Lossless Compression of Digital Audio

Mat Hans, Ronald W. Schafer*
Client and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-1999-144
November, 1999

Email: mat_hans@hp.com
rws@ece.gatech.edu

audio cooling, lossless compression, Internet audio

The first part of this paper surveys and classifies the best performing currently available lossless compression algorithms for stereo-CD-quality digital audio signals sampled at 44.1 kHz and quantized to 16 bits. This study finds that these algorithms appear to have reached a limit in compression that is very modest compared to what can be achieved with lossy audio coding technology. With this limit as a target, we designed a computationally efficient algorithm for lossless audio compression (which we call AudioPaK). This new lossless compression algorithm uses only a small number of integer arithmetic operations and performs as well, or better than most state-of-the-art lossless compression algorithms. The main operations of the algorithm are prediction with integer coefficients and Golomb-Rice coding. The second part of the paper presents the complete architecture of AudioPaK including suggestions on parallelizing parts of the algorithm using the MMX instruction set.

* Center for Signal & Image Processing at the School of Electrical and Computer Engineering, Georgia Institute of the Technology, Atlanta, Georgia.

While much work has been done in the area of lossy compression of speech and audio signals, relatively little has been published on lossless compression of these signals. Although lossless audio compression is not likely to become a dominating technology, it may become a useful complement to lossy compression algorithms in some applications. This is because, as we will see, lossless compression algorithms rarely obtain a compression ratio larger than 3:1, while lossy compression algorithms allow compression ratios to range up to 12:1 and higher. For lossy algorithms, the higher the compression ratio becomes, the lower the resulting final audio quality, but when the lowest possible data rate is required, lossy techniques are the only alternative. However, lossless audio coding of stereo CD quality digital audio signals sampled at 44.1 kHz and quantized to 16 bits could become an essential technology for digital music distribution over the Internet because some consumers will want to acquire the *best* possible quality of an audio recording for their high-fidelity stereo system. Lossy audio compression technologies such as MPEG or MP3 may not be acceptable for this application.

Internet resources are limited and will likely remain constrained because the demand for service continues to increase at least as fast as than the rate of growth of transmission capacity. Therefore compression will continue to be important for Internet transmission of audio signals. It is reasonable to suggest, for example, that music distribution applications will offer to the consumer highly compressed audio clips for browsing and selection. After selection, the consumer who is accustomed to audio CD quality may require access to a losslessly compressed copy of the original — a copy without any distortion due to the compression algorithm.

Music distribution over the Internet is not the only application for which lossless audio compression technology is of interest. This technology can also be used for archiving and mixing of high-fidelity recordings in professional environments. In this case, lossless compression avoids the degradations incurred in multiple encoding and decoding when using lossy compression coders. This problem arises particularly in the post production industry and occurs when the signals are modified or when different coding formats are used. Lossless audio coding is also used in the new DVD standard for storage of audio signals at higher resolution and sampling rates. [6]

The first section of this paper is a survey and a classification of the current state-of-the-art lossless audio compression algorithms (lossless coders). This study finds that lossless audio coders have reached a limit in what can be achieved for lossless compression of audio. The second section is a complete description of a new lossless audio coder called AudioPaK, which has low algorithmic complexity and performs as well or even better than most of the lossless audio coders that have been described in the literature.

1 State-of-the-Art for Lossless Audio Coders

Lossless compression is not a new problem in the general context of computing. A variety of utilities such as PkZip are available and widely used to compress text and program files for storage or transmission over the Internet. These programs are designed to be very effective for text data. For example, a PostScript version of this paper occupied 4,560,999 bytes of storage while the PkZipped version occupied only 696,041 bytes of storage – a compression ratio of 4,560,999/696,041 = 6.55. On the other hand, track 04, one of the audio files consisting of 16 bit samples used in our tests occupied 33,715,964 bytes of storage before compression and 31,573,915 bytes after compression with PkZip. In this case, the compression ratio was only 1.07. What is needed are algorithms specifically designed for digital audio files. Although lossless coding of audio has not received a great deal of attention so far, a number of different algorithms have been developed, and code for testing these algorithms has been made available in almost all cases. The compression algorithms to be discussed in this paper are listed in Table 1. This list represents all the lossless coders that we found in the literature

Name of algorithm	Authors
AudioPaK	M. Hans, R. Schafer [11]
DVD	P. Craven et al [4, 5, 6]
LTAC	M. Purat, T. Liebchen, P. Noll [19]
MUSICompress	A. Wegener [23]
OggSquish	C. Montgomery [18]
Philips	A. Bruekers, A. Oomen, R. van der Vleuten [1]
Shorten	T. Robinson [21]
Sonarc	R. Sprague [22]
WA	D. Lee [16]

Table 1 List of lossless compression algorithms discussed in this paper.

or through searching the Web. We believe that these coders represent the state-of-the-art in lossless audio compression. Shorten [21], MUSICompress [23], LTAC [19], DVD [4, 5, 6] and Philips' algorithms [1] are described in publications in the literature. Information concerning the other coders — Sonarc [22], OggSquish v98.9 [18] and WA [16] — comes from personal communications with the designers of these algorithms. AudioPaK is a new lossless audio coder described in Section 2.

1.1 Basic Principles of Lossless Compression

Figure 1 is a block diagram representation of the operations involved in compressing a single audio channel. All of the techniques studied are based on the principle of first removing redundancy from the signal and then coding the resulting signal with an efficient digital coding scheme. Each of the steps shown in Figure 1 will be discussed in general terms and in terms of specific coders.

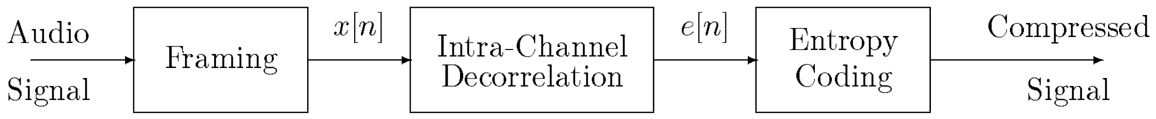

Figure 1 The basic operations in most lossless compression algorithms.

Figure 1 shows the operations for a single channel of lossless audio compression. Although the two channels of a stereo recording are generally not independent, the dependence is often weak and difficult to take into account. Therefore, the multichannel case of lossless compression has not received much attention in the literature. Generally, stereo channels are compressed separately without attempting to take advantage of the correlation that might exist between the left and right channels or by simply coding the left channel and the difference between the two channels.

1.1.1 Framing

The framing operation in Figure 1 is introduced to provide for *editibility*, an important and necessary property for most applications dealing with digital audio. It is often important to quickly and simply edit a compressed audio bit stream, and the sheer volume of data prohibits repetitive decompression of the entire signal preceding the region to be edited. Therefore, a practical solution is to divide the digital audio signal into independent frames of equal time duration. This duration should not be too short, since significant overhead may result from the header that is prefixed to each frame. This header is important, since it defines the compression algorithm parameters, which can change on a frame-to-frame basis to follow the varying characteristics of the input signal over time. Furthermore, depending on the application, this header may include additional data such as multimedia and synchronization information. On the other hand, the frame duration should not be too long, since this would limit the temporal adaptivity and would make editing of the compressed audio bit stream more difficult. The algorithms surveyed compromised with a 13 to 26 ms frame duration [1, 3, 11, 21]. This time interval translates to 576 and 1152 samples for a sampling rate of 44.1 kHz.

1.1.2 Intra-channel decorrelation

The purpose of the second stage of the typical lossless audio coder in Figure 1 is to remove redundancy by decorrelating the samples within a frame. The lossless audio coders in Table 1 use two basic approaches for intra-channel decorrelation. Most algorithms remove redundancy by some type of modified linear predictive modeling of the signal. In this approach, a linear predictor is applied to the signal samples in each block resulting in a sequence of prediction error samples. The predictor parameters therefore represent the redundancy that is removed from the signal, and the losslessly coded predictor parameters and prediction error together represent the signal in each block. A second, less common, approach is to obtain a low bit-rate quantized or lossy representation of the signal, and then losslessly compress the difference between the lossy version and the original signal. These approaches are discussed in more detail below.

Predictive Modeling: The basic principle of predictive modeling is to predict the value of a sample $x[n]$ using the preceding samples $x[n-1]$, $x[n-2]$, etc. Figure 2 is a diagram of intra-channel decorrelation using the predictive modeling scheme. If the predictor performs well,

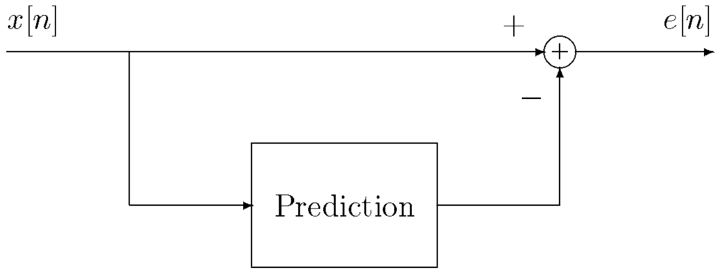

Figure 2 Prediction model for the intra-channel decorrelation block.

the prediction error $e[n]$ will be uncorrelated and therefore have a flat frequency spectrum. Likewise, $e[n]$ will on average be smaller than $x[n]$ and it will therefore require fewer bits for its digital representation.

The prediction models that are used in lossless compression of audio are based on the structure of Figure 3. This diagram depicts the equation

$$e[n] = x[n] - Q\left\{ \sum_{k=1}^{M} \hat{a}_k x[n-k] - \sum_{k=1}^{N} \hat{b}_k e[n-k] \right\}, \tag{1}$$

where $Q\{\ \}$ denotes quantization to the same word length as the original signal $x[n]$. In the diagram, $\hat{A}(z)$ and $\hat{B}(z)$ are z-transform polynomials (with quantized coefficients) rep-

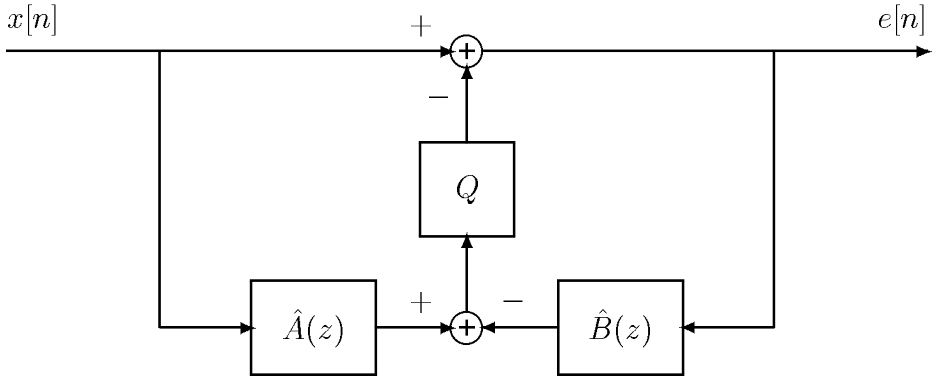

Figure 3 General structure for prediction.

resenting the feedforward $(\sum_{k=1}^{M} \hat{a}_k x[n-k])$ and feedback $(\sum_{k=1}^{N} \hat{b}_k e[n-k])$ terms in (1) respectively. If the polynomial $\hat{B}(z) = 0$, the feedback terms are omitted, and the prediction error filter has finite impulse response (neglecting the quantizer Q for the moment) and thus a predictor with only $\hat{A}(z)$ is an FIR predictor. If the feedback terms are present $(\hat{B}(z) \neq 0)$, the predictor is an infinite impulse response (IIR) predictor.

The quantization operation in Figure 3 makes the predictor a nonlinear predictor, but since the quantization is to 16 bit precision, it is reasonable to neglect the quantization in understanding the first-order effects of the predictor and in developing methods for estimating the predictor parameters. The quantization is necessary for lossless compression since we want to be able to reconstruct $x[n]$ exactly from $e[n]$ remotely and possibly on a different machine architecture. Therefore $e[n]$ is usually represented with the same fixed-point integer quantization scheme as $x[n]$ so as not to introduce new quantization levels below the least significant bit. Reconstructing $x[n]$ from $e[n]$ can be done by simply solving (1) for $x[n]$ in terms of $e[n]$, which leads to

$$x[n] = e[n] + Q\left\{\sum_{k=1}^{M} \hat{a}_k x[n-k] - \sum_{k=1}^{N} \hat{b}_k e[n-k]\right\}, \tag{2}$$

which is depicted in Figure 4.

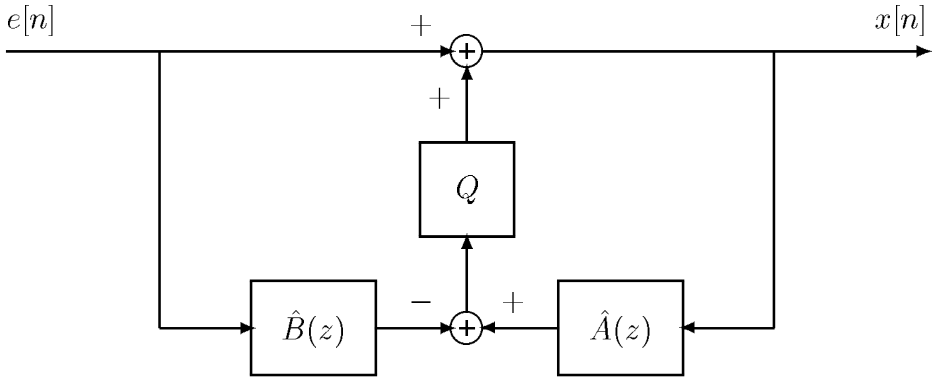

Figure 4 General structure for reconstruction.

Linear predictors are widely used in speech and audio processing [15, 20]. In most applications, an FIR predictor is used, and the coefficients of the prediction filter $\hat{A}(z)$ are determined to minimize the mean-squared prediction error. Without the quantizer and with $\hat{B}(z) = 0$ in Figure 3, the prediction coefficients can be found for the FIR case by simply solving a set of linear equations. [20] When FIR linear prediction is used in lossless audio compression, the prediction coefficients can be found by standard methods and then quantized for use in Figure 3 (with $\hat{B}(z) = 0$). As shown in Figure 4, the same coefficients are used in the reconstruction of $x[n]$ from $e[n]$. Therefore the prediction coefficients must be quantized and encoded as part of the lossless representation. Generally a new set of coefficients is determined for each frame of the signal thereby adapting the predictor to the changing local structure of the signal. The method of determination of the predictor coefficients ranges from estimation of the minimum mean-squared predictor for each block to the simple choice of a predictor from a small library of fixed predictors.

In many of the lossless audio compression algorithms in our survey, the predictor for a given block is chosen from a finite library of coefficient sets to avoid the computation required to estimate the optimum predictor coefficients. In these cases, the identity of the predictor that was used for a given frame is all that need be encoded with the compressed prediction error. This is the approach that is followed for example in the case where both feedforward and feedback terms are involved in the prediction. This is because the general solution for the minimum mean-squared predictor is much more complicated in the IIR case. For this reason, IIR prediction has not been widely used in other applications such as speech processing. In lossless compression, Craven et al [4, 5] have shown that while IIR prediction has the potential to perform better because it can match a wider range of spectral shapes than FIR prediction with the same number of coefficients, the improvement in compression performance demonstrated so far is modest at best.

In general, a minimum mean-squared predictor for audio signals will involve fractional coefficients. These must be quantized and coded as part of the lossless representation, and the predictor must be implemented with fixed-point arithmetic. However, when the prediction coefficients are integers, it is particularly simple to compute the prediction error with integer arithmetic that can easily be done exactly the same way on different computation platforms.

The following list shows the algorithms that are based on predictive modeling according to the type of prediction that is used:

- Finite impulse response (FIR) model: *Shorten [21], Sonarc [22], WA [16], Philips [1].*

- Infinite impulse response (IIR) model: *OggSquish [18], DVD [4, 5].*

- Integer coefficients FIR model (fixed or adaptive): *Shorten [21], HEAR [3], WA [16], MUSICompress [23], AudioPaK.*

The details of how the predictor model is chosen at each frame can be found in each of the above references.

Lossy Coding Model: The second approach to lossless compression is based on obtaining an approximation to $x[n]$ by some sort of lossy coding scheme that maintains the signal wave shape with no time shift. Although this approach has some potentially attractive features, we found only one algorithm that clearly used this approach. M. Purat et al. [19] proposed a method based on transform coding of the audio signal. [15] The basic idea of this method is presented in Figure 5. This algorithm uses an orthonormal transform (a discrete cosine

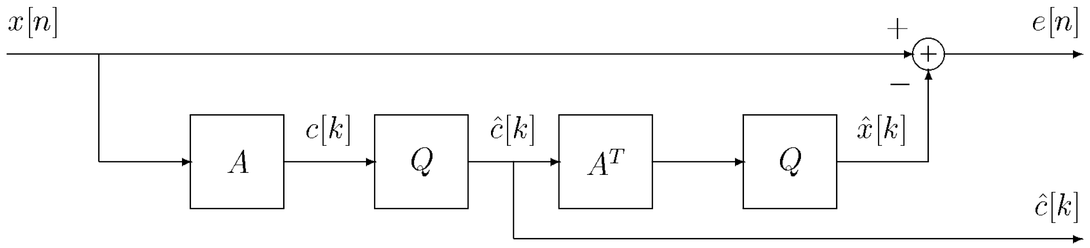

Figure 5 Details of the intra-channel decorrelation block proposed by M. Purat et al. in their Lossless Transform Audio Coder (LTAC). Both signals $\hat{c}[k]$ and $e[n]$ are entropy coded and transmitted. Q stands for integer truncation. [19]

transform) to obtain a quantized version of the signal. The cosine transform and its inverse are indicated by the matrix multiplications by A and A^T in Figure 5. The use of the cosine transform is motivated by the well known energy compaction property of the cosine transform, which permits a good reconstruction from only a few quantized values of the cosine transform. [15] The input signal $x[n]$ is transformed and quantized with an unitary quantization step size $\Delta = 1$. The resulting coefficients $\hat{c}[k]$ are then entropy coded. Because the inverse transformation followed by a dequantization step results in an approximation, $\hat{x}[n]$, of the original signal $x[n]$, the decompression steps are duplicated at the coder side to compute the residual error $e[n] = x[n] - \hat{x}[n]$. This error is entropy coded and transmitted along with the coded coefficients $\hat{c}[k]$. At the decoder, the original signal can be obtained from $x[n] = \hat{x}[n] + e[n]$ provided that the implementation of the inverse cosine transformation and quantization at the decoder is identical to that at the coder.

A potential advantage of this and other lossy-coding-based approaches is that the lossless representation contains within it a lossy lower data rate representation of the audio signal, which could serve as a "thumbnail sketch" of the signal. MUSICompress [23] also is structured to provide for a lossy compression mode of operation. This approach could be useful in situations where progressive transmission of the signal is desired or where audio signals are broadcast over the Internet to destinations with varying computational resource and data rate. [13]

1.1.3 Entropy coding

Entropy coding removes redundancy from the residual signal $e[n]$ (and the DCT coefficients in the transform-based method). In this process, no information is lost. The following three methods are used in the algorithms in Table 1: Huffman coding, run length coding, and Rice coding. The first two methods are well known and we refer the interested reader to [9] for further information. On the other hand, Rice coding, which is used in some of the coders, is less well known, so we will provide a brief summary of the method.

The Rice code is characterized by only one parameter, denoted henceforth as m. In fact, this code is the Huffman code for a Laplacian probability density function, which is found to be a good approximation for the distribution of the prediction residual samples $e[n]$ for all the intra-channel decorrelation operations discussed above [1, 3, 19, 21]. To form this code, a number is divided into three parts; a sign bit, the m low-order bits, and the remaining high-order bits. The first part of a code word is a single bit indicating the sign of $e[n]$. The second part consists of the m least-significant bits of the binary representation of the absolute value of $e[n]$. The third part of the code word is made of N consecutive zeroes, where N has the same binary representation as the yet unused most significant bits of the absolute value of $e[n]$. Finally, to end this series of zeroes, a bit set to 1 is inserted. Table 2 gives examples of Rice codes for $m = 3$.

Number	**Sign bit**	m **Lower bits**	**Number of 0's**	**Full code**
0	0	000	0	00001
18	0	010	2	0010001
-12	1	100	1	110001

Table 2 Examples of Rice codes for $m = 3$.

All the lossless audio coders presented in the literature that use the Rice code (LTAC [19], Philips [1], Shorten [21], WA [16]) define the single parameter m to be of constant value over an entire frame. This value is found by means of a full search or by estimation. Such an estimation was first given in [21] as:

$$m = \log_2(\log_e(2)E(|e[n]|)), \tag{3}$$

where $E(\,)$ is the mathematical expectation function (average).

1.1.4 Summary of Compression Algorithm Characteristics

The characteristics of the compression algorithms of this study are summarized in this section.

AudioPaK (Version 1.1) This coder is described in detail in Section 2.

DVD Standard This algorithm is based on the work of Craven et al [4, 5, 6]. It uses IIR prediction and Huffman coding for the prediction error signal.

Lossless Transform Audio Compression (Version 1.01 for DOS) LTAC is the only coder that uses transform coding for decorrelation. The Rice coding scheme is used to pack the transform coefficients and the residual errors.

MUSICompress (Version 1.2 for Windows) MUSICompress uses a predictive type computational structure where adaptively varied approximations are subtracted from the original signal to produce an error signal that is losslessly coded. The nature of the approximations is not specified in [23], so it is possible that MUSICompress should be placed in the lossy-coding-based class. MUSICompress uses a block floating-point representation and Huffman coding to code both the approximation and the residual signal.

OggSquish (Version 98.9 for Unix) The lossless audio coder, which comes with OggSquish version 98.9 uses IIR linear prediction and Huffman coding.[1]

[1]OggSquish is not exactly an audio coder, it is a DSP engine that runs any coding program the user

Philips This algorithm designed by Philips [1] uses a 10th-order FIR linear predictor along with the Rice coding scheme.

Shorten (Version 2.1 for DOS) Two commands were used: `-p 0` and `-p 10`.[2] These two commands define different coders, which differ by their intra-channel decorrelation block. The program option `-p 0` uses an FIR predictor with integer coefficients (see Section 2.2) while `-p 10` uses a 10th-order minimum mean-squared linear FIR predictor. Both coders use the Rice coding scheme to encode the prediction error signal.

Sonarc (Version 2.1h for DOS) Sonarc uses FIR linear prediction and Huffman coding. If the default arguments are used, a fixed-point integer version of Schur's algorithm is used to compute the FIR prediction coefficients. If the `-x` argument is included in the arguments then the prediction coefficients are determined by the "winner" of a contest between autocorrelation, covariance, Schur's, and Burg's algorithms.

Waveform Archiver (Version 1.1 for DOS) WA uses FIR linear predictors for the intra-channel decorrelation block. The entropy coding block uses Rice coding. WA offers 5 levels of coding: `-c1`, `-c2`, `-c3`, `-c4`, and `-c5`. The compression ratio was found to increase modestly with the level number. The setting `-c5` gives the best compression ratios with significant increase in computation time over the setting `-c1`.

---

wants [18]. For example, OggSquish is capable of running scripts that generate ADPCM, MPEG, TwinVQ, etc. The 98.9 version of OggSquish includes a lossless coder designed by the authors of OggSquish. In this article we refer to this coder as OggSquish.

[2]The commands used for Shorten were:

```
-p 0:  shorten -b 1152 -v 2 -t s16 -p 0 FILE.pcm FILEP0.shn
-p 10:  shorten -b 1152 -v 2 -t s16 -p 10 FILE.pcm FILEP10.shn
```

1.2 Performance Evaluation

With the exception of the DVD Standard, all the compression algorithms discussed so far either had executable files available for downloading or they had been previously tested on material that was also accessible to us. To compare their performance, we selected a representative set of six audio files for testing each of the algorithms. All these files have the CD format, i.e. 44.1 kHz, stereo, 16 bits, and are briefly described in the following list and in Table 3.

- Track 4 of [17] (3 min 11 s): Madonna's "Like a Virgin" song.
- Track 20 of [7] (39 s): Saxophone, low frequency musical signal.
- Track 27 of [7] (20 s): Castanets, high frequency musical signal.
- Track 48 of [7] (28 s): Voice quartet.
- Track 66 of [7] (18 s): Wind ensemble.
- Track L=R (18 s): this track was constructed using the left channel of Track 66 of [7]. The right channel is an exact copy of the left channel. Therefore, the difference between the left and right channel is zero. This track serves as a probe to allow us to determine if a given compression scheme takes advantage of inter-channel correlation.

Table 3 summarizes the original file size in bytes and the estimated first-order entropy H_0 for both left and right channels ($x[n]$).[3] Recall that H_0 is the entropy of a source where each

[3] The estimate was obtained by computing a histogram with 2^{16} bins and normalizing by the number of samples to obtain probability estimates. The entropy estimate is then the sum over all bins of $p_i \log_2 p_i$.

symbol (sample) is independent and identically distributed. For signals such as digital audio, where there is considerable sample-to-sample correlation, the entropy will be significantly lower than H_0. However, these entropy estimates, which are all less than 16 bits, suggest that some degree of lossless compression should be possible in all cases. Indeed, our results show that compressed files average about 5 bits/sample.

Track	**File size (bytes)**	**H_0 Left channel (bits per sample)**	**H_0 Right channel (bits per sample)**
Track 04	33,715,920	14.33	14.28
Track 20	6,879,600	10.86	11.03
Track 27	3,528,000	9.27	9.26
Track 48	4,939,200	11.59	11.48
Track 66	3,175,200	10.81	10.92
Track L=R	3,175,200	10.83	10.83

Table 3 File size and first-order entropy for left and right channels ($x[n]$).

Tables 4 and 5 group the lossless coders depending on the arithmetic used. Table 4 groups the integer and fixed-point arithmetic coders while Table 5 groups the floating-point arithmetic coders. AudioPaK is the only coder to use only integer arithmetic. These tables give the compression ratio

$$r = \frac{\text{original file size}}{\text{compressed file size}}$$

for the six experimental audio files. To convert these numbers into bits/sample to compare to the entropy estimates, simply divide 16 by r. For example, $r = 3$ is equivalent to 5.33 bits/sample.

At the outset, it is important to note that most of the recordings from [7] contain a significant amount of silence. In fact, we found that a compression ratio of about 1.3 may

Track	Integer arithmetic	Fixed-point arithmetic	
	AudioPaK	**MUSICompress**	**Sonarc**
Track 04	1.39	1.37	1.42
Track 20	3.12	2.93	3.13
Track 27	2.47	2.46	2.71
Track 48	2.56	2.41	2.65
Track 66	2.46	2.33	2.55
Track L=R	4.93	4.58	2.56

Table 4 Compression ratios for integer and fixed-point arithmetic lossless audio coders (we set the frame size to 1152 samples for the integer coders, and we used the default arguments for the fixed-point coders).

Track	Floating-point arithmetic					
	Shorten `-p0`	**Shorten** `-p10`	**OggSquish**	**LTAC**	**Sonarc** `-x`	**WA** `-c5`
Track 04	1.38	1.43	1.43	1.41	1.42	1.46
Track 20	3.11	2.72	3.01	3.11	3.16	3.28
Track 27	2.46	2.69	2.67	2.70	2.72	2.83
Track 48	2.54	2.32	2.53	2.65	2.69	2.77
Track 66	2.46	2.42	2.51	2.54	2.57	2.64
Track L=R	2.47	2.44	5.01	2.70	2.58	5.28

Table 5 Compression ratios for floating-point arithmetic lossless audio coders (we set the frame size to 1152 samples for Shorten and we used the default arguments for OggSquish, Sonarc, LTAC, and WA).

be easily obtained by compressing only these silent segments for Tracks 20, 27, 48, and 66.

The data in Tables 4 and 5 show several things. First note that the compression ratios obtained by the various state-of-the-art lossless audio coders for the same audio file are very similar. For example, Track 20 compression ratios are in the range 2.72 to 3.28, which is equivalent to at most a 1 bit per sample difference. For the other experimental audio files, these differences are at most: 0.85 bits per sample for Track 27, 1.12 bits for Track 48, 0.80 bits for Track 66, and 0.72 bits for Track 4.

Compression ratios associated with Track L=R (this audio file was created to probe which coders took advantage of inter-channel dependencies) show that only for AudioPaK, MUSI-Compress, OggSquish, and WA is the compression ratio twice that of Track 66, indicating that these coders attempt to take advantage of the dependence between channels in a way that works perfectly if the two channels are perfectly correlated.

Finally, these tables show that coder WA with the `-c5` argument consistently gives the best compression ratio for the 6 experimental audio files, although in most cases it performed only slightly better than other algorithms. This is one of the few coders, which could not provide real-time compression on our 166 MHz MMX Pentium machine. In general, floating point computations are not desirable for data compression. They generally are slower than integer computations, and floating point formats can vary between computational platforms.

The Philips algorithm [2] and the DVD Standard algorithm [4, 5, 6] are the only algorithms in the list for which no executable was available. We have not attempted a comparison with the DVD algorithm since it is designed for higher sampling rates and was evaluated on signals that were not available to us. However, the work on which the DVD algorithm is based [5] shows little if any advantage of the IIR predictor over an FIR predictor. In the case of the Philips algorithm, however, the compression ratio was published in [1] (frame size was set to 1024 samples) for compression of all 70 Tracks of [7]. To allow a relative comparison with the other coders we also compressed all 70 Tracks using AudioPaK. Table 6 summarizes results of this comparison. The italicized numbers were published in [1], and the other number represents the performance of AudioPaK on the entire SQAM CD. The results

Philips	**Shorten**	**OggSquish**	**AudioPaK**
3.31	*2.33*	*3.16*	2.88

Table 6 Compression ratio for complete CD [7].

in Table 6 must be interpreted with care. Since we were not able to compare the Philips compression algorithm on the individual files of our test set, we cannot draw the conclusion that this algorithm will perform uniformly better than all the other algorithms. The SQAM CD contains many simple sounds such as pure tones, single instruments, and silence which might be more easily compressed by some algorithms and not others. Other results in [1] show the Philips algorithm performing similarly to OggSquish on individual files that were unavailable for our study. These earlier individual comparisons showed significantly lower compression ratios than the 3.31 reported for the entire SQAM disk.

1.3 Conclusion of our study

In conclusion, our study of currently available lossless audio coders suggests that the current technology has reached a limit in what can be achieved for lossless compression of audio.

Furthermore, in spite of a wide range of approaches and complexities, the different algorithms perform similarly on the same audio signals with generally less than one bit per sample difference in compressed bit rate.

Therefore, a coder compressing at this limit with the least number of arithmetic operations would define the best technology for lossless audio compression. That is, the design of a lossless audio coder should now focus on *reducing algorithm complexity.* With this in mind, we designed a simple, lossless audio coder, which we called AudioPaK. This coder uses only a few integer arithmetic operations on both the coder and the decoder side. The main operations are prediction with integer coefficients and Golomb coding, which are done on a frame basis. As summarized in Tables 4, 5, and 6, our algorithm performs as well or even better than most state-of-the-art lossless coders. A complete description of AudioPaK is given in the next section.

2 AudioPaK: a low complexity lossless coder

AudioPaK, is well suited for Internet transmission because of its low instruction complexity and good compression performance. The algorithm is conveniently described in terms of the same structure (Figure 1) that was used to discuss all the other lossless compression algorithms. AudioPaK differs from the other algorithms in how the redundancy removal and coding are implemented. The details of the algorithm are discussed below. Information on the structure of the compressed bit stream is given in Appendix A.

2.1 Framing

As in the case of all the other lossless audio coders in our study, AudioPaK divides the input audio signal into independent frames. The length of a frame is a coder parameter and may be set at coding time; however, there are good practical reasons for using frame sizes that are a multiple of 192, the number of samples carried by an AES/EBU frame.[4] Table 7 shows the results for frame lengths ranging from 192 to 4608 in multiples of 192. This data shows that the compression ratios are not very sensitive to the frame length and suggests that a good length for 44.1 kHz, 16 bit audio sampled streams is 1152 samples.

Track	**192**	**576**	**1152**	**2304**	**4608**
Track 04	1.38	1.39	1.39	1.38	1.36
Track 20	3.02	3.14	3.17	3.17	3.17
Track 27	2.50	2.49	2.46	2.42	2.37
Track 48	2.48	2.55	2.56	2.56	2.56
Track 66	2.42	2.47	2.47	2.46	2.42

Table 7 Compression ratios for the left channel of experimental audio files using AudioPaK with different frame sizes (in number of samples).

2.2 Intra-Channel Decorrelation

Much of the arithmetic in the algorithms described in Section 1 is in the computation of the prediction error signal so this area is a prime target for simplification. Therefore, the intra-channel decorrelation operation uses a very simple FIR adaptive prediction method

[4]The AES/EBU is a standard interconnection digital interface established by the Audio Engineering Society (AES) and the European Broadcasting Union (EBU). It is a serial transmission format for linearly represented digital audio data, which permits transmission of two-channel digital audio information, including both audio and non audio data, from one professional audio device to another.

using only integer coefficients. This approximation was first proposed in Shorten [21] and will be discussed in detail here.

The prediction method uses the following four simple FIR predictors, each of which has only integer coefficients:

$$\begin{cases} \hat{x}_0[n] &= 0 \\ \hat{x}_1[n] &= x[n-1] \\ \hat{x}_2[n] &= 2x[n-1] - x[n-2] \\ \hat{x}_3[n] &= 3x[n-1] - 3x[n-2] + x[n-3] \end{cases}$$

These predictors are FIR predictors since they involve linear combinations of past samples. However, they also have an interesting interpretation in terms of the pth-order polynomials that they define. Recall that a $(p-1)$-order polynomial can be found that passes through the previous p data points $x[n-1], x[n-2], \ldots, x[n-p]$. This polynomial can be evaluated at the nth sample time to obtain the predicted value $\hat{x}[n]$. These polynomials and the predicted values that they produce are illustrated in Figure 6 for a typical set of previous samples.

An interesting property of these polynomial approximations is that the resulting residual signals, $e_p[n] = x[n] - \hat{x}_p[n]$, can be efficiently computed in the following recursive manner:

$$\begin{cases} e_0[n] &= x[n] \\ e_1[n] &= e_0[n] - e_0[n-1] \\ e_2[n] &= e_1[n] - e_1[n-1] \\ e_3[n] &= e_2[n] - e_2[n-1] \end{cases}$$

This permits the entire set of prediction error signals to be computed without any multiplications.[5]

[5]We show in Appendix B that this algorithm can be parallelized to take advantage of Intel's MMX SIMD architecture.

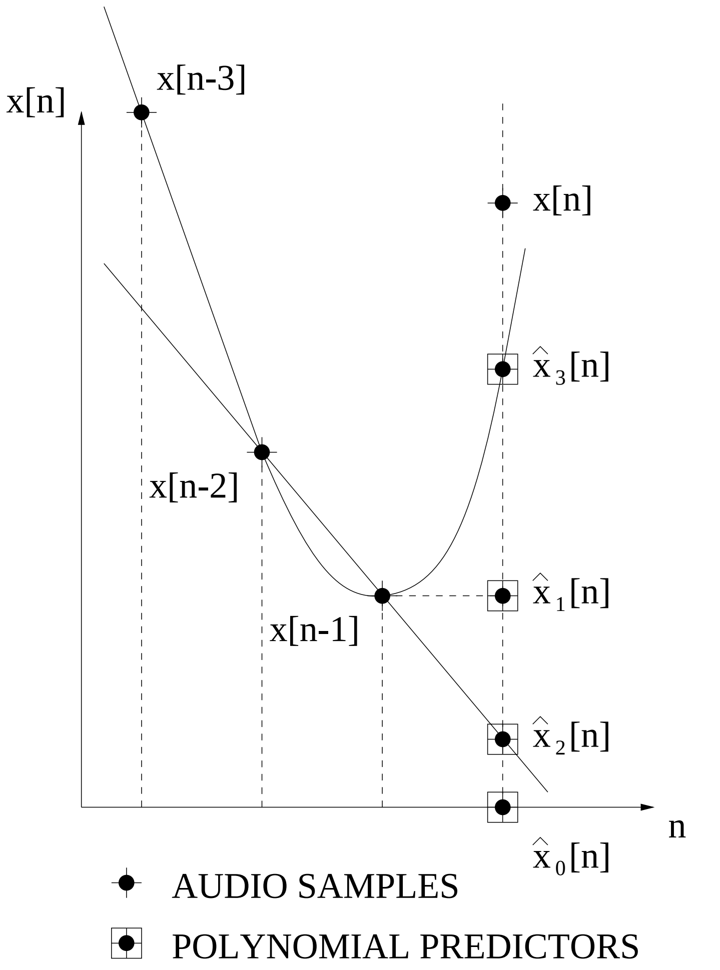

Figure 6 Polynomial approximation interpretation of the predictors used by AudioPaK for intra-channel decorrelation.

For each frame, the four prediction residuals $e_0[n], e_1[n], e_2[n], e_3[n]$ are computed at every sample in the frame and the absolute values of these residuals are averaged (summed) over the complete frame. The residual with the smallest sum of magnitudes over all samples in the frame is then defined as the best approximation for that frame, and that predictor is used for that frame. The information on which predictor was used can be coded with two bits of information, and that becomes part of the overhead information for the frame. Table 8 summarizes how many times each predictor was used in coding the experimental audio tracks.

Track	e_0	e_1	e_2	e_3	**Total frames**
Track 04	80	4948	5694	3912	14634
Track 20	621	364	1300	701	2986
Track 27	334	300	666	232	1532
Track 48	288	252	337	1267	2144
Track 66	301	30	407	641	1379

Table 8 Number of times each residual is chosen during coding of a left channel frame.

As seen in Table 8, there is no strong preference for any of the predictors; i.e., there is an advantage in varying the predictor from frame-to-frame. More light is shed on these predictors by considering their frequency responses. It is easily shown that the z-transform system function of the pth-order predictor is

$$A_p(z) = (1 - z^{-1})^p,$$

and the magnitude of the frequency response of the pth-order predictor is

$$|A_p(e^{j\omega T})| = |2\sin(\omega T/2)|^p, \tag{4}$$

where T is the sampling rate. Figure 7 shows plots of (4) for the four predictors used in AudioPaK. (The frequency axis in this plot is normalized so that half the sampling rate is at normalized frequency 1.) Note that because of the pth-order zero at $z = -1$, the predictors

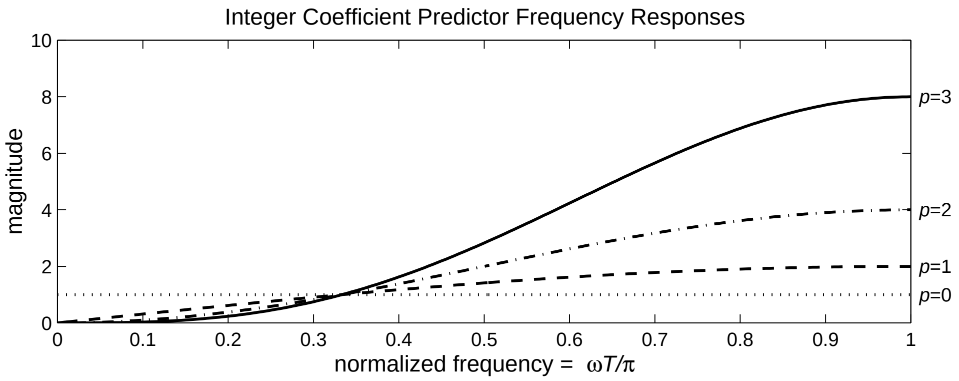

Figure 7 Frequency responses of FIR predictors used by AudioPaK for intra-channel decorrelation.

for $p = 1, 2, 3$ all have a high frequency boost that increases with increasing p. Specifically, the gain at frequency 22.05 kHz is 0, 20, 40, and 60 dB. Thus, the choice of predictor can also be thought of in terms of selecting the right amount of high frequency boost to flatten the overall spectrum of the prediction error.

The pth-order pole at $z = -1$ would be problematic in many applications of linear predictive modeling because the corresponding reconstruction system of Figure 4 would be unstable. However, in the lossless case, if we start the reconstruction system with p exact samples from the previous frame, there are no quantization errors to accumulate and the reconstruction is exact.

The question naturally arises as to how much performance is sacrificed by using simple predictors. The evaluations in Section 1 of the Shorten algorithm suggest that simplification of the prediction operation by using low-order predictors with integer coefficients does not degrade performance appreciably. Indeed, as Table 5 shows, there is little difference in performance between the simple integer predictors and the optimum 10th-order linear predictor, and, in fact, in some cases the high-order predictor performs slightly worse.

2.3 Inter-channel decorrelation

AudioPaK can take advantage of inter-channel dependencies present in stereo channel audio streams. This is done when the coder is set in stereo mode. In this mode, in addition to approximating the right stereo channel samples, we compute the polynomial approximation of the difference between the left and right channel along with the associated residuals. The smallest value among the 8 sums of absolute residuals (4 from the right channel and 4 from the difference) defines the best approximation.

Table 9 presents the estimated first-order entropy H_0 for the left error channel, the right error channel in dual mode and the right error channel in stereo mode.[6] These measures

Track	**Left error channel**	**Right error channel**	
		Dual mode	**Stereo mode**
Track 04	12.10	12.27	12.09
Track 20	6.03	6.17	6.17
Track 27	7.58	7.52	7.52
Track 48	7.18	7.16	7.16
Track 66	7.82	7.90	7.87

Table 9 First-order entropy H_0 in bits per sample for left and right error channels ($e[n]$).

show very little improvement in compression when the inter-channel decorrelation is switched on. This is not surprising since the inter-channel decorrelation operation does not take into

[6]The dual channel mode compresses the left and right channels separately.

account time delays and other differences that are common between channels of a stereo signal. Because we seek a coder with minimum algorithm complexity, this suggests using AudioPaK in dual mode.

2.4 Entropy coding

Silent frames can easily be detected with residuals $e_0[n], e_1[n]$ and efficiently entropy coded with an escape code. If the silent frame is made of all 0 value samples then the sum of $|e_0[n]|$ is zero and if the silent frame is made of values other than 0 (-1 or 1 as sometimes found) then the sum of $|e_1[n]|$ is zero (recall that $e_1[n] = x[n] - x[n-1]$).

When the frame is not of constant value, we use Golomb coding along with a mapping to reorder the integer numbers. This code is used in the new JPEG-LS lossless image compression scheme [14]. A brief description of this code follows.

Golomb Coding: Golomb codes are defined to be optimal for exponentially decaying probability distributions of *positive* integers [8]. Given a unique parameter m, the Golomb code divides a positive integer n into two parts: a binary representation of ($n \bmod m$) and a unary representation of ($\left\lfloor \frac{n}{m} \right\rfloor$). [10]

If $m = 2^k$, then the code word for n consists of the k least significant bits of n, followed by the number formed by the remaining most significant bits of n in unary representation and a stop bit. The length of this code word is $k + 1 + \left\lfloor \frac{n}{2^k} \right\rfloor$.

Because the residuals $e_i[n]$ are not all positive, a mapping $M(\,)$ is done to reorder the values as positive integers:

$$M(e_i[n]) = \begin{cases} 2e_i[n] & \text{if } e_i[n] \geq 0 \\ 2|e_i[n]| - 1 & \text{otherwise} \end{cases} \tag{5}$$

An estimate for the parameter k is given in [24] and is used in AudioPaK. It is based on the expectation $E(|e_i[n]|)$ already computed in the intra-channel decorrelation block and is

$$k = \lceil \log_2(E(|e_i[n]|)) \rceil \tag{6}$$

The parameter k is defined to be constant over an entire frame and takes values between 0 and $(b-1)$ in the case of b bit audio samples. This parameter can be estimated efficiently without any floating-point operations as follows:

```
for (k = 0, N = framesize; N < AbsError; k++, N *= 2) {NULL;}
```

where `framesize` is the number of samples in a frame and `AbsError` is the sum of the absolute values of the residual signal over the complete frame.

Context modeling: We also experimented with varying the value of the parameter k within a frame. These experiments followed the context modeling idea found in the new JPEG-LS standard. [14] Context modeling suggests defining regions (contexts) for which the statistical behavior of the residual signal $e[n]$ are similar. In the new lossless image compression standard, JPEG-LS, the contexts are built using the local gradient, which captures the level of smoothness or edginess surrounding a pixel. For example, pixels in an edge area are grouped together as are pixels in flat regions. Once these regions are defined, the coder adaptively chooses the best entropy coding parameter for each region.

We measured the maximum compression ratio possible using context modeling for AudioPaK. This maximum was found by using the best entropy coding parameter for every residual sample $e[n]$. This means setting the parameter k for each residual $e[n]$ to give the smallest code word length. Using our experimental audio files we found a maximum compression ratio improvement ranging from 6 to 10%. In order to achieve this rather small improvement, it would be necessary to infer the context from the signal data at a computational cost that is not justified by the savings in bit rate.

2.5 Arithmetic complexity

The number of integer additions and multiplications per second that AudioPaK requires in dual mode for a 44.1 kHz, stereo, 16 bit audio stream is summarized in Table 10.

Coder	0.8 MOPS
Decoder	from 0.35 to 0.8 MOPS

Table 10 Arithmetic complexity for AudioPaK in dual mode. MOPS stands for million operations per second. Note that the MOPS for the decoder depends on the predictor used most frequently.

2.6 Bit rate variability

Because of the non-stationary nature of audio signals, the compression scheme has a variable bit rate. Figure 8 illustrates this variability. In this figure the compression ratios over frames number 3550 and 3599 for the left channel of Track 04 of [17] are presented. Also depicted are the minimum and maximum sample values in each frame along with the mean and variance value over each frame. Note the relationship between the variance (which also defines the energy) and the compression ratio. As the energy of a frame increases, the compression ratio decreases.

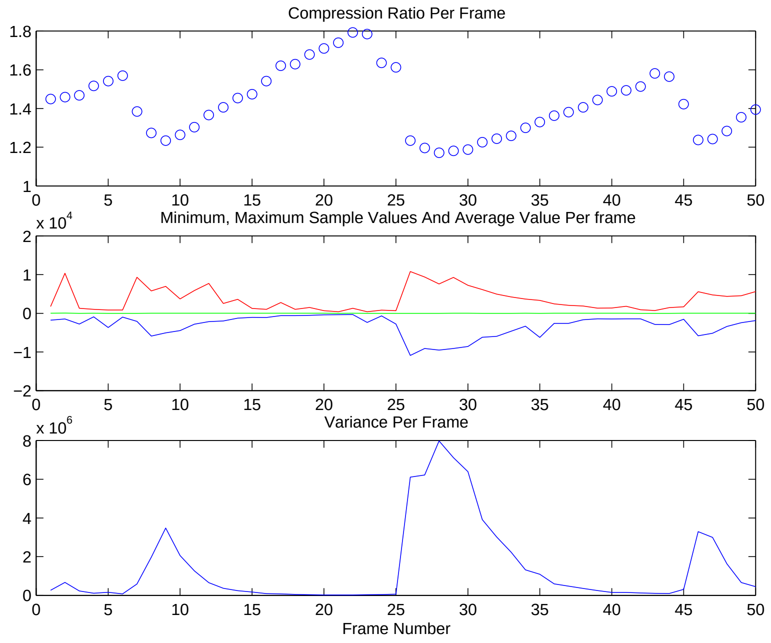

Figure 8 Variable bit rate nature of coder.

In some applications it may be of interest to have a smoothed peak rate instead of a strong fluctuation in instantaneous data rate. A solution to this problem was proposed by Craven et al. in [5] who suggested inserting buffers at both the coder and the decoder sides.

3 Conclusion

Lossless audio compression is likely to play an important part in music distribution over the Internet, DVD audio, digital audio archiving, and mixing. Because common lossless compression utilities such as PkZip do a poor job of compressing audio streams, a different technology adapted to these signals is required.

All current state-of-the-art lossless audio compression algorithms are built around removal of local dependency of the audio samples. As discussed in this paper, these algorithms have reached a limit in compression that is very modest compared to lossy compression technology. Using the basic principles of framing, prediction, and entropy coding, we designed a new coder – AudioPak – to be implemented using only a small number of integer arithmetic operations. As shown in Tables 4 and 5, AudioPaK performs as well or better than most state-of-the-art coders.

Acknowledgments

The authors wish to thank Dr. Fred Kitson and Hewlett-Packard Laboratories for generous support and encouragement of this work. Also, we wish to thank Marcelo Weinberger and Gadiel Seroussi (main designers of JPEG-LS) from the Visual Computing Department at Hewlett-Packard Laboratories, Dennis Lee (author of WA), Christopher Montgomery (author of OggSquish), Richard Spraque (author of Sonarc), Al Wegener (author of MUSICompress), and Roberta Eklund for their time and contribution.

A Structure of compressed file header and frames

While designing AudioPaK, we defined a structure for the compressed files. These included a file header and the compressed frames. The header indicates:

- the coding mode (single, stereo or dual channel) defined using 2 bits;

- the size of the frame in samples, defined with 16 bits;[7]

- the number of zero value samples appended in the last frame, defined with 16 bits;

- and in case of stereo or dual channel modes, a 1 bit flag stating if the original file contained more left than right samples.

After this header, the frames are coded separately. In the single channel mode the bit syntax depends on whether or not the frame is constant. This syntax is as follows:

- If the frame is constant then:

$$\begin{cases} \text{First bit set to 1.} \\ \text{16 next bits define the first sample.} \end{cases}$$

- If the frame is not constant then:

$$\begin{cases} \text{First bit set to 0.} \\ \text{Two next bits define which polynomial predictor was used (0, 1, 2, or 3).} \\ \text{4 bits define the Golomb parameter } k. \\ 16p \text{ bits define the first } p \text{ samples of the frame (p is the polynomial order).} \\ \text{Remaining bits define Golomb entropy codes for residual signal.} \end{cases}$$

In the case of stereo or dual channel modes, two bits are added at the beginning of a single channel frame header syntax. These bits define what channel the frame encapsulates and are set to 00 for left channel, 01 for right channel, 10 for the difference between left and right channel (11 is not used).

[7]We chose little-endian format to allow portability of the compressed file over different machine architecture.

B Intra-channel decorrelation implementation

AudioPaK's intra-channel decorrelation algorithm can be parallelized. The core of the intra-channel decorrelation block is the following `for(;;)` loop

```
for (i=1; i<framesize; i++) {
    e[i]  = x[i] - x[i-1];
    abs_sum += abs(e[i]);
}
```

which can be unrolled for Intel's MMX SIMD architecture as follows:[8]

```
for (i=1; i<framesize; i+=4) {
    e[i]    = x[i]   - x[i-1];
    e[i+1]  = x[i+1] - x[i];
    e[i+2]  = x[i+2] - x[i+1];
    e[i+3]  = x[i+3] - x[i+2];
    abs_sum1 += abs(e[i]);
    abs_sum1 += abs(e[i+1]);
    abs_sum2 += abs(e[i+2]);
    abs_sum2 += abs(e[i+3]);
}
abs_sum = abs_sum1 + abs_sum2;
```

The MMX instructions inside this loop can be:

```
for (i=1; i<framesize; i+=4) {
    /*
      Register MM0 contains four 16 bit words
        x[i+3] | x[i+2] | x[i+1] | x[i]
      and MM1 contains
        x[i+2] | x[i+1] | x[i]   | x[i-1]
     */
    PSUBW     MM0, MM1

    /*
      Now MM0 contains
```

[8]The MMX study summarized here assumes the audio samples to have 16 bit resolution. We only counted the arithmetic operations for a simple cycle estimate. We assumed the memory reads and writes for both C and MMX code to be similar.

```
          e[i+3] | e[i+2] | e[i+1] | e[i]
        Assume MM4 and MM5 contain constants
          0 | 0 | 0 | 0 and 1 | 1 | 1 | 1
       */
      PCMPGTW   MM4, MM0 ; if one of the four words in MM0
                         ; is negative then corresponding
                         ; word in MM4 will be equal to -1
                         ; (0XFFFF), otherwise it will be 0.
      PADDW     MM5, MM4 ; if one of the four words in MM4
                         ; is -1 then corresponding word
                         ; in MM5 will be 0, otherwise 1.
      PADDW     MM5, MM4 ; if one of four words in MM0 is
                         ; negative then corresponding word
                         ; in MM5 will be -1, otherwise 1.
      PMADDWD   MM0, MM5 ; MM0 will contains 32 bit results
                         ; of abs(e[i+3])+ abs(e[i+2]) and
                         ; abs(e[i+1])+ abs(e[i]).
      PADDD     MM7, MM0 ; MM7 will contain 32 bit results
                         ; of abs_sum2 and abs_sum1
  }
  abs_sum = abs_sum1 + abs_sum2;
```

Assuming a 1 cycle latency for the `abs( )` macro call, the inside of the C unrolled `for(;;)` loop estimates to 12 cycles (4 macro calls and 8 additions) while the above MMX code estimates to 6 cycles. These estimates suggest a factor 2 reduction in cycles associated to arithmetic operations.

References

[1] A. Bruekers, A. Oomen, and R. van der Vleuten. Lossless Coding for DVD Audio. *101st AES Convention, Los Angeles*, 4358, November 1996.

[2] A. Bruekers, A. Oomen, and R. van der Vleuten. Stated during 101st AES Convention, Los Angeles, presentation of [1]. Philips, November 1996.

[3] C. Cellier, P. Chenes, and M. Rossi. Lossless Audio Data Compression for Real Time Applications. *95th AES Convention, New York*, 3780, October 1993.

[4] P. Craven and M. Gerzon. Lossless Coding for Audio Discs. *Journal of Audio Engineering Society*, 44(9):706–720, September 1996.

[5] P. Craven, M. Law, and J. Stuart. Lossless Compression Using IIR Prediction Filters. *102nd AES Convention, Munich*, 4415, March 1997.

[6] DVD Forum. "DVD Specifications for Read-Only Disc" Part 4 AUDIO SPECIFICATIONS, Version 1.0, March 1999.

[7] European Broadcasting Union. SQAM (Sound Quality Assessment Material). CD # 422 204–2, 1988.

[8] R.G. Gallager and D.C. Van Voorhis. Optimal Source Codes for Geometrically Distributed Integer Alphabets. *IEEE Transactions on Information Theory*, March 1975.

[9] A. Gersho and R. Gray. *Vector Quantization And Signal Compression.* Kluwer Academic, 1992.

[10] S.W.Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, July 1966.

[11] M. Hans. "Optimization of Digital Audio for Internet Transmission". Ph.D. Thesis, Georgia Institute of Technology, May, 1998.

[12] M. Hans and R. W. Schafer. "AudioPaK — An Integer Arithmetic Lossless Audio Codec". In *Data Compression Conference*, 1998.

[13] M. Hans and R. W. Schafer. "An MPEG Audio Layered Transcoder." *105th AES Convention, San Francisco*, 4812, September, 1998.

[14] ISO/IEC JTC1/SC29/WG1. JPEG-LS: emerging lossless/near-lossless compression standard for continuous-tone images JPEG Lossless. Information available at `http://www.disc.org.uk/public`, 1997.

[15] N. Jayant and P. Noll. *Digital Coding of Waveforms, Principles and Applications to Speech and Video.* Prentice-Hall, 1984.

[16] D. Lee. Personal communications on Waveform Archiver (WA). Information available at `http://www.ecf.utoronto.ca/~denlee/wa_perf.htm`, August 1997.

[17] Madonna. The Immaculate Collection. CD # 26440–2.

[18] C. Montgomery. Personal communications on OggSquish. Information available at `http://www.xiph.com`, August 1997.

[19] M. Purat, T. Liebchen, and P. Noll. Lossless Transform Coding of Audio Signals. *102nd AES Convention, Munich*, 4414, March 1997.

[20] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[21] T. Robinson. SHORTEN: Simple lossless and near-lossless waveform compression. Technical Report 156, Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ UK, December 1994.

[22] R. Spraque. Personal communications on Sonarc. Information available at `sonarc@compuserve.com`, August 1997.

[23] A. Wegener. MUSICompress: Lossless, Low-MIPS Audio Compression in Software and Hardware. In *Proceedings of the International Conference on Signal Processing Applications and Technology*, September 1997. Information available at `http://members.aol.com/sndspace`.

[24] M. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm. In *Data Compression Conference*, 1996. Information available at `http://www.hpl.hp.com/loco`.