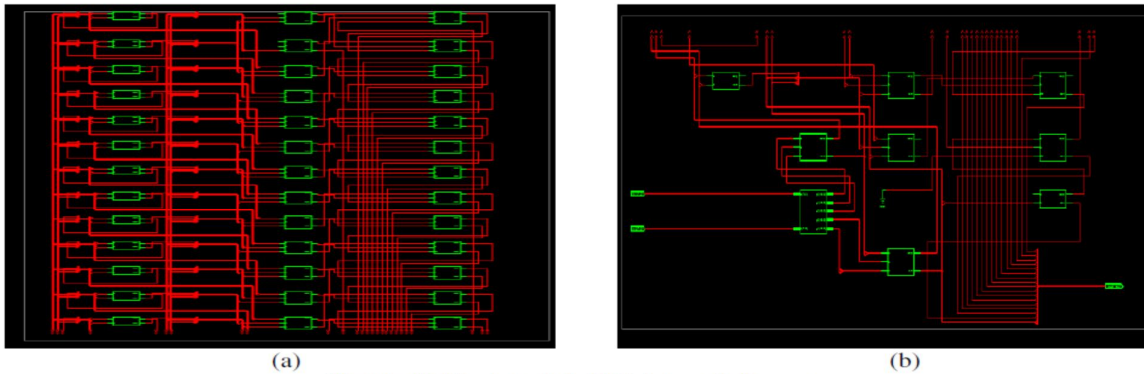
The advantage of this method is the halving of the number of partial products. Reduces the propagation delay, complexity and power consumption in the circuit. Booth multipliers save costs (time and area) for adding partial products. With the higher radix the number of additions is reduced and the redundant Booth code reduces costs for generating partial products in a higher radix system.

V. SIMULATION RESULTS



(a) (b)

Figure.18 Top module timing diagrams



(a) (b)

Figure.19 Final module RTL internal diagram

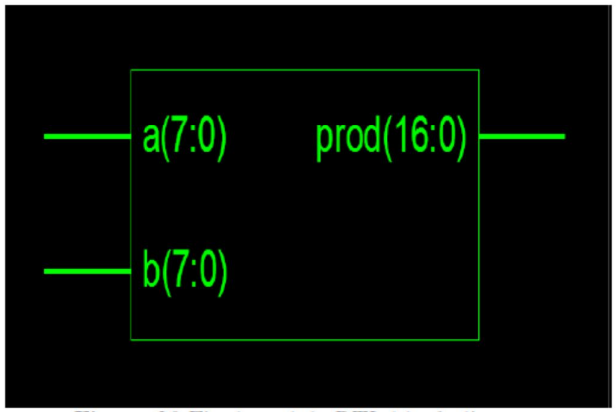


Figure.20 Final module RTL block diagram

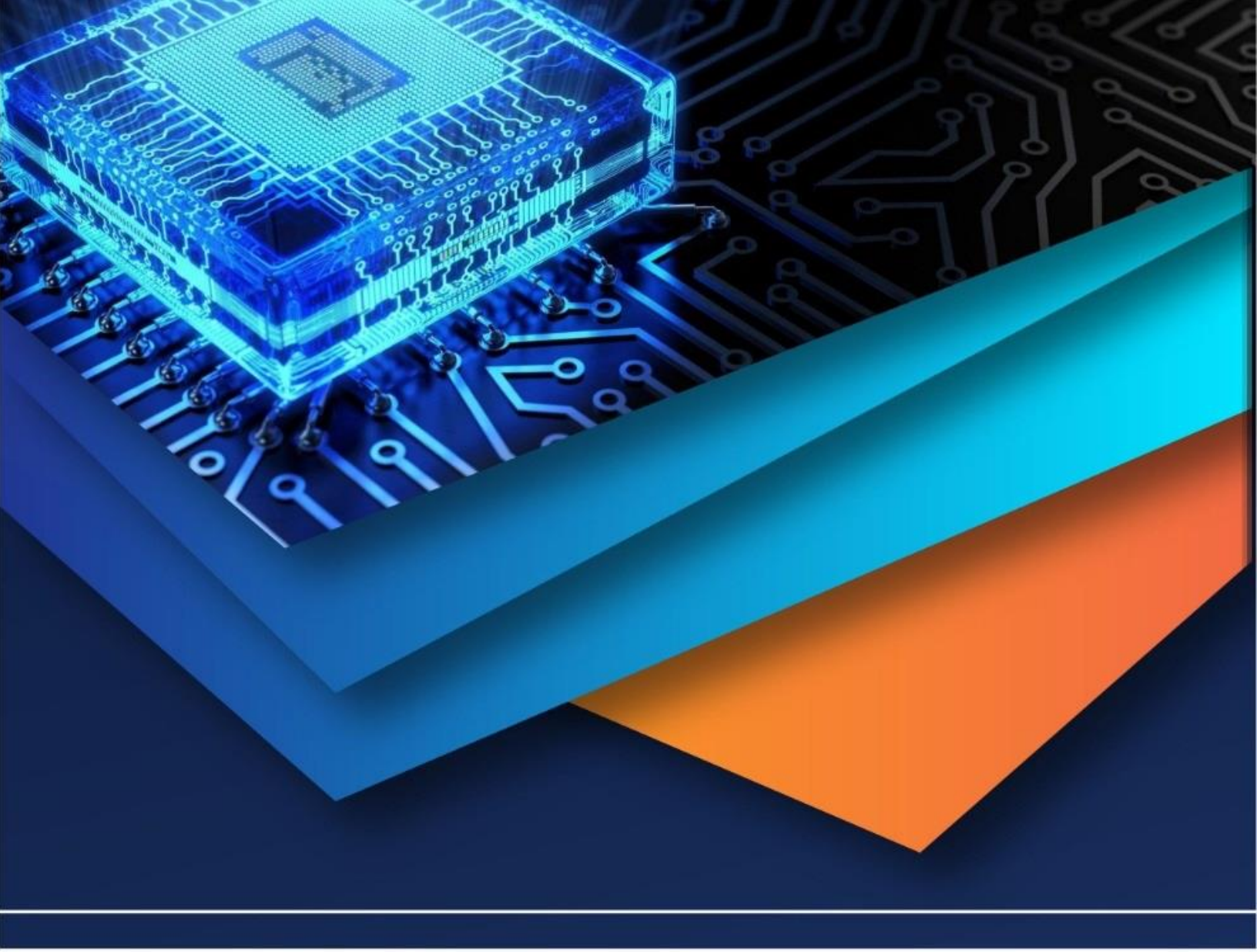
VI. CONCLUSION

This is the advanced and more sophisticated algorithm for designing the Radix-4 based High Speed Multiplier for ALU's Using Minimal Partial Products. Xilinx is used to produce Top module timing diagram and Final module RTL internal diagram. It produces minimum partial products, which intern reduces the critical path delay. Since the DSP processors are common in all digital electronic Devices so it will be useful one. It can be extended to radix-8.but the complexity associated with the radix-8 is high. But partial products will be reduced to n/3.

REFERENCES

[1] Young-Ho Seo and Dong-Wook Kim, "A New VLSI Architecture of arallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm" IEEE Trans. Very Large Scale Integration (VLSI) Systems, Vol. 18, No. 2, Feb 2010 http://www.pgembeddedsystems.com:80/index_files/VLSI IEEE PAPERS.pdf
[2] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw- Hill, 1984.
[3] Information Technology-Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard, ISO/IEC 13818-1, 2, 3, 1994.
[4] JPEG 2000 Part I Fina1119l Draft, ISO/IEC JTC1/SC29 WG1.
[5] O. L. MacSorley, "High speed arithmetic in binary computers," Proc.IRE, vol. 49, pp. 67–91, Jan. 1961.
[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] A. D. Booth, "A signed binary multiplication technique," Quart. J.Math., vol. IV, pp. 236–240, 1952 .http://www.ece.rutgers.edu/~bushnell/dsdwebsite/booth.pdf
[9] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964. http://lapwww.epfl.ch/courses/ comparith/Papers/1_Wallace_mult.pdf
[10] N. R. Shanbag and P. Juneja, "Parallel implementation of a 4_4-bitmultiplier using modified Booth's algorithm," IEEE J. Solid-State Circuits, vol. 23, no. 4, pp. 1010–1013, Aug. 1988.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)