Area Efficient Complex Floating Point Multiplier for Reconfigurable FFT/IFFT Processor Based on Vedic Algorithm

L.P. Thakare\textsuperscript{a}  Dr. A.Y. Deshmukh\textsuperscript{b}

\textsuperscript{a}Research Scholar, Department of Electronics Engineering, G.H. Raisoni College of Engineering, Nagpur-440016, INDIA, laxman.thakare@raisoni.net

\textsuperscript{b}Professor, Department of Electronics Engineering, G.H. Raisoni College of Engineering, Nagpur-440016, INDIA, amol.deshmukh@raisoni.net

Abstract

The need of complex multipliers in mathematics seems to be a very important aspect. The application used as a complex operation with respect to the real and imaginary numbers together. In many of the practical aspect the design substantiated to be a aiding hand for co-functionality units. Since to carry out the operations more correctly so as to ellicit the technical peccadilloes. The results show that proposed FFT consumes very less resources in terms of slices, flip flops and multipliers to provide cost effective solution for DSP applications The proposed design has a desideratum for high efficacy eliciting structure in terms of accuracy. The device used in this proposed design is 7a30tcs324-3.

Keywords: FFT, IFFT, Complex Multipliers

1. INTRODUCTION

The Complex multiplier is a leaf out of a tree of complex algebra. It has its own utilization within various domains which can be performs using the mathematical operations from core tenets. It can be used in a Fast Fourier Transform to carry out the reckon of discrete Fourier transform. But it should have the advantage over re configurability. When it is tattled to solve such problems then the design should be like that so as to intend it for distinct combinations of possible numbers within desired range. But when the complex multiplication of two floating numbers is there then the IEEE formats are essentially be the cynosure of designers. Array multiplier, Booth Multiplier are some of the standard design issues which approaches used in implementation of binary multiplier which are well suitable for VLSI implementation. It proposed that the module used has inputs of 32-bit width. Before the operation Special cases are treated separately without Adder and some other resources. In addition to normal and subnormal numbers, in \textsuperscript{1}. It proposed that floating-point multiply operation has several different parts. The first part is multiply the significant of two given numbers using ordinary integer multiplication. Because floating point numbers are stored in sign magnitude form. In \textsuperscript{2}. It proposed the algorithms for sticky bit generation algorithms for the design used. The advantages &
disadvantages of different methods have also discussed for more accuracy. A method is provided to detect the "sticky bit" when generated as a result of an alignment operation. The detection of the sticky bit is done in parallel with the alignment operation and 35 uses parallel two ways propagation of group enables. The Mathematician Argand represented a complex number in a diagram known as Argand's diagram. A complex number $x + jy$ can be represented by a point P whose coordinates are (x, y). The axis of x is called the real axis and the axis of y is called the imaginary axis. The distance OP is the Modulus and the angle, OP makes with the x-axis, is the argument of $x + jy$.

![Argand diagram](image)

**Fig1. FLOATING MULTIPLIER**

Basically the above delineated circuit is used for multiplication of two floating point numbers. There is no definite logic level for representation of decimal point in digital circuit. So it is herculean to store the decimal point into the storing elements like flip flops, registers, memories etc in true form. So we ought to cogitate that how we can store the floating number. So we have a IEEE formats for different ranges like single precision, double precision, quad precision etc. In this paper single precision format is preferred.

The structure of multiplexer has very less usage in earlier papers. The exponent unit, Vedic multiplier unit & the sign generation logic elicits there results simultaneously. The result elicits by the Vedic multiplier has a nexus with the exponent unit. The status of MSB bit decides the selection of the one input signal out of two inputs of multiplexer shown above.

The sign bit will help the resultant bit generation by xoring the two signs of input numbers. Same sign will give positive form of result, whereas different sign numbers will give negative form of a result.

**2. Vedic mathematics**

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It covers explanation of several modern mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus. His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamhiji constructed 16 sutras (formulae) and 16 Up a sutras (sub formulae) after extensive research in Atharva Veda. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing.
3. IMPLEMENTATION OF MULTIPLIER USING VM ALGORITHM

Multiplication methods are extensively discussed in Vedic mathematics. Various tricks and short cuts are suggested by VM to optimize the process. These methods are based on concept of
1) Multiplication using deficits and excess
2) Changing the base to simplify the operation.

Various methods of multiplication proposed in VM
a) UrdhvaTiryagbhyam - vertically and crosswise
b) Nikhilam navatashcharamam Dashatah: All from nine and last from ten
c) Anurupyena: Proportionately
d) Vinculum

3.1 URDHVA TIRYAGBHYAM

Urdhva – Triyagbhyam is the general formula applicable to all cases of multiplication and also in the division of a large number by another large number. It means vertically and crosswise.

3.1.1 FLOATING ADDER

It is well known that during addition of two floating numbers, the numbers should be in single precision format. In order to perform proper reckoning of addition of two numbers some steps must have to be imitated which has been elucidated below. Before outset any reckoning, first compare the exponents of two numbers which have to be added. Once we have finished with comparison then we may come across two conditions. Either the comparison answer will be obtained as zero or nonzero. If the answer is zero then it implies that the exponents were equal. In this case no question of shifting of mantissa of smaller number will arise. But if the answer is nonzero then connotation is that one of the exponents out of two given numbers exponents is smaller than other one. Then complete the exponents. Shift the mantissa of number towards Dexter side whose exponent is smaller. Shift should be like a logical shift. Number of Shifting of bits must be equals to the comparison answer obtained. Do not shift larger number, keep it as it is. Perform the addition of un-shifted mantissa of larger number & shifted mantissa of smaller number. Check whether
the obtained result in step 4 is normalized or not. If normalized which means ‘1,’ form then do not shift decimal point & do not vary the exponent, the resultant exponent will be the exponent of larger number & resultant mantissa will be the first 23 bits of answer of mantissa addition just after decimal point of MSB bit. If un-normalized then shifting will become necessary. Shift the decimal point either anti- Dexter side & increment the exponent of larger number or Shift the decimal point Dexter side & decrement the exponent of larger number. In this case the resultant exponent will be the varied exponent and the resultant mantissa will be the first 23 bits of answer of mantissa addition just after decimal point which had shifted earlier in order to get normalized form of mantissa addition answer. In case of equal exponents of given numbers. The resultant exponent might be cull as exponent of any one number out of two, if the result of mantissa addition is normalized.

EX- $6815.46 + 3112.18 = 9927.64$

A=$6815.46=01000101110101001111101110101110$ (IEEE 32 bit format)
B=$3112.18=01000101010000101000001011100001$ (IEEE 32 bit format)

Ma= 1.10101001111101110101110 (mantissa of number A)
Mb= 1.10000101000001011100001 (mantissa of number B)

| Ma | 1.10101001111101110101110 |
| Mb | + 0.11000010100000101110000 |
|    | 10.0110110001111010001111 |
| Result | 0.10001100 0011011000111010001111 |

3.1.2 FLOATING SUBTRACTOR

It is well known that during subtraction of two floating numbers, the numbers should be in single precision format. In order to perform proper reckoning of subtraction of two numbers some steps must have to be imitated which has been elucidated below.

Before outset any reckoning, first compare the exponents of two numbers which have to be subtracted. Once we have finished with comparison then we may come across two conditions. Either the comparison answer will
be obtained as zero or nonzero. If the answer is zero then it implies that the exponents were equal. In this case no question of shifting of mantissa of smaller number will arise. But if the answer is nonzero then connotation is that one of the exponents out of two given numbers exponents is smaller than other one. Then complete the exponents.

Shift the mantissa of number towards Dexter side whose exponent is smaller. Shift should be like a logical shift. Number of Shifting of bits must be equal to the comparison answer obtained. Do not shift larger number, keep it as it is. Perform the subtraction of un-shifted mantissa of larger number & shifted mantissa of smaller number. Check whether the obtained result in step 4 is normalized or not. If normalized which means ‘1.’ form then do not shift decimal point & do not vary the exponent, the resultant exponent will be the exponent of larger number & resultant mantissa will be the first 23 bits of answer of mantissa subtraction just after decimal point of MSB bit. If un-normalized then shifting will become necessary. Shift the decimal point either anti-Dexter side & increment the exponent of larger number or Shift the decimal point Dexter side & decrement the exponent of larger number. In this case the resultant exponent will be the varied exponent and the resultant mantissa will be the first 23 bits of answer of mantissa subtraction just after decimal point which had shifted earlier in order to get normalized form of mantissa subtraction answer. In case of equal exponents of given numbers. The resultant exponent might be cull as exponent of any one number out of two, if the result of mantissa subtraction is normalized.

EX: - 0.1814 – 9419.81 = -9419.6286
A= 0.1814 = 0 01111100 01110011100000011101011 (IEEE 32 bit format)
B= -9419.81= 1 10001100 00100110010111100111101 (IEEE 32 bit format)
Ma= 1. 01110011100000011101011 (mantissa of number A)
Mb= 1. 00100110010111100111101 (mantissa of number B)
Mr= 1.00100110010111010000100
Er= Eb =10001100
Result= 1 10001100 00100110011011000100

3.1.4 Complex Floating Multiplier

Let a, b, c, d be the floating numbers.
In case of complex multiplication the above terms can be arrange as
(a + jb) and (c + jd)
Now the equation of complex multiplier can be written as
= (a + jb) (c + jd)
= ac + j(ad) + j(bc) – bd
= (ac – bd) + j(ad + bc)
Floating Subtractor Floating Adder

![Complex Floating Multiplier](image)
Table No 1. Synthesis Report Values

<table>
<thead>
<tr>
<th>Parameters of Complex Floating Multiplier</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO Buffers</td>
<td>192</td>
</tr>
<tr>
<td>Combinational path delay</td>
<td>34.245ns</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>59.91 MB</td>
</tr>
<tr>
<td>Number of slice LUT’s</td>
<td>21000 out of 48424</td>
</tr>
<tr>
<td>1-bit Xor gates</td>
<td>43518</td>
</tr>
<tr>
<td>Global maximum fan outs</td>
<td>100000</td>
</tr>
</tbody>
</table>

\[
(0.85 - j 15.30) (-41.6 - j 69.2) = -1094.12 + j 577.66
\]

4. CONCLUSION

The designed Complex floating point multiplier using Vedic sutra multiplier can be work for any floating point numbers within range of single precision format. The combine adder subtractor unit can save the efforts of reckoning compared to two different modules. Hardware part utilization is 43% so designed multiplier is area efficient. The device used in this proposed design is 7a30tcs324-3. The separate modules can be used but depending on the usage of an application.

REFERENCES