# Logarithmic Arithmetic as an Alternative to Floating-Point: A Review

**Manik Chugh** and **Behrooz Parhami**
Dept. Electrical & Computer Eng., Univ. of California
Santa Barbara, CA 93106-9560, USA
parhami@ece.ucsb.edu

ABSTRACT: *The logarithmic number system (LNS) has found appeal in digital arithmetic because it allows multiplication and division to be performed much faster and more accurately than with the widely used floating-point (FP) number formats. We review the sign/logarithmic number system and present a comparison of various techniques and architectures for performing arithmetic operations efficiently in LNS. As a case study, we describe the European logarithmic microprocessor, a device built in the framework of a research project launched in 1999. Comparison of the arithmetic performance of this microprocessor with that of a commercial superscalar pipelined FP processor leads to the conclusion that LNS can be successfully deployed in general-purpose systems.*

KEYWORDS: ALU design, computation errors, interpolation, instruction-set architecture, logarithmic number system, machine arithmetic, performance per watt, real arithmetic.

## 1. Introduction

Proposals for the logarithmic number system (LNS) began in 1971, when Kingsburg and Rayner [2] introduced "logarithmic arithmetic" for digital signal processing. A similar logarithmic number system was proposed by Swartzlander and Alexopoulos [3] in 1975. Instead of using 2's-complement format for logarithms, numbers were scaled to avoid negative logarithms.

The sign/logarithmic number system overcomes the slowness of multiplication and division with conventional weighted numbers, while also avoiding the problems inherent in a residue number system (RNS). This advantage, however, is offset by the fact that addition and subtraction operations require a fairly complex procedure to be applied to logarithms. Despite its attractive properties, until fairly recently, only a few implementations of LNS arithmetic were attempted, all of which were restricted to low-precision applications, the difficulty in performing addition and subtraction on long words being the principal reason. LNS addition and subtraction require lookup tables whose size grows exponentially (several times $2^l$ words), with logarithms that are of width $l$ bits. For this reason, implementations described in the early literature were limited to 8-12 bits of fractional precision [4], [5].

It was recognized early on that LNS can offer an advantage over floating-point (FP) representation only if LNS addition and subtraction can be performed with the speed and accuracy at least equal to those of FP. However, achieving this goal is complicated by the fact that these operations require the evaluation of nonlinear functions [6], [7], [8], [9].

Contemplating the development of LNS-based commercial microprocessors, Lewis et al. [10], Paliouras et al. [11], and Arnold [12] proposed architectures for LNS-based processors in the late 1990s and early 2000s, but did not present a finished design or extensive simulation results. At about the same time, a European project, initiated by Coleman et al. [13], [14], laid down the foundations for the development of such a commercial digital system, dubbed the European logarithmic microprocessor (ELM), which provided performance similar to commercial superscalar pipelined floating-point processors [15].

Modern computation-intensive applications, with their increased algorithmic complexities as well larger problem sizes and data sets, are becoming bounded by the speed of FP operations. Real-time applications in this class are exemplified by RLS-based algorithms, subspace methods required in broadcasting and cellular telephony, Kalman filtering, and Riccati-like equations for advanced real-time control. Graphics systems [16] provide another case in point.

The Gravity Pipe supercomputer (GRAPE), that won the Gordon Bell Prize in 1999, used LNS representation and arithmetic. LNS is commonly used as part of hidden Markov models, exemplified by the Viterbi algorithm, applied in speech recognition and DNA sequencing. The past two decades have seen substantial efforts to explore the applicability of LNS as a viable alternative to FP for general-purpose processing of single-precision real numbers, demonstrating LNS as an alternative to floating-point, with improved accuracy and speed.

Collecting pertinent references and information about ongoing efforts in one place has been a primary motivation for writing this paper.

## 2. Background and Terminology

The LNS representation of a number $x$ consists of the number's sign $S_x$ and the binary logarithm $L_x$ of its magnitude. LNS representation equivalent to the 32-bit (single precision) IEEE standard FP format [13] has a 31-bit logarithm part that forms a 2's-complement fixed-point value ranging from –128 to approximately +128. The real numbers represented are signed and have magnitudes ranging from $2^{-128}$ to $\sim 2^{+128}$ (i.e., from $2.9 \times 10^{-39}$ to $3.4 \times 10^{+38}$). The smallest representable positive value, $40000000_{16}$, is used as a special code for zero, while $C0000000_{16}$ is dedicated to represent NaN.

LNS's uniform geometric error characteristics across the entire range of values leads to roughly an additional 1/2 bit of precision compared with a FP representation using the same number of bits. Thus, in signal-processing applications, LNS offers better signal/noise ratio as well as a better dynamic range.

LNS multiplication and division are defined as:

$$\text{Multiplication: } S_p = S_x \oplus S_y, \quad L_p = L_x + L_y \qquad (1)$$
$$\text{Division: } S_q = S_x \oplus S_y, \quad L_q = L_x - L_y \qquad (2)$$

Given $x$ and $y$, with $|x| \geq |y|$, $z = x \pm y$ is computed as:

$$\begin{aligned} S_z &= S_x \\ L_z &= \log_2|x \pm y| = \log_2|x(1 \pm y/x)| \qquad (3) \\ &= \log_2|x| + \log_2|1 \pm y/x| \\ &= \log_2|x| + \log_2|1 \pm 2^{(L_y - L_x)}| \end{aligned}$$

Let $d = L_y - L_x \leq 0$ and $\phi^{\pm}(d) = \log_2|1 \pm 2^d|$. The value of $\phi^{\pm}(d)$, shown in Fig. 1, can be read out from a ROM, but this is infeasible for wide words. Schemes based on interpolation are discussed in Section 3. For now, we will assume that a ROM is sufficient for our purpose.

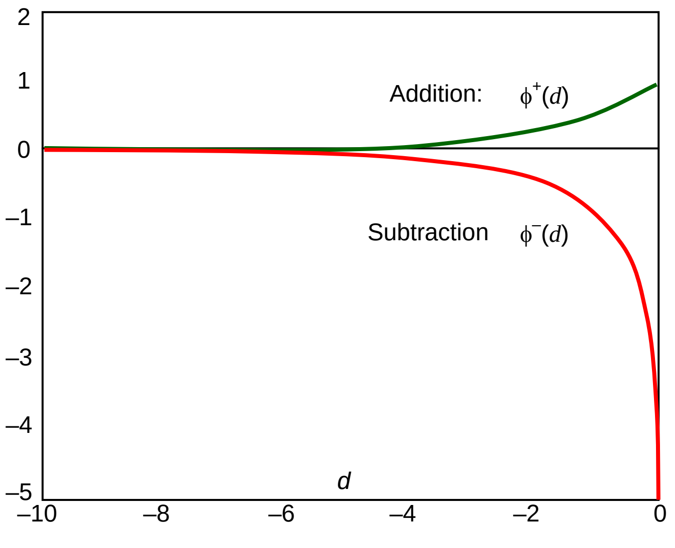

Fig. 1. Plots of $\phi^+(d)$ and $\phi^-(d)$ as functions of $d$.

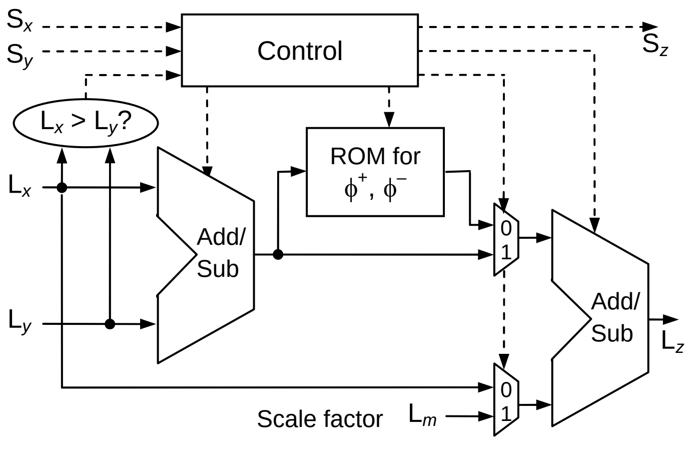

Fig. 2. A complete four-function ALU for LNS [21].

Realization of Eqns. (1), (2), and (3) by means of a comparator, an adder, a subtractor, a read-only memory (ROM) table, two multiplexers, and a small amount of peripheral logic can result in a simple four-function ALU [3], as shown in Fig. 2. The operation latency in this ALU is $T_{OP} = T_{COMP} + 2T_{ADD} + T_{ROM}$, where $T_{COMP}$ is the delay of a comparator. For convenience, we have assumed the peripheral logic delay to be negligible. In practice, the comparator may be implemented with a subtractor, leading to $T_{OP} = 3T_{ADD} + T_{ROM}$.

## 3. Addition and Subtraction

In this section, we focus on the efficient calculation of the nonlinear term $\phi^{\pm}(d) = \log_2|1 \pm 2^d|$, where $d = L_y - L_x$ (or $d = j - i$, taking $L_y$ as $j$ and $L_x$ as $i$ for simplicity). Implementation work began with Swartzlander and Alexopoulos's 1975 paper [3], with a 12-bit device [4], while a 1988 scheme extended the width to 20 bits [5]. Both designs were direct implementations of Eqn. (2), with a ROM covering all possible values of $\phi^{\pm}(d)$.

In the preceding simple scheme, table sizes increase exponentially with the word width, limiting its practical utility to about 20 bits. Lewis's 1991 design [6] extended the word width to 28 bits by implementing the lookup table for $d$ values only at intervals of $\Delta$. Any negative value of $d$ satisfies $d = -h\Delta - \delta$ for some integer value $h$, leading to the Taylor-series expansion of $F(d)$, of which only the first-order term was included in Lewis's design:

$$F(d) = F(-h\Delta) - \frac{D(-h\Delta)\delta}{1!} + \frac{D'(-h\Delta)\delta^2}{2!} - \cdots$$

Lewis's scheme exposed a further problem intrinsic to LNS arithmetic: the difficulty of interpolating $\phi^-(d)$ in the region $-1 < d < 0$ (Fig. 1). To maintain accuracy, it is necessary to implement a large number of successively smaller intervals as $d \to 0$.

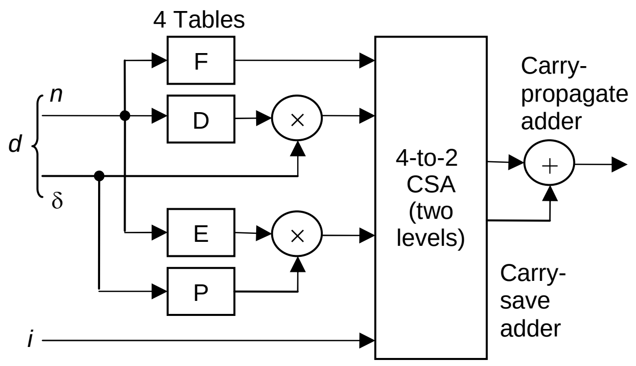

Fig. 3: An LNS adder/subtractor implementation [13].

Coleman et al. [13] published the design of a 32-bit logarithmic adder/subtractor, using a first-order Taylor-series approximation, augmented with concurrent error estimation to form a correction term. The critical path of their design (Fig. 3) contains a ROM, a multiplier, and two adders. For the range $-1 < d < 0$, they employed a range-shifter for transforming a subtraction into one having $d < -1$, with an extra latency of one ROM stage, a carry-propagate adder, and a carry-save adder.

## 4. ELM's Hardware Architecture

As noted by Coleman et al. [14], interpolation is difficult for subtractions in the region $-1 < d < 0$. Because the range $-1 < d < -0.5$ is not as problematic, the smaller range $-0.5 < d < 0$ is targeted for range shifting to reduce the required table size. The range-shift algorithm obviates such problematic subtractions by transforming $i$ and $j$ into new values that yield $d < -1$. A full description of the theory underlying this unit is given in [13]. It has a delay of one ROM access, a carry-propagate addition, and a carry-save stage.

The ALU for ELM has two separate circuits for add/subtract and multiply/divide. The add/subtract circuit is preceded by a range shifter and associated control logic, followed by a mux to select either the unmodified or range-shifted inputs, depending on the value of $d$. In the special cases involving one or two zero operands, or when the two operands are equal, the result may either follow one of the operands or be zero itself. The two input operands and the value 0 are therefore made available to a final 4-way mux, the setting of which is determined by the control logic.

For comparison, Coleman et al. also designed a 32-bit FP unit and a 32-bit fixed-point unit on the same lines as the LNS unit. As far as possible, blocks were reused from the LNS design. They did not design a fixed-point or FP divider for comparing division times with LNS, but it is known that division typically takes 2-3 times as long as multiplication.

Table 1. Latencies of VLSI arithmetic circuits (ns).

| Operation | Fixed | FP | LNS |
|---|---|---|---|
| Add | 4 | 28 | 28 |
| Subtract | 4 | 28 | 28/42 |
| Multiply | 32 | 22 | 4 |
| Divide | -- | -- | 4 |

Depending on whether the range-shifter is required in a particular subtraction, two timing values are shown for subtraction in Table 1. Assuming that the range-shifter comes into play in 50% of subtractions on average, the mean subtraction time amounts to 35 ns. Assuming an equal mix of additions and subtractions, the average add/subtract time is about 31 ns, and the average LNS multiply/divide time is 4 ns. Thus, the LNS unit would reach roughly twice the speed of FP at an add/multiply ratio of about 40/60 percent.

LNS ALUs have markedly different characteristics from their FP counterparts and therefore require some reevaluation of the surrounding microprocessor design to deploy them to best advantage. Paliouras et al. [11] used Lewis's interleaved memory function interpolators [9] as the basis for the addition unit in a proposed very long instruction word (VLIW) device optimized for filtering and comprised of two independent ALUs, each with a 4-stage pipelined adder and a single-cycle multiplier.

Arnold [12] also proposed a VLIW device with one single-cycle multiplier and a 4-stage pipelined adder, suggesting that large-scale replication of functional units could lead to difficulties with the complexity of the multiplexing paths leading to and from the registers.

Because variable delay time complicates the pipeline control algorithm, Coleman et al. [13] decided against pipelining the adder unit. Instead, they suggested a fully interlocked pipeline in which a single-cycle ALU executes all operations except LNS addition/subtraction. The latter are handled by the multicycle ALU, which executes with a 3-cycle latency.

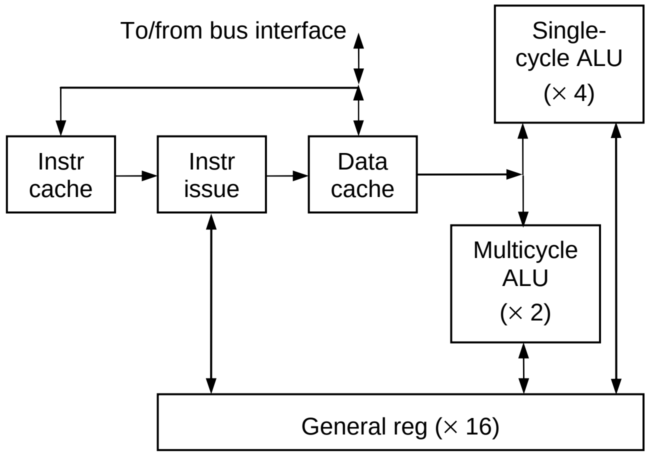

Fig. 4. ELM pipeline [14].

## 6. Accuracy and Speed

Coleman et al. [13] used a simulation model of their LNS unit to compare errors produced by the 32-bit LNS and FP systems. The LNS or 32-bit FP value was used as input to the 32-bit implementation under test, while the 80-bit value was supplied to its 80-bit counterpart. In each case, the 80-bit algorithm returned an accurate result and from this, error in the 32-bit system was derived. Values of $|e|_{\text{av rel arith}}$ were calculated for both the 32-bit LNS and 32-bit FP implementations over the entire result file. We see from Fig. 5 that the objective of an LNS addition algorithm with substantially the same error as FP has been achieved, the only discrepancy occurring for subtraction with closely matched operands.

ELM was fabricated in 0.18 μm technology and all of the measurements reported by Coleman et al. [14] were taken from the system running at 125 MHz. As the FP comparison basis, they chose the Texas Instruments TMS320C6711 DSP chip, a superscalar VLIW device with load/store architecture that runs at 150 MHz.

ELM has a lower latency than TMS in almost all cases, and hence a significantly higher scalar throughput. The figures suggest that ELM can be expected to offer around twice the performance of TMS. Figure 6 contain error comparisons for multiply-accumulate. In cases involving a large proportion of division and/or square-root operations, ELM offers even greater advantage.

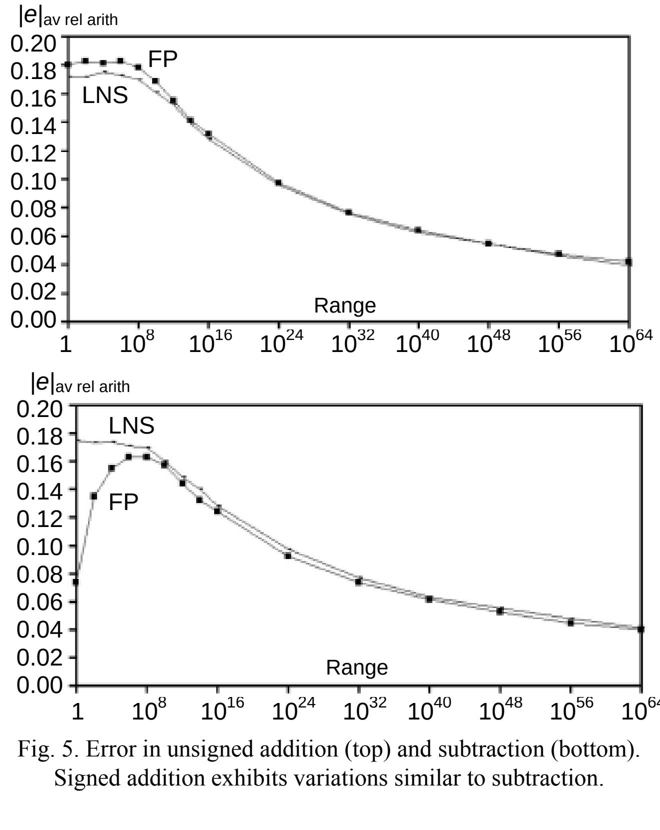

Fig. 5. Error in unsigned addition (top) and subtraction (bottom). Signed addition exhibits variations similar to subtraction.

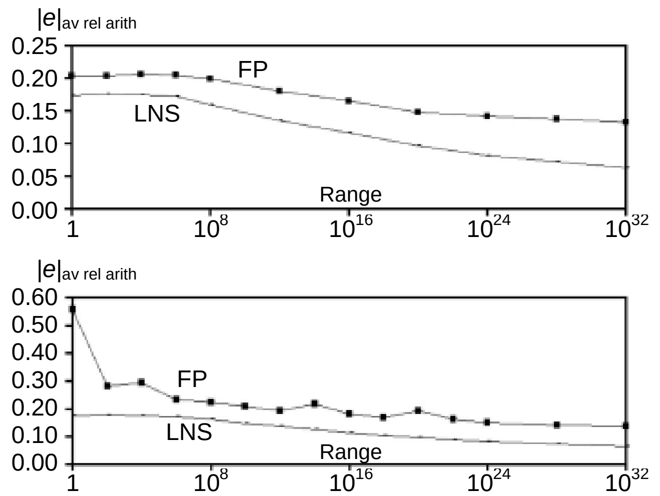

Fig. 6. Error characteristics of unsigned (top) and signed (bottom) multiply-accumulate, $ax + y$. Sum-of-products, $ax + by$, exhibits fairly similar variations.

## 7. Case Studies

We have examined in some detail two case studies in digital signal processing (recursive least-squares, RLS) and numerical methods (Laguerre algorithm) [16]. RLS algorithm results are plotted in Fig. 7 in terms of signal-to-noise ratio (SNR). The LNS implementation is seen to offer a significant gain in accuracy for signals of wide dynamic range. Over the narrower portion of the range, the LNS outperformed FP by an average of 2.7 dB. For the wider half, this gain was 10.5 dB. Since 6 dB corresponds approximately to 1 bit, this represents a gain in accuracy of from 0.5 bit to 1.5 bits.

The Laguerre algorithm [19], which computes the roots of a polynomial, requires square-rooting. In the FP version, this was evaluated via an initial approximation from a 4K word ROM (covering the entire FP range), followed by 2 Newton-Raphson iterations. In all cases, LNS error was between 0.74 and 0.79 that of FP. A similar consistency was observed with increasing $n$, as shown in Fig 8. The improvement is seen to correspond to about 0.5 bit at $n = 10$.

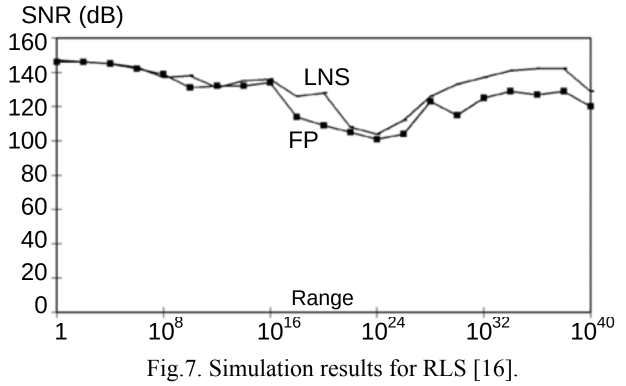

Fig.7. Simulation results for RLS [16].

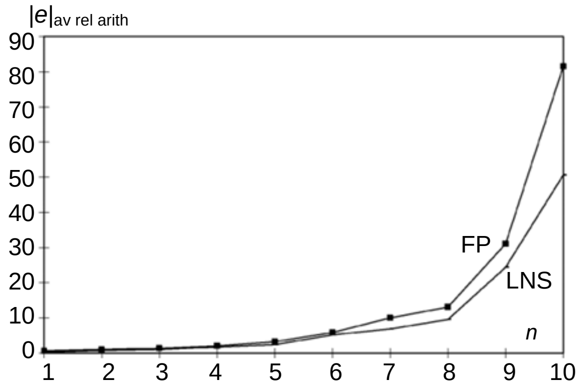

Fig.8. Simulation results for Laguerre [16].

## 8. Conclusion

We have come to understand that LNS can offer an overall advantage over FP system only if addition and subtraction can be performed with the speed and accuracy at least equal to that of FP [22]. Satisfying this goal is complicated by the latter operations requiring interpolation of a nonlinear function. Besides simple table lookup, techniques/architectures for interpolation of data using expansion series have been investigated, and ways to minimize the error have been devised.

The European logarithmic microprocessor, a device built around the research project launched in 1999 which aimed to develop a microprocessor based on the LNS, demonstrated that LNS's average performance exceeds that of FP, in terms of both speed and accuracy. ELM researchers have not offered a comparison of their LNS-based microprocessor with other commercial systems running LNS units. Examples of the latter hybrid systems include a Motorola microcontroller chip (120 MHz LNS 1 GFLOP), which has several parallel LNS units, and the IMI-500 graphics workstation from Interactive Machines Inc.

Comparisons among different LNS-based systems, and performance evaluation for a broader set of applications, constitute suitable areas for further study.

## References

[1] H. L. Garner, "Number Systems and Arithmetic," in *Advances in Computers*, Vol. 6, F. L. Alt and M. Rubinoff (eds.), Academic Press, 1965.

[2] N. G. Kingsbury and P. J. W. Rayner, "Digital Filtering Using Logarithmic Arithmetic," *Electronics Letters*, Vol. 7, pp. 56-58, 1971.

[3] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," *IEEE Trans. Computers*, Vol. 24, pp. 1238-1242, 1975.

[4] J. H. Lang, C. A. Zukowski, R. O. LaMaire, and C. H. An, "Integrated-Circuit Logarithmic Units," *IEEE Trans. Computers*, Vol. 34, pp. 475-483, 1985.

[5] F. J. Taylor, R. Gill, J. Joseph, and J. Radke, "A 20 Bit Logarithmic Number System Processor," *IEEE Trans. Computers*, Vol. 37, pp. 190-200, 1988.

[6] M. Combet, H. Van Zonneveld, and L. Verbeek, "Computation of the Base Two Logarithm of Binary Numbers," *IEEE Trans. Electronic Computers*, Vol. 14, pp. 863-867, 1965.

[7] D. Marino, "New Algorithm for the Approximate Evaluation in Hardware of Binary Logarithms and Elementary Functions," *IEEE Trans. Computers*, Vol. 21, pp. 1416-1421, 1972.

[8] F. J. Taylor, "An Extended Precision Logarithmic Number System," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 31, pp. 232-234, 1983.

[9] D. M. Lewis, "An Architecture for Addition and Subtraction of Long Word Length Numbers in the Logarithmic Number System," *IEEE Trans. Computers*, Vol. 39, pp. 1326-1336, 1990.

[10] D. Yu and D. M. Lewis, "A 30-b Integrated Logarithmic Number System Processor," *IEEE J. Solid-State Circuits*, Vol. 26, pp. 1433-1440, 1991.

[11] V. Paliouras, J. Karagiannis, G. Aggouras, and T. Stouraitis, "A Very-Long Instruction Word Digital Signal Processor Based on the Logarithmic Number System," *Proc. 5th IEEE Int'l Conf. Electronics, Circuits and Systems*, Vol. 3, pp. 59-62, 1998.

[12] M. G. Arnold, "A VLIW Architecture for Logarithmic Arithmetic," *Proc. Euromicro Symp. Digital System Design*, 2003, pp. 294-302.

[13] J. N. Coleman, E. I. Chester, C. Softley, and J. Kadlec, "Arithmetic on the European Logarithmic Microprocessor," *IEEE Trans. Computers*, Vol. 49, pp. 702-715, 2000; erratum, Vol. 49, p. 1152, 2000.

[14] J. N. Coleman, C. I. Softley, J. Kadlec, R. Matousek, M. Tichy, Z. Pohl, A. Hermanek, and N. F. Benschop, "The European Logarithmic Microprocessor," *IEEE Trans. Computers*, Vol. 57, pp. 532 - 546, 2008.

[15] J. N. Coleman and E. I. Chester, "A 32-Bit Logarithmic Arithmetic Unit and Its Performance Compared to Floating-Point," *Proc. 14th IEEE Symp. Computer Arithmetic*, 1999, pp. 142-151.

[16] J. N. Coleman, C. I. Softley, J. Kadlec, R. Matousek, M. Licko, Z. Pohl, and A. Hermanek, "Performance of the European Logarithmic Microprocessor," *Proc. SPIE Annual Meeting*, 2003, pp. 607-617.

[17] Texas Instruments, *TMS320C3x User's Guide*, 1997.

[18] C. H. Chen, R.-L. Chen, and C.-H. Yang, "Pipelined Computation of Very Large Word-Length LNS Addition/Subtraction with Polynomial Hardware Cost," *IEEE Trans. Computers*, Vol. 49, pp. 716-726, 2000.

[19] W. H. Press, et al., *Numerical Recipes in Pascal*, Cambridge, 1989.

[20] Wikipedia, "Logarithmic Number System," http://en.wikipedia.org/wiki/Logarithmic_number_system (accessed on April 30, 2013).

[21] Parhami, B., *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford, 2nd ed., 2010.

[22] Parhami, B. and M. Chugh, "A Fresh Look at Sign/Logarithmic Computer Arithmetic: Implementation Challenges and Performance Benefits," in preparation.