Lossless Compression of Audio Data

ROBERT C. MAHER

OVERVIEW

Lossless data compression of digital audio signals is useful when it is necessary to minimize the storage space or transmission bandwidth of audio data while still maintaining archival quality. Available techniques for lossless audio compression, or lossless audio *packing*, generally employ an adaptive waveform predictor with a variable-rate entropy coding of the residual, such as Huffman or Golomb–Rice coding. The amount of data compression can vary considerably from one audio waveform to another, but ratios of less than 3 are typical. Several freeware, shareware, and proprietary commercial lossless audio packing programs are available.

12.1 INTRODUCTION

The Internet is increasingly being used as a means to deliver audio content to end-users for entertainment, education, and commerce. It is clearly advantageous to minimize the time required to download an audio data file and the storage capacity required to hold it. Moreover, the expectations of end-users with regard to signal quality, number of audio channels, meta-data such as song lyrics, and similar additional features provide incentives to compress the audio data.

12.1.1 Background

In the past decade there have been significant breakthroughs in audio data compression using lossy *perceptual* coding [1]. These techniques lower the bit rate required to represent the signal by establishing perceptual error criteria, meaning that a model of human hearing perception is

used to guide the elimination of excess bits that can be either reconstructed (redundancy in the signal) or ignored (inaudible components in the signal). Such systems have been demonstrated with "perceptually lossless" performance, i.e., trained listeners cannot distinguish the reconstructed signal from the original with better than chance probability, but the reconstructed waveform may be significantly different from the original signal. Perceptually lossless or near-lossless coding is appropriate for many practical applications and is widely deployed in commercial products and international standards. For example, as of this writing millions of people are regularly sharing audio data files compressed with the MPEG 1 Layer 3 (MP3) standard, and most DVD video discs carry their soundtracks encoded with Dolby Digital multichannel lossy audio compression.

There are a variety of applications, however, in which lossy audio compression is inappropriate or unacceptable. For example, audio recordings that are to be stored for archival purposes must be recoverable bit-for-bit without any degradation, and so lossless compression is required. Similarly, consumers of audio content may choose to purchase and download a losslessly compressed file that provides quality identical to the same music purchased on a conventional audio CD. Furthermore, lossy compression techniques are generally not amenable to situations in which the signal must pass through a series of several encode/decode operations, known as *tandem* encode/decode cycles. This can occur in professional studio applications where multiple audio tracks are additively mixed together or passed through audio effects devices such as EQ filters or reverberators and then reencoded. Tandeming can also occur in broadcasting or content distribution when the signal must be changed from one data format to another or sent through several stages of intermediate storage. Audible degradations due to lossy compression will accumulate with each encode/decode sequence, and this may be undesirable.

12.1.2 Expectations

Lossy audio compression such as MP3 is appropriate for situations in which it is necessary to specify the best perceived audio quality at a specific, guaranteed bit rate. Lossless compression, on the other hand, is required to obtain the lowest possible bit rate while maintaining perfect signal reconstruction. It is important to be aware that the bit rate required to represent an audio signal losslessly will vary significantly from one waveform to another depending on the amount of redundancy present in the signal. For example, a trivial file containing all "zero" samples (perfect silence) would compress down to an integer representing the number of samples in the file, while an audio signal consisting of white noise would thwart any attempt at redundancy removal. Thus, we must be prepared to accept results in which the bit rate of the losslessly compressed data is not significantly reduced compared to the original rate.

Because most audio signals of interest have temporal and spectral properties that vary with time, it is expected that the lossless compression technique will need to adapt to the short-term signal characteristics. The time varying signal behavior will imply that the instantaneous bit rate required to represent the compressed signal will vary with time, too. In some applications, such as storing an audio data file on a hard disk, the major concern is the average bit rate since the size of the resulting file is to be minimized. In other applications, most notably in systems requiring real-time transmission of the audio data or data retrieval from a fixed-bandwidth storage device such as DVD, there may also be concern about the peak bit rate of the compressed data.

A plethora of bit resolutions, sample rates, and multiple channel formats are in use or have been proposed for recording and distribution of audio data. This means that any technique proposed for lossless compression should be designed to handle pulse code modulation (PCM) audio samples with 8- to 32-bit resolution, sample rates up to 192 kHz, and perhaps six or more audio channels. In fact, many of the available commercial lossless compression methods include special features to optimize their performance to the particular format details of the audio data [5].

12.1.3 Terminology

A variety of terms and informal colloquial phrases are used in the description of lossless audio data compression. In this chapter several of these terms may be used interchangeably and it is helpful to keep this in mind so that no special distinction is to be inferred unless specific mention is given in the text. A summary of these terms is given next.

Lossless compression and lossless packing both refer to methods for reducing the number of data bits required to represent a stream of audio samples. Some authors choose to use the term *packing* instead of *compression* in order to avoid potential confusion between the lossy and lossless data compression, and between data compression and dynamic range compression, as in the audio dynamics processing device known as a gain compressor/limiter. In this chapter the full expression "lossless data compression" is used.

The performance of a lossless data compressor can be interpreted in several ways. One is the *compression ratio*, which is the size of the input data file divided by the size of the output data file. Sometimes the performance is stated as a *percentage*, either the percentage reduction in size or the percentage of the original file size that remains after compression. Another interpretation is to describe data compression as a way to minimize the *bit rate* of the signal, where the bit rate is the number of bits required to represent the data divided by the total playing time. And in some contexts it is useful to describe data compression in terms of the average *number of bits per sample* or the average *reduction in bits per sample*. Clearly, the compression ratio is most helpful when trying to determine how much file system space would be saved by compressing the audio data, while the interpretation in terms of bit rate is more meaningful when considering transmission of the data through a communications channel.

Finally, the act of compressing and decompressing the audio data is sometimes referred to as *encoding* and *decoding*. In this context the encoded data is the losslessly compressed data stream, while the decoded data is the recovered original audio waveform. In this chapter we use *compression* and *encoding* interchangeably.

12.2 PRINCIPLES OF LOSSLESS DATA COMPRESSION

The essence of lossless data compression is to obtain efficient redundancy removal from a bitstream [9]. Common lossless compressors such as `WinZIP` and `StuffIT` are used on arbitrary computer data files and usually provide compressed files roughly half the size of the original. However, the compression model (e.g., *LZ77*) commonly used is poorly matched to the statistical characteristics of binary audio data files, and the compressed audio files are typically still about 90% of the original file size. On the other hand, good audio-specific lossless compressors can achieve output files that are 30–50% of the original file size [4]. Nonetheless, it is important to remember that there are no guarantees about the amount of compression obtainable from an arbitrary audio data file: It is even possible that the compressed file ends up larger than the original due to the overhead of data packing information! Clearly the lossless algorithm should detect this situation and not "compress" that file.

12.2.1 Basic Redundancy Removal

How can an audio-specific lossless compression algorithm work better than a general-purpose Ziv–Lempel algorithm? To examine this question, consider the audio signal excerpt displayed in Fig. 12.1. This excerpt is approximately 5 s of audio taken from a 24-bit PCM data file with 48-kHz sample rate (1.152 million bits/s/channel).

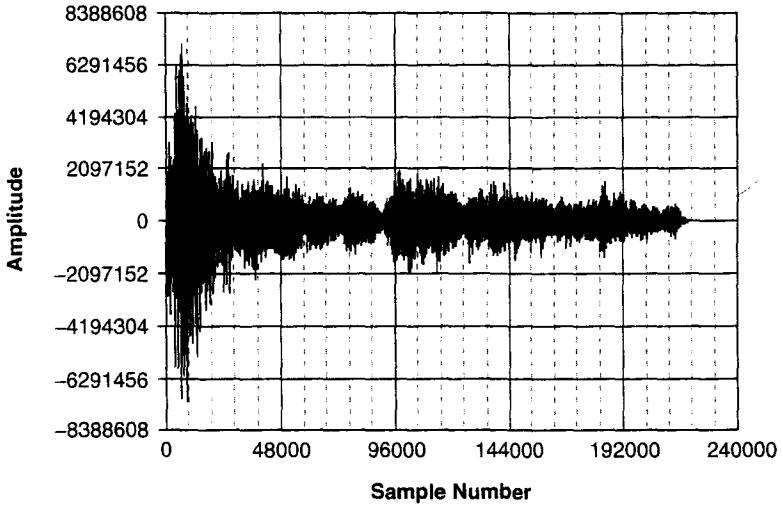

FIGURE 12.1
Excerpt (5 s) from an audio recording (48-kHz sample rate, 24-bit samples). Note that on the vertical amplitude scale, $8{,}388{,}608 = 2^{23}$; $4{,}194{,}304 = 2^{22}$; and $2{,}097{,}152 = 2^{21}$.

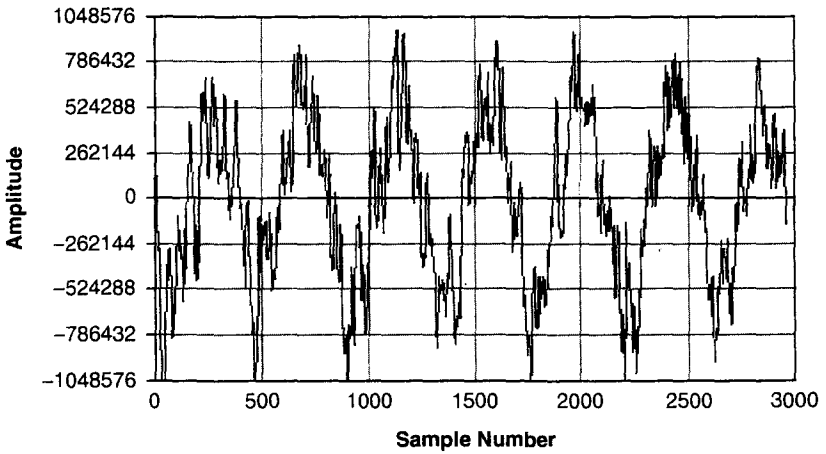

FIGURE 12.2
Enlarged excerpt, approximately 60 ms, from near the middle of the audio recording of Fig. 12.1 (48-kHz sample rate, 24-bit samples). Note that amplitude $1{,}048{,}576 = 2^{20}$.

An enlarged portion of the signal in Fig. 12.1 is shown in Fig. 12.2.

The Ziv–Lempel methods take advantage of patterns that occur repeatedly in the data stream. Note that even if the audio waveform is roughly periodic and consistent as in Fig. 12.2, the audio samples generally do not repeat exactly due to the asynchronous relationship between the waveform period and the sample rate, as well as the presence of noise or other natural fluctuations. This property of audio signals makes them quite different statistically from structured data such as English text or computer program source code, and therefore we should not be surprised that different compression strategies are required.

12.2.2 Amplitude Range and Segmentation

Let us do a heuristic look at the excerpt in Fig. 12.1. One characteristic of this signal is that its amplitude envelope (local peak value over a block of samples) varies with time. In fact, a substantial portion of the excerpt has its amplitude envelope below one-quarter of the full-scale value. In this case we know that at least the 2 most significant bits (MSBs) of the PCM word will not vary from sample to sample, other than for changes in sign (assuming 2's complement numerical representation). If we designed an algorithm to identify the portions of the signal with low amplitude, we could obtain a reduction in bit rate by placing a special symbol in the data stream to indicate that the n most significant bits are the same for each of the following samples, and thus we can eliminate the redundant n MSBs from each subsequent sample [2]. We could also go further by indicating somehow that several samples in a row would share the same sign. Of course, we would need to monitor the signal to detect when the low-amplitude condition was violated, so in addition to the "same MSB" symbol we might also choose to include an integer indicating the number of samples, or block size, for which the condition remained valid. Thus, we could expect to get a reduction in bit rate by *segmentation* of the signal into blocks where the signal's amplitude envelope was substantially less than full scale. It is clear, of course, that the amount of compression will depend upon how frequently low-amplitude blocks occur in the signal.

We can also examine another amplitude-related feature of audio data. As mentioned above, the audio excerpt shown in Figs. 12.1 and 12.2 was obtained from a data file of 24-bit PCM samples. However, if we look carefully at the data for this example, we can discover another heuristic strategy for data compression [2]. Some of the audio data sample values taken from near the beginning of the segment shown in Fig. 12.2 are given here as numerical values:

Sample No.	Decimal Value	Binary Value (24-bit 2's complement)
0	91648	00000001:01100110:00000000
1	95232	00000001:01110100:00000000
2	107008	00000001:10100010:00000000
3	89344	00000001:01011101:00000000
4	42240	00000000:10100101:00000000
5	-19456	11111111:10110100:00000000
6	-92672	11111110:10010110:00000000
7	-150784	11111101:10110011:00000000
8	-174848	11111101:01010101:00000000
9	-192512	11111101:00010000:00000000
10	-215552	11111100:10110110:00000000
11	-236544	11111100:01100100:00000000
12	-261120	11111100:00000100:00000000
13	-256000	11111100:00011000:00000000
14	-221440	11111100:10011111:00000000
15	-222208	11111100:10011100:00000000

As discussed above, note that the MSBs of this excerpt do not carry much information. In fact, this group of 16 samples is low enough in amplitude that the 6 most significant bits can be replaced by a single bit indicating the sign of the number, giving a tidy bit rate reduction of 5 bits per sample. Looking further, notice that even though the data file is stored with 24-bit resolution, the actual data samples do not contain information in the 8 least significant bits (LSBs). This is because the original data stream was actually obtained from a digital audio tape that included only the standard 16-bit per sample audio resolution, and thus the 8 LSBs are filled with zeros.

Again, we can obtain significant data compression merely by storing a special format code that identified the m LSBs (8 in this case) as being redundant for the file and not actually store them. The decoder would detect the special format code and reinsert the m LSBs in the decoded output file. Although such a distinct situation of excess data resolution might seem contrived—and we certainly cannot count on this peculiarity in all data files—it is mentioned here as an example of some of the special circumstances that can be detected to aid in lossless audio compression.

12.2.3 Multiple-Channel Redundancy

Common audio data formats such as the Compact Disc generally store a two-channel (stereo) audio signal. The left and right channel data streams are stored entirely independently. Various new data formats for DVD and other distribution systems will allow four, or six, or perhaps more separate audio channels for use in multichannel surround loudspeaker playback systems. In any case, it is common to find that there is at least some correlation between the two channels in a stereo audio signal or among the various channels in a multichannel recording, and therefore it may be possible to obtain better compression by operating on two or more channels together [3, 6]. This is referred to as *joint stereo coding* or *interchannel coding*. Joint coding is particularly useful for audio signals in which two or more of the channels contain the same signal, such as a dual-mono program, or when one or more of the channels is completely silent for all or most of the recording. Some systems take advantage of interchannel correlation by using a so-called *matrixing* operation to encode L and $(L - R)$, or perhaps $(L + R)$ and $(L - R)$, which is similar to FM broadcast stereo multiplexing. If the L and R channels are identical or nearly identical, the difference signal $(L - R)$ will be small and relatively easy to encode.

12.2.4 Prediction

In many cases of practical interest audio signals exhibit a useful degree of sample-to-sample correlation. That is, we may be able to *predict* the value of the next audio sample based on knowledge of one or more preceding samples [7–9]. Another way of stating this is that if we can develop a probability density function (PDF) for the *next* sample based on our observation of previous samples, and if this PDF is non-uniform and concentrated around a mean value, we will benefit by storing only (a) the minimum information required for the decoder to re-create the same signal estimate and (b) the error signal, or *residual*, giving the sample-by-sample discrepancy between the predicted signal and the actual signal value. If the prediction is very close to the actual value, the number of bits required to encode the residual will be fewer than the original PCM representation. In practice the signal estimate is obtained using some sort of adaptive linear prediction.

A general model of a signal predictor is shown in Fig. 12.3a [3, 7]. The input sequence of audio samples, $x[n]$, serves as the input to the prediction filter, $P(z)$, creating the signal estimate $\tilde{x}[n]$. The prediction is then subtracted from the input, yielding the error residual signal, $e[n]$.

In a practical prediction system it is necessary to consider the numerical precision of the filter $P(z)$, since the coefficient multiplications within the filter can result in more significant bits in $\tilde{x}[n]$ than in the input signal $x[n]$. This is undesirable, of course, since our interest is in minimizing the output bit rate, not adding more bits of precision. The typical solution is to quantize the predicted value to the same bit width as the input signal, usually via simple truncation [2]. This is indicated in Fig. 12.3b. As will be mentioned later in the section on practical design issues, special care must be taken to ensure that the compression and decompression calculations occur identically on whatever hardware platform is used.

An alternative predictor structure that is popular for use in lossless audio compression is shown in Fig. 12.4 [3]. This structure incorporates two finite impulse response (FIR) filters, $A(z)$ and

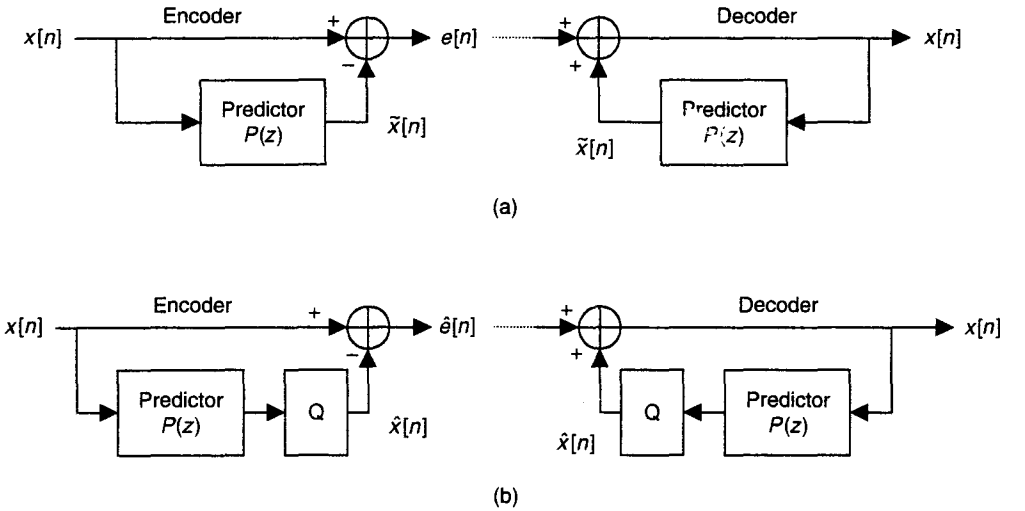

FIGURE 12.3
(a) Basic predictor structure for lossless encoder/decoder. (b) Structure with explicit quantization (truncation) to original input bit width.

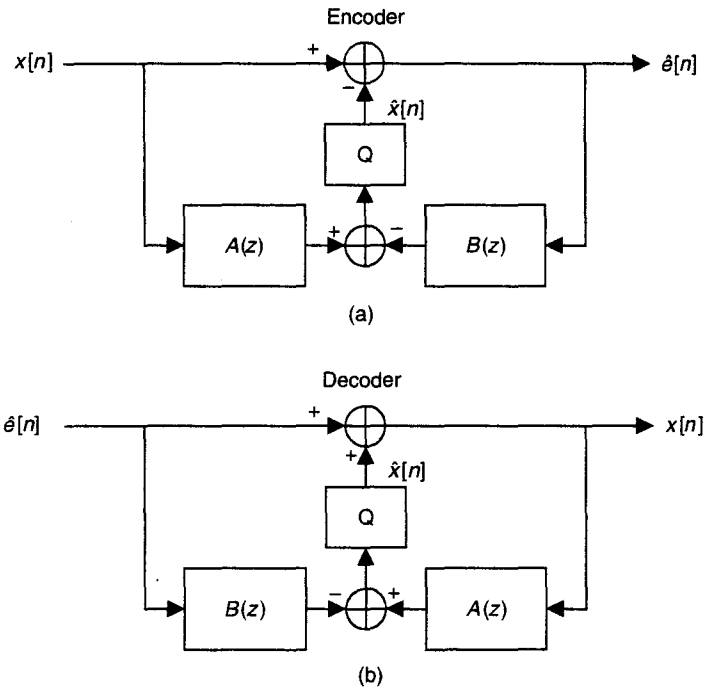

FIGURE 12.4
Alternative predictor structure.

$B(z)$, in a feed-forward and feed-back arrangement similar to an infinite impulse response (IIR) Direct Form I digital filter structure, but with an explicit quantization prior to the summing node. Note also that filter $B(z)$ can be designated as a null filter, leaving the straightforward FIR predictor of Fig. 12.3b. The advantage of selecting $A(z)$ and $B(z)$ to be FIR filters is that the coefficients can be quantized easily to integers with short word lengths, thereby making an integer implementation possible on essentially any hardware. Because this structure is intended for signal prediction and not to approximate a specific IIR filter transfer function, the details of the coefficients in $A(z)$ and $B(z)$ can be defined largely for numerical convenience rather than transfer function precision.

The use of an FIR linear predictor ($B(z) = 0$) is quite common for speech and audio coding, and the filter coefficients for $A(z)$ are determined in order to minimize the mean-square value of the residual $e[n]$ using a standard linear predictive coding (LPC) algorithm [7, 8]. No such convenient coefficient determination algorithm is available for the IIR predictor ($B(z) \neq 0$), which limits the widespread use of the adaptive IIR version. In lossless compression algorithms that utilize IIR predictors it is common to have multiple sets of fixed coefficients from which the encoder chooses the set that provides the best results (minimum mean-square error) on the current block of audio samples [3, 5]. In fact, several popular lossless compression algorithms using FIR prediction filters also include sets of fixed FIR coefficients in order to avoid the computational cost of calculating the optimum LPC results [7].

Once the encoder has determined the prediction filter coefficients to use on the current block, this information must be conveyed to the decoder so that the signal can be recovered losslessly. If an LPC algorithm is used in the encoder, the coefficients themselves must be sent to the decoder. On the other hand, if the encoder chooses the coefficients from among a fixed set of filters, the encoder needs only to send an index value indicating which coefficient set was used.

The choice of predictor type (e.g., FIR vs IIR), predictor order, and adaptation strategy has been studied rather extensively in the literature [7]. Several of the lossless compression packages use a low-order linear predictor (order 3 or 4), while some others use predictors up to order 10. It is interesting to discover that there generally appears to be little additional benefit to the high-order predictors, and in some cases the low-order predictor actually performs better. This may seem counterintuitive, *but keep in mind that there often is no reason to expect that an arbitrary audio signal should fit a predictable pattern*, especially if the signal is a complex combination of sources such as a recording of a musical ensemble.

12.2.5 Entropy Coding

After the basic redundancy removal steps outlined above, most common lossless audio compression systems pass the intermediate predictor parameters and prediction error residual signal through an entropy coder such as Huffman, Golomb–Rice, or run length coding [8, 9].

The appropriate statistical model for the prediction error residual signal is generally well represented by a Laplacian probability density function. The Laplacian PDF $l(x)$ is given by

$$l(x) = \frac{1}{\sqrt{2}\sigma} e^{\frac{-\sqrt{2}}{\sigma}|x|},$$

where σ^2 is the variance of the source distribution and $|x|$ is the absolute value (magnitude) of x (in our case x is the error residual signal, $e[n]$).

In order for the entropy coder to be matched to the Laplacian distributed source the coding parameters must be matched to the assumed or estimated variance of the error residual signal. For example, in an optimal Huffman code we would like to select the number of least significant bits, m, so that the probability of generating a code word $m + 1$ bits long is 0.5, the probability

of generating a code word $m + 2$ bits long is $2^{-(m+1)}$, and so forth. This gives [8]:

$$m = \log_2\left(\log(2)\frac{\sigma}{\sqrt{2}}\right)$$
$$= \log_2(\log(2)E(|x|)),$$

where $E(\ )$ is the expectation operator. Thus, if we determine empirically the expected value of the error residual signal for a segment of data, we can choose a reasonable value of m for the entropy code on that segment. Details on Huffman coding are provided in Chapter 3. Arithmetic coding is discussed in Chapter 4.

12.2.6 Practical System Design Issues

Given the basic rationale for lossless audio coding, we should now consider some of the implementation details that will provide a practical and useful system.

12.2.7 Numerical Implementation and Portability

It is highly desirable in most applications to have the lossless compression and decompression algorithms implemented with identical numerical behavior on any hardware platform. This might appear trivial to achieve if the algorithm exists in the form of computer software, but ensuring that the same quantization, rounding, overflow, underflow, and exception behavior occurs on any digital processor may require special care. This is particularly true if the word size of the audio data exceeds the native word size of the processor, such as 20-bit audio samples being handled on a DSP chip with 16-bit data registers. Similarly, care must be taken if the compression/decompression algorithm is implemented using floating point arithmetic since floating point representations and exception handling may vary from one processor architecture to another.

Most commercially available lossless compression software packages appear to address the arithmetic issue by computing intermediate results (e.g., prediction filter output) with high precision, then quantize via truncation to the original integer word size prior to the entropy coding step [7]. As long as the target processors handle arithmetic truncation properly, the results will be compatible from one processor to another.

12.2.8 Segmentation and Resynchronization

In a practical lossless compression system we will need to break up the input audio stream into short segments over which the signal characteristics are likely to be roughly constant. This implies that a short block size is desired. However, the prediction filter coefficients will probably be determined and sent to the decoder for each block, indicating that we would want the block size to be relatively long to minimize the overhead of transmitting the coefficients. Thus, we will need to handle a trade-off between optimally matching the predictor to the varying data (short block length is better) and minimizing the coefficient transmission overhead (long block is better). In commercially available systems the block length is generally between 10 and 30 ms (around 1000 samples at a 44.1-kHz sample rate), and this appears to be a reasonable compromise for general audio compression purposes [7].

In many applications it may be required—or at least desirable—to start decoding the compressed audio data at some point other than the very beginning of the file without the need to decode the entire file up to that point. In other words, we may want to jump ahead to a particular edit point to extract a sound clip, or we may want to synchronize or time-align several different

recordings for mixdown purposes. We may also need to allow rapid resynchronization of the data stream in the event of an unrecoverable error such as damaged or missing data packets in a network transmission. Since the bit rate varies from time to time in the file and the entropy coding will also likely be of variable length, it is difficult to calculate exactly where in the file to jump in order to decode the proper time segment. Thus, in a practical lossless compression system it is necessary for the encoder to provide framing information that the decoder can use to determine where in the uncompressed time reference a particular block of compressed data is to be used. The compressed data stream must be designed in such a way to meet the practical framing requirements while still obtaining a good compression ratio [5].

12.2.9 Variable Bit Rate: Peak versus Average Rate

As mentioned previously, the characteristics of typical audio signals vary from time to time and therefore we must expect the required bit rate for lossless compression to vary as well. Since the bit rate will vary, a practical lossless system will be described by both an *average* and a *peak* bit rate. The average rate is determined by dividing the total number of bits in the losslessly compressed stream by the total audio playing time of the uncompressed data. The peak bit rate is the maximum number of bits required for any short-term block of compressed audio data, divided by the playing time of the uncompressed block of audio. It might appear that the peak bit rate would never exceed the original bit rate of the digital audio signal, but because it is possible that the encoder must produce prediction filter coefficients, framing information, and other miscellaneous bits along with the data stream itself, the total overhead may actually result in a higher peak bit rate.

The average compressed bit rate spec is most relevant for storage of audio files on computer hard disks, data tapes, and so forth, since this specifies the reduction in the storage space allocated to the file. We obviously would like the average bit rate to be lower than the rate of the original signal, otherwise we will not obtain any overall data compression. Most published comparisons of lossless audio compression systems compare the average compressed bit rate, because this is often the most relevant issue for an audio file that is simply going to be packed into a smaller space for archival purposes [4, 7].

Nonetheless, the peak bit rate is a significant issue for applications in which the short-term throughput of the system is limited [2]. This situation can occur if the compressed bit stream is conveyed in real time over a communication channel with a hard capacity limitation, or if the available buffer memory in the transmitter and/or receiver does not allow the accumulation of enough excess bits to accommodate the peak bit rate of the compressed signal. The peak bit rate is also important when the compressed data are sent over a channel shared with other accompanying information such as compressed video, and the two compressed streams need to remain time-synchronized at all times. Thus, most commercial lossless audio data compression systems employ strategies to reduce the peak-to-average ratio of the compressed bit stream [5].

12.2.10 Speed and Complexity

In addition to the desire for the greatest amount of compression possible, we must consider the complexity of the compression/decompression algorithms and the feasibility of implementing these algorithms on the available hardware platforms. If we are interested mainly in archiving digital audio files that are stored on hard disks or digital tape, it is natural simply to use a compression/decompression computer program. Ideally we would like the computer program to be sufficiently fast that the compression and decompression procedures would be limited by the access rate of the hard disk drive, not the computation itself.

In some applications the audio content producer may wish to distribute the compressed recordings to mass-market consumers, perhaps via the Internet. In this situation the end-user needs only the decompressor: The compression procedure for the recording is performed once "back at the factory" by the content producer. Since only the decompressor is distributed to the consumer, the producer may have the opportunity to choose an *asymmetrical* compression/decompression algorithm in which the complexity and computational cost of the compression process can be arbitrarily high, while the decompression process is made as fast and simple as possible. As long as the decompression algorithm is flexible enough to take advantage of highly optimized bit streams, the content producer has the luxury of iteratively adjusting the prediction coefficients, block sizes, etc., to get the lowest compressed bit rate [5].

12.3 EXAMPLES OF LOSSLESS AUDIO DATA COMPRESSION SOFTWARE SYSTEMS

At the time of this writing, at least a dozen lossless audio compression software packages are available as freeware, shareware, or commercial products [4]. Three examples are summarized next. These packages were chosen to be representative of the current state-of-the-art, but no endorsement should be inferred by their inclusion in this section.

12.3.1 Shorten

The `Shorten` audio compression software package is based on the work of Tony Robinson at Cambridge University, Cambridge, United Kingdom [8]. The `Shorten` package provides a variety of user-selectable choices for block length and predictor characteristics. `Shorten` offers a standard LPC algorithm for the predictor or a faster (but less optimal) mode in which the encoder chooses from among four fixed polynomial predictors. The residual signal after the predictor is entropy encoded using the Rice coding technique. Thus, the `Shorten` algorithm follows the general lossless compression procedure described above: Samples are grouped into short blocks, a linear predictor is chosen, and the residual signal is entropy coded.

The `Shorten` software can also be used in a lossy mode by specifying the allowable signal-to-error power level. In this mode the quantization step size for the residual signal is increased. Note that the lossy mode of `Shorten` is not based on an explicit perceptual model, but simply a waveform error model.

`Shorten` is distributed as an executable for Microsoft Windows PCs. A free evaluation version and a full-featured commercial version (about $30) are available for downloading from www.softsound.com. The full version provides a real-time decoder function, the ability to create self-extracting compressed files, and several advanced mode options.

The `Shorten` application runs as a regular Windows application with a simple graphical user interface. The software allows the user to browse for ordinary Windows audio files (`*.wav` and raw binary) and then compress/decompress them. The compressed files are assigned the file extension `*.shn`. Audio files compressed losslessly by `Shorten` are typically between 40 and 60% of the original file size. The software runs more than 10 times faster than real time on typical PC hardware.

Several other software packages based on the original work of Tony Robinson are also available. These include programs and libraries for Unix/Linux and Macintosh, as well as plug-ins for popular Windows audio players such as `WinAmp`.

12.3.2 Meridian Lossless Packing (MLP)

The DVD Forum, an industry group representing manufacturers of DVD recording and playback equipment, has developed a special standard for distribution of high-quality audio material using DVD-style discs. This standard, known as DVD-Audio, provides for up to six separate audio channels, sample rates up to 192 kHz, and PCM sample resolution up to 24 bits. Typical DVD-Audio discs (single sided) can store nearly 90 min of high-resolution multichannel audio.

Among the special features of the DVD-Audio specification is the option for the content producer to use lossless compression on the audio data in order to extend the playing time of the disc. The specified lossless compression algorithm is called *Meridian Lossless Packing*, invented by Meridian Audio of Cambridge, United Kingdom [5]. MLP was developed specifically for multichannel, multiresolution audio data, and it includes support for a wide range of professional formats, downmix options, and data rate/playing time trade-offs. Consumer electronics devices for playing DVD-Audio discs incorporate the MLP decoder for real-time decompression of the audio data retrieved from the disc.

The MLP system uses three techniques to reduce redundancy. Lossless interchannel decorrelation and matrixing are used to eliminate correlation among the input audio channels. Next, an IIR waveform predictor is selected from a predetermined set of filters in order to minimize intersample correlation for each block. Finally, Huffman coding is used to minimize the bit rate of the residual signals.

Because MLP is designed for use in off-line DVD-Audio production, the compression system operator can use trial and error to find the best combination of sample resolution, bit rate, and channel format to obtain the best signal quality for a given duration of the recording. The complexity of the decoder remains the same no matter how much time the production operator spends selecting the optimum set of compression parameters.

MLP is intended for use in professional audio production and distribution so its designers have incorporated many features for flexibility and reliability. For example, MLP includes full "restart" resynchronization information approximately every 5 ms in the compressed audio stream. This allows the stream to be cued to a particular point in time without decoding the entire prior compressed stream and also allows rapid recovery from serious transmission or storage errors. Another useful professional feature is the ability to include copyright, ownership, and error detection/correction information. MLP also provides a way for the content producer to specify which audio channel is intended for which loudspeaker (e.g., front center, left rear) to achieve the intended audio playback environment.

12.3.3 Sonic Foundry Perfect Clarity Audio (PCA)

The *Perfect Clarity Audio* lossless audio codec uses a combination of an adaptive predictor, stereo interchannel prediction, and Huffman coding of the residual signal [10]. PCA is distributed as part of the proprietary software products from Sonic Foundry, Inc., including `Sound Forge`, `Vegas`, and `ACID`. PCA is designed both for long-term archival backup of audio material and for economical temporary storage of audio tracks that are in the process of being edited or mixed. Because the data are stored losslessly, the user can edit and reedit the material as much as desired without accumulating coding noise or distortion.

The PCA package is intended for mono or stereo audio files with 16-bit or 24-bit PCM sample resolution. Like `Shorten` and MLP, a compressed file is typically about half the size of the original audio file. The encoding process is sufficiently fast to compress at 10 times real time on typical circa 2001 PC hardware, e.g., 1 min of audio is compressed in about 5 s. The decoding

process requires a lower computation rate than encoding and can be accomplished typically with only a few percent of the available CPU cycles during playback.

PCA incorporates an error detection mechanism to flag any storage or transmission errors that might occur during file handling. PCA also provides the ability to include summary information and accompanying text along with the compressed audio data.

12.4 CONCLUSION

Lossless audio compression is used to obtain the lowest possible bit rate while still retaining perfect signal reconstruction at the decoder. The combination of an adaptive signal predictor with a subsequent lossless entropy coder is the preferred method for lossless audio compression. This is the basic framework utilized in the most popular audio compression software packages. All of the popular lossless audio compression algorithms obtain similar bit rate reductions on real-world audio signals, indicating that the practical limit for lossless audio compression performance has been reached. Typical compressed file sizes are between 40 and 80% of the original file size, which compares rather poorly to the performance of lossy perceptual audio coders which can achieve "perceptually lossless" performance at 10% or less of the original file size. Nonetheless, in applications requiring perfectly lossless waveform coding, the advantages of an audio-specific compressor compared to a general-purpose data compressor are often significant: Audio files compressed with `WinZip` are typically 80–100% of the original file size.

12.5 REFERENCES

1. Brandenburg, K., 1998. Perceptual coding of high quality digital audio. In *Applications of Digital Signal Processing to Audio and Acoustics* (M. Kahrs and K. Brandenburg, Eds.), Chap. 2, pp. 39–83, Kluwer Academic, Dordrecht/Norwell, MA.
2. Craven, P. G., and Gerzon, M. A., 1996. Lossless coding for audio discs. *Journal of the Audio Engineering Society*, Vol. 44, No. 9, pp. 706–720, September 1996.
3. Craven, P. G., Law, M. J., and Stuart, J. R., 1997. Lossless compression using IIR prediction filters. In *Proceedings of the 102nd Audio Engineering Society Convention, Munich, Germany*, March 1997, Preprint 4415.
4. Dipert, B., 2001. Digital audio gets an audition Part One: Lossless compression. *EDN Magazine*, Jan. 4, 2001, pp. 48–61. Available at http://www.e-insite.net/ednmag/contents/images/60895.pdf.
5. Gerzon, M. A., Craven, P. G., Stuart, J. R., Law, M. J., and Wilson, R. J., 1999. The MLP lossless compression system. In *Proceedings of the AES 17th International Conference, Florence, Italy*, September 1999, pp. 61–75.
6. Hans, M., 1998. *Optimization of Digital Audio for Internet Transmission*, Ph.D. Thesis, Georgia Institute of Technology, May 1998, Available at http://users.ece.gatech.edu/~hans/thesis.zip.
7. Hans, M., and Schafer, R. W., 1999. Lossless Compression of Digital Audio, Hewlett-Packard Technical Report HPL-1999-144, November 1999, Available at http://www.hpl.hp.com/techreports/1999/HPL-1999-144.html.
8. Robinson, T., 1994. SHORTEN: Simple Lossless and Near-Lossless Waveform Compression, Technical Report CUED/F-INFENG/TR.156, Cambridge University Engineering Department, Cambridge, UK, December 1994, Available at http://svr-www.eng.cam.ac.uk/reports/svr-ftp/robinson_tr156.ps.Z.
9. Sayood, K., 2000. *Introduction to Data Compression*, 2nd ed., Morgan Kaufmann.
10. Sonic Foundry, Inc., 2001. Private communication, Madison, WI, May 2001.