MPEG-4 ALS – The Standard for Lossless Audio Coding

Tilman Liebchen

LG Electronics, Berlin, Germany

Published in:

The Journal of the Acoustical Society of Korea, vol. 28, no. 7, October 2009

Abstract

The MPEG-4 Audio Lossless Coding (ALS) standard belongs to the family MPEG-4 audio coding standards. In contrast to lossy codecs such as AAC, which merely strive to preserve the subjective audio quality, lossless coding preserves every single bit of the original audio data. The ALS core codec is based on forward-adaptive linear prediction, which combines remarkable compression with low complexity. Additional features include long-term prediction, multichannel coding, and compression of floating-point audio material. This paper describes the basic elements of the ALS codec with a focus on prediction, entropy coding, and related tools, and points out the most important applications of this standardized lossless audio format.

Keywords: Audio Coding, Lossless, Compression, Linear Prediction, MPEG, Standard, ALS

1. Introduction

Lossless audio coding enables the compression of digital audio data without any loss in quality due to a perfect (i.e. bit-identical) reconstruction of the original signal. On the other hand, modern *perceptual* audio coding standards such as MP3 or AAC are always *lossy*, since they never fully preserve the original audio data. Those lossy coding methods are typically not well suited for certain applications such as editing or archiving, since transcoding or post-processing can reveal originally masked distortions.

As a part of 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, also including 32-bit floating-point signals. Thus, virtually all known input formats from CD quality (44.1 kHz, 16-bit) to high-end audio (e.g. 96/192 kHz, 24-bit, multi-channel) are supported.

The first version of the MPEG-4 ALS standard was published in 2006 [2]. Since then, some corrigenda have been issued, and additional tools such as reference software [3] and conformance bitstreams [4] have been standardized and released. The latest description of

MPEG-4 ALS has been fully integrated into the 4th Edition of the MPEG-4 audio standard [5], which was ultimately published in 2009. Furthermore, MPEG has recently started the standardization of an ALS "Simple Profile" [6] on industry request in order to facilitate market adoption.

This article is based on a previous publication on MPEG-4 ALS [7], but contains some additional, revised, and updated information. The following chapters provide a detailed description of the codec. After an overview of the codec structure 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. Coding results for a variety of audio material (including high-resolution and multi-channel) are given in section 6, while section 7 provides a discussion of application scenarios for lossless audio coding in general and MPEG-4 ALS in particular.

2. Structure of the Codec

In most *lossy* MPEG coding standards, only the decoder is specified in detail. However, a *lossless* coding scheme usually requires the specification of some (but not all) encoder portions. Since the encoding process has to be perfectly reversible without loss of information, several parts of both encoder and decoder have to be implemented in a deterministic way.

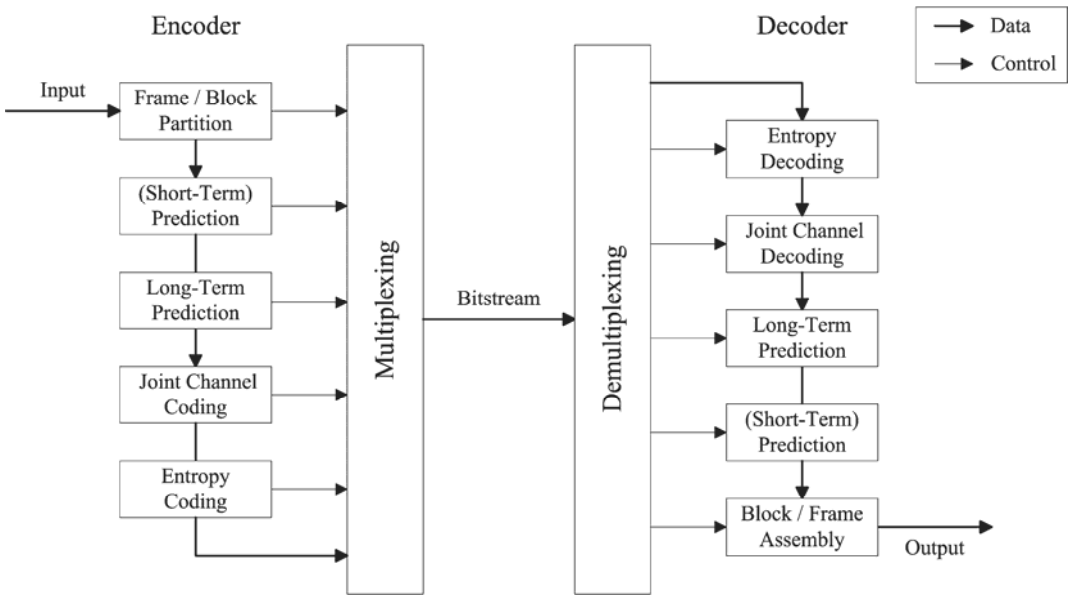

Fig. 1 – Block diagram of MPEG-4 ALS encoder and decoder.

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

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

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

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

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

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 [8].

Forward-adaptive prediction with orders around $K = 10$ is widely used in speech coding, and can be employed for lossless audio coding as well [9] [10]. The maximum order of most forward-adaptive lossless prediction schemes is still rather small, e.g. $K = 32$ [11]. An exception is the special 1-bit lossless codec for the Super Audio CD, which uses predictor orders of up to 128 [12].

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 [13]. Most systems are based on the LMS algorithm or a variation thereof, which has also been proposed for lossless audio coding [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, but in the following, only the forward-adaptive predictor and related tools are discussed.

3.2. Forward-Adaptive Prediction

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

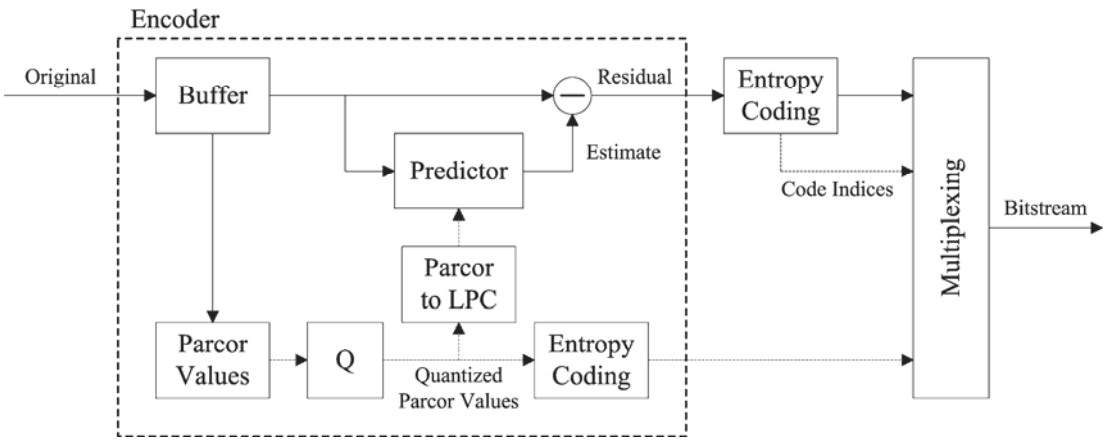

Fig. 2 – Encoder of the forward-adaptive prediction scheme.

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 [9]. Furthermore, the algorithm inherently calculates the corresponding parcor coefficients as well.

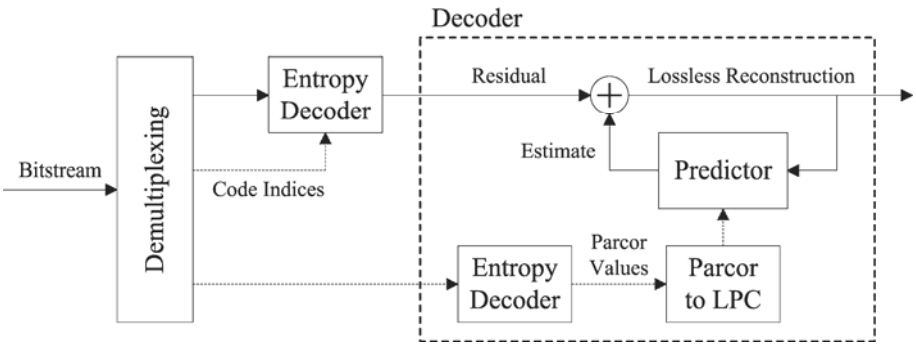

Figure 3 – Decoder of the forward-adaptive prediction scheme.

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

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

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 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 [8]. 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).

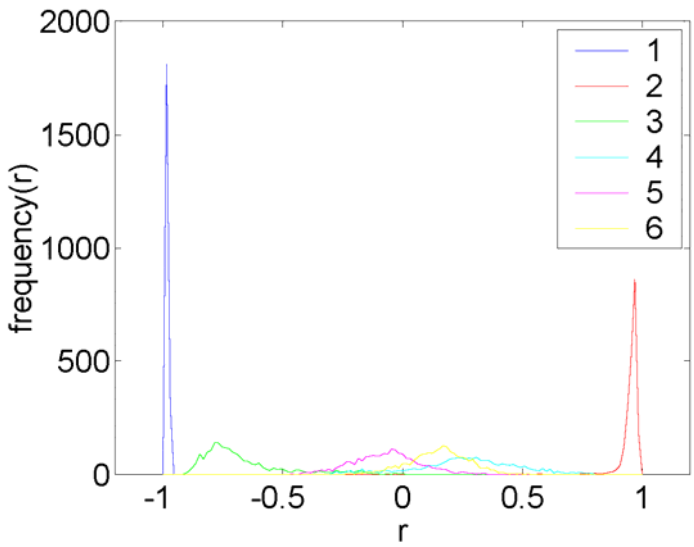

Fig. 4 – Measured distributions of parcor coefficients $r_1 \ldots r_6$, for 48 kHz, 16-bit audio material.

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

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

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

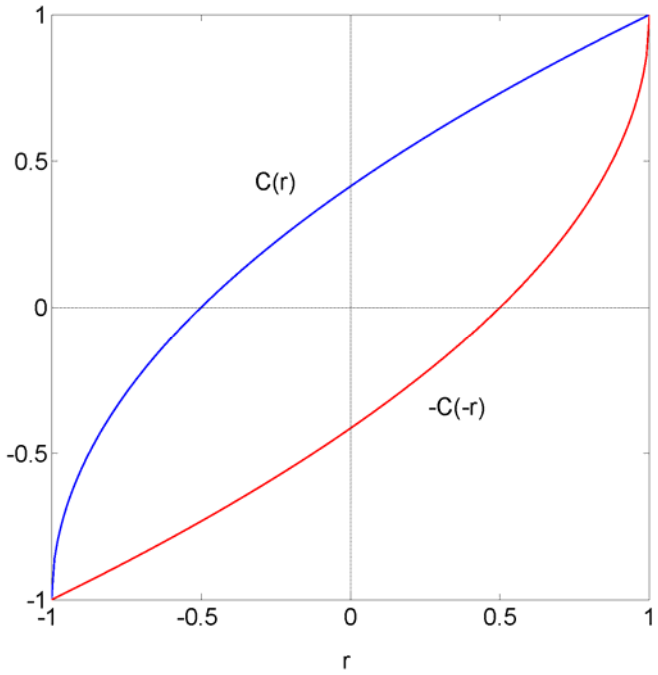

Fig. 5 – Compander functions $C(r)$ and $-C(-r)$.

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:

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

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

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

$$a_k = \lfloor 64 r_k \rfloor$$

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 the predictor coefficients h_k as described in section 3.1. 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 a fixed-point 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 block of samples 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. 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 twice the 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).

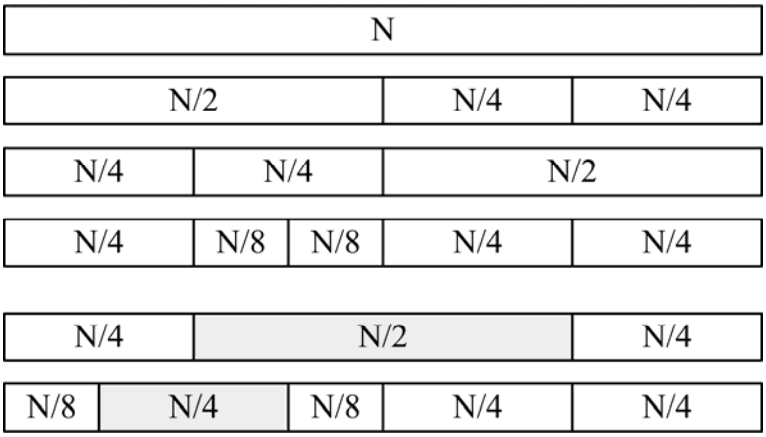

Fig. 6 – Block switching examples. The last two partitions are not allowed due to the positions of the shaded blocks.

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 in the standard. Possible methods may range from evaluating of the signal statistics to closed-loop 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 could either assume zeros, or 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).

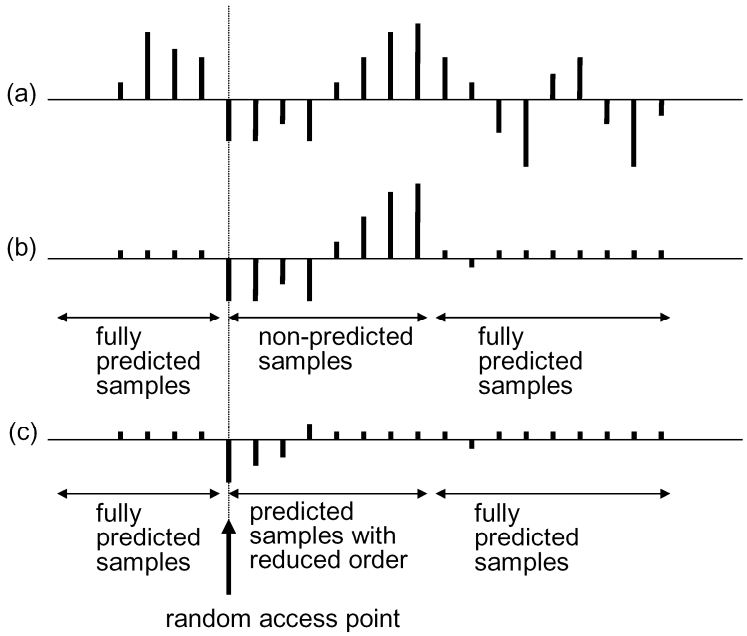

Fig. 7 – Prediction in random access frames: (a) original signal, (b) residual for conventional prediction scheme, (c) residual for progressive prediction.

Since the predictor coefficients h_k are calculated recursively from the quantized parcor coefficients a_k anyway, it is possible to calculate each set of coefficients 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

$$\varepsilon(n) = e(n) - \sum_{j=-2}^{2} \gamma_{\tau+j} \cdot e(n-\tau+j),$$

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

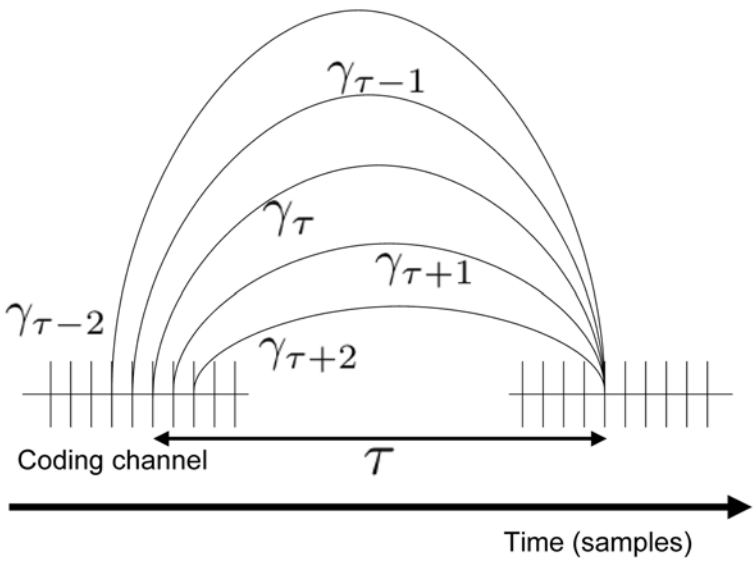

Fig. 8 – Subtraction with long-term prediction.

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

$$e(n) = \varepsilon(n) + \sum_{j=-2}^{2} \gamma_{\tau+j} \cdot e(n-\tau+j)$$

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

$$d(n) = x_2(n) - x_1(n)$$

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.

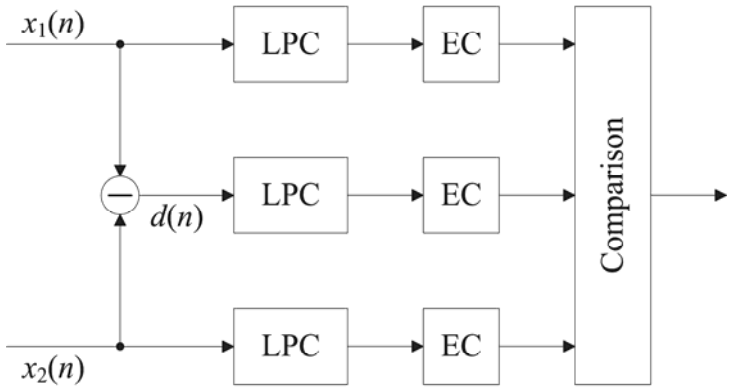

Fig. 9 – Switched difference coding (LPC – prediction, EC – entropy coding).

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

The MPEG-4 ALS standard defines a cross prediction scheme called MCC (*Multi-Channel Coding*) that can be optionally used as a substitute for difference coding, or as a supplement to it. The MCC scheme operates in the residual domain, i.e. it is applied after normal prediction (and possibly LTP). Each residual channel $e_c(n)$ can be predicted using any other channel $e_r(n)$, $r \neq c$, the so-called reference channel, while each reference channel can be used for the prediction of several channels [18].

MCC can be applied on a frame by frame basis, depending on whether it improves compression compared to independent coding. In addition, it can also be switched with difference coding of channel pairs in each frame.

There are two different types of cross prediction filters that can be chosen from. The first type is a 3-tap filter that calculates the prediction error according to

$$\varepsilon_c(n) = e_c(n) - \sum_{j=-1}^{1} \gamma_j \cdot e_r(n+j),$$

where γ_j are the MCC coefficients. The second type of cross prediction uses a 6-tap prediction filter consisting of the 3-tap filter described above, together with an additional 3-tap long-term prediction filter, thus

$$\varepsilon_c(n) = e_c(n) - \sum_{j=-1}^{1} \gamma_j \cdot e_r(n+j) - \sum_{j=-1}^{1} \gamma_{\tau+j} \cdot e_r(n+\tau+j),$$

where τ is the lag and $\gamma_{\tau+j}$ are the coefficients of the long-term part, as shown in Figure 10. The possible range of lag values is restricted and depends on the sampling frequency.

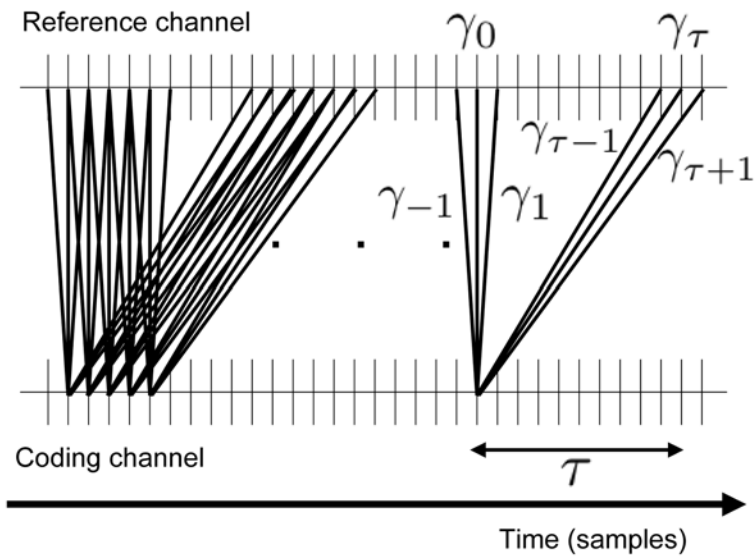

Fig. 10 – MCC with 6-tap filter.

There are different methods to determine the best lag and the optimum coefficients [19] [20], but the standard merely defines their transmission format. For each channel, the reference channel has to be indicated. The chosen lag is directly transmitted, while the coefficients are quantized and entropy coded prior to transmission. Due to this forward-adaptive approach, the parameter estimation is entirely left to the encoder, while the decoder can reconstruct the original residuals $e_c(n)$ by simply applying the inverse MCC filters. Some compression results for multi-channel signals using MCC are provided in section 6.2.

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, 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 already 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.

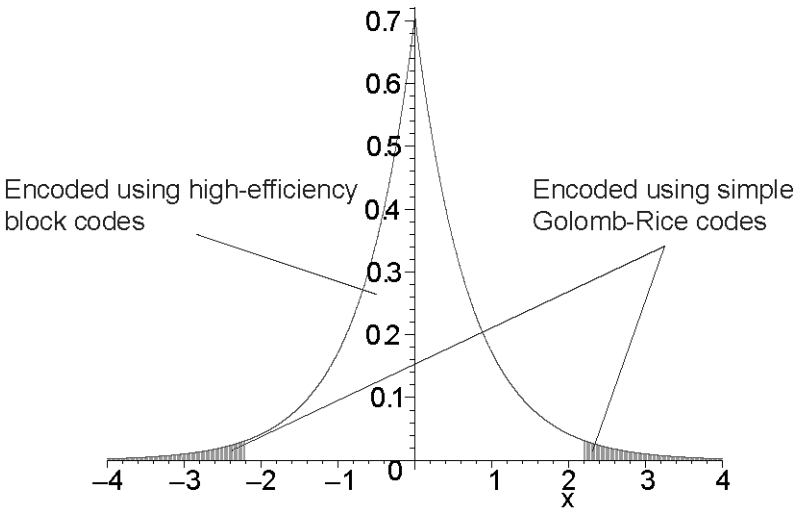

Fig. 11 – Partition of the residual distribution.

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 11): Those residuals that belong to a central region of the distribution, $|e(n)| < e_{max}$, and those that belong to its tails. The residuals in tails are simply re-centered (i.e. for $e(n) > e_{max}$ we have $e_0(n) = e(n) - e_{max}$) and encoded using Rice codes as described earlier. However, to encode residuals in the center of the distribution, the BGMC encoder splits them into MSB and LSB components first, then it encodes MSBs using block Gilbert-Moore (arithmetic) codes, and finally it transmits the (noise-like) 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.

Some more detailed descriptions of the entropy coding schemes used in MPEG-4 ALS are given in [21] [22].

6. Compression Results

In the following, different encoding modes of the MPEG-4 ALS reference codec [3] 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, no LTP, no MCC), 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 [11] at maximum compression ("flac --best").

Apart from the bitstream syntax, the ALS standard does not specify how to realize some encoder features such as predictor adaptation or block length switching. Since the ALS reference encoder is not an optimized implementation, further improvements in terms of compression, speed, and trade-off between those two are still possible.

6.1. Stereo Results

The stereo 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. In the following, the compression rate is defined as the remaining percentage of data, i.e.

$$C = \frac{\text{Compressed File Size}}{\text{Original File Size}} \cdot 100\% ,$$

thus smaller values stand for better compression. The results for all examined audio formats are listed in Table 1.

Format (stereo)	**FLAC** (best)	**ALS** (low)	**ALS** (medium)	**ALS** (max)
48 kHz / 16-bit	48.6	46.5	45.3	44.6
48 kHz / 24-bit	68.4	63.9	63.2	62.7
96 kHz / 24-bit	56.7	47.4	46.3	46.1
192 kHz / 24-bit	45.3	38.4	37.6	37.5
Average	**54.8**	**49.1**	**48.1**	**47.7**

Table 1 – Comparison of average compression rates for different audio formats.

The results show that MPEG-4 ALS at all complexity levels clearly outperforms FLAC, particularly for high-definition material (e.g. 96 kHz / 24-bit). On average, ALS provides a relative improvement of more than 10%.

6.2. Multi-Channel Results

The 5.1 multi-channel material for this test was taken from high resolution (96 kHz, 24-bit) DVD-Audio and Super Audio CD (SACD) releases. The analog output signals were then digitized again with 48 kHz sampling rate and 16-bit resolution. The compression results, which are classified into different music categories, are shown in Table 2.

Music Category	**No ICC** (absolute)	**JS** (relative)	**MCC** (relative)	**MCC+JS** (relative)
Jazz	36.0	0.0	0.4	0.4
Pop	38.7	0.9	2.2	2.2
Classical	32.8	0.0	0.1	0.1
Rock	33.3	1.7	1.6	1.8
Average	**35.2**	**0.65**	**1.08**	**1.13**

Table 2 – Compression rates for 5.1 multi-channel data (48 kHz / 16-bit) and relative savings by different inter-channel coding methods.

Most importantly, it should be noted that ALS compresses all of these 5.1 signals to around one third of their original size, even if no inter-channel coding is used. The better compression compared to stereo signals is characteristic, since the rear channels and (in particular) the LFE channel typically contain less information.

The additional savings depend on both the music category and the applied inter-channel coding methods. The classical category does not really benefit from any of the methods, while the jazz category only shows small savings for MCC. Significant improvements are achieved for the pop and rock categories, with savings of up to 1.7% for 2-channel difference coding (JS) and up to 2.2% for 6-channel MCC.

6.3. Complexity

The complexity of different codecs strongly depends on the actual implementation, particularly that of the encoder. Thus, we essentially restrict our analysis to the ALS reference decoder [3], a simple C code implementation with no further optimizations. The tests were conducted on a 1.7 GHz Pentium-M system with 1024 MB of memory, whose performance is roughly comparable to the latest Netbook computers with Atom CPU. The average CPU load for real-time decoding of various audio formats, encoded at the different complexity levels described earlier, is shown in Table 3.

Format (stereo)	**ALS** (low)	**ALS** (medium)	**ALS** (max)
48 kHz / 16-bit	1.6	4.7	18.2
48 kHz / 24-bit	1.8	5.3	19.1
96 kHz / 24-bit	3.6	11.6	23.1
192 kHz / 24-bit	6.7	19.4	24.4

Table 3 – Average CPU load (percentage on a 1.7 GHz Pentium-M), depending on audio format and encoder complexity.

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. At the lower complexity levels, the CPU load is almost negligible. Furthermore, these results indicate that even high resolution multi-channel material (e.g. 5.1 or 7.1) can be easily decoded in real-time.

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. A more detailed complexity analysis can be found in [23] [24] [25].

7. Applications

MPEG-4 ALS defines a simple architecture of efficient and fast lossless audio compression techniques for professional and consumer applications. It offers many interesting features, most of which are 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 [26]
- Fast random access to the encoded data
- Storage in the MPEG-4 file format (allows multiplex with video and metadata)
- High flexibility of codec parameters for various applications

Examples for the use of lossless audio coding in general and MPEG-4 ALS in particular can be found in both the professional and consumer market:

- 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 of 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

with the original quality of the CD, but still receive the corresponding files at reduced data rates.

MPEG-4 ALS can be easily transcoded into arbitrary other (lossy or lossless) formats, e.g. for legacy devices that only support MP3, or if allowed bit rates are very restricted. In contrast to transcoding between different lossy formats, starting from a lossless representation always allows to achieve the best possible quality.

Furthermore, MPEG-4 ALS is not restricted to audio signals, since it can also be used to compress many other types of time-series signals, such as seismic or medical (ECG, EEG) data [27].

8. 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.

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.

By all means, this global standard for lossless audio coding will facilitate interoperability between different hardware and software platforms, thus promoting long-lasting multivendor support.

References

[1] ISO/IEC 14496-3:2005, “Information technology – Coding of audio-visual objects – Part 3: Audio,” International Standards Organization, Geneva, Switzerland, 2005.

[2] ISO/IEC 14496-3:2005/Amd.2:2006, “Information technology – Coding of audio-visual objects – Part 3: Audio, Amendment 2: Audio Lossless Coding (ALS), new audio profiles and BSAC extensions,” International Standards Organization, Geneva, Switzerland, 2006.

[3] ISO/IEC 14496-5:2001/Amd.10:2007, “Information technology – Coding of audio-visual objects – Part 5: Reference Software, Amendment 10: SSC, DST, ALS and SLS reference software,” International Standards Organization, Geneva, Switzerland, 2007, ALS reference software available at http://www.nue.tu-berlin.de/mp4als.

[4] ISO/IEC 14496-4:2004/Amd.19:2007, “Information technology – Coding of audio-visual objects – Part 5: Conformance Testing, Amendment 19: Audio Lossless Coding (ALS),” International Standards Organization, Geneva, Switzerland, 2007.

[5] ISO/IEC 14496-3:2009, "Information technology – Coding of audio-visual objects – Part 3: Audio (4th Edition)," International Standards Organization, Geneva, Switzerland, 2009.

[6] ISO/IEC JTC1/SC29/WG11 (MPEG), "ISO/IEC 14496-3:200x/PDAM 2, ALS Simple Profile and Transport of SAOC," Doc. N10826, London, UK, July 2009.

[7] T. Liebchen, T. Moriya, N. Harada, Y. Kamamoto, and Y. Reznik, "The MPEG-4 Audio Lossless Coding (ALS) Standard – Technology and Applications," 119th AES Convention, New York, USA, 2005.

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

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

[10] 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, Los Angeles, 1996.

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

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

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

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

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

[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, Montreal, Canada, 2004.

[19] Y. Kamamoto, N. Harada, and T. Moriya, "A Multichannel Linear Prediction Method for the MPEG-4 ALS Compliant Encoder," Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, USA, 2007.

[20] Y. Kamamoto, N. Harada, and T. Moriya, "Multichannel Linear Prediction Method Compliant with the MPEG-4 ALS", IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E91-A, March 2008.

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

[22] Y. Reznik, “Coding of Prediction Residual in MPEG-4 Standard for Lossless Audio Coding (MPEG-4 ALS),” Proc. IEEE ICASSP, Montreal, Canada, 2004.

[23] ISO/IEC JTC1/SC29/WG11 (MPEG), “Verification Report on MPEG-4 ALS,” Doc. N7686, Nice, France, October 2005, available at http://www.chiariglione.org/mpeg/working_documents/mpeg-04/audio/als_vt.zip.

[24] T. Moriya and N. Harada and Y. Kamamoto, “An enhanced encoder for the MPEG-4 ALS Lossless Coding standard,” 121st AES Convention, San Francisco, 2006.

[25] N. Harada, T. Moriya, and Y. Kamamoto, “An implementation of MPEG-4 ALS standard compliant decoder on ARM core CPUs,” 125th AES Convention, San Francisco, 2008.

[26] N. Harada, T. Moriya, H. Sekigawa, and K. Shirayanagi, “Lossless Compression of IEEE Floating-Point Audio Using Approximate-Common-Factor Coding,” 118th AES Convention, Barcelona, Spain, 2005.

[27] Y. Kamamoto, N. Harada, and T. Moriya, “Interchannel Dependency Analysis of Biomedical Signals for Efficient Lossless Compression by MPEG-4 ALS,” Proc. IEEE ICASSP , Las Vegas, USA, 2008.