Lossless Audio Coding

Mathieu Hans and **Ronald Schafer**
Technical Report (CSIP TR-97-07)
Center for Signal & Image Processing (CSIP)
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA
email:{hans,rws}@ece.gatech.edu

November 30, 1997

Abstract

This Technical Report surveys and classifies the currently available and best performing lossless audio codecs. Our study suggests that these codecs reached a limit in compression that is very modest compared to lossy audio coding technology. Assuming this limit to be near the theoretical entropy, we designed a simple, lossless audio codec—AudioPaK—, which uses only a few integer arithmetic operations and performs as well, or better than most state-of-the-art lossless codecs. The main operations of this codec are polynomial prediction and Golomb coding. These operations are done on a frame basis. The complete architecture of AudioPaK is presented.

1 Why Lossless Audio Coding ?

Lossless audio coding of stereo CD quality digital audio signal sampled at 44.1 KHz and quantized on 16 bits will become an essential technology for digital music distribution over the Internet because 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 ISO MPEG or Dolby AC-3 may not be acceptable for this application.

Lossless audio compression will not become a dominating technology, but it will complement the lossy compression algorithms. As we will see, lossless compression algorithms rarely obtain a compression ratio larger than 3:1. On the other hand, lossy compression algorithms allow compression ratios to range from 4:1 up to 40:1 and higher. Obviously, for lossy algorithms, the higher the compression ratio becomes the lower the resulting final audio quality.

Because the Internet resources are and will stay constrained (especially for bandwidth) it is reasonable to suggest that music distribution applications will offer to the consumer highly compressed versions of audio clips for browsing and selection. After selection, the consumer who is accustomed to audio CD quality should have access to a losslessly compressed copy of the original — a copy without any distortion coming from the compression algorithm.

Music distribution over the web is not the only application for which lossless audio compression technology is of interest. This technology can also be used for the following:

- Archiving and mixing of high-fidelity recordings in professional environments. This addresses the growing concern around multiple coding when using lossy compression codecs. This problem applies particularly to the post production industry and occurs when the signals are processed or different coding formats are used.

- Future high-density CD formats, such as DVD audio. Current proposals, e.g. [7] suggest using lossless technology to preserve sound quality and to convey more precision on more channels.

Section 2 of this report is a survey and a classification of current state-of-the-art lossless audio codecs. This study suggests that these codecs reached a limit in what can be achieved for lossless compression of audio. Section 3 is a complete description of a simple, lossless audio codec called AudioPaK, which has low algorithmic complexity and performs as well, or even better than most lossless audio codecs.

2 State-of-the-Art Lossless Audio Coders

Lossless compression of data is not a new technology. It is used by well known compression utilities such as PkZip, compress, or gzip to reduce the size of text and binary files. This compression is *lossless* because the files after decompression are identical to the originals.

Unfortunately, these utilities, which mostly use a variant of Lempel and Ziv's algorithms [1], do not succeed in compressing audio data very well. While most text files are compressed with factors greater than 2:1, audio files are hardly compressed at all. Table 1 presents typical compression ratios obtained with the PkZip application.

The compression utilities based on Lempel and Ziv's algorithm replace a group of letters with a pointer to where they have occurred earlier in the text. For audio files, letters are replaced with audio samples. Unfortunately, recurring sequences of the same audio samples are very rare, so compression is very low.

Type of File	Compression Ratio
44.1 kHz, 16-bit Mono PCM File	1.07
Latex file of this report	3.27

Table 1 Typical ratios of original file size over compressed file size obtained with the utility PkZip for an audio file and a text file.

Furthermore, these compression utilities overlook an important signal characteristic — the strong dependence between neighboring audio samples. This is why state-of-the-art lossless audio coders all include a decorrelation stage in their algorithm to reduce this statistical dependency. We will see that the following two types of decorrelation stages are found within these audio coders: predictive modeling and transform coding. The decorrelated signal is losslessly compressed (also referred to in the literature as "packed") using entropy coding methods. Practically, all state-of-the-art coders use at least one of the following methods: Huffman coding, run length coding, or Rice coding.

Figure 1 The three major blocks defining a state-of-the-art lossless audio coder for a single digital audio channel.

Figure 1 is a block diagram representation of the operations involved in compressing a *single* audio channel. The multi-channel case has not received much attention in the literature. Generally, stereo channels are compressed separately without taking advantage of the existing correlation between the left and right channels or by simply coding the left channel and the difference between the two channels.

2.1 Framing

The framing operation 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 of edition. Therefore, a practical solution is to divide the audio signal into frames. Hence, a framing operation is introduced as the first operation in the coder 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 prefixed header added to each frame. This header is important, since it defines the compression algorithm

parameters, which can change on a frame basis to follow the varying statistics of the input signal over time. Further, 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, this would make editing of the audio compressed bit stream more difficult. As a good compromise, state-of-the-art codecs use a 13 to 26 ms frame duration [2, 4, 10, 17], which translates to 576 and 1152 samples for a sampling rate of 44.1 kHz.

2.2 Intra-Channel Decorrelation

The second block of a lossless audio coder decorrelates the samples within an audio frame. Both predictive modeling and transform coding are used for this purpose.

The following is a classification of state-of-the-art lossless audio codecs that we found in the literature. This classification is based on the method used by the codecs for their intra-channel decorrelation block.

Predictive modeling

Coding with linear prediction

- Finite Impulse Response model (FIR): *Shorten [17], Sonarc [18], WA [13], Philips [2].*
- Infinite Impulse Response model (IIR): *OggSquish [15], Craven et al. [5, 6].*

Coding with approximation

- Polynomial approximation (fixed or adaptive): *Shorten [17], HEAR [4], WA [13], MUSICompress [21], AudioPaK.*

Transform coding

Orthonormal transforms *M. Purat et al. [16].*

Let us briefly present these decorrelation methods.

2.2.1 Predictive Modeling

The principle is to predict the value of a new 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.

As suggested in the above classification of state-of-the-art lossless audio codecs, two predictive models are used. One model is defined with linear predictors (FIR and IIR) and the other is defined with approximations (polynomial).

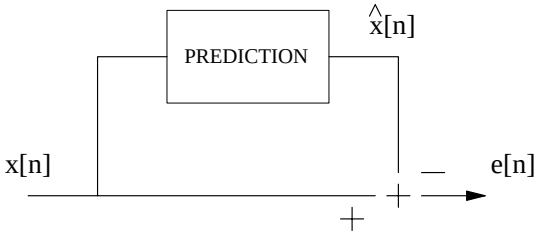

Figure 2 Prediction model for the intra-channel decorrelation block.

For example, in the case of the finite impulse response model, the predicted value $\hat{x}[n]$ of the new sample $x[n]$ is defined by

$$\hat{x}[n] \quad = \quad \sum_{k=1}^{p} a_k x[n-k]$$

where coefficients a_k are set to minimize the residual signal $e[n] = x[n] - \hat{x}[n]$.

Linear predictors are commonly used in speech and audio processing, and for more information the interested reader may want to refer to [11], for example. As for the polynomial prediction model, an example will be given in the following when presenting AudioPaK.

2.2.2 Transform Coding

The use of transform coding is not very common in lossless compression. In fact, it is used in only one codec proposed by M. Purat et al. [16]. The details of the intra-channel decorrelation operation are presented in Figure 3. The proposed coder uses an

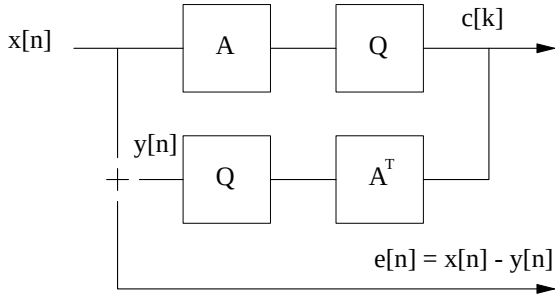

Figure 3 Details of the intra-channel decorrelation block proposed by M. Purat et al. in their Lossless Transform Audio Codec (LTAC). Both signals $c[k]$ and $e[n]$ are entropy coded and transmitted. Q stands for integer truncation.

orthonormal transform (a Discrete Cosine Transform) to reduce the statistical dependencies between audio samples. The input signal $x[n]$ is transformed and quantized. The resulting coefficients $c[k]$ are then entropy coded. Because the inverse transformation followed by a dequantization step results in an approximation, $y[n]$, of the original

signal $x[n]$, the decompression steps are duplicated at the coder side to compute the residual error $e[n] = x[n] - y[n]$. This error is entropy coded and transmitted along with the coded coefficients $c[k]$.

2.3 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. Three methods are used in the known codecs: 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, let us briefly summarize the Rice coding scheme.

The Rice code has the interesting property of being characterized by only one parameter, 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 residual $e[n]$ for all the intra-channel decorrelation operations discussed above [2, 4, 16, 17]. To form this code, a number is divided into 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 $m = 3$.

All the lossless audio codecs presented in the literature that use the Rice code 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 [17] as

$$m = \log_2(\log_e(2)E(|e[n]|))$$

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

AudioPaK, our lossless audio codec, uses the Golomb code that is slightly different from the Rice code. This code will be explained when describing the codec.

Number	Sign Bit	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 $m = 3$.

2.4 What to Expect from these Codecs

Tables 3 and 4 group the codecs depending on the arithmetic used. Table 3 groups the integer and fixed-point arithmetic codecs while Table 4 groups the floating-point arithmetic codecs. AudioPaK is the only codec to use integer arithmetic.

	Integer Arithmetic	Fixed-point Arithmetic	
Audio File	**AudioPaK**	**MUSICompress**	**Sonarc**
Track 04 of [14]	1.39	1.37	1.42
Track 20 of [20]	3.12	2.93	3.13
Track 27 of [20]	2.47	2.46	2.71
Track 48 of [20]	2.56	2.41	2.65
Track 66 of [20]	2.46	2.33	2.55
Track L=R	4.93	4.58	2.56

Table 3 Compression ratios for state-of-the-art integer and fixed-point arithmetic lossless audio codecs (we set the frame size to 1152 samples for the integer codecs, and we used the default arguments for the fixed-point codecs).

	Floating-point Arithmetic					
Audio File	**Shorten** `-p0`	**Shorten** `-p10`	**OggSquish**	**LTAC**	**Sonarc** `-x`	**WA** `-c5`
Track 04 of [14]	1.38	1.43	1.43	1.41	1.42	1.46
Track 20 of [20]	3.11	2.72	3.01	3.11	3.16	3.28
Track 27 of [20]	2.46	2.69	2.67	2.70	2.72	2.83
Track 48 of [20]	2.54	2.32	2.53	2.65	2.69	2.77
Track 66 of [20]	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 4 Compression ratios for state-of-the-art floating-point arithmetic lossless audio codecs (we set the frame size to 1152 samples for Shorten and we used the default arguments for OggSquish, Sonarc, LTAC, and WA).

These tables give the compression ratio r

$$r = \frac{\text{Original File Size}}{\text{Compressed File Size}}$$

for 6 experimental audio files. 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 5.

Audio File	File Size (bytes)	H_0 Left channel (bits per sample)	H_0 Right channel (bits per sample)
Track 04 of [14]	33,715,920	14.33	14.28
Track 20 of [20]	6,879,600	10.86	11.03
Track 27 of [20]	3,528,000	9.27	9.26
Track 48 of [20]	4,939,200	11.59	11.48
Track 66 of [20]	3,175,200	10.81	10.92
Track L=R	3,175,200	10.83	10.83

Table 5 File size and first-order entropy for left and right channels ($x[n]$) of the 6 experimental audio files.

- Track 4 of [14] (3 min 11 s): Madonna's "Like a Virgin" song.
- Track 20 of [20] (39 s): Saxophone, low frequency musical signal.
- Track 27 of [20] (20 s): Castanets, high frequency musical signal.
- Track 48 of [20] (28 s): Voice quartet.
- Track 66 of [20] (18 s): Wind ensemble.
- Track L=R (18 s): this track was constructed using the left channel of Track 66 of [20]. The right channel is an exact copy of the left channel. Therefore, the difference between the left and right channel is zero.

Table 5 summarizes the original file size in bytes and the estimated first-order entropy H_0 for both left and right channels ($x[n]$). H_0 is the theoretical lower bound for the number of bits per sample if independence between samples is assumed.

It is important to note that most of the recordings from [20] contain a significant amount of silence. In fact, we showed that a compression ratio of about 1.3 may be easily obtained by compressing only these silent segments for Tracks 20, 27, 48, and 66.

The compression algorithms presented in this report are Shorten [17], MUSICompress [21], Sonarc [18], OggSquish v98.9 [15], LTAC [16], WA [13], Philips [2], and AudioPaK. Shorten, MUSICompress, LTAC, and Philip's algorithms are described in the literature. Information concerning the other codecs — Sonarc, OggSquish v98.9, and WA — come from personal communications with the designers of these algorithms. As for AudioPaK, it is a lossless audio codec described in Section 3.

A brief description of these codecs is given here.

Shorten (Version 2.1 for DOS) Two commands were used: `-p 0` and `-p 10`.[1] These two commands define different codecs, which differ by their intra-channel decorrelation block. `-p 0` uses a polynomial approximation method and `-p 10` a linear prediction method using a 10th-order FIR filter. Both codecs use the Rice coding scheme.

MUSICompress (Version 1.2 for Windows) MUSICompress codes with approximation. It uses a block floating-point representation and the Huffman coding scheme to code both the approximation and the residual 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 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.

OggSquish (Version 98.9 for Unix) The lossless audio codec, which comes with OggSquish version 98.9 uses IIR linear prediction and Huffman coding.[2]

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.

Waveform Archiver (Version 1.1 for DOS) WA uses a polynomial and linear predictor 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 (as well as the time to encode) increases with the level number. `-c5` gives the best compression ratios.

Philips This algorithm designed by Philips [2] uses a 10th-order FIR linear predictor along with the Rice coding scheme. This is the only codec in the list for which no executable was available. This codec does not allow real-time compression [3]. To allow comparison with the other codecs we compressed the complete 70 Tracks of [20] using AudioPaK. Table 6 summarizes the compression ratios.

AudioPaK (Version 1.1) This codec is described in detail in Section 3.

---

[1] 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
```

[2] OggSquish is not exactly an audio coder per se, it is a DSP engine that runs any coding program the user wants [15]. For example, OggSquish is capable of running scripts that generate ADPCM, MPEG, TwinVQ, etc. The 98.9 version of OggSquish includes a lossless codec designed by the authors of OggSquish. In this article we refer to this codec as OggSquish.

Lossless Audio Codec	Philips	AudioPaK
Entire CD [20]	3.31	2.88

Table 6 Compression ratio for complete CD [20]. Philips is a floating-point arithmetic codec. AudioPaK is an integer arithmetic codec. The compression ratio for Philips was published in [2] (frame size was set to 1024 samples).

It is noteworthy that the compression ratios obtained by the various state-of-the-art lossless audio codecs for a 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 codecs took advantage of inter-channel dependencies) show that only for AudioPaK, MUSICompress, OggSquish, and WA is the compression ratio twice that of Track 66, indicating that these coders take advantage of the dependence between channels.

Finally, these tables show that codec WA with the `-c5` argument gives the best compression ratio for the 6 experimental audio files. However, this is one of the few codecs, which could not provide real-time compression on our 166 MHz MMX Pentium machine.

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

If this is true, a codec compressing at this limit with the least number of arithmetic operations will define the best technology for lossless audio compression. That is, the design of a lossless audio codec should now focus on *reducing algorithm complexity*.

We designed a simple, lossless audio codec, that we called AudioPaK, which uses only a few integer arithmetic operations on both the coder and the decoder side. The main operations of this codec are polynomial prediction and Golomb coding, and are done on a frame basis. As summarized in Tables 3, 4, and 6, our coder performs as well, or even better than most state-of-the-art lossless codecs. A complete description of this codec is given in the next Section.

3 AudioPaK: An Integer Arithmetic Lossless Audio Codec

3.1 Framing

As for all other state-of-the-art lossless audio codecs, AudioPaK divides the input audio signal into independent frames. The length of a frame is a codec parameter and may be set at coding time; however, there are good practical reasons for using frame sizes multiple of 192, the number of samples carried by an AES/EBU frame. Table 7 suggests that the optimal length for 44.1 kHz, 16 bit audio sampled streams is 1152 samples.

3.2 Intra-Channel Decorrelation

The intra-channel decorrelation operation uses a very simple adaptive polynomial approximation method. This approximation was first proposed in [17].

The polynomial coefficients are those specified by fitting a p-order polynomial to the last p data points $x[n-1], x[n-2], \ldots, x[n-p]$. We consider four approximations:

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

Figure 4 graphically depicts these four approximations.

Frame Size	**192**	**576**	**1152**	**2304**	**4608**
Track 04 of [14]	1.38	1.39	1.39	1.38	1.36
Track 20 of [20]	3.02	3.14	3.17	3.17	3.17
Track 27 of [20]	2.50	2.49	2.46	2.42	2.37
Track 48 of [20]	2.48	2.55	2.56	2.56	2.56
Track 66 of [20]	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).

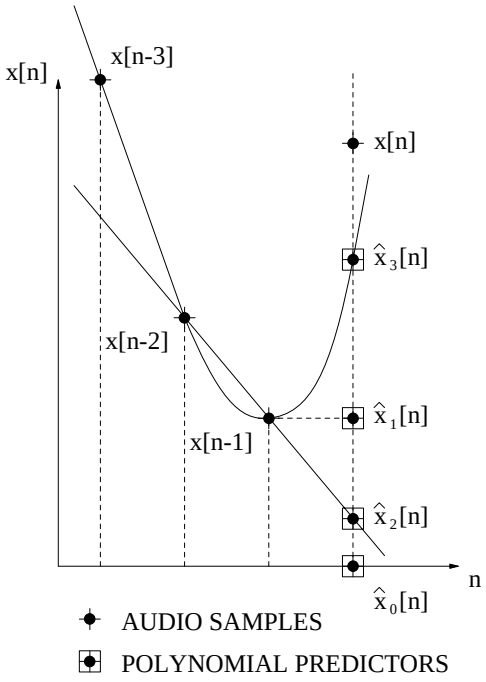

Figure 4 This figure represents the four polynomial approximations of $x[n]$ considered by AudioPaK's intra-channel decorrelation block.

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

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

For each frame, the four residuals $e_0[n], e_1[n], e_2[n], e_3[n]$ are computed as well as the sums of the absolute values of these residuals over the complete frame. The residual with the smallest sum magnitude is then defined as the best approximation. Table 8 summarizes how many times each residual was used in coding the experimental audio tracks.

We show in the appendix that this algorithm can be easily parallelized to take advantage of the new Intel's MMX SIMD architecture.

3.3 Inter-Channel Decorrelation

AudioPaK can take advantage of inter-channel dependencies present in stereo channel audio streams. This is done when the codec is set in stereo mode. In this mode, in addition to approximating the right stereo channel samples, we compute the approxi-

Residual	e_0	e_1	e_2	e_3	Total Frames
Track 04 of [14]	80	4948	5694	3912	14634
Track 20 of [20]	621	364	1300	701	2986
Track 27 of [20]	334	300	666	232	1532
Track 48 of [20]	288	252	337	1267	2144
Track 66 of [20]	301	30	407	641	1379

Table 8 Number of times each residual was chosen during coding of the left channel for the experimental audio files.

mation 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[3] and the right error channel in stereo mode. These measures show very little improvement in compression when the inter-channel decorrelation is switched on. This is not surprising knowing how simply the inter-channel decorrelation block is defined. Because we seek a codec with minimum algorithm complexity, this suggests using AudioPaK in dual mode.

H_0	Left Error Channel	Right Error channel	
		Dual Mode	Stereo Mode
Track 04 of [14]	12.10	12.27	12.09
Track 20 of [20]	6.03	6.17	6.17
Track 27 of [20]	7.58	7.52	7.52
Track 48 of [20]	7.18	7.16	7.16
Track 66 of [20]	7.82	7.90	7.87

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

3.4 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 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.

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

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 [12]. 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 writes a positive integer n into two parts: a binary representation of ($n \bmod m$) and a unary representation of ($\left\lfloor \frac{n}{m} \right\rfloor$) [19].

If m is a power of 2, $m = 2^k$, then the code word for n consists of 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:

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

An estimate for the parameter k is given in [22] 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

$$k \quad = \quad \lceil \log_2(E(|e_i[n]|)) \rceil$$

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.

Additionally, we experimented with varying the value of parameter k within a frame. These experiments followed the context modeling idea found in the new JPEG-LS standard [12].

Context Modeling Context modeling suggests defining regions (contexts) for which the statistical behavior of the residual signal $e[n]$ are similar. In the new 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 such improvements, further work will be required to define contexts that can be inferred from the data.

3.5 Arithmetic Complexity

The number of integer operations per second for AudioPaK 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 number of operations for the decoder depends on the predictor used.

3.6 Structure of Compressed File Header and Frames

We defined a file header for AudioPaK's compressed files to indicate the following:

- The coding mode (single, stereo or dual channel) defined using 2 bits.
- The size of the frame in samples, defined with 16 bits.[4]
- The number of zero value samples appended in the last frame, defined with 16 bits.
- 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

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

[4]We chose Little-Endian format to allow portability of the compressed file over different machine architecture.

- If the frame is not constant then

  First bit set to 0.
  Two next bits define which polynomial predictor was used (0, 1, 2, or 3).
  $\log_2(b)$ bits define the Golomb parameter k.
  $p \times b$ bits define the first p samples of the frame (p is the polynomial order).
  Remaining bits define Golomb entropy codes for residual signal.

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).

3.7 Bit Rate Variability

Because of the non-stationary nature of audio signals, the compression scheme has a variable bit rate. Figure 5 illustrates this variability. In this figure the compression ratios over frames number 3550 and 3599 for the left channel of Track 04 of [14] 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.

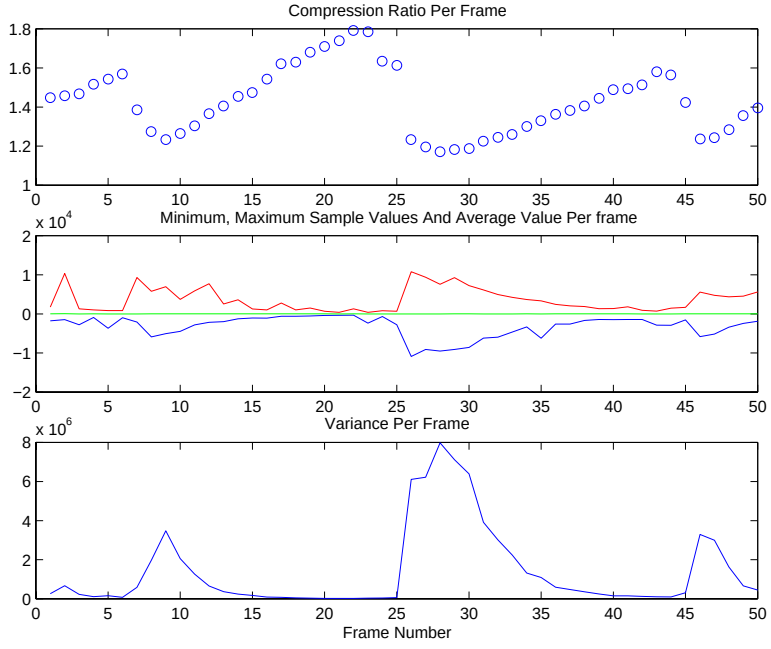

Figure 5 Variable bit rate nature of codec.

In some applications it may be of interest to have a smoothed peak rate instead of a strong fluctuation in instantaneous data rate. A good solution to this problem was proposed by Craven et al. in [6] who suggested inserting buffers at both the coder and the decoder sides. This solution can be included within AudioPaK.

4 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 codecs are built around one important signal characteristic: local dependency of audio samples. As we report in this paper, these codecs reached a limit in compression, which is very modest compared to the lossy technology. Assuming this limit to be near the theoretical entropy we designed a simple lossless audio codec — AudioPaK — using only a small number of integer arithmetic operations. AudioPaK performs as well, or better than most state-of-the-art codecs. The complete architecture of this codec was presented.

Acknowledgments

The authors wish to thank Marcelo Weinberger and Gadiel Seroussi (main designers of JPEG-LS) from the Visual Computing Department at Hewlett-Packard Labs in Palo Alto, 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 Intra-Channel Decorrelation Implementation

AudioPaK's intra-channel decorrelation algorithm can be easily 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 example, for Intel's MMX SIMD architecture as follows:[5]

[5]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.

```
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
        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] T.C. Bell, J.G. Cleary, and I.H. Witten. *"Text Compression"*. Prentice-Hall, 1990.

[2] A.A.M.L. Bruekers, A.W.J. Oomen, and R.J. van der Vleuten. "Lossless Coding for DVD Audio". *101st AES Convention*, November 1996. 4358.

[3] A.A.M.L. Bruekers, A.W.J. Oomen, and R.J. van der Vleuten. Stated during 101st AES Convention presentation of [2]. Philips, November 1996.

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

[5] P. Craven and M. Gerzon. "Lossless Coding for Audio Discs". *J. Audio Eng. Soc.*, 44(9):706–720, September 1996.

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

[7] Acoustic Renaissance for Audio. "A Proposal for the High-Quality Audio Application of High-Density CD Carriers". Available at `http://www.meridian.co.uk/ara`, January 1996.

[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] M.C. Hans. Personal Communications. Research Lab Book, 1997.

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

[12] 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.

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

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

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

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

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

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

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

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

[21] 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`.

[22] M.J. 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`.