Audio Engineering Society
Convention Paper
Presented at the 116th Convention
2004 May 8–11 Berlin, Germany

MPEG-4 Audio Lossless Coding

Tilman Liebchen[1], Yuriy Reznik[2], Takehiro Moriya[3], and Dai Tracy Yang[4]

[1] *Technical University of Berlin, Germany*

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

[3] *NTT Human and Information Science Lab, Atsugi, Japan*

[4] *University of Southern California, Los Angeles, CA, USA*

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

ABSTRACT

Lossless coding will become the latest extension of the MPEG-4 audio standard. The lossless audio codec of the Technical University of Berlin was chosen as reference model for *MPEG-4 Audio Lossless Coding (ALS)*. The MPEG-4 ALS encoder is based on linear prediction, which enables high compression even with moderate complexity, while the corresponding decoder is straightforward. The paper describes the basic elements of the codec as well as some additional features, gives compression results, and points out envisaged applications.

1. INTRODUCTION

Lossless audio coding enables the compression of digital audio data without any loss in quality due to a perfect reconstruction of the original signal. The MPEG audio subgroup is currently working on the standardization of lossless coding techniques for high-definition audio signals. As an extension to MPEG-4 Audio [1], the amendment

"ISO/IEC 14496-3:2001/AMD 4, Audio Lossless Coding (ALS)" will define efficient methods for lossless coding.

1.1. History

In July 2002, MPEG issued a call for proposals [2] to initiate the submission of technology for lossless audio coding. This call basically supported two different approaches, either a hierarchical system consisting of a lossy core codec (e.g. MPEG-AAC [3]) and a lossless enhancement layer, or a lossless-only codec. By December 2002, seven companies submitted one or more codecs which met the basic requirements. In the following, those submissions were evaluated in terms of compression efficiency, complexity and flexibility [4]. In March 2003, the audio subgroup decided to proceed at first with the standardization of a lossless-only codec, while further investigating hierarchical methods as well.

The lossless-only codec of the Technical University of Berlin (TUB), which offered the highest compression among all submissions, was chosen as reference model [5]. In July 2003, the codec attained working draft status. In October 2003, an alternative entropy coding scheme, proposed by RealNetworks, was added to enable even better compression, and in December 2003, extensions by TUB for coding of multi-channel and 32-bit material were integrated [6]. A proposal from NTT to support floating-point audio data is expected to be adopted in March 2004.

1.2. Features

MPEG-4 ALS defines 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 256 channels (including 5.1 surround).
- Support for 32-bit floating-point audio data.
- Fast random access to the encoded data.
- Optional storage in MP4 file format (allows multiplex with video).

Below, we describe the basics of the MPEG-4 ALS Codec [7], give some compression results, point out applications and outline the standardization process.

2. CODEC OVERVIEW

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.

The MPEG-4 ALS codec uses forward-adaptive *Linear Predictive Coding (LPC)* to reduce bit rates compared to PCM, leaving the optimization entirely to the encoder. Thus, various encoder implementations are possible, offering a certain range in terms of efficiency and complexity. This section gives an overview of the basic encoder and decoder functionality.

2.1. Encoder Overview

The MPEG-4 ALS encoder (Figure 1) typically consists of these main building blocks:

- *Buffer*: Stores one audio frame. A frame is divided into blocks of samples, typically one for each channel.
- *Coefficients Estimation and Quantization*: Estimates (and quantizes) the optimum predictor coefficients for each block.
- *Predictor*: Calculates the prediction residual using the quantized predictor coefficients.
- *Entropy Coding*: Encodes the residual using different entropy codes.
- *Multiplexing*: Combines coded residual, code indices and predictor coefficients to form the compressed bitstream.

For each channel, a prediction residual is calculated using linear prediction with adaptive predictor coefficients and (preferably) adaptive prediction order

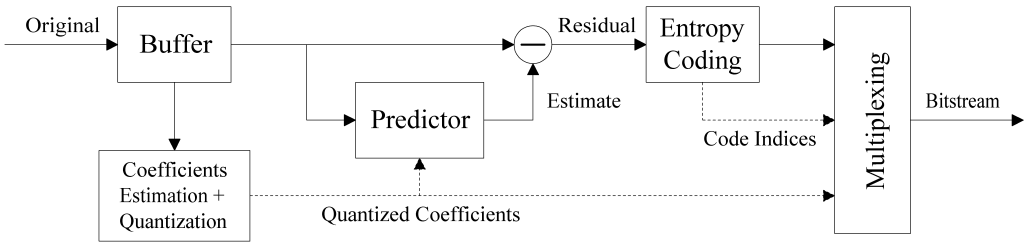

Fig. 1: MPEG-4 ALS encoder

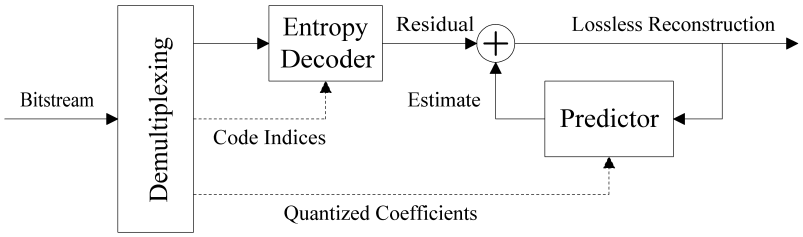

Fig. 2: MPEG-4 ALS decoder

in each block. The coefficients are quantized prior to filtering and transmitted as side information. The prediction residual is entropy coded using one of several different entropy codes. The indices of the chosen codes have to be transmitted. Finally, a multiplexing unit combines coded residual, code indices, predictor coefficients and other additional information to form the compressed bitstream. The encoder also provides a CRC checksum, which is supplied mainly for the decoder to verify the decoded data. On the encoder side, the CRC can be used to ensure that the compressed data is losslessly decodable.

Additional encoder options comprise block length switching, random access and joint stereo coding (see section 3). The encoder might use these options to offer several compression levels with differing complexities. However, the differences in terms of coding efficiency usually are rather small, so it may be appropriate to abstain from the highest compression in order to reduce the computational effort. Coding results for a variety of audio material will be given in section 4.

2.2. Decoder Overview

The MPEG-4 ALS decoder (Figure 2) is significantly less complex than the encoder. It decodes the entropy coded residual and, using the predictor coefficients, calculates the lossless reconstruction signal. The computational effort of the decoder mainly depends on the order of the predictor chosen by the encoder. Since the maximum order usually depends on the encoder's compression level, higher compressed files might take slightly longer to decode. Apart from the predictor order, the decoder complexity is nearly independent from the encoder options.

3. ENCODER DESCRIPTION

3.1. Linear Prediction

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

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

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

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

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

In forward linear prediction, the optimal predictor coefficients h_k (in terms of a minimized variance of

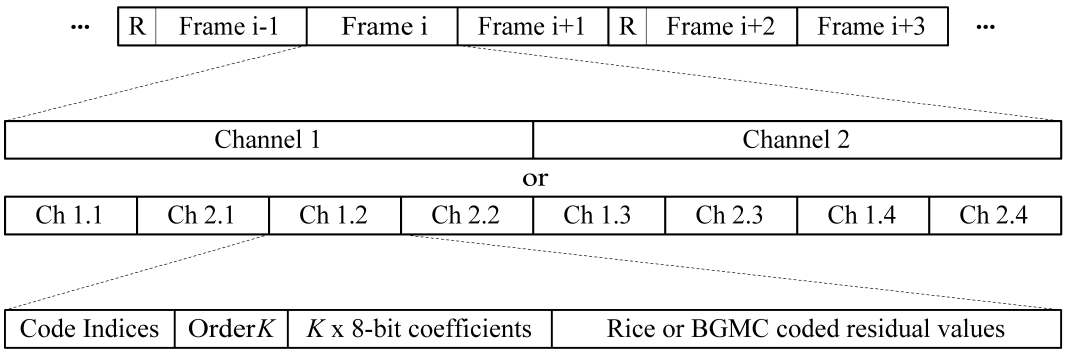

Fig. 3: Bitstream structure (R – random access information)

the residual) are usually estimated for each block by the autocorrelation method or the covariance method [8]. 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].

Increasing the predictor order decreases the variance of the prediction error, leading to a smaller bit rate for the residual. On the other hand, the bit rate for the predictor coefficients will rise with the number of coefficients to be transmitted. Thus, the task is to find the optimal order which minimizes the total bit rate.

The Levinson-Durbin algorithm 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 calculated, 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 predictor order. The optimal order is set at the point where the total bit rate no longer decreases.

3.2. Quantization of Predictor Coefficients

Direct quantization of the predictor coefficients h_k is not very efficient for transmission, since even small quantization errors might produce large spectral errors. Although parcor (reflection) coefficients are less sensitive to quantization, they are still too sensitive when their magnitude is close to unity. In order to expand the region near unity, an arcsine function is applied to the parcor coefficients. The behavior of the resulting *arcsine coefficients* is very similar to the more familiar log-area ratio (LAR) coefficients.

For a predictor filter of order K, a set of parcor coefficients $\gamma_k, k = 1 \ldots K$, can be estimated using the Levinson-Durbin recursion. Those coefficients are converted to arcsine coefficients using

$$\alpha_k = \arcsin(\gamma_k). \tag{3}$$

The value of each α_k is restricted to $[-\pi/2, +\pi/2]$. A linear 8-bit quantization is applied to the arcsine coefficients, which is equivalent to a non-linear quantization of the corresponding parcor coefficients. Only the 8-bit indices of the quantized arcsine coefficients are finally transmitted.

However, the direct form predictor filter uses predictor coefficients h_k according to (1). In order to employ identical coefficients in the encoder and the decoder, the h_k values have to be derived from the quantized arcsine values in both cases.

While it is up to the encoder how to determine a set of suitable arcsine coefficients, the conversion of those values α_k back to predictor coefficients h_k has to be exactly the same in both encoder and decoder.

3.3. Block Length Switching

The basic version of the encoder uses one sample block per channel in each frame, where 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

into four shorter sub-blocks to adapt to transient segments of the audio signal. Thus, either one long block or four short sub-blocks are used in each frame, e.g. 1×4096 or 4×1024 samples (Figure 3).

3.4. Random Access

Random access enables fast access to any part of the encoded audio signal without costly decoding of previous parts. The encoder optionally generates bitstream information allowing random access at intervals of several frames by inserting frames that can be decoded without decoding previous frames. In those *random access frames*, no samples from previous frames are used for prediction. Each random access frame starts with an info field (Figure 3) that specifies the distance in bytes to the next random access frame, thus enabling a fast search inside the compressed file.

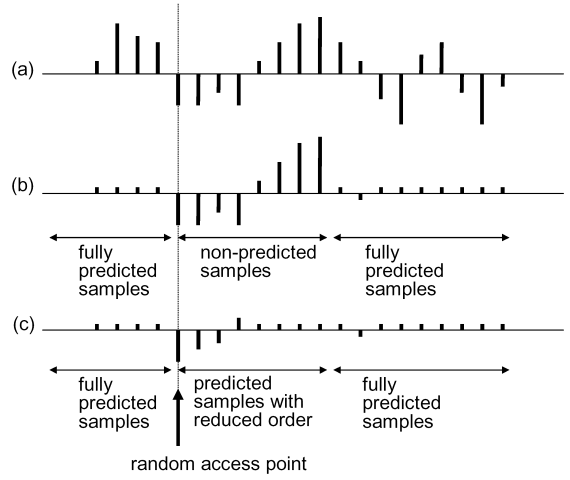

Fig. 4: Prediction in random access frames: (a) original signal, (b) residual for conventional prediction scheme, (c) residual for progressive prediction

Prediction at the beginning of random access frames typically is poor, since samples from the end of the previous frame cannot be used, and the first K samples have to be transmitted directly. In order to minimize this problem, a method called *progressive prediction* [10], which makes use of all samples available, was recently proposed to MPEG. While it is of course not possible to predict the first sample of an 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 4). Since the LPC predictor coefficients are calculated recursively from the quantized arcsin/parcor coefficients in any case, it is possible to calculate each LPC coefficient set from orders 1 to K without additional costs.

In the case where each frame allows random access, the conventional scheme produces an absolute overhead of 0.2-0.3% compared to coding without random access. Progressive prediction reduces this overhead to approximately 0.05%.

3.5. 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. 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

$$d(n) = x_2(n) - x_1(n) \tag{4}$$

instead of $x_1(n)$ or $x_2(n)$. Switching between $x_1(n)$, $x_2(n)$ and $d(n)$ in particular frames depends on which two signals can be coded most efficiently. Such prediction with switchable difference coding is beneficial in cases where two channels are very similar. In the case of multi-channel material, the channels can be rearranged internally in order to assign suitable channel pairs.

3.6. Entropy Coding of the Residual

In default 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 shown in Figure 1. 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.

Alternatively, the encoder can use a more complex and efficient coding scheme called BGMC (Block Gilbert-Moore Codes), proposed by RealNetworks [11]. In BGMC mode, the encoding of residuals is

accomplished by splitting the distribution in two categories (Figure 5): 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.

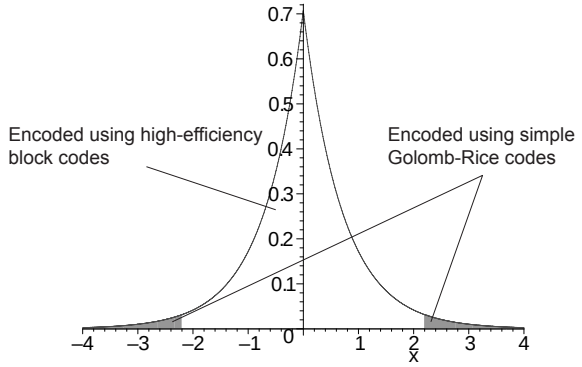

Fig. 5: Partition of the residual distribution.

A more detailed description of the entropy coding schemes used in MPEG-4 ALS is given in [12] and [13].

3.7. Floating-Point Audio Data

While conventional lossless audio codecs only support PCM data, MPEG-4 ALS will also offer efficient compression of floating-point material. The 32-bit IEEE floating-point format [14] is widely used in professional audio applications, where audio signals are typically normalized to ±1.0.

The lossless floating-point compression scheme was originally proposed by NTT [15]. The algorithm splits the floating-point signal into a truncated integer signal and a difference signal which contains the remaining fractional part. The integer signal is then compressed using the MPEG-4 ALS encoder

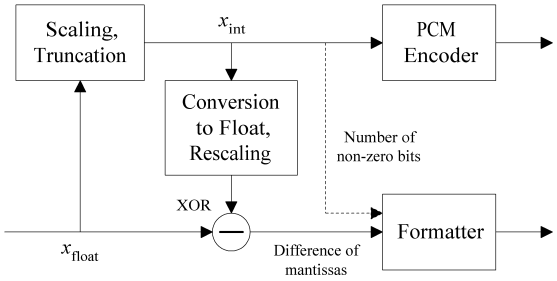

Fig. 6: Floating-point encoder.

for PCM signals, while the difference signal is coded separately (Figure 6).

The floating-point signal is converted to integer by scaling with a factor (e.g. 2^{15} for 16-bit PCM) and subsequent truncation. This integer value is then converted to float and rescaled again, resulting in a float value with the same sign and exponent as the original one. The subtraction from the original float value can therefore be carried out by a bit-wise XOR of the mantissas. The number m of resulting non-zero mantissa bits can be determined from the truncated integer value x_{int} using

$$m = \begin{cases} 23 - k & \text{if } 2^k \le x_{int} < 2^{k+1} \\ 32 & \text{if } x_{int} = 0 \end{cases} \qquad (5)$$

For each sample, only m bits for the difference of mantissas have to be transmitted.

The decoder simply decodes the PCM signal and the difference signal, where the number of difference bits to decode for each sample is calculated using Eq. 5. The PCM signal is converted to float and rescaled as in the encoder, and the difference is then added to its mantissa in order to obtain the original float value.

An extensive description of the floating-point coding scheme can be found in [16].

4. COMPRESSION RESULTS

In the following, the MPEG-4 ALS encoder [17] was compared with the two most popular programs for lossless audio compression: The open-source codec FLAC [18], and Monkey's Audio (MAC) [19], a codec identified by MPEG as the current state-of-the-art algorithm. Both codecs were run with

options providing maximum compression (`flac -8` and `mac -c4000`). The results for the ALS encoder were determined for both Rice and BGMC coding of the residual (ALS/R and ALS/B), with random access each 500 ms, and all other options set to maximum compression as well (`mp4als -6 -r5 [-b]`).

The tests were conducted on a 1.2 GHz Pentium III-M PC, with 512 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. Since not all of these formats were supported by FLAC, there are a few empty boxes in our comparison tables.

4.1. Compression Ratio

In the following, the compression ratio is defined as

$$C = \frac{CompressedFileSize}{OriginalFileSize} \cdot 100\%, \qquad (6)$$

where smaller values mean better compression. The results for the examined audio formats are shown in Table 1.

Format	FLAC	MAC	ALS/R	ALS/B
48/16	48.6	45.3	46.5	46.0
48/24	68.4	63.2	64.0	63.6
96/16	36.2	30.9	31.1	30.4
96/24	56.7	48.1	47.1	46.7
192/16	–	22.2	21.9	21.1
192/24	–	39.1	38.2	37.8
Total		**41.3**	**41.1**	**40.6**

Table 1: Comparison of the average compression ratios for different audio formats (sampling frequency in kHz / resolution in bits).

The compression ratios of the MPEG-4 ALS encoder are significantly better than those of FLAC, and comparable to those of Monkey's Audio. Still, ALS delivers better overall compression. Particularly for high-definition material (i.e. 96 kHz / 24-bit and above), the results are clearly superior. The use of BGMC even further improves compression, at the expense of a slightly increased encoder and decoder complexity.

Format	FLAC	MAC	ALS/R	ALS/B
48/16	117.2	61.6	88.6	107.2
48/24	197.3	79.5	120.6	112.2
96/16	226.6	136.9	158.0	206.8
96/24	387.7	163.3	218.4	228.8
192/16	–	97.9	116.6	148.0
192/24	–	108.2	177.4	195.6
Total		**647.4**	**879.6**	**998.6**

Table 2: Compression time (in seconds) measured for different audio formats.

Format	FLAC	MAC	ALS/R	ALS/B
48/16	10.3	68.4	22.6	26.5
48/24	21.4	72.2	23.6	26.6
96/16	26.9	134.2	32.2	45.5
96/24	43.9	144.2	45.7	52.8
192/16	–	104.4	24.8	33.7
192/24	–	113.8	51.6	53.1
Total		**637.2**	**200.5**	**238.2**

Table 3: Decompression time (in seconds) measured for different audio formats.

4.2. Complexity

The total compression and decompression times for the whole set of test material, which equals 450 seconds of audio for each format, are shown in Table 2 and Table 3, respectively. The MPEG-4 ALS encoder, at maximum compression, is somewhat slower than Monkey's Audio, but one should take into account that the tested ALS codec binary was not yet optimized for speed. However, while encoder and decoder complexity of Monkey's Audio are almost equal (symmetric codec), the ALS decoder is nearly three times faster than Monkey's Audio [5].

For example, MPEG-4 ALS is able to decode a 96 kHz / 24-bit stereo signal nearly ten times faster than real-time on the 1.2 GHz Pentium test system. These results show that the coding scheme is capable of processing even multichannel data at a great speed.

Furthermore, the ALS encoder is designed to offer different compression levels. While the *maximum* level (`-6`) achieves the highest compression at the expense of slowest encoding speed, the faster *medium*

level (-3), compared to the results in Table 1, only leads to less than 0.5% degradation [20]. Using the medium level, encoding is typically three times faster than at the maximum level, while decoding is still 20-30% faster [21]. Thus, MPEG-4 ALS can be used even on hardware with low computing power.

4.3. Floating-Point Results

Three types of 32-bit floating-point audio data were artificially generated. The material from the PCM test set was converted into the normalized (±1.0) floating-point format, using different resolutions and additional gain factors:

- Direct conversion (gain factor = 1.0) of 16-bit material, i.e. $2^{15} \rightarrow 1.0$
- Direct conversion (gain factor = 1.0) of 24-bit material, i.e. $2^{23} \rightarrow 1.0$
- Conversion of 24-bit material with a gain factor of 2.99, leading to full 32-bit floating-point resolution.

The results for all three types at different sampling frequencies are shown in Table 4.

Bits / Gain	48 kHz	96 kHz	192 kHz
16 / 1.0	23.3	15.6	11.0
24 / 1.0	48.0	35.3	28.7
24 / 2.99	67.4	54.8	49.1

Table 4: Compression ratios for different types of floating-point material.

As in the case of PCM material, compression improves with rising sampling frequency. For the directly converted material with limited resolution, high compression is achieved in general, since not all bits of the floating-point format are actually used. When the floating-point material has full 32-bit resolution (last row), the compression ratio is limited to approximately 50%. However, general compression schemes such as gzip achieve almost no compression at all for this kind of material [16].

5. APPLICATIONS

Applications for lossless audio coding exist in different areas, including archival systems, studio operations, and file transfer for collaborative working or music distribution over the internet (download services). In general, lossless coding is required wherever audio data is designated to be stored, transmitted, or processed without introducing any coding artifacts - even if they would be imperceptible.

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

6. 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.

MPEG-4 ALS is expected to be an international standard by the end of 2004.

7. 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 N5040, "Call for Proposals on MPEG-4 Lossless Audio Coding," *61st MPEG Meeting, Klagenfurt, Austria*, July 2002.

[3] M. Bosi et al., "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 45, no. 10, October 1997.

[4] ISO/IEC JTC1/SC29/WG11 N5383, "Report on Responses to Call for Proposals for Lossless Audio Coding," *63rd MPEG Meeting, Awaji Island, Japan*, October 2002.

[5] ISO/IEC JTC1/SC29/WG11 N5576, "Analysis of Audio Lossless Coding Performance, Complexity and Architectures," *64th MPEG Meeting, Pattaya, Thailand*, March 2003.

[6] ISO/IEC JTC1/SC29/WG11 N6133, "WD 2 of ISO/IEC 14496-3:2001/AMD 4, Audio Lossless Coding (ALS)," *67th MPEG Meeting, Hawaii, USA*, December 2003.

[7] T. Liebchen, "MPEG-4 Lossless Coding for High-Definition Audio," *115th AES Convention*, 2003.

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

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

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

[11] ISO/IEC JTC1/SC29/WG11 M9893, Y. Reznik, "Proposed Core Experiment for Improving Coding of Prediction Residual in MPEG-4 Lossless Audio RM0," *65th MPEG Meeting, Trondheim, Norway*, July 2003.

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

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

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

[15] D. Yang and T. Moriya, "Lossless Compression for Audio Data in the IEEE Floating-Point Format," *115th AES Convention*, 2003.

[16] D. Yang, T. Moriya, and T. Liebchen, "A Lossless Audio Compression Scheme with Random Access Property," *Proc. IEEE ICASSP*, 2004.

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

[18] "FLAC," *flac.sourceforge.net*.

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

[20] ISO/IEC JTC1/SC29/WG11 M9314, T. Liebchen, "Technology for MPEG-4 Lossless Audio Coding," *63rd MPEG Meeting, Awaji Island, Japan*, October 2002.

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