DESIGN AND IMPLEMENTATION OF RADIX-4 BASED HIGH SPEED MULTIPLIER FOR ALU'S USING MINIMAL PARTIAL PRODUCTS

S. Shafiulla Basha[1], Syed. Jahangir Badashah[2]
[1]Asstt. Prof., E.C.E Department, Y.S.R.E.C of Y.V.U, Proddatur, Y.S.R. District, A.P., India
[2]Associate Prof., E.C.E Department, MEC, Kadapa, Y.S.R. District, A.P., 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].

Speed and Size

In this, when performance of circuits is compared, it is always done in terms of circuit speed, size and power. A good estimation of the circuit's size is to count the total number of gates used. The actual chip size of a circuit also depends on how the gates are placed on the chip – the circuit's layout. Since we do not deal with layout in this report, the only thing we can say about this is that regular circuits are usually smaller than non-regular ones (for the same number of gates), because regularity allows more compact layout. The physical delay of circuits originates from the small delays in single gates, and from the wiring between them. The delay of a wire depends on how long it is. Therefore, it is difficult to model the wiring delay; it requires knowledge about the circuit's layout on the chip [1]. The gate delay, however, can easily be modeled by saying that the output is delayed a constant amount of time from the latest input. What we can say about the wiring delay is that larger circuits have longer wires, and hence more wiring delay. It follows that a circuit with a regular layout usually has shorter wires and hence less wiring delay than a non-regular circuit. Therefore, if circuit delay is estimated as the total gate delay, one should also have in minded the circuit's size and amount of regularity, when comparing it to other circuits. "Delay" usually refers to the "worst-case delay". That is, if the delay of the output is dependent on the inputs given, it is always the largest possible output delay that sets the speed. Furthermore, if different bits in the output have different worst-case delays, it is always the slowest bit that sets the delay for the whole output. The slowest path between any input bit and any output bit is called the "critical path".

Objective

The main objective of this paper is to design and implementation of a Multiplier and Accumulator. A multiplier which is a combination of Modified Booth and SPST (Spurious Power Suppression Technique) Wallace tree are designed taking into account the less area consumption of booth algorithm because of less number of partial products and more speedy accumulation of partial products and less power consumption of partial products addition using SPST adder approach. Booth Wallace multiplier is hardware efficient and performs faster than Booth's multiplier. Booth Wallace multiplier consumes 40% less power compared to Booth multiplier. The results reveal that the hardware requirement for implementing hearing aid using Booth Wallace multiplier is less when compared with that of a booth multiplier [9].

1.1 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.

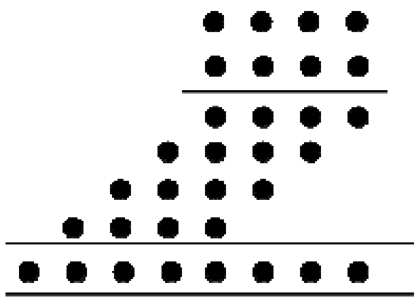


Figure.1 Basic Multiplication

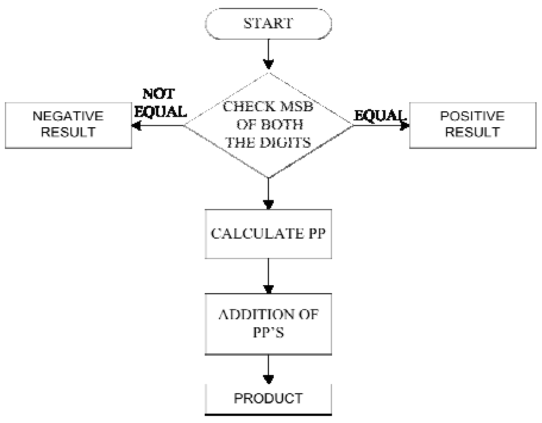


Figure.2 Signed Multiplication Algorithm

Binary Multiplication

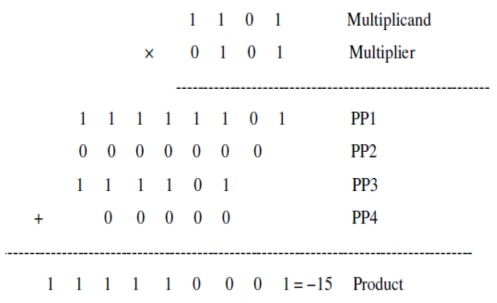
In the binary number system the digits, called bits, are limited to the set [0, 1]. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial-products simple and efficient. Summing these partial-products is the 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 $x = x_{n-1}.....x_1\ x_0$ and the multiplier $y = y_{n-1}.....y_1y_0$ in order to form the product up to n shifted copies of the multiplicand are to be added for unsigned multiplication [2].

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, 1101_2 (-3_{10}) and 0101_2 (5_{10}), 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].



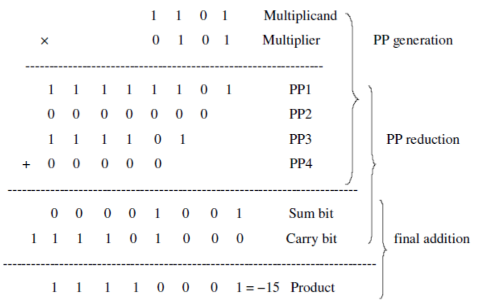


Figure.3 Multiplication calculations by hand **Figure.4** Multiplication Operation in hardware

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

Overview of MAC

A multiplier can be divided into three operational steps. The first is radix-4 Booth encoding in which a 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

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

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm.

$$X = \sum_{i=0}^{N/2-1} d_i 4_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \ldots\ldots\ldots\ldots\ldots (3)$$

If (2) is used, multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

If these equations are used, the afore-mentioned multiplication–accumulation results can be expressed as

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^i Y + \sum_{j=0}^{2N-1} z_i 2^i. \quad \ldots\ldots\ldots\ldots\ldots (5)$$

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.

2.1. 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 A_i x B_i is always fed back into the 32-bit accumulator and then added again with the next product A_i x B_i. This MAC unit is capable of multiplying and adding with previous product consecutively up to as many as times.

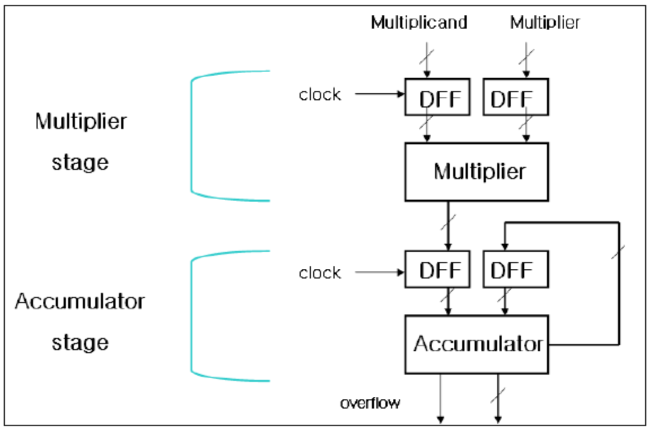


Figure.5 Simple Multiplier and Accumulator Architecture

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]:

$$F = \Sigma A_iB_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

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.

3.1 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].

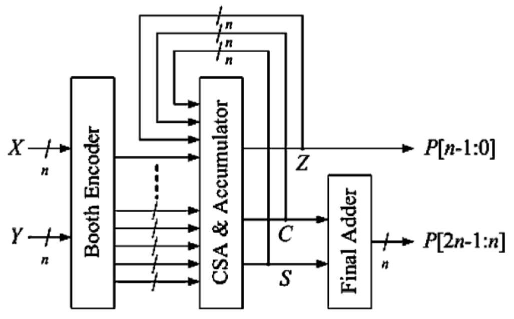


Figure.6 Hardware architecture of the proposed MAC.

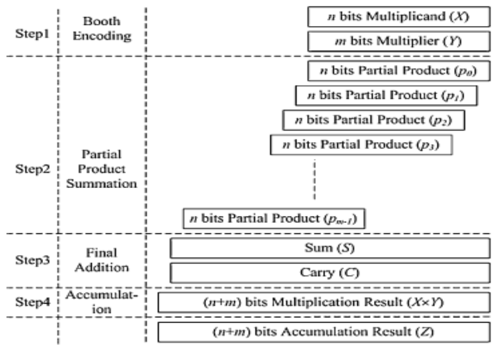

Step1
Booth Encoding
n bits Multiplicand (X)
m bits Multiplier (Y)
Step2
Partial Product Summation
n bits Partial Product (p0)
n bits Partial Product (p1)
n bits Partial Product (p2)
n bits Partial Product (p3)
n bits Partial Product (pm-1)
Step3
Final Addition
Sum (S)
Carry (C)
Step4
Accumulation
(n+m) bits Multiplication Result (X×Y)
(n+m) bits Accumulation Result (Z)

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.

3.2 Derivation of MAC Arithmetic

Basic Concept: If an operation to multiply 2–bit numbers and accumulates into a 2-bit number is considered, the critical path is determined by the 2-bit accumulation operation. If a pipeline scheme is applied for each step in the standard design of Figure.6, the delay of the last accumulator must be reduced in order to improve the performance of the MAC. The overall performance of the proposed MAC is improved by eliminating the accumulator itself by combining it with the CSA function. If the accumulator has been eliminated, the critical path is then determined by the final adder in the multiplier. The basic method to improve the performance of the final adder is to decrease the number of input bits. In order to reduce this number of input bits, the multiple partial products are compressed into a sum and a carry by CSA. The number of bits of sums and carries to be transferred to the final adder is reduced by adding the lower bits of sums and carries in advance within the range in which the overall performance will not be degraded. A 2-bit CLA is used to add the lower bits in the CSA.
In addition, to increase the output rate when pipelining is applied, the sums and carries from the CSA are accumulated instead of the outputs from the final adder in the manner that the sum and carry from the CSA in the previous cycle are inputted to CSA. Due to this feedback of both sum and carry, the number of inputs to CSA increases, compared to the standard design and. In order to efficiently solve the increase in the amount of data, CSA architecture is modified to treat the sign bit.
Equation Derivation: The aforementioned concept is applied to express the proposed MAC arithmetic. Then, the multiplication would be transferred to a hardware architecture that complies with

the proposed concept, in which the feedback value for accumulation will be modified and expanded for the new MAC.
First, if the multiplication in (4) is decomposed and rearranged, it becomes

$$X \times Y = d_0 2Y + d_1 2^2 Y + d_2 2^4 Y + \ldots + d_{N/2-1} 2^{N-2} Y. \quad \ldots(7)$$

If this is divided into the first partial product, sum of the middle partial products, and the final partial product, it can be expressed as. The reason for separating the partial product addition as is that three types of data are fed back for accumulation, which are the sum, the carry, and the pre added results of the sum and carry from lower bits.

$$X \times Y = d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y. \quad \ldots\ldots\ldots\ldots\ldots(8)$$

Now, the proposed concept is applied to in (5). If is first divided into upper and lower bits and rearranged, (8) will be derived. The first term of the right-hand side in (8) corresponds to the upper bits. It is the value that is fed back as the sum and the carry. The second term corresponds to the lower bits and is the value that is fed back as the addition result for the sum and carry.

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

The second term can be separated further into the carry term and sum term as

$$\sum_{i=N}^{2N-1} z_i 2^i = \sum_{i=0}^{N-1} z_{N+i} 2^i 2^N = \sum_{i=0}^{N-2} (c_i + s_i) 2^i 2^N. \quad \ldots\ldots\ldots\ldots\ldots(10)$$

Thus,

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N. \quad \ldots\ldots\ldots\ldots\ldots(11)$$

the MAC arithmetic i

$$P = \left(d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y \right) + \left(\sum_{i=0}^{N-1} z_i 2^i 2^N + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N \right). \quad \ldots(12)$$

$$P = \left(d_0 2Y + \sum_{i=0}^{N-1} z_i 2^i \right) + \left(\sum_{i=1}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{N-2} c_i 2^i 2^N \right) + \left(d_{N/2-1} 2^{N-2} Y + \sum_{i=0}^{N-2} s_i 2^i 2^N \right). \quad \ldots\ldots(13)$$

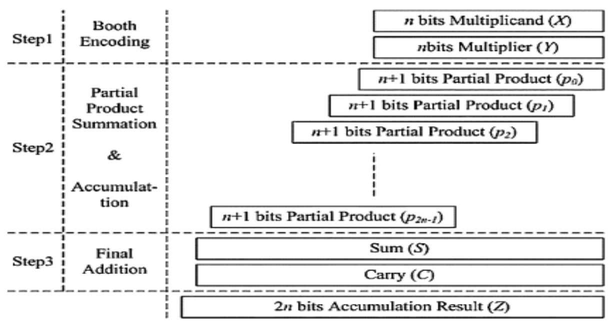

Step1
Booth Encoding
n bits Multiplicand (X)
n bits Multiplier (Y)
Step2
Partial Product Summation & Accumulation
n+1 bits Partial Product (p0)
n+1 bits Partial Product (p1)
n+1 bits Partial Product (p2)
n+1 bits Partial Product (p2n-1)
Step3
Final Addition
Sum (S)
Carry (C)
2n bits Accumulation Result (Z)

Figure.8 Proposed arithmetic operation of multiplication and accumulation.

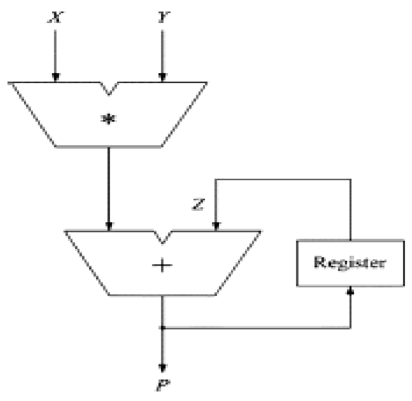


Figure.9 Hardware architecture of general MAC.

3.3 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.

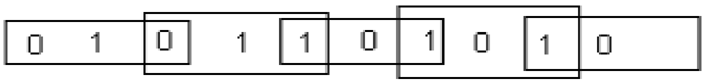


Figure.10 Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product [15]. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table.1 shown below

Table.1 booth-4 encoding

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

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.

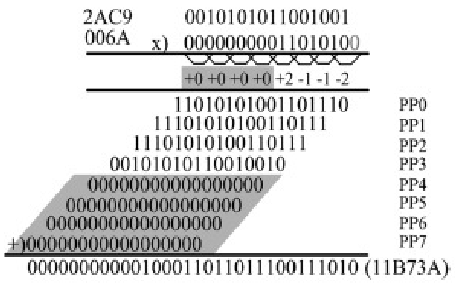


Figure.11 Illustration of multiplication using modified Booth encoding.

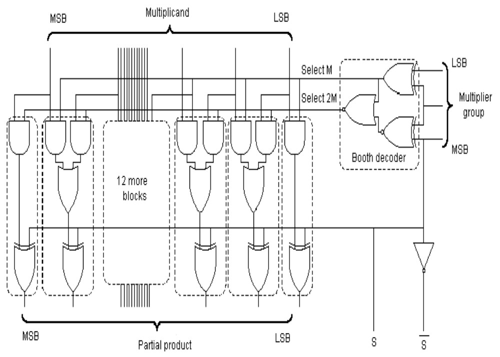


Figure.12 Booth partial product selector logic

The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

Partial product generator

The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2's complement it is negative.

The partial product is generated by doing AND between ‘a’ and ‘b’ which are a 4 bit vectors as shown in Figure. If we take, four bit multiplier and 4-bit multiplicand we get sixteen partial products in which the first partial product is stored in ‘q’. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

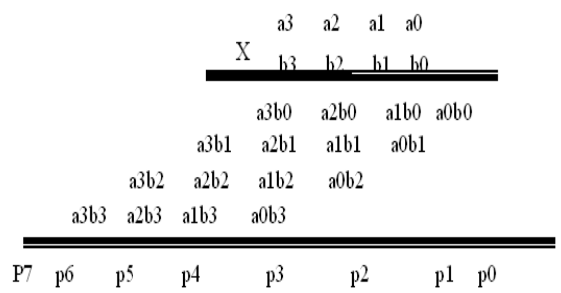


Figure.13 Booth partial products Generation.

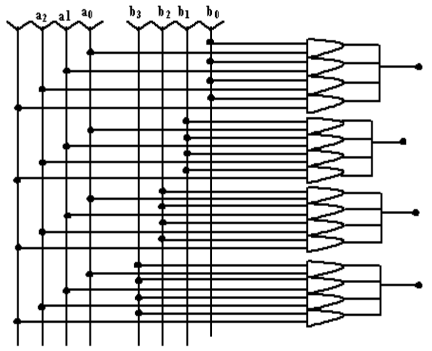


Figure.14 Booth single partial product selector logic

The multiplication second step reduces the partial products from the preceding step into two numbers while preserving the weighted sum. The sough after product P is the sum of those two numbers. The two numbers will be added during the third step The "Wallace trees" synthesis follows the Dadda's algorithm, which assures of the minimum counter number. If on top of that we impose to reduce as late as (or as soon as) possible then the solution is unique. The two binary number to be added during the third step may also be seen a one number in CSA notation (2 bits per digit) [13].

Multiplication consists of three steps:

1) The first step to generate the partial products;
2) The second step to add the generated partial products until the last two rows are remained;
3) The third step to compute the final multiplication results by adding the last two rows.

The modified Booth algorithm reduces the number of partial products by half in the first step. We used the modified Booth encoding (MBE) scheme proposed in. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table.2 shows the rules to generate the encoded signals by MBE scheme and Figure.15 shows the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals as shown in Figure.16

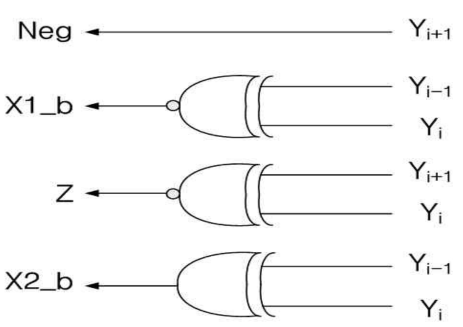


Figure.15 Booth Encoder

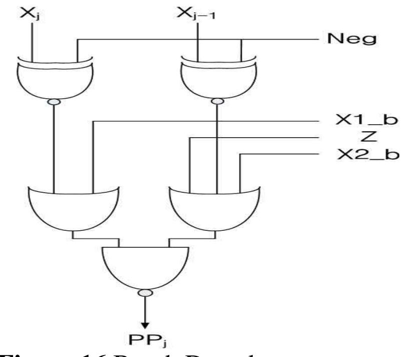


Figure.16 Booth Decoder

Figure.14 shows the generated partial products and sign extension scheme of the 8-bit modified Booth multiplier. The partial products generated by the modified Booth algorithm are added in parallel using the Wallace tree until the last two rows are remained [9]. The final multiplication results are generated by adding the last two rows. The carry propagation adder is usually used in this step.

Table.2 Truth table for MBE Scheme

Y_{i+1}	Y_i	Y_{i-1}	Value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

Table.3 Characteristics of CSA

	[6]	[17]	The Proposed
Number System	2's Complement	1's Complement	1's Complement
Sign Extension	Used	Used	Not Used
Accumulation	Result Data of Final Addition	Result Data of Final Addition	Sum and Carry of CSA
CSA Tree	FA, HA	FA, 2 bits CLA	FA, HA, 2 bits CLA
Final Adder	$2n$ bits	$(n+2)$ bits	n bits

3.4 Proposed CSA Architecture

The architecture of the hybrid-type CSA that complies with the operation of the proposed MAC is shown in Figure.3.17, which performs 8-bit operation [7]. In Figure.17 S_i is to simplify the sign expansion and N_i is to compensate 1's complement number into 2's complement number. S[i] and C[i] correspond to the i^{th} bit of the feedback sum and carry. Z[i] is the i^{th} bit of the sum of the lower bits for each partial product that were added in advance and Z'[i] is the previous result. In addition, $P_j[i]$ corresponds to the i^{th} bit of the j^{th} partial product. Since the multiplier is for 8 bits, totally four partial products are generated from the Booth encoder. This CSA requires at least four rows of FAs for the four partial products. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For n X n -bit MAC operation, the level of CSA is (n/2+1). The white square in Figure.17 represents an FA and the gray square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input.

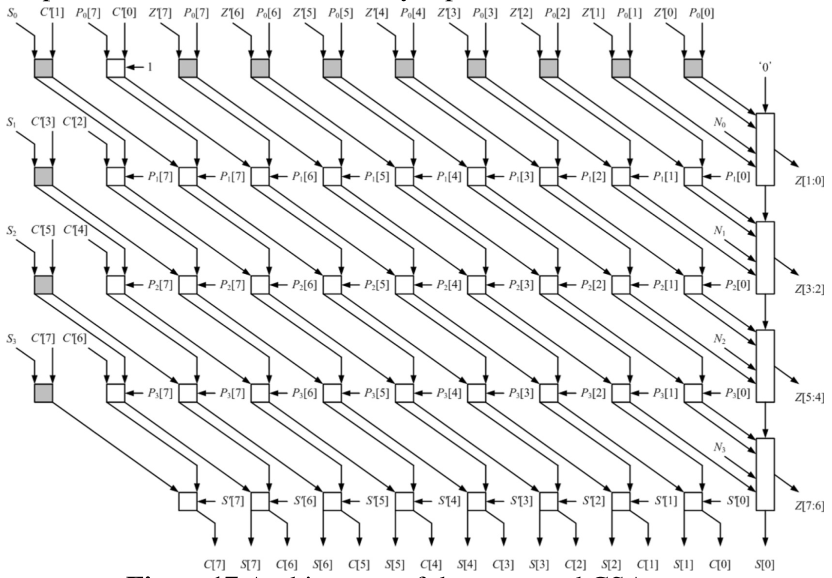
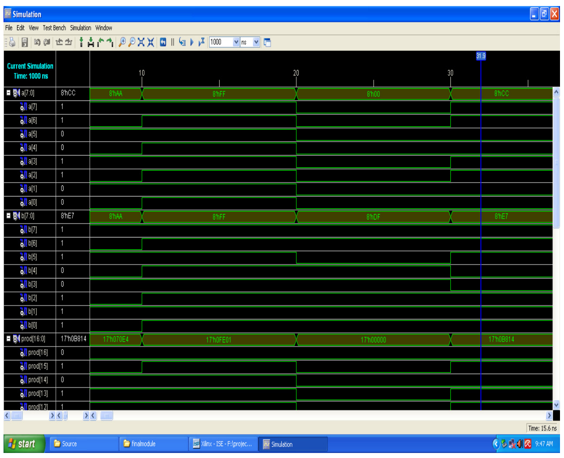


Figure17 Architecture of the proposed CSA tree.

The critical path in this CSA is determined by the 2-bit CLA. It is also possible to use FAs to implement the CSA without CLA. However, if the lower bits of the previously generated partial product are not processed in advance by the CLAs, the number of bits for the final adder will increase. When the entire multiplier or MAC is considered, it degrades the performance. In Table.3, the characteristics of the proposed CSA architecture have been summarized and briefly compared with other architectures. For the number system, the proposed CSA uses 1'scomplement, but ours uses a modified CSA array without sign extension. The biggest difference between ours and the others is the type of values that is fed back for accumulation. Ours has the smallest number of inputs to the final adder.

IV. Simulation Results


(a)

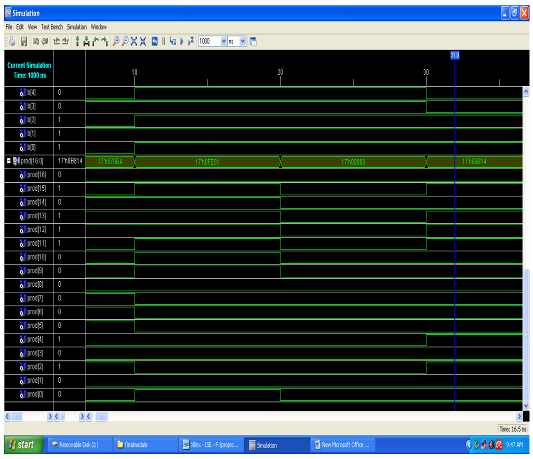
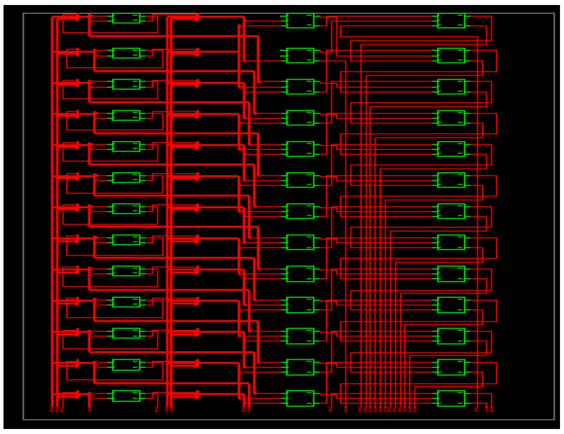

(b)

Figure.18 Top module timing diagrams


(a)

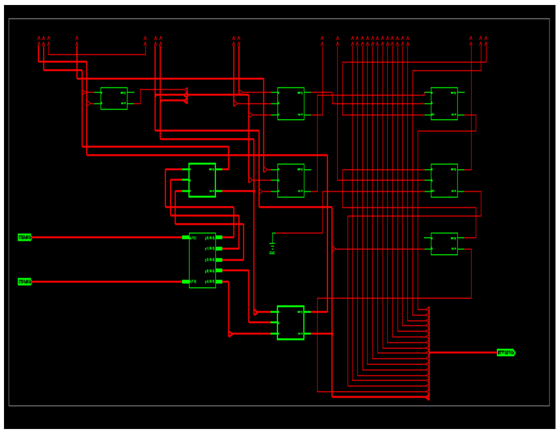

(b)

Figure.19 Final module RTL internal diagram

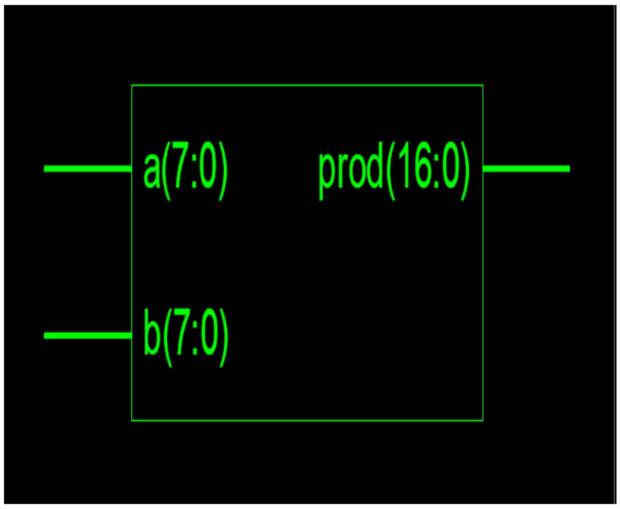


Figure.20 Final module RTL block diagram

V. 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.

Low power consumption is there in case of radix-4 booth multiplier because it is a high speed parallel multiplier.

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 Fina11191 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.

[11] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54_54 regularstructured tree multiplier," IEEE J. Solid-State Circuits, vol. 27, no. 9, pp. 1229–1236, Sep. 1992.

[12] J. Fadavi-Ardekani, "M_N Booth encoded multiplier generator using optimizedWallace trees," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 1, no. 2, pp. 120–125, Jun. 1993.

[13] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K.Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54_54 multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, vol. 30, no. 3, pp. 251–257, Mar. 1995. http://www.ece.ucdavis.edu/~vojin/CLASSES/EEC280/Web-page/papers/Use%20of%20Pass-Transistor%20Logic/54x54mult-CMOS-Okhubo-CICC94.pdf

[14] A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed-point IIR digital filters," J. Circuits, Syst.,Comput., vol. 7, no. 3, pp. 191–209, 1997.

[15] A. Tawfik, F. Elguibaly, M. N. Fahmi, E. Abdel-Raheem, and P.Agathoklis, "High-speed area-efficient inner-product processor," Can. J. Electr. Comput. Eng., vol. 19, pp. 187–191, 1994.

[16] XILINX Synthesis and Simulation Design Guide. **http://**www.**xilinx**.com/itp/**xilinx**10/books/docs/sim/sim.pdf

Biographies

Shaik Shafiulla Basha received B.Tech. degree in Electronics & Communication Engineering from Sri Venkateswara University, Tirupathi in 2001, M.Tech. in Digital Systems and Computer Electronics from Jawaharlal Nehru Technological University, Hyderabad in 2006. He is having an experience of 9 years, in the field of teaching, presently working as Assistant Professor in the department of ECE, Y.S.R. Engineering College of Yogi Vemana University. He is a life time member of IETE & ISTE.

Syed Jahangir Badashah received B.E. degree in Electronics & Communication Engineering from Gulbarga University, in 2002, M.E.in Applied Electronics from Sathyabama University in 2005.He is currently doing research in image processing from Sathyabama University. He is having an experience of 10 years, in the field of teaching, presently working as Associate Professor in the department of ECE, Madina Engg College, Kadapa. He is a life time member of IETE & ISTE.