On Lifting-Based Fixed-Point Complex Multiplications and Rotations

Oscar Gustafsson

Conference article

Cite this conference article as:


DOI: https://doi.org/10.1109/ARITH.2017.10

Proceedings Symposium on Computer Arithmetic, 1063-6889, No. 2017

Copyright: IEEE

The self-archived postprint version of this conference article is available at Linköping University Institutional Repository (DiVA):
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-139336
On Lifting-Based Fixed-Point Complex Multiplications and Rotations

Oscar Gustafsson

Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden
Email: oscar.gustafsson@liu.se

Abstract—Lifting-based complex multiplications and rotations are integer invertible, i.e., an integer input value is mapped to the same integer output value when rotating forward and backward. This is an important aspect for lossless transform-based source coding, but since the structure only require three real-valued multiplications and three real-valued additions it is also a potentially attractive way to perform complex multiplications when the coefficient has unity magnitude. In this work, we consider two aspects of these structures. First, we show that both the magnitude and angular error is dependent on the angle of input value and derive both exact and approximated expressions for these. Second, we discuss how to design such structures without the typical separation into three subsequent matrix multiplications. It is shown that the proposed design method allows many more values which are integer invertible, but can not be separated into three subsequent matrix multiplications with fixed-point values. The results show good correspondence between the error approximations and the actual error as well as a significantly increased design space.

1. Introduction

Complex-valued multiplications are standard operation in many different computational algorithms. A complex-valued multiplication by two values $a + jb$ and $c + jd$, where $j = \sqrt{-1}$, to form a result $e + jf$ results in the computations

$$e + jf = (a + jb)(c + jd) = ac - bd + j(ad + bc),$$

or on matrix form

$$\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} c & -d \\ d & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Hence, four real-valued multiplications and two real-valued additions are required to perform one complex-valued multiplication.

To reduce the number of multiplications, a number of different approaches has been proposed [1], where the basic approach is often referred to as Gauss multiplication [2]. Using these approaches, three real-valued multiplications are enough at the expense of five real-valued additions, which can be reduced to three real-valued additions if the sum and difference of the real and imaginary values of one of the terms is pre-computed and stored. For example, in (1), $c + d$ and $c - d$. More advanced approaches have also been proposed [3], [4].

Now, consider the rotation of a complex value by an angle $\theta$ rad. Here, we multiply with

$$e^{j\theta} = \cos(\theta) + j \sin(\theta) = c + jd.$$  

This type of rotations are imminent in certain DSP algorithms, the complex form in e.g. the discrete Fourier transform, and real-valued rotations as in (2) in e.g. the discrete Cosine transform (DCT).

Let us denote the binary fixed-point approximation of a continuous variable $x$ as $\hat{x}$. Then, the rotation in (3) it is expected that the magnitude in equal to one, i.e.,

$$\hat{c}^2 + \hat{d}^2 = 1.$$  

Assuming $B$ fractional bits and multiplying both sides of the equation with $2^{2B}$, we get

$$(2^B \hat{c})^2 + (2^B \hat{d})^2 = 2^{2B}$$

leading to that $2^B \hat{c}$ and $2^B \hat{d}$ are integers. Hence, for (4) to hold, there must exist a Pythagorean triple, $x^2 + y^2 = z^2$ for integer $x$, $y$, and $z$, where $z = 2^B$. However, it can be shown that $z$ must be on the form $4N + 1$ for integer $N$ and therefore there are no values $\hat{c}$ and $\hat{d}$ in any binary fixed-point representation where (4) holds unless one of $c$ and $d$ is 0, and, hence, the other term is 1.

One consequence of this is that it is not possible to find a set of coefficients that exactly rotates back with the same angle. Let us denote the fixed-point approximation of a continuous variable $x$ as $\hat{x}$ and a rotation matrix with fixed-point coefficients

$$\mathbf{R} = \begin{bmatrix} \hat{c} & -\hat{d} \\ \hat{d} & \hat{c} \end{bmatrix}.$$  

1. Strictly speaking it is not possible in a binary floating-point format either. However, since the lifting-based structure is primarily of interest in integer/fixed-point arithmetic, we limit the discussion to this case.
The inverse is then equal to
\[
R^{-1} = \frac{1}{\det R} \begin{bmatrix} \hat{c} & \hat{d} \\ -\hat{d} & \hat{c} \end{bmatrix} = \frac{1}{\hat{c}^2 + \hat{d}^2} \begin{bmatrix} \hat{c} & \hat{d} \\ -\hat{d} & \hat{c} \end{bmatrix}. \tag{7}
\]

Ignoring that \(\hat{c}^2 + \hat{d}^2 \neq 1\) and assigning
\[
R^{-1} = \begin{bmatrix} \hat{c} & \hat{d} \\ -\hat{d} & \hat{c} \end{bmatrix} \tag{8}
\]
results in a forward-backward rotation as
\[
RR^{-1} = \begin{bmatrix} \hat{c} & -\hat{d} \\ \hat{d} & \hat{c} \end{bmatrix} \begin{bmatrix} \hat{c} & \hat{d} \\ -\hat{d} & \hat{c} \end{bmatrix} = \begin{bmatrix} \hat{c}^2 + \hat{d}^2 & \hat{c}\hat{d} - \hat{d}\hat{c} \\ \hat{d}\hat{c} - \hat{c}\hat{d} & \hat{c}^2 + \hat{d}^2 \end{bmatrix} = \begin{bmatrix} \hat{c}^2 + \hat{d}^2 & 0 \\ 0 & \hat{c}^2 + \hat{d}^2 \end{bmatrix}. \tag{9}
\]

Hence, the consequence is that there is a magnitude error introduced from each forward backward transform. This in turn means that a set of integer values at the input will not map to the same integer values at the output after running a forward and a backward transform. This is a problem in e.g., transformed based source coding [5], [6], where it is expected that the coded data can be exactly restored to provide losslessness.

One proposed solution to this is to use so-called lifting-based complex multipliers [5], [7], [8], [9]. Here, the original rotation in (2) is split into three subsequent matrix multiplications. For example, as
\[
R = \begin{bmatrix} 1 & \frac{c-1}{d} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{c-1}{d} \\ 0 & 1 \end{bmatrix} \tag{12}
\]
\[
= \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix}, \tag{13}
\]
where \(g = \frac{c-1}{d} \). Now,
\[
R^{-1} = \left( \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 1 & -g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{14}
\]

By doing this, independent of coefficient quantization errors, representation, etc of \(d\) and \(g = \frac{c-1}{d}\), applying the forward and backward rotation will always result in the identity matrix as (note that nothing is assumed about \(d\) and \(g\) here)
\[
RR^{-1} = \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 + \hat{d}g & \hat{g}(2 + \hat{d}g) \\ \hat{g} & 1 + \hat{d}g \end{bmatrix}. \tag{15}
\]

It should be noted that these three matrix multiplications only require three real-valued multiplications and three real-valued additions in total, as is clear from Fig. 1. Furthermore, the complex multiplication can be realized using three multiply-add operations, which may be beneficial when implemented on a fixed-point processor [7]. There are four different variants of lifting-based decomposition [8], where one is shown in (12). All can be written as in (13) and realized as in Fig. 1, although the value of \(g\) differs and what is now \(d\), is one of \(\pm d, \pm c\). Depending on the angle of rotation, a structure can be selected to yield
\[
|g| \leq \frac{1 - \cos \left( \frac{\pi}{8} \right)}{\sin \left( \frac{\pi}{8} \right)} \approx 0.414
\]
and low round-off noise [9]. The rest of the analysis is without loss of generality based on (12) and (13).

Some related error analysis was presented in [10], where error approximations for DCTs implemented using lifting-based rotators are derived.

In this work we consider two aspects of this. First, in the next section, we show that, although the above holds, both the magnitude and the angle of rotation is input signal dependent. This is opposed to the rotation in (6) which has a constant magnitude of \(\sqrt{\hat{c}^2 + \hat{d}^2}\) and an angle of rotation equal to \(\arctan \left( \frac{\hat{d}}{\hat{c}} \right)\). Second, in Section 3, we discuss how rotations with the same property, but not necessarily possible to separate in three matrices representable in binary fixed-point representation, can be designed. Both the magnitude and angle analysis as well as the design method can be directly applied to any of the four lifting-based decompositions. In Section 4, we provide some numerical results from the previous discussion. Finally, some concluding remarks are given.

2. Numerical Errors

When evaluating the expression in (13) with fixed-point coefficients, we obtain
\[
R = \begin{bmatrix} 1 & \hat{g} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \hat{d} & 1 \end{bmatrix} \begin{bmatrix} 1 & \hat{g} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 + \hat{d}\hat{g} & \hat{g}(2 + \hat{d}\hat{g}) \\ \hat{g} & 1 + \hat{d}\hat{g} \end{bmatrix}. \tag{17}
\]

2. Without loss of generality, we assume \(\hat{c}\) and \(\hat{d}\), as well as where applicable later introduced variables, to be non-negative when determining the angles of the rotations. This leads to that all rotation angles \(\theta\) are in the range \(0 \leq \theta \leq 90^\circ\). However, the signs of the rotation matrix coefficients can be freely selected to end up with a rotation in the required quadrant and the related \(\arctan\) functions can straightforwardly be adapted to the correct sign when required.
Clearly, inserting \(\hat{g} = \frac{c - 1}{d}\) we obtain
\[
1 + \hat{d}\hat{g} = 1 + \hat{d}\frac{c - 1}{d} = c \tag{18}
\]
and
\[
\hat{g} \left(2 + \hat{d}\hat{g}\right) = \frac{c - 1}{d} \left(2 + \hat{d}\frac{c - 1}{d}\right) = -\hat{d}. \tag{19}
\]
However, \(\hat{g}\) is a quantized value, and, hence, only approximating \(\frac{c - 1}{d}\). Therefore, we can conclude that the resulting rotation can be written as
\[
R = \begin{bmatrix} \hat{h} & -\hat{k} \\ \hat{d} & \hat{h} \end{bmatrix}, \tag{20}
\]
where \(\hat{k} = -\hat{g}(2 + \hat{d}\hat{g})\) and \(\hat{h} = 1 + \hat{d}\hat{g}\), when the matrix is derived based on (13).

Here, it can also be noted that if \(\hat{d}\) and \(\hat{g}\) are represented with \(B\) fractional bits, \(\hat{h}\) has \(2B\) fractional bits and \(\hat{k}\) has \(3B\) fractional bits. If the rotation is to be implemented directly as in (20), it is advantageous if all the coefficients have the same word length. This is considered in the next section.

Clearly, \(\hat{d} \approx \hat{k}\), but these will never be identical (if they are identical, then the magnitude cannot be one as discussed earlier). Furthermore, consider the determinant:
\[
\det R = \hat{h}\hat{n} + \hat{d}\hat{k}
\]
\[
= \left(1 + \hat{d}\hat{g}\right)^2 + \hat{d} \left(-\hat{g}\left(2 + \hat{d}\hat{g}\right)\right)
\]
\[
= 1 + 2\hat{d}\hat{g} + \hat{d}^2\hat{g}^2 - 2\hat{d}\hat{g} - \hat{d}^2\hat{g}^2 = 1. \tag{21}
\]
Hence, it is clear that the determinant of the matrix is always one when realized as in (13).

The matrix inverse of \(R\) in (20) is easily expressed as
\[
R^{-1} = \begin{bmatrix} \hat{h} & \hat{k} \\ -\hat{d} & \hat{h} \end{bmatrix}. \tag{22}
\]

Let us now consider the result of the rotation in (20) when applied to rotating an input value \(n + jo = re^{j\alpha}\) as
\[
\begin{bmatrix} l \\ m \end{bmatrix} = \begin{bmatrix} \hat{h} & -\hat{k} \\ \hat{d} & \hat{h} \end{bmatrix} \begin{bmatrix} n \\ o \end{bmatrix}
\]
\[
= \begin{bmatrix} \hat{h}n - \hat{k}o \\ \hat{d}n + \hat{h}o \end{bmatrix} \tag{23}
\]
\[
= \begin{bmatrix} \tilde{hr} \cos(\alpha) - \hat{k}r \sin(\alpha) \\ \hat{d}r \cos(\alpha) + \hat{hr} \sin(\alpha) \end{bmatrix}. \tag{24}
\]

To determine the errors in the magnitude and angle, the "correct" rotation\(^3\) is here defined as:
\[
R' = \begin{bmatrix} \hat{h} & -\gamma \\ \gamma & \hat{h} \end{bmatrix}, \tag{25}
\]

where \(\gamma\) is the geometric mean value of \(\hat{d}\) and \(\hat{k}\), so \(\gamma = \sqrt[\hat{d}]{\hat{k}}\). In this way, \(\hat{h}^2 + \hat{k}^2 = \hat{h}^2 + \gamma^2 = 1\) and, with \(\hat{k} = \hat{d} + \delta\), selecting the positive solution to the second-order equation
\[
\hat{d} = -\delta + \sqrt{4\gamma^2 + \delta^2} \frac{2}{2}, \tag{26}
\]
\[
\hat{k} = \delta + \sqrt{4\gamma^2 + \delta^2} \frac{2}{2}. \tag{27}
\]
The angle of the rotation in (25) is \(\arctan \left(\frac{\gamma}{h}\right)\).

The squared magnitude of the result \(l + jm\) from (23) is
\[
|l + jm|^2
\]
\[
= (\hat{h}n - \hat{k}o)^2 + (\hat{d}n + \hat{h}o)^2 \tag{28}
\]
\[
= (\hat{h}n - \delta + \sqrt{4\gamma^2 + \delta^2} \frac{2}{o})^2 +
\]
\[
\left(-\delta + \sqrt{4\gamma^2 + \delta^2} \frac{2}{n} + \hat{h}o\right)^2 \tag{29}
\]
\[
= r^2 \left(1 + \frac{\delta^2}{2} +
\right.
\]
\[
\left.-\delta \sqrt{\frac{\delta^2}{4}} + 1 \sin \left(2\alpha + \arctan \left(\sqrt{\frac{\delta^2}{\gamma^2}} \right) \right) \right) \tag{30}
\]
\[
\approx r^2 \left(1 + \frac{\delta^2}{2} - \delta \sin \left(2\alpha + \arctan \left(\frac{\gamma}{h} \right) \right) \right). \tag{31}
\]

From this we can see that the error of the squared magnitude is a small constant plus a value depending on the angle of the input value. Hence, the magnitude error of the result is input signal dependent. The average squared magnitude of a rotation as in (17) is
\[
1 + \frac{\delta^2}{2}, \tag{32}
\]
where \(\delta = \hat{k} - \hat{d}\). Also note that \(\arctan \left(\frac{\gamma}{h}\right)\) is the angle of the “correct” rotation in (25), providing a phase shift of the error.

Similarly, the error in the rotation angle can be written as in (33) and approximated as in (34). Again, similarly to the magnitude error, we see that the angle error is input signal dependent.

As later illustrated in the results, this is in contrast to the traditional quantized rotation matrix as in (6), where the resulting magnitude, although never one, is constant and the rotation angle is constant, both exclusively determined by the coefficients of the rotation matrix. This leads to that the integer invertible rotators in (17) and (20) are not linear operator from a complex-valued perspective. However, we do not discuss the implications of this further here.

\(^3\) Note that (20) is no longer a complex multiplication, so this operation is not linear in the complex domain, nor is it a scaled rotation. This leads to that we must define a “correct” rotation as a reference point for later analysis as the results show that both the magnitude and angle of rotation varies with the input data.
3. Design Method

An alternative interpretation of (20), (21), and (22) is that if \(H^2 + Dk = 1\), the rotation is integer invertible. Hence, the natural question is: how can we determine such matrices? We are in addition interested in matrices where \(d \equiv k\).

To move the problem into integers, assume that the fixed-point coefficients are represented using \(B\) fractional bits and multiply each coefficient to get:

\[
H = \hat{h}2^B, \quad D = \hat{d}2^B, \quad K = \hat{k}2^B,
\]

leading to \(H^2 + DK = 2^B\). Furthermore, set \(K = D + \Delta\). This now gives:

\[
2^B = H^2 + D^2 + D\Delta
\]

and, solving for \(D\),

\[
D = -\frac{\Delta \pm \sqrt{\Delta^2 + (2^{(B+1)})^2 - (2H)^2}}{2}.
\]

With

\[
P = \sqrt{\Delta^2 + (2^{(B+1)})^2 - (2H)^2}
\]

(38) can be written as

\[
P^2 + (2H)^2 = \Delta^2 + \left(2^{(B+1)}\right)^2.
\]

Introducing \(Q = 2H\) and \(S = 2^{(B+1)}\), (41) can be written as

\[
P^2 + Q^2 = \Delta^2 + S^2.
\]

From this it can be seen that the problem of determining an integer invertible matrix of the form in (20) is equal to determining an integer that can be expressed as two different sums of squares. With the additional requirement that one of the squared integers must be a power of two, \(S\), and the integer to which its square is summed must be “small”.

Once such \(P, Q, S,\) and \(\Delta\) are found, the corresponding integer and fractional fixed-point matrix values can be readily obtained as:

\[
H = \frac{Q}{2}, \quad D = -\frac{\Delta \pm P}{2}, \quad K = D + \Delta = \frac{\Delta \pm P}{2}
\]

and

\[
\hat{h} = \frac{H}{2^B} = \frac{Q}{S}, \quad \hat{d} = \frac{D}{2^B} = \frac{-\Delta \pm P}{S}, \quad \hat{k} = \frac{K}{2^B} = \frac{\Delta \pm P}{S},
\]

respectively. Here it should be noted that \(D\) and \(K\) can be interchanged. Furthermore, the sign of \(H/h\) can be freely chosen as well as the sign of \(D/d\) and \(K/k\) as long as they are the same for the two latter.

First, note that if all of \(P, Q, S,\) and \(\Delta\) are even there will be another set of integers \(\frac{Q}{2}, \frac{Q}{4}, \frac{Q}{8},\) and \(\frac{Q}{16}\). Hence, we are only interested in solutions where at least one of \(P, Q, S,\) and \(\Delta\) is odd, as the solutions with all even coefficients can easily be obtained by shifting the coefficients from a solution with at least one odd coefficient. It can be shown that if both numbers in one pair are odd, the other pair also contains odd numbers, and if the two numbers in a pair have opposite parity, the numbers in the other pair also have opposite parity. Since, \(Q = 2H\) is even and \(S = 2^{(B+1)}\) is even, it follows from this that both \(\Delta\) and \(P\) are odd, and that the resulting sum is odd. This makes sense since \(D = \frac{\Delta \pm P}{2}\) should be an integer, so \(\Delta\) and \(P\) must have the same parity.

Consider the value of the sum of the two squares, \(T = P^2 + Q^2 = \Delta^2 + S^2\). It has been shown that if \(T\) is a composite number only containing prime factors of the form \(4N+1\) or 2 and at least two factors of the form \(4N+1\), there will be at least two different ways to express \(T\) as sums of squares. In fact, if there are \(U\) prime factors of the form \(4N+1\), there are \(2^U\) different sums of squares combinations, although some of them may be identical if there are several identical prime factors. Since we earlier established that the sum is odd for the cases we are interested in, no prime factors in \(T\) are two.

Hence, to design integer invertible rotators, we want to find odd integers \(T\) which are a power of four, since \(S^2 = 2^{2(B+1)} = 4^{(B+1)}\), plus a small odd integer squared, which are composite numbers containing at least two prime factors of the form \(4N+1\). If there are more than two such factors there will be several possible resulting rotations.

Here, note that for a number with odd \(\Delta = 2\Gamma + 1\)

\[
T = 4^{(B+1)} + \Delta^2 = 4^{(B+1)} + (2\Gamma + 1)^2 = 4^{(B+1)} + 4\Gamma^2 + 4\Gamma + 1.
\]
Hence, \( T \equiv 1 \mod 4 \) and all prime factors are of the form \( 4N + 1 \). This leads to that every candidate number \( T = 4^B + \Delta^2 \) will either be a prime or result in, at least one, integer invertible rotator.

To find the factors, a simple search can be performed, although there exist efficient algorithms \([11]\). In Mathematica, the command PowersRepresentations[\( T, 2, 2 \)] can be used. It can be noted that if the prime factors are known, it is possible to derive the terms using closed forms expressions, which for the case of two factors are straightforward. Once the terms are found for a candidate \( T = 4^B + \Delta^2 \), odd \( \Delta \), there will be one or more pairs, (\( P, Q \)), given that \( T \) is composite. Based on the earlier discussion, one will be even and one will be odd. Assigning the even term to be \( Q \), the expressions in (43) to (48) can be applied to obtain the matrix values.

Although it was stated earlier that we are only interested in solutions where at least one of the terms are odd, the terms can be multiplied by two, interchanging \( P \) and \( Q \), and obtain another solution with \( \Delta = 2 \) and one bit longer word length. If the interchange is not done, the solution will just be a scaled version of the previous one, but given the requirement that \( Q \) must be even, it is possible to use the earlier \( P \) value as \( Q \) resulting in a new solution. This is useful to extend the search space further, and, as will be seen in the results section, may result in solutions with a smaller \( \Delta \) for certain word lengths. Further multiplications by two will only result in scaling the values.

Finally, it should be noted that there are solutions to (42) when \( Q = S \) and \( P = \Delta \) or vice versa. However, for the \( Q = S \) solution, either \( D \) or \( K \) is zero, while for the \( P = S \) solution, either \( d \) or \( k \) is larger than one. Hence, these solutions are not of practical interest.

4. Results

4.1. Design Example

To illustrate the procedure of determining the possible rotations, consider the values \( B = 2 \), so \( S = 2^3 = 8 \), and \( \Delta = 1 \). Now,
\[
T = 4^B + \Delta^2 = 4^3 + 1^2 = 65. \tag{52}
\]
65 is not a prime, and a search finds that
\[
65 = 7^2 + 4^2 = 7P^2 + Q^2. \tag{53}
\]
As 4 is the even number, we assign \( Q = 4 \) and \( P = 7 \). From this and (43) to (45), selecting the positive solution to the second-order equation
\[
H = \frac{Q}{2} = 2 \tag{54}
\]
\[
D = \frac{-\Delta + P}{2} = 3 \tag{55}
\]
\[
K = D + \Delta = 4. \tag{56}
\]
As discussed earlier, \( D \) and \( K \) may be interchanged and the signs selected to end up in the correct quadrant. This leads to the rotation matrix
\[
R = \frac{1}{2^2} \begin{bmatrix} 2 & -4 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{2} \\ \frac{3}{4} & \frac{1}{2} \end{bmatrix}. \tag{57}
\]
The determinant is
\[
\det R = \frac{1}{2} \times 1 - \frac{3}{4} \times -1 = \frac{1}{4} + \frac{3}{4} = 1. \tag{58}
\]
Now, consider the separation of \( R \) into three matrices as in (17). Here, \( \hat{g} \) can be determined from \( \hat{g} = \frac{k-1}{d^3} \). For \( \hat{g} \) to be a fixed-point number, \( d \) must be a power of two, which is not the case in the considered example. However, interchanging \( \hat{k} \) and \( \hat{d} \)
\[
\hat{g} = \frac{\frac{7}{1} - \frac{1}{2}}{1} = -\frac{1}{2} \tag{59}
\]
and
\[
\begin{bmatrix} \frac{1}{2} - \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}. \tag{60}
\]
Multiplying all integers by two and exchanging \( P \) and \( Q \), so \( \Delta = 2 \), \( P = 8 \), and \( Q = 14 \), give a rotation with \( B = 3 \) as
\[
H = \frac{Q}{2} = 7 \tag{61}
\]
\[
D = \frac{-\Delta + P}{2} = 3 \tag{62}
\]
\[
K = D + \Delta = 5 \tag{63}
\]
and
\[
R = \frac{1}{2^3} \begin{bmatrix} 7 & -5 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} \frac{7}{8} & -\frac{5}{8} \\ \frac{3}{8} & \frac{5}{8} \end{bmatrix}. \tag{64}
\]
This rotation can not be separated into three matrices represented using binary fixed-point coefficients, and, hence, can not be derived using the technique of rounding \( g = \frac{c-1}{d^3} \), despite being integer invertible.

4.2. Numerical Errors

To illustrate the numerical behavior of lifting-based rotations, consider rotation by \( 45^\circ = \frac{\pi}{4} \) rad. Using six fractional bits, a candidate lifting-based rotation with \( \delta = 1 \) is
\[
R = \frac{1}{2^6} \begin{bmatrix} 46 & -45 \\ 44 & 46 \end{bmatrix}. \tag{65}
\]
Separating a rotation into three matrices as in (13) leads to
\[
d = \sin \left( \frac{\pi}{4} \right) = \frac{1}{\sqrt{2}} \Rightarrow \hat{d} = \frac{45}{64}. \tag{66}
\]
\[
g = \cos \left( \frac{\pi}{4} \right) \frac{1}{d} = 1 - \sqrt{2} \Rightarrow \hat{g} = -\frac{27}{64}. \tag{67}
\]
and the lifting-based rotation
\[
\begin{bmatrix} 1 & -\frac{27}{64} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{45}{64} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{27}{64} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2881}{20896} & -\frac{188179}{20896} \\ \frac{2881}{20896} & \frac{188179}{20896} \end{bmatrix}. \tag{68}
\]
This rotation has $\Delta = 3959$, but with 18 fractional bits, so $\delta = \frac{\Delta}{2^{16}} \approx 0.00156$. Hence, in this case, one may expect the errors to be similar.

For comparison, using the rotation in (3), when rounded to six fractional bits, yields

$$R = \frac{1}{2^6} \begin{bmatrix} 45 & -45 \\ 45 & 45 \end{bmatrix}.$$  \hfill (67)

In Fig. 2, the resulting magnitude and angle are shown when the value to be rotated is taken from the unit circle. As predicted, both the magnitude and angle of the lifting-based/integer invertible rotations have a sinusoidal magnitude and angle depending on the angle of the input value. The rotation matrix in (67) has a constant, but non-unity, magnitude and a constant angle. In this particular case, the angle is exactly correct. However, this is not the case in general.

To see the correctness of the error derivations and the corresponding approximations, the magnitude and angle errors are shown in Fig. 3 for the rotation in (65). The “correct” rotation here has a magnitude of 1 and angle of $\arctan\left(\frac{c}{d}\right) \approx 44.0486^\circ$. It is clear that the derivation is correct and it is even hard to distinguish the difference between the actual value, computed from performing the actual rotation based on the same results as shown in Fig. 2, and the approximations from (31) and (34). It is also clear that the assumption of the “correct” rotation $R'$ in (25) is valid since the error curves in Fig. 3 are symmetric around 0.

4.3. Resulting Rotations

In Table 1 all resulting values of $P$ and $Q$ are shown with $B \leq 13$ and smallest possible odd value of $\Delta$. When $\Delta = 1$, the resulting structures for one bit longer word length and $\Delta = 2$ are also shown. From these values $H$ and $D/K$ are determined based on (43) to (45). In addition, $-G = -g2^B$ is shown for those rotations which are separable into three matrices with fixed-point coefficients as discussed in the design example. For all these rotations, $D$ is chosen to be the power of two value in the $D/K$ column, although the values are still listed in ascending order. All rotations without a value in the $G$ column can not be separated. The first two rows of Table 1 shows the two rotations from the design example.

From Table 1 it is clear that already for these short word lengths and small $\Delta$ range covered, there are many more rotations which can not be separated in three matrix multiplications with integer/fixed-point values compared to those that can. Out of the 59 listed rotations, only eleven can be separated. This shown the importance of Section 3, as the other 48 can only be obtained using the proposed design method. Hence, a much larger design space of integer invertible rotations is obtained using the proposed design method as compared to the earlier one based on quantizing the $d$ and $g$ values in (13).

5. Conclusion

In this work, lifting-based complex multiplications and rotations were considered. It was shown that both the magnitude and angle error is input signal dependent. Closed form expressions were derived in (30) and (33) with approximating expressions provided in (31) and (34) for the magnitude and angle, respectively. Furthermore, a new method to design these integer invertible rotations was proposed which yields a much larger design space compared to separating the rotation in three matrix multiplications.

Although the design method can generate all possible rotations with a given coefficient word length and within a given difference between the included terms, there is
TABLE 1. POSSIBLE INTEGER REVERSIBLE ROTATIONS WITH $B \leq 13$, SMALLEST POSSIBLE ODD $\Delta$, AND, WHEN APPLICABLE, $\Delta = 2$.

<table>
<thead>
<tr>
<th>$B$</th>
<th>$\Delta$</th>
<th>$P$</th>
<th>$Q$</th>
<th>$H$</th>
<th>$D/K$</th>
<th>$-G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>3/4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
<td>14</td>
<td>6</td>
<td>4/7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>25</td>
<td>10/12</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>31</td>
<td>8</td>
<td>4</td>
<td>15/16</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>16</td>
<td>62</td>
<td>9</td>
<td>7/9</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>40</td>
<td>50</td>
<td>20</td>
<td>10/12</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>127</td>
<td>16</td>
<td>8</td>
<td>63/64</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>103</td>
<td>76</td>
<td>38</td>
<td>51/52</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>89</td>
<td>92</td>
<td>46</td>
<td>44/45</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>112</td>
<td>62</td>
<td>31</td>
<td>55/57</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>72</td>
<td>156</td>
<td>78</td>
<td>100/103</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>511</td>
<td>32</td>
<td>16</td>
<td>255/256</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>473</td>
<td>196</td>
<td>98</td>
<td>236/237</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>409</td>
<td>308</td>
<td>154</td>
<td>204/205</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>287</td>
<td>424</td>
<td>212</td>
<td>143/144</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>281</td>
<td>428</td>
<td>214</td>
<td>140/141</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>167</td>
<td>484</td>
<td>242</td>
<td>133/134</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>127</td>
<td>496</td>
<td>248</td>
<td>63/64</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>481</td>
<td>904</td>
<td>452</td>
<td>240/241</td>
<td>328</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>64</td>
<td>1022</td>
<td>511</td>
<td>31/33</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>184</td>
<td>178</td>
<td>89</td>
<td>91/93</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>339</td>
<td>2344</td>
<td>1172</td>
<td>1679/1680</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3041</td>
<td>2744</td>
<td>1372</td>
<td>1520/1521</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>511</td>
<td>4064</td>
<td>2032</td>
<td>255/256</td>
<td>128</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2456</td>
<td>3278</td>
<td>1639</td>
<td>1227/1229</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2352</td>
<td>2554</td>
<td>1777</td>
<td>1675/1677</td>
<td>1285</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>8191</td>
<td>128</td>
<td>64</td>
<td>4095/4096</td>
<td>4092</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8161</td>
<td>712</td>
<td>356</td>
<td>4080/4081</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7681</td>
<td>2848</td>
<td>1424</td>
<td>3840/3841</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6887</td>
<td>4436</td>
<td>2218</td>
<td>3443/3444</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6553</td>
<td>4916</td>
<td>2458</td>
<td>3276/3277</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5017</td>
<td>6476</td>
<td>3238</td>
<td>2508/2509</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4327</td>
<td>6956</td>
<td>3478</td>
<td>2163/2164</td>
<td>1285</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>4688</td>
<td>6718</td>
<td>3359</td>
<td>2343/2345</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6082</td>
<td>2744</td>
<td>3041</td>
<td>2743/2745</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8128</td>
<td>1022</td>
<td>511</td>
<td>4063/4065</td>
<td>1285</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>7711</td>
<td>14456</td>
<td>7228</td>
<td>3585/3586</td>
<td>4092</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>256</td>
<td>16382</td>
<td>8191</td>
<td>127/129</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1428</td>
<td>16322</td>
<td>8161</td>
<td>711/713</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5696</td>
<td>15362</td>
<td>7681</td>
<td>2847/2849</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>8872</td>
<td>13774</td>
<td>6887</td>
<td>4435/4437</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>9832</td>
<td>13106</td>
<td>6553</td>
<td>4915/4917</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>12952</td>
<td>10034</td>
<td>5017</td>
<td>6475/6477</td>
<td>1285</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>13912</td>
<td>8654</td>
<td>4327</td>
<td>6955/6957</td>
<td>1285</td>
</tr>
</tbody>
</table>

currently no other way to design a rotation for a given angle than to tabulate all possible rotations and select the most suitable.

The results illustrate the derived behavior of the rotations and show that the error approximations follow the error closely. A table of all integer invertible rotations with a word length up to 13 fractional bits, the smallest odd difference between the included term plus potentially one more difference, given that it is two. These results show that the number of rotations increases from eleven using the earlier method with separable matrix multiplications to 59 with the proposed design method.

One may expect a similar behavior for other lifting-based schemes, e.g., 3-D rotations in [12] or the quaternion multiplier in [13].

References


