2009 19th IEEE International Symposium on Computer Arithmetic

A Dual-Purpose Real/Complex Logarithmic Number System ALU

Mark G. Arnold
Computer Science and Engineering
Lehigh University
maab@lehigh.edu

Sylvain Collange
ELIAUS
Université de Perpignan Via Domitia
sylvain.collange@univ-perp.fr

***Abstract*—The real Logarithmic Number System (LNS) allows fast and inexpensive multiplication and division but more expensive addition and subtraction as precision increases. Recent advances in higher-order and multipartite table methods, together with cotransformation, allow real LNS ALUs to be implemented effectively on FPGAs for a wide variety of medium-precision special-purpose applications. The Complex LNS (CLNS) is a generalization of LNS which represents complex values in log-polar form. CLNS is a more compact representation than traditional rectangular methods, reducing the cost of busses and memory in intensive complex-number applications like the FFT; however, prior CLNS implementations were either slow CORDIC-based or expensive 2D-table-based approaches. This paper attempts to leverage the recent advances made in real-valued LNS units for the more specialized context of CLNS. This paper proposes a novel approach to reduce the cost of CLNS addition by re-using a conventional real-valued LNS ALU with specialized CLNS hardware that is smaller than the real-valued LNS ALU to which it is attached. The resulting ALU is much less expensive than prior fast CLNS units at the cost of some extra delay. The extra hardware added to the ALU is for trigonometric-related functions, and may be useful in LNS applications other than CLNS. The novel algorithm proposed here is implemented using the FloPoCo library (which incorporates recent HOTBM advances in function-unit generation), and FPGA synthesis results are reported.**

Keywords: Complex Arithmetic, Logarithmic Number System, hardware function evaluation, FPGA.

I. Introduction

The usual approach to complex arithmetic works with pairs of real numbers that represent points in a rectangular-coordinate system. To multiply two complex numbers, denoted in this paper by upper-case variables, $\bar{X}$ and $\bar{Y}$, using rectangular-coordinates, involves four real multiplications:

$$\Re[\bar{X}\bar{Y}] = \Re[\bar{X}]\cdot\Re[\bar{Y}] - \Im[\bar{X}]\cdot\Im[\bar{Y}] \quad (1)$$

$$\Im[\bar{X}\bar{Y}] = \Re[\bar{X}]\cdot\Im[\bar{Y}] + \Im[\bar{X}]\cdot\Re[\bar{Y}], \quad (2)$$

where $\Re[\bar{X}]$ is the real part and $\Im[\bar{X}]$ is the imaginary part of a complex value, $\bar{X}$. The bar indicates this is an exact linearly-represented unbounded-precision number. There are many implementation alternatives for the real arithmetic in (1) and (2): fixed-point, Floating-Point (FP), or more unusual systems like the scaled Residue Number System (RNS) [8] or the real-Logarithmic Number System (LNS)[19]. Swartzlander et al. [19] analyzed fixed-point, FP, and LNS usage in an FFT implemented with rectangular coordinates, and found several advantages to using logarithmic arithmetic to represent $\Re[\bar{X}]$ and $\Im[\bar{X}]$. Several hundred papers [26] have considered conventional, real-valued LNS; most have found some advantages for low-precision implementation of multiply-rich algorithms, like (1) and (2).

This paper considers an even more specialized number system in which complex values are represented in log-polar coordinates. The advantage is that the cost of complex multiplication and division is reduced even more than in rectangular LNS at the cost of a very complicated addition algorithm. This paper proposes a novel approach to reduce the cost of this log-polar addition algorithm by using a conventional real-valued LNS ALU, together with additional hardware that is less complex than the real-valued LNS ALU to which it is attached.

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. CLNS was inspired by a nineteenth-century paper by Mehmke [16] on manual usage of such log-polar representation, and shares only the polar aspect with highly-unusual complex-level-index representation [20]. The initial CLNS implementations, like the manual approach, were based on straight table lookup, which grows quite expensive as precision increases since the lookup involves a two-dimensional table.

Cotransformation [2] cuts the cost of CLNS significantly by converting difficult cases for addition and subtraction into easier cases, but the table sizes are still large.

Lewis [14] overcame such large area requirements in the design of a 32-bit CLNS ALU by using a CORDIC algorithm, which is significantly less expensive; however, the implementation involves many steps, making it rather slow. Despite the implementation cost, CLNS may be preferred in certain applications. For example, Arnold et al. [3], [4] showed that CLNS is significantly more compact than a comparable rectangular fixed-point representation for a radix-two FFT, making the memory and busses in the system much less expensive. These savings counterbalance the extra cost of the CLNS ALU if the precision requirements are low enough. Vouzis et al. [21] analyzed a CLNS approach for Orthogonal Frequency Division Multiplexing (OFDM) demodulation of Ultra-Wide Band (UWB) receivers. To reduce the implementation cost of

1063-6889/09 $25.00 © 2009 IEEE
DOI 10.1109/ARITH.2009.26

such CLNS applications, Vouzis et al. [22] proposed using a Range-Addressable Lookup Table (RALUT), similar to [17].

II. Real-valued LNS

The format of the real-valued LNS [18] has a base-b (usually, $b = 2$) logarithm (consisting of k_L signed integer bits and f_L fractional bits) to represent the absolute value of a real number and often an additional sign bit to allow for that real to be negative. We can describe the ideal logarithmic transformation and the quantization with distinct notations. For an arbitrary non-zero real, $\bar{x}$, the ideal (infinite precision) LNS value is $x_L = \log_b |\bar{x}|$. The exact linear $\bar{x}$ has been transformed into an exact but non-linear x_L, which is then quantized as $\hat{x}_L = \lfloor x_L 2^{f_L} + 0.5 \rfloor 2^{-f_L}$. In analogy to the FP number system, the k_L integer bits behave like an FP exponent (negative exponents mean smaller than unity; positive exponents mean larger than unity); the f_L fractional bits are similar to the FP mantissa. Multiplication or division simply require adding or subtracting the logarithms and exclusive-ORing the sign bits.

To compute real LNS sums, a conventional LNS ALU computes a special function, known as $s_b(z) = \log_b(1 + b^z)$, which, in effect, increments the LNS representation by 1.0. (The s reminds us this function is only used for *sums* when the sign bits are the *same*.) The LNS addition algorithm starts with $x_L = \log_b |\bar{x}|$ and $y_L = \log_b |\bar{y}|$ already in LNS format. The result, $t_L = \log_b(\bar{x}+\bar{y})$ is computed as $t_L = y_L + s_b(x_L - y_L)$, which is justified by the fact that $\bar{t} = \bar{x} + \bar{y} = \bar{y}(\bar{x}/\bar{y} + 1)$, even though such a step never occurs in the hardware. Since $\bar{x} + \bar{y} = \bar{y} + \bar{x}$, we can always choose $z_L = -|x_L - y_L|$ [18] by simultaneously choosing the maximum of x_L and y_L. In other words, $t_L = \max(x_L, y_L) + s_b(-|x_L - y_L|)$, thereby restricting $z < 0$.

Analogously to s_b, there is a similar function to compute the logarithm of the *differences* with $t_L = \max(x_L, y_L) + d_b(-|x_L - y_L|)$, where $d_b(z) = \log_b |1 - b^z|$. The decision whether to compute s_b or d_b is based on the signs of the real values.

Early LNS implementations [18] used ROM to lookup $\hat{s}_b$ and $\hat{d}_b$ achieving moderate ($f_L = 12$ bits) accuracy. More recent methods [11], [12], [13], [23], [24] can obtain accuracy near single-precision floating point at reasonable cost. The goal here is to leverage the recent advances made in real-valued LNS units for the more specialized context of CLNS.

III. Complex logarithmic number system

CLNS uses a log-polar approach for $\bar{X} \neq 0$ in which we define, for hardware simplicity,

$$\log_b(\bar{X}) = X = X_L + X_\theta \cdot i, \tag{3}$$

where $X_L = \log_b(\sqrt{\Re[\bar{X}]^2 + \Im[\bar{X}]^2})$ 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[1]. This can be converted back exactly to rectangular form:

$$\begin{aligned} \Re[\bar{X}] &= b^{X_L} \cos(X_\theta) \\ \Im[\bar{X}] &= b^{X_L} \sin(X_\theta). \end{aligned} \tag{4}$$

In a practical implementation, both the logarithm and angle will be quantized:

$$\begin{aligned} \hat{X}_L &= \lfloor X_L 2^{f_L} + 0.5 \rfloor 2^{-f_L} \\ \hat{X}_\theta &= \lfloor X_\theta 2^{f_\theta + 2}/\pi + 0.5 \rfloor 2^{-f_\theta}. \end{aligned} \tag{5}$$

We scale the angle by $4/\pi$ so that the quantization near the unit circle will be roughly the same in the angular and radial axes when $f_L = f_\theta$ while allowing the complete circle of 2π radians to be represented by a power of two. The rectangular value, $\tilde{X}$, represented by the quantized CLNS has the following real and imaginary parts:

$$\begin{aligned} \Re[\tilde{X}] &= b^{\hat{X}_L} \cos(\pi/4 \hat{X}_\theta) \\ \Im[\tilde{X}] &= b^{\hat{X}_L} \sin(\pi/4 \hat{X}_\theta). \end{aligned} \tag{6}$$

To summarize our notation, an arbitrary-precision complex value $\bar{X} \neq 0$ is transformed losslessly to its ideal CLNS representation, X. Quantizing X to $\hat{X}$ produces an absolute error perceived by the user in the rectangular system as $|\bar{X} - \tilde{X}|$.

Complex multiplication and division are trivial in CLNS. Given the exact CLNS representations, X and Y, the result of multiplication, $Z = X + Y$, can be computed by two parallel adders:

$$\begin{aligned} Z_L &= X_L + Y_L \\ Z_\theta &= (X_\theta + Y_\theta) \bmod 2\pi. \end{aligned} \tag{7}$$

The modular operation for the imaginary part is not strictly necessary, but reduces the range of angles that the hardware needs to accept. In the quantized system, this reduction comes at no cost in the underlying binary adder because of the $4/\pi$ scaling.

For example, if $\bar{X} = -1 + i$ and $\bar{Y} = 4i$, the $b = 2$ polar-logarithmic representations are $X_L = 0.5 \log_b(1^2 + 1^2) = 0.5, X_\theta = 3\pi/4$ and $Y_L = 0.5 \log_b(4^2) = 2.0, Y_\theta = \pi/2$. The representation of the product is computed simply as $Z_L = X_L + Y_L = 0.5 + 2.0 = 2.5$ and $Z_\theta = X_\theta + Y_\theta = 3\pi/4 + \pi/2 = 5\pi/4$. We can verify manually that this is correct: $\bar{Z} = b^{2.5}(\cos(5\pi/4) + i \sin(5\pi/4)) = 4\sqrt{2}(-\sqrt{2}/2 - \sqrt{2}/2i) = -4 - 4i$.

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

Division is like multiplication, except the representations are subtracted. For example, $\bar{Z} = \bar{X}/\bar{Y} = (-1 + i)/(4i) = 0.25 + 0.25i$ 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.

[1]Unless $b = e$, $\log_b$ in this paper is defined slightly differently than the typical definition $\ln(\bar{X})/\ln(b)$ used in complex analysis.

The difficult issue for all variations of LNS is addition and subtraction. Like conventional LNS, CLNS utilizes $\bar{X}+\bar{Y} = \bar{Y}(\bar{Z}+1)$, where $\bar{Z} = \bar{X}/\bar{Y}$. This is valid for all complex values, except when $\bar{X} = -\bar{Y}$, a case that can be handled specially, as in real LNS.

The complex-addition logarithm, $S_b(Z)$ is a function [16] that accepts a complex argument, Z, the log-polar representation of $\bar{Z}$, and returns the corresponding representation of $\bar{Z}+1$. Thus, to find the representation, T of the sum, $\bar{T} = \bar{X}+\bar{Y}$ simply involves

$$T = Y + S_b(X-Y), \tag{8}$$

which is implemented by two subtractors, a complex approximation unit, and two adders:

$$\begin{array}{rcl} T_L & = & Y_L + \Re[S_b(X-Y)] \\ T_\theta & = & (Y_\theta + \Im[S_b(X-Y)]) \bmod 2\pi. \end{array} \tag{9}$$

The major cost in this circuit is the complex S_b unit. Subtraction simply requires adding πi to Y before performing (8).

IV. Novel Addition Algorithm

The complex addition logarithm, $S_b(Z)$, has the same algebraic definition as its real-valued ($s_b(z)$) counterpart, and simple algebra on the complex $Z = Z_L + iZ_\theta$ reveals the underlying polar conversion required to increment a CLNS value by one:

$$\begin{array}{rcl} S_b(Z) & = & \log_b(1+b^Z) \\ & = & \log_b(1+b^{iZ_\theta+Z_L}) \\ & = & \log_b(1+(\cos(Z_\theta)+i\sin(Z_\theta))b^{Z_L}) \\ & = & \log_b(1+\cos(Z_\theta)b^{Z_L}+i\sin(Z_\theta)b^{Z_L}). \end{array} \tag{10}$$

This complex function can have its real and imaginary parts computed separately. In all of the prior CLNS literature [3], [4], [14], the real part[2] is derived from (10) as:

$$\begin{array}{rcl} \Re(S_b(Z)) & = & \log_b\left(\sqrt{(1+\cos(Z_\theta)b^{Z_L})^2+(\sin(Z_\theta)b^{Z_L})^2}\right) \\ & = & \frac{\log_b\left(1+2\cos(Z_\theta)b^{Z_L}+(\cos(Z_\theta)b^{Z_L})^2+(\sin(Z_\theta)b^{Z_L})^2\right)}{2} \\ & = & \frac{\log_b\left(1+2\cos(Z_\theta)b^{Z_L}+(\cos^2(Z_\theta)+\sin^2(Z_\theta))b^{2Z_L}\right)}{2} \\ & = & \frac{\log_b(1+2\cos(Z_\theta)b^{Z_L}+b^{2Z_L})}{2}, \end{array} \tag{11}$$

and the complex part is derived from (10) as:

$$\Im(S_b(Z)) = \arctan(1+\cos(Z_\theta)b^{Z_L}, \sin(Z_\theta)b^{Z_L}), \tag{12}$$

where $\arctan(x,y)$ is the four-quadrant function, like `atan2(y,x)` in many programming languages. To produce $\bar{X}+\bar{Y} = -1+5i$ given $\bar{X} = -1+i$ and $\bar{Y} = 4i$ similar to the earlier examples, the inputs are represented by $X = 0.5+3\pi/4i$, $Y = 2.0+\pi/2i$, and the quotient is represented by $Z = -1.5+\pi/4i$. Since $\sin(\pi/4) = \cos(\pi/4) = \sqrt{2}/2$,

$$\begin{array}{rcl} \Re[S_b(Z)] & = & 0.5\cdot\log_2(1+2\cdot 2^{-1.5}\sqrt{2}/2+2^{-3}) \\ & = & \log_2(\sqrt{26}/4) \approx 0.350 \\ \Im[S_b(Z)] & = & \arctan(1+b^{-1.5}\sqrt{2}/2, b^{-1.5}\sqrt{2}/2) \\ & = & \arctan(1.25, 0.25) \approx 0.197, \end{array} \tag{13}$$

[2][2] has a typo in which $\cos(Z_\theta)b^{Z_L}$ should have been $2\cos(Z_\theta)b^{Z_L}$.

thus from (9), the result is:

$$\begin{array}{rl} T = & (2.0+\pi/2i) + (\log_2(\sqrt{26})-2.0) \\ & + \arctan(1.25, 0.25)i) \\ = & \log_2(\sqrt{26}) + (\arctan(1.25, 0.25)) + \pi/2)i \\ \approx & 2.350220 + 1.768191i. \end{array} \tag{14}$$

This is correct since $\sqrt{26}\cdot\cos(T_\theta)+\sqrt{26}\cdot\sin(T_\theta)i \approx -1+5i$.

In the prior literature, the functions (11) and (12) are computed directly from the complex Z, either using fast, but costly, simple lookup [21], or less expensive methods like cotransformation [2], CORDIC [14] or content-addressable memory [22]. The proposed approach uses a different derivation in which the real and imaginary parts are manipulated separately. Unlike the prior literature, in the proposed technique the real values in the intermediate computations are represented as conventional LNS real values; the angles remain as scaled fixed-point values at every step. The advantages of the proposed technique are lower cost and the ability to reuse the same hardware for both real- and complex-LNS operations. In order to use this novel approach, a different derivation from (10) is required, while still considering this a single complex function of a complex variable:

$$\begin{array}{rcl} S_b(Z) & = & \log_b(1+b^Z) \\ & = & \log_b(1+\text{sgncos}(Z_\theta)b^{\text{logcos}(Z_\theta)}b^{Z_L} \\ & & +i\cdot\text{sgnsin}(Z_\theta)b^{\text{logsin}(Z_\theta)}b^{Z_L}), \end{array} \tag{15}$$

where $\text{sgncos}(Z_\theta)$ and $\text{sgnsin}(Z_\theta)$ are the signs of the respective real-valued trigonmetric functions and where $\text{logcos}(Z_\theta) = \log_b|\cos(Z_\theta)|$ and $\text{logsin}(Z_\theta) = \log_b|\sin(Z_\theta)|$ are real-valued functions compatible with a real-valued LNS ALU. For simplicity, (15) ignores the trivial cases, when Z_θ is a multiple of $\pi/2$, which cause $\sin(Z_\theta)$ or $\cos(Z_\theta)$ to be zero. These cases are resolved in Table I. In order to calculate the complex-valued function (15) using a conventional real LNS, there are two cases, ($-\pi/2 < Z_\theta < \pi/2$ and ($Z_\theta < -\pi/2$ or $Z_\theta > \pi/2$)), that choose whether the "1+" operation is implemented with the real s_b function or the real d_b function. Case 1 ($-\pi/2 < Z_\theta < \pi/2$) is when "1+" means a sum (s_b) is computed:

Case 1.

$$\begin{array}{rcl} S_b(Z) & = & \log_b((1+b^{\text{logcos}(Z_\theta)+Z_L}) \\ & & +i\cdot\text{sgnsin}(Z_\theta)b^{\text{logsin}(Z_\theta)+Z_L}) \\ & = & \log_b(b^{s_b(\text{logcos}(Z_\theta)+Z_L)} \\ & & +i\cdot\text{sgnsin}(Z_\theta)b^{\text{logsin}(Z_\theta)+Z_L}). \end{array} \tag{16}$$

Case 2 ($Z_\theta < -\pi/2$ or $Z_\theta > \pi/2$) is when "1+" means a difference (d_b) is computed:

Case 2.

$$\begin{array}{rcl} S_b(Z) & = & \log_b((1-b^{\text{logcos}(Z_\theta)+Z_L}) \\ & & +i\cdot\text{sgnsin}(Z_\theta)b^{\text{logsin}(Z_\theta)+Z_L}) \\ & = & \log_b(b^{d_b(\text{logcos}(Z_\theta)+Z_L)} \\ & & +i\cdot\text{sgnsin}(Z_\theta)b^{\text{logsin}(Z_\theta)+Z_L}). \end{array} \tag{17}$$

Up to this point, instead of giving separate real and imaginary parts, we have described a complex function equivalent to (10) in a novel way that eventually will be amenable to being

computed with a conventional real LNS ALU. Of course, to use such an ALU, it will be necessary to compute the real and imaginary parts separately. In Case 1 ($-\pi/2 < Z_\theta < \pi/2$), the real part of (16) can be described as:

$$\begin{aligned}
&\Re(S_b(Z)) \\
&\quad = \Re\left(\log_b \begin{pmatrix} b^{s_b(\mathrm{logcos}(Z_\theta)+Z_L)} \\ +i\cdot \mathrm{sgnsin}(Z_\theta) b^{\mathrm{logsin}(Z_\theta)+Z_L} \end{pmatrix}\right) \\
&\quad = \log_b\left(\sqrt{b^{2s_b(\mathrm{logcos}(Z_\theta)+Z_L)} + b^{2(\mathrm{logsin}(Z_\theta)+Z_L)}}\right) \\
&\quad = \tfrac{1}{2}\log_b(b^{2s_b(\mathrm{logcos}(Z_\theta)+Z_L)} + b^{2(\mathrm{logsin}(Z_\theta)+Z_L)}) \\
&\quad = \tfrac{1}{2}\log_b \begin{pmatrix} b^{2(\mathrm{logsin}(Z_\theta)+Z_L)} \\ +s_b(2(s_b(\mathrm{logcos}(Z_\theta)+Z_L)-(\mathrm{logsin}(Z_\theta)+Z_L))) \end{pmatrix} \\
&\quad = \mathrm{logsin}(Z_\theta) + Z_L + \\
&\qquad \tfrac{1}{2}s_b(2(s_b(\mathrm{logcos}(Z_\theta)+Z_L) - (\mathrm{logsin}(Z_\theta)+Z_L))).
\end{aligned} \tag{18}$$

There is a similar derivation from (17) for Case 2 ($Z_\theta < -\pi/2$ or $Z_\theta > \pi/2$):

$$\begin{aligned}
&\Re(S_b(Z)) \\
&\quad = \Re\left(\log_b \begin{pmatrix} b^{d_b(\mathrm{logcos}(Z_\theta)+Z_L)} \\ +i\cdot \mathrm{sgnsin}(Z_\theta) b^{\mathrm{logsin}(Z_\theta)+Z_L} \end{pmatrix}\right) \\
&\quad = \log_b\left(\sqrt{b^{2d_b(\mathrm{logcos}(Z_\theta)+Z_L)} + b^{2(\mathrm{logsin}(Z_\theta)+Z_L)}}\right) \\
&\quad = \tfrac{1}{2}\log_b(b^{2d_b(\mathrm{logcos}(Z_\theta)+Z_L)} + b^{2(\mathrm{logsin}(Z_\theta)+Z_L)}) \\
&\quad = \tfrac{1}{2}\log_b \begin{pmatrix} b^{2(\mathrm{logsin}(Z_\theta)+Z_L)} \\ +s_b(2(d_b(\mathrm{logcos}(Z_\theta)+Z_L)-(\mathrm{logsin}(Z_\theta)+Z_L))) \end{pmatrix} \\
&\quad = \mathrm{logsin}(Z_\theta) + Z_L + \\
&\qquad \tfrac{1}{2}s_b(2(d_b(\mathrm{logcos}(Z_\theta)+Z_L) - (\mathrm{logsin}(Z_\theta)+Z_L))).
\end{aligned} \tag{19}$$

The subexpression $-(\mathrm{logsin}(Z_\theta) + Z_L)$ in (18) and (19) becomes ∞ when $Z_\theta = 0$. This case, in fact, is trivially $\Re(S_b(Z)) = s_b(Z_L)$, which is equivalent to $\Re(S_b(Z)) = s_b(\mathrm{logcos}(Z_\theta) + Z_L)$ when $Z_\theta = 0$. Summarizing the cases for the real part in our novel method, we have:

$$\Re(S_b(Z)) = \begin{cases}
\mathrm{logsin}(Z_\theta) + Z_L + \\
\tfrac{1}{2}s_b(2(s_b(\mathrm{logcos}(Z_\theta)+Z_L) - (\mathrm{logsin}(Z_\theta)+Z_L))) \\
\qquad -\pi/2 < Z_\theta < 0 \text{ or } 0 < Z_\theta < \pi/2 \\
s_b(\mathrm{logcos}(Z_\theta)+Z_L), \qquad Z_\theta = 0 \\
\mathrm{logsin}(Z_\theta) + Z_L + \\
\tfrac{1}{2}s_b(2(d_b(\mathrm{logcos}(Z_\theta)+Z_L) - (\mathrm{logsin}(Z_\theta)+Z_L))) \\
\qquad Z_\theta < -\pi/2 \text{ or } Z_\theta > \pi/2.
\end{cases} \tag{20}$$

The imaginary part uses the four-quadrant arctangent. The computation of the four-quadrant arctangent [25] depends on the signs of its two arguments:

$$\arctan(x,y) = \begin{cases}
-\arctan(x,-y), & y < 0 \\
\pi - \arctan(-y/x), y \geq 0, & x < 0 \\
\arctan(y/x), y \geq 0, & x > 0 \\
\pi/2, & y > 0, x = 0.
\end{cases} \tag{21}$$

Eliminating the recursion in (21), and noting the value passed to the one-argument arctangent is now always non-negative, we have

$$\arctan(x,y) = \begin{cases}
-\pi + \arctan(|y|/|x|), & y < 0, x < 0 \\
-\arctan(|y|/|x|), & y < 0, x > 0 \\
\pi - \arctan(|y|/|x|), & y \geq 0, x < 0 \\
\arctan(|y|/|x|), & y \geq 0, x > 0 \\
-\pi/2, & y < 0, x = 0 \\
\pi/2, & y > 0, x = 0.
\end{cases} \tag{22}$$

It is obvious for the range of Z_θ involved that $y = \sin(Z_\theta) b^{Z_L} < 0$ when $Z_\theta < 0$. The condition when $x = 1 + \cos(Z_\theta) b^{Z_L} < 0$ is slightly more complicated, involving $\cos(Z_\theta) b^{Z_L} < -1$, a condition that can only happen when $\mathrm{sgncos}(Z_\theta) < 0$ (i.e., $Z_\theta < -\pi/2$ or $Z_\theta > \pi/2$) and $|\cos(Z_\theta) b^{Z_L}| > 1$. In LNS, the latter condition is equivalent to $\mathrm{logcos}(Z_\theta) + Z_L > 0$. The opposite condition $x > 0$ happens when the cosine is positive, ($-\pi/2 < Z_\theta < \pi/2$), or when $\cos(Z_\theta) b^{Z_L} > -1$. In LNS, the latter condition is equivalent to $\mathrm{logcos}(Z_\theta) + Z_L < 0$. Combining these conditions with (16), (17) and (22), we have:

$$\Im(S_b(Z)) = \begin{cases}
-\pi + \arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
Z_\theta < 0 \text{ and } (Z_\theta < -\pi/2 \text{ or } Z_\theta > \pi/2) \\
\qquad \text{and } \mathrm{logcos}(Z_\theta) + Z_L > 0 \\
-\arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\quad (Z_\theta < 0 \text{ and } (-\pi/2 < Z_\theta < \pi/2)) \\
\qquad \text{or } \mathrm{logcos}(Z_\theta) + Z_L < 0 \\
\pi - \arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
Z_\theta \geq 0 \text{ and } (Z_\theta < -\pi/2 \text{ or } Z_\theta > \pi/2) \\
\qquad \text{and } \mathrm{logcos}(Z_\theta) + Z_L > 0 \\
\arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\quad (Z_\theta \geq 0 \text{ and } (-\pi/2 < Z_\theta < \pi/2)) \\
\qquad \text{or } \mathrm{logcos}(Z_\theta) + Z_L < 0 \\
-\pi/2, \\
Z_\theta < 0 \text{ and } (Z_\theta < -\pi/2 \text{ or } Z_\theta > \pi/2) \\
\qquad \text{and } \mathrm{logcos}(Z_\theta) + Z_L = 0 \\
\pi/2, \\
\quad Z_\theta \geq 0 \text{ and } (Z_\theta < -\pi/2 | Z_\theta > \pi/2) \\
\qquad \text{and } \mathrm{logcos}(Z_\theta) + Z_L = 0\ .
\end{cases} \tag{23}$$

Simplifying the conditions, we have:

$$\Im(S_b(Z)) = \begin{cases}
-\pi + \arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\qquad Z_\theta < -\pi/2 \text{ and } \mathrm{logcos}(Z_\theta) + Z_L > 0 \\
-\arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\quad Z_\theta < 0 \text{ and } (Z_\theta > -\pi/2 \text{ or } \mathrm{logcos}(Z_\theta) + Z_L < 0) \\
\pi - \arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\qquad Z_\theta > \pi/2 \text{ and } \mathrm{logcos}(Z_\theta) + Z_L > 0 \\
\arctan\left(\frac{|\sin(Z_\theta) b^{Z_L}|}{|1+\cos(Z_\theta) b^{Z_L}|}\right), \\
\quad Z_\theta \geq 0 \text{ and } (Z_\theta < \pi/2 \text{ or } \mathrm{logcos}(Z_\theta) + Z_L < 0) \\
-\pi/2, \qquad Z_\theta < -\pi/2 \text{ and } \mathrm{logcos}(Z_\theta) + Z_L = 0 \\
\pi/2, \qquad Z_\theta > \pi/2 \text{ and } \mathrm{logcos}(Z_\theta) + Z_L = 0.
\end{cases} \tag{24}$$

The result produced for $|\sin(Z_\theta)b^{Z_L}|/|1+\cos(Z)b^{Z_L}|$ in LNS depends on whether s_b or d_b is required to produce $1+\cos(Z_\theta)b^{Z_L}$, which again depends on the sign of the cosine (either positive with $-\pi/2 < Z_\theta < \pi/2$ or negative with $Z_\theta < -\pi/2, Z_\theta > \pi/2$).

$$\log\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right) =$$

$$\begin{cases} -\left(s_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L)\right), \\ \qquad\qquad -\pi/2 < Z_\theta < \pi/2 \\ -\left(d_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L)\right), \\ \qquad\qquad Z_\theta < -\pi/2 \text{ or } Z_\theta > \pi/2. \end{cases} \tag{25}$$

Note that the intermediate expression in (25) is the negative of that used for the real part in the CLNS ALU derived above. In two cases, we need to expand the arctangent derivation into s_b and d_b subcases so that we can substitute the intermediate expression into the arctangent.

$$\Im(S_b(Z)) =$$
$$\begin{cases} -\pi + \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \quad Z_\theta < -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ -\arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta \le -\pi/2 \text{ and} \\ \quad (Z_\theta > -\pi/2 \text{ or } \text{logcos}(Z_\theta)+Z_L < 0) \\ -\arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad -\pi/2 < Z_\theta < 0 \text{ and} \\ \quad (Z_\theta > -\pi/2 \text{ or } \text{logcos}(Z_\theta)+Z_L < 0) \\ \pi - \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \quad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta \ge \pi/2 \text{ and} \\ \quad (Z_\theta < \pi/2 \text{ or } \text{logcos}(Z_\theta)+Z_L < 0) \\ \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad 0 < Z_\theta \le \pi \\ -\pi/2, \\ \quad Z_\theta < -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0 \\ \pi/2, \\ \quad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0. \end{cases} \tag{26}$$

Again eliminating impossible conditions,

$$\Im(S_b(Z)) =$$
$$\begin{cases} -\pi + \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta < -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ -\arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta \le -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L < 0 \\ -\arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad -\pi/2 < Z_\theta < 0 \\ \pi - \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad Z_\theta \ge \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L < 0 \\ \arctan\left(\frac{|\sin(Z_\theta)b^{Z_L}|}{|1+\cos(Z_\theta)b^{Z_L}|}\right), \\ \qquad 0 < Z_\theta \le \pi/2 \\ -\pi/2, \;\; Z_\theta < -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0 \\ \pi/2, \qquad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0. \end{cases} \tag{27}$$

There is a similar problem as in (20) about the singularity of $\text{logsin}(Z_\theta)$ in the case of $Z_\theta = 0$. Again, this is trivial since $\Im[S_b(Z)] = 0$ in this case. Taking this into account, we can substitute the LNS computation of the quotient into the arctangent:

$$\Im(S_b(Z)) =$$
$$\begin{cases} -\pi + \arctan(b^{-(d_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad Z_\theta \le -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ -\arctan(b^{-(d_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad Z_\theta \le -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L < 0 \\ -\arctan(b^{-(s_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad -\pi/2 < Z_\theta < 0 \\ 0, \qquad Z_\theta = 0 \\ \pi - \arctan(b^{-(d_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L > 0 \\ \arctan(b^{-(d_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L < 0 \\ \arctan(b^{-(s_b(\text{logcos}(Z_\theta)+Z_L)-(\text{logsin}(Z_\theta)+Z_L))}), \\ \qquad 0 < Z_\theta \le \pi/2 \\ -\pi/2, \qquad Z_\theta < -\pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0 \\ \pi/2, \qquad Z_\theta > \pi/2 \text{ and } \text{logcos}(Z_\theta)+Z_L = 0. \end{cases} \tag{28}$$

V. Implementation

Figure 1 shows a straightforward implementation of (20) and (28). The operation of the sign-logic unit is summarized in Table I. This logic operates in two modes: complex mode, which is the focus of this paper, and real mode (which uses the s_b/d_b unit to perform traditional LNS arithmetic). In real mode, $Z_\theta = 0$ is used to represent a positive sign, and $Z_\theta = -\pi$ is used to represent a negative sign. Given the angular quantization, the LNS sign bit corresponds to the most significant bit of $\hat{Z}_\theta$, with the other bits masked to zero.

In order to minimize the cost of CLNS hardware implementation, we exploit several techniques to transform the arguments to the function-approximation units into limited

TABLE I
TABLE OF REQUIRED CLNS ALU SIGN LOGIC.

Mode	Z_θ	$\text{logcos}(Z_\theta) + Z_L$	4Q correct	LNS op	Mux
Complex	$Z_\theta \le -\pi/2$	$\text{logcos}(Z_\theta) + Z_L > 0$	$-\pi + \arctan$	d_b	0
		$\text{logcos}(Z_\theta) + Z_L = 0$	$-\pi/2$	d_b	0
		$\text{logcos}(Z_\theta) + Z_L < 0$	$-\arctan$	d_b	0
	$-\pi/2 < Z_\theta < 0$		$-\arctan$	s_b	0
	$Z_\theta = 0$		0	s_b	1
	$0 < Z_\theta < \pi/2$		$\arctan$	s_b	0
	$Z_\theta \ge \pi/2$	$\text{logcos}(Z_\theta) + Z_L > 0$	$\pi - \arctan$	d_b	0
		$\text{logcos}(Z_\theta) + Z_L = 0$	$\pi/2$	d_b	0
		$\text{logcos}(Z_\theta) + Z_L < 0$	$\arctan$	d_b	0
Real	$Z_\theta = 0$		0	s_b	1
	$Z_\theta \neq 0$	$\text{logcos}(Z_\theta) + Z_L < 0$	0	d_b	1
		$\text{logcos}(Z_\theta) + Z_L > 0$	$-\pi$	d_b	1

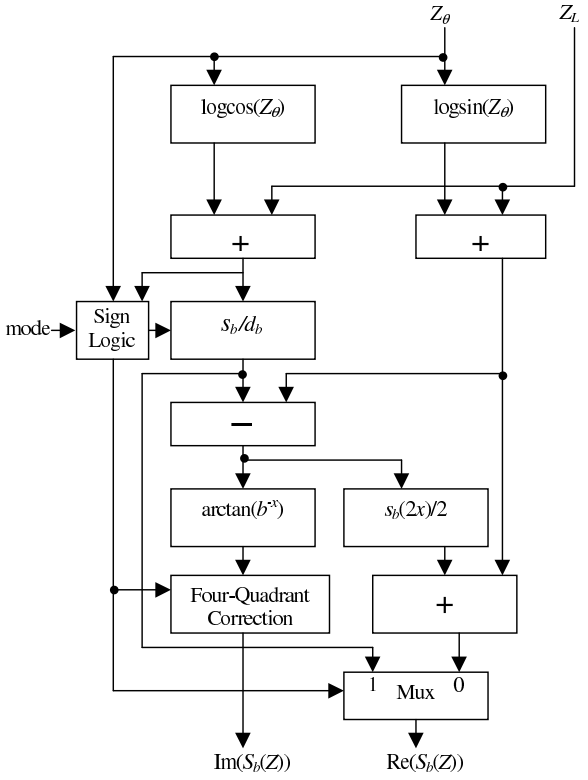

Fig. 1. CLNS ALU with Complex and Real Modes.

ranges where approximation is affordable. Assuming $b = 2$, the reduction for the addition logarithm is typical of real-LNS implementations:

$$s_2(x) = \begin{cases} 0, x < -2^{w_{ez}} \\ \log_2(1 + 2^x), -2^{w_{ez}} < x \le 0 \\ x + s_2(-x), 0 < x < 2^{w_{ez}} \\ x, x \ge 2^{w_{ez}}, \end{cases} \tag{29}$$

where $w_{ez} \le k_L$ describes the wordsize needed to reach the "essential zero" [18], which is the least negative value that causes $\log_2(1 + 2^x)$ to be quantized to zero. We use similar range reduction for the subtraction logarithm when z is far from zero, together with cotransformation [23] for z near zero. The scaled arctangent is nearly equal to s_2 for negative arguments (only because we choose $b = 2$ and $4/\pi$ scaling), although the treatment of positive arguments reflects the asymptotic nature of the arctangent:

$$a_2(x) = \begin{cases} 0, x < -2^{w_{ez}} \\ 4/\pi \arctan(2^x), -2^{w_{ez}} < x \le 0 \\ 2 - a_2(-x), 0 < x < 2^{w_{ez}} \\ 2, x \ge 2^{w_{ez}}. \end{cases} \tag{30}$$

The range reduction for the trigonmetric functions involves one case for each octant:

$$c_2(x) = \begin{cases} c_2(-x), x < 0 \\ \text{logcos}(\pi/4x), 0 \le x < 1 \\ d_2(2 \cdot c_2(2 - x)), 1 \le x < 2 \\ d_2(2 \cdot c_2(x - 2)), 2 \le x < 3 \\ c_2(4 - x), 3 \le x < 4 \\ c_2(x - 8), x \ge 4. \end{cases} \tag{31}$$

We can reuse (31) to yield $\text{logsin}(\pi/4x) = c_2(x + 2)$ as well as $\text{logcos}(\pi/4x) = c_2(x)$.

VI. DIRECT FUNCTION LOOKUP

There are several possible hardware realizations for (20) and (28). When f_L and f_θ are small, it is possible to use a direct table lookup for $\hat{s}_b$, $\hat{d}_b$, $\hat{a}_b$ and $\hat{c}_b$. This implementation allows round-to-nearest results for each functional unit, reducing roundoff errors in (20) and (28). Figure 2 shows the errors ($\sigma \approx 0.05$) in computing the real part (20) as a function of Z_L and Z_θ with such round-to-nearest tables when $f_L = f_\theta = 7$. Figure 3 shows errors ($\sigma \approx 0.002$) for the imaginary part (28). The errors in these and later figures spike dramatically near $Z_L = 0$, $Z_\theta = \pm\pi$ because of the inherent singularities in $S_b(Z)$ at these points.

The memory requirements for each of the $\hat{s}_b$, $\hat{d}_b$, $\hat{a}_b$ direct-lookup tables (with a $\hat{Z}_L$ input) is $2^{w_{ez}+f_L}$ words, assuming reduction rules like (29) and (30), which allow the interval spanned by these tables to be as small as $(-2^{w_{ez}}, 0)$. The trigonometric tables, with $\hat{Z}_\theta$ inputs, require 2^{2+f_θ} words because $-2 < \hat{Z}_\theta < 2$ (equivalent to $-\pi < Z_\theta < \pi$) and two integer bits are sufficient to cover this dynamic range. Instead of the more involved (31) rule used in later sections, $c_b(x) = c_b(-x)$ suffices for direct lookup.

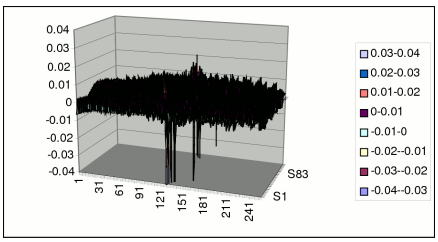

Fig. 2. Error in $\Re[S_b(Z)]$ using Direct Lookup with $f_L = f_\theta = 7$.

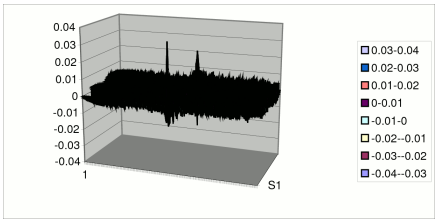

Fig. 3. Error in $\Im[S_b(Z)]$ using Direct Lookup with $f_L = f_\theta = 7$.

Since there are two instances of trigonometric tables and four instances of other tables in Figure 1, the total number of words required for naïve direct lookup is $2^{f+3}(2^{w_{ez}-1}+1)$ assuming $f = f_L = f_\theta$. Table II shows the number of words required by prior 2D-direct lookup methods [4], [22] compared to our proposed 1D-direct lookup implementation of (20) and (28). Savings occur for $f \geq 5$, and grow exponentially more beneficial as f increases.

For $f \geq 8$, the cost of direct implementation grows exponentially, although at a slower rate than the prior methods. One approach to reduce table size is to use interpolation on a smaller table. The cost of a simple linear interpolation unit is related to the size of the region of the interpolation and to the absolute maximum value of the second derivative in that region. Table III gives the range and the derivatives for the required functions. The d_b function is not considered as it may be computed via cotransformation [23]. It is clear that logsin has a singularity similar to d_b, which is the reason we avoid evaluating it directly and instead have chosen to use the relationship $\cos(x - \pi/2) = \sin(x) = \sqrt{1-\cos^2(x)}$ in (31). Because of the quantization of $\hat{c}_2$, this approximation is imperfect very near the singularity, but appears to deal with most cases adequately.

The maximum value (≈ 1.7) of the second derivative for logcos in its restricted range is large; however, the fact the range is restricted to $0 \leq x \leq 1$ mitigates this. It is well known [13] that the absolute maximum value (≈ 0.17) of the second derivative of s_b is modest; however, this table needs to cover a much larger range than logcos. The absolute maximum value (≈ 0.15) of the arctangent-exponential function is about the same as s_2 and its range is equally large. These values predict that interpolation from a uniformly-spaced table will be the best choice for c_b, but would require significant memory for s_b and a_b. Instead, it would be more effective to optimize the s_b and a_b approximations into subdomains, as explained in the next section. Such subdomains are effective for s_b and a_b (because their absolute second derivatives span many binades from 0.0 to over 0.1), but are not effective for c_b (because its absolute second derivative only varies one binade from $\pi^2/(16\ln(b))$ to $\pi^2/(8\ln(b))$).

TABLE II
MEMORY FOR PRIOR AND NOVEL DIRECT METHODS.

f	[4]	[22]	Novel	Saving
4	6886	597	640	-7%
5	27862	2916	1280	56%
6	127742	20203	2560	87%
7	566761	115249	5120	95%
8	2512012	604578	10240	98%

VII. Optimized Function-Unit Synthesis

The novel CLNS algorithm requires several special-function units, which should be optimized to minimize the cost of the units, especially as the precision, f, increases. These units were generated using enhanced versions of existing tools to simplify generating and optimizing synthesizable VHDL for these units.

A. FPLibrary

FPLibrary is an implementation of floating-point and LNS operators for FPGAs developed by Detrey and De Dinechin [11]. It provides addition, multiplication, division and square-root operators for both systems, with a user-defined precision, scaling up to IEEE single precision. LNS addition and subtraction are implemented using multipartite tables, generated in VHDL for every combination of parameters. It was later extended by Vouzis et al. to perform the LNS subtraction using cotransformation [23].

B. HOTBM

A method for function evaluation using tables and polynomial interpolations called HOTBM was developed by Detrey and De Dinechin [12]. The input domain of the function to be evaluated is split into regular intervals, and the function is approximated by a minimax polynomial inside each interval. Polynomials are evaluated as a sum of terms, each term being computed either by a powering unit and a multiplier or a table. The size of each component (table, multiplier and powering unit) is tuned to balance the rounding error and avoid unnecessary computations. As an implementation of this method, the authors also provide a tool written in C++ which generates hardware function evaluators in synthesizable VHDL. An exhaustive search in the parameter space is performed during generation to estimate the most area- and delay-efficient design.

C. FloPoCo

FloPoCo, for Floating-Point Cores, is intended to become a superset of both FPLibrary and HOTBM. Like HOTBM, it consists of a C++ program generating VHDL code. In addition to the usual floating-point and fixed-point arithmetic

TABLE III
TABLE OF FUNCTIONS AND DERIVATIVES.

function	1st derivative	2nd derivative	Range	|Max 2nd|
$s_b(x)$	$\frac{b^x}{b^x+1}$	$\ln(b)\frac{b^x}{(b^x+1)^2}$	$-w_{ez} < x \leq 0$	$\ln(b)/4 \approx 0.173$
$\mathrm{logsin}(\pi/4x)$	$\frac{\pi}{4\ln(b)}\cot(\frac{\pi}{4}x)$	$\frac{-\pi^2}{16\ln(b)\sin(\frac{\pi}{4}x)^2}$	$0 \leq x \leq 1.0$	∞
$\mathrm{logcos}(\pi/4x)$	$-\frac{\pi}{4\ln(b)}\tan(\frac{\pi}{4}x)$	$\frac{-\pi^2}{16\ln(b)\cos(\frac{\pi}{4}x)^2}$	$0 \leq x \leq 1.0$	$\frac{\pi^2}{8\ln(b)} \approx 1.78$
$4/\pi \arctan(b^x)$	$\frac{4\ln(b)}{\pi}\frac{b^x}{b^{2x}+1}$	$\frac{4b^x\ln(b)^2}{\pi(1+b^{2x})} - \frac{8b^{3x}\ln(b)^2}{\pi(1+b^{2x})^2}$	$-w_{ez} < x \leq 0$	$\frac{\ln(b)^2}{\pi} \approx 0.153$

TABLE IV
PARAMETERS USED FOR SYNTHESIS.

Size		Param		Order		
k	f	t	j	$\hat{s}_b$	$\hat{a}_b$	$\hat{c}_b$
5	8	-8	5	1	1	1
5	10	-8	6	1	1	1
5	12	-8	7	1	1	1
5	14	-4	8	1	1	2
5	16	-4	9	1	1	1
5	18	-4	10	2	2	2
5	20	-4	11	2	2	2

operators, it can generate more exotic operators, such as long accumulators [10] or constant multipliers [9]. It aims at taking advantage of FPGA flexibility in order to increase efficiency and accuracy compared to a naïve translation of a software algorithm to an FPGA circuit. Although it includes an implementation of HOTBM for fixed-point function evaluation and a whole range of floating-point operators, the LNS part of FPLibrary was not ported to FloPoCo by its developers.

D. A CLNS coprocessor

We extended the FloPoCo library with real LNS arithmetic. To perform the LNS addition, the input domain of s_b is split into three subdomains : $[-2^{w_{ez}}, -8[$, $[-8, -4[$ and $[-4, 0[$, where w_{ez} is such that s_b evaluates to zero in $]-\infty, -2^{w_{ez}}[$. This is done to account for the exponential nature of $s_b(z)$ for smaller values z. The same partitioning was used in FPLibrary. We then use an HOTBM operator to evaluate s_b inside each interval. This partitioning scheme allows significant area improvements compared to a regular partitioning as performed by HOTBM.

Likewise, the domain of d_b is split into intervals. The lower range $[-2^{w_{ez}}, t[$ is evaluated using several HOTBM operators, just like s_b. As we get closer to the singularity of d_b near 0, the function becomes harder to evaluate using polynomial approximations. The upper range $[t, 0[$ is evaluated using cotransformation. This approach is similar to the one used by Vouzis et al. in FPLibrary [23].

We used this extended FloPoCo library to generate the VHDL code for the functional units in a CLNS ALU at various precisions. One component was generated for each function involved in the CLNS addition. Real LNS functions s_b and d_b are generated by the extended FloPoCo. The function $\mathrm{logcos}(\pi/4x)$ is implemented as a HOTBM component, and $4/\pi \arctan(2^x)$ is evaluated using several HOTBM components with the same input domain decomposition as s_b.

Given the complexity of HOTBM and the influence of synthesizer optimizations, it is difficult to estimate precisely the best value of the d_b threshold t, cotransformation parameter j, and HOTBM order using simple formulas. We synthesized the individual units for a Virtex-4 LX25 with Xilinx ISE 10.1 using various parameters. For each precision tested, the combination of parameters yielding the smallest area on this configuration was determined. Table IV sums up the parameters obtained. We plan to automate this search in the future by using heuristics relying on the latency/area estimation framework integrated in FloPoCo.

E. Time/Area Tradeoff

The synthesis results are listed in detail in Tables V and VI, where α_s, $\alpha_{s/d}$, α_c and α_a are the areas of the respective units, and τ_s, $\tau_{s/d}$, τ_c and τ_a are the corresponding delays. One implementation option would be to follow the combinational architecture in Figure 1, in which case the area is roughly $\alpha_s + 2\alpha_{s/d} + \alpha_c + \alpha_a$ because one d_b together with a_b implement both logcos and logsin. (We neglect the area of four fixed-point adders, a mux and the control logic.) The delay is $2\tau_{s/d} + \tau_c + \max(\tau_a, \tau_s)$.

An alternative, which saves area ($\alpha_{s/d} + \alpha_c + \alpha_a$), uses three cycles to complete the computation (so that the same hybrid unit may be reused for all three s_b/d_b usages). The total area is less than double the area $\alpha_{s/d}$ of a standalone real LNS unit. The delay is roughly $3\tau_{s/d}$, since the hybrid unit has the longest delay. Despite using only about half the area of the combinational alternative, the three-cycle approach produces the result in about the same time as the combinational option.

It is hard to make direct comparisons, but the CORDIC implementation of CLNS reported in [14] uses ten stages. Some of those stages have relatively large multipliers, similar to what is generated for our circuit by FloPoCo. If such CORDIC stages have delay longer than 0.3 of our proposed clock cycle, our approach would be as fast or faster than [14].

Figure 4 shows the errors ($\sigma \approx 0.05$) in computing the real part (20) as a function of Z_L and Z_θ using the $f = 7$ function units generated by FloPoCo. This is roughly similar to the errors when computing the direct-lookup with the round-to-nearest method shown in Figure 2. In contrast, Figure 5 shows errors ($\sigma \approx 0.007$) for the imaginary part (28), which is significantly more than the errors for direct-lookup round-to-nearest shown in Figure 3, especially for $z_L > 0$. To determine the cause of this, simulations were run in which each of the function units generated by FloPoCo were replaced

TABLE V
AREA OF FUNCTION APPROXIMATION UNITS (SLICES) ON XILINX VIRTEX-4.

k	f	α_s	$\alpha_{s/d}$	α_c	α_a	$\alpha_s + 2\alpha_{s/d} + \alpha_c + \alpha_a$	$\alpha_{s/d} + \alpha_c + \alpha_a$
5	8	62	130	23	49	394	202
5	10	117	227	55	125	751	407
5	12	195	436	86	173	1326	695
5	14	364	854	133	329	2534	1316
5	16	559	1384	228	560	4115	2172
5	18	941	2418	300	752	6829	3470
5	20	1295	4230	398	1100	11253	5728

TABLE VI
LATENCY OF FUNCTION APPROXIMATION UNITS (NS) ON XILINX VIRTEX-4.

k	f	τ_s	$\tau_{s/d}$	τ_c	τ_a	$2\tau_{s/d}+\tau_c + \max(\tau_a,\tau_s)$	$3\tau_{s/d}$
5	8	6.969	16.817	5.556	6.843	46.159	50.451
5	10	12.161	24.021	6.78	12.563	67.385	72.063
5	12	14.171	25.86	11.846	14.38	77.946	77.58
5	14	16.55	31.045	13.04	16.689	91.819	93.135
5	16	18.4	33.231	16.662	18.947	102.071	99.693
5	18	23.975	36.652	16.795	20.244	114.074	109.956
5	20	23.233	39.312	18.124	19.935	119.981	117.936

with round-to-nearest units. When both logsin and logcos are computed with round-to-nearest, the imaginary result, shown in Figure 6, has smaller errors ($\sigma \approx 0.004$) which are much closer to those of Figure 3. Our conjecture is that using $\sin(x) = \sqrt{1-\cos^2(x)}$ contributes to the extra errors in Figure 5, which are then magnified by the slightly larger roundoff errors from all the other FloPoCo units.

VIII. LOW-ACCURACY METHOD

Some complex-number applications, like OFDM [21], may produce acceptable results even using low-accuracy techniques, i.e., where there is a large systematic error in the result due to the method even if performed with large word sizes. Mitchell [15] described two techniques for very-low-accuracy logarithmic arithmetic: one obtains the logarithm, $x = \log_2(\bar{x})$, and the other obtains the antilogarithm, 2^x. Mitchell approximates the logarithm by determining the position of the leading significant bit in $\bar{x}$ and shifting a corresponding amount:

$$m(\bar{x}) = \begin{cases} 2\bar{x} - 2, & 0.5 \le \bar{x} \le 1.0 \\ 4\bar{x} - 3, & 0.25 \le \bar{x} \le 0.5 \\ \ldots & \\ 2^n \cdot \bar{x} - n + 1, & 2^{-n} \le \bar{x} \le 2^{-n+1}. \end{cases} \quad (32)$$

The case of $\bar{x} = 0$ is undefined since in that case it is impossible to find a leading one. For the inverse problem, Mitchell uses:

$$2^x \approx m^{-1}(x) = 2^{\text{int}(x)} \cdot (\text{frac}(x) + 1). \quad (33)$$

where $\text{int}(x)$ is the integer part and $\text{frac}(x)$ is the fractional part. Arnold [5] suggested that Mitchell's method can be used as a low-accuracy (around four bits), but not necessarily low-precision (i.e., not necessarily performed with small word sizes since the cost of (32) and (33) only increase linearly with word size), approximation for the addition and subtraction logarithms:

$$s_2(x) \approx \begin{cases} m^{-1}(-x), x \le 0 \\ x + m^{-1}(x), x > 0 \end{cases} \quad (34)$$

$$d_2(x) \approx \begin{cases} -m^{-1}(-x+1), x \le -1 \\ x + m(-x), -1 < x < 0 \\ m(x), 0 < x < 1 \\ x + m^{-1}(x-1), x \ge 1. \end{cases} \quad (35)$$

The d function is undefined at $\bar{x} = 0$. Because the scaled arctangent is highly similar in value to $s_2(x)$, it is reasonable to use a similar approximation:

$$a_2(x) \approx \begin{cases} m^{-1}(x), x \le 0 \\ 2 - m^{-1}(-x), x > 0. \end{cases} \quad (36)$$

For the remaining functions, we can use range reduction like (31) together with an approximation like $c_2(x) \approx m^{-1}(2 \cdot m(|x|) - 89/128)/2$ for $-0.5 < x < 0.5$, where the $-89/128$ is an approximation to $2\log_2(\pi/2)$. Because Mitchell's method can be implemented without tables, it is possible to perform these operations with very minimal hardware. For example, [7] reports $f = 8$ serial-arithmetic implementations of Mitchell's method using about twenty CLBs.

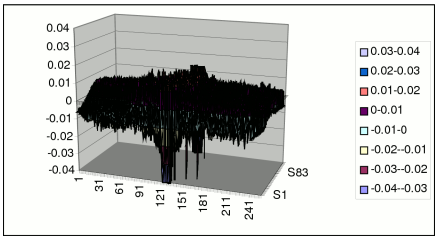

Fig. 4. Error in $\Re[S_b(Z)]$ using FloPoCo with $f_L = f_\theta = 7$.

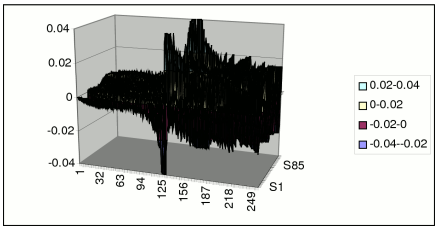

Fig. 5. Error in $\Im[S_b(Z)]$ using FloPoCo with $f_L = f_\theta = 7$.

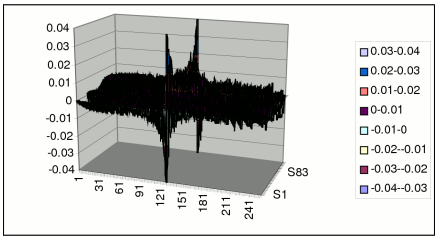

Fig. 6. Error in $\Im[S_b(Z)]$ using nearest sin/cos with $f_L = f_\theta = 7$.

IX. Conclusions

A new addition algorithm for complex log-polar addition was proposed that is based around an existing real-valued LNS ALU. The proposed design allows this ALU to continue to be used for real arithmetic, in addition to the special complex functionality described in this paper. The novel CLNS algorithm requires extra function units for log-trigonometric functions, which may have application beyond complex polar representation. Three implementation options for the novel special units were considered: medium-accuracy direct lookup, higher-accuracy interpolation (as generated by a tool called FloPoCo), and low-accuracy Mitchell's method. The errors resulting from the former two accurate alternatives were studied. (The suitability of the low-accuracy Mitchell's method for a given application requires extensive simulative study beyond the scope here.) As expected, round-to-nearest direct lookup tables give the lowest roundoff errors. We show that the errors in the FloCoPo implementation can be reduced by using a more accurate logsin unit than the one we proposed here based on Pythagorian calculations from logcos.

We compared these implementations of our novel CLNS algorithm to all known prior approaches. Our novel method has speed comparable to CORDIC, and uses orders-of-magnitude less area than prior 2D-table lookup approaches. As such, our approach provides a good compromise between speed and area. Considering that CLNS offers smaller bus widths than conventional rectangular representation of complex numbers, our proposed algorithm makes the use of this rather unusual number system in practical algorithms, like the OFDM, more feasible.

In the course of implementing the CLNS ALU, we uncovered and corrected several bugs in the HOTBM implementation of FloPoCo. We also made it easier to use by allowing the user to select the input range and scale of the target function instead of having to map its ranges manually to $[0, 1[\rightarrow [-1, 1[$. Hence, our work contributes to the maturity of the FloPoCo tool beyond the field of LNS.

X. Acknowledgments

We would like to thank the referees, especially referee 1, for the very helpful comments.

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," *13th IEEE Symposium on Computer Arithmetic*, Asilomar, California, pp. 190-199, July 1997.

[2] 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 Transactions on Computers*, vol. 47, no. 7, pp. 777-786, July 1998.

[3] M. G. Arnold, T. A. Bailey, J. R. Cowles, C. Walter, "Analysis of Complex LNS FFTs," *Signal Processing Systems SIPS 2001: Design and Implementation*, Francky Catthoor and Marc Moonen, Editors, IEEE Press, Antwerp, Belgium, pp. 58-69, 26-28 September 2001.

[4] M. Arnold, T. Bailey, J. Cowles and C. Walter, "Fast Fourier Transforms using the Complex Logarithm Number System," *Journal of VLSI Signal Processing*, Springer, vol 33, pp. 325-335, 2003.

[5] M. G. Arnold, "LPVIP: A Low-Power ROM-less ALU for Low-Precision LNS," *14th International Workshop on Power and Timing Modeling, Optimization, Simulation*, Santorini, Greece, LNCS 3254, pp. 675-684, 15-17 Sept. 2004.

[6] M. G. Arnold, "Approximating Trigonometric Functions with the Laws of Sines and Cosines Using the Logarithmic Number System," *8th EuroMicro Symposium on Digital Systems Design*, pp. 48-53, Porto, Portugal, 30 Aug. - 3 Sept. 2005.

[7] M. Arnold, P. Vouzis, "A Serial Logarithmic Number System ALU," *10th EuroMicro Conference on Digital System Design*, pp. 151-154, Lübeck, Germany, 27-31 August, 2007.

[8] N. Burgess, "Scaled and Unscaled Residue Number System to Binary Conversion Techniques Using the Residue Number System," *13th IEEE Symposium on Computer Arithmetic*, Asilomar, CA, pp. 250–257, Aug. 1997.

[9] N. Brisebarre, F. de Dinechin, and J. M. Muller, "Integer and Floating-Point Constant Multipliers for FPGAs," *Application-specific Systems, Architectures and Processors*, IEEE, pp. 239-244, 2008.

[10] F. de Dinechin, B. Pasca, O. Creţ and R. Tudoran, "An FPGA-specific Approach to Floating-Point Accumulation and Sum-of-Products," *Field-Programmable Technology*, IEEE, pp. 33-40, 2008.

[11] J. Detrey and F. de Dinechin, "A Tool For Unbiased Comparison Between Logarithmic and Floating-Point Arithmetic," *Journal of VLSI Signal Processing*, Springer, vol. 49, no. 1, pp. 161–175, 2007.

[12] J. Detrey, F. de Dinechin, "Table-Based Polynomials for Fast Hardware Function Evaluation," *16th International Conference on Application-specific Systems, Architectures and Processors*, IEEE, Samos, Greece, pp. 328-333, July 2005.

[13] D. M. Lewis, "An Architecture for Addition and Subtraction of Long Word Length Numbers in the Logarithmic Number System," *IEEE Transactions on Computers*, vol. 39, pp. 1325-1336, Nov. 1990.

[14] D. M. Lewis, "Complex Logarithmic Number System Arithmetic Using High Radix Redundant CORDIC Algorithms," *14th IEEE Symposium on Computer Arithmetic*, Adelaide, Australia, pp. 194-203, 14–16 April 1999.

[15] J. N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IEEE Transactions on Electronic Computers*, vol. EC-11, pp. 512-517, August 1962.

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

[17] R. Muscedere, V. S. Dimitrov, G. A. Jullien, and W. C. Miller, "Efficient Conversion from Binary to Multi-Digit Multidimensional Logarithmic Number Systems Using Arrays of Range Addressable Look-Up Tables," *13th International Conference on Application-specific Systems, Architectures and Processors*, San Jose, pp. 130-138, July 2002.

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

[19] E. E. Swartzlander, et al., "Sign/Logarithm Arithmetic for FFT Implementation," *IEEE Transactions on Computers*, vol. C-32, pp. 526–534, 1983.

[20] P. R. Turner, "Complex SLI Arithmetic: Representation, Algorithms and Analysis," *11th Symposium on Computer Arithmetic,* Windsor, Ontario, IEEE Computer Society Press, pp. 18-25, 29 June-4 July, 1993.

[21] P. Vouzis, M. G. Arnold and V. Paliouras, "Using CLNS for FFTs in OFDM Demodulation of UWB Receivers," *IEEE International Symposium on Circuits and Systems*, Kobe, Japan, pp. 3954-3957, 23-26 May 2005.

[22] P. Vouzis and M. G. Arnold, "A Parallel Search Algorithm for CLNS Addition Optimization," *IEEE International Symposium on Circuits and Systems*, Kos, Greece, 21-24 May 2006.

[23] P. Vouzis, S. Collange, M. Arnold, "Cotransformation Provides Area and Accuracy Improvement in an HDL Library for LNS Subtraction," *10th EuroMicro Conference on Digital System Design*, pp. 85-93, Lübeck, Germany, 27-31 August, 2007.

[24] P. Vouzis, S. Collange, M. Arnold, "LNS Subtraction Using Novel Cotransformation and/or Interpolation," *IEEE 18th International Conference on Application-specific Systems, Architectures and Processors*, pp. 107-114, Montreal, Quebec, Canada, 9-11 July, 2007.

[25] `http://www.wikipedia.org/wiki/arctangent`, cited 14 Oct, 2008.

[26] `http://www.xlnsresearch.com`.