

Avoiding Oddification to Simplify MPEG-1 Decoding with LNS

Mark G. Arnold

Lehigh University

19 Memorial Drive West

Bethlehem, PA 18015

Telephone: (610) 758-3285

Fax: (610) 758-6279

Email: marnold@eecs.lehigh.edu

Abstract— Low-precision Logarithmic Number System (LNS) arithmetic can reduce the power consumption for MPEG decoding compared to conventional fixed-point techniques. Although this introduces small numeric errors which violate the IEEE-1180 standard for the Inverse Discrete Cosine Transform (IDCT), the visual effects of such error may be tolerable for portable battery-powered devices, like video phones, that have limited-resolution displays. The MPEG standard achieves video compression by quantization of the data fed to the IDCT. The MPEG decoder must multiply this data by dequantization factors. Such multiplication, by itself, is trivial with LNS since adding logarithms is equivalent to multiplication. The IEEE-1180 standard suggests oddification, where fixed-point data is forced to become odd after dequantization to minimize IDCT mismatch between the encoder and the decoder. Oddification poses an implementation problem for data in LNS format. This paper suggests that the visual effect of LNS without oddification is nearly indistinguishable from LNS with oddification, meaning that the benefits of LNS in MPEG are even greater than previously expected.

Keywords: Computer Arithmetic, Logarithmic Number Systems (LNS), MPEG, Multimedia, Inverse Discrete Cosine Transform (IDCT), IDCT Mismatch, Low Power Design, Oddification.

I. INTRODUCTION

One recent trend is that multimedia devices need to be portable and battery-powered, thus battery life becomes a very important factor. This paper will suggest that some minor violation of multimedia standards (compatible from a user's perspective, but in mild violation from an implementor's perspective) offer significant improvement in battery life.

The Motion Picture Expert Group (MPEG) video compression standards [14] utilize a standard (IEEE-1180) dealing with still-image representation. The latter in turn depends on standards for number representation (IEEE-754).

II. NUMBER REPRESENTATION

Fixed point and floating point are the usual real-number representations used in multimedia. This paper introduces a non-standard number representation used here to extend battery life.

A. Floating Point

We can describe a floating-point number system as a mapping of a representation, x , into a real value X where

$$X = (-1)^{x_S} \cdot x_M \cdot r^{x_E}, \quad (1)$$

r is the radix and the word x is subdivided into three parts: the sign ($x_S \in \{0, 1\}$), the mantissa (F -bit wide x_M) and the exponent ($x_E = \lfloor \log_r |x| \rfloor$). IEEE-754 uses $r = 2$, single (32-bit x with $F = 23$) and double (64-bit x with $F = 52$) precision with hidden-bit normalization (i.e., there is an assumed 1 in x_M which is not counted towards the total F).

B. Fixed Point

IEEE-754 is a general-purpose floating-point system. Specialized applications, such as MPEG decoding, often avoid IEEE-754, and use a fixed-point number system instead, where the mapping from the signed integer word x to the real X is based on a simpler, fixed scaling:

$$X = x \cdot r^{-F}, \quad (2)$$

and the integer constant F determines the absolute precision of representation. In addition to these bits, the word size, W , must consider the dynamic range of the application. For MPEG, this requires 12 bits, so $W = F + 12$.

C. Logarithmic Number Systems (LNS)

The Logarithmic Number System (LNS) offers advantages over fixed point and floating point. LNS maps a representation, x (consisting of x_S and x_L), into a real value x as

$$X = (-1)^{x_S} \cdot b^{x_L}, \quad (3)$$

where $x_L = 2^{-F} \lfloor 2^F \cdot \log_b |X| + 0.5 \rfloor$ is a fixed-point base- b logarithm, x_S is the value sign bit. ($x_S = 1$ means negative.) An advantage of LNS is that it can cover the same dynamic range with fewer bits than fixed point. For MPEG this means $W = F + 6$. Although a special bit could be set aside to represent the exact value zero, an acceptable and more efficient approach is to use the smallest value (2^{-16} for $W = F + 6$) as an approximation of zero.

Multiplications and divisions are implemented as simple fixed-point addition (and XORing of the signs) without introduction of any additional relative error. This is in contrast to floating point, which often has to round a product.

The problem with LNS is addition and subtraction. Given x and y , when $x_S = y_S$, the hardware uses

$$\log_b |X + Y| = x_L + s_b(x_L - y_L), \quad (4)$$

where $s_b(z) = \log_b(1 + b^z)$ is known as the addition logarithm [10]. When the $x_S \neq y_S$,

$$\log_b |X + Y| = x_L + d_b(x_L - y_L), \quad (5)$$

where $d_b(z) = \log_b |1 - b^z|$ is known as the subtraction logarithm. Unfortunately, s_b and d_b cannot be computed exactly, and approximating them is often expensive [2].

Paliouras and Stouraitis [16] discovered that the choice of the base of the logarithms, b , has a significant effect on LNS power consumption. If the designer chooses b so there is no excess dynamic range, LNS power consumption is 50% of the equivalent fixed-point. Sacha and Irwin [17] investigated similar power-consumption advantages of LNS in the context of digital filters, and Sullivan [18] did so for an LNS hearing aid. C. C. Chen's simulations [7] show that in a context similar to the one considered here floating point requires two more bits of precision compared to LNS.

Elsewhere [2], [4] modifications to the Berkeley MPEG tools [5] are reported that allow experimentation with LNS-based MPEG decoding. LNS-based MPEG decoding offers one-tenth the power consumption of fixed point [2], [3]. This is possible because such designs assume that LNS and fixed point are equivalent as long as the video output looks roughly the same on a limited resolution display.

III. MPEG DECODING

The numerical heart of MPEG decoding is the Two-Dimensional Inverse Discrete Cosine Transform (2D-IDCT) [14]:

$$f(x, y) = \sum_{u=0}^7 \sum_{v=0}^7 F(u, v) \cdot \frac{C_u}{2} \cdot \frac{C_v}{2} \cdot \cos \left[\frac{(2x+1)u\pi}{16} \right] \cdot \cos \left[\frac{(2y+1)v\pi}{16} \right], \quad (6)$$

where $F(u, v)$ is an input matrix value previously computed using the Discrete Cosine Transform (DCT) in the encoder, $f(x, y)$ is the reconstructed output matrix value, and the result is scaled by:

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases} \quad (7)$$

The IEEE-1180 standard [12] allows implementers flexibility in carrying out an approximation of (6), as several faster algorithms exist [13], [7], [6], but this standard defines that the absolute error of an approximate implementation must not exceed the exact result (computed with IEEE-754 floating point) by more than 1. The factor-of-ten power savings with LNS reported in [3] are possible only by allowing a few errors larger than 1. Although the research related to this paper [2] is the first attempt to use LNS for MPEG, others have suggested relaxing IEEE-1180 [13], [1], [9], [8].

MPEG uses three kinds of pictures: I, P and B. I pictures are completely specified by the results of the IDCTs for that picture. P and B pictures use data from other pictures to achieve motion compression. P depends on previous I or P; B depends on earlier or later I or P. A Group of Pictures (GOP) consists of an I picture together with zero or more B and/or P pictures.

Because IEEE-1180 allows different IDCT algorithms, an IDCT approximation in the decoder may behave differently than one used in the encoder. In a series of P pictures, such errors may accumulate and become visible. The larger the GOP, the more severe this IDCT mismatch may become. The IEEE-1180 standard suggests and MPEG-1 uses oddification, which is described in the next section, to minimize IDCT mismatch between the encoder and the decoder. MPEG-2 uses a more complex method of mismatch control involving only $F(7, 7)$ [14]; this is not considered here.

IV. ODDIFICATION

The purpose of oddification is to avoid producing a fixed-point result whose fractional part is exactly 0.5, and therefore rounding in the decoder differently than in the encoder [14]. At most, this will make a difference of 1 in the integer result of the IDCT. The concern that motivated oddification in the IEEE-1180 [12] and MPEG-1 standards is that biased errors could accumulate in a series of P frames that reference the same block in which the encoder consistently rounds 0.5 one way but the decoder rounds the other way [14]. Here, on the other hand, the use of low-precision LNS introduces relatively unbiased noise larger than anything that oddification would prevent in fixed point, yet the experiments discussed in [2], [3], [4] suggest such arithmetic produces reasonable visual results. Since oddification occurs after dequantization (i.e., multiplication by two quantization integers), the LNS precision



Fig. 1. Result of decoding `ka15p` using floating point with oddification (IEEE-1180) for (a) Frame 46 does not exhibit IDCT mismatch because it is an I frame; (b) Frame 45 does exhibit IDCT mismatch because it is the result of a long string of P frames.

required to distinguish consistently the oddified data from the original data is typically 3 to 6 bits larger than the precision required to represent the original data. Put another way, bit-for-bit, there will be little difference between the original dequantized data and the corresponding oddified data after they are converted to LNS for $F \leq 6$. There are some LNS implementation advantages to performing the dequantization multiplications in LNS without worrying about the effectively pointless oddification step.

In order to study these effects, six MPEG files were encoded from the identical input video source: 74 time-lapse frames of the author’s wife (Karolyn) working at a laptop as shot by an inexpensive 144×176 webcam at a stationary position with an unchanging background. This bears some resemblance to the usage of a video phone. For the purpose of this experiment, none of these files were encoded to have B-type pictures. These files differ only in the GOP size: `ka1p.mpg`, `ka5p.mpg`, `ka10p.mpg`, `ka15p.mpg`, `ka45p.mpg` and `ka73p.mpg` have GOP sizes of 1, 5, 10, 15, 45 and 73 respectively.

Oddification attempts to reduce artifacts that might occur in a long sequence of P pictures. The longer the sequence, the more opportunity there is for IDCT mismatch artifacts to accumulate. Such artifacts are often considered acceptable because cost metrics, such as the number of IDCTs (39k, 11k, 8k, 7k, 5.5k and 5k respectively) and bytes (193k, 56k, 40k, 33k, 25k and 23k respectively), decrease as the GOP size increases. Although the MPEG standard allows up to 132 P pictures to occur before forced updating with a new I picture [14], realistic MPEG files seldom exceed the number of P frames considered here.

In `ka15p.mpg`, the I frames (which allow us to start anew) occur every 15 frames. For example, frame 31 is an I frame. Frames 32–45 are P frames that act like a series of dominos: the error in frame 31 gets propagated to frame 32, which in turn adds some of its own and propagates these errors to frame 33, etc. The last frame to be affected by the accumulation is frame 45. Frame 46 starts over with a new I picture. Figure 1 compares frame



Fig. 2. Result of decoding frame 45 of `ka15p` using $F = 6$ LNS (a) without oddification and (b) with oddification.

45 (which has 14 frames of accumulated errors) in contrast to frame 46 (which is free of this source of error) using the IEEE-1180 standard reference IDCT (IEEE-754 floating point with oddification). Frame 45 has significant artifacts to the left of Karolyn’s head (because her head has been there in previous frames) even though this frame was decoded according to the standard. In frame 46, the background near Karolyn’s head clarifies, and her glasses are no longer obscured in the noise because the frame was created from scratch using the data in the I picture. The general noisy appearance of frame 45 in contrast to the smoother frame 46 (more visible in black and white here than in actual color) may be attributable to noise in occasional frames from the inexpensive camera.

Figure 2 shows that such artifacts are not primarily the result of the kind of arithmetic used by the decoder. Figure 2 (b) shows frame 45 decoded using $F = 6$ LNS with normal oddification. $F = 6$ is enough precision for the LNS to be almost identical to Figure 1 (b). Figure 2 (a) shows how $F = 6$ LNS decodes the same frame without oddification. In each of the three versions, visible mismatch occurs. Although using LNS and omitting oddification slightly aggravate the problem, overall the visual quality is nearly as good as the IEEE-1180 version, as can be seen by comparing Figures 1 (b) and 2 (a).

A quantitative error metric is the absolute value of the proposed IDCT minus the output of the IEEE-1180 reference computed with oddification and floating point. Although this quantitative measure cannot express all of the possible artifacts (especially the accumulation of error in long strings of P frames), the more similar the quantitative measures are to each other, the more likely it is that additional artifacts will not appear in the video. In this regard, histograms are useful to show the number of cases in which this absolute error is of a particular magnitude. For example, looking at such histograms can give us a better view of the overall effect of the LNS precision, F , on the IDCT performance.

Figure 3 shows histograms for the IDCT errors in decoding the entire file `ka15p.mpg` with $5 \leq F \leq 7$. It is im-

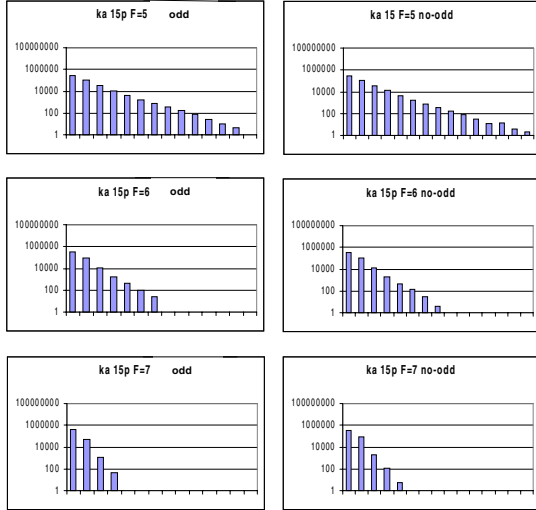


Fig. 3. Histograms with and without oddification for $5 \leq F \leq 7$ LNS using a GOP size of 15.

portant to notice is that the number of errors is plotted on a logarithmic scale. The bar on the left of each histogram indicates the number of cases in which the proposed IDCT result is identical to that produced by the ideal IEEE-1180 standard. The bar next to this indicates the number of cases in which the absolute error is 1. To comply with the IEEE-1180 standard, all of the cases would have to be accounted for in one of these two bars. With any of the low-precision LNS histograms plotted in Figure 3, this is not the case, and the bars taper off in nearly a straight line, which considering the log scale means an exponential decay in the actual number of cases. With this file (having a GOP size of 15), there is only a slight distinction between LNS with oddification and LNS with no oddification. This distinction is mainly noticeable by the fact that there are extra bars (15 versus 13, 8 versus 7, and 5 versus 4) on the right, indicating a handful of slightly larger outlier cases. Of course, these outliers may propagate through several P frames, but what is important to realize is that having chosen to use low-precision LNS with such triangular distribution, omitting oddification produces an essentially indistinguishable distribution. (Not shown are histograms for higher precision LNS: With oddification, LNS eventually become completely IEEE-1180 compliant (two bars) before $F = 12$. Without oddification even high-precision LNS histograms remain incompatible (more than two bars) with the letter of the standard. Also omitted here due to space limitation are similar histograms for fixed point. At high precision, fixed point and LNS are essentially identical (IEEE-1180 compliant). For the same low-precision F , fixed point often has bi-modal and unpredictable histograms that exhibit almost an order-of-magnitude greater error than LNS [2].)

The histograms in Figure 3 do not fully reflect how omit-

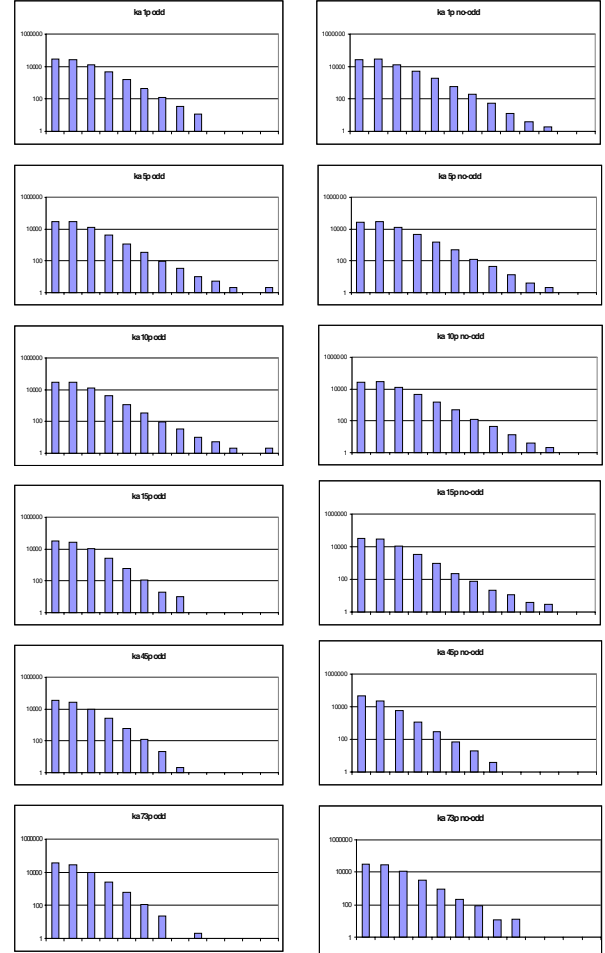


Fig. 4. Absolute pixel-difference histograms for the final P frame in a GOP that compares $F = 6$ LNS using (left column) and not using (right column) oddification against IEEE-1180 (IEEE-754 floating point with oddification) for GOP sizes of 1, 5, 10, 15, 45 and 73.

ting oddification impacts the final video output. Since the errors from several IDCT computations can contribute to the final video output after a long sequence of P frames, we need to look at the pixel output at the completion of MPEG decoding, rather than at each IDCT output. To measure such pixel output, we subtract the red, green and blue data for pixels of an interesting frame from the corresponding data in a comparison frame. (For the comparison frame we use the frame produced by an IEEE-1180-compliant decoder with oddification using IEEE-754 floating point.) The absolute differences between the comparison data and LNS with oddification are computed on the final frame in a GOP for several GOP sizes. Also, the errors between the comparison data and LNS without oddification are computed in a similar way. Figure 4 shows pixel-difference histograms, with and without oddification, for ka1p.mpg, ka5p.mpg, ka10p.mpg, ka15p.mpg, ka45p.mpg and ka73p.mpg (GOP sizes of 1, 5, 10, 15, 45

and 73, respectively) using $F = 6$ LNS. In each case except `ka73p.mpg`, the frame used for the comparison is the 45th. In `ka73p.mpg`, the frame used is the 73rd.

The histograms in Figure 4 have a slight trend for the pixel differences to cluster closer to 0 as the GOP size increases. At first glance, this seems counterintuitive to the expected situation in which errors increase as GOP size increases. In fact, this trend does not indicate a reduction in error, but rather that the nature of the pixel differences becomes similar as the GOP size increases. For example, consider the `ka15p.mpg` and `ka45p.mpg` histograms. There are eight bars in the `ka15p.mpg` histogram with oddification, but there are eleven bars without oddification. In contrast, there are eight bars in both `ka45p.mpg` histograms. For `ka15p.mpg`, the standard deviation is 0.90 with oddification and 0.97 without oddification. In contrast, for `ka45p.mpg`, the standard deviation is 0.89 with oddification and 0.75 without oddification.

To put these in perspective, if we compute the absolute pixel difference between the `ka45p.mpg` IEEE-1180 comparison frame and the original image from which frame 45 of `ka45p.mpg` was encoded, we would obtain a histogram with 80 bars. The standard deviation of this data is 7.67, which is about than a factor of ten wider than the histograms in Figure 4. Thus, in this example at least, the significant source of pixel error is the motion compression inherent in the MPEG standard, and not the choice of LNS or oddification.

V. IMPLEMENTATION

The absolute value of the data output from the Variable Length Code/Run Length (VLC/RL) decoder prior to oddification and dequantization is less than 64. The advantage for omitting oddification in an LNS implementation is that dequantization simply involves multiplication, but oddification involves incrementation or decrementation. Multiplication is an easy operation in LNS that introduces no error that was not already inherent in the quantized data from the MPEG file. Incrementation or decrementation in LNS involve the s_b or d_b functions that will introduce error. If we can avoid oddification without significant visual artifacts, then integer-to-LNS conversion can occur directly on the output of the VLC/RL decoder. In fact, since VLC/RL decoding is often table driven [5], the integer-to-LNS conversion can be merged into the VLC/RL table itself at almost no cost. The cost of oddification is, of course, eliminated, and the cost of dequantization (two multiplications) is replaced by the lower cost of LNS implementation (two additions).

VI. CONCLUSION

Previous research has shown that low-precision LNS arithmetic can extend the battery life for an MPEG decoder. Using such LNS introduces small errors into the

video output which may be acceptable for display on limited resolution devices. Because of the special nature of LNS arithmetic, avoiding oddification does not contribute much additional error, but does make significant cost savings in the hardware.

REFERENCES

- [1] Carlos Alvarez, et al., "Fuzzy Memorization for Floating Point Multimedia Applications," Computer Architecture TC Newsletter, pp. 10-11, Oct. 2001.
- [2] M. G. Arnold, Logarithmic Number Systems for MPEG and Multimedia Applications, PhD dissertation, University of Manchester Institute of Science and Technology, 2002.
- [3] Mark G. Arnold, "Reduced Power Consumption for MPEG Decoding with LNS," Application-specific Systems, Arch., & Processors, IEEE, San Jose, CA, pp. 65-75, 17-19 July 2002.
- [4] M. Arnold, "LNS for Low Power MPEG Decoding," SPIE Advanced Signal Processing Algorithms Architecture and Implementations XII, Seattle, July 7-11, 2002.
- [5] Berkeley MPEG Tools, bmc.berkeley.edu/research/mpeg/.
- [6] Wen-Hsiung Chen, C. Harrison Smith and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," IEEE Transactions on Communications, vol. COM-25, no. 9, pp.1004-1009, Sept. 1977.
- [7] C. C. Chen and Y. Y. Chen, "Error Analysis of DCT Algorithms in Floating Point and Logarithmic Number Systems," 9th VLSI Design / CAD Symp., Nan-Tow, Taiwan, pp. 313-316, 1998.
- [8] F. Fang, T. Chen, and R. A. Rutenbar, "Lightweight Floating-Point Arithmetic: Case Study of Inverse Discrete Cosine Transform," EUROCHIP Journal on Applied Signal Processing, Special Issue on Implementation of DSP and Communication Systems, Sept. 2002.
- [9] F. Fang, T. Chen and R. A. Rutenbar, "Floating-Point Bit-Width Optimization for Low-Power Signal Processing Applications," ICASSP 2002, Orlando, FL, U.S.A., May 2002.
- [10] N. G. Kingsbury and P. J. W. Rayner, "Digital Filtering Using Logarithmic Arithmetic," Electronics Letters, vol. 7, no. 2, pp. 56-58, 28 Jan. 1971.
- [11] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985, IEEE, 1985.
- [12] IEEE Standard Specifications for the Implementations of the 8×8 Inverse Discrete Cosine Transform, IEEE Standard 1180-1990, March 1991.
- [13] S. Miettens, P. de With and C. Hentschel, "New Scalable DCT Computation for Resource-Constrained Systems," Proc. of Signal Processing Systems SIPS 2001: Design and Implementation, IEEE Press, Antwerp, Belgium, pp. 285-296, 26-28 Sept. 2001.
- [14] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg and D. J. LeCall, MPEG Video Compression Standard, Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.
- [15] R. E. Morley, G. L. Engel, T. J. Sullivan and S. M. Natarajan, "VLSI Based Design of a Battery-Operated Digital Hearing Aid," Proc. ICASSP-88, pp. 2512-2515, Apr. 1988.
- [16] V. Paliouras and T. Stouraitis, "Low Power Properties of the Logarithmic Number System," Proc. 15th IEEE Symp. on Computer Arithmetic Vail, CO, pp. 229-236, 11-13 June 2001.
- [17] J. R. Sacha and M. J. Irwin, "The Logarithmic Number System for Strength Reduction in Adaptive Filtering," Intl. Symp. on Low Power Electronics and Design, Monterey Convention Center, Monterey, California, pp. 10-12, Aug. 1998.
- [18] 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, Dept. of Electrical Engineering, Washington University, St. Louis, Missouri, 1993.
- [19] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," IEEE Trans. on Computers, vol. C-24, pp. 1238-1242, Dec 1975.
- [20] J. Watkinson, The MPEG Handbook, Focal Press, Oxford, 2001.