Unrestricted Faithful Rounding is Good Enough for Some LNS Applications

Mark G. Arnold and Colin Walter
UMIST
PO Box 88
Manchester M60 1QD UK
{marnold,c.walter}@co.umist.ac.uk

Abstract

We propose relaxing the restricted form of faithful rounding used in prior 32-bit Logarithmic Number System (LNS) implementations. Unrestricted faithful rounding yields a three- to six-fold savings in VLSI ROM size (or four- to six- fold savings in FGPA table size) with only modest increase in error. This can be acceptable for the DSP and multimedia applications in which the non-standard LNS is a candidate for adoption. Such applications are cost sensitive, and the tremendous cost reduction of the LNS model proposed here should argue very favourably for its adoption.

1. Introduction

A wide variety of applications in areas such as digital signal processing (DSP), and control systems make heavy use of real addition, subtraction, multiplication and division. Fixed-point number systems can maintain constant absolute precision using scaled integer arithmetic. Although fixed-point arithmetic is fast, it is not as power efficient (limiting battery life) for multiply-intensive applications [22] compared to the alternative we will consider here. Also, fixed point usually needs hand-crafted scaling factors at each step of the computation. Often, the designer must experiment with many different scalings to optimise these design parameters. Overflow is also a continual worry for the designer. For example, the dynamic range of a signed 32-bit fixed-point system is only from 0 to $2^{31} - 1$. In application areas such as mobile communications and multimedia, where time-to-market pressures are paramount, designers often decide it is not worth the effort to use fixed point.

Instead, it is easier for designers to use floating-point number systems that automatically attempt to maintain constant relative precision. This is because the floating-point system carries scaling information (the exponent) within each machine representation in addition to a fixed-point mantissa and sign. Consequently, floating-point systems have a much larger dynamic range than fixed-point systems using the same number of bits. For example, the popular IEEE-754 standard [11] specifies single-precision with an 8-bit integer exponent ($1 \leq \mathrm{F}_e \leq 254$), a 23-bit mantissa ($\mathrm{F}_m$) and a sign bit ($\mathrm{F}_s$). The mantissa is a fixed-point fraction scaled by 2^{-23}, i.e., $2^{23} \cdot \mathrm{F}_m$ is an integer. The real value represented by IEEE-754 single precision[1] is $(1-2\mathrm{F}_s)(1+\mathrm{F}_m)2^{-127+\mathrm{F}_e}$. The dynamic range of normalised single precision is from 2^{-127} to 2^{127}. The hardware that implements this is more complicated and requires more power than that for fixed point. An important disadvantage of floating point is that the absolute spacing of representable numbers increases abruptly at exact powers of the base—doubling at binades like 0.5, 1.0, 2.0.

A less well-known option for such applications is the Logarithmic Number System (LNS) which offers many of the advantages of fixed-point and floating-point number systems. The central mathematical idea behind LNS, which is to maintain a logarithmic representation throughout all computation (including addition), was first described by Leonelli[9] in 1803. In 1971, Kingsbury and Rayner [13] developed the first modern application for LNS: digital filtering. Several researchers, notably Swartzlander et al. [23], rediscovered this simple mathematical idea and applied it to many application areas, such as the FFT. Interest in LNS is growing. For example, Stouraitis [21], Lewis [16] and Chen and Yang [6] describe practical ways to achieve higher precision. A thorough bibliography [26]

[1] Ignoring special cases such as denormals, subnormals ($\mathrm{F}_e = 0$) and NaNs ($\mathrm{F}_e = 255$). The -127 is a bias that reflect choices about overflow and underflow.

0-7695-1150-3/01 $10.00

contains over one hundred references.

LNS maintains perfectly constant relative precision by replacing the mantissa with a fixed-point, rather than integer, exponent. The value of a 32-bit LNS representation similar to IEEE-754 is $(1 - 2L_s)2^{-128+L_e}$, where L_s is the sign bit (of the value) and $0.0 \leq L_e < 256.0$ is an unsigned fixed-point logarithm[2] (having the same scaling as F_e and occupying the same number of bits as the combination of F_e and F_m). The above is an example of a possible LNS that has 8 bits before and 23 bits after the radix point of L_e to create a system similar to IEEE-754 single precision.

When precision requirements are low to moderate and multiplication is more frequent than addition, LNS is more cost effective than floating point. For such applications, LNS is faster [7, 16] and/or consumes less power [20, 22] than similar fixed-point or floating-point alternatives. These advantages have been noted in applications such as battery-powered hearing aids [20], digital filtering [21, 13], neural nets [3], graphics[3], and hidden-Markov speech recognition [24]. The footprint of such applications is similar to burgeoning markets for portable computing/communication devices.

Another advantage of LNS is that relative precision is constant rather than wobbling as with conventional floating point, and thus can be more accurate in the sense of having a lower worst case error. The significance is not lost at discrete points but rather diminishes gradually. The LNS representation is thus inherently more accurate in this sense than floating point (for the same number of bits). Prior LNS implementations [16, 7] have attempted to implement nearly this full accuracy by computing the logarithmic approximation of a sum with about three more bits of significance than the target precision ($F = 23$ above) so that the rounded result is better than the equivalent floating-point result. The novel contribution of this paper is to explore modest weakening of this accuracy criterion in exchange for large hardware savings.

2. LNS Arithmetic

LNS represents a real number using a sign bit and a finite approximation of the logarithm of the absolute value of that real number. The representation of a product or quotient is formed by adding or subtracting the logarithms[4] and XORing the sign bits.

Thus, as has been noted many times in the literature, when the operands are exact, the product or quotient is exact. This is an advantage compared to floating point, where only a small subset of products and quotients of exact operands produce exact results.

In practice, however, LNS input operands are seldom exact. More typically, the LNS product or quotient propagates the relative errors of the inputs. Unlike floating point, no additional relative error accrues due to truncation. A weaker way to characterize this is that LNS inherently provides round-to-nearest products and quotients.

LNS addition and subtraction are not so easy, either to implement or to analyze. Given real values x and y, addition uses the property that $x+y = y\cdot(1+x/y)$. The advantage of this form is that a dyadic function $(x+y)$ has been reduced to a monadic function (increment). In order to carry out the increment operation in LNS, the hardware needs an implementation of the addition logarithm,

$$s_b(z) = \log_b(1 + b^z) \tag{1}$$

and the subtraction logarithm,

$$d_b(z) = \log_b|1 - b^z|. \tag{2}$$

When the signs of the operands are the same, $\log_b|x + y| = \log_b|y| + s_b(\log_b|x| - \log|y|)$. When the signs of the operands differ, a similar approach holds, using d_b, but the case of $x = -y$ must be handled specially, which is analogous to the special case of division by zero in the floating point system. Because of commutativity, the roles of x and y could be interchanged, thus it is common (but not required) to assume $|x| < |y|$.

3. Review of the Brown Model

The Brown model [5] postulates axioms that describe essential abstract properties of a floating-point number system without specifying much about implementation detail. Although it is possible to include many of the desirable features of the IEEE-754 floating-point standard [11] into an LNS implementation [2], the Brown model is a good starting point to develop a model of LNS. In Brown's notation, x' is the smallest interval bounded by model numbers (i.e. having exact representations) that contains an arbitrary real or interval x. If x is real but not a model number, the end points of x' will be adjacent model numbers (the

[2] $L_e = 0$ is used to represent the smallest value ≈ 0.0.

[3] Graphics (as opposed to signal processing) applications sometimes require delicate decisions be made consistently about which side of a surface a point is on. This might be a challenge with logarithmic systems, because calculation of differences is difficult in LNS. Even so, this does not appear to have prevented the application to LNS to graphics. For example, animation for Jay-Jay the Jet and other television series was performed on the IMI500 graphics engine, a 12-bit LNS processor designed by George P. Semerau, Joseph Edwards and Frank Ford Little [14].

[4] With a compensation for the bias of -128.

nearest and next-nearest points to x). In other words, x lies between these two points.

Brown's main axiom states that for a binary operator $*$ (such as $+$ or $\cdot$) if the interval $x' * y'$ does not overflow, then the model representation of $x * y$ is contained in $(x' * y')'$. We are concerned with the implementation of a feasible LNS system that satisfies this axiom. To do so, we need only consider the case that x and y are model numbers. Thus, x' and y' are also model numbers. In LNS, if we multiply or divide, we will always obtain another model number (unless the desired result overflows). The problem is LNS addition. The perfect result, $x + y$, is not a model number. However, it is possible to satisfy the axiom by ensuring that the hardware returns either of the two adjacent end points that define the interval $(x + y)'$.

4. Rounding Modes

IEEE-754 demands that the implementation be capable of a round-to-nearest mode for the arithmetic operations, which is stronger than what the Brown model requires. Although the cost of implementing a round-to-nearest mode is reasonable for floating-point arithmetic, this is not so for LNS.

The Brown model only demands what has been called a faithful rounding mode [8].

> Definitions: Round-to-nearest mode always produces the nearest representable point to the exact value. Faithful rounding mode produces either the nearest or next-nearest point. Faithful is further categorised here as unrestricted or restricted. Unrestricted faithful rounding may choose either nearest or next-nearest representations, regardless of the resulting error. Restricted faithful rounding may choose the next-nearest LNS point only if the error is no worse than the worst possible floating-point error. For this reason, we also refer to this as Better-than-Floating-Point (BTFP) rounding mode. For this paper, nearest means the exact LNS value which has the ratio with x that is closest to 1.0. This is slightly different than IEEE-754, where it means the exact floating-point value which has the difference with x that is closest to 0.0.

5. Is Round-to-Nearest Impractical?

Unlike floating point, where the sums and differences of exact operands often (but not always) produce exact results, LNS sum and differences never produce exact results (except in the case $x = y$) because s_b and d_b are irrational functions that must be approximated and rounded to some finite precision.[5] For low-precision implementations, these functions can be pre-computed to high accuracy, rounded any way desired (including round-to-nearest) and burned into a small ROM (direct-table-lookup). As the precision increases, the size of the ROM for direct-table lookup grows exponentially. For precision approaching that of IEEE-754 single precision, the most practical approaches instead involve either interpolation [15, 16] of s_b or computation of logs and anti-logs [6]. Both approximation approaches face the table maker's dilemma: without computing the function to exorbitantly high precision [18], there are cases where it is impossible to round to the nearest. Thus, round to nearest is impractical.

Instead, all previous LNS implementations obtaining precision comparable to IEEE-754 have at best used faithful rounding. Because of the difficulty of computing d_b, some implementations [16, 1, 3] have further relaxed accuracy requirements near the singularity of d_b. This can be justified by the belief that accurate d_b results are often irrelevant in this region when small differences are added to larger quantities. This depends on the application, and requires experimentation and/or very careful analysis. This willingness to accept less than faithful rounding near the singularity will be referred to as relaxed-difference-rounding.

6. Better-than-Floating-Point (BTFP) is Possible

Despite some acceptance of relaxed-difference-rounding, two recent VLSI interpolator designs implementing LNS addition [16, 7] have chosen a rather strong and restricted form of faithful LNS rounding over all regions (including the singularity[6]). This extra strong form of faithful rounding will be referred to here as better-than-FP (BTFP). As this name implies, this guarantees that for LNS and floating-point representations with comparable range and precision, an LNS unit that adheres to a BTFP mode will produce a smaller worst-case relative error than the comparable floating-point unit that adheres to an IEEE-754 round-to-nearest mode. The actual errors in both systems are random variables which can be characterized by several statistics, of which the worst-case error is the easiest

[5] The problems of multiplication in floating-point are analogous to those of addition in LNS, but not as severe since floating-point multiplication does not involve an irrational function.

[6] In considering design alternatives prior to fabrication of his interpolator, Lewis [16] examined both a strong and weak error model (the same as strong except the weak model uses relaxed-difference-rounding).

to derive. For a particular computation, the LNS unit may be more or less accurate than the floating-point one, but over a large random set of computations, the upper bound on the BTFP LNS errors will be smaller than the upper bound on the IEEE-754 errors.

This may seem paradoxical since BTFP is a faithful mode that does not always round to the nearest LNS representation whilst IEEE-754 always does round to the nearest floating-point representation. Yet BTFP mode is possible because of the inherent superiority of the LNS representation. This superiority is the result of the following: floating point exhibits a wobbling relative precision which makes the maximum relative error twice that of the minimum relative error. On the other hand, LNS maintains a constant relative precision, which is roughly $\log_e(2) \approx 0.69$ of the maximum floating-point relative error.

This $\log_e(2)$ factor makes it possible for approximations of s_b or d_b near the midpoint between the nearest and next-nearest quantised LNS representations to be rounded either way and still achieve results that on average are superior to floating point.

Figure 1 shows the LNS number line in the region near 1.0. The picture would be analogous for any representable point as relative errors are being considered (divide by the number in question to arrive at 1.0). The left end point is the real value 1.0, which has an exact LNS representation of 0. The right end point represents the smallest exact value larger than one, namely $\epsilon = \sqrt[2^F]{2} \approx 1 + \log_e(2) \cdot 2^{-F}$, assuming base $b = 2$ and that there are F bits of precision. Suppose that a real number $1.0 < x < \epsilon$ is to be represented. The value of x lies between the nearest and next nearest end points of the interval $[1, \epsilon]$. In unrestricted faithful rounding analogous to the Brown model, either end point would be an acceptable rounding. In BTFP rounding however, the rounding must go to the nearest when x is sufficiently close to an end point. Only when x is in a special region defined in the middle of the interval is it possible for the rounding to go either way. Figure 1 shows this situation as a point in this special region, and two arrows diverging from this point indicating the two acceptable rounding results. Let $\mu = 2^{-F}$ be the weight attached to the least-significant bit of the $b = 2$ LNS representation. The representations below the number line in Figure 1 are fixed-point logarithms that correspond to the real values above the number line. Obviously, zero represents 1.0. μ represents $\epsilon \approx 1 + \log_e(2) \cdot \mu$.

If the number of bits for the logarithm were not fixed at F, the exact LNS representation of the real x would be $\log_2(x)$. Instead, the quantized hardware must choose either zero or μ for the representation of x which lies in the interval $[1, \epsilon]$.

There are two cases possible: the nearest representation could either be on the right (μ) or on the left (0). Without loss of generality, we will only consider the case when the nearest point is on the right, which makes $\mu/2 < \log_2(x) < \mu$. (The picture would be symmetrical if we assumed the nearest point were instead on the left.)

> Definition: δ is the distance from the geometric midpoint of two adjacent exact LNS representations in which rounding to either point is allowed for BTFP results.

For a distance of δ to the right of $\mu/2$, rounding to the right or left end point is permitted for BTFP rounding. The real value represented by the rightmost point that can be rounded to the left and still maintain BTFP behaviour is $2^{\mu/2+\delta} \approx 1 + \log_e(2)(\mu/2 + \delta)$. The maximal BTFP relative error is thus $\log_e(2)(\mu/2 + \delta)$. Setting this equal to the maximal IEEE-754 round-to-nearest error ($\mu/2 = \log_e(2)(\mu/2 + \delta)$) and solving for $\delta = (\log_2(e) - 1) \cdot \mu/2 \approx 0.2215\mu$ defines the region where arbitrary rounding is permitted. This ensures a result at least as accurate as IEEE-754.

Lewis [16] used a similar derivation to determine data-path widths in a practical BTFP interpolator. The intent here is slightly different: to estimate the probability that rounding to next nearest is permitted, assuming x is uniformly distributed in the $\mu/2 < x < \mu$ interval. This probability is $\rho = \delta/(\mu/2) = \log_2(e) - 1 \approx 0.443$.

7. Is BTFP Cost Effective?

Although BTFP mode is initially appealing as a design goal, the question this paper addresses is whether that goal is a cost-effective one. To do this, we need to consider the cost, which relates primarily to the ROM sizes required. The following only considers s_b interpolation, as this avoids the question of d_b. The results are similar for d_b interpolation, but to arrive at these results requires the designer to make several decisions about implementing the singularity region that are not relevant to the issue at hand. Furthermore, redundant LNS [1] allows useful computation with only s_b by keeping separate positive and negative sums that are combined only at the end of a computation, thereby avoiding d_b computations until the end.

Lewis [16] uses quadratic Lagrange interpolation with a ROM containing 768 words[7] to achieve BTFP rounding of s_b for $F = 23$. Each word consists of F

[7] In general, Lewis uses roughly $3 \cdot 2^{\lceil 1+(F-4)/3 \rceil}$ words.

bits plus sufficient guard bits[8]. Thus, Lewis requires a memory with 10 address bits.[9] Lewis' simulations indicate that a ROM with this number of words is the smallest possible to achieve the BTFP goal with two-multiplier Lagrange interpolation.

Coleman's LNS interpolator [7] also achieves BTFP results using two fixed-point multipliers. Coleman does not use Lagrange interpolation, but rather a novel "error correcting" scheme. The number of ROM words for s_b in Coleman's interpolator is double that of Lewis's.

We have designed a new quadratic interpolator similar to the Lagrange interpolator proposed by Lewis[16] but that uses the minimal amount of ROM to achieve unrestricted faithful rounding. By relaxing our design goal from BTFP faithful rounding to unrestricted faithful rounding, our s_b interpolation with $F = 23$ can be implemented with as few as 234 words. This is a three- and six-fold improvement in ROM size compared to the prior implementations [16] and [7], respectively. Our interpolator only requires eight address bits.

Is BTFP mode worth over three times the ROM size? As this is in part a philosophical question, the extent to which empirical evidence can persuade LNS designers to use or abandon BTFP mode depends upon underlying assumptions. If LNS is seriously being considered as a replacement for floating-point in general-purpose computation [10], then perhaps the peace of mind that BTFP mode affords is worth its extreme cost regardless of empirical evidence. On the other hand, if (more realistically) the designer sees LNS filling a niche role in areas like signal processing or multimedia, empirical evidence about accuracy may persuade the designer to use unrestricted faithful rounding instead.

8. Simulation

To study the effects of alternative rounding methods, addition and subtraction routines were implemented that randomly choose either the nearest or next-nearest representable LNS point to the "exact" result (computed with double-precision floating point from inputs that were similarly truncated during earlier computations). By choosing between nearest and next nearest at random, we avoid concerns over the merits of various prior interpolation techniques, and focus instead on what accuracy criteria such methods should strive towards. The random selection is performed without regard to the value of z. This overestimates the effect of rounding to non-nearest since for $b = 2$ and $z < -25$, the nearest value to $s_b(z)$ is zero, which is easily realisable in an actual hardware implementation.

The routines allow specification of the probability, p, that the next nearest is chosen. Also the routines can be configured either for unrestricted or for restricted random selection between the nearest and the next nearest. If the selection is unrestricted, any result, no matter how close to the nearest point, may be chosen at random to round to the next nearest. If selection is restricted, only those results where BTFP relative error is possible may be chosen at random to round to the next nearest.

When $p = 0$ either in the unrestricted case or in the restricted case, the routines implement round-to-nearest. In the restricted case, for situations where rounding to next nearest would be BTFP, the decision whether to round to next nearest is based upon $p/\rho \approx 2.25p$, which makes the outcome comparable to the unrestricted case. This was done on the assumption of a uniform distribution of the truncated part of results, which was empirically observed in the simulations described below.

For the unrestricted case, $0 \leq p \leq 1$ is meaningful, where $p = 1$ rounds every result to the next nearest, which has the effect of dropping about 1.5 bits of precision. For the restricted case, $0 \leq p \leq \rho$ is meaningful, where $p = \rho$ rounds every result to the next nearest when BTFP behaviour is possible.

9. FFT Example

One area in which LNS has been applied successfully is DSP [21]. One of the most common algorithms in DSP is the Fast Fourier Transform (FFT), which comes in many variations and has been the subject of previous LNS studies [17, 23, 12]. One of the simplest variations is the radix-two FFT, which takes 2^n complex points that specify a periodic time domain signal, and produces 2^n complex points that specify the corresponding frequency spectra. A 2^n-point FFT uses n stages, where each stage involves 2^n complex multiply-accumulate operations. Each complex product involves one of the constant 2^n roots of unity. The implementation here of each complex multiply accumulate involves four real additions and four real multiplications.

The FFT is reasonable for this study as the structure of the FFT allows an error in a single computation of the first stage to have the potential to propagate to every computation in the final stage. Whether such an

[8]The minimum number of guard bits required has typically been determined by exhaustive simulation [15, 16] and is not considered here. Two to four guard bits are typical.

[9]In custom VLSI, this memory would only occupy an area proportional to the 768 words used, but in an FPGA, the memory configurations are available only in power-of-two sizes, hence this table would require 1024 words.

error propagates depends on the relative magnitudes involved. (Adding a large exact number to a small inexact one diminishes the relative error; adding a small exact number to a large inexact one does not.) In this particular case, the input is a real-valued 25% duty-cycle square wave plus complex white noise (generated pseudo-randomly). Since this makes the real components larger than the imaginary ones for the majority of computations in the early stages, in general slightly greater error is propagated to the real components of the final stage than to the corresponding imaginary components. Thus, error in the real components provides a metric to compare the effects of variations in rounding.

The FFT is re-run 250 times for a particular rounding at a particular precision (F) and number of FFT stages (n) but with different pseudo-random noise. Thus, $250n2^n$ complex multiply-accumulate operations occur, which would be implemented in hardware by using $1000n2^n$ LNS interpolations. The 2^n real results from those 250 runs are compared against a full double-precision FFT, and the normalised RMS error (i.e. RMS scaled by 2^F) is tallied for a particular F and n. In other words, each normalised RMS error is calculated from $250 \cdot 2^n$ numbers that are the result of $1000n2^n$ roundings, which is on average $4n$ rounding steps per number. The experiment ran with values of n between 5 and 11, corresponding to between 20 and 44 rounding steps. For each n, the simulation is run for F between 17 and 26. This range ($F >> n$) was chosen to avoid any artifacts inherent to the FFT that occur when the points on the unit circle are not represented with adequate resolution.

We can derive a lower bound to verify that our simulations are sensible. Let E_p be the average BTFP error at probability p, E_0 be the round-to-nearest LNS error (BTFP error at $p = 0$), E_ρ be the maximal BTFP error, E_{FP} be the maximal floating-point error and $U_p \geq E_p$ be the unrestricted error at probability p. By the definition of BTFP, $E_{\mathrm{FP}} \geq E_\rho$. Thus, $E_0/E_{\mathrm{FP}} \leq E_0/E_\rho$. In theory, round-to-nearest LNS is at most $\log_e(2)$ better than the round-to-nearest floating-point error, i.e. $\log_e(2) \leq E_0/E_{\mathrm{FP}}$. We therefore expect $\log_e(2) \leq E_0/E_\rho$. Of course, at $p = \rho$, we expect $E_p/E_\rho = 1$. From these two points, a simple linear model would then predict $E_p/E_\rho \geq \log_e(2)+\log_e(2)p \approx 0.69 + 0.69p$ for $0 \leq p \leq \rho$, which all of the following simulation results obey fairly closely.

To put the results in this form requires two steps. First, the simulator determines the normalised RMS errors ($E_\rho \cdot 2^F$) from restricted faithful rounding with $p = \rho$, which represents the worst case that could still be considered better than floating point. Then, the simulator is run with both restricted ($E_p \cdot 2^F$) and unrestricted ($U_p \cdot 2^F$) rounding for values of p from 0.01 to 0.06. The normalised RMS errors reported from both of these runs are divided by the corresponding $E_\rho \cdot 2^F$ (computed with the same F and n) to give a ratio of how much worse the extreme case of BTFP (and by extension, floating point) is than the LNS rounding under consideration.

The data for restricted rounding with an FFT size of $n = 11$ are plotted in Figure 2. The triangles are the observed error ratios (for each precision $17 \leq F \leq 26$) and $E_p/E_\rho \approx 0.71 + 0.73p$ is the fitted line. The standard deviation of the difference between the fitted line and the observed points is 0.0067. Similar results were observed for:

$$\begin{aligned} n = 10, &\quad E_p/E_\rho \approx 0.72 + 0.75p \\ n = 9, &\quad E_p/E_\rho \approx 0.72 + 0.78p \\ n = 8, &\quad E_p/E_\rho \approx 0.72 + 0.71p \\ n = 7, &\quad E_p/E_\rho \approx 0.72 + 0.81p \\ n = 6, &\quad E_p/E_\rho \approx 0.72 + 0.72p \\ n = 5, &\quad E_p/E_\rho \approx 0.73 + 0.65p. \end{aligned}$$

The standard deviations between these fitted lines and the observed points grows to 0.0014 at $n = 5$. This may be attributable to the fact that the number of roundings per point plotted decreases from 22,528,000 to 160,000. The fluctuation in slopes for $n \leq 7$ may be related to this as well.

Similar simulations were performed for unrestricted faithful rounding. The data for unrestricted rounding with an FFT size of $n = 11$ are plotted in Figure 3. Here dashes are the observed error ratios and $U_p/E_\rho \approx 0.72 + 2.25p$ is the fitted line. Similar results were observed for:

$$\begin{aligned} n = 10, &\quad U_p/E_\rho \approx 0.72 + 2.34p \\ n = 9, &\quad U_p/E_\rho \approx 0.72 + 2.28p \\ n = 8, &\quad U_p/E_\rho \approx 0.72 + 2.10p \\ n = 7, &\quad U_p/E_\rho \approx 0.72 + 2.16p \\ n = 6, &\quad U_p/E_\rho \approx 0.73 + 2.07p \\ n = 5, &\quad U_p/E_\rho \approx 0.73 + 2.07p. \end{aligned}$$

The standard deviations are similar to the restricted case, and thus the slopes for larger n may be more realistic. It appears the slope does increase slightly as n increases.

It is worth seeing how far this linear model holds. One interesting point is when unrestricted rounding has the worst possible probability permitted in restricted rounding. With $n = 11$, our model predicts $U_\rho/E_\rho = 1.72$. In our simulations we observed

$1.65 \leq U_\rho/E_\rho \leq 1.72$, which is in close agreement to the prediction. Taking this to the extreme, we can compare the probabilistic results above to a highly pessimistic scenario when rounding is always to the second-nearest value (i.e., $p = 1$). Here, our model predicts $\leq U_1/E_\rho = 2.97$ whilst we observe $1.80 \leq U_\rho/E_\rho \leq 2.92$, which is in close agreement. Thus lose about one and one half bits of precision in this pessimistic case.

By solving $0.72 + 2.3p < 1$ we estimate $p < 0.12$ will yield $U_p < E_\rho$, which seems a reasonable criterion for the design of an unrestricted faithful interpolator. In other words, if we design an unrestricted interpolator that rounds to the next nearest no more than 12 per cent of the time, we would expect FFT results at least as good as the extreme case of BTFP, which in turn would be at least as good as IEEE-754.

From the fitted lines, we can estimate $U_p/E_\rho - E_p/E_\rho \approx 1.6p$. Using this, we can estimate the percentage increase (q) of U_p over E_p as $q = 1.6p/(0.72+2.3p)$. A design goal of holding $U_p < qE_p$ could be satisfied by $p < 0.72q/(1.6 - 2.3q)$. For example, a five-per-cent bound requires $p < 0.024$ whilst a ten-per-cent bound requires $p < 0.052$. In fact, the tradeoffs in the design of a practical unrestricted interpolator are more complicated than these estimates [4]. Even so, the simulation data from which these estimates were derived are sufficiently encouraging that such hardware would be usable for LNS applications similar to the FFT, where marginal additional RMS errors of this nature will not interfere with the functionality of the application.

10. FPGA Implementation

There is increasing interest in FPGA-based reconfigurable computing, which enables the best number representation to be selected at will. Table-based LNS arithmetic seems well suited to the resources of FPGA architectures [24]. Unfortunately, prior LNS designs require too much table space for $F = 23$. For example, the Xilinx Virtex FPGA family [25] offers two kinds of resources: 4-input-bit Look Up Tables (LUT) used primarily to implement logic, and larger block RAMs configurable for 8- to 12- address bits. The chip we are using (XCV300) has 3072 LUT slices and 64K bits of block RAM on chip that must be shared between LNS tables and data storage. With prior techniques [16, 7], if it would fit at all, the table for an $F = 23$ LNS adder would be too big to allow any reasonable on-chip data storage. With the memory savings proposed in this paper, the block RAM of such an FPGA could be configured with one or two LNS adders plus data storage. We have shown how 24K bits (compared to [16]) or 56K bits (compared to [7]) per adder could be released for data storage in such a chip (assuming power-of-two table sizes as required by the FPGA architecture). This represents 37 or 87 per-cent savings, respectively.

The total system must include the cost of the multiply/add interpolator datapath. Assuming four guard bits, Xilinx Foundation 2.1 reports that a datapath similar to [16] requires 9 per-cent of the available LUT slices. While there are many alternative datapath designs too numerous to describe here, the 9 per-cent cost figure is a good estimate of the cost of the datapath for either the prior or the proposed interpolators. Since the predefined area assigned to block RAM cannot be traded for the predefined area assigned to LUTs, it is hard to describe a single figure of merit (like area) for FPGA implementations as would be done for VLSI implementations. In an attempt to do so, the proposed LNS tables can be implemented with 16 per-cent of LUTs instead of block RAM. This compares favourably with a large 64 per-cent (for [16]) and an impossible 128 per-cent (for [7]). Thus, the design alternatives require 25, 73 and 137 per-cent of the LUT slices. Under these assumptions, our design offers almost a three-fold (73/25) improvement compared to [16] and more than a five-fold (137/25) improvement compared to [7]. Since custom ROMs in VLSI are denser than LUTs, the cost savings in VLSI for unrestricted faithful rounding would not be as great, but would still be worthwhile.

Observations similar to this paper in the domains of fixed- and floating-point numbers have led to proposals for truncated multipliers [19]. There, the conclusion is that by going from an error of ulp/2 to ulp (or slightly more), substantial cost and power savings can be achieved. The estimates of three- to five-fold cost reduction above did not consider the additional savings that accrue from truncated multiplication in the interpolator. Such substitution may be permitted because of our unrestricted faithful rounding model.

11. Conclusions

Although some early LNS implementations produce unrestricted faithful [15] or even weaker than faithful rounding [21], the recent trend has been towards restricted faithful rounding [16, 7]. We question this latter criterion, and give preliminary evidence that it is not the most appropriate model for LNS rounding in the applications for which LNS is a leading contender.

We have shown for a well-known DSP application (FFT) that there is only a small distinction between the less costly unrestricted faithful rounding proposed here and the restricted faithful (BTFP) rounding used

by these prior F=23 LNS implementations. Perhaps because all the data is typically comparable in magnitude, FFT calculations appear to be fairly resilient against arithmetic peculiarities introduced by unrestricted faithful rounding. Whether other plausible LNS applications would be as tolerant as FFT's is the subject of our current research.

This paper reports the result of probabilistic rounding simulations and avoids details of interpolator hardware implementation. The probability p provides a concise figure of merit for faithful interpolators. Elsewhere we describe details on implementation of unrestricted-faithful interpolators[4].

12. Acknowledgments

The authors wish to thank the referees for their comments that have pointed new directions for our research. Also, M. Arnold wishes to thank XLNS Research (Boulder, CO) for the generous support provided and J. R. Deen for his assistance.

References

[1] M. G. Arnold, et al., "Redundant Logarithmic Arithmetic," IEEE Trans. Comput., vol. 39, pp. 1077–1886, Aug. 1990.

[2] M. G. Arnold, et al. "Applying Features of IEEE 754 to Sign/Logarithm Arithmetic," IEEE Trans. Comput., vol. 41, pp. 1040–1050, Aug. 1992.

[3] M. G. Arnold, et al., "Arithmetic Co- transformations in the Real and Complex Logarithmic Number Systems," IEEE Trans. Comput., vol. 47, no. 7, pp. 777-786, July 1998.

[4] M. Arnold, "A Pipelined LNS ALU," accepted for IEEE Workshop on VLSI, Orlando, Florida, 19 Apr 2001.

[5] W. S. Brown, "A Simple but Realistic Model of Floating-Point Computation," ACM Transactions on Mathematical Software, vol. 7, no. 4, pp. 445-480, Dec. 1981.

[6] C. Chen and C. H. Yang, "Pipelined Computation of Very Large Word-Length LNS Addition/Subtraction with Polynomial Hardware Cost," IEEE Trans. Comput., vol. 49, no. 7, pp. 716-726, July 2000.

[7] J. N. Coleman, E. I. Chester, C. I. Softley, and J. Kadlac, "Arithmetic on the European Logarithmic Microprocessor," IEEE Trans. Comput., vol. 49, no. 7, pp. 702-715, July 2000.

[8] D. Das Sarma and D. W. Matula, "Faithful Bipartite ROM Reciprocal Tables," 12th Symposium on Computer Arithmetic, pp. 17-28, 1995.

[9] K. F. Gauss, Werke, vol. 8, pp. 121-128, 1900.

[10] W. N. Holmes, "Composite Arithmetic: Proposal for a New Standard," Computer, vol. 30, No. 3, pp. 65-73, Mar. 1997.

[11] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985, IEEE, 1985.

[12] S. J. Kidd, "Implementation of the Sign-Logarithm Arithmetic FFT," Royal Signals and Radar Establishment Memorandum 3644, Malvern, 1983.

[13] N. G. Kingsbury and P. J. W. Rayner, "Digital Filtering Using Logarithmic Arithmetic," Electron. Lett., vol. 7, no. 2, pp. 56-58, Jan 28, 1971.

[14] F. F. Little, private communications, 29 Jul 1999 and 3 Aug 1999.

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

[16] D. M. Lewis, "Interleaved Memory Function Interpolators with Application to an Accurate LNS Arithmetic Unit," IEEE Trans. Comput., vol. 43, no. 8, pp. 974-982, Aug. 1994.

[17] S. Pan et al., "A 32b 64-Matrix Parallel CMOS Processor," 1999 IEEE International Solid-State Circuits Conference, San Francisco, pp. 15-17, Feb. 1999.

[18] M. Schulte and E. Swartzlander, "Exact Rounding of Certain Elementary Functions," 11th IEEE Symposium on Computer Arithmetic, Windsor, Ontario, pp. 138-145, June, 1993.

[19] M. J. Schulte, J. G. Jansen, and J. E. Stine, "Reduced Power Dissipation Through Truncated Multiplication," Proceedings of the IEEE Volta Memorial Workshop on Low Power Design, Como, Italy, pp. 61-69, Mar. 1999.

[20] T.J. Sullivan, Estimating the Power Consumption of Custom CMOS Digital Signal Processing Integrated Circuits for Both the Uniform and Logarithmic Number Systems, PhD Dissertation, Washington Univ, St. Louis, Missouri, 1993.

[21] T. Stouraitis, Number System Theory, Analysis, and Design, PhD Dissertation, University of Florida, Gainesville, 1986.

[22] V. Paliouras and T. Stouraitis, "Logarithmic Number System for Low-Power Arithmetic," PATMOS 2000: International Workshop on Power and Timing Modeling, Optimization and Simulation, Gottingen, Germany, 13-15 Sept., pp. 285-294, 2000.

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

[24] M. Wazlowski, et al., "Performing Log-Scale Addition on a Distributed Memory MIMD Multicomputer with Reconfigurable Computing Capabilities," Proceedings of the 1995 International Conference on Parallel Processing, pp. III-211 - III-214, 1995.

[25] Programmable Logic Data Book, Xilinx, San Jose, 1999.

[26] M. Winkel, www.xlnsresearch.com has an extensive bibliography of LNS-related articles.

Figure 1: The exact real values (1.0 and ε, shown as circles) are above the number line, with their LNS representations below. If an arbitrary real value is in the region defined by δ, it can round either to ε or 1.0 (nearest or next-nearest) and still be BTFP. If the value is larger than $2^{\mu/2+\delta}$, the value must round to ε (nearest). This figure only considers the case when the real value is between $2^{\mu/2}$ and ε. The maximum probability allowed in BTFP mode for rounding to next nearest is $\rho=\delta/(\mu/2)$.

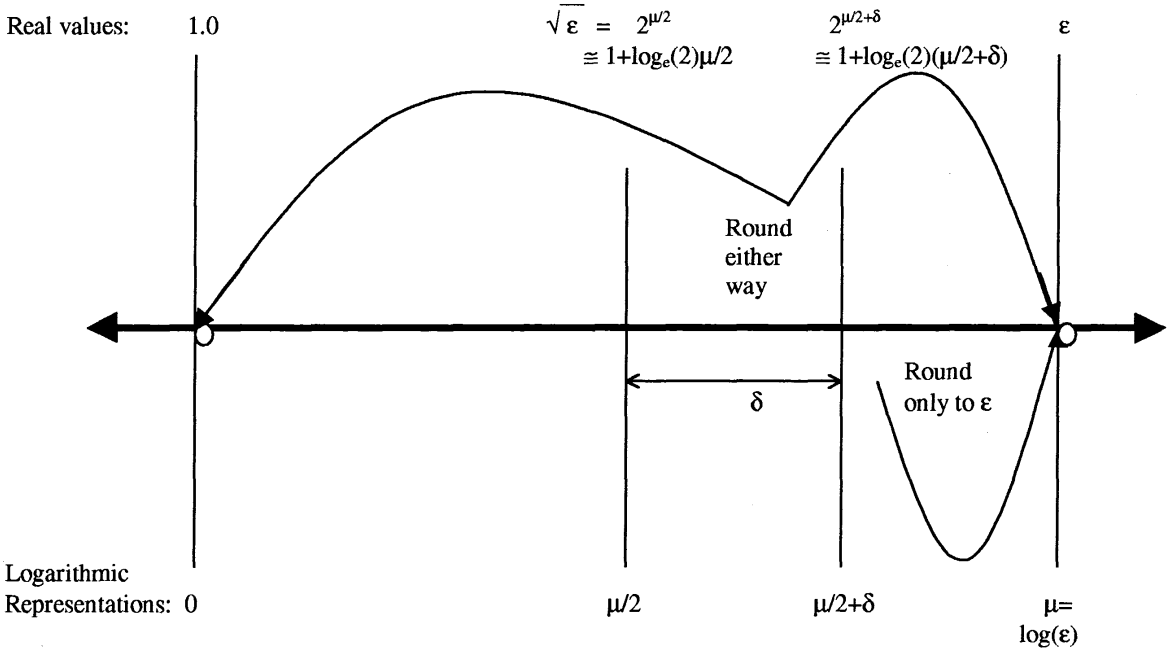

Figure 2: Observed points (triangles) and fitted line, 0.71 + 0.73*p*, in 250 simulations of the FFT with n=11 and 17 ≤ F ≤ 26 for restricted faithful LNS rounding. The abscissa is 100 times the probability (p) of rounding to next nearest and and the ordinate is RMS error as a ratio of maximal (p=ρ) BTFP error.

Figure 3: Observed points (dashes) and fitted line, 0.72 + 2.25*p*, in 250 simulations of the FFT with n=11 and 17 ≤ F ≤ 26 for unrestricted faithful LNS rounding. The abscissa and ordinate are the same as in Figure 2.