ANALYSIS OF COMPLEX LNS FFTS

M. Arnold, T. Bailey and J. Cowles
Computer Sci. Dept.
University of Wyoming
Laramie, WY 82071
{ marnold,tbailey,cowles}@uwyo.edu

C. Walter
Dept. of Computation
UMIST
Manchester M60 1QD, UK
c.walter@co.umist.ac.uk

Abstract - The complex-logarithmic number system (CLNS), which represents each complex point in log/polar coordinates, may be practical to implement the Fast Fourier Transform (FFT). The roots of unity needed by the FFT have exact representations in CLNS and do not require a ROM.

We present an error analysis and simulation results for a radix-two FFT that compares a rectangular fixed-point representation of complex numbers to CLNS. We observe that CLNS saves 9-12 bits in word-size for 256-1024 point FFTs compared to the fixed-point number system while producing comparable accuracy.

Keywords: complex numbers, polar coordinates, fixed-point arithmetic, logarithmic number system (LNS), addition, Fast Fourier Transform (FFT)

INTRODUCTION

Typical implementations of the Fast Fourier Transform (FFT) [2],[9] need complex arithmetic. The conventional approach to complex arithmetic uses pairs of real numbers that represent points in a rectangular-coordinate system. To multiply two complex numbers, X and Y, represented in a rectangular-coordinate system requires separate computation of $\Re[X] \cdot \Re[Y] - \Im[X] \cdot \Im[Y]$ and $\Re[X] \cdot \Im[Y] + \Im[X] \cdot \Re[Y]$, where $\Re[X]$ is the real part of X and $\Im[X]$ is the imaginary part. There are many ways the real arithmetic for $\Re[X]$ and $\Im[X]$ can be implemented: fixed-point, floating-point, or the less known real-logarithmic number system [11],[10]. Swartzlander et. al. [12] as well as Hongyuan and Lee [4] compared these three techniques for a one-dimensional radix-two FFT implemented with rectangular-coordinate complex arithmetic, and found several advantages to using logarithmic arithmetic to represent $\Re[X]$ and $\Im[X]$. Kidd [5] reached a similar conclusion in an analysis of two-dimensional FFTs implemented with rectangular-coordinate use of LNS.

Arnold et al. [1] introduced a generalization of logarithmic arithmetic, known as the Complex-Logarithmic Number System (CLNS), which represents each complex point in log/polar coordinates. Lewis [6] recently designed a 32-bit CLNS ALU using a CORDIC algorithm. This paper presents an error analysis for a conventional radix-two FFT that compares a rectangular fixed-point representation of complex numbers to CLNS. The conclusion, which has been verified by simulation, shows that CLNS is significantly more compact than a comparable fixed-point representation with no loss of speed. This should translate into lower switching activity during the calculation of the FFT, thereby lower power consumption. Paliouras and Stouraitis [8] have already shown such activity is lower in conventional LNS when word lengths are similar to fixed-point, thus we expect a further savings due to the more compact CLNS representation described here.

COMPLEX LOGARITHMIC NUMBER SYSTEM

We make the distinction between the mathematical concept of an exact value to be represented and its possibly finite-precision representation. Variables and functions used for exact complex values are shown in upper-case.

To begin, let us consider only the case of exact representations. An upper-case variable without a subscript, such as X, describes an exact complex value whose representation is unspecified. An upper-case variable with a subscript describes some aspect of representing a complex value in a particular but exact technique. The most common approach is the rectangular representation, $X_R = \Re[X] + i\Im[X]$, where $i = \sqrt{-1}$.

We are considering an alternative log-polar approach[1], $X_P = X_L + X_\theta \cdot i$, where X_L is the logarithm (base b) of the length of a vector in the complex plane and $-\pi < X_\theta \leq \pi$ is the angle of that vector:

$$\begin{aligned} X_L &= 0.5 \cdot \log_b(\Re[X]^2 + \Im[X]^2) \\ X_\theta &= \arctan(\Re[X], \Im[X]) \end{aligned} \tag{1}$$

The $\arctan(\Re[X], \Im[X])$ function handles all four quadrants, including cases such as $\Re[X] = 0$ or $\Im[X] = 0$ (like `atan2` in C). On output, this can be converted back to rectangular form:

$$\begin{aligned} \Re[X] &= b^{X_L}\cos(X_\theta) \\ \Im[X] &= b^{X_L}\sin(X_\theta) \end{aligned} \tag{2}$$

ARITHMETIC ALGORITHMS

Multiplication and division of complex values are trivial in this system. Given the exact polar-complex-logarithmic representations, X_P and Y_P, the result of multiplication, $Z_P = X_P + Y_P$, can be computed by two parallel adders:

$$\begin{aligned} Z_L &= X_L + Y_L \\ Z_\theta &= X_\theta + Y_\theta \end{aligned} \tag{3}$$

[1]Unless $b = e$, our approach differs slightly (for hardware simplicity and consistency with [1] and [7]) from $\log_b(X) = \ln(X)/\ln(b)$ in complex analysis as we do not adjust X_θ by $1/\ln(b)$.

For example, if $X = -1 + i$ and $Y = 4i$, the $b = 2$ polar-logarithmic representations are $X_L = 0.5\log_b(1^2 + 1^2) = 0.5, X_\theta = \arctan(-1,1) = 3\pi/4$ and $Y_L = 0.5\log_b(4^2) = 2.0, Y_\theta = \arctan(0,4) = \pi/2$. The representation of the product is computed simply as $Z_L = X_L + Y_L = 0.5 + 2.0 = 2.5$ and $Z_\theta = X_\theta + Y_\theta = 3\pi/4 + \pi/2 = 5\pi/4$. A quick check reveals that this is correct: $Z = b^{2.5}(\cos(5\pi/4) + i\sin(5\pi/4)) = 4\sqrt{2}(-\sqrt{2}/2 - \sqrt{2}/2i) = -4 - 4i$. In an actual implementation it is not necessary to convert the polar-log representation (Z_P) back to rectangular form, except for output, thus multiplication is performed in the time of a fixed-point addition.

The equivalent operation in fixed-point or floating-point arithmetic requires four (potentially expensive) multiply operations. The same operation in conventional (rectangular) LNS requires two (possibly expensive) LNS-addition operations. Thus, CLNS multiplication offers a significant improvement compared to any rectangular representation.

The conjugate of the value is represented by the conjugate of the representation, which can be formed by negating Z_θ. For example, $Z^* = -4 + 4i$ is represented as $Z_\theta = -5\pi/4$ with the same Z_L. The negative can be formed by adding π to Z_θ. Thus, $-Z = 4 + 4i$ is represented as $Z_\theta = \pi/4$ with the same Z_L.

Division is like multiplication, except the representations are subtracted. For example, $Z = X/Y = (-1 + i)/(4i) = 0.25 + 0.25i$ is computed as $Z_L = 0.5 - 2.0 = -1.5, Z_\theta = 3\pi/4 - \pi/2 = \pi/4$. Again, a superfluous conversion back to rectangular form, $Z = b^{-1.5}(\cos(\pi/4) + i\sin(\pi/4)) = 1/(2\sqrt{2})(\sqrt{2}/2 + \sqrt{2}/2i) = 0.25 + 0.25i$, verifies this calculation.

In practice, when repetitively computing this way, Z_θ will include a large unnecessary multiple of 2π, which can be eliminated by proper quantization, as described in the next section.

The difficult issue for all variations of LNS is addition and subtraction. The most obvious approach is to convert out of LNS format, do the addition, and finally convert back. The cost of three such conversions probably eliminates any advantage one can obtain from LNS multiplication. Instead, all forms of LNS (including CLNS) can use the observation that $X + Y = Y(Z + 1)$, where $Z = X/Y$. This reduces the cost of LNS addition down to that of computing $Z + 1$ and the minor cost for LNS multiplication and division. Thus, the expensive aspect of the problem involves only one variable, rather than two. This simplification is important for CLNS, as Z is a complex variable.

The complex-addition logarithm, $S_b(Z_P)$ is a function [7] that accepts a complex argument, Z_P, the log-polar representation of Z, and returns the corresponding representation of $Z + 1$. Thus, to find the representation, T_P of the sum, $T = X + Y$ simply involves

$$T_P = Y_P + S_b(X_P - Y_P), \tag{4}$$

which is implemented by two subtractors, two ROM lookups, and two adders:

$$\begin{aligned} T_L &= Y_L + \Re[S_b(X_P - Y_P)] \qquad (5) \\ T_\theta &= Y_\theta + \Im[S_b(X_P - Y_P)]. \end{aligned}$$

$S_b(Z_P)$ condenses the following three step process into a single function: a) converting Z_P to rectangular coordinates using (2); b) adding the real value 1.0 to the rectangular equivalent; and c) converting back to log-polar using (1). We can view the resulting complex function in terms of its real and imaginary parts; thus the definition of $S_b(Z_P)$ can be given[2] as:

$$\begin{aligned} \Re[S_b(Z_P)] &= 0.5 \cdot \log_b(1 + 2b^{Z_L}\cos(Z_\theta) + b^{2 \cdot Z_L}) \\ \Im[S_b(Z_P)] &= \arctan(1 + b^{Z_L}\cos(Z_\theta), b^{Z_L}\sin(Z_\theta)) \end{aligned} \tag{6}$$

For example, if again $X = -1 + i$ and $Y = 4i$, let us consider how to determine the sum, $X+Y = -1+5i$. As we saw, the operands are represented by $X_P = 0.5 + 3\pi/4i$, $Y_P = 2.0 + \pi/2i$, and the quotient is represented by $Z_P = -1.5 + \pi/4i$. Since $\sin(\pi/4) = \cos(\pi/4) = \sqrt{2}/2$,

$$\begin{aligned} \Re[S_b(Z_P)] &= 0.5 \cdot \log_2(1 + 2 \cdot 2^{-1.5}\sqrt{2}/2 + 2^{-3}) = \log_2(\sqrt{26}/4) \approx 0.350 \\ \Im[S_b(Z_P)] &= \arctan(1 + b^{-1.5}\sqrt{2}/2, b^{1.5}\sqrt{2}/2) = \arctan(1.25, .25) \approx 0.197, \end{aligned}$$

thus from (5), the result is:

$$\begin{aligned} T_P &= (2.0 + \pi/4i) + (\log_2(\sqrt{26}) - 2.0) \ + \arctan(1.25, .25)i) \\ &= \log_2(\sqrt{26}) + (\arctan(1.25, .25)) + \pi/4)i \approx 2.350220 + 1.768191i. \end{aligned}$$

A quick check reveals that $\sqrt{26} \cdot \cos(T_\theta) + \sqrt{26} \cdot \sin(T_\theta)i \approx -1 + 5i$.

Unlike conventional LNS, which needs a separate function to deal with the $1 - Z$ term that arises in subtraction, the complex-addition logarithm covers the extreme cases of $1 + Z$ and $1 - Z$ as well as cases in between, such as $1 + iZ$. Thus, to implement subtraction, it is only necessary to add π to Y_θ before carrying out the above algorithm (4).

FINITE EFFECTS

Hardware realizations do not have unlimited resources to devote to the representation of a complex value; thus we are forced to truncate this ideal representation to a finite-precision one. The goal of this paper is to describe the effects that quantization has on FFTs. Variables used for finite-precision (integer) complex representations are shown in lower-case. We represent a complex value, X, as a quantized complex-logarithmic representation, $x = x_L + x_\theta \cdot i$, composed of the quantized logarithm of the length of a vector in the complex plane,

$$x_L = q\left(\frac{X_L}{\Delta_L}\right), \tag{7}$$

[2]There was a typographical error in formula (38) of [1]. When $Z_\theta = 0$, the correct $\Re[S_b(Z_P)]$ yields $\log_b(1 + b^{Z_L})$, the real addition logarithm [11]. Similarly, when $Z_\theta = \pi$, $\Re[S_b(Z_P)]$ yields $\log_b|1 - b^{Z_L}|$, the real subtraction logarithm.

and the quantized angle of that vector,

$$x_\theta = q\left(\frac{X_\theta}{\left(\frac{2\pi}{m_\theta}\right)}\right) \bmod m_\theta. \tag{8}$$

Here the function q provides a way to quantize its real argument to an integer, the constant Δ_L determines the precision of the radial representation and the constant m_θ determines the precision of the angle. To represent pure negative real numbers m_θ must be divisible by two[3] and to represent pure imaginary numbers m_θ must be divisible by four. Using modulus operations in all angular formulae prevents x_θ from ever going outside the range for which the complex-addition logarithm is tabulated. Choosing m_θ as a power of two means these modulus operations occur at no cost in binary[4].

FFT ERROR ANALYSIS

This section presents an error analysis of a radix-two Fast Fourier Transform (FFT) for the traditional rectangular fixed-point representation and for the complex-logarithmic representation proposed here using a set of assumptions that are more favorable for the traditional approach.

The FFT requires a sequence of butterfly calculations of the form $A+W\cdot B$, where A and B are complex values from a previous stage (or from the input) and W is a complex root of unity. There are $k = \log_2(n)$ stages in the FFT of a set of n numbers, n being a power of two. We assume $n \leq m_\theta$.

Let $\mathrm{err}^2(C)$ be the squared absolute error of C. Let $\mathrm{mag}^2(C)$ be the squared magnitude of C. The expected statistics are taken over all values after the ith stage of the FFT: err_i^2 is the expected squared error, and mag_i^2 is the expected squared magnitude. The squared absolute error, $\mathrm{err}^2(A + W \cdot B)$, that occurs in $A + W \cdot B$ is due to the propagated error from A, W and B as well as due to limited-precision representation (either rectangular fixed-point or polar-logarithmic). We will assume that the exact value of W can be represented in the hardware unit that computes $A + W \cdot B$. This assumption holds for a CLNS unit where m_θ is a multiple of n, and approximately holds for a fixed-point unit that carries out the computation of $A + W \cdot B$ in double precision. So

$$\mathrm{err}^2(A + W \cdot B) = \mathrm{err}^2(A) + \mathrm{err}^2(B)\mathrm{mag}^2(W) + \zeta^2, \tag{9}$$

where ζ^2 is the error due to rounding $A + W \cdot B$ to fit in the finite representation. Since W is a root of unity, its magnitude is one, and so $\mathrm{mag}^2(W) = 1$. Thus:

$$\mathrm{err}^2(A + W \cdot B) = \mathrm{err}^2(A) + \mathrm{err}^2(B) + \zeta^2. \tag{10}$$

[3]The case of $m_\theta = 2$ is isomorphic to the conventional real-valued LNS.

[4]m_θ could be the product of relatively prime moduli if the Residue Number System (RNS) were used for x_θ instead.

For a rectangular fixed-point representation of complex numbers, the representational error is absolute. Let δ^2 be the squared representational error inherent to the input and ζ_i^2 be the squared representational error introduced by quantizing a value to the nearest fixed-point representation after the ith stage of the FFT. The expected error after each stage of the FFT is given by

$$\begin{aligned} \mathrm{err}_0^2 &= \delta^2 & (11) \\ \mathrm{err}_i^2 &= \mathrm{err}_{i-1}^2 + \mathrm{err}_{i-1}^2 + \zeta_i^2. & (12) \end{aligned}$$

(12) derives from recursive application of (10), but ζ_i^2 varies for different stages of the FFT. For the first and second stages, $\zeta_i^2 = 0$ because $W = \pm 1$ or $\pm i$ and so no error is introduced by the computation. In later stages, higher roots of unity are used and thus error occurs because of the quantization of the result. We assume that the squared error for a W other than ± 1 or $\pm i$ is δ^2. Thus,

$$\zeta_1^2 = 0, \zeta_2^2 = 0, \zeta_3^2 = \frac{1}{2}\delta^2, \zeta_4^2 = \frac{3}{4}\delta^2, \ldots, \zeta_k^2 = (1 - 2^{2-k})\delta^2, \ldots \tag{13}$$

In other words, one half of the points $(\pm\sqrt{2}/2 \pm \sqrt{2}/2i)$ in the third stage are not exact. In the fourth stage, only four (± 1 and $\pm i$) of the sixteen values of W are exact, etc. The resulting squared error after k stages of the fixed-point FFT is:

$$\mathrm{err}_k^2 = 2^k \cdot \left(\frac{7}{6} - 2^{-k} + \frac{4}{3} \cdot 4^{-k}\right) \cdot \delta^2 = \left(\frac{7}{6} \cdot 2^k - 1 + \frac{4}{3} \cdot 2^{-k}\right) \cdot \delta^2 \approx \frac{7}{6} \cdot 2^k \cdot \delta^2. \tag{14}$$

For CLNS, the representational error is proportional to the magnitude of the represented points. Let ε^2 be the squared relative error introduced by quantizing a value to the nearest logarithmic representation. We assume this relative error, ε^2, is independent of the FFT stage used. The expected squared error after each stage of the FFT is given by:

$$\begin{aligned} \mathrm{err}_0^2 &= \varepsilon^2 \mathrm{mag}_0^2 & (15) \\ \mathrm{err}_i^2 &= \mathrm{err}_{i-1}^2 + \mathrm{err}_{i-1}^2 + \varepsilon^2 \mathrm{mag}_i^2. & (16) \end{aligned}$$

Solving for the expected error after k stages of the FFT:

$$\mathrm{err}_k^2 = \varepsilon^2 \cdot 2^k \cdot \sum_{i=0}^{k} \frac{\mathrm{mag}_i^2}{2^i} \tag{17}$$

The pairing of butterfly calculations ($A + W \cdot B$ and $A - W \cdot B$) at each stage of the FFT causes the expected squared magnitude to double, and so $\mathrm{mag}_i^2 = 2 \cdot \mathrm{mag}_{i-1}^2$. Thus:

$$\mathrm{err}_k^2 = \varepsilon^2 \cdot 2^k \cdot \mathrm{mag}_0^2(k+1) \tag{18}$$

As a typical example, suppose $n = 256$, so that $k = 8$. Let the initial distribution of complex numbers be chosen randomly from a uniform circular disk of radius one centered at the origin. For this distribution, $\text{mag}_0^2 = 0.5$. Applying (14) we have:

$$\text{err}^2_{\text{Rect}} = \frac{7}{6} \cdot 256 \cdot \delta^2 \approx 299.7\delta^2. \tag{19}$$

By comparison, applying (18) yields:

$$\text{err}^2_{\text{CLNS}} = 256 \cdot 0.5 \cdot 9 \cdot \varepsilon^2 = 1152\varepsilon^2. \tag{20}$$

At first glance fixed-point appears to be four times better but we will show later that CLNS has advantages that more than compensate for this disparity.

FFT SIMULATION

(14) and (18) were tested to see how accurately they predict the squared errors encountered in fixed-point- and CLNS- FFTs, respectively. In each experiment, the predicted squared error was compared against simulation results using appropriate limited-precision arithmetic for $n = 256$ and $n = 1024$ with noise, impulse plus noise and single frequency plus noise inputs. The noise was randomly chosen from a uniform distribution on a unit disk.

Each FFT was run using IEEE-754 `double` arithmetic. The FFT was then re-run with the same input data at progressively coarser precisions using both the rectangular-fixed-point and complex-logarithmic systems. The squared error for each run was calculated by subtracting each point of the limited-precision result from the corresponding point of the `double` result. Our simulation then reports the mean of the square of the magnitude of these differences.

For each of the experiments we conducted, we found the observed errors were in close agreement with those predicted by (14) and (18). For example, Table 1 shows the observed and predicted squared errors as well as their ratio in the rectangular-fixed-point system for a 1024 point FFT of a noisy sine wave, $(0.37 - 0.28i)e^{23i\theta} + \text{noise}(0.53)$.

Table 1. Rectangular Error.

Precision	Observed	Predicted	Ratio
$1.0 \cdot 10^{-5}$	$2.15 \cdot 10^{-8}$	$1.99 \cdot 10^{-8}$	1.08
$1.0 \cdot 10^{-4}$	$1.99 \cdot 10^{-6}$	$1.99 \cdot 10^{-6}$	1.00
$1.0 \cdot 10^{-3}$	$1.94 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$	0.97
$1.0 \cdot 10^{-2}$	$1.99 \cdot 10^{-2}$	$1.99 \cdot 10^{-2}$	1.00
$1.0 \cdot 10^{-1}$	$2.00 \cdot 10^{-0}$	$1.99 \cdot 10^{-0}$	1.01

Because we assume quantization errors are uniformly distributed, δ^2 is one sixth of the fixed-point precision, and so the predicted squared error is $7/6 \cdot 1024 \cdot 1/6 \approx 199.11$ times the square of the precision. As the ratio column indicates, the prediction from (14) closely follows the observed simulation results. Table 2 shows results for a similar CLNS simulation using the same input as Table 1.

Table 2. CLNS Error.

Precision	Observed	Predicted	Ratio
$1.0 \cdot 10^{-5}$	$8.16 \cdot 10^{-8}$	$6.68 \cdot 10^{-8}$	1.22
$1.0 \cdot 10^{-4}$	$6.46 \cdot 10^{-6}$	$6.68 \cdot 10^{-6}$	0.97
$1.0 \cdot 10^{-3}$	$6.79 \cdot 10^{-4}$	$6.68 \cdot 10^{-4}$	1.02
$1.0 \cdot 10^{-2}$	$6.68 \cdot 10^{-2}$	$6.68 \cdot 10^{-2}$	1.00
$1.0 \cdot 10^{-1}$	$6.68 \cdot 10^{-0}$	$6.68 \cdot 10^{-0}$	1.00

Here $\text{mag}_0^2 = 0.3557$ and so the predicted squared error is $11 \cdot 1024 \cdot 1/6 \cdot 0.3557 \approx 667.77$ times the square of the relative precision. The ratio column indicates that the prediction from (18) closely matches the observed simulation results up to a precision of 10^{-5}, a useful level of precision for most FFT applications.

COMPRESSION ADVANTAGE OF CLNS FFTS

For realistic values of $k \leq 10$, when $\varepsilon^2 \approx \delta^2$, the squared error for the rectangular representation (14) is about three to four times better than that of the complex-logarithmic representation (18), but this is not the proper comparison for us to consider. Rather, we are interested in the number of bits required to achieve the same amount of error using these two representations.

For the rectangular representation, we must represent values between $-n$ to $+n$ in both the real and imaginary directions. If the distance between represented points in the rectangular fixed-point system is Δ_R, then the total number of points to be represented is

$$\text{points}_{\text{Rect}} = \left(\frac{2n}{\Delta_R}\right) \cdot \left(\frac{2n}{\Delta_R}\right) = \frac{2^{2+2k}}{\Delta_R^2} \tag{21}$$

and the number of bits is

$$\text{bits}_{\text{Rect}} = 2 + 2k - \log_2(\Delta_R^2). \tag{22}$$

For CLNS, we must represent values with magnitudes between some minimum value and n. We have found that a minimum value of $\frac{1}{n}$ is adequate for most distributions. For example, if n points are randomly chosen from a uniform distribution on a disk of radius one, the probability of at least one point with magnitude less than $\frac{1}{n}$ is about $\frac{1}{n}$. In the radial direction, the number of points is

$$\frac{\log_2(n) - \log_2(\frac{1}{n})}{\log_2(1+\Delta_L)} \approx \frac{2k}{\log_2(e) \cdot \Delta_L}. \tag{23}$$

To simplify the discussion, we will assume that in the angular direction, the number of points is $m_\theta = 2\pi/\Delta_L$. This assumption has the effect of making the absolute precision of points near ± 1 and $\pm i$ be similar to that of a fixed-point system. Thus, for the CLNS, the number of points is

$$\text{points}_{\text{CLNS}} = \frac{4\pi \cdot \log_e(2) \cdot k}{\Delta_L^2} \tag{24}$$

and the number of bits is

$$\text{bits}_{\text{CLNS}} = \log_2(4\pi \log_e(2)) + \log_2(k) - \log_2(\Delta_L^2). \tag{25}$$

If we assume a uniform distribution of the points to be quantized, then $\delta^2 = \Delta_R^2/6$ for the rectangular representation. From (14) we have

$$\text{err}_k^2 = \frac{7}{6} \cdot 2^k \cdot \frac{1}{6} \cdot \Delta_R^2 \tag{26}$$

or

$$\frac{1}{\Delta_R^2} = \frac{7}{6} \cdot 2^k \cdot \frac{1}{6} \cdot \frac{1}{\text{err}_k^2} \tag{27}$$

for the rectangular representation. Combining (22) with (27), we have

$$\text{bits}_{\text{Rect}} = 3k - 0.36 - \log_2(\text{err}_k^2). \tag{28}$$

In a similar way for CLNS, $\varepsilon^2 \approx \Delta_L^2/6$ and so from (18) we have

$$\text{err}_k^2 = (k+1) \cdot 2^k \cdot \frac{1}{6} \cdot \Delta_L^2 \cdot \text{mag}_0^2 \tag{29}$$

or

$$\frac{1}{\Delta_L^2} = (k+1) \cdot 2^k \cdot \frac{1}{6} \cdot \frac{\text{mag}_0^2}{\text{err}_k^2}. \tag{30}$$

Combining (25) with (30), we have

$$\text{bits}_{\text{CLNS}} = k + \log_2(k^2 + k) + 0.54 + \log_2(\text{mag}_0^2) - \log_2(\text{err}_k^2). \tag{31}$$

The first terms in (28) and (31) depend on $k = \log_2(n)$, the number of stages in the FFT. The last term depends on the required accuracy of the calculation. Given that the initial input values fall within the unit circle, $\text{mag}_0^2 < 1$, we have:

$$\begin{aligned} n = 8,\quad & \text{bits}_{\text{Rect}} = 23.6 - \log_2(\text{err}_k^2), \quad \text{bits}_{\text{CLNS}} < 14.7 - \log_2(\text{err}_k^2); \\ n = 10,\quad & \text{bits}_{\text{Rect}} = 29.6 - \log_2(\text{err}_k^2), \quad \text{bits}_{\text{CLNS}} < 17.3 - \log_2(\text{err}_k^2). \end{aligned}$$

CLNS requires about 9 fewer bits than a fixed-point representation for the same accuracy in a 256-point FFT and about 12 fewer bits in a 1024-point FFT. These savings are significant, since it results in narrower bus-widths throughout the entire processor and lower propagation delay for many arithmetic units. A consequence of the narrower bus width is that bus-switching activity will be lower for the CLNS processor, resulting in lower power consumption due to bus activity.

SPEED AND COST

Let β_C = bits$_{\text{CLNS}}$. The most straightforward CLNS unit requires no more than[5] $3\beta_C$ full adders and a ROM with β_C address input bits. By appropriate application of the identities $S_b(-Z_P) = S_b(Z_P) - Z_P$ and $S_b(Z_P^*) = S_b^*(Z_P)$, two address inputs can be eliminated. The former relationship is well-known for real-valued LNS; the latter conjugate relationship was described by Mehmke [7]. An additional β_C full adders allow computation of Z_L, $-Z_L$, Z_θ and $-Z_\theta$ in parallel. From these four outputs, it is possible to detect the signs of Z_L and Z_θ and to form Z_P, $-Z_P$, Z_P^* or $-Z_P^*$. One of these four possible complex values is used as the address input to the S_b ROM, and in turn the output of the ROM forms the CLNS representation of the sum:

$$\begin{array}{lll}
z_L < 0 & Z_\theta > 0 & T_P = S_b(Z_P) + \max_L(X_P, Y_P) \\
z_L < 0 & Z_\theta < 0 & T_P = S_b^*(Z_P^*) + \max_L(X_P, Y_P) \\
z_L > 0 & Z_\theta > 0 & T_P = S_b^*(-Z_P^*) + \min_L(X_P, Y_P) \\
z_L > 0 & Z_\theta < 0 & T_P = S_b(-Z_P) + \min_L(X_P, Y_P)
\end{array}$$

where $\max_L(X_P, Y_P)$ is X_P and $\min_L(X_P, Y_P)$ is Y_P if $Z_L > 0$, otherwise $\max_L(X_P, Y_P)$ is Y_P and $\min_L(X_P, Y_P)$ is X_P.

Frey and Taylor [3] observed when tabulating the real-valued functions ($\Re[S_b(Z_L)]$ and $\Re[D_b(Z_L)]$) for negative real arguments ($Z_L < -1$) that the number of discardable address bits is $\lfloor -Z_L \rfloor$. This is due to the fact that the derivative for the tabulated function is less than the weights of the discarded bits. Here, to generalize to CLNS let $\Re = \Re[S_b(Z_P)]$ and $\Im = \Im[S_b(Z_P)]$. The partial derivatives are:

$$\frac{\partial \Re}{\partial Z_L} = \frac{\partial \Im}{\partial Z_\theta} = \frac{b^{Z_L}\,(b^{Z_L} + \cos(Z_\theta))}{1 + b^{2Z_L} + 2\,b^{Z_L}\,\cos(Z_\theta)} \tag{32}$$

$$\frac{\partial \Im}{\partial Z_L} = -(\ln(b))^2 \frac{\partial \Re}{\partial Z_\theta} = \frac{b^{Z_L}\,\sin(Z_\theta)}{(1 + b^{2Z_L} + 2\,b^{Z_L}\,\cos(Z_\theta))\,\ln(b)} \tag{33}$$

For $Z_L < -1$, $\Re[S_b'(Z_L)] \ge \frac{\partial \Re}{\partial Z_L} \ge \Re[D_b'(Z_L)]$. Since for $Z_L < -1$, $|\frac{\partial \Re}{\partial Z_L}| > |\frac{\partial \Im}{\partial Z_L}|$, table reduction of Z_L in CLNS can proceed analogously to [3], saving two address bits for $\beta_C = 16$. Assuming[6] we can eliminate one bit from the address bits for Z_θ, CLNS requires a $\beta_C \cdot 2^{\beta_C - 5}$ bit ROM. The roots of unity (W) used by the FFT have exact representations in CLNS, and do not have to be stored in a ROM.

Let β_R = bits$_{\text{Rect}}$. A fixed-point implementation requires $\beta_R^2 + \beta_R$ full adders. Since for realistic values of n, as derived in the previous section, $\beta_C + 9 \le \beta_R$, the savings are at least $\beta_C^2 + 15 \cdot \beta_C + 90$ full adders. In contrast to CLNS, W must be stored in a $\beta_R \cdot 2^n$ bit fixed-point ROM, which is smaller than the CLNS S_b table described above. For $n = 8$, 16-bit CLNS is roughly

[5]Since W is a root of unity, $W_L = 0$; thus we can eliminate roughly $\beta_C/2$ full adders (for $W_L + B_L$) leaving only $W_\theta + B_\theta$ to implement $W \cdot B$.

[6]At the cost of some additional error.

equivalent to 26-bit fixed point[7]. Thus CLNS requires a ROM with 2^{15} bits compared to $26 \cdot 2^8$ for fixed point, or about five times more ROM[8] for CLNS.

Because, for an idealized combinational-logic implementation, the real and imaginary parts can be processed in parallel, a CLNS butterfly implementation only requires the time for three fixed-point additions and one table lookup (S_b). In comparison, a rectangular real-valued LNS implementation requires the time for five fixed-point additions and three table lookups (W and two S_b); and a fixed-point implementation requires the time for a multiplication, two additions and the lookup of W. In many combinational-logic technologies, the CLNS approach would have shortest delay. A more realistic implementation might pipeline these operations, thus the clock frequency would be limited by the larger of the ROM access time and the adder time. If we assume fast ROMs, the clock frequency will be limited by the bus width—which is about 62.5 percent better for CLNS than for fixed-point.

CONCLUSIONS

The complex-logarithmic number system may have practical utility in special-purpose DSP hardware for the FFT. Although other non-traditional representations for complex values are possible, such as RNS, CLNS seems to offer greater storage efficiency. We have analyzed polar-logarithmic representation in contrast to the traditional rectangular fixed-point representation for a radix-two FFT algorithm whose input contains at least some noise and whose input is within the unit circle. We have shown that to obtain outputs of equivalent accuracy requires 9-12 fewer bits for CLNS than for the fixed-point number system with typical size FFTs. These results were verified using simulations for 256- and 1024-point FFTs with various realistic input distributions. CLNS gives higher-speed FFT performance at the expense of area for the S_b ROM, which is offset[9] by savings in area for full adders. The lower switching activity of the compact CLNS representation may offer equivalent FFT results using less power.

ACKNOWLEDGMENTS

The authors acknowledge the support provided for the initial investigation of CLNS [1] by Mark Winkel, of XLNS Research (`www.xlnsresearch.com`). The authors also wish to thank the referees for their valuable comments.

[7]Actually, as described in the previous section, β_R can be as small as 25, but $\Re[X]$ and $\Im[X]$ need equal-sized busses; thus β_R is the next even integer.

[8]A reduction of just two more address bits makes the CLNS S_b ROM similar in size to the fixed-point W ROM. Other techniques [1] provide such savings, but space limitations prevent discussion of interpolation techniques that make this possible.

[9]The $n = 8$, 16-bit CLNS version takes 26,112 bits more than the equivalent 26-bit fixed-point one, however the latter takes 636 more full adders. This averages to 40.9 bits per full adder, which is a reasonable estimate of their relative areas in some CMOS technologies; thus the simple CLNS implementation proposed here (without interpolation) may require similar area as an analogous simple fixed-point implementation.

References

[1] M. G. Arnold, T. A. Bailey, J. R. Cowles and M. D. Winkel, "Arithmetic Co-transformations in the Real and Complex Logarithmic Number Systems," *IEEE Trans. Comput.*, vol. 47, no. 7, pp. 777-786, July 1998.

[2] R. N. Bracewell, *The Fourier Transform and Its Applications,* 2nd ed., revised, New York: McGraw Hill, pp. 370-384, 1986.

[3] M. L. Frey and F. J. Taylor, "A Table Reduction Technique for Logarithmically Architected Digital Filters," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-31, No. 3, pp. 718-719, 1985.

[4] W. Hongyuan and S. C. Lee, "Comment on 'Sign/Logarithm Arithmetic for FFT Implementation'," *IEEE Trans. Comput.*, vol. C-35, pp. 482–484, 1986.

[5] S. J. Kidd, "Implementation of the sign-logarithm arithmetic FFT," *Royal Signals and Radar Establishment Memorandum 3644*, Malvern, 1983.

[6] D. M. Lewis, "Complex logarithmic number system arithmetic using high radix redundant CORDIC algorithms," *Proc. 14th IEEE Symposium on Computer Arithmetic* Adelaide, Australia, pp. 194-203, 14–16 April 1999.

[7] R. Mehmke, "Additionslogarithmen für Complexe Grössen," *Zeitschrift für Mathematik und Physik*, vol. 40, pp. 15-30, 1895.

[8] V. Paliouras and T. Stouraitis, "Low power Properties of the Logarithmic Number System," *Proc. 15th IEEE Symposium on Computer Arithmetic* Vail, Colorado, pp. 229-236, 11–13 June 2001.

[9] R. W. Ramirez, *The FFT: Fundamentals and Concepts,* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[10] T. Stouraitis, "Logarithmic Number System Theory, Analysis, and Design," Ph.D. Dissertation, University of Florida, Gainesville, 1986.

[11] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," *IEEE Trans. Comput.*, vol. C-24, pp. 1238–1242, Dec 1975.

[12] E. E. Swartzlander, et al., "Sign/Logarithm Arithmetic for FFT Implementation," *IEEE Trans. Comput.*, vol. C-32, pp. 526–534, 1983.