Implementation of area-efficient radix-4 complex Number division

Arya B Mohanan¹ and Nimmy M Philip ²

¹PG Scholar, Department of Electronics and Communication Engineering, Federal Institute of Science & Technology, Angamali, India. Email: aryabmohanan@gmail.com

²Assistant Professor, Department of Electronics and Communication Engineering, Federal Institute of Science & Technology, Angamali, India. Email: nimmyphilip@fisat.ac.in

Abstract: Complex division is a critical mathematical operation in signal processing, control systems, microwave systems etc... Division of complex numbers is a necessary evil in DSP systems also. So it demands reliability and accuracy. We present a design and implementation of a radix-4 complex division unit with error detection using hardware redundancy technique. Conventional complex division method consumes more area and have higher critical path delay. In this paper an area-efficient division algorithm with concurrent error detection (CED) is proposed. We present schemes to provide complex number division architecture based on (Sweeney, Robertson and Tocher) SRT-division with error checking. Assessments are performed for the proposed design. The design is synthesized using Xilinx ISE 14.2.

Keywords -. Concurrent error detection (CED), SRT-division, Robertson diagram, Radix-4 arithmetic, Hardware redundancy, FPGA

1. Introduction

Complex division is commonly used in various applications in signal processing [1] and control theory. Radix-4 complex division is a critical mathematical operation. Complex division arithmetic is also used in control theory in order to find out Root locus [2]. Microwave sub systems use complex division arithmetic to find the frequency response[3] A technique for high radix complex division has been proposed in earlier and that approach is based on operand prescaling and digit recurrence [4]. This design uses a slightly modified radix-4 (Sweeney, Robertson, and Tocher) SRT division to calculate the quotient and the remainder of the complex division.

A method for division of complex numbers in higher radix is proposer in [5].The existing methods for complex division in higher radix are based on operand prescaling and digit recurrence. Another relevant method for complex division is based on coordinate Rotational Digit Computer (CORDIC) algorithm [6]. For the purpose of improving the performance of division algorithm an efficient and error free complex conjugate multiplication is needed.

The design of fast dividers is an important issue in high speed computing because division accounts for a significant fraction of the total arithmetic operation. Most implementations for the division are based on the SRT algorithm that uses a recurrence producing one quotient digit for each step. The speed of such SRT based dividers is mainly determined by the complexity of the quotient digit selection. The use of a quotient selection table (QST) significantly reduces the complexity of quotient section. However the table size increases drastically with high radices. The table size can be reduced significantly by estimating the quotient digit instead of finding the exact one.
For a complex number there will be a real part and an imaginary part. It have easiness in
calculation the real and imaginary parts can be handled separately. In complex conjugate
multiplication part both numerator and denominator are multiplied by the conjugate. The product
obtained by multiplying numerator and conjugate will also have two parts, a real part and an imaginary
part. But the product obtained by multiplying denominator and complex conjugate will have the real
part only. This real value is treated as the divisor in final division. Both the real and imaginary parts
obtained from the product of numerator and conjugate are treated as dividend.

A Number system with the complex number \(-1+i\) as the base is developed. This permits the
representation in binary form of any complex number \(a+ib\). Operations with complex number are
usually performed by dealing with the real and imaginary parts separately and combining the two as a
final step. It might be an advantage in some problems to treat a complex number as a unit and carry out
all operations in this form. The number system to be described permits the representation of a complex
number as a single binary number to a degree of accuracy limited only by capacity of computer. It is
binary in that only the two symbols ‘0’ and ‘1’ are used; however the base is not 2, but the complex
number \(-1+i\). The quantity \(-1-1\) would be equally suitable and in fact, for real numbers it is immaterial
which of these two we consider the base. A simple program would allow arithmetic operations to be
performed in this mode.

In digital systems, errors can happen through various causes including alpha particles from
package decay, cosmic rays creating energetic neutrons and protons, and thermal neutrons.
Counteracting natural faults has been a subject for a number of hardware architectures in different
domains. In signal processing for instance, and for cryptographic architectures, many research works
have been carried out to achieve reliable and fault-immune structures. Moreover, concerning the finite
field arithmetic architectures, various concurrent error detection (CED) multipliers for polynomial
basis and normal basis of GF(2\(^m\)) have been proposed using parity codes and recomputing with shifted
operands (RESO) schemes [7].

Various real number and complex number division methods are there, in the proposed method
SRT division algorithm is used because it provides efficiency in terms of area compared to
conventional division methods. The division results quotient and remainder in radix-4 format. The
dividend and divisor should be converted to corresponding radix-4 equivalent number. The decimal to
radix-4 conversion can be achieved by successive division by 4 (as we done in decimal to binary
conversion using two). The contributions of this paper are

- We propose error detection approach for the complex conjugate multiplication part using CED
technique. The properties of radix-4 complex division is compared with the properties conventional
complex division.
- The design is simulated using ModelSim Pe 5.5, synthesized using Xilinx ISE 14.2.
- Finally through simulations we benchmark the benefits of proposed division scheme. The result
of these simulations show acceptable efficiency in terms of area which ensures reliability and hardware
assurance for the proposed approaches.

This paper is organized as follows. Section 2 discusses the proposed design regarding complex
conjugate multiplication and complex division using SRT division algorithm. Section 3 explains the
results and simulations and the comparison between proposed method and conventional complex
division method.

2. Conventional complex division
A straightforward way to implement complex division is to use conventional division formula. This
formula may lead to overflows during intermediate computations.
\[
\frac{A+iB}{C+iD} = \frac{(AC + BD + i(BC-AD))}{C^2 + D^2}
\]
3. Proposed method

SRT dividers are common in modern floating point units. Higher division performance is achieved by retiring more quotient bits in each cycle. Previous research has shown that realistic stages are limited to radix-2 and radix-4. Higher radix dividers are therefore formed by a combination of low-radix stages. In this paper, we present an analysis of the effects of radix-2 and radix-4 SRT divider architectures and circuit families on divider area and performance. We show the performance and area results for a wide variety of divider architectures and implementations. We conclude that divider performance is only weakly sensitive to reasonable choices of architecture but significantly improved by aggressive circuit techniques. A simple and widely implemented class of division algorithm is digit recurrence. The most common implementation of digit recurrence division in modern microprocessors is SRT division, taking its name from the initials of Sweeney, Robertson and every iteration.

3.1 Complex conjugate multiplication

Complex multiplications in DSP systems generally use a more efficient indirect approach, referred to as the Golub’s method. For two complex numbers with \( y = a + bj \) and \( z = c + dj \), with direct implementation one can reach the following results:

\[
x = y \cdot z = (a + bj)(c + dj)
\]  
\[
ac + j(ad) + j(bc) + j^2(bd)
\]  
\[
ac - bd + j(ad + bc)
\]  
\[
x_{\text{real}} = ac - bd
\]  
\[
x_{\text{imag}} = ad + bc
\]

The direct implementation of complex multiplication requires four multiplications and two additions. The indirect multiplication on the other hand is as follows:

\[
y = a + bj \quad z = c + dj
\]  
\[
x = y \cdot z = (t_2 - t_3) + j(t_1 - t_2 - t_3)
\]  
\[
x_{\text{real}} = t_2 + t_3 \quad x_{\text{imag}} = t_1 + t_2 - t_3
\]  
\[
t_1 = (a + b) \cdot (c + d) \quad t_2 = a \cdot c \quad t_3 = b \cdot d
\]

Indirect implementation on the other hand requires three multiplications and five additions. The indirect way is more area efficient because multiplication requires more area compared to additions. This indirect method is known as Golub’s method for complex multiplication.
To implement an effective error detection method the propagation of faults throughout the circuit need to be analyzed. For error detection a checker 3 method (low complexity concurrent error detection for complex multiplication) is used. This was found to be the most efficient in terms of area compared to other techniques for adders and multipliers. This separates the calculation of real and imaginary parts and in turn prevents error propagation form \((a*c)+(b*d)\) to \(x_{\text{imag}}\).

\[
X_{\text{real}} \, + X_{\text{img}} = t_2 - t_3 + t_1 - t_2 - t_3 = t_1 - 2t_3 = t_1 - 2b*d
\]  
\[
X_{\text{real}} \, + X_{\text{img}} + 2b*d = t_1
\]

This module performs three tasks

- Multiplication
- Normalize the divisor (denominator part)
- Real and imaginary part separation

### 3.2 Radix-4 SRT division module

The design of the complex divider consists of several modules: multiplication module, normalizing module, real and imaginary iteration module, shared ROM for the quotient Selection. Multiplication in the numerator is performed using Golub’s multiplication method. The multiplication module multiplies the complex conjugate of the denominator to the numerator and denominator to rationalize the denominator to a real number. This is an iterative algorithm produces fixed number of quotient bits per step.

Each step consists of subtracting a multiple of the divisor from the product of radix and remainder after previous iteration. The quotient is accumulated in each iteration. The quotient digit decision is made as a part of each iteration. The required quotient digits for each iteration is fetched from a previously stored location. The proposed design uses a radix-4 SRT division technique. Thus, the division scheme for radix-4 is \(R[j + 1] = 4 \times (R[j] - q[j] \times D)\) with two quotient sets; \({-3, -2, -1, 0, 1, 2, 3}\) is called the maximally redundant set and \({-2, -1, 0, 1, 2}\) is referred to as the minimally redundant set. The maximally redundant set requires less address bits to determine the quotient; thus, it requires a smaller LUT compared to the minimally redundant; nonetheless, it requires the computation of \(3^*\) which leads to additional hardware and delay. We would like to emphasize that the quotient \(q[j]\) is obtained once every iteration; this is based on the quotient digit set which is determined for a particular radix based on the constant \(h\). In turn, \(h\) can also be used to reduce the size of the quotient set \({-3, -2, -1, 0, 1, 2, 3}\) or \({-2, -1, 0, 1, 2}\) based on the requirements (speed vs. precision), this determines the precision and speed of the implemented hardware.
4. Results and observations

Complex conjugate multiplication for \((a+ib) \times (c-id)\) format

| /gobmul/a | 48 | 28 |
| /gobmul/bi | 8 | 8 |
| /gobmul/c | 6 | 6 |
| /gobmul/di | 4 | 4 |
| /gobmul/ar | 320 | 320 |
| /gobmul/si | -144 | 114 |
| /gobmul/w1 | 288 | 288 |
| /gobmul/w2 | -32 | -32 |
| /gobmul/w3 | 256 | 256 |
| /gobmul/w5 | 112 | 112 |
| /gobmul/w6 | 55 | 55 |
| /gobmul/w7 | 2 | 2 |
| /gobmul/c0 | 0 | 0 |
| /gobmul/c1 | 0 | 0 |

Fig. 3 Simulation result for conjugate multiplication

Complex conjugate multiplication for \((c+id) \times (c+id)\) format, This result give the rationalized denominator in real number format.
Complex division is done using SRT algorithm with quotient selection

Complex division using conventional formula

![Simulation result for rationalized denominator](image1)

![Simulation result for Complex division using SRT division algorithm](image2)

![Simulation result for complex division using conventional formula](image3)
TABLE 1 comparison using Xilinx ISE 14.2

<table>
<thead>
<tr>
<th></th>
<th>Radix-4 SRT complex division</th>
<th>Conventional complex division</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of 4 input LUTs used</td>
<td>542</td>
<td>586</td>
</tr>
<tr>
<td>Number of 4 input LUTs available</td>
<td>9312</td>
<td>1920</td>
</tr>
<tr>
<td>Utilization</td>
<td>5%</td>
<td>30%</td>
</tr>
<tr>
<td>Critical path delay</td>
<td>30.385ns</td>
<td>39.560ns</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper we have presented two complex division methods. The division scheme uses SRT algorithm and conventional division formula. The results obtained shows that SRT method is more area efficient and has low critical path delay. The conjugate multiplication in the proposed division has error detection scheme which ensures the accuracy of the division module.

REFERENCES