www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 6 Issue 2 Feb. 2017, Page No. 20308-20312
Index Copernicus Value (2015): 58.10, DOI: 10.18535/ijecs/v6i2.28

Review on 32-Bit IEEE 754 Complex Number Multiplier Based on FFT Architecture using BOOTH Algorithm

Ms. Anuja A. Bhat[1] , Prof. Mangesh N.Thakare[2]
[1](Department of EXTC, B. D. College of Engineering,Sewagram Wardha ,RTMNU, INDIA)
bhatanuja@gmail.com
[2(]Department of EXTC, B. D. College of Engineering, Sewagram Wardha ,RTMNU, INDIA)
mnt_ent@rediffmail.com

ABSTRACT: *In this paper, we have shown an review of High Speed, less power and less delay 32-bit IEEE 754 Floating Point Complex Multiplier using Booth Algorithm which includes 32-bit Floating Point Adder, Subtractor and Multiplier. Multiplication is an important fundamental function in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform (FFT), filtering and in microprocessors in its arithmetic and logic unit (ALU). Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. The main objective of this research is to reduce delay, power and to increase the speed. The coding will be done in VHDL, synthesis and simulation will be done using Xilinx ISE simulator.*

KEYWORD: *Adder, Subtractor , Multiplier , VHDL, Xilinx.*

1. INTRODUCTION

The fundamental and the core of all the digital signal processors (DSPs) are its multipliers, and the speed of the DSPs is mainly determined by the speed of its multiplier. Multipliers are key components of many high performance systems such as microprocessors, FIR filters, digital signal processors, etc. Performance of a system is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Since multiplication dominates the execution time of most DSP application so there is need of high speed multiplier. Complex number operations are the backbone of many digital signal processing algorithms, which mostly depend on extensive number of multiplication. Complex multiplication is of immense importance in Digital Signal Processing (DSP) and Image Processing (IP).To implement the hardware module of Discrete Fourier Transformation (DFT), Discrete Cosine Transformation (DCT), Discrete Sine Transformation (DST) and modern broadband communications; large numbers of complex multipliers are required.

1.1 COMPLEX NUMBER MULTIPLICATION

Complex number multiplication is performed using four real number multiplications and two additions/ subtractions. In real number processing, carry needs to be propagated from the least significant bit (LSB) to the most significant bit (MSB) when binary partial products are added. Therefore, the addition and subtraction after binary multiplications limit the overall speed. Furthermore, multiplier is generally the most area consuming. Hence, optimizing the area and speed of the multiplier is a major design issue. However, speed and area are usually conflicting constraints so that improving speed results mostly in larger areas. A whole spectrum of multipliers with different area speed constraints has been designed with fully serial multipliers at one end of the spectrum and fully parallel Multipliers at the other end. Multipliers have moderate performance in both speed and area. Binary floating point numbers multiplication is one of the basic functions used in digital signal processing (DSP) application.

Example: - A = 3 + 2j& B = 1 + 7j
C = A * B
C = (3 + 2j)(1 + 7j)
C= (3×1) + (3×7j) + (2j×1) + (2j×7j)
C= 3 + 21j + 2j + $14j^2$
C = 3 + 21j + 2j - 14 ---(since $j^2 = -1$)
C = -11 + 23j

1.2 IEEE 754 FORMAT

The IEEE 754 standard provides the format for representation of Binary Floating point numbers in computers. The Binary Floating point numbers are represented in Single precision and Double precision formats. The Single precision format consists of 32 bits and the Double precision format consists of 64 bits. The formats are composed of 3 fields; Sign, Exponent and Mantissa.. Following figure shows IEEE Format for single and double precision bit.

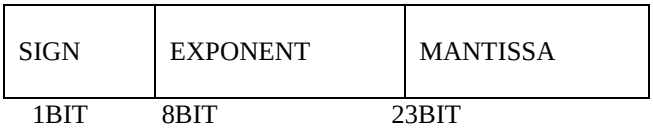

Fig 1.1 : IEEE Format for single precision

The performance of any processor depends upon its power dissipation and delay, so there is always the need of high speed multipliers and the performance of any floating point multiplier depends upon the performance of mantissa calculation unit**.** Floating point arithmetic is considered to be a very interesting topic for researchers as floating point numbers are widely used in many applications and they are also able to retain their resolution and accuracy as compared to fixed point numbers.

Nowadays, almost every language has a floating point data type, computers from PCs to supercomputers have floating point compilers and every operating system must respond to floating point exceptions. The major application areas of floating point numbers today are in the field of medical applications, image processing, motion sensing, scientific computing, audio applications, DSP applications etc. These applications usually involve floating point calculations with single as well as double precision format. The double precision format nowadays very commonly used on PCs, because it has wider range of numbers compared to single precision. For this reason, most of the designers of computer systems provide support for executing double-precision format along with single- precision format. Today, the need for faster processing speed is continuously driving the researchers towards major improvements in existing technologies, as well as to search for new algorithms also. With day to day development and advancement in technology, there is a need to design and implement low power multiplier using Booth algorithm technique.

1.3 BOOTH ALGORITHM

Multiplication is an important fundamental function in arithmetic operation. Signed multiplication is a careful process. With unsigned multiplication there is no need to take the sign of the number into consideration. However in signed multiplication the same process cannot be applied because the signed number is in a 2's compliment form which would yield an incorrect result if multiplied in a similar fashion to unsigned multiplication. That's where Booth's algorithm comes in. Booth's algorithm preserves the sign of the result.

Booth's algorithm is a well known method for 2's complement multiplication. It speeds up the process by analyzing multiple bits of multiplier at a time. This widely used scheme for two's complement multiplication was designed by Andrew D. Booth in 1951. Booth algorithm is an elegant way for this type of multiplication which treats both positive and negative operands uniformly. It allows n bit multiplication to be done using fewer than n additions or subtractions, thereby making possible faster multiplication. Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

There are 2 methods that you should know before attempting Booth's algorithm. Right shift circulant and right-shift arithmetic.

Right-shift circulant (**RSC)**, is simply shifting the bit, in a binary string, to the right 1 bit position and take the last bit in the string and append it to the beginning of the string.

Example:

10110

after right-shift circulant now equals – 01011

Right-shift arithmetic (**RSA)**, is where you add 2 binary number together and shift the result to the right 1 bit position.

Example:

0100

+0110

result = 1010

Now shift all bits right and put the first bit of the result at the beginning of the new

string:

result 1010

shift 11010

According to Booth's multiplication algorithm among the two input binary numbers the one with minimum number of bit changes is considered as multiplier and the other as a multiplicand in order to reduce the time taken for calculating the multiplication product.

The steps for performing booth multiplication are as follows:

Let the multiplicand be 'B' and multiplier be 'Q'.

Assume initially value of 'A' and 'Q-1' is zero.

The main step is to check last two bits.

There will be iterations according to the number of multiplier.

For example, if the multiplier is of 2-bit then 2 Iterations will be done, for 4-bit multiplier 4 iterations are done, and so on.

Now, the algorithm starts, first the last two digits are checked and if the two bits are "00" or "11" the only Arithmetic Right Shift is done.

And if the last two bits are "01", then A is added with B, and result is stored into A.

If the last two bits are "10", then A is subtracted from B, and result is stored into A.

Finally, the result obtained is coded in binary form which gives the desired output.

In this way multiplication of any two numbers is performed using booth algorithm.

Example :-

Multiply **14** * **-5** using 5-bit numbers.

14 in binary : 01110

-14 in binary : 10010

5 in binary : 00101

-5 in binary : 11011

Result : **-70** in binary : **11101 11010** (10-bit result).

2.RELATED WORK

In this paper, the author proposed a design of complex floating point multiplier technique. The FFT/IFFT results shown by using this technique consumes very less resources in terms of slices, flipflops and multipliers to provide a cost effective solution for DSP applications. The proposed design has a high efficiency and high accuracy [1].

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 to 11/2 and compress partial product term by a

ratio of 3:2. Carry save-adders (CSA) is used to enhance the speed of addition process for the system [2].

In this paper, the author presented a high speed complex multiplier design (ASIC) using Vedic Mathematics. The idea for designing the multiplier and adder/ subtractor unit is adopted from ancient Indian mathematics"Vedas". The partial products and sums are generated in one step which reduces the carry propagation from LSB to MSB. The implementation of the Vedic mathematics and their application to the complex multiplier ensure substantial reduction of propagation delay in comparison with DA based architecture and parallel adder based implementation which are most commonly used architectures [3].

This paper presented designing of complex number multiplier .An ancient Indian mathematics "Vedas" is used for designing the multiplier unit. The Urdhva Tiryakb-hyam sutra (method) was selected for implementation complex multiplication. Any multi-bit multiplication can be reduced down to single bit multiplication and addition by using Urdhva Tiryakbhyam sutra is performed by vertically and crosswise. The partial products and sums are generated in single step which reduces the carry propagation from LSB to MB by using these formulas [4].

This paper presented a design of efficient complex number multiplier using the Veda Sutra " UrdhvaTiryakbhyam" from ancient Indian vedic mathematic .The fundamental and core of all digital signal processor DSP's are its multiplier and the speed of DSP's is determined by the speed of its multiplier, the "UrdhvaTiryakbhyam" method has been selected for implementation. Multiplication using this sutra is performed vertically or crosswise.The partial products and sums are generated in one step, which reduces the carry propagation from LSB to MSB [6].

This paper proposed the work deals with the design and implementation of complex multiplier / mixers using Field Programmable Gate Array (FPGA) chip with low cost and high speed . Here two devices of FPGA are chosen to implement the design for achieving the task of mixer system implementation. To achieve high speed data, a parallel two multiplier is user with Wallace Tree method [10].

This paper proposed the design of 8-bit vedic multiplier using the techniques of Ancient Indian Vedic mathematics that have been modified to improve the performance. The work in this paper has proved the efficiency of UrdhvaTiryakbhyam Vedic method for multiplication which strikes a difference in the actual process of multiplication itself [11].

3. PROPOSED WORK

The Proposed work is to design a high speed 32-bit IEEE 754 Floating Point Complex Multiplier using Booth algorithm. Complex multiplier can be composed with four real multipliers, one adder and one sub-tractor. The complex number always consists of two terms one is real and another is imaginary. For the complex multiplier these two terms are very necessary. This block diagram mainly consists of four floating point Booth algorithm. Following figures shows a proposed block diagram the implementation of 32 bit floating point complex multiplier.

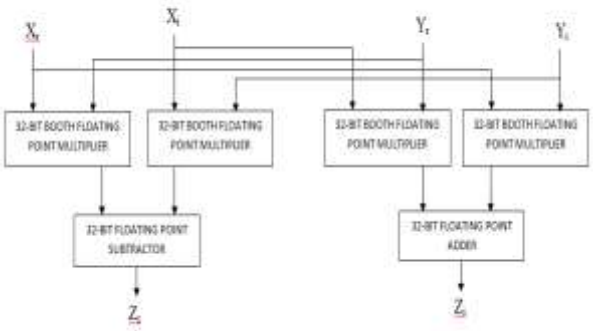

Fig 1.2 : Proposed Block Diagram of 32 Bit Floating Point Complex Multiplier.

MODULE 1 : FLOATING POINT ADDER (FPA)

The following flowchart shows the operation of FPA

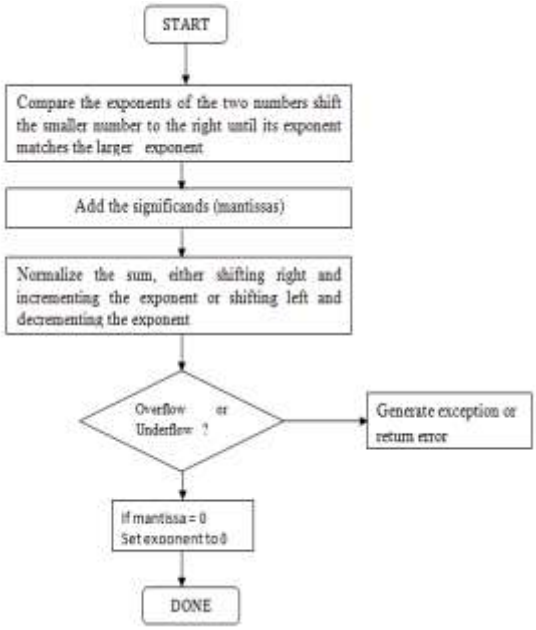

Fig 1.3 : Flowchart of floating point adder

Performing Floating Point Addition Result = X + Y = (Xm * 2^Xe) + (Ym * 2^Ye) involves the following steps :

1) Align binary point :
   - Initial result exponent : the larger of Xe, Ye
   - Compute exponent difference : Ye – Xe
   - If Ye > Xe Right shift Xm that many positions to form Xm 2 ^Xe -Ye

- If Xe > Ye Right shift Ym that many positions to form Ym 2^Ye – Xe

2) Compute sum of aligned mantissas :
i.e Xm*2^Xe-Ye + Ym or Xm+Xm*2^Ye-Xe

3) If normalization of result is needed, then a normalization steps follows :
   - Left shift result, decrement result exponent (eg. if result is 0.001xx..) or
   - Right shift result, increment result exponent (eg. if result is 10.1xx..)
   Continue until MSB of data is 1

4) Check result exponent :
   - If larger than maximum exponent allowed return exponent overflow
   - If smaller than minimum exponent allowed return exponent underflow
5) If result mantissa is 0, may need to set the exponent to zero by a special step to return a proper zero.

Example :

X=2345.125 =100100101001.001 represented as:

Y=0.75 =0.11 represented as:

Xe > Ye initial result exponent = Ye = 10001010 = 138base10

Xe - Ye = 10001010 - 01111110 = 00000110 = 12base 10

Shift Ym 12base10 postions to the right to form Ym = 0.000000000000110000000000

Xm + Ym = 1.00100101001001000000000 + 0.000000000000110000000000
=1. 00100101001111000000000

Result is

0	10001010	00100101001111000000000

- If the exponents differ by more than 24,the smaller number will be shifted right entirely out of the mantissa field,producing a zero mantissa.

  -The sum will then equal the larger number.

  -Such truncation errors occur when the numbers differ by a factor of more than 2^24,which is approximately 1.6*10^7.

  -Thus,the precision of IEEE single precision floating point arithmetic is approximately 7 decimal digits.

- Negative mantissa are handled by first converting to 2`s complement and then performing the addition.

  -After the addition is performed,the result is converted back to sign-magnitude form.

- When adding numbers of opposite sign,cancelltion may occur,resulting in a sum which is arbitrarily small,or even zero if the numbers are equal in magnitude.

  -Normalization in this case may require shifting bye the total number of bits in the mantissa,resulting in a large loss of accuracy.

0	10001010	00100101001001000000000

0	01111110	10000000000000000000000

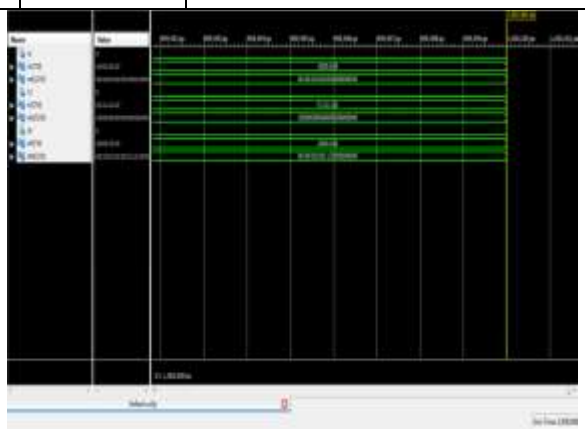

Fig 1.4 : Simulation results of floating point adder

4. CONCLUSION AND FUTURE SCOPE

The multiplier is designed for a less delay and less power consumption. Hence, a 32-bit IEEE 754 Complex Multiplier with high speed will be the probable outcome of this work.Also we can design and simulate this project in 64-bit using double precision floating point complex multiplier. But it's complexity is more.This project can be implemented on FPGA / CPLD kit.

REFERENCES

1. L P. Thakare , Dr. A Y. Deshmukh , "*Area Efficient Complex Floating Point Multiplier For Reconfigurable FFT/IFFT Processor Based On Vedic Algorithm* ", Science Direct / 7th International Conference on Communication, Computing and Virtualization 2016.

2. Rizalafande CheIsmail ,Razaidi Hussin , "*High Performance Complex Number Multiplier Using Booth-Wallace Algorithm* ", ICSE2006 Proc. 2006, Kuala Lumpur, Malaysia.

3. Prabir Saha , Arindam Banerjee , Partha Bhattacharyya , Anup Dandapat , *" High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics " ,* Proceeding of the 2011 IEEE Students' Technology Symposium 14-16 January, 2011, lITKharagpur .

4. Rajashri K. Bhongade, Sharada G .Mungale, Karuna Bogawar , " *Vhdl Implementation and Comparison of Complex Mul-tiplier Using Booth's and Vedic Algorithm* " , COMPUSOFT, An international journal of advanced computer technology, 3 (3), March-2014 (Volume-III, Issue-III) .

5. Rajashri Bhongade ,S.G.Mungale , Karuna Bogavar , "*Performance Evaluation of High Speed Complex Multiplier Using Vedic Mathematics* ",International Journal of Innovative Research in Advanced Engineering (IJIRAE)Volume 1 Issue 1 (April 2014).

6. Laxman P. Thakare , A. Y. Deshmukh , Gopichand D. Khandale , "*VHDL Implementation of ComplexNumber Multiplier Using Vedic Mathematics* ", Springer 2014.

7. Gopichand D. Khandale , Laxman P. Thakare , Dr. A. Y. Deshmukh , "*Performance Evaluation of Complex Multiplier Using Advance Algorithm ",* International Journal of Electronics and Computer Science Engineering 1018 Available Online at www.ijecse.org ISSN- 2277-1956.

8. Ankush Nikam, Swati Salunke, Sweta Bhurse , "*Design and Implementation of 32bit Complex Multiplier using Vedic Algorithm",* International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 4 Issue 03, March-2015 644.

9. Man Yan (Raymond) Kong , J.M. Pierre Langlois , Dhamin Al-Khalili , "*Efficient FPGA Implementation of Complex Multipliers using the Logarithmic Number System* ",2008 IEEE.

10. Ali Mohammed Hassan Al-Bermani , Raya Kahtan Mohamed , "*Design And Implementation Of High Speed Complex Multiplier Using Fpga***",** The 1st Regional Conference of Eng. Sci. NUCEJ Spatial ISSUE vol.11,No.1, 2008 pp91-97.

11. Swaroop A. Gandewar , Mamta Sarde , "*Design Of Vedic Multiplier For Complex Numbers For Enhanced Computation Using Vhdl*"**,** International Journal of Industrial Electronics and Electrical Engineering, ISSN: 2347-6982 Volume-2, Issue-5, May-2014.