Audio Engineering Society

Convention Paper

Presented at the 119th Convention
2005 October 7–10 New York, NY, USA

The MPEG-4 Audio Lossless Coding (ALS) Standard - Technology and Applications

Tilman Liebchen[1], Takehiro Moriya[2], Noboru Harada[2], Yutaka Kamamoto[2], and Yuriy A. Reznik[3]

[1] *Technical University of Berlin, Berlin, Germany*

[2] *NTT Communication Science Laboratories, Atsugi, Japan*

[3] *RealNetworks, Inc., Seattle, WA, USA*

Correspondence should be addressed to Tilman Liebchen (liebchen@nue.tu-berlin.de)

ABSTRACT
MPEG-4 Audio Lossless Coding (ALS) is a new extension of the MPEG-4 audio coding family. The ALS core codec is based on forward-adaptive linear prediction, which offers remarkable compression together with low complexity. Additional features include long-term prediction, multichannel coding, and compression of floating-point audio material. In this paper authors who have actively contributed to the standard describe the basic elements of the ALS codec with a focus on prediction, entropy coding, and related tools. We also present latest developments in the standardization process and point out the most important applications of this new lossless audio format.

1. INTRODUCTION

Lossless audio coding permits the compression of digital audio data without any loss in quality due to a perfect reconstruction of the original signal. The MPEG audio subgroup has recently completed the standardization of lossless coding techniques for high-definition audio signals. As an addition to the MPEG-4 audio standard [1], Audio Lossless Coding (ALS) provides methods for lossless coding of audio signals with arbitrary sampling rates, resolutions of up to 32 bit, and up to 2^{16} channels.

In July 2003, the lossless codec from Technical University of Berlin was chosen as the first working draft. Since then, further improvements and extensions have been integrated. The final technical specification has been issued in July 2005 [2], thus MPEG-4 ALS is expected to become an international standard by the end of 2005.

The paper constitutes an update of previous publications on MPEG-4 ALS [3][4][5][6]. The following chapters will provide a more detailed description of the codec. After an overview of the codec structure

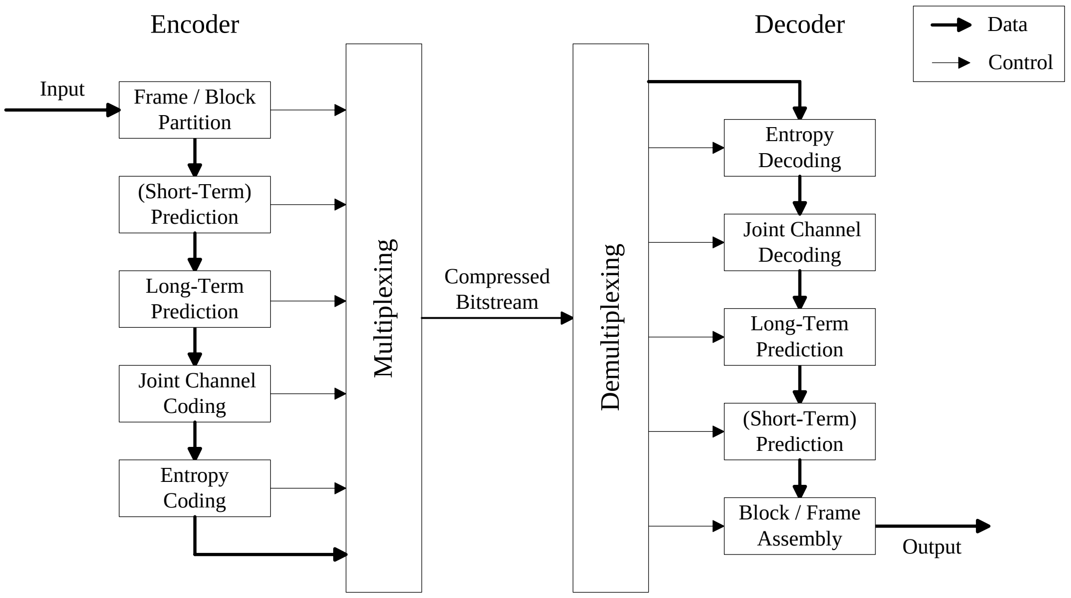

Fig. 1: Block diagram of MPEG-4 ALS encoder and decoder.

in section 2, section 3 puts the main focus on linear prediction together with block length switching, random access, and long-term prediction. Section 4 illustrates methods for joint channel coding, and section 5 describes the entropy coding scheme for the prediction residual. In section 6, the codec extensions for compression of floating-point audio data are presented. Coding results for a variety of audio material (including high-resolution and floating-point) are given in section 7, while section 8 provides a discussion of application scenarios for lossless audio coding in general and MPEG-4 ALS in particular.

2. STRUCTURE OF THE CODEC

The basic structure of the ALS encoder and decoder is shown in Figure 1. The input audio data is partitioned into frames. Within a frame, each channel can be further subdivided into blocks of audio samples for further processing (block length switching). For each block, a prediction residual is calculated using forward adaptive prediction. The basic (short-term) prediction can be combined with long-term prediction. Inter-channel redundancy can be removed by joint channel coding, either using difference coding of channel pairs or multi-channel coding. The remaining prediction residual is finally entropy coded. The encoder generates bitstream information allowing for random access at intervals of several frames. The encoder can also provide a CRC checksum, which the decoder may use to verify the decoded data.

3. LINEAR PREDICTION

Linear prediction is used in many applications for speech and audio signal processing. In the following, only FIR predictors are considered.

3.1. Prediction with FIR Filters

The current sample of a time-discrete signal $x(n)$ can be approximately predicted from previous samples $x(n-k)$. The prediction is given by

$$\hat{x}(n) = \sum_{k=1}^{K} h_k \cdot x(n-k), \qquad (1)$$

where K is the order of the predictor. If the predicted samples are close to the original samples, the residual

$$e(n) = x(n) - \hat{x}(n) \qquad (2)$$

has a smaller variance than $x(n)$ itself, hence $e(n)$ can be encoded more efficiently.

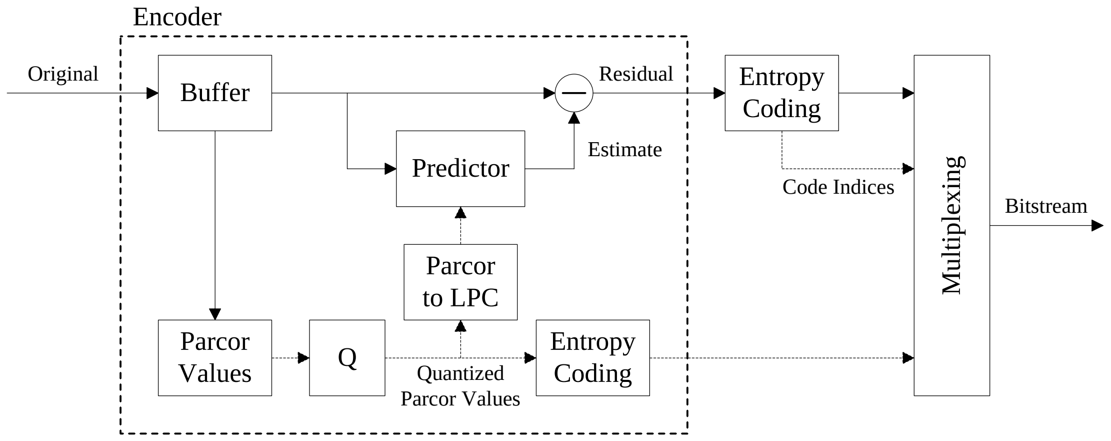

Fig. 2: Encoder of the forward-adaptive prediction scheme.

The procedure of estimating the predictor coefficients from a segment of input samples, prior to filtering that segment, is referred to as forward adaptation. In that case, the coefficients have to be transmitted. If the coefficients are estimated from previously processed segments or samples, e.g. from the residual, we speak of backward adaptation. This procedure has the advantage that no transmission of the coefficients is needed, since the data required to estimate the coefficients is available to the decoder as well [7].

Forward-adaptive prediction with orders around 10 is widely used in speech coding, and can be employed for lossless audio coding as well [8][9]. The maximum order of most forward-adaptive lossless prediction schemes is still rather small, e.g. $K = 32$ [10]. An exception is the special 1-bit lossless codec for the Super Audio CD, which uses predictor orders of up to 128 [11].

On the other hand, backward-adaptive FIR filters with some hundred coefficients are commonly used in many areas, e.g. channel equalization and echo cancellation [12]. Most systems are based on the LMS algorithm or a variation thereof, which has also been proposed for lossless audio coding [13][14][15]. Such LMS-based coding schemes with high orders are applicable since the predictor coefficients do not have to be transmitted as side information, thus their number does not contribute to the data rate. However, backward-adaptive codecs have the drawback that the adaptation has to be carried out both in the encoder and the decoder, making the decoder significantly more complex than in the forward-adaptive case. MPEG-4 ALS specifies an optional backward-adaptive predictor as well [2], but in the following, only the forward-adaptive predictor and related tools are discussed.

3.2. Forward-Adaptive Prediction

This chapter describes the forward-adaptive prediction scheme. A block diagram of the corresponding encoder is shown in Figure 2.

The encoder consists of several building blocks. A buffer stores one block of input samples, and an appropriate set of parcor coefficients is calculated for each block. The number of coefficients, i.e. the order of the predictor, can be adapted as well. The quantized parcor values are entropy coded for transmission, and converted to LPC coefficients for the prediction filter, which calculates the prediction residual. The final entropy coding of the residual is described in section 5.

In forward-adaptive linear prediction, the optimal predictor coefficients h_k (in terms of a minimized variance of the residual) are usually estimated for each block by the autocorrelation method or the covariance method [16]. The autocorrelation method, using the Levinson-Durbin algorithm, has the additional advantage of providing a simple means to iteratively adapt the *order* of the predictor [8]. Furthermore, the algorithm inherently calculates the corresponding parcor coefficients as well.

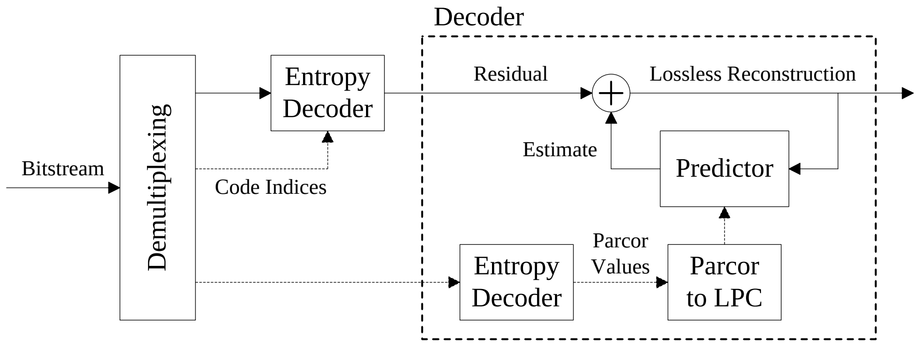

Fig. 3: Decoder of the forward-adaptive prediction scheme.

The decoder (Figure 3) is significantly less complex than the encoder, since no adaptation has to be carried out. The transmitted parcor values are decoded, converted to LPC coefficients, and are used by the inverse prediction filter to calculate the lossless reconstruction signal. The computational effort of the decoder mainly depends on the predictor orders chosen by the encoder. Since the average order is typically well below the maximum order, prediction with greater maximum orders does not necessarily lead to a significant increase of decoder complexity.

3.3. Adaptation of the Prediction Order

Another crucial point in forward-adaptive prediction is to determine a suitable prediction order. It is of course straightforward to use the same prediction order for all blocks of samples, thus adapting only the *values* of the coefficients. However, an adaptive choice of the *number* of predictor taps is extremely beneficial in order to account for varying signal statistics and different block lengths (see section 3.5), as well as to minimize the amount of side information spent for transmitting the sets of coefficients.

Assumed that the values of the coefficients are adaptively chosen, increasing the order of the predictor successively reduces the variance of the prediction error, and consequently leads to a smaller bit rate R_e for the coded residual. On the other hand, the bit rate R_c for the predictor coefficients will rise with the number of coefficients to be transmitted. Thus, the task is to find the optimum order which minimizes the total bit rate. This can be expressed by minimizing

$$R_{total}(K) = R_e(K) + R_c(K) \tag{3}$$

with respect to the prediction order K. As the prediction gain rises monotonically with higher orders, R_e decreases with K. On the other hand R_c rises monotonically with K, since an increasing number of coefficients have to be transmitted.

The search for the optimum order can be carried out efficiently by the Levinson-Durbin algorithm, which determines recursively all predictors with increasing order. For each order, a complete set of predictor coefficients is calculated. Moreover, the variance σ_e^2 of the corresponding residual can be derived, resulting in an estimate of the expected bit rate for the residual. Together with the bit rate for the coefficients, the total bit rate can be determined in each iteration, i.e. for each prediction order. The optimum order is found at the point where the total bit rate no longer decreases.

While it is obvious from Eq. (3) that the coefficient bit rate has a direct effect on the total bit rate, a slower increase of R_c also allows to shift the minimum of R_{total} to higher orders (where R_e is smaller as well), which would lead to even better compression. As MPEG-4 ALS supports prediction orders up to $K = 1023$, efficient though accurate quantization of the predictor coefficients plays an important role in achieving maximum compression.

3.4. Quantization of Predictor Coefficients

Direct quantization of the predictor coefficients h_k is not very efficient for transmission, since even small quantization errors may result in large deviations from the desired spectral characteristics of the optimum prediction filter [7]. For this reason, the quantization of predictor coefficients in MPEG-4 ALS is based on the parcor (reflection) coefficients r_k, which can be calculated by means of the Levinson-Durbin algorithm. In that case, the resulting values are restricted to the interval $[-1, 1]$. Although parcor coefficients are less sensitive to quantization, they are still too sensitive when their magnitude is close to unity. The first two parcor coefficients r_1 and r_2 are typically very close to -1 and $+1$, respectively, while the remaining coefficients $r_k, k > 2$, usually have smaller magnitudes. The distributions of the first coefficients are very different, but high-order coefficients tend to converge to a zero-mean gaussian-like distribution (Figure 4).

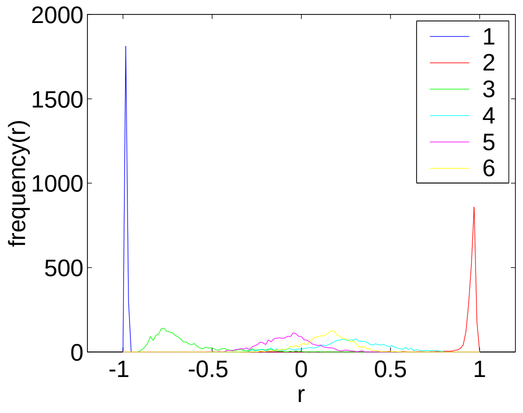

Fig. 4: Measured distributions of parcor coefficients $r_1 \ldots r_6$, for 48 kHz, 16-bit audio material.

Therefore, only the first two coefficients are companded based on the following function:

$$C(r) = -1 + \sqrt{2}\sqrt{r+1} \qquad (4)$$

This compander results in a significantly finer resolution at $r_1 \to -1$, whereas $-C(-r_2)$ can be used to provide a finer resolution at $r_2 \to +1$ (see Figure 5).

However, in order to simplify computation, $+C(-r_2)$ is actually used for the second coefficient, leading to an opposite sign of the companded value. The two companded coefficients are then quantized using a simple 7-bit uniform quantizer. This results in the following values:

$$a_1 = \left\lfloor 64\left(-1 + \sqrt{2}\sqrt{r_1+1}\right)\right\rfloor \qquad (5)$$

$$a_2 = \left\lfloor 64\left(-1 + \sqrt{2}\sqrt{-r_2+1}\right)\right\rfloor \qquad (6)$$

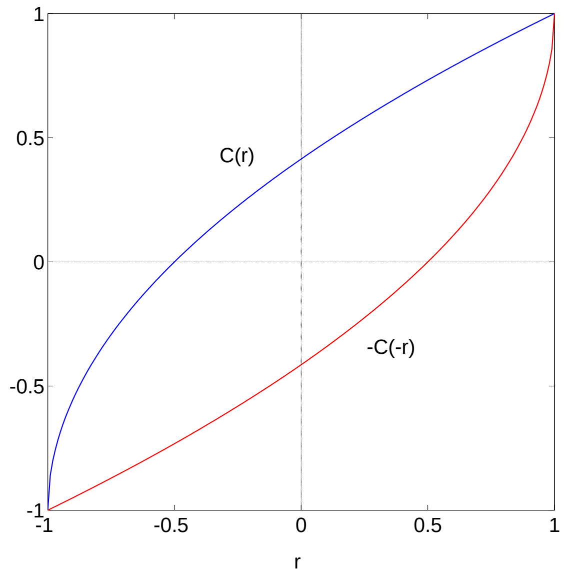

Fig. 5: Compander functions $C(r)$ and $-C(-r)$.

The remaining coefficients $r_k, k > 2$ are not companded but simply quantized using a 7-bit uniform quantizer again:

$$a_k = \lfloor 64 r_k \rfloor \qquad (7)$$

In all cases the resulting quantized values a_k are restricted to the range $[-64, +63]$. These quantized coefficients are re-centered around their most probable values, and then encoded using Golomb-Rice codes. As a result, the average bit rate of the encoded parcor coefficients can be reduced to approximately 4 bits/coefficient, without noticeable degradation of the spectral characteristics. Thus, it is possible to employ very high orders up to $K = 1023$, preferably in conjunction with large block lengths (see section 3.5).

However, the direct form predictor filter uses predictor coefficients h_k according to Eq. (1). In order to employ identical coefficients in the encoder and the decoder, these h_k values have to be derived from the quantized a_k values in both cases (see Figures 2 and 3). While it is up to the encoder how to determine a set of suitable parcor coefficients, MPEG-4 ALS specifies an integer-arithmetic function for conversion between quantized values a_k and direct predictor coefficients h_k which ensures their identical reconstruction in both encoder and decoder.

3.5. Block Length Switching

The basic version of the encoder uses one sample block per channel in each frame. The frame length can initially be adjusted to the sampling rate of the input signal, e.g. 2048 for 48 kHz or 4096 for 96 kHz (approximately 43 ms in each case).

While the frame length is constant for one input file, optional *block length switching* enables a subdivision of a frame into shorter blocks in order to adapt to transient segments of the audio signal. Previous versions of the MPEG-4 ALS codec [5] already included a simple block switching mechanism which allowed to encode each frame of N samples using either one full length block ($N_B = N$) or four blocks of length $N_B = N/4$. Meanwhile, an improved and more flexible block switching scheme was designed, and each frame of length N can be hierarchically subdivided into up to 32 blocks. Arbitrary combinations of blocks with $N_B = N$, $N/2$, $N/4$, $N/8$, $N/16$, and $N/32$ are possible within a frame, as long as each block results from a subdivision of a superordinate block of double length. Therefore, a partition into $N/4 + N/4 + N/2$ is possible, whereas a partition into $N/4 + N/2 + N/4$ is not (Figure 6).

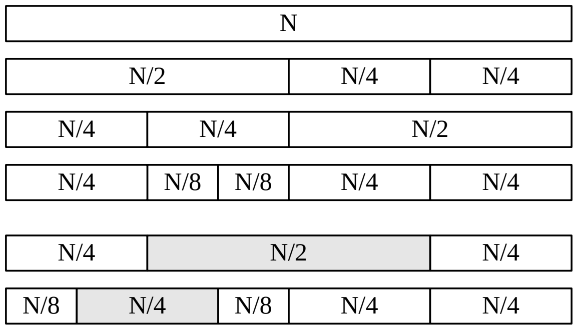

Fig. 6: Block switching examples. The last two partitions are not allowed due to the positions of the shaded blocks.

Block length switching allows the use of both very short and very long blocks within the same audio signal. For stationary segments, long blocks with high predictor orders may be chosen, while for transient segments short blocks with lower orders are more convenient. As the maximum block length is bounded by the frame length, the latter has to be chosen such that a reasonable range of block lengths is covered. For instance, a frame length of $N = 8192$ enables blocks with lengths $N_B = 8192$, 4096, 2048, 1024, 512, and 256.

The choice of a suitable block partition is entirely left to the encoder, and thus not further specified by MPEG. Possible methods may range from evaluating of the signal statistics to exhaustive search algorithms. The actual partition has to be transmitted as side information, which takes at most 32 bits per frame. Since the decoder still has to process the same number of samples per frame, block switching enables significantly improved compression without increasing the decoder complexity.

3.6. Random Access

Random access stands for fast access to any part of the encoded audio signal without costly decoding of previous parts. It is an important feature for applications that employ seeking, editing, or streaming of the compressed data.

In order to enable random access, the encoder has to insert frames that can be decoded without decoding previous frames. In those *random access frames*, no samples from previous frames may be used for prediction. The distance between random access frames can be chosen from 255 to one frame. Depending on frame length and sampling rate, random access down to some milliseconds is possible.

However, prediction at the beginning of random access frames still constitutes a problem. A conventional K-th order predictor would normally need K samples from the previous frame in order the predict the current frame's first sample. Since samples from previous frames may not be used, the encoder has either to assume zeros, or to transmit the first K original samples directly, starting the prediction at position $K + 1$.

As a result, compression at the beginning of random access frames would be poor. In order to minimize this problem, the MPEG-4 ALS codec uses *progressive prediction* [17], which makes use of as many available samples as possible. While it is of course not feasible to predict the first sample of a random access frame, we can use first-order prediction for the second sample, second-order prediction for the third sample, and so forth, until the samples from position $K + 1$ on are predicted using the full K-th order predictor (Figure 7).

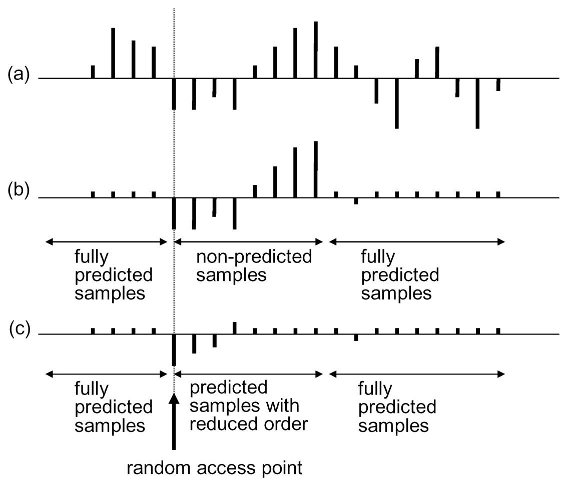

Fig. 7: Prediction in random access frames: (a) original signal, (b) residual for conventional prediction scheme, (c) residual for progressive prediction.

Since the predictor coefficients h_k are calculated recursively from the quantized parcor coefficients a_k anyway, it is possible to calculate each coefficient set from orders 1 to K without additional costs.

In the case of 500 ms random access intervals, this scheme produces an absolute overhead of only 0.01-0.02% compared to continuous prediction without random access.

3.7. Long-Term Prediction

It is well known that most audio signals have harmonic or periodic components originating from the fundamental frequency or pitch of musical instruments. For example, one period of a 220 Hz sine wave corresponds to 218 samples at 48 kHz sampling rate and to 872 samples at 192 kHz sampling rate. Such distant sample correlations are difficult to remove with the standard forward-adaptive predictor, since very high orders would be required, thus leading to an unreasonable amount of side information. In order to make more efficient use of the correlation between distant samples, MPEG-4 ALS employs a dedicated long-term prediction (LTP) scheme with lag and gain values as parameters.

At the encoder, the short-term LPC residual signal $e(n)$ of the standard predictor is additionally predicted using

$$\tilde{e}(n) = e(n) - \left(\sum_{j=-2}^{2} \gamma_{\tau+j} \cdot e(n-\tau+j) \right), \qquad (8)$$

where τ denotes the sample lag, γ denotes the quantized gain value, and $\tilde{e}$ denotes the new residual after long-term prediction. The most preferable lag (τ) and gain (γ) values are determined in order to reduce the amplitude of the residual signal, and these parameters are transmitted as side information. The LTP residual $\tilde{e}(n)$ constitutes a substitute for the short-term residual $e(n)$. Therefore, $\tilde{e}(n)$ is used instead of $e(n)$ for all further processing steps (including entropy coding and possibly multi-channel prediction).

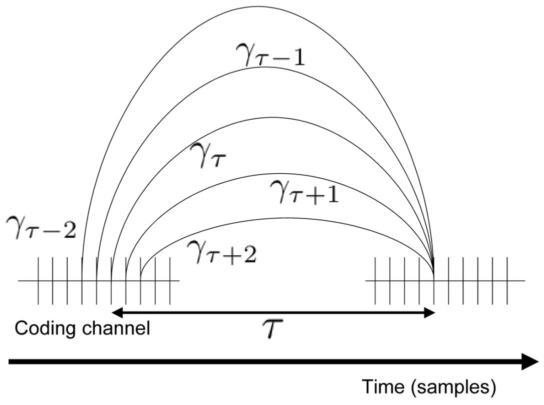

Fig. 8: Subtraction with long-term prediction.

At the decoder, the reverse process is carried out (Figure 8), using the following recursive filtering:

$$e(n) = \tilde{e}(n) + \left(\sum_{j=-2}^{2} \gamma_{\tau+j} \cdot e(n-\tau+j) \right). \qquad (9)$$

The reconstructed residual signal $e(n)$ is then used for short-term LPC synthesis again.

4. JOINT CHANNEL CODING

Joint channel coding can be used to exploit dependencies between the two channels of a stereo signal, or between any two channels of a multi-channel signal.

4.1. Difference Coding

While it is straightforward to process two channels $x_1(n)$ and $x_2(n)$ independently, a simple way to exploit dependencies between these channels is to encode the difference signal

$$d(n) = x_2(n) - x_1(n) \tag{10}$$

instead of $x_1(n)$ or $x_2(n)$. Switching between $x_1(n)$, $x_2(n)$ and $d(n)$ in each block can be carried out by comparison of the individual signals, depending on which two signals can be coded most efficiently (see Figure 9). Such prediction with switched difference coding is beneficial in cases where two channels are very similar. In the case of multi-channel material, the channels can be rearranged by the encoder in order to assign suitable channel pairs.

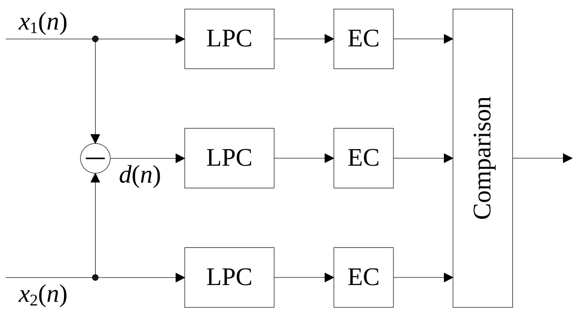

Fig. 9: Switched difference coding (LPC - prediction, EC - entropy coding).

Besides simple difference coding, MPEG-4 ALS also supports a more complex scheme for exploiting inter-channel redundancy between arbitrary channels of multi-channel signals, which is described in the following section.

4.2. Multi-Channel Coding

Lossless audio coding technology may be widely used for compressing various multi-channel signals, such as wave field synthesis signals, bio-medical and seismic signals as well as surround audio signals.

To improve compression performance for these multi-channel signals, adaptive subtraction from reference channels with weighting factors is applied based on inter-channel dependencies of the time domain prediction residual signal [18]. There are three modes for each channel and each frame. The first mode is independent coding, the second mode uses 3 taps, as shown in Figure 10, while the third mode uses 6 taps, additionally including 3 delayed taps as shown in Figure 11. At least one channel has to be encoded in independent coding mode in order to decode all channels losslessly.

For the 3-tap mode, the operation is performed according to

$$\hat{e}^c(n) = e^c(n) - \left(\sum_{j=-1}^{1} \gamma_j \cdot e^r(n+j)\right), \tag{11}$$

where $\hat{e}^c(n)$ and $e^c(n)$ are residual signals of the coding channel, and $e^r(n)$ is the residual of the reference channel. The reference channel is searched among available channels and the index is coded together with multi-tap gain parameters γ_j, which can be calculated by solving the normal equation

$$\boldsymbol{\gamma} = \mathbf{X}^{-1} \cdot \mathbf{y}, \tag{12}$$

where

$$\begin{aligned}
\boldsymbol{\gamma} &= (\gamma_{-1}, \gamma_0, \gamma_{+1})^T \\
\mathbf{X} &= \begin{pmatrix} \mathrm{e}^{r\,T}_{-1} \cdot \mathrm{e}^r_{-1} & \mathrm{e}^{r\,T}_{-1} \cdot \mathrm{e}^r_0 & \mathrm{e}^{r\,T}_{-1} \cdot \mathrm{e}^r_{+1} \\ \mathrm{e}^{r\,T}_{-1} \cdot \mathrm{e}^r_0 & \mathrm{e}^{rT}_0 \cdot \mathrm{e}^r_0 & \mathrm{e}^{rT}_0 \cdot \mathrm{e}^r_{+1} \\ \mathrm{e}^{r\,T}_{-1} \cdot \mathrm{e}^r_{+1} & \mathrm{e}^{rT}_0 \cdot \mathrm{e}^r_{+1} & \mathrm{e}^{r\,T}_{+1} \cdot \mathrm{e}^r_{+1} \end{pmatrix} \\
\mathbf{y} &= \left(\mathrm{e}^{cT} \cdot \mathrm{e}^r_{-1}, \mathrm{e}^{cT} \cdot \mathrm{e}^r_0, \mathrm{e}^{cT} \cdot \mathrm{e}^r_{+1}\right)^T \\
\mathbf{e}^c &= (e^c(0), e^c(1), e^c(2), \cdots, e^c(N-1))^T \\
\mathbf{e}^r_{-1} &= (e^r(-1), e^r(0), e^r(1), \cdots, e^r(N-2))^T \\
\mathbf{e}^r_0 &= (e^r(0), e^r(1), e^r(2), \cdots, e^r(N-1))^T \\
\mathbf{e}^r_{+1} &= (e^r(1), e^r(2), e^r(3), \cdots, e^r(N))^T
\end{aligned}$$

Here N is the number of samples per frame and $\mathbf{e}^T$ denotes the transposed vector of $\mathbf{e}$. The decoder reconstructs the original residual signal by applying simply the reverse operation:

$$e^c(n) = \hat{e}^c(n) + \left(\sum_{j=-1}^{1} \gamma_j \cdot e^r(n+j)\right) \tag{13}$$

The reconstructed residual signal $e^c(n)$ is used for short-term LPC synthesis or the preceding LTP decoding process.

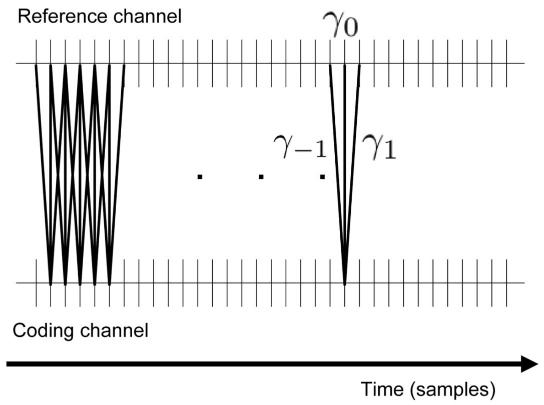

Fig. 10: Three-tap subtraction.

For the 6-tap mode, adaptive subtraction is carried out using

$$\hat{e}^c(n) = e^c(n) - \left(\sum_{j=-1}^{1} \gamma_j \cdot e^r(n+j) + \sum_{j=-1}^{1} \gamma_{\tau+j} \cdot e^r(n+\tau+j) \right), \qquad (14)$$

where the lag parameter τ can be estimated by cross correlation between the coding channel and the reference channel, and multi-tap gain parameters γ can be obtained by minimizing the energy of subtracted residual sequence, similar to Eq. (12). In the decoder, vice versa, the original residual signal can be reconstructed by the following process:

$$e^c(n) = \hat{e}^c(n) + \left(\sum_{j=-1}^{1} \gamma_j \cdot e^r(n+j) + \sum_{j=-1}^{1} \gamma_{\tau+j} \cdot e^r(n+\tau+j) \right) \qquad (15)$$

Again, the reconstructed residual signal $e^c(n)$ is used for short-term LPC synthesis or the LTP decoding process.

5. ENTROPY CODING OF THE RESIDUAL

In simple mode, the residual values $e(n)$ are entropy coded using Rice codes. For each block, either all values can be encoded using the same Rice code,

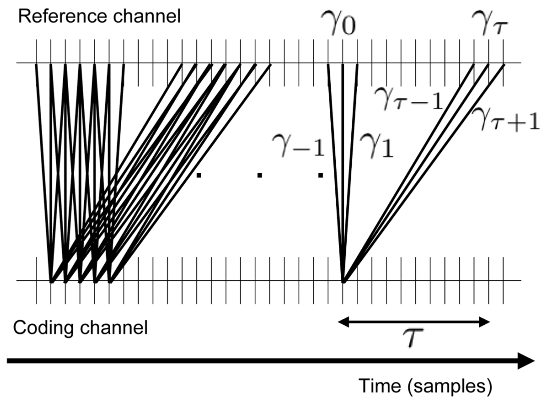

Fig. 11: Multi-tap subtraction with time difference.

or the block can be further divided into four parts, each encoded with a different Rice code. The indices of the applied codes have to be transmitted, as shown in Figure 2. Since there are different ways to determine the optimal Rice code for a given set of data, it is up to the encoder to select suitable codes depending on the statistics of the residual.

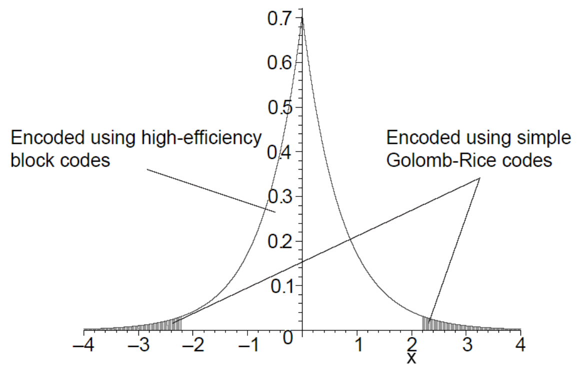

Fig. 12: Partition of the residual distribution.

Alternatively, the encoder can use a more complex and efficient coding scheme called BGMC (Block Gilbert-Moore Codes). In BGMC mode, the encoding of residuals is accomplished by splitting the distribution in two categories (Figure 12): Residuals that belong to a central region of the distribution, $|e(n)| < e_{\max}$, and ones that belong to its tails. The residuals in tails are simply re-centered (i.e. for $e(n) > e_{\max}$ we have $e_t(n) = e(n) - e_{\max}$) and encoded using Rice codes as described earlier. However, to encode residuals in the center of the distribution, the BGMC encoder splits them into LSB and MSB components first, then it encodes MSBs using block Gilbert-Moore (arithmetic) codes, and finally it transmits LSBs using direct fixed-lengths codes. Both parameters $e_{\max}$ and the number of directly transmitted LSBs are selected such that they only slightly affect the coding efficiency of this scheme, while making it significantly less complex.

More detailed descriptions of the entropy coding schemes used in MPEG-4 ALS are given in [3][19].

6. FLOATING-POINT AUDIO DATA

In addition to integer audio signals, MPEG-4 ALS also supports lossless compression of audio signals in the IEEE 32-bit floating-point format [20]. The floating-point sequence is modeled by the sum of an integer sequence and a residual sequence. The integer sequence is compressed using the basic ALS tools for integer data, while the residual sequence is compressed separately.

6.1. Encoder for Floating-Point Audio Data

Figure 13 shows the integrated lossless encoder for integer and floating-point data. In the proposed encoding scheme for 32-bit floating-point data, an input sequence **X** is decomposed into a common multiplier A, a multiplicand sequence **Y**, and a difference sequence **Z**. **X** and **Z** are vectors containing floating-point values. **Y** is a sequence of truncated integers and A is a scalar floating-point number. Thus, the input sequence can be written as

$$\mathbf{X} = A \otimes \mathbf{Y} + \mathbf{Z}, \tag{16}$$

where $\otimes$ is the multiplication with rounding. The rounding mode is set to ”round to nearest, to even when tie”.

First, the common multiplier A is estimated by analyzing the input signal vector **X** in the current frame using rational approximation of ACF (Approximate Common Factor). The common multiplier A is normalized to $1.0 \leq A < 2.0$. If an appropriate value of ACF can not be found in the multiplier estimation module, the common multiplier A is set to 1.0. In that case the input signal **X** is directly truncated to integer **Y**, and the number of necessary bits for the difference mantissa is uniquely determined by the corresponding truncated integer y.

The integer sequence **Y** is compressed using the basic ALS tools for integer data, while the difference sequence **Z** is separately compressed by the masked Lempel-Ziv tool.

6.2. Estimation of the ACF

Detecting the common multiplier A is not straightforward. So far, we have devised a reasonable estimation procedure by analyzing the input signal **X** in every frame, using a rational approximation of the ACF. The estimation is similar to finding the greatest common divisor with the condition that the input signal $\boldsymbol{X}$ may have an error at the ”unit in the last position” (*ulp*) due to rounding-off or chopping for truncation. The range of the error is between [-1/2, +1/2] of *ulp*. The relationship among the estimated values of A, y_i and x_i can be expressed as

$$x_i - \frac{1}{2} \cdot ulp_{x_i} \leq A \cdot y_i \leq x_i + \frac{1}{2} \cdot ulp_{x_i}, \tag{17}$$

where x_i is the mantissa part of the ith floating-point sample in **X**, y_i is the corresponding mantissa part of ith floating-point sample in **Y**, and A is a normalized common multiplier. Rational approximation using the continued fraction is applied to estimate the ACF. The interval function for the rational approximation is

$$\frac{x_i - \frac{1}{2} \cdot ulp_{x_i}}{\hat{x} + \frac{1}{2} \cdot ulp_{\hat{x}}} \leq \frac{y_i}{\hat{y}_i} \leq \frac{x_i + \frac{1}{2} \cdot ulp_{x_i}}{\hat{x} - \frac{1}{2} \cdot ulp_{\hat{x}}}, \tag{18}$$

where $\hat{x}$ is a selected sample of x_i in the frame, and $\hat{y}_i$ is the estimated corresponding value of y_i calculated by $\hat{x}$ and ith sample x_i. The common multiplier A is normalized to $1.0 \leq A < 2.0$ since the mantissa bits of the floating-point data are also normalized to $1.0 \leq A < 2.0$. The mantissa bits of $y \times 1.5$ and that of $y \times 3$ are the same if the original mantissa bits of y are the same.

Once a reliable A is found, computing **Y** and **Z** from **X** and A is straightforward.

6.3. Masked-Lempel-Ziv Compression

Masked-LZ compression is one kind of dictionary-based compression scheme. It is very similar to other Lempel-Ziv compression variants, such as the LZW compression scheme, in that there is a dictionary of previously encountered strings. The longest match string of input characters is searched using the string

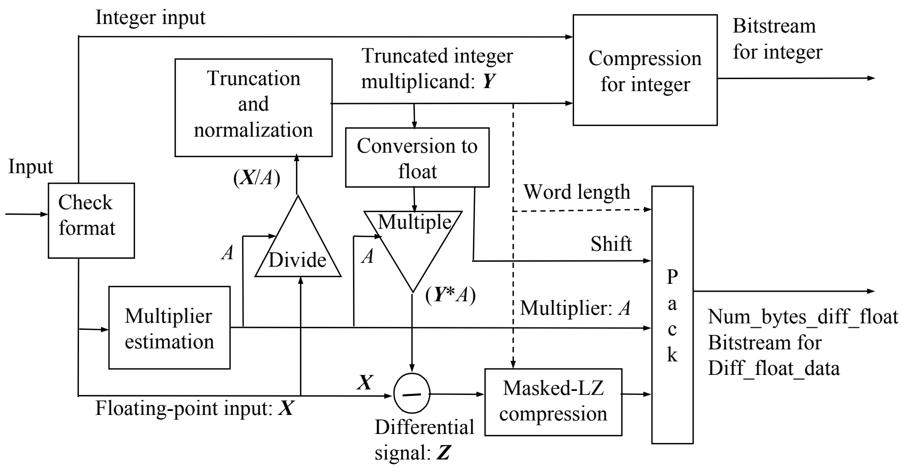

Fig. 13: MPEG-4 ALS encoder for floating-point audio signals.

stored in the dictionary. The main difference is the way in which the input characters are compared with the characters of the string stored in the dictionary. Masked-LZ uses a mask to compare them. The mask contains information about the bits of concern and not of concern and uses it to compare two characters.

6.4. Decoder for Floating-Point Audio Data

For floating-point data, the integer multiplicand sequence $\mathbf{Y}$ is reconstructed and the multiplier A is multiplied with it in order to obtain the floating-point sequence $(\mathbf{Y} \otimes A)$. The rounding mode "round to nearest, to even when tie" is used to round off the operation after the multiplication. The difference sequence is decoded by the Masked-LZ decompression module and converted to a floating-point format sequence $\mathbf{Z}$. If the multiplier A equals 1.0, the difference sequence is decoded using the word-length information, which is defined from the value of the corresponding integer value. Additional bits longer than the necessary bit length are cut off (thrown away) since they are dummy bits added by the encoder.

Both sequences, $(\mathbf{Y} \otimes A)$ and $\mathbf{Z}$, are summed to generate the output floating-point sequence. The operation of the floating-point multiplication, truncation, and summation are emulated by integer multiplication and summation in the decoding process. The integrated decoder is shown in Figure 14.

An extensive description of compression techniques for floating-point audio data can be found in [21].

7. COMPRESSION RESULTS

In the following, different encoding modes of the MPEG-4 ALS reference codec [22] are compared in terms of compression and complexity. The results for several audio formats were determined for a low complexity level ($K \leq 15$, Rice Coding), a medium level ($K \leq 30$, BGMC), and a maximum compression level ($K \leq 1023$, BGMC), all with random access of 500 ms. The results are also compared with the popular lossless audio codec FLAC [10] at maximum compression (`flac -8`).

Apart from the bitstream syntax, MPEG does not specify how to realize some encoder features such as predictor adaptation or block length switching. Even though we used the best ALS encoder implementation so far, future improvements in terms of compression, speed, and trade-off between those two are still possible.

The tests were conducted on a 1.7 GHz Pentium-M system, with 1024 MB of memory. The test material was taken from the standard set of audio sequences for MPEG-4 Lossless Coding. It comprises nearly 1 GB of stereo waveform data with sampling rates of 48, 96, and 192 kHz, and resolutions of 16 and 24 bits.

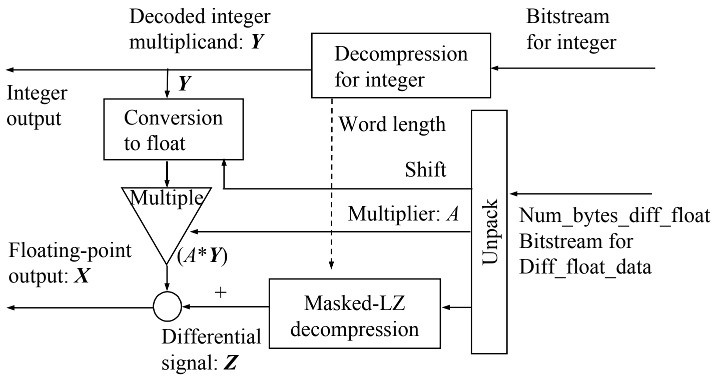

Fig. 14: MPEG-4 ALS decoder for floating-point audio signals.

7.1. Compression Ratio

In the following, the compression ratio is defined as

$$C = \frac{CompressedFileSize}{OriginalFileSize} \cdot 100\%, \qquad (19)$$

where smaller values mean better compression. The results for the examined audio formats are shown in Table 1 (192 kHz material is not supported by the FLAC codec).

Format	FLAC	ALS low	ALS medium	ALS max
48/16	48.6	46.5	45.3	44.6
48/24	68.4	63.9	63.2	62.7
96/24	56.7	47.4	46.3	46.1
192/24	–	38.4	37.6	37.5
Total	–	**48.9**	**48.1**	**47.7**

Table 1: Comparison of average compression ratios for different audio formats (kHz/bits).

The results show that MPEG-4 ALS at all complexity levels clearly outperforms FLAC, particularly for high-definition material (i.e. 96 kHz / 24-bit).

7.2. Complexity

The complexity of different codecs strongly depends on the actual implementation, particularly that of the encoder. As mentioned earlier, the ALS encoder is just a snapshot of an ongoing development. Thus, we essentially restrict our analysis to the ALS reference decoder [22], a simple C code implementation with no further optimizations. However, the low complexity level is in any case less complex than e.g. FLAC.

The average CPU load for real-time decoding of various audio formats, encoded at different complexity levels, is shown in Table 2. Even for maximum complexity, the CPU load of the MPEG-4 ALS reference decoder is only around 20-25%, which in return means that file based decoding is at least 4-5 times faster than real-time.

Format	ALS low	ALS medium	ALS max
48/16	1.6	4.7	18.2
48/24	1.8	5.3	19.1
96/24	3.6	11.6	23.1
192/24	6.7	19.4	24.4

Table 2: Average CPU load (percentage on a 1.7 GHz Pentium-M), depending on audio format (kHz/bits) and ALS encoder complexity.

The MPEG-4 ALS codec is designed to offer a wide range of operating points in terms of compression and complexity. While the maximum compression level achieves the highest compression at the expense of slowest encoding and decoding speed, the faster medium level only slightly degrades compression, but decoding is significantly less complex than for the maximum level (around 5% CPU load for 48 kHz material). Using the low complexity level only degrades compression by approximately 1% compared to the medium level, but the decoder complexity is further reduced by a factor of three (less than 2% CPU load for 48 kHz material). Thus, MPEG-4 ALS data can be decoded even on hardware with very low computing power.

7.3. Compression Ratio for Floating-Point Data

Since currently there are only few tools available for lossless compression of floating-point audio, MPEG-4 ALS is compared with WinZip [23], which is one of the most popular programs for dictionary based lossless compression.

A set of 48 kHz test data for floating-point was generated by weighted mixing of integer data using professional audio editing tools. The original integer input sequences sampled at 48 kHz in the stereo 24-bit integer format were chosen from the test sets for MPEG-4 Lossless Coding again. A set of 96 kHz test data, consisting of several original recordings, was supplied by a professional sound studio.

The results in Table 3 show that ALS compresses floating-point audio data much more effectively than a general-purpose compression tool such as WinZip.

Format	**Zip**	**ALS** medium
48 kHz / float	87.9	59.3
96 kHz / float	89.5	47.8

Table 3: Comparison of average compression ratios for 32-bit floating-point data.

8. APPLICATIONS

MPEG-4 ALS defines a simple architecture of efficient and fast lossless audio compression techniques for both professional and consumer applications. It offers many features not included in other lossless compression schemes:

- General support for virtually any uncompressed digital audio format.
- Support for PCM resolutions of up to 32-bit at arbitrary sampling rates.
- Multi-channel / multi-track support for up to 2^{16} channels (including 5.1 surround).
- Support for 32-bit floating-point audio data.
- Fast random access to the encoded data.
- Optional storage in MPEG-4 file format (allows multiplex with video and metadata).

Examples for the use of lossless audio coding in general and MPEG-4 ALS in particular include both professional and consumer applications:

- Archival systems (broadcasting, studios, record labels, libraries)
- Studio operations (storage, collaborative working, digital transfer)
- High-resolution disc formats
- Internet distribution of audio files
- Online music stores (download)
- Portable music players

In the case online music stores, downloads of the latest CD releases will no longer be restricted to lossy formats such as MP3 or AAC. Instead, the consumer can purchase all tracks in full quality of the original CD, but still receive the corresponding files at reduced data rates.

Furthermore, MPEG-4 ALS is not restricted to audio signals, since it can also be used to compress many other types of signals, such as medical (ECG, EEG) or seismic data.

A global standard will facilitate interoperability between different hardware and software platforms, thus promoting long-lasting multivendor support.

9. CONCLUSION

MPEG-4 Audio Lossless Coding (ALS) is a highly efficient and fast lossless audio compression scheme for both professional and consumer applications which offers many innovative features. It is based on a codec developed by Technical University of Berlin. Further improvements and extensions were contributed by RealNetworks and NTT.

Maximum compression can be achieved by means of high prediction orders together with efficient quantization of the predictor coefficients and adaptive block length switching. Using low and medium complexity modes, real-time encoding and decoding is possible even on low-end devices.

In the course of standardization, MPEG-4 ALS has reached the FDAM (Final Draft Amendment) stage [2] in July 2005. It is therefore expected to become an international standard by the end of 2005.

10. REFERENCES

[1] ISO/IEC 14496-3:2001, "Information technology - Coding of audio-visual objects - Part 3: Audio," *International Standard*, 2001.

[2] ISO/IEC JTC1/SC29/WG11 N7016, "Text of 14496-3:2001/FDAM 4, Audio Lossless Coding (ALS), new audio profiles and BSAC extensions," *73rd MPEG Meeting, Poznan, Poland*, July 2005.

[3] T. Liebchen and Y. Reznik, "MPEG-4 ALS: An Emerging Standard for Lossless Audio Coding," *Data Compression Conference, Snowbird, USA*, 2004.

[4] T. Liebchen, Y. Reznik, T. Moriya, and D. Yang, "MPEG-4 Audio Lossless Coding," *116th AES Convention*, 2004.

[5] T. Liebchen, "An Introduction to MPEG-4 Audio Lossless Coding," *Proc. IEEE ICASSP*, 2004.

[6] T. Liebchen and Y. Reznik, "Improved Forward-Adaptive Prediction for MPEG-4 Audio Lossless Coding," *118th AES Convention*, 2005.

[7] W. B. Kleijn and K. K. Paliwal, *Speech Coding and Synthesis*, Elsevier, Amsterdam, 1995.

[8] T. Robinson, "SHORTEN: Simple lossless and near-lossless wavform compression," *Technical report CUED/F-INFENG/TR.156, Cambridge University Engineering Department*, 1994.

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

[10] "FLAC - Free Lossless Audio Codec," *http://flac.sourceforge.net*.

[11] E. Janssen and D. Reefman, "Super Audio CD: An Introduction," *IEEE Signal Processing Magazine*, July 2003.

[12] G.-O. Glentis, K. Berberidis, and S. Theodoridis, "Efficient Least Squares Adaptive Algorithms For FIR Filtering," *IEEE Signal Processing Magazine*, July 1999.

[13] G. Schuller, B. Yu, and D. Huang, "Lossless Coding of Audio Signals Using Cascaded Prediction," *Proc. ICASSP 2001, Salt Lake City*, 2001.

[14] R. Yu and C. C. Ko, "Lossless Compression of Digital Audio Using Cascaded RLS-LMS Prediction," *IEEE Trans. Speech and Audio Processing*, July 2004.

[15] "Monkey's Audio," *www.monkeysaudio.com*.

[16] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[17] T. Moriya, D. Yang, and T. Liebchen, "A Design of Lossless Compression for High-Quality Audio Signals," *International Congress on Acoustics, Kyoto, Japan*, 2004.

[18] T. Moriya, D. Yang, and T. Liebchen, "Extended Linear Prediction Tools for Lossless Audio Coding," *Proc. IEEE ICASSP*, 2004.

[19] Y. Reznik, "Coding of Prediction Residual in MPEG-4 Standard for Lossless Audio Coding (MPEG-4 ALS)," *Proc. IEEE ICASSP*, 2004.

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

[21] N. Harada, T. Moriya, H. Sekigawa, and K. Shirayanagi, "Lossless Compression of IEEE Floating-Point Audio Using Approximate-Common-Factor Coding," *118th AES Convention*, 2005.

[22] "MPEG-4 ALS Reference Software," *ftp://ftlabsrv.nue.tu-berlin.de/mp4lossless/*.

[23] "WinZip," *www.winzip.com*.