IOSR Journal of Engineering (IOSR JEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 59-63

Path Delay Optimized Booth Radix-8 Architecture

.s.abinaya , h.fathima , hema .k.s, h.johara, A.Rajeswari
ug student,final year ece department, dhanalkshmi srinivasan college of engineering and technology, mamallapuram,kancheepuram district.
ug student,final year ece department, dhanalkshmi srinivasan college of engineering and technology, mamallapuram,kancheepuram district.
ug student,final year ece department, dhanalkshmi srinivasan college of engineering and technology, mamallapuram,kancheepuram district.
ug student,final year ece department, dhanalkshmi srinivasan college of engineering and technology, mamallapuram,kancheepuram district.
: M.E Assistant professor dhanalkshmi srinivasan college of engineering and technology, mamallapuram,kancheepuram district.

Abstract: *To accomplish of cellular systems for 5G field emphatically . To investigate various antenna and diversity selection in MU- MIMO system with help of CSI framework. To maximize the efficiency of the system time domain pilot aided channel estimation technique are proposed for multiple users. To analyze the performance of this novel techniques for MU MIMO shows better utility compared to Other technique.*

I. Introduction

Introduction :

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate(MAC) and inner product are among some of the frequently used Computation- Intensive Arithmetic Functions(CIAF) currently implemented in many Digital Signal Processing (DSP)applications such as convolution, Fast Fourier Transform(FFT), filtering and in microprocessors in its arithmetic and logic unit [1]. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip. The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications.

Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal andimage processing applications [2]. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications.

1.2 Arithmetic Operations and Units

The tasks of a VLSI chip are the processing of data and the control of internal or external system components. This is typically done by algorithms which are based on logic and arithmetic operations on data items [10]. Applications of arithmetic operations in integrated circuits are manifold. Microprocessors and DSPs typically contain adders and multipliers in their data path. Special circuit units for fast division and square-root operations are sometimes included as well. Adders, incrementers/decrementers, and comparators are often used for address calculation and flag generation purposes in controllers.

ASICs use arithmetic units for the same purposes. Depending on their application, they may even require dedicated circuit components for special arithmetic operators, such as for finite field arithmetic used in cryptography, error correction coding, and signal processing. Some of the basic arithmetic operations are listed below [2].

- Shift/extension operations
- Equality and magnitude comparison
- Incrementation / Decrementation
- Negation
- Addition/ Subtraction
- Multiplication
- Division 3
- Square root
- Exponentiation
- Logarithmic Functions

• Trigonometric Functions

For trigonometric and logarithmic functions as well as for exponentiation, various iterative algorithms exist which make use of simpler arithmetic operations. Multiplication, division and square root can be performed using serial or parallel methods. In both methods, the computation is reduced to a sequence of conditional additions/subtractions and shift operations.

Existing speed-up techniques try to reduce the number of required addition/subtraction operations to improve their speed. Subtraction corresponds to the addition of a negated operand. The addition of two n-bit binary numbers can be regarded as an elementary operation. The algorithm for negation of a number depends on the chosen number representation [42] and is usually accomplished by bit inversion and incrementation. Increment and decrement operations are simplified additions with one input operand being constantly 1 or -1. Equality and magnitude comparison operations can also be regarded as simplified additions with only some of the respective addition flags, but no sum bits are used as outputs [3].

1.2.1 Implementation in Hardware:

This short overview shows that the addition is the key arithmetic operation, which most other operations are based on. Its implementation in hardware is therefore crucial for the efficient realization of almost every arithmetic unit in VLSI.

This is in terms of circuit size, computation delay, and power consumption. Addition is a prefix problem [3], which means that each result is dependent on all input bits of equal or lower magnitude. Propagation of a carry signal from each bit position to all higher bit positions is necessary. Carry - propagate adders [31] perform this operation immediately.

The required carry - propagation from the least to the most significant bit results in a considerable circuit delay, which is a function of the word length of the input operands [5]. The most efficient way to speed up addition is to avoid carry propagation thus saving the carries for later processing. This allows the addition of two or more numbers in a very short time, but yields results in a redundant number representation [2]. The redundant representation forms the basis of carry-save adders [2]. They play an important role in the efficient implementation of multi-operand addition circuits.

They are very fast, their structure simple, but the potential for further optimization is minimal. The binary carry-propagate adder therefore, is one of the most often used and most crucial building blocks in digital VLSI design. Various well-known methods exist for speeding up carry - propagation in adders, offering very different performance characteristics, advantages and disadvantages.

II. Existing Methods

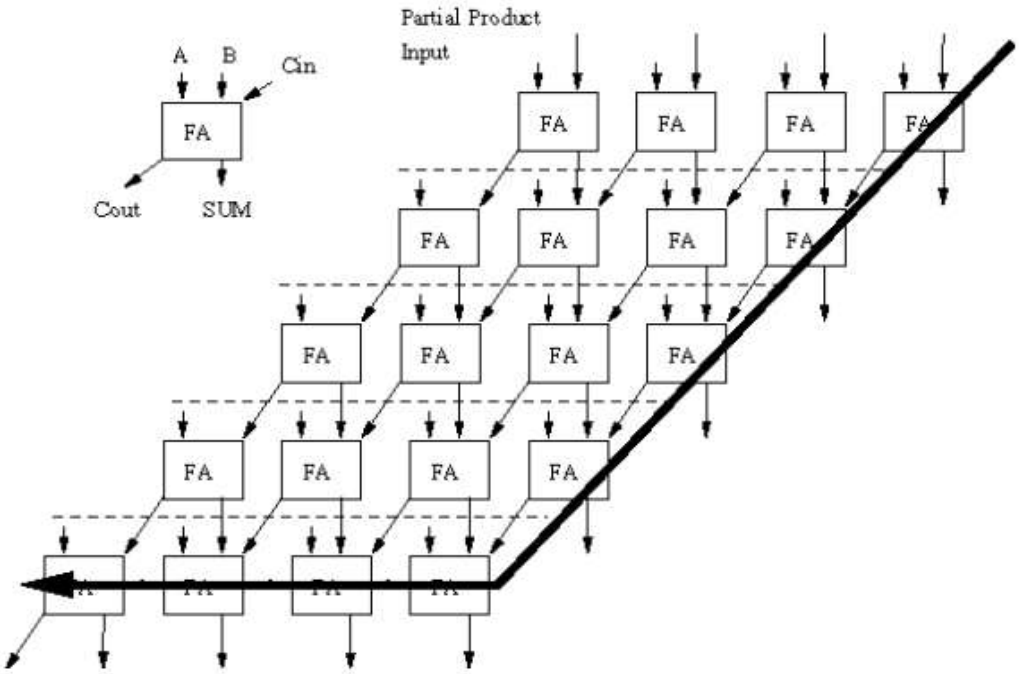
Overview of MAC:

A multiplier can be divided into three operational steps. The first is radix-2Booth 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 architecture of a multiplier, which is the fastest, uses radix-2Booth encoding that generates partial products. If radix-2 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 adoptive Booth encoding.

Multipliers Methods :

Array multiplier :

The array multiplier originates from the multiplication parallelogram. As shown in Figure, each stage of the parallel adders should receive some partial product inputs. The carryout is propagated into the next row. The bold line is the critical path of the multiplier. In a non pipelined array multiplier, all of the partial products are generated at the same time. It is observed that the critical path consists of two parts: vertical and horizontal. Both have the same delay in terms of full adder delays and gate delays. For an n-bit by n-bit array multiplier, the vertical and the horizontal delays are both the same as the delay of an n-bit full adder. For a 16- bit 6



by 16-bit multiplier, the worst-case delay is 32 where is the worst-case full adder delay.

One advantage of the array multiplier comes from its regular structure. Since it is regular, it is easy to layout and has a small size. The design time of an array multiplier is much less than that of a tree multiplier. A second advantage of the array multiplier is its ease of design for a pipelined architecture. A fully pipelined array of the multiplier with a stage delay equal to the delay of a 1-bit full adder plus a register has been successfully designed for high-speed DSP applications at Bell Laboratories. Also it can be easily pipelined by inserting latches after CSA (carry save adders) or after every few rows.

Tree Multiplier:

The result of the multiplication is obtained by first generating partial products and then adding the partial products. The critical path of a multiplier depends on the delay of the carry chain through all of the adders. Consider a full adder with A and B as the adder inputs and C is the carry input. The full adder produces a bit of summand and a bit of carry out.

III. Working Of The Project

Introduction

In the modern world, we need system which will run at high speed. Multipliers play an important part in today's DSP and DIP applications. In our case for DCT computation multiplications are used larger in number. Therefore, speed improvement in multiplier is important. Advances in technology have permitted many researchers to design multipliers which offer both high-speed and unique hardware structure, thereby making them suitable for specific VLSI implementation.

In any multiplication algorithm, the multiplication operation is carried out by summation of decomposed partial products. For high-speed multiplication we need to apply a booth radix recoding multiplication algorithm. In recent days in all high radix booth algorithm recoding is changed from 2s-complement format to a signed-digit representation from the defined set. This is called modified booth algorithm..

Booth's Algorithm

An ALU addition operation can be very time consuming when done repeatedly. Recognizing the fact that computers can shift bits faster than adding bits, Booth developed an algorithm, which reduces the number of additions that take place when multiplying two numbers [1]. Booth's algorithm multiples two numbers in 2's complement form [1].

It creates an initial guess for the product, which is zeros followed by the multiplier on the right half of the product [1]. Instead of using one bit of the 9 multiplier to determine whether we need to add and shift or merely shift the intermediate step of multiplication, Booth's algorithm uses two right most bits of the product to determine the next step [1]. The algorithm [1] for 4-bit multiplier is outlined in this section. This algorithm was expanded to 64-bit when it was implemented in verilog code.

Booth's Algorithm:
1) Let multiplicand = 4 bits, multiplier = 4 bits, and output = 8 bits.
Product: {0000, Multiplier, 0}
2) Now, take the two least significant bits of the product and depending on the value proceed with one the following:
a) 00:No changes to product.
b) 01:Add the multiplicand to the left side of the product.
c) 10: Subtract the multiplicand from the left side of the product.
d) 11:No changes to product.
3) Right shift the product by 1 bit.
4) Repeat the process x number of times, where x is the number of bits in the multiplicand.

In step two, the two least significant bits of the product are "10", therefore, in step three, the multiplicand is subtracted from the four most significant bits of the product and the product is shifted right 1 bit. Now, the two least significant bits are "01" which translates to adding the multiplicand to the left side of the product and shift 1 bit to the right. The procedure has a total of four steps since the multiplicand is only four bits wide. Shift the product one last time to the right to arrive at the final answer. As a check, 4 multiplied by 4 is equal to 16, which in 10000 in binary.

Booth's Algorithm Example 1:
Multiplicand: 410 = 01002
Multiplier: 410 = 01002

Multiplicand	Step	Product
0100	Initial	0000 0100 0
0100	Shift	0000 0010 0

Input			R4 Digit	Output		
b_{2j+1}	b_{2j}	b_{2j-1}	y_j^{R4}	$sign_j$	$\times 2_j$	$\times 1_j$
0	0	0	0	0	0	0
0	0	1	1	0	0	**1**
0	1	0	1	0	0	**1**
0	1	1	2	0	**1**	0
1	0	0	-2	**1**	**1**	0
1	0	1	-1	**1**	0	**1**
1	1	0	-1	**1**	0	**1**
1	1	1	0	**1**	0	0

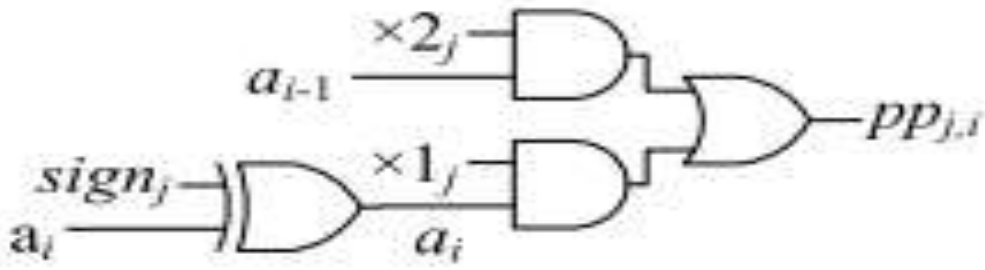


Figure 3.2 i-bit partial product generator based on accurate radix-4 encoding and the approximate

IV. Performance

Speed:

Booth multiplier is faster than array multiplier and Booth multiplier. As the number of bits increases from 8x8 bits to 16x16 bits, the timing delay is greatly reduced for Booth multiplier as compared to other multipliers. Booth multiplier has the greatest advantage as compared to other multipliers over gate delays and regularity of structures.

Area:

The area needed for Booth multiplier is very small as compared to other multiplier architectures i.e. the number of devices used in Booth multiplier are 213 while Booth and Array Multiplier is 349 .Thus the result shows that the Booth multiplier is smallest and the fastest of the reviewed architectures. The Booth cube architecture proved to exhibit improved efficiency in terms of speed and area compared to Booth and Array Multiplier. Due to its parallel and regular structure, this architecture can be easily realized on silicon and can work at high speed without increasing the clock frequency.

It has the advantage that as the number of bits increases, the gate delay and area increase very slowly as compared to the square and cube architectures of other multiplier architecture. Speed improvements are gained by parallelizing the generation of partial products with their concurrent summations. It is demonstrated that this design is quite efficient in terms of silicon area/speed. Such a design should 18 enable substantial savings of resources in the FPGA when used for image/video processing application.

V. Conclusions

In this paper, we analyze the performance of modified Booth and approximation based implication for improved system performance. Initially we analyzes the Booth and its funtionlity using MODEL SIM. The proposed algorithm is based on LSB encoded approximation to finely save external resource utilization. We also illustrated the performance of proposed booth algorithm in numerical simulations, and our algorithm shows a significant complexity reduction and energy measure improvement compared to conventional booth, while the trade off is much lower compared to conventinal approach.