

A Parallel Search Algorithm for CLNS Addition Optimization

Panagiotis D. Vouzis and Mark G. Arnold
 Computer Science and Engineering Dept.
 Lehigh University
 Bethlehem, PA 18015, USA
 Email: {vouzis, maab}@lehigh.edu

Abstract—We present analytical formulas for the calculation of the memory requirements for a system using the Complex Logarithmic Number System (CLNS). Certain properties of the CLNS addition algorithm allow replacement of a large memory by smaller multiple memories and combinational circuitry. The multiple memories combined with a parallel search algorithm offer memory savings up to 90% at the cost of extra hardware. However, depending on the particular case, there is a tradeoff between memory and combinational logic used.

I. INTRODUCTION

This work focuses on the reduction of the memory required for the addition in Complex Logarithmic Number System (CLNS) representation. CLNS was first introduced by Arnold et al. [1], and is a generalization of the Logarithmic Number System (LNS) [2]. A potential application of CLNS is the Fast Fourier Transform (FFT) since it requires the processing of complex numbers; thus the use of the CLNS is an attractive alternative, since it has been proven to be significantly more compact and requires fewer bits for the same accuracy than a comparable fixed-point representation [3]. In [4] the impact of the base b of the CLNS representation on the Signal-to-Noise Ratio (SNR) is investigated, and it is shown that there is an optimal base that offers increased SNR performance, with a reduction on the memory required for CLNS addition and on the dependence of the SNR on the input's standard-deviation variations.

Additionally, CLNS simplifies the operations of complex multiplication and division since they are converted to fixed-point addition and subtraction respectively. However, CLNS addition requires the tabulation of two 3D functions which can increase considerably the complexity of an FFT implementation. In this paper a tabulation technique is proposed, combined with the co-transformation in [1], that can mitigate this. The 3D functions are decomposed to several 2D ones, and a parallel search algorithm is incorporated using a technique somewhat similar to the Range Addressable Look-Up Table (RALUT) proposed in [5]. The proposed approach reduces considerably the total memory of the system at the cost of some small combinational circuitry without compromising the accuracy of the tabulation.

II. INTRODUCTION TO CLNS

CLNS represents a complex value, X , with a log-polar pair, $X_{CP} = (X_{CL}, X_{\theta})$, whose components are:

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

X_{CL} is quantized with k integral and l fractional bits in a two's complement representation. The quantization for X_{θ} is arbitrary. In this work, the step near the unit circle is the same in both angular and radial dimensions. This implies that $0 \leq X_{\theta} \leq 2\pi$ is represented with a $(3, l)$ format scaled by 2^{-l} , and the complete CLNS representation occupies $k + 2 \cdot l + 3$ bits.

On output, this log-polar pair is converted back to rectangular coordinates that describe the same value:

$$\Re[X] = b^{X_{CL}} \cos(X_{\theta}), \quad \Im[X] = b^{X_{CL}} \sin(X_{\theta}). \quad (2)$$

CLNS multiplication and division are easy, e.g., to divide X by Y :

$$Z_{CL} = X_{CL} - Y_{CL}, \quad Z_{\theta} = (X_{\theta} - Y_{\theta}) \bmod 2\pi. \quad (3)$$

CLNS addition uses $T = X + Y = Y(Z + 1)$, where $Z = X/Y$. This needs two subtractors, two ROMs, and two adders:

$$\begin{aligned} T_{CL} &= Y_{CL} + \Re[S_b(Z_{CP})] \\ T_{\theta} &= Y_{\theta} + \Im[S_b(Z_{CP})] \bmod 2\pi, \end{aligned} \quad (4)$$

where Z_{CP} is given by (3). $S_b(Z_{CP})$ implements i) conversion of Z_{CP} via (2); ii) incrementing this; and iii) conversion back via (1). Lewis [6] proposed a 32-bit $b = e$ CLNS ALU based on CORDIC rather than ROM lookup. $b = e$ simplifies Lewis' hardware since the CORDIC algorithm naturally gives the complex logarithm i) and antilogarithm iii) as base e . A different b would require scaling by $\ln(e)$. A higher-speed implementation, such as envisioned here, can use precomputed ROM to hold $S_b(Z_{CP})$ so b is arbitrary. Then $S_b(Z_{CP})$ is precomputed in terms of its real and imaginary parts:

$$\Re[S_b(Z_{CP})] = 0.5 \cdot \log_b(1 + 2b^{Z_{CL}} \cos(Z_{\theta}) + b^{2 \cdot Z_{CL}}) \quad (5)$$

$$\Im[S_b(Z_{CP})] = \arctan(1 + b^{Z_{CL}} \cos(Z_{\theta}), b^{Z_{CL}} \sin(Z_{\theta})). \quad (6)$$

Because of commutativity, we may assume $0 \leq Z_{\theta} \leq \pi$.

III. CLNS MEMORY REQUIREMENTS

The number of words required for the storage of a function depends on the size of the interval that its arguments need to be tabulated for, and on the accuracy of the representation used. The straightforward tabulation of the 3D functions (5)

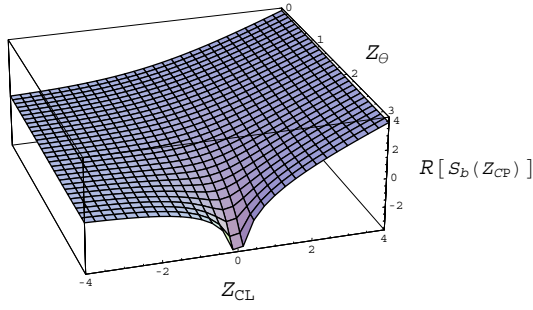


Fig. 1. The real part of $S_b(Z_{CP})$ for $b = 2$.

and (6) leads to large memory requirements due to the exponential dependence on the number of fractional bits of the representation. If $I_{Z_{CL}}$ is the interval of the Z_{CL} axis that needs to be spanned, either for the real or the imaginary part of $S_b(Z_{CP})$, then the number of words required is

$$w(l) = \frac{I_{Z_{CL}} \cdot \pi}{2^{-2l}}, \quad (7)$$

where l is the number of fractional bits of the representation [4]. A major reduction in memory requirements can be achieved by the technique of co-transformation presented in [1]. The functions (5) and (6) can be calculated according to

$$\Re[S_b(Z_{CP})] = F_b(Z_\theta) + \Re[S_b(T)] \quad (8)$$

$$\Im[S_b(Z_{CP})] = \frac{Z_\theta}{2} + \Im[S_b(T)], \quad (9)$$

where $F_b(Z_\theta) = \log_b(2 + 2 \cos(Z_\theta))/2$, and T is a transformed complex argument with

$$\Re(T) = D_b(Z_{CL}) - F_b(Z_\theta), \quad \Im(T) = \frac{Z_\theta}{2}, \quad (10)$$

and $D_b(Z_{CL}) = \log_b(|1 - b^{Z_{CL}}|)$ is the function used in the real logarithmic subtraction. The precondition requirement for this co-transformation is $Z_{CL} > 0$. Thus, all the values of the imaginary part of Z_{CP} are transferred to the interval $(0, \pi/2)$, with the extra cost of tabulating the functions $F_b(Z_\theta)$ and $D_b(Z_{CL})$. Using the notion of essential zero [7] for $D_b(Z_{CL})$, it holds that for $Z_{CL} > -\log_b(1 - b^{-2^{-l}})$ the value of the function is smaller than $2^{-l} \approx 0$, thus the number of words required is equal to

$$w_{D_b}(b, l) = \left\lceil -\frac{\log_b(1 - b^{-2^{-l}})}{2^{-l}} \right\rceil. \quad (11)$$

Similarly, for $F_b(Z_\theta)$ the number of words is

$$w_{F_b}(b, l) = \left\lceil \frac{\cos^{-1}\left(\frac{b^{2^{-l+1}} - 2}{2}\right)}{2^{-l}} \right\rceil. \quad (12)$$

Due to the finite precision of the CLNS representation, the required interval in which the function $S_b(Z_{CP})$ has to be stored is also finite. The memory requirements depend on the accuracy and on the value of the base b of the representation.

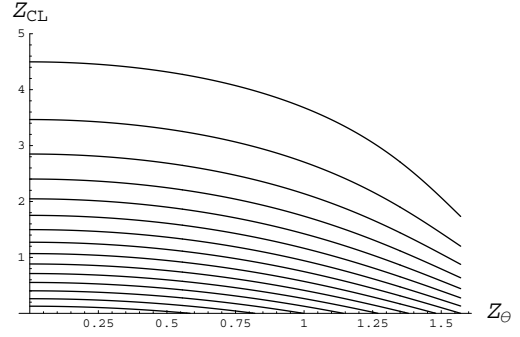


Fig. 2. All the boundary lines of $\Re[S_b(Z_{CP})] - Z_{CL}$, for $b = 2$ and $l = 4$.

A. The real part of $S_b(Z_{CP})$

It can be observed that $\lim_{Z_{CL} \rightarrow +\infty} \Re[S_b(Z_{CP})] = Z_{CL}$, thus, if the fractional bits of the two's complement representation are l , then it is sufficient to store only the values that satisfy the inequality

$$\Re[S_b(Z_{CP})] - Z_{CL} > 2^{-l}, \quad (13)$$

with $Z_{CL} > 0$ and $Z_\theta \in (0, \pi/2)$. For the rest of the values the function is a tautology, i.e., the result of the function is equal to the input argument Z_{CL} . Fig. 2 depicts not only the contour line that determines the effective zero of $\Re[S_b(Z_{CP})] - Z_{CL}$ (uppermost line), but also the contour lines where the function changes value at a step of 2^{-l} .

The calculation of the effective zero for $\Re[S_b(Z_{CP})] - Z_{CL}$ is one step towards the reduction of the memory requirements of CLNS addition, for only the number of necessary words to be stored is specified. Thus, if a particular 2-tuple (Z_{CL}, Z_θ) falls inside the area of the effective zero, then the value of the $\Re[S_b(Z_{CP})]$ is calculated as Z_{CL} . In this way a considerable number of expensive memory accesses can be saved.

In addition, the knowledge of the boundaries can be further exploited, i.e., instead of tabulating the whole 3D plane, only the lines need to be stored, and the evaluation of the function can be performed by determining the region of the 2D plane that the 2-tuple (Z_{CL}, Z_θ) falls in. This property allows the substitution of the 3D LUT by a Range Addressable LUT (RALUT) [5] depicted in Fig. 3. In contrast to simple RALUT used for real-valued MDLNS conversion in [5], Fig. 3 also includes m_{\Re} ROMs addressed by Z_θ that store contour lines, where for a particular input of Z_θ each ROM outputs a distinct value for a particular contour line. These values are then compared in parallel to Z_{CL} , and the region where the 2-tuple (Z_{CL}, Z_θ) falls in is specified, hence the corresponding quantity $\Re[S_b(Z_{CP})] - Z_{CL}$ can be produced in the form of a binary word by the m_{\Re} -to- $\log_2(m_{\Re})$ encoder. The output of the encoder is added to Z_{CL} and the final outcome is the actual value of $\Re[S_b(Z_{CP})]$. For the evaluation of the contents of the ROMs the analytic solution of (13) is required:

$$Z_{\Re,CL}(b, Z_\theta, l, m) = -\log_b\left(\sqrt{b^{m2^{-l+1}} - \sin^2(Z_\theta) - \cos(Z_\theta)}\right). \quad (14)$$

From Fig. 2 it can be observed that the spanned interval of the Z_θ axis is not the same for all the lines; thus, the maximum

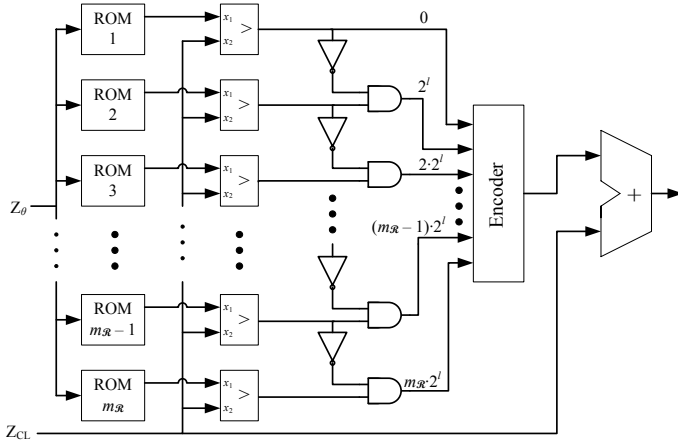


Fig. 3. The implementation of the parallel search algorithm.

value of Z_θ that needs to be tabulated for the m -th ROM is given by the solution of the equation

$$Z_{\mathfrak{R},\text{CL}}(b, Z_\theta, l, m) = 0 \Rightarrow Z_{\mathfrak{R},\theta}(b, l, m) = \cos^{-1}\left(\frac{b^{m2^{-l+1}} - 2}{2}\right). \quad (15)$$

However, in any case $Z_\theta < \pi/2$ must hold. Thus, if only the m -th boundary curve of $Z_{\mathfrak{R},\text{CL}}(b, Z_\theta, l, m)$ is to be tabulated, the number of words required is

$$w_{\mathfrak{R},2\text{D}}(b, l, m) = \left\lceil \frac{Z_{\mathfrak{R},\theta}(b, l, m)}{2^{-l}} \right\rceil. \quad (16)$$

Additionally, if the function above the m -th boundary line is to be tabulated in a 3D manner, the biggest value of Z_θ is given by (15), while the biggest value of Z_{CL} is given by (14) for $Z_\theta = 0$:

$$Z_{\mathfrak{R},\text{CL}}(b, l, m) = -\log_b(b^{m2^{-l}} - 1). \quad (17)$$

The number of words required for this tabulation is

$$w_{\mathfrak{R},3\text{D}}(b, l, m) = \left\lceil \frac{Z_{\mathfrak{R},\theta}(b, l, m) \cdot Z_{\mathfrak{R},\text{CL}}(b, l, m)}{2^{-2l}} \right\rceil. \quad (18)$$

An implementation for $\Re[S_b(Z_{\text{CP}})] - Z_{\text{CL}}$ that utilizes the tabulation of $m_{\mathfrak{R}}$ consecutive boundary lines of $Z_{\mathfrak{R},\text{CL}}(b, Z_{\text{CL}}, l, m)$ and a 3D tabulation of $\Re[S_b(Z_{\text{CP}})]$ delimited by the boundaries described requires

$$w_{\mathfrak{R}}(b, l, m_{\mathfrak{R}}) = \sum_{m=1}^{m_{\mathfrak{R}}} [w_{\mathfrak{R},2\text{D}}(b, l, m)] + w_{\mathfrak{R},3\text{D}}(b, l, m_{\mathfrak{R}} + 1) \quad (19)$$

words. $m_{\mathfrak{R}} = 0$ corresponds to the case that $\Re[S_b(Z_{\text{CP}})]$ is completely tabulated in a 3D manner (only one ROM and no boundary lines).

B. The imaginary part of $S_b(Z_{\text{CP}})$

The ideas used in the previous section to minimize the memory requirements for the real part of $S_b(Z_{\text{CP}})$ can be used for its imaginary counterpart. For $Z_{\text{CL}} > 0$ $\lim_{Z_{\text{CL}} \rightarrow +\infty} \Im[S_b(Z_{\text{CP}})] = Z_\theta^-$. For values of Z_{CL} greater than the effective zero the function $\Im[S_b(Z_{\text{CP}})]$ becomes a tautology since it produces as a result the value of the argument Z_θ .

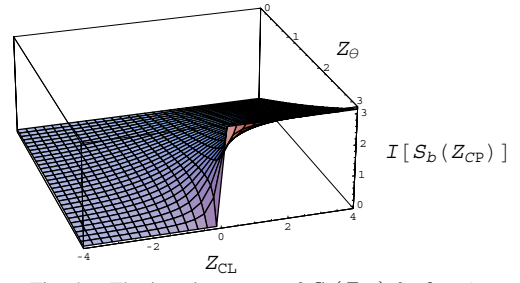


Fig. 4. The imaginary part of $S_b(Z_{\text{CP}})$ for $b = 2$.

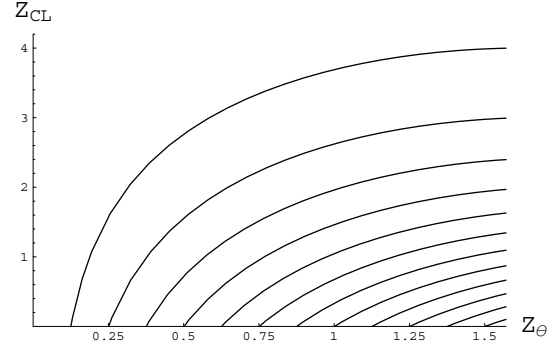


Fig. 5. All the boundary lines of $\Im[S_b(Z_{\text{CP}})] - Z_\theta$, for $b = 2$ and $l = 4$.

Fig. 5 depicts a number of contour lines of $\Im[S_b(Z_{\text{CP}})] - Z_\theta$, which are described by the formula

$$Z_{\Im,\text{CL}}(b, Z_\theta, l, m) = \log_b \left(\frac{\sin(Z_\theta - m2^{-l})}{\sin(m2^{-l})} \right). \quad (20)$$

In this case, the only modification needed to the circuit depicted in Fig. 3 is that to the output of the encoder Z_θ is added, and not Z_{CL} .

The tabulation of these contour lines begins from the point on Z_θ axis given by the solution of the equation

$$Z_{\Im,\text{CL}}(b, Z_\theta, l, m) = 0 \Rightarrow Z_{\Im,\theta}(l, m) = m2^{-l+1} \quad (21)$$

and ends at $\pi/2$. The number of words required in this case is

$$w_{\Im,2\text{D}}(l, m) = \left\lceil \frac{\pi/2 - Z_{\Im,\theta}(l, m)}{2^{-l}} \right\rceil. \quad (22)$$

For the remaining 3D tabulation the maximum value of Z_{CL} is achieved at $Z_\theta = \pi/2$. Since $Z_{\Im,\text{CL}}(b, \pi/2, l, m) = \log_b(\tan(m2^{-l}))$, the number of words is

$$w_{\Im,3\text{D}}(b, l, m) = \left\lceil \frac{(\pi/2 - Z_{\Im,\theta}(l, m)) \cdot Z_{\Im,\text{CL}}(b, \pi/2, l, m)}{2^{-2l}} \right\rceil. \quad (23)$$

Finally, for the tabulation of the $\Im[S_b(Z_{\text{CP}})]$ in a combination of m_{\Im} 2D ROMs and one 3D ROM,

$$w_{\Im}(b, l, m_{\Im}) = \sum_{m=1}^{m_{\Im}} [w_{\Im,2\text{D}}(l, m)] + w_{\Im,3\text{D}}(b, l, m_{\Im} + 1), \quad (24)$$

number of words is required.

Hence, the total number of words for the tabulation of the real and the imaginary part of $S_b(Z_{\text{CP}})$ after using the proposed

parallel search algorithm combined with co-transformation is given by the sum of (11), (12), (19), and (24):

$$w(b, l, m) = w_{D_b}(b, l) + w_F(b, l) + w_{\Re}(b, l, m) + w_{\Im}(b, l, m). \quad (25)$$

All this analysis enables the study of the memory requirements for the CLNS addition circuit, and can help in the design of the most efficient table size by exploring the impact of the parameters b, l, m .

IV. COMPARISON

The most straightforward approach for the tabulation of $S_b(Z_{CP})$ is to consider both positive and negative values of Z_{CL} , and $Z_\theta \in (0, \pi)$, as depicted in Fig. 1 and 4. The CLNS can be optimized by the proper choice of the base b for the implementation of the 128-point FFT. The optimization is performed with the criterion of making the highest possible SNR for the output. It is proven that a base bigger than the value $b = 2$ results in both increased SNR and reduced memory [4]. However, a pure 3D tabulation results in excessive memory size, which can be decreased considerably by the use of the parallel search algorithm proposed here.

The circuitry used for the implementation of this novel algorithm introduces an extra cost, which must be taken into account. From the architecture depicted in Fig. 3, it is apparent that the cost of the extra circuitry increases linearly with the number, m , of ROMs used. The memory savings though, follow a lower pace; i.e., a doubling of the number of ROMs used does not double the memory savings. Thus, a reasonable approach is to use a number of 2D ROMs that decrease the memory size to a desired level, and then tabulate the remaining function in a 3D scheme. Although full elimination of the 3D ROMs is possible in every case, resulting in the maximum possible memory savings, for the cases of increased precision, l , the implementation of the parallel search algorithm logic becomes sizeable. The best choice for the number of 2D ROMs depends on the memory and logic resources available for a particular implementation.

Table I presents the breakdown of the implementation of the parallel search algorithm, together with the memory savings compared to [4]. Different choices for the number of boundary lines to be tabulated are presented. (Although the tabulation scheme in [4] uses both positive and negative Z_{CL} , for the purposes of comparison we assume that only positive values of Z_{CL} are used for both approaches.) Thus, the memory savings are due to the use of co-transformation, and the new organization of the ROMs presented here. The number of the LUTs required for each case are taken after synthesis of the implementation of the parallel search algorithm using as target technology a Spartan II Field Programmable Gate Array (FPGA) from Xilinx. Given that in such a device one LUT holds 32 bits, each LUT is equivalent to 3 or 4 words. It is apparent that the LUTs added by the proposed technique are relatively insignificant. For exhaustive exploration of all the possible choices of m , the number of its different values that have to be studied is 2^l . However, here for demonstration

TABLE I
MEMORY SAVINGS ACHIEVED BY THE USE OF THE PARALLEL SEARCH ALGORITHM, AND THE HARDWARE COST FOR ITS IMPLEMENTATION.

(k, l)	b	Words [4]	m	Words	LUTs	Sav.
(3, 4)	2.00	6886	15	597	137	91.33%
(3, 5)	2.28	27862	16	2916	156	89.53%
(3, 6)	2.35	127742	16	20203	192	84.18%
(3, 6)	2.35	127742	32	11160	232	91.26%
(3, 7)	2.44	566761	16	115249	205	79.67%
(3, 7)	2.44	566761	32	77717	423	86.29%
(3, 7)	2.44	566761	64	42060	884	92.58%
(3, 8)	2.50	2512012	16	604578	219	75.93%
(3, 8)	2.50	2512012	32	451999	459	82.01%
(3, 8)	2.50	2512012	64	302601	948	87.95%
(3, 8)	2.50	2512012	128	161628	2356	92.14%

the maximum value of m used is 128, which provides a memory saving of at least 75%. Only up to 11-bit word lengths are considered, because for longer words the memory size becomes impractical for a real implementation. The values of the bases b are the ones proposed in [4], apart from the case $(k, l) = (3, 4)$ with $b = 2$ which is only studied here.

V. CONCLUSIONS

An optimization technique is proposed for the implementation of CLNS addition. Since the main source of delay, area, and power consumption of such an implementation is the memory, a parallel search algorithm is proposed offering tradeoff between memory and combinational logic. Depending on the type of the available resources, more combinational logic and less memory can be used, or vice versa. These resources can be optimized for a particular implementation technology where the relative cost of memory and logic will determine the optimum tradeoff, as was discussed in the case of the Xilinx FPGA. Since [1] has proven low-precision FFTs using naïve CLNS takes no more area than equivalent precision FFTs using fixed point, the proposed CLNS parallel search algorithm should offer significant area savings for applications, like OFDM, where such FFTs are used.

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. on Computers*, vol. 47, pp. 777–786, July 1998.
- [2] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," *IEEE Trans. on Computers*, vol. 24, pp. 1238–1242, Dec. 1975.
- [3] M. Arnold, T. Bailey, J. Cowles, and C. Walter, "Fast Fourier Transform Using the Complex Logarithmic Number System," *Journal of VLSI Signal Processing*, vol. 33, no. 3, pp. 325–335, 2003.
- [4] P. Vouzis, M. G. Arnold, and V. Paliouras, "Using CLNS for FFTs in OFDM Demodulation of UWB Receivers," in *IEEE ISCAS'05*, (Kobe, Japan), pp. 3954–3957, May 2005.
- [5] 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," in *ASAP'02*, (San Jose, CA), pp. 130–138, July 2002.
- [6] D. M. Lewis, "Complex Logarithmic Number System Arithmetic Using High-Radix Redundant CORDIC Algorithms," in *14th IEEE Symposium on Computer Arithmetic*, (Adelaide, Australia), pp. 194–203, April 1999.
- [7] V. Paliouras, "Optimization of LNS Operations for Embedded Signal Processing Applications," in *IEEE ISCAS'02*, vol. 2, (Phoenix-Scottsdale, AZ), pp. 744–747, May 2002.