UNIVERSITY OF JOENSUU
COMPUTER SCIENCE AND STATISTICS
DISSERTATIONS 17

Haibin Huang

Lossless Audio Coding for MPEG-4

Academic dissertation

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism in the Louhela Auditorium of the Science Park, Länsikatu 15, Joensuu, on October 12th 2007, at 12 o'clock.

UNIVERSITY OF JOENSUU
2007

Supervisor Professor Pasi Fränti
Department of Computer Science and Statistics
University of Joensuu
FIN-80101, Joensuu, FINLAND

Reviewers Professor Martti Juhola
Department of Computer Sciences
University of Tampere
FIN-33014, Tampere, FINLAND

Professor Ioan Tabus
Institute of Signal Processing
Tampere University of Technology
FIN-33101, Tampere, FINLAND

Opponent Dr. Adriana Vasilache
Nokia Research Center
FIN-33720, Tampere, FINLAND

ISBN 978-952-219-018-5 (printed)
ISBN 978-952-219-019-2 (PDF)
ISSN 1796-8100 (printed)
ISSN 1796-8119 (PDF)
Computing Reviews (1998) Classification: I.2.7, I.5.1, I.5.4, I.5.3, G.1.6

Joensuun yliopistopaino
Joensuu 2007

Lossless Audio Coding for MPEG-4

Haibin Huang

Department of Computer Science and Statistics
University of Joensuu
P.O.Box 111, FIN-80101 Joensuu, FINLAND
huang@cs.joensuu.fi

University of Joensuu, Computer Science and Statistics, Dissertations 17
Joensuu, 2007, 86 pages

Abstract

Lossless audio compression has been an active research field for the past decade. The main application of the technology is in the archival of digital audio data. With today's wide spread of broadband networks, the technology has also gradually become popular in online music sharing and distribution.

Lossless audio compression can be categorized into two different technical approaches: predictive coding and transform coding. In the first approach, linear prediction is first applied on the audio signals to reduce the inter-sample correlations. The resulted residual signal is subsequently entropy-coded. Because audio signals are generally highly non-stationary in nature, the main challenge in predictive code design is on the prediction part, which should maintain a close tracking the audio signal and generate the smallest possible residual signal. In the second approach, the audio signals are first converted to the frequency domain via an integer-to-integer mapping transform, and the spectral data are subsequently entropy-coded. The challenges here lie in the design of the integer-mapping transform to keep the approximation noise low, as well as efficient coding of the spectral data.

For predictive coding, a cascaded RLS-LMS predictor is proposed. The predictor consists of a cascade of linear predictors with different numbers of taps and tracking speeds. A short predictor with fast tracking speed tracks those fast-changing audio components, e.g., transient signals, whereas long predictors with slower tracking speed but higher tracking accuracy handle the slow-changing audio components such as harmonics. The cascaded predictor maintains a high level of prediction gain for various types of music signals. Since most audio signals come with two

channels, joint-stereo prediction is proposed in the RLS-LMS predictor to exploit signal correlation from both intra-channel and inter-channel, and can increase the lossless compression ratio by several percent.

For transform coding, a Lifting-Matrix based design method is proposed for deriving integer-mapping transforms. The method reduces the number of rounding operations in the transform by an order of magnitude, and significantly improves the approximation accuracy of the integer transform. In bit-plane coding of the spectral data, a number of contextual information such as the frequency band, distance-to-lazy-plane, and significant state, are used in context based arithmetic code to improve the compression ratio.

Keywords: Audio coding, Lossless audio coding, MPEG-4 Audio Lossless Coding, MPEG-4 Scalable Lossless Coding

Acknowledgements

The work presented here was carried out partly at the Institute for Infocomm Research, Singapore, and partly at the Department of Computer Science, University of Joensuu, Finland, during years 2004-2007.

I am very grateful to Professor Susanto Rahardja and Dr. Xiao Lin for guiding the earlier part of my work, and Professor Pasi Fränti for supervising the latter part of the work. I also owe many thanks to Dr. Rongshan Yu and Dr. Dongyan Huang for co-operative works, and my other colleagues in both Singapore and Finland for their helps in my professional and scientific growth.

I am also deeply indebted to my family who has provided me consistent supports during the course of my study, especially my wife Yen Ping who took the majority share of baby-sitting our three-year-old son Qirui.

Singapore, Thursday, 20 September 2007

Haibin Huang

List of original publications

P1. H. Huang, R. Yu, X. Lin, S. Rahardja, Method for Realising Reversible Integer Type-IV Discrete Cosine Transform, *Electronics Letters*, Vol 40, Number 8, pp. 514-515, April 2004.

P2. H. Huang, S. Rahardja, R. Yu, X. Lin, Integer MDCT with Enhanced Approximation of the DCT-IV, *IEEE Trans. on Signal Processing*, Vol 54, Number 3, pp. 1156-1159, March 2006.

P3. H. Huang, X. Lin, S. Rahardja, R. Yu, Integer DFT with High Transform Accuracy, *Proc. 4th Int. Conf. on Information, Communications and Signal Processing* (ICICS 2005), pp .1471-1474, Bangkok, Thailand, December 6-9, 2005.

P4. H. Huang, P. Fränti, D. Huang, S. Rahardja, Cascaded RLS-LMS Prediction in MPEG-4 Lossless Audio Coding, *IEEE Trans. on Audio, Speech and Language Processing*, accepted for publication.

P5. R. Yu, X. Lin, S. Rahardja, C. C. Ko, H. Huang, Improving Coding Efficiency for MPEG-4 Audio Scalable Lossless Coding, *Proc. 2005 Int. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP 2005), pp. III169-III172, Philadelphia, USA, March 18-23, 2005.

Contents

Chapter 1

Introduction

Audio compression is a branch of *data compression* that focuses on reducing the size of audio files. Audio data usually have large file sizes. For example, a 4-minute long music piece takes about 42 mega bytes (MB) on a compact disk (CD). Directly sending uncompressed raw audio data over networks takes long transmission times. In the case of portable music player such as iPod, the large file size of uncompressed audio also limits the total number of songs that can be stored in that player.

There are two types of audio compression technologies: *lossy audio compression* [7, 63] and *lossless audio compression* [29]. The former compresses audio data by removing information that is perceptually unimportant to the human auditorial system. The discarded information is perpetually lost and can not be recovered in decoding. Examples of lossy audio compression technologies include *MPEG-1 Layer-3 (MP3)*, *MPEG-2/4 Advanced Audio Coding (AAC)* [41], and *Dolby AC-3*. They can generally achieve over 10 times file size reduction while still maintaining sound '*transparency*', i.e., the decoded audio sounds indistinguishable to the original input audio.

Lossless audio compression removes only redundant information in the audio data. The original waveform can be fully recovered in the decoding process. Lossless audio compression generally reduces the file size by a factor of 2. This difference in compression ratios between lossless and lossy audio compressions can be explained this way - both approaches try to preserve a copy of the original audio data, the lossless approach keeps a bit-to-bit exact copy of the original waveform whereas the lossy approach only keeps a '*perceptually-equivalent*' copy of the original - the compressed file sounds indistinguishable to the original during playback. This relaxed condition gives the lossy approach more freedom in trimming information in the audio data and therefore higher compression ratio than the lossless approach.

In the digital audio market, the majority of audio contents are encoded in lossy formats such as MP3 and AAC. For example, Apple's iTunes Online Music Store has already sold over 3 billions AAC-encoded songs. Because of small file sizes and good sound quality, lossy-coded digital audio contents dominate in the mass consumer market. Applications of lossy audio compression include music distribution over internet as well as on-the-go entertainment through portable music players. Lossless audio compression, however, maintains a niche, archival-oriented market. Audio engineers use lossless compression during audio data processing to achieve high-fidelity (Hi-Fi) with lessened storage requirements. Audiophiles who have little patience in tolerating any loss in the sound quality use lossless compression to archive and playback CD collections. Trans-coding between lossy formats generally degrades the sound quality. A master copy of the original audio data can be first created using lossless compression, and then rendered into various lossy formats.

Table 1.1: List of lossless audio compressors

Codec Name	**Compression Ratio**	**Encoding Speed**	**Decoding Speed**	**Category**
Apple Lossless [3]	low	average	fast	proprietary
FLAC [11]	low	fast	fast	open source
LA [6]	high	slow	slow	proprietary
Monkey's [4]	average	average	average	open source
MPEG-4 ALS [60]	high	average	average	standard
MPEG-4 SLS [60]	average	average	average	standard
OptimFROG [18]	high	slow	slow	proprietary
Real Lossless [72]	average	average	fast	proprietary
Shorten [76]	low	fast	fast	open source
TTA [81]	average	fast	fast	open source
WavPack [8]	low	fast	fast	open source
WMA Lossless [57]	average	average	average	proprietary

The field of lossless audio compression has grown rapidly during the last decade. Only a few lossless audio compressors existed in the mid 90s [12, 13, 75]. Now there are already over 30 varieties downloadable from internet. This fast growth in the number of lossless compressors is due to the fact that lossless audio compression has become useful and affordable to general users as a result of high broadband penetration, powerful CPUs and low storage costs. Table 1.1 lists a number of state-of-the-art lossless audio codecs. In Table 1.1, the results are referenced from [40], which used the default settings of the lossless compressors in the measurements. In Table 1.1, Lossless Audio (LA) [6] and OptimFROG [18] provide best compression results. Free Lossless Audio Codec (FLAC) [11] and Monkey's Audio [4] are popu-

lar formats used in internet music sharing. Online music download service providers also have proprietary lossless codecs, e.g., Apple Computer (Apple Lossless [3]), Microsoft (WMA-Lossless [57]), and Real Networks (Real-Lossless [72]). The MPEG-4 ALS [60] and SLS [60] are standardized codecs and the latest to join the family. Comprehensive comparisons of these codecs can be found in the following websites: HydrogenAudio [40], Speek's site [78], Hans Heiden's site [30], and Synthetic Soul [79].

Lossless audio compressors can be categorized into: *predictive coders* and *transform coders*. These two coder types are described below in Section 1.1 and Section 1.2, respectively.

1.1 Predictive Coder

Digital audio signals usually exhibit high level of correlations among neighboring samples. A typical audio waveform is shown in Figure 1.1. Predictive coders exploit these correlations by means of *linear prediction*. A general diagram of predictive coders is shown in Figure 1.2.

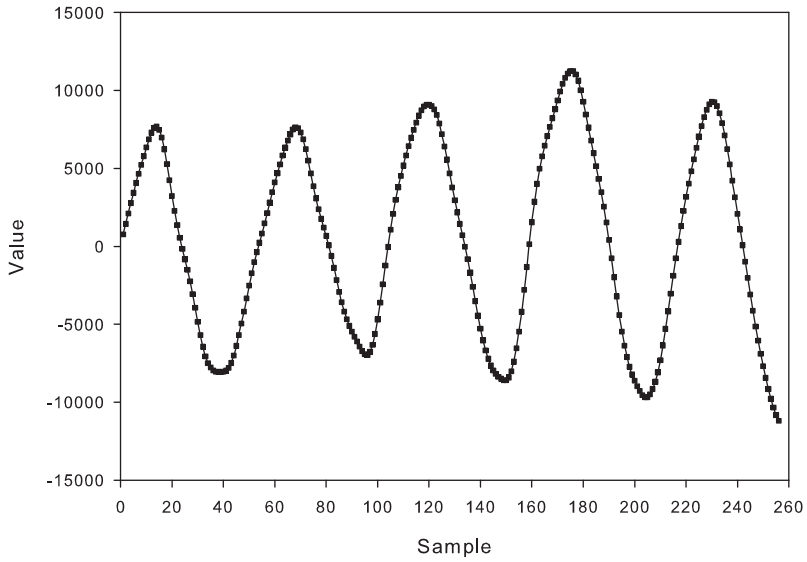

Figure 1.1: Exemplary digital audio waveform

In Figure 1.2, at the encoder side, a predictor generates an estimate for every input audio sample. The estimate is subtracted from the input sample to generate a residual signal. The residual signal generally has smaller amplitudes

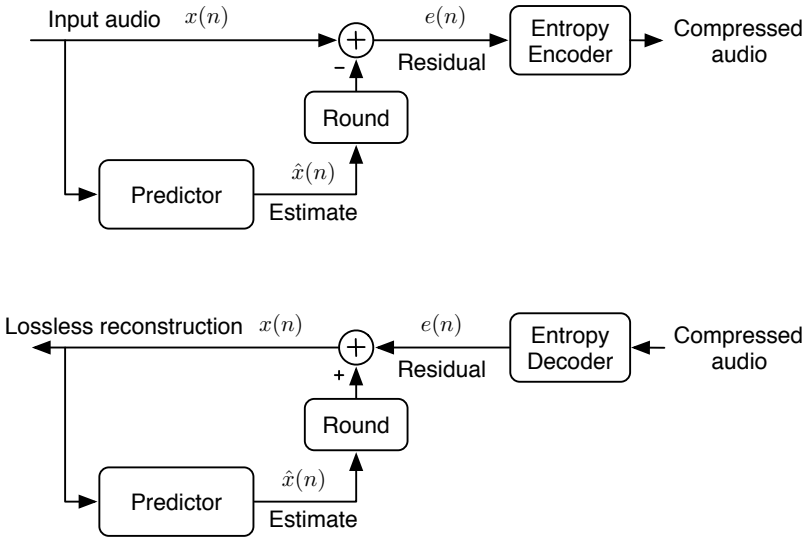

Figure 1.2: Block diagram of predictive coder: encoder (top) and decoder (bottom)

than the original input. The residual signal appears noise-like, and generally shows non-uniform probability distributions. For example, after *Linear Predictive Coding (LPC)* prediction, the residual distribution has been found to be well approximated by *Laplacian distributions* [90]. An entropy coder further compresses the residual signal to generate code tables and indexes that are transmitted to the decoder. At the decoder side, a reverse process takes place where original audio samples are re-constructed by adding the residual signal from the entropy decoder with the estimate signal generated by a predictor identical to that in the encoder. For predictive coders, research efforts have been focused on designing better predictors [2, 9, 12, 13, 19, 20, 21, 33, 36, 44, 49, 50, 53, 54, 77, 85, 89] as well as improving the entropy coding efficiency [1, 27, 28, 52, 69].

1.2 Transform Coder

Transforms like the Discrete Cosine Transform (DCT) [71] have a so-called '*energy-compacting*' capability. Audio signals usually contain many harmonic components. Transforming such signals to the frequency-domain gives a representation where energies of the audio signals are packed to a few large spectral peaks, and the rest part of the spectrum is made of small components. As an example, the DCT transform of the audio waveform in Figure 1.1 is shown in Figure 1.3. The frequency-domain representation of audio signals has more concentrated probability density

distribution than time-domain audio samples, and can therefore be more efficiently compressed by entropy coders. Transform coders first convert time-domain audio samples into the frequency-domain, and then entropy-code the resulted *transform coefficients*. Figure 1.4 shows a diagram of a typical transform coder.

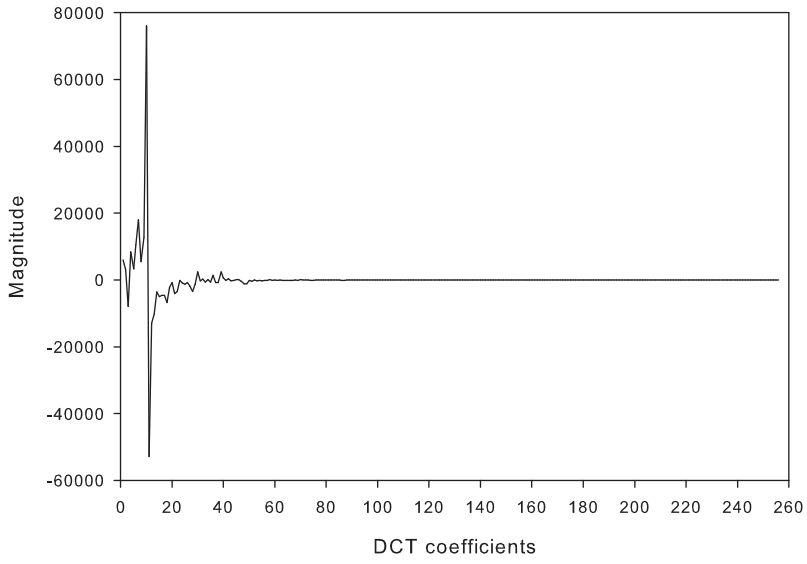

Figure 1.3: DCT transform of the example waveform

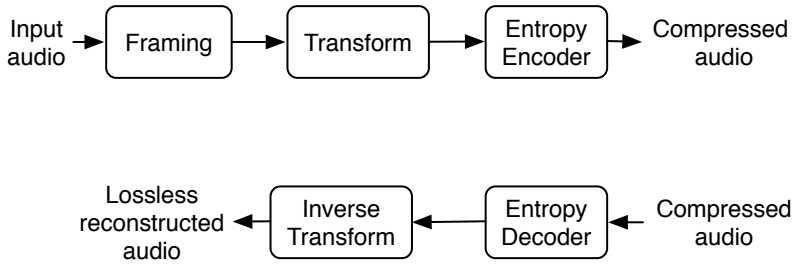

Figure 1.4: Block diagram of transform coder: encoder (top) and decoder (bottom)

Transform coders employ a special type of transform called *integer transform*. An integer transform maps integer-valued input signals to integer-valued output signals. This integer-to-integer mapping property is important for lossless compression as no floating-point output values are generated by the transform that would require more bits to store than integer values. Integer transforms are also reversible, i.e.,

there exist corresponding inverse transforms that perfectly reconstruct the original input signal from the transform coefficients. In contrast, floating-point transforms have hardware-dependent numerical accuracy. It is difficult to make them perfectly reversible across different hardware platforms. For example, audio samples transformed by a floating-point transform on a 32-bit processor can not be guaranteed to be fully restored by an inverse transform running on a 16-bit processor. Therefore, floating-point transforms are not suitable for lossless compression.

In Figure 1.4, an entropy coder processes the transform coefficients either by directly coding the coefficients one after another, or alternatively by employing bit-plane codings. In the latter, the transform coefficients are viewed as layers of bit-planes. The *Most Significant Bit (MSB)* bits of the coefficients are defined as the first bit-plane, followed by the second bit-plane consisting of the second MSB bits, and so on, until the *Least Significant Bit (LSB)* plane is reached. The entropy coder scan through one bit-plane after another in order of decreasing bit significance. Research works on transform coders are focused on finding suitable integer transforms [10, 15, 16, 17, 23, 24, 25, 26, 31, 34, 35, 37, 38, 39, 45, 55, 56, 58, 62, 65, 66, 68, 80, 82, 87, 96] and designing efficient bit-plane coding techniques [46, 47, 48, 64, 88, 90, 93, 94, 95].

1.3 Comparison of Technologies

Both predictive coders and transform coders can reduce the size of the audio file to about half that of the original. Predictive coders generally achieve slightly higher compression than transform coders. They are also less complex for implementation. Predictive coders satisfy the needs of ordinary users in applications such as CD archival and online music sharing, and constitute the majority of today's lossless audio coders. On the other hand, transform coders have a unique feature. Since modern perceptual audio coders, such as MP3 and AAC, also process audio signals in the frequency domain, it is possible to embed a perceptual coder within a transform coder. The resulted *hybrid lossy/lossless* audio coder can generate two outputs at the same time: a small lossy-coded file and a '*lossless correction*' file. The small lossy-coded file is readily decodable by prevailing lossy audio decoders, while together with the lossless correction file, the original audio data can be perfectly reconstructed.

The hybrid lossy/lossless audio approach has two distinctive advantages. It enables a new business model for online music distribution, where a customer can freely download and pre-listen to a lossy-coded music that is full-length but with limited-quality, and depending on whether that piece of music is desired, the customer can purchase a correction file that upgrades the preview version of the music to the full-quality, lossless version. The second advantage is that the hybrid lossy/lossless audio coder is compatible to lossy coders, i.e., the bit-stream generated by the hybrid

coder can be partially decoded by lossy coders. For example, audio players supporting only lossy audio can still decode the lossy-coded file by a hybrid coder, but not the correction file. Compatibility is an important issue for new technologies to enter existing markets, as it is generally preferred that the new technology will gradually replace old ones instead of abruptly obsolete currently-deployed equipments.

Transform coders can employ bit-plane codings to code the transform coefficients. The coding process starts from the MSB bit-plane and progressively proceeds to the LSB bit-plane. An inherent characteristics of bit-plane codings is the *scalability* of the encoded bit-stream. The bit-stream can be truncated at any point with the remaining retained part still decodable. Bit-stream scalability is useful in broadcasting applications where the server monitors available channel bandwidth and dynamically scales the bit-stream for the channel.

1.4 Motivation of Thesis

Transform coders use integer transforms to convert audio samples into the frequency domain for processing. Integer transforms are prototyped from common transforms. For example, the Modified Discrete Cosine Transform (MDCT) [67] is widely used in audio coding. An integer transform prototyped from the MDCT is called *Integer MDCT (IntMDCT)*. The integer-valued outputs of an integer transform should closely approximate those of its prototype transform. The design goal for integer transforms is to make the *approximation error*, defined as the difference between the output values of an integer transform and those of its prototype transform, as small as possible.

For designing integer transforms, Daubechies and Sweldens developed a method [14] by first decomposing a prototype transform into a series of *Givens rotations*, and then replacing each Givens rotation with three so-called *lifting steps*. Each lifting step performs a rounding operation which generates a round-off error. The round-off errors generated in the lifting steps will propagate through the computational steps within the integer transform, and appear at the integer transform output in the form of approximation errors. In the present method, the total number of rounding operations is $O(N\log_2 N)$ where N is the block length of the transform. In this method, when the block length of the transform increases, the approximation error is found to increase too. In the case of IntMDCT in audio coding, when the transform size is 1024, the root-mean-squared value of the approximation error is about 2 bits, which can lead to audible artifacts in the coded audio signal.

Inspired by a research work in the area of image compression [82], *matrix-based multi-dimensional lifting* was proposed in [**P1**, **P2**, **P3**] which extended the concept of lifting steps from scalar space to vector space. Integer transforms developed using

matrix-based multi-dimensional liftings requires only $O(N)$ number of rounding operations for transforming one block of N audio samples. In the case of IntMDCT, the root-mean-square value of the approximation error has been reduced to a fixed value of 0.3 bit, regardless of what transform block length is used. Applied to lossless audio compression, the improved IntMDCT brings a 2% increment in the compression ratio, and also causes no audible artifacts in the coded audio signal.

Predictive coders use predictors to generate a residual signal from the input audio samples, and then code the residual with entropy coding. How efficiently a predictor performs is determined by the *prediction gain*, defined as the ratio between the power of the original input signal and that of the residual. Present technologies for linear prediction include LPC prediction [70] and *adaptive prediction* [32]. In the LPC prediction, for each block of audio samples, an *LPC analysis* [70] is first run to determine the set of predictor coefficients that gives the highest prediction gain for that block. These coefficients are quantized and used by the predictor at the encoder side to generate the residual signal. As the decoder also requires these coefficients to run the predictor at its side, these coefficients must be previously coded and transmitted to the decoder. For stationary signals, the LPC predictor gives the highest prediction gain and is the optimal predictor.

For non-stationary signals such as audio, the input samples are first divided into blocks. With the assumption that the signals within each block are near-stationary, LPC analysis is run for each block of samples individually. However, a compromise has to be made on the choice of the block length. A too large block length will violate the stationarity assumption and reduces the prediction gain, while a too small one will increase the overheads for transmitting the coefficients.

In contrast, adaptive predictors dynamically track audio signals by adaptively updating the predictor coefficients. For audio signals, adaptive predictors generally produce higher prediction gains than LPC predictors. It is also not necessary to transmit the predictor coefficients, as the predictor at the decoder side runs exactly in the same manner as that in the encoder. Because the predictor at the decoder side needs to update the coefficients consistently, decoders employing adaptive predictors generally have higher computational complexities than those using LPC predictors.

Cascaded prediction is widely used by adaptive predictors, where a number of predictors are connected in cascade with the residual signal from one predictor passed to the next one as input. For audio signals, cascaded predictors generally provide higher prediction gains than a single predictor. This is due to the fact that audio signals generally contain both fast and slow varying components. It is hard for a single predictor to track the fast and slow varying components simultaneously, while not so for cascaded predictors as the predictors can be individually tuned to the these components.

In cascaded prediction, the residual signal of the last predictor in the cascade

is output as the 'final' residual that is sent to the entropy coder. This poses a potential problem. If one predictor in the cascade somehow fails to track the signal, the residual signal it produces may become corrupted and contain very large samples. When such a residual signal is passed to the next predictor, a chain-reaction may occur that all the subsequent predictors start to 'run crazy' and the final residual signal becomes unusable. A solution to this problem was given in [77] where a linear combiner was added to the cascaded predictor. Instead of using the output residual of the last predictor as the final residual, the linear combiner first generates a 'final' estimate signal by summing up the estimate signal from each predictor in the cascade, and then subtracts this final estimate from the original input sample to generate the final residual. If a predictor in the cascade stops to work properly, the linear combiner will simply turn off the inputs from this predictor and all those following it, and only use the estimate signals before this predictor to generate the final estimate signal.

In [**P4**], the cascaded predictor of [77] is improved in several aspects. In [77], the cascaded predictor consists of three predictors whose orders are 512, 128, and 16 respectively. For audio signals that contain strong fast-varying components, the performance of the cascaded predictor degrades due to the slow tracking speed of the first predictor that has a very long order. In [**P4**], a short predictor is inserted at the beginning of the cascade, and fast-tracking adaptive algorithms such as the *Recursive Least Squares (RLS)* [32] algorithm is used to update the coefficients of the short predictor. This short predictor significantly reduces the fast-varying components in audio signals. Its output residual can then be better handled by the rest of the predictors in the cascade. Furthermore, without any loss in performance, [**P4**] uses a low-complexity adaptive algorithm, called *Sign-Sign Least Mean Square (SS-LMS)* algorithm [32], to replace the original method in [77] for predictor coefficients updating.

For audio signals, the correlations among samples in the same channel are usually referred to as *intra-channel correlations*. On the other hand, for stereo audio signals, there are also correlations between samples in the two channels. Such correlations are called *inter-channel correlations*. The predictor in [77] considered only intra-channel correlations. In [**P4**], *joint-stereo prediction* is developed which uses audio samples from both channels to estimate the signal in each individual channel. It is shown that joint-stereo prediction can improve the compression ratio by 3% compared with compressing each channel individually.

1.5 MPEG Standardization

In July 2002, MPEG issued a Call for Proposal [42] to initiate the submission of technologies for lossless audio compression. By December 2002, seven submissions were received from Fraunhofer IIS (FhG), Institute for Infocomm Research (I2R), Microsoft, Nippon Telecom (NTT), Real Networks, Samsung, and Technical University Berlin (TUB), respectively. These submissions covered two different technical approaches: a lossless-only predictive coder, and a scalable transform coder that uses a hybrid lossy/lossless structure. MPEG decided to support both approaches and selected for each approach a reference model (RM). The lossless only RM is a combination of submissions by TUB, NTT, and Real Networks, whereas the scalable RM is a combination of technologies by FhG and I2R [91, 92]. In the next three year, these RMs were consistently improved in efficiency, complexity and flexibility. In 2006, two international standards were published based on the RMs, which are: *ISO/IEC 14496-3:2005/Amd 2:2006 Audio Lossless Coding (ALS), new audio profiles and BSAC extensions*, and *ISO/IEC 14496-3:2005/Amd 3:2006 Scalable Lossless Coding (SLS)*. In the SLS standard, [**P1**, **P2**, **P3**] contributed to the IntMDCT transform, and [**P5**] contributed to the entropy coding. In the ALS standard, [**P4**] contributed to the prediction.

1.6 Organization of Thesis

The rest of the thesis is organized as follows: the next chapter introduces the MPEG-4 ALS, which is followed by a description of the MPEG-4 SLS. Chapter 4 summarizes the publications, and the performance of the lossless audio compressors is compared in Chapter 5. A conclusion of the thesis is drawn in Chapter 6, which is followed by the original publications.

Chapter 2

MPEG-4 Audio Lossless Coding

MPEG-4 Audio Lossless Coding (ALS) is a state-of-the-art technology for lossless audio compression [54, 61]. MPEG-4 ALS is a predictive coding technology whose structure is given in Figure 2.1. The input audio data is partitioned into blocks. For each block, a prediction residual is calculated using *short-term prediction* and then *long-term prediction*. After that, the prediction residual is entropy-coded.

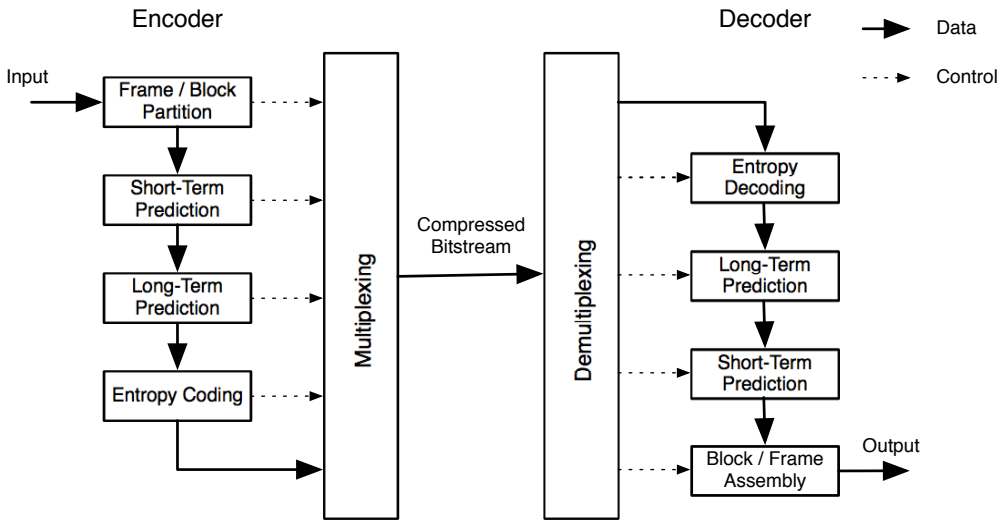

Figure 2.1: Block diagram of ALS encoder and decoder

2.1 Prediction

MPEG-4 ALS uses both short-term prediction and long-term prediction. The short-term prediction exploits correlations among neighboring audio samples. Since audio signals commonly contain repetitions of waveform segments at certain intervals, long term prediction is used to predict the repetition of a waveform segment from its previous occurrences. Three types of predictors are used in ALS: *LPC* [51], *RLS-LMS* [**P4**], and *LTP* [59], where the first two are short-term predictors, and the last one is a long-term predictor.

2.1.1 LPC Prediction

The LPC predictor is shown in Figure 2.2. For each block of input samples, a set of *partial correlation (parcor) coefficients* is first calculated using the *Levinson-Durbin algorithm* [70]. The set of parcor coefficients is optimal in the sense that the resulted residual signal has the minimal variance. The parcor coefficients are quantized and converted to LPC coefficients, which are used by a predictor to generate the residual signal. The quantization to the parcor coefficients is necessary as these coefficients also need to be entropy-coded and transmitted to the decoder, which is shown in Figure 2.3. In the decoder, the prediction residual and the parcor coefficients are first entropy decoded. The parcor values are converted to LPC coefficients in the same manner as that in the encoder. The original audio samples are regenerated by adding the residual to the estimate signal from the predictor.

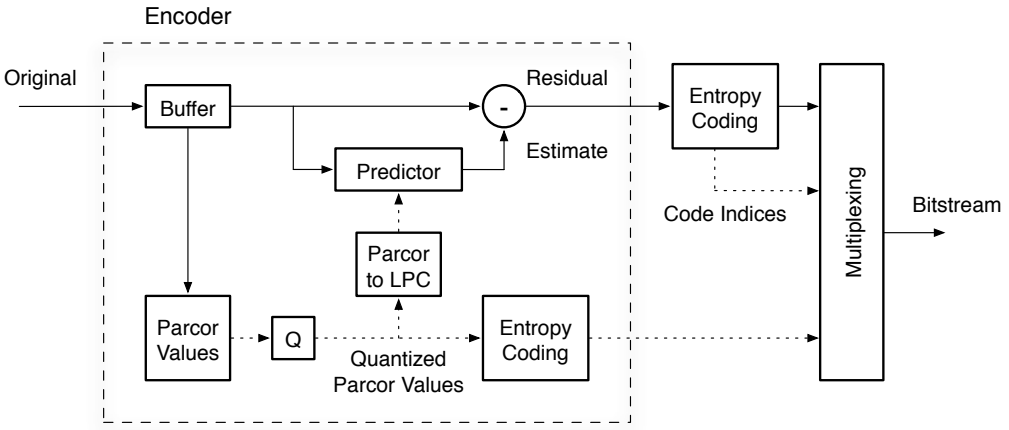

Figure 2.2: Block diagram of LPC predictor [51] (encoder)

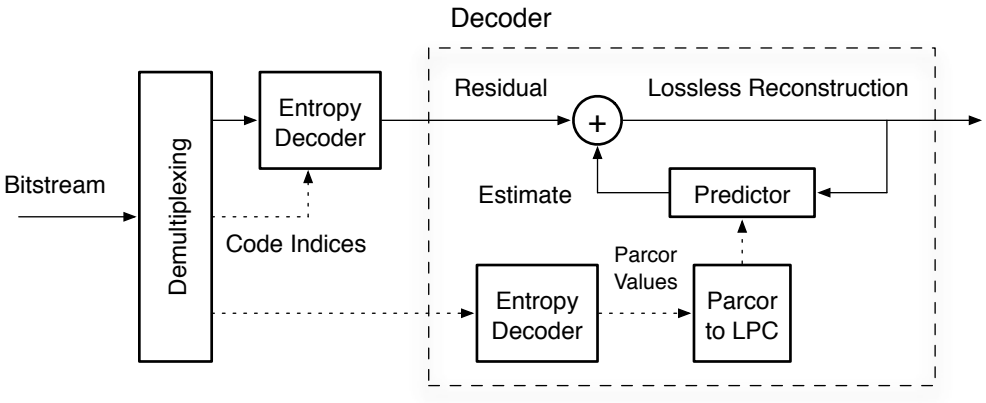

Figure 2.3: Block diagram of LPC predictor [51] (decoder)

2.1.2 RLS-LMS Prediction

The RLS-LMS predictor is an adaptive predictor using the cascaded prediction structure. The block diagram of the RLS-LMS predictor is shown in Figure 2.4. The predictor consists of a cascade of simple prediction stages in the sequence of: a *Differential Pulse Coded Modulation (DPCM)* predictor, a *Recursive Least Square (RLS)* predictor, and a series of *Least Mean Square (LMS)* predictors. The input samples pass through the prediction stages sequentially, with the residual of one stage serving as the input to the next stage. The estimates from each prediction stage are summed up by a *linear combiner* to generate the estimate of the current input sample.

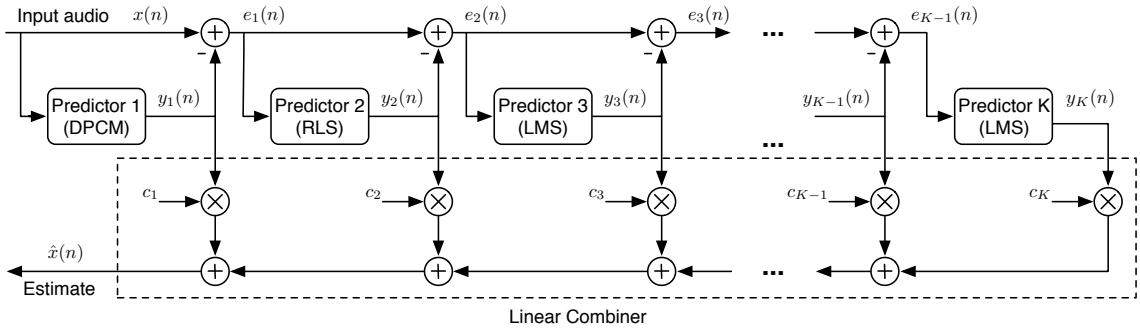

Figure 2.4: Structure of the RLS-LMS predictor

DPCM Predictor

The first predictor stage in the cascade is the DPCM predictor, which is a simple first-order predictor with the coefficient set to 1, i.e., the previous input sample is used as the estimate of the current input sample. The DPCM predictor is given by

$$\begin{aligned} y_1(n) &= x(n-1), \\ e_1(n) &= x(n) - y_1(n), \end{aligned}$$

where $y_1(n)$ is the estimate of the first prediction stage, $e_1(n)$ is the prediction residual, and $x(n-1)$ is the previous input sample. The DPCM predictor removes the DC components in the audio signals, which, if not removed, will deviate the following adaptive prediction stages from converging to the optimal functioning point.

RLS Predictor

The RLS predictor is the second predictor in the cascade. The RLS algorithm [32] is used to adapt the predictor coefficients. The algorithm is initialized by setting the inverse auto-correlation matrix $\mathbf{P}$ as follows

$$\mathbf{P}(0) = \delta \mathbf{I},$$

where δ is a small positive number, $\mathbf{I}$ is an $M_2 \times M_2$ identity matrix, and M_2 is the order of the RLS predictor. The predictor coefficient vector $\mathbf{a}_2(n)$, defined as

$$\mathbf{a}_2(n) = [a_{2,1}(n), a_{2,2}(n), \ldots, a_{2,M_2}(n)]^T,$$

is initialized by

$$\mathbf{a}_2(0) = \mathbf{0}.$$

Here, the symbol T denotes the operation of vector transpose.

Define

$$\mathbf{e}_1(n) = [e_1(n-1), e_1(n-2), \ldots, e_1(n-M_2)]^T$$

as the RLS predictor input vector, for each instance of time, $n = 1, 2, \ldots$, the following calculations are made

$$\begin{aligned} \mathbf{v}(n) &= \mathbf{P}(n-1)\mathbf{e}_1(n), \\ m &= \begin{cases} \frac{1}{\mathbf{e}_1^T(n)\mathbf{v}(n)} & \text{if } \mathbf{e}_1^T(n)\mathbf{v}(n) \neq 0 \\ 1 & \text{else,} \end{cases} \\ \mathbf{k}(n) &= m\mathbf{v}(n), \end{aligned}$$

$$
\begin{aligned}
y_2(n) &= \mathbf{a}_2^T(n-1)\mathbf{e}_1(n), \\
e_2(n) &= e_1(n) - y_2(n), \\
\mathbf{a}_2(n) &= \mathbf{a}_2(n-1) + \mathbf{k}(n)e_2(n), \\
\mathbf{P}(n) &= \mathrm{Tri}\{\lambda^{-1}(\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{v}^T(n))\}.
\end{aligned}
$$

The forgetting factor, λ, is a positive value slightly smaller than 1. Tri{.} denotes the operation of first computing the lower triangular part of $\mathbf{P}(n)$, and then copying the values in the lower triangular to the upper triangular according to

$$p_{i,j} = p_{j,i},$$

where $p_{i,j}$ is the element of matrix $\mathbf{P}(n)$ at the i^{th} row and the j^{th} column. The RLS predictor has a fast tracking speed, and is used to remove the fast changing components in audio signals.

LMS Predictor

The RLS-LMS predictor has a series of LMS prediction stages. The *Normalized Least Mean Square (NLMS)* algorithm [77] is used to adapt the coefficients of the LMS predictors. For the LMS predictor in the k^{th} stage, the coefficient vector

$$\mathbf{a}_k(n) = [a_{k,1}(n), a_{k,2}(n), \ldots, a_{k,M_k}(n)]^T$$

is initialized by

$$\mathbf{a}_k(0) = \mathbf{0},$$

where M_k is the order of the predictor.

Define

$$\mathbf{e}_{k-1}(n) = [e_{k-1}(n-1), e_{k-1}(n-2), \ldots, e_{k-1}(n-M_k)]^T$$

as the input vector to the k^{th} prediction stage, for each instance of time, $n = 1, 2, \ldots$, the following calculations are made

$$
\begin{aligned}
y_k(n) &= \mathbf{a}_k^T(n-1)\mathbf{e}_{k-1}(n), \\
e_k(n) &= e_{k-1}(n) - y_k(n), \\
\mathbf{a}_k(n) &= \mathbf{a}_k(n-1) + \frac{e_k(n)\mathbf{e}_{k-1}(n)}{1 + \mu_k \mathbf{e}_{k-1}^T(n)\mathbf{e}_{k-1}(n)}
\end{aligned}
$$

where μ_k is the stepsize of the NLMS algorithm. Since the fast-changing components have already been removed from the audio signals by the RLS predictor, the LMS predictors then remove the remaining slow-changing components in the audio signals.

Linear Combiner

The linear combiner multiplies a set of coefficients to the estimates from the DPCM, RLS, and LMS prediction stages. The results are summed up together to provide the estimate of the current input sample. The Sign-Sign LMS algorithm is used to adapt the coefficients of the linear combiner.

The coefficient vector is defined as

$$\mathbf{c}(n) = [c_1(n), c_2(n), \ldots, c_K(n)]^T,$$

where K is the number of prediction stages in the cascade. The input vector is given by

$$\mathbf{y}(n) = [y_1(n), y_2(n), \ldots, y_K(n)]^T.$$

The estimate of the RLS-LMS predictor is calculated as

$$\hat{x}(n) = \mathbf{c}^T(n)\mathbf{y}(n).$$

The linear combiner coefficients are updated according to

$$\mathbf{c}(n+1) = \mathbf{c}(n) + \alpha \,\mathrm{sgn}[\mathbf{y}(n)]\,\mathrm{sgn}[x(n) - \hat{x}(n)],$$

where the sgn function is defined as

$$\mathrm{sgn}[r] = \begin{cases} 1 & r > 0 \\ 0 & r = 0 \\ -1 & r < 0 \end{cases}$$

and α is the stepsize of the Sign-Sign LMS algorithm.

Joint-Stereo Prediction

For mono audio signals, there are intra-channel correlations among samples in the channel. In the case of stereo audio signals, there are also inter-channel correlations between samples in different channels. Both intra-channel and inter-channel correlations are exploited by the RLS-LMS predictor through *joint-stereo prediction*, where past samples from both L and R audio channels are used in estimating the current sample of each channel. This joint-stereo prediction is implemented in the second prediction stage as illustrated in Figure 2.5.

In Figure 2.5, the intra-channel predictor $\mathbf{a}_L$ generates an estimate for the L channel by using L channel samples. At the same time, the inter-channel predictor $\mathbf{b}_L$ generates another estimate by using samples in the R channel. The two estimates

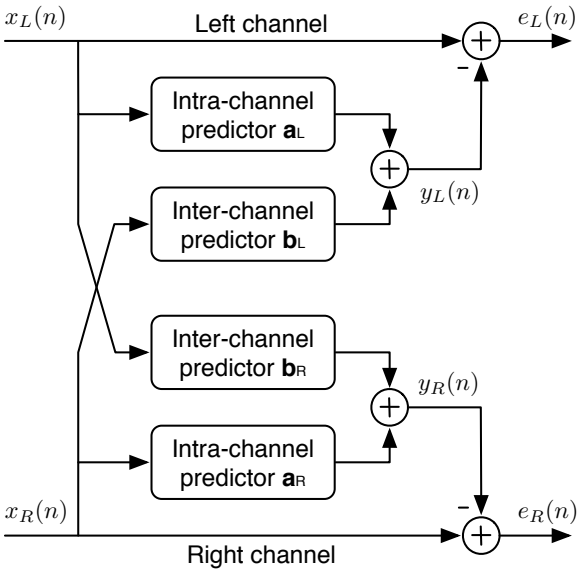

Figure 2.5: Joint-stereo prediction

are added together to give the output estimate of the RLS prediction stage for the L channel. Let M_a and M_b be the orders of the intra-channel predictor $\mathbf{a}_L$ and inter-channel predictor $\mathbf{b}_L$, respectively, and the L channel estimate is given by

$$y_L(n) = \sum_{m=1}^{M_a} a_{L,m} x_L(n-m) + \sum_{m=1}^{M_b} b_{L,m} x_R(n-m),$$

where $a_{L,m}$ and $b_{L,m}$ are coefficients of predictor $\mathbf{a}_L$ and predictor $\mathbf{b}_L$, respectively. $x_L(n)$ and $x_R(n)$ are input samples in the L and R channels.

Similarly, the R channel estimate is given by

$$y_R(n) = \sum_{m=1}^{M_a} a_{R,m} x_R(n-m) + \sum_{m=0}^{M_b-1} b_{R,m} x_L(n-m),$$

where $a_{R,m}$ and $b_{R,m}$ are coefficients of the intra-channel predictor $\mathbf{a}_R$ and the inter-channel predictor $\mathbf{b}_R$, respectively.

In joint-stereo prediction, the coefficients of the intra- and inter-channel predictors are updated using the RLS algorithm. For the L channel, the input vector and coefficient vector are given by

$$\begin{aligned} \mathbf{x}_L(n) &= [x_L(n-1), \ldots, x_L(n-M_a), x_R(n-1), \ldots, x_R(n-M_b)]^T, \\ \mathbf{w}_L(n) &= [a_{L,1}(n), \ldots, a_{L,M_a}(n), b_{L,1}(n), \ldots, b_{L,M_b}(n)]^T, \end{aligned}$$

respectively. For the R channel, the input vector and coefficient vector are given by

$$\begin{aligned}\mathbf{x}_R(n) &= [x_R(n-1),\ldots,x_R(n-M_a),x_L(n),\ldots,x_L(n-M_b+1)]^T,\\ \mathbf{w}_R(n) &= [a_{R,1}(n),\ldots,a_{R,M_a}(n),b_{R,1}(n),\ldots,b_{R,M_b}(n)]^T,\end{aligned}$$

respectively. It is shown in [**P4**] that joint-stereo prediction can improve the lossless compression ratio by 3% compared with individually compressing each audio channel.

2.1.3 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. Long-term prediction captures these periodic components and further reduces the prediction residual after short-term prediction. This is depicted as follows:

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

where $\tilde{e}(n)$ and $e(n)$ are the prediction residuals after and before long-term prediction, respectively. The summation term in the equation gives the value of long-term prediction, in which τ denotes the sample lag and γ denotes the quantized gain value. The values of τ and γ are chosen in such a way that the variance of the residual signal is minimized. These values are transmitted as side information to the decoder.

At the decoder, the reverse process is carried out using the following equation:

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

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

2.2 Entropy Coding

In ALS, linear prediction is performed on the input audio samples to generate a residual signal, which has a smaller dynamic range than the input signal. The distribution of the residual signal can be closely modeled by a Laplacian (or two-sided geometric) distribution. The residual signal is entropy-coded with the Rice

code [74]. Alternatively, the residual can also be coded by a more complex and efficient coding scheme called the block Gilbert-Moore code (BGMC) [22, 73]. In BGMC, the residual distribution is further partitioned into three parts: a central region, flanked by two tail regions as shown in Figure 2.6. The residuals in the tail regions are simply re-centered and coded with Rice codes. Residuals within the central region are further split into those that belong to the Least Significant Bit (LSB) and the Most Significant Bit (MSB) parts. The LSB parts are directly transmitted using fixed-length codes without any processing, while the MSB parts are coded with the more efficient block Gilbert-Moore arithmetic code [73].

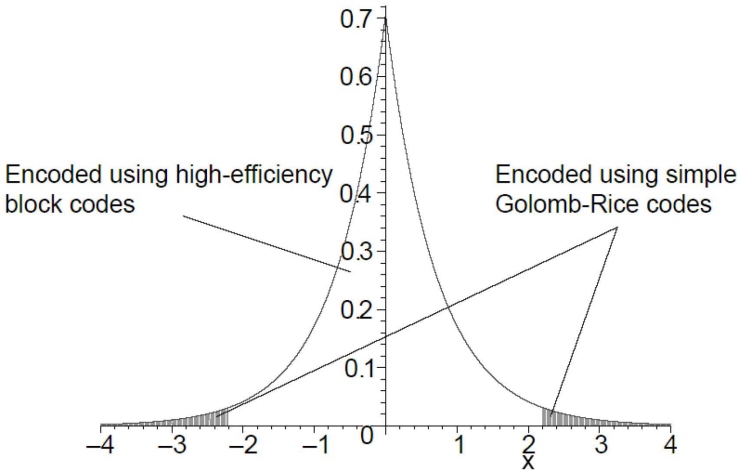

Figure 2.6: Partition of the residual distribution [73]

2.3 Summary

This chapter gives an introduction of the MPEG-4 ALS technology. Both short-term prediction and long-term prediction are utilized in ALS, where the former exploits the correlations among neighboring audio samples, and the latter captures the periodic components in audio signals, in order to reduce the prediction residual. In short-term prediction, there are two choices of predictors: the LPC predictor and the RLS-LMS predictor. The LPC predictor calculates the predictor coefficients using the Levinson-Durbin algorithm and transmits those coefficients to the decoder. Its decoder has low computational complexity because calculation of predictor coefficients is not required. On the other hand, the RLS-LMS predictor uses cascaded adaptive predictors to track audio signals, which are non-stationary. A smaller residual signal is achieved, but since the decoder also needs to adaptively update

the predictor coefficients, the decoding complexity is higher than the LPC predictor. The residual signal is finally coded using either the Rice code, or the more complex but efficient Gilbert-Moore arithmetic code.

Chapter 3

MPEG-4 Scalable Lossless Coding

MPEG-4 Scalable Lossless Coding (SLS) is a transform coding technology for lossless audio compression [43, 95]. It uses a hybrid lossy/lossless coding structure as shown in Figure 3.1. Input PCM audio samples are first transformed by a reversible, integer-to-integer transform called Integer Modified Discrete Cosine Transform (IntMDCT) into IntMDCT coefficients. These coefficients are passed to a two-layered structure consisting of a core layer and an enhancement layer. The core layer uses lossy audio coders such as the MPEG-4 AAC to generate a lossy-coded bit-stream. In the AAC coder, the IntMDCT coefficients are quantized at step-sizes controlled by a psychoacoustic model so that the quantization noise is best masked by the human auditory system. The quantized coefficients are subsequently entropy-coded to generate the lossy bit-stream. In the enhancement layer, it is not necessary to code again the part of information in the IntMDCT coefficients that has already been coded by the core layer. An error mapping process thus removes this part of information from the IntMDCT coefficients, and generates a residual signal. The residual signal is subsequently entropy-coded using bit-plane codings to generate a scalable bit-stream that complements the lossy-coded bit-stream of the core layer.

In the SLS decoder, the lossy bit-stream is decoded by an AAC decoder to generate a lossy audio signal for playback. This bit-stream is also used by an error mapping process to reconstruct the part of IntMDCT coefficients coded in the core layer. The enhancement bit-stream is entropy-decoded back to the IntMDCT residual, which is added to the output of the error mapping process to recover the full IntMDCT coefficients. These coefficients are transformed by an Integer Inverse

Modified Discrete Cosine Transform (IntIMDCT) back to the original audio samples.

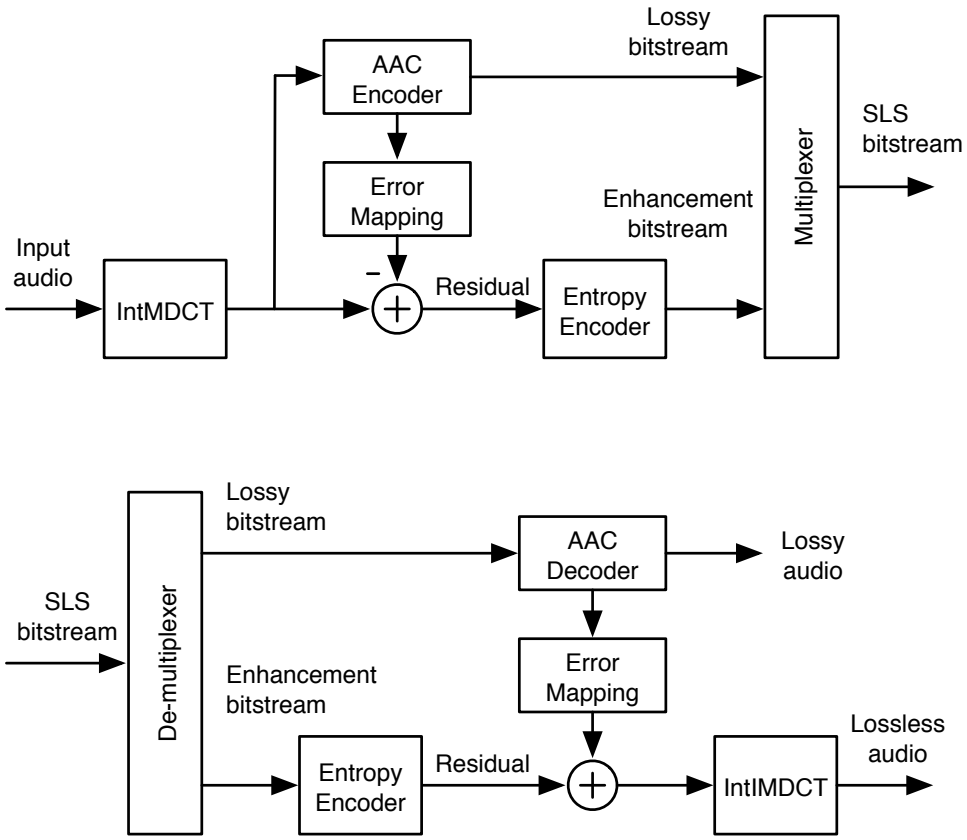

Figure 3.1: Block diagram of MPEG-4 SLS: encoder (top) and decoder (bottom)

3.1 IntMDCT

Modern perceptual audio coders such as the MPEG-2/4 AAC, Dolby AC-3, and Windows Media Audio (WMA) encode audio signals in the frequency domain. For converting audio to the frequency domain, the Modified Discrete Cosine Transform (MDCT) is selected because of its good energy-compacting capability and its *critical-sampling* property, i.e., N output samples are generated for N input samples. Critical sampling is important to coding applications as no extra samples are generated during frequency domain processing.

IntMDCT was introduced in [16] as a reversible, integer-to-integer transform

that closely approximates the MDCT. IntMDCT inherits the good properties of the MDCT such as energy-compacting and critical-sampling, and is used in SLS as well. IntMDCT is derived from the MDCT, which can be decomposed into a *Windowing* operation and a *Type-IV Discrete Cosine Transform (DCT-IV)* [83, 84] as shown in Figure 3.2.

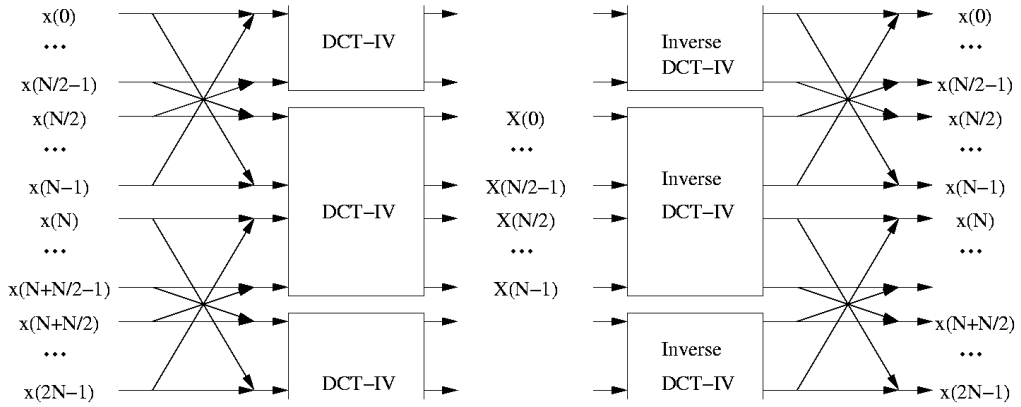

Figure 3.2: MDCT and inverse MDCT by Windowing and DCT-IV [15]

The Windowing operation consists of a group of Givens rotations, also commonly known as *butterflies*. In IntMDCT, each Givens rotation is implemented using three lifting steps [14] as follows:

$$\begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix} \mapsto$$

$$\begin{pmatrix} 1 & -\frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -w(k) & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix}$$

$$k = 0, \cdots, \tfrac{N}{2} - 1$$

where $w(k)$ is the window coefficients of MDCT [16].

After each lifting step, a rounding operation is applied to ensure data values remain in the integer domain. Every lifting step can be simply inverted by reversing the data flow direction and toggling the sign of the summation. This process is illustrated by Figure 3.3.

3.1.1 Integer DCT-IV

In IntMDCT, the DCT-IV is calculated by an invertible integer transform, called *Integer DCT-IV*. An efficient scheme, called *Matrix Lifting* [**P1**, **P2**, **P3**], or alterna-

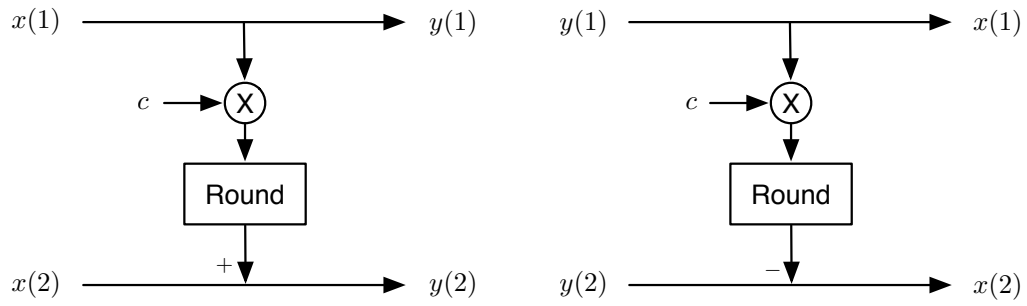

Figure 3.3: Lifting step: forward integer-to-integer mapping (left) and inverse integer-to-integer mapping (right)

tively *Multi-Dimensional Lifting* [86], was proposed to design the Integer DCT-IV in SLS. The resulted integer DCT-IV algorithm uses the minimal number of rounding operations, and demonstrates the lowest approximation error to the DCT-IV.

The basic principle behind the Matrix Lifting scheme is shown by the following block matrix decomposition, where $\mathbf{T}$ denotes an invertible matrix and $\mathbf{I}$ the identity matrix:

$$\begin{pmatrix} \mathbf{T} & 0 \\ 0 & \mathbf{T}^{-1} \end{pmatrix} = \begin{pmatrix} -\mathbf{I} & 0 \\ \mathbf{T}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{T} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} 0 & \mathbf{I} \\ \mathbf{I} & \mathbf{T}^{-1} \end{pmatrix}$$

The three blocks in this decomposition are called *lifting matrices*. Similar to the conventional 2×2 lifting step, lifting matrices can be converted to invertible integer mappings by rounding the floating-point values after matrix multiplications, and toggling the signs of the summation operations. The process is illustrated in Figure 3.4.

By applying the Matrix Lifting scheme to the DCT-IV, the following decomposition is obtained:

$$\tfrac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{C}_N^{IV} & 0 \\ 0 & \mathbf{C}_N^{IV} \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{I} & 0 \\ 0 & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{\mathbf{C}_N^{IV}}{\sqrt{2}} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ -\sqrt{2}\mathbf{C}_N^{\mathrm{IV}} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{\mathbf{C}_N^{IV}}{\sqrt{2}} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\frac{1}{2}\mathbf{I} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{I} & \mathbf{I} \end{pmatrix}$$

In the case of stereo signals, this decomposition is used to obtain an integrated calculation of the Mid/Sum transform and the Integer DCT-IV for the left and the right channels. Combined with the Windowing operation, the above decomposition gives the so-called *Stereo IntMDCT* which transforms two block of input samples together. For Stereo IntMDCT, the number of rounding operations is $6N$ per channel pair, or equivalently, 3 rounding operations per sample.

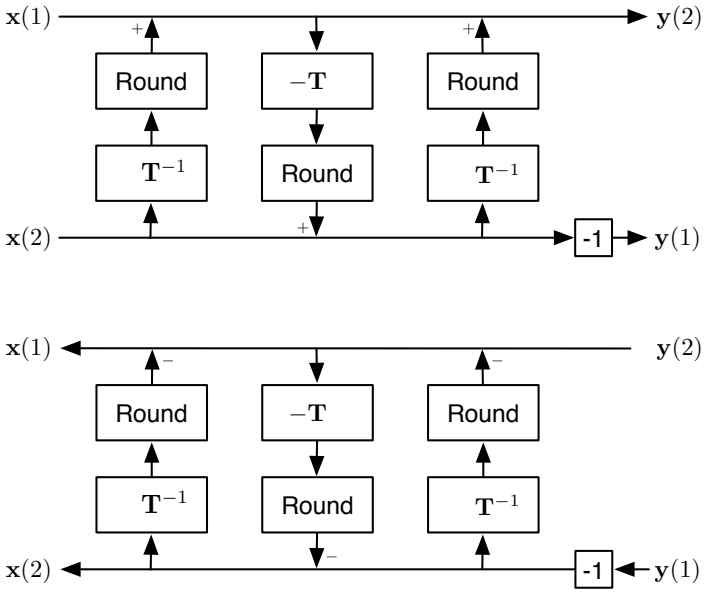

Figure 3.4: Matrix lifting: forward integer-to-integer mapping (top) and inverse integer-to-integer mapping (bottom)

In the case of mono signals, the same structure can be utilized by including some additional lifting steps to obtain the Integer DCT-IV of one block, or *Mono IntMDCT*. For Mono IntMDCT, 4 rounding operations per sample are required. The Mono IntMDCT is given as follows:

$$\mathbf{C}_N^{IV} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & -\mathbf{I} \end{pmatrix} \begin{pmatrix} 1 & & & & & cs(0) \\ & \cdots & & & \cdots & \\ & & 1 & cs(\frac{N}{2}-1) & & \\ & & & 1 & & \\ & & & & \cdots & \\ & & & & & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & & & & \\ & \cdots & & & \\ & & 1 & & \\ & & s(\frac{N}{2}-1) & 1 & \\ & \cdots & & & \cdots \\ s(0) & & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & cs(0) \\ & \cdots & & & \cdots \\ & & 1 & cs(\frac{N}{2}-1) & \\ & & & 1 & \\ & & & & \cdots \\ & & & & 1 \end{pmatrix}$$
$$\begin{pmatrix} \mathbf{I} & \frac{\mathbf{C}_N^{IV}}{\sqrt{2}} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ -\sqrt{2}\mathbf{C}_N^{\mathrm{IV}} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{\mathbf{C}_N^{IV}}{\sqrt{2}} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\frac{1}{2}\mathbf{I} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{I} & \mathbf{I} \end{pmatrix} \mathbf{QP}$$

with the values

$$s(k) = \sin\frac{(2k+1)\pi}{4N},\ cs(k) = \frac{\cos\frac{(2k+1)\pi}{4N} - 1}{\sin\frac{(2k+1)\pi}{4N}},\ k = 0, \cdots, \frac{N}{2} - 1$$

and the permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ with

$$\begin{aligned}
&\mathbf{P}_{4k,4k} = \mathbf{P}_{4k+1,4k+1} = \mathbf{P}_{4k+2,4k+3} = \mathbf{P}_{4k+3,4k+2} = 1\ ,\ k = 0, \cdots, \frac{N}{4} - 1 \\
&\mathbf{P}_{k,l} = 0\ ,\ \text{else}
\end{aligned}$$

i.e., every second pair of values is swapped, and

$$\begin{aligned}
&\mathbf{Q}_{k,2k} = \mathbf{Q}_{N/2+k,2k+1} = 1\ ,\ k = 0, \cdots, \frac{N}{2} - 1 \\
&\mathbf{Q}_{k,l} = 0\ ,\ \text{else}
\end{aligned}$$

i.e., the even indices are arranged in the first half, the odd indices are arranged in the second half. Thus the DCT-IV is basically decomposed into 8 lifting matrices.

3.1.2 Noise Shaping

Due to the rounding operations in the lifting steps, IntMDCT introduces an approximation noise to the audio spectral data. This approximation noise is found to be audible for some audio signals that contain only a small amount of energy in the high frequency band. To solve this problem, a noise shaping technique [86] is utilized to shape the approximation noise to a low-pass characteristics. The reason is that most audio signals contain more energy in the low frequency band than in the high frequency band, shaping the IntMDCT approximation noise to a low-pass shape moves the noise spectrum under that of the audio signal, creating a masking effect that makes the noise inaudible at the presence of audio signal. The noise shaping filter is illustrated in Figure 3.5, where a first-order low-pass filter is used to shape the rounding noise in the lifting step to a desired low-pass characteristics. The IntMDCT approximation noises before and after noise shaping are compared in Figure 3.6.

3.2 Error Mapping

The error mapping process reconstructs the portion of the IntMDCT spectral that has already been encoded by the AAC core layer. The AAC core encoder can be viewed as a quantization and coding process [41], where the IntMDCT spectral data

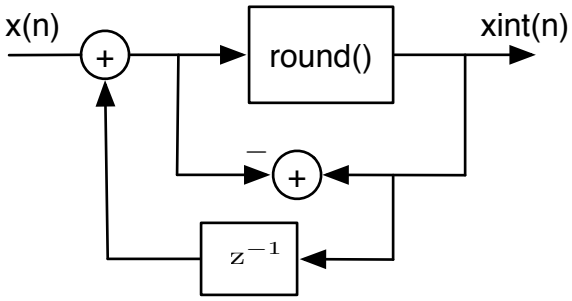

Figure 3.5: Noise shaping filter for IntMDCT [88]

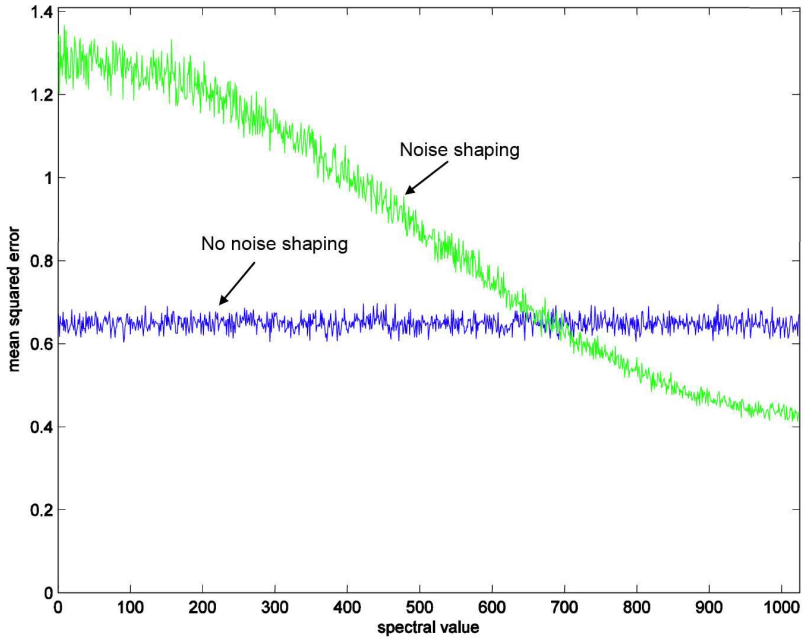

Figure 3.6: Mean squared approximation error of IntMDCT before and after noise shaping [88]

are first grouped into different *scale factor bands (sfb)*, and then quantized with different quantization step-size and entropy-coded to produce the AAC bit-stream.

In error mapping, let $i[k]$ be the quantized IntMDCT coefficient $c[k]$ for the scale factor band s, a reconstructed spectral value is calculated according to:

$$thr(i[k]) = \begin{cases} \mathrm{sgn}(i[k]) \left[\sqrt[4]{2^{scale\_factor(s)}} (|i[k]| - C)^{4/3} \right] , & i[k] \neq 0 \\ 0 , & i[k] = 0 \end{cases}$$

where $scale\_factor[s]$ is the scale factor that determines the quantization step size for sfb s, and the rounding offset is given by $C = 0.4054$. The error mapping processing then calculates the IntMDCT spectral residual in the following way:

$$e[k] = \begin{cases} c[k] - \lfloor thr(i[k]) \rfloor , & i[k] \neq 0 \\ c[k] , & i[k] = 0 \end{cases} \qquad k = 1, \cdots, 1023.$$

The process is illustrated in Figure 3.7. It is found that the amplitude of $e[k]$ can be well approximated by a one-side Laplacian (Geometrical) distribution.

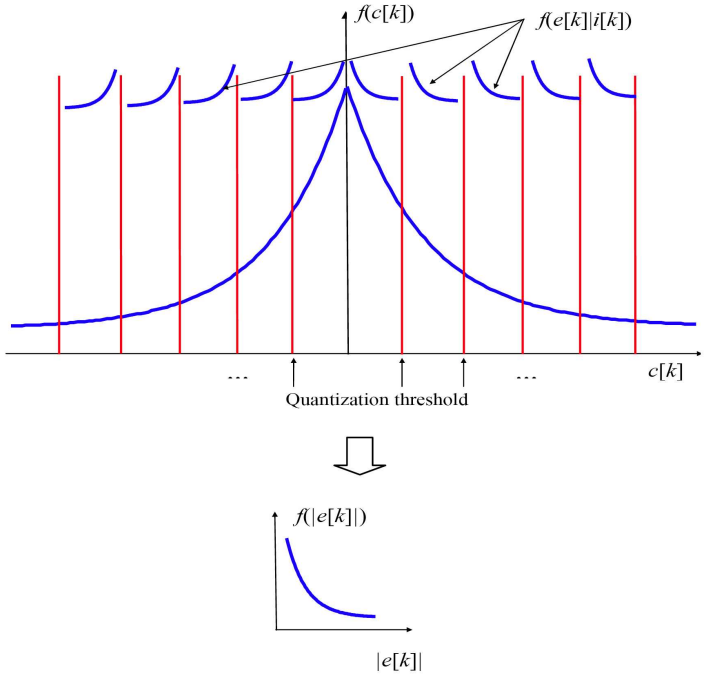

Figure 3.7: Illustration of the error mapping process [88]. Top: The pdf of the Laplacian distributed IntMDCT coefficient and the conditional pdf of the IntMDCT residual; Bottom: The unconditioned pdf of the amplitude of the IntMDCT residual.

3.3 Bit-plane Coding

The IntMDCT spectral residual is coded in the bit-plan coding approach. The IntMDCT spectral residual is first represented in the binary format, and then scanned and coded symbol by symbol from the Most Significant Bit (MSB) to the Least Significant Bit (LSB) to generate the compressed bit-stream. This process is illustrated in Figure 3.8. The bit-plane coding approach provides a natural way to generate a scalable bit-stream. Any lossy bit-stream at a given intermediate rate can be obtained by simply truncating the bit-stream.

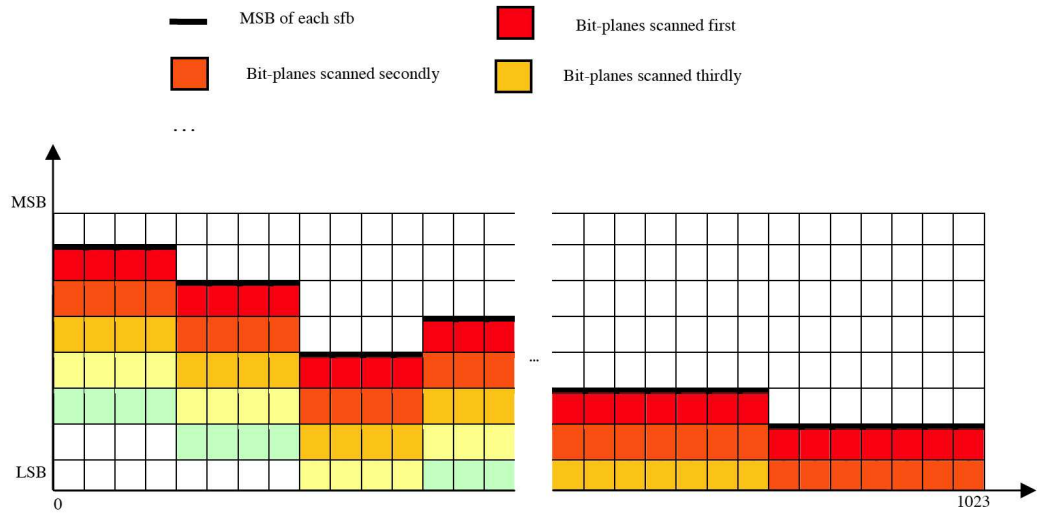

Figure 3.8: The bit-plane order in MPEG-4 SLS [88].

In bit-plane coding, two types of entropy codes can be selected. The first one, called *Bit-Plane Golomb Code (BPGC)* [90], is a arithmetic code using a pre-determined frequency assignment rule from the statistical properties of a geometrically distributed source. Despite its simplicity, BPGC achieves good lossless compression ratio for a geometrically distributed integer source.

The second code, called *Context-Based Arithmetic Code (CBAC)* [**P5**], gives slightly better coding efficiency. It is inspired by the fact that the probability distribution of bit-plane symbols is usually correlated with their frequency location and the significance state of the adjacent spectral lines. These correlations are captured in CBAC in the form of various contexts, which include the location of the IntMDCT spectral data and significant states of their adjacent spectral lines. These contexts determine the frequency tables used by CBAC to code the IntMDCT residual.

There is also a so-called *Low Energy Mode Code (LEMC)* for coding IntMDCT

spectral data from sfbs with extremely low energy. Such sfbs exist in the high frequency regions of the audio spectral or during the silence portions of the audio signal. Details of BPGC/CBAC/LEMC can be found in [**P5**].

3.4 Summary

MPEG-4 SLS in a transform-domain scalable lossy to lossless audio coder. It uses an AAC coder to generate the lossy portion of the bit-stream. For the lossless portion, IntMDCT is used to transform the audio signals to the frequency. The IntMDCT algorithms are derived using Lifting-Matrices, giving the algorithms very low approximation noise. An IntMDCT spectral residual is generated by removing the portion of spectral data that has already been coded by the AAC coder. The IntMDCT residual is coded with bit-plane codings BPGC, CBAC, and LEMC.

Chapter 4

Summary of the Publications

IN **the first paper** [**P1**], we propose a method for realizing reversible integer-to-integer DCT type-IV (DCT-IV), an important component of the integer modified discrete cosine transform (IntMDCT). Firstly, we construct a transform matrix using two DCT-IV matrices. Next this transform matrix is factorized into three lifting matrices. After that, we approximate each lifting matrix with a reversible, integer-to-integer, nonlinear transform. The derived integer DCT-IV transform requires $1.5N$ rounding operations for transforming one block of audio samples, where N is the number of samples in the block. Its computational complexity is 1.5 times that of a DCT-IV computation. Compared with other state-of-the-art integer DCT-IV algorithms, the proposed one achieved the most accurate approximation to the floating-point DCT-IV transform. As a result, it has been applied into the MPEG-4 Scalable Lossless Audio Coding (SLS).

In the second paper [**P2**], we propose an alternative approach for deriving the integer DCT-IV algorithm. The DCT-IV matrix is directly factorized into five lifting matrices. Each lifting matrix is then approximated by a reversible, integer-to-integer, nonlinear transform. The derived integer DCT-IV transform requires rounding operations for transforming one block of audio samples. Unlike the integer DCT-IV algorithm in [**P1**], which transforms two data blocks together, the proposed algorithm transforms a single block each time. Compared with other state-of-the-art single-block based integer DCT-IV algorithms, the proposed algorithm achieved the second best approximation accuracy to the floating-point DCT-IV transform. Its complexity is slightly more than 2 times that of a DCT-IV computation.

In the third paper [**P3**], the matrix factorization method in [**P1**] is used to derive an integer DFT algorithm. A transform matrix is first constructed using two normalized DFT matrices. This transform matrix is factorized into three lifting

matrices. The proposed integer DFT algorithm is derived by approximating each lifting matrix with non-linear transforms similar to those in [**P1**, **P2**]. The number of rounding operations is $3N$ per block. The proposed algorithm is so far the most accurate integer DFT. Its complexity is 1.5 times that of a DFT computation.

In the fourth paper [**P4**], we propose a cascaded RLS-LMS predictor for reducing the correlations among adjacent audio samples. The predictor has been incorporated into the MPEG-4 Audio Lossless Coding (ALS) international standard. The predictor consists of cascaded stages of simple linear predictors, with the prediction error at the output of one stage passed to the next stage as the input signal. A linear combiner adds up the intermediate estimates at the output of each prediction stage to give a final estimate of the RLS-LMS predictor. In the predictor cascade, the first prediction stage is a simple first-order predictor with a fixed coefficient value 1. The second prediction stage uses the Recursive Least Square algorithm to adaptively update the predictor coefficients. The subsequent prediction stages use the Normalized Least Mean Square algorithm to update the predictor coefficients. The coefficients of the linear combiner are then updated using the Sign-Sign Least Mean Square algorithm. For stereo audio signals, the RLS-LMS predictor uses both intra-channel prediction and inter-channel prediction, which has resulted in a 3% improvement in compression ratio over using only the intra-channel prediction.

In the fifth paper [**P5**], we propose two coding methods for improving the coding efficiency of SLS, namely, the context-based arithmetic code (CBAC) method and the low energy mode code method. In the CBAC method, we use three types of context, namely, the *frequency band (FB)* context, the *distance to lazy (D2L)* context, and the *significant state (SS)* context, to further refine the BPGC coding process for better compression of the IntMDCT residual. In the second method, a low energy code is used to replace the BPGC process for IntMDCT spectral data from time/frequency regions with extremely low-energy level. Since these spectral data are dominated by the rounding errors of the IntMDCT algorithm, they have a probability distribution that is far away from the Laplacian distribution. The proposed methods improve the compression ratio performance of the MPEG-4 SLS encoder by 2.61% and 1.52% for the MPEG 48kHz/16bit testing set and 96kHz/24bit testing set, respectively. These methods have been successfully incorporated into the MPEG-4 Scalable Lossless Coding international standard.

The contributions of the thesis can be summarized as follows. In [**P1**, **P2**, **P3**, **P4**], the author is responsible for the development and implementation of the algorithm, running experiments, and writing of the paper. In [**P5**], the author took part in the development of the algorithm, running experiments, and writing of the paper.

Chapter 5

Summary of the Results

The performances of MPEG-4 lossless audio coding tools have been compared with other state-of-the-art lossless audio coders in [**P4**]. The experiments were run on a Pentium-4 2.4 GHz PC using standard MPEG testing samples [5], and the results are summarized in Table 5.1. In Table 5.1, speeds of the lossless compressors are measured in terms of *real-time*, defined as the ratio of the encoding/decoding time over the actually play time of a sound clip. The lossless compressors used are divided into two categories: standard coders from MPEG, and non-standard, proprietary ones. The first category includes the MPEG-4 SLS RM8 [60] and the MPEG-4 ALS RM18 [60] running in two predictor modes: the RLS-LMS mode and the LPC mode. Both these coders were implemented in 32-bit fixed-pointed C code. Coders for the second category are selected according to the following considerations: OptimFROG [18] is reported [30, 40, 78, 79] to be one of the top lossless audio coders in terms of the compression ratio, and is therefore used as a major benchmark for our comparison. Because of the widespread popularity over internet, the Monkey's coder [4] is also included in the comparison. The compression ratio results are plotted in Figure 5.1.

Clearly, the highest compression ratio is provided by OptimFROG 4.600ex, closed followed by the ALS RM18 coder running in the RLS-LMS predictor mode. This suggests that the latter becomes one of the top performers in the field in terms of the lossless compression ratio. Among the five tested coders, the ALS RM18 coder running in the RLS-LMS predictor mode gives the slowest encoding/decoding speed. This slow speed of the coder results from the high complexity of the RLS-LMS predictor, which consists of a computationally intensive RLS filter, as well as large-order LMS filters.

The ALS reference software RM18 was implemented in 32-bit fixed-point arithmetic. In the RLS-LMS predictor, to guarantee convergence of the adaptive coeffi-

cients update recursions, computations inside the RLS recursions must be performed with a high numerical precision [32]. This requirement is generally not a problem for implementations done in double-precision floating-point, but remains a challenging issue for fixed-point implementations because of the much smaller dynamic range to represent values in fixed-point. In the RLS and NLMS recursions, each multiplication/division operation is coupled with necessary pre-scaling, post-scaling and range comparison instructions to keep the numerical precision high. As a result, a multiplication done in one floating-point step can only be achieved by several fixed-point steps. This overhead with fixed-point implementation also contributes to the high-complexity of the RLS-LMS predictor.

Table 5.1: Comparison of lossless compressors

Lossless Compressors		Compression Ratio	Speed (× real-time)	
			Encode	Decode
Standard	ALS RM18 (RLS-LMS)	2.172	0.4	0.4
	ALS RM18 (LPC)	2.148	20	40
	SLS RM8	2.123	22	25
Non-standard	OptimFROG 4.600ex	2.174	1.2	2.1
	Monkey's Audio 3.99	2.097	6	6

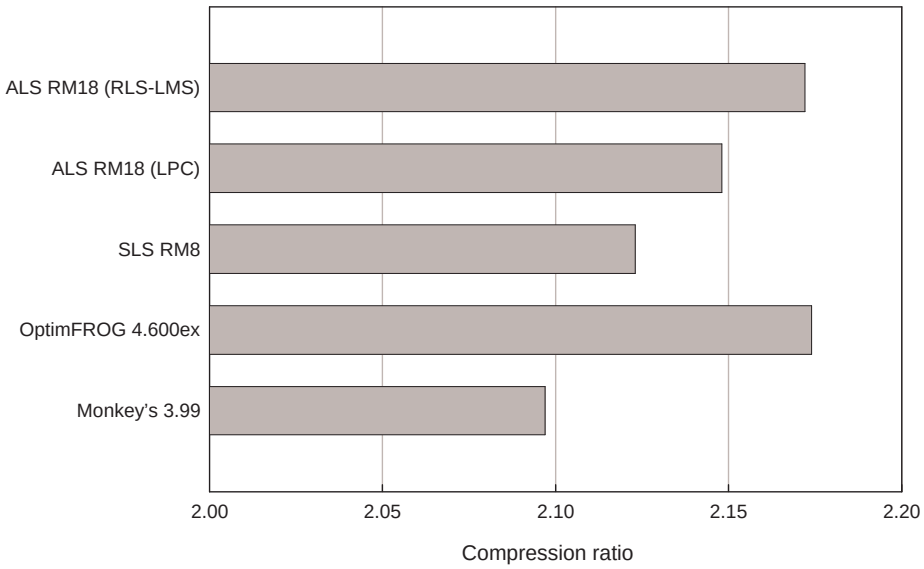

Figure 5.1: Comparison of lossless compressors

Chapter 6

Conclusions

In this thesis, we presented new technologies for lossless audio compression. The matrix-lifting method was proposed for developing integer transforms, where the target transform is first factorized into a series of lifting matrices, and then converted to integer transform. Compared to previous methods that utilize conventional 2-by-2 lifting steps, the new method reduces the number of rounding operations in the integer transform from $O(N\log_2 N)$ to $O(N)$. As a result, the approximation accuracy of the integer transform is greatly improved, boosting the compression ratio in lossless compression, as well as improving the quality of the perceptually-coded audio in a hybrid lossy/lossless coding structure. IntMDCT algorithms developed using this method have been adopted in the MPEG-4 SLS standard.

The proposed cascaded RLS-LMS prediction improved the linear prediction gain on audio signals, which are generally non-stationary. By using cascaded stages of DPCM, RLS, and LMS predictors, the RLS-LMS predictor generates a smaller residual signal than conventional LPC prediction, leading to a smaller file size after entropy coding. Combining joint-stereo prediction with RLS-LMS prediction further improves the compression ratio. The RLS-LMS prediction technology has been adopted in the MPEG-4 ALS standard, which demonstrates the highest level of compression ratio among state-of-the-art lossless coders.

MPEG-4 lossless audio coding technologies support various application scenarios. For example, SLS provides scalable bit-stream that can be arbitrarily truncated to suit different transmission bandwidth requirements. With LPC prediction, ALS gives a good balance between compression ratio and encoding/decoding speed, and with RLS-LMS prediction, it provides a very high compression ratio.

References

[1] Adistambha, K., Ritz, C. H., Burnett, I. S., and Lukasiak, J. A codebook-based cascade coder for embedded lossless audio coding. In *120th AES Convention* (Paris, France, May 2006).

[2] Adistambha, K., Ritz, C. H., and Lukasiak, J. Embedded lossless audio coding using linear prediction and cascade coding. In *8th Int. Symp. on DSP and Communication Systems (DSPCS'05) and 4th Workshop on the Internet, Telecommunications and Signal Processing (WITSP'05)* (Sunshine Coast, Australia, December 2005), pp. 44–49.

[3] Apple Inc. Apple Lossless Audio Codec (ALAC) (on-line), http://www.apple.com/itunes/import.html (20.9.07).

[4] Ashland, M. Monkey's Audio (APE) (on-line), http://www.monkeysaudio.com/ (20.9.07).

[5] Audio Research Labs. (on-line), http://www.audioresearchlabs.com/ (20.9.07).

[6] Bevin, M. Lossless Audio (LA) (on-line), http://www.lossless-audio.com/ (20.9.07).

[7] Bosi, M., Brandenburg, K., Quackenbush, S., Fielder, L., Akagiri, K., Fuchs, H., Dietz, M., Herre, J., Davidson, G., and Oikawa, Y. ISO/IEC MPEG-2 Advanced Audio Coding. *J. Audio Eng. Soc. 45*, 10 (1997), 789–814.

[8] Bryant, D. WavPack (WV) (on-line), http://www.wavpack.com/ (20.9.07).

[9] Cellier, C., Chenes, P., and M. Rossi, t. .

[10] Chen, Y. J., Oraintara, S., and Nguyen, T. Video compression using integer DCT. In *Proc. IEEE Int. Conf. Image Processing (ICIP'00)* (Vancouver, Canada, September 200), vol. 2, pp. 844–847.

[11] Coalson, J. Free Lossless Audio Codec (FLAC) (on-line), http://flac.sourceforge.net/ (20.9.07).

[12] Craven, P. G., and Gerzon, M. Lossless coding for audio discs. *J. Audio Eng. Soc. 44*, 9 (1996), 706–720.

[13] Craven, P. G., Law, M., and Stuart, J. Lossless compression using IIR prediction filters. In *102nd AES Convention* (Munich, Germany, March 1997).

Preprint 4415.

[14] Daubechies, I., and Sweldens, W. Factoring wavelet transform into lifting steps. *J. Fourier Anal. Appl. 4*, 3 (November 1998), 245–267.

[15] Geiger, R., Herre, J., Koller, J., and Brandenburg, K. INTMDCT - a link between perceptual and lossless audio coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'02)* (Orlando, USA, May 2002), vol. 2, pp. 1813–1816.

[16] Geiger, R., Sporer, T., Koller, J., and Brandenburg, K. Audio coding based on integer transforms. In *111th AES Convention* (New York, USA, September 2001).

[17] Geiger, R., Yokotani, Y., and Schuller, G. Improved integer transforms for lossless audio coding. In *Conf. Rec. 37th Asilomar Conf. on Signals, Systems and Computers* (USA, November 2003), vol. 2, pp. 2119–2123.

[18] Ghido, F. OptimFROG (OFR) (on-line), http://www.losslessaudio.org/ (20.9.07).

[19] Ghido, F. An asymptotically optimal predictor for stereo lossless audio compression. In *Proc. IEEE Data Compression Conf. (DCC'03)* (Salt Lake City, USA, March 2003), p. 429.

[20] Ghido, F., and Tabus, I. Accounting for companding nonlinerities in lossless audio compression. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'07)* (Honolulu, USA, April 2007), vol. 1, pp. I261–264.

[21] Ghido, F., and Tabus, I. Adaptive design of the preprocessing Stage for lossless audio compression. In *122nd AES Convention* (Vienna, Austria, May 2007).

[22] Gilbert, E. N., and Moore, E. F. Variable-length binary encodings. *Bell Syst. Tech. J. 38* (July 1959), 932–967.

[23] Giurcaneanu, C. D., and Tabus, I. Low-complexity transform coding with integer-to-integer transforms. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'01)* (Salt Lake City, USA, May 2001), vol. 4, pp. 2601–2604.

[24] Giurcaneanu, C. D., Tabus, I., and Astola, J. Linear prediction from subbands for lossless audio compression. In *Proc. IEEE Nordic Signal Processing Symposium (NORSIG'98)* (Vigso, Denmark, June 1196), pp. 225–228.

[25] Giurcaneanu, C. D., Tabus, I., and Astola, J. Forward and backward design of predictors for lossless audio coding. In *Proc. 12th Int. Conf. on Control Systems and Computer Science (CSCS-12)* (Bucharest, Romania, May 1999), vol. 1, pp. 396–401.

[26] Giurcaneanu, C. D., Tabus, I., and Astola, J. Integer wavelet transform based lossless audio compression. In *Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP'99)* (Antalya, Turkey, June 1999), vol. 1, pp. 378–3821.

[27] Giurcaneanu, C. D., Tabus, I., and Astola, J. Adaptive context-based sequential prediction for lossless audio compression. *Signal Processing 80* (2000), 2283–2294.

[28] Giurcaneanu, C. D., Tabus, I., and Stanciu, C. Lossless audio compression with optimal codes for Laplacian distribution. In *Proc. 1st South-East European Symposium on Interdisciplinary Approaches in Fractal Analysis (IAFA'03)* (Bucharest, Romania, May 2003), pp. 73–78.

[29] Hans, M., and Schafer, R. W. Lossless compression of digital audio. *IEEE Signal Processing Mag. 18* (July 2001), 21–32.

[30] Hans Heijden's Lossless Comparison. (on-line), http://web.inter.nl.net/users/hvdh/lossless/lossless.htm (20.9.07).

[31] Hao, P., and Shi, Q. Matrix factorizations for reversible integer mapping. *IEEE Trans. Signal Processing 49* (October 201), 2314–2324.

[32] Haykin, S. *Adaptive Filter Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1999.

[33] Huang, H., Fränti, P., Huang, D., and Rahardja, S. Cascaded RLS-LMS prediction in MPEG-4 lossless audio coding. *IEEE Trans. on Audio, Speech, and Language Processing accepted for Publication*.

[34] Huang, H., Lin, X., Rahardja, S., and Yu, R. A method for realizing reversible integer type-IV discrete cosine transform (IntDCT-IV). In *Proc. IEEE Int. Conf. Signal Processing (ICSP'04)* (Beijing, China, August 2004), vol. 1, pp. 101–104.

[35] Huang, H., Lin, X., Rahardja, S., and Yu, R. Integer DFT with high transform accuracy. In *Proc. 4th Int. Conf. on Information, Communications and Signal Processing (ICICS'05)* (Bangkok, Thailand, December 2005), pp. 1471–1474.

[36] Huang, H., Rahardja, S., Lin, X., Yu, R., and Fränti, P. Cascaded RLS-LMS prediction in MPEG-4 lossless audio coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'06)* (Toulouse, France, May 2006), vol. 5, pp. V181–184.

[37] Huang, H., Rahardja, S., Yu, R., and Lin, X. A fast algorithm of integer MDCT for lossless audio coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'04)* (Montreal, Canada, May 2004), vol. 4, pp. IV177–IV180.

[38] Huang, H., Rahardja, S., Yu, R., and Lin, X. Integer MDCT with enhanced approximation of the DCT-IV. *IEEE Trans. on Signal Processing 54*, 3 (March 2006), 1156–1159.

[39] Huang, H., Yu, R., Lin, X., and Rahardja, S. Method for realising reversible integer type-IV discrete cosine transform. *Electro. Lett. 40*, 8 (April 2004), 514–515.

[40] Hydrogen Audio's Lossless Comparision. (on-line), http://wiki.hydrogenaudio.org/index.php?title=Lossless_comparison (20.9.07).

[41] ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group. Information Technology - Coding of Audiovisual Objects, Part 3. Audio, Subpart 4 Time/Frequency Coding, 1998.

[42] ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group. Final Call for Proposals on MPEG-4 Lossless Audio Coding. N5208, Shanghai, China, October 2002.

[43] ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group. Text of 14496-3:2001/FDAM 5, Scalable Lossless Coding (SLS). N7366, Poznan, Poland, July 2005.

[44] Komatsu, K., and Sezaki, K. Design of lossless LOT and its performance evaluation. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'00)* (Istanbul, Turkey, June 2000), vol. 4, pp. 2119–2122.

[45] Li, J. A progressive to lossless embedded audio coder (PLEAC) with reversible modulated lapped transform. In *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'03)* (Baltimore, USA, July 2003), vol. 3, pp. III221–4.

[46] Li, J. Low noise reversible MDCT (RMDCT) and its application in progressive-to-lossless embedded audio coding. *IEEE Trans. Signal Processing 53*, 5 (May 2005), 1870–1880.

[47] Li, T., Rahardja, S., and Koh, S. N. Perceptually prioritized bit-plane coding for High-Definition Advanced Audio Coding. In *Proc. 8th IEEE Int. Symp. on Multimedia (ISM'06)* (San Diego, USA, December 2006), pp. 245–252.

[48] Li, T., Rahardja, S., and koh, S. N. Perceptually adaptive bit-plane coding for scalable audio. *Electro. Lett. 43*, 1 (January 2007), 60–61.

[49] Liebchen, T. Lossless audio coding using adaptive multichannel prediction. In *113th AES Convention* (Los Angeles, USA, October 2002). Preprint 5680.

[50] Liebchen, T. An introduction to MPEG-4 Audio Lossless Coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'04)* (Montreal, Canada, May 2004), vol. 3, pp. 1012–15.

[51] Liebchen, T., Moriya, T., Harada, N., Kamamoto, Y., and Reznik, Y. The MPEG-4 Audio Lossless Coding (ALS) standard - technology and applications. In *119th AES Convention* (New York, USA, October 2005).

[52] Liebchen, T., Purat, M., and Noll, P. Lossless transform coding of audio signals. In *102nd AES Convention* (Munich, Germany, September 1997). Preprint 4414.

[53] Liebchen, T., and Reznik, Y. Improved forward-adaptive prediction for MPEG-4 Audio Lossless Coding. In *118th AES Convention* (Barcelona, Spain, May 2005).

[54] Liebchen, T., Reznik, Y., Moriya, T., and Yang, D. MPEG-4 Audio Lossless Coding. In *116th AES Convention* (Berlin, Germany, May 2004).

[55] Magotra, N., McCoy, W., Livingston, F., and Stearns, S. Loss-

less data compression using adaptive filters. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'95)* (Detroit, USA, May 1995), vol. 2, pp. 1217–1220.

[56] Malvar, H. S. *Signal Processing with Lapped Transform.* Artech House, Boston MA, 1992.

[57] Microsoft. Windows Media Audio Lossless (WMAL) (on-line), http://www.microsoft.com/windows/windowsmedia/9series/codecs/audio.aspx (20.9.07).

[58] Moriya, T., Iwakami, N., Jin, A., and Mori, T. A design of lossy and lossless scalable audio coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'00)* (Istanbul, Turkey, June 2000), vol. 32, pp. II889–892.

[59] Moriya, T., Yang, D., and Liebchen, T. Extended linear prediction tools for lossless audio coding. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'04)* (Montreal, Canada, May 2004), vol. 3, pp. III1008–11.

[60] MPEG-4 ALS and SLS Reference Software. (on-line), http://isotc.iso.org/livelink/livelink/fetch/2000/2489/186491/PubliclyAvailable Standards.htm?nodeid=3851696&vernum=84 (20.9.07).

[61] MPEG-4 Audio Lossless Coding (ALS). (on-line), http://www.nue.tu-berlin.de/forschung/ projekte/lossless/mp4als.html (20.9.07).

[62] Oraintara, S., Chen, Y. J., and Nguyen, T. Q. Integer fast Fourier transform. *IEEE Trans. Signal Processing 50*, 3 (March 2002), 607–618.

[63] Painter, T., and Spanias, A. Perceptual coding of digital audio. *Proc. of the IEEE 88*, 4 (April 2000), 451–515.

[64] Park, S. H., Kim, Y. B., Kim, S. W., and Seo, Y. W. Multi-rate bit-sliced bit rate scalable audio coding. In *103rd AES Convention* (New York, USA, September 1997).

[65] Philips, W. The lossless dct for combined lossy/lossless image coding. In *Proc. IEEE Int. Conf. Image Processing (ICIP'98)* (Chicago, USA, October 1998), vol. 3, pp. 871–875.

[66] Plonka, G., and Tasche, M. Invertible integer DCT algorithms. *Appl. Comput. Harmon. Anal. 15* (1998), 247–269.

[67] Princen, J., Johnson, A., and Bradley, A. Subband/Transform coding using filter bank designs based on time domain aliasing cancellation. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'87)* (Dallas, USA, April 1987), vol. 12, pp. 2161–2164.

[68] Purat, M., Liebchen, T., and Noll, P. Lossless transform coding of audio signals. In *102nd AES Convention* (Munich, Germany, March 1997). Preprint 4414.

[69] Qiu, T. Lossless audio coding based on high order context modeling. In *Proc. 4th IEEE Workshop on Multimedia Signal Processing (MMSP'01)* (Cannes, France, October 2001), pp. 575–580.

[70] Rabiner, L. R., and Schafer, R. W. *Digital Processing of Speech Signals*. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[71] Rao, K. R., and Yip, P. *Discrete Cosine Transform: Algorithms, Advantages, and Applications*. Academic Press, Boston MA, 1990.

[72] Real Networks. RealAudio Lossless (RAL) (on-line), http://www.realnetworks.com/products/codecs/realaudio.html (20.9.07).

[73] Reznik, Y. A. Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS). In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'04)* (Montreal, Canada, May 2004), vol. 3, pp. 1024–7.

[74] Rice, R. F. Some practical universal noiseless coding techniques. *JPL Tech. Reps 79-22* (1979).

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

[76] Robinson, T. Shorten (SHN) (on-line), http://www.etree.org/shnutils/shorten/ (20.9.07).

[77] Schuller, G. D. T., Yu, B., Huang, D., and Edler, B. Perceptual audio coding using adaptive pre- and post-filters and lossless compression. *IEEE Trans. Speech Audio Processing 10*, 6 (September 2002), 379–390.

[78] Speek's Lossless Comparison. (on-line), http://members.home.nl/w.speek/comparison.htm/ (20.9.07).

[79] Synthetic Soul's Lossless Comparison. (on-line), http://synthetic-soul.co.uk/comparison/lossless/ (20.9.07).

[80] Tran, T. D. The binDCT: fast multiplierless approximation of the DCT. *IEEE Signal Processing Lett. 7* (June 2000), 141–144.

[81] True Audio. (TTA) (on-line), http://www.true-audio.com/ (20.9.07).

[82] Wang, J., Sun, J., and Yu, S. 1-D and 2-D transforms from integer to integer. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'03)* (Hong Kong, China, April 2003), vol. 2, pp. II549–552.

[83] Wang, Z. Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Trans. Acoust., Speech, Signal Processing 32* (August 1984), 803–816.

[84] Wang, Z. On computing the discrete Fourier and cosine transforms. *IEEE Trans. Acoust., Speech, Signal Processing 33* (October 1985), 1341–1344.

[85] Yang, Y., Giurcaneanu, C. D., and Tabus, I. An application of the piecewise autoregressive model in lossless audio coding. In *Proc. 7th Nordic Signal Processing Symposium (NORSIG'06)* (Reykjavik, Iceland, June 2006), pp. 326–329.

[86] Yokotani, Y., Geiger, R., Schuller, G. D. T., Oraintara, S., and Rao, K. R. Lossless audio coding using the IntMDCT and rounding error

shaping. *IEEE Trans. Audio Speech Language Processing 14*, 6 (November 2006), 2201–2211.

[87] Yokotani, Y., and Oraintara, S. Lossless audio compression using integer modified discrete cosine transform. In *Proc. Int. Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'03)* (Awaji Island, Japan, December 2003), pp. 120–126.

[88] Yu, R., Geiger, R., Rahardja, S., Herre, J., Lin, X., and Huang, H. MPEG-4 scalable to lossless audio coding. In *117th AES Convention* (San Francisco, USA, October 2004). Preprint 6183.

[89] Yu, R., and Ko, C. C. Lossless compression of digital audio using cascaded RLS-LMS prediction. *IEEE Trans. Speech Audio Processing 11* (November 2003), 532–537.

[90] Yu, R., Ko, C. C., Rahardja, S., and Lin, X. Bit-plane Golomb code for sources with Laplacian distributions. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'03)* (Hong Kong, China, April 2003), vol. 4, pp. IV277–280.

[91] Yu, R., Lin, X., and Rahardja, S. Advanced Audio Zip - a scalable perceptual and lossless audio codec. ISO/IEC JTC1/SC29/WG11 M9134, Awaja, Japan, December 2002.

[92] Yu, R., Lin, X., Rahardja, S., and Huang, H. Technical description of I2R's proposal for MPEG-4 Audio Scalable Lossless Coding (SLS): Advanced Audio Zip (AAZ). ISO/IEC JTC1/SC29/WG11 M10035, Brisbane, Australia, October 2003.

[93] Yu, R., Lin, X., Rahardja, S., and Huang, H. MPEG-4 scalable to lossless audio coding - emerging international standard for digital audio compression. In *Proc. 2005 7th IEEE Workshop on Multimedia Signal Processing (MMSP'05)* (Shanghai, China, October 2005), pp. 1–4.

[94] Yu, R., Lin, X., Rahardja, S., Ko, C. C., and Huang, H. Improving coding efficiency for MPEG-4 Audio Scalable Lossless Coding. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'05)* (Philadelphia, USA, March 2005), vol. 3, pp. III169–172.

[95] Yu, R., Rahardja, S., Lin, X., and Ko, C. C. A fine granular scalable to lossless audio coder. *IEEE Trans. Audio Speech Language Processing 14*, 4 (July 2006), 1352–1363.

[96] Zeng, Y., Cheng, L., Bi, G., and Kot, A. C. Integer DCTs and fast algorithms. *IEEE Trans. Signal Processing 49*, 11 (November 2001), 2274–2782.

1

Publication P1

H. Huang, R. Yu, X. Lin, S. Rahardja, Method for Realising Reversible Integer Type-IV Discrete Cosine Transform, *Electronics Letters*, Vol 40, Number 8, pp. 514-515, April 2004.

Method for realising reversible integer type-IV discrete cosine transform

H. Huang, R. Yu, X. Lin and S. Rahardja

A novel method for realising reversible integer discrete cosine transform type IV (DCT-IV) is presented. The method is derived by first constructing a transform matrix using DCT-IV matrix, then factorising the transform matrix into the product of three lifting matrices. The proposed method exhibits low rounding number and is the most accurate to the floating-point DCT-IV transform.

Introduction: Reversible integer transforms are important tools for signal lossless compression. Such a transform has integer inputs and generates integer outputs. The original integer inputs can be fully recovered through the corresponding inverse integer transform. In audio signal processing, the modified discrete cosine transform (MDCT) is widely used, e.g. in MPEG-4 advanced audio coding (AAC). With the rapid growth of broadband access at home and ever-decreasing storage cost, internet audio with scalable quality up to lossless will soon become a reality. Seeing this trend, MPEG recently issued a call for proposal (CFP) for audio lossless coding (ALS) [1]. Integer MDCT has been adopted in the CFP. This encouraged us to search for integer MDCT (IntMDCT) algorithms which not only lead to efficient implementation, but also approximate the MDCT as close as possible, so that good properties of the MDCT are retained, e.g. spectral compaction. It is shown in [2] that MDCT can be efficiently realised by cascading a band of Givens rotations and a DCT-IV transform block. It is well known that a Givens rotation can be realised by three lifting steps for integer implementation [3]. The remaining task is to find an efficient integer DCT-IV (IntDCT-IV) algorithm. In this Letter, we propose a novel method for realising reversible integer DCT-IV (IntDCT-IV). The proposed method is very simple to implement, has low complexity and offers extremely low approximation error, thereby providing an attractive tool for application in the currently hot area of lossless coding.

Definition: Let $\mathbf{C}_N^{IV}$ be the $N \times N$ transform matrix of type-IV DCT

$$\mathbf{C}_N^{IV} = \sqrt{\frac{2}{N}}\left[\cos\left(\frac{(m+0.5)(n+0.5)\pi}{N}\right)\right] \quad m, n = 0, 1, \ldots, N-1 \tag{1}$$

where N is the transform size which is a power of two, $N = 2^i$, i is an integer greater than zero, m and n are matrix indices.

Definition: A lifting matrix is a $2N \times 2N$ matrix of the following form

$$\mathbf{L}_{2N} = \begin{bmatrix} \pm\mathbf{I}_N & \mathbf{A}_N \\ \mathbf{O}_N & \pm\mathbf{I}_N \end{bmatrix} \tag{2}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix, $\mathbf{O}_N$ is the $N \times N$ zero matrix, $\mathbf{A}_N$ is an arbitrary $N \times N$ matrix. For lifting matrix $\mathbf{L}_{2N}$, reversible integer to integer mapping is realised in the same way as the 2×2 lifting step in [3]. The only difference is that rounding is applied to a vector instead of a single variable. Obviously, the transposition of $\mathbf{L}_{2N}$, $\mathbf{L}_{2N}^T$ is also a lifting matrix.

Definition: Let $\mathbf{T}_{2N}$ be a $2N \times 2N$ transform matrix

$$\mathbf{T}_{2N} = \begin{bmatrix} \mathbf{O}_N & \mathbf{C}_N^{IV} \\ \mathbf{C}_N^{IV} & \mathbf{O}_N \end{bmatrix} \tag{3}$$

Theorem: Matrix $\mathbf{T}_{2N}$ can be factorised into three lifting matrices as follows

$$\mathbf{T}_{2N} = \begin{bmatrix} \mathbf{I}_N & \mathbf{O}_N \\ -\mathbf{C}_N^{IV} & \mathbf{I}_N \end{bmatrix} \cdot \begin{bmatrix} -\mathbf{I}_N & \mathbf{C}_N^{IV} \\ \mathbf{O}_N & \mathbf{I}_N \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_N & \mathbf{O}_N \\ \mathbf{C}_N^{IV} & \mathbf{I}_N \end{bmatrix} \tag{4}$$

From (4), we can simultaneously compute the integer DCT-IV of two $N \times 1$ vectors.

Results: In Table 1, we use the mean-squared-error (MSE) to show the performance of the proposed method in terms of transform accuracy. The measuring scheme is shown in Fig. 1, where Fig. 1*a* is for forward transform and Fig. 1*b* is for inverse transform. The scheme was proposed by MPEG-4 lossless audio coding group during evaluation process for a new lossless audio coding standard [4].

Table 1: MSE comparison of IntDCT-IV and IntIDCT-IV algorithms (48 kHz/16-bit test set)

N	IntDCT-IV	IntIDCT-IV
16	0.232	0.198
32	0.231	0.197
64	0.231	0.196
128	0.231	0.196
256	0.230	0.195
512	0.231	0.195
1024	0.231	0.195
2048	0.230	0.195
4096	0.230	0.195

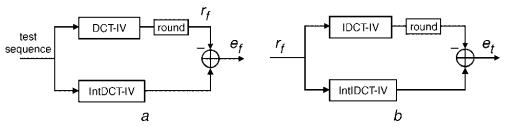

Fig. 1 *Performance evaluation diagram for IntDCT-IV and IntIDCT-IV algorithms*

a IntDCT-IV
b IntIDCT-IV

Definition: The MSEs for the forward IntDCT-IV and integer inverse DCT-IV (IntIDCT-IV) are given as

$$MSE = \frac{1}{K}\sum_{j=0}^{K-1}\frac{1}{N}\sum_{i=0}^{N-1} e_i^2 \tag{5}$$

where the error signal e is e_f for IntDCT-IV, and e_r for IntIDCT-IV as in Fig. 1. is the total number of sample blocks used in the evaluation. A total of 450 seconds with 15 different types of music files are used in the 48 kHz/16-bit test set. It can be seen from Table 1 that the MSE of the proposed method is irrelevant to processing block length and the approximation error is extremely low.

Conclusion: We propose a novel algorithm for realising reversible integer type-IV DCT. This algorithm requires only $1.5N$ number of rounding operations for every block of N input samples. As a result, the approximation error is extremely low. One important application of the proposed algorithm is for the audio lossless coding which utilises IntMDCT with an IntDCT-IV core inside. The proposed algorithm is suitable for both mono and stereo applications. In mono application, two consecutive blocks of samples are grouped and processed together. This introduces a signal delay of one block length when compared to single-block processing. However, in stereo applications, this extra block delay can be prevented, if simultaneous sample blocks from the left and the right channel are grouped and processed together.

15 January 2004
Electronics Letters online no: 20040315
doi: 10.1049/el:20040315

H. Huang, R. Yu, X. Lin and S. Rahardja (*Agency for Science, Technology and Research (A*STAR), Institute for Infocomm Research (I^2R), 21 Heng Mui Keng Terrace, 119613 Singapore*)

E-mail: rsusanto@i2r.a-star.edu.sg

References

1 'Call for proposals on MPEG-4 lossless audio coding' (Shanghai, October 2002) (ISO/IEC JTC1/SC29/WG11)
2 Malvar, H.S.: 'Signal processing with lapped transforms' (Artech House, Norwood, MA, 1992), pp. 199–201

3 Daubechies, I., and Sweldens, W.: 'Factoring wavelet transforms into lifting steps', *Tech. Rep.* (Bell Laboratories, Lucent Technologies, 1996)

4 Sweldens, W.: 'Coding of moving pictures and audio: work plan for evaluation of integer MDCT for FGS to lossless experimentation framework' (Pattaya, Thailand, March 2003) (ISO/IEC JTC 1/SC 29/WG 11 N5578)

2

Publication P2

H. Huang, S. Rahardja, R. Yu, X. Lin, Integer MDCT with Enhanced Approximation of the DCT-IV, *IEEE Trans. on Signal Processing*, Vol 54, Number 3, pp. 1156-1159, March 2006.

Correspondence

Integer MDCT With Enhanced Approximation of the DCT-IV

Haibin Huang, *Member, IEEE*,
Susanto Rahardja, *Senior Member, IEEE*,
Rongshan Yu, *Member, IEEE*, and Xiao Lin, *Senior Member, IEEE*

***Abstract*—Integer transform plays an important role in lossless signal compression. In order to achieve high accuracy to its corresponding theoretical transform, the rounding number should be as low as possible. At the same time, the matrix factorization for reversible integer mapping should be handled with care, especially when the processing block length is high $(N > 16)$. In this correspondence, a new method for realizing reversible integer discrete cosine transform type IV (DCT-IV) is proposed that is a key component used in audio compression. The proposed method exhibits low rounding number $(2.5N)$ and low complexity $O(N \log_2 N)$ as well as well as accurately represents its counterpart floating-point DCT-IV transformation.**

***Index Terms*—Discrete cosine transform, fast transforms, integer transforms, modified discrete cosine transform.**

I. Introduction

Reversible transforms that map integers to integers are important tools for lossless coding schemes. For integer inputs, such a transform generates integer outputs that approximate its floating-point prototype transform outputs. With perfect reconstruction property, the original integer inputs can be completely recovered by passing the integer outputs through the inverse transform.

Recently, integer transform has attracted a lot of attentions for applications in image and audio coding [1]–[7]. With the applications that are currently in high demand, there is a need to have a fast and efficient algorithm implementation for such integer transform. Particularly for high-order $N(N > 16)$, the fast implementation still poses a challenge in research.

In audio coding application, the modified discrete cosine transform (MDCT) is widely used. The MDCT is a linear modulated lapped transform (MLT) which is based on the concept of time-domain aliasing cancellation (TDAC). It is known that the MDCT can provide critical sampling with good frequency selectivity. With that important property, it is employed in MPEG-4 advanced audio coding (AAC) system [8].

Recently, MPEG issued a call for proposal on lossless audio coding for its extension 3. An integer transform derived from the MDCT is used which is known as integer MDCT (IntMDCT). This integer transform offers an excellent representation of audio signal and retains the important properties of the MDCT. In addition, the IntMDCT provides a unique integer to integer mapping and with all the above, it is a perfect candidate for audio lossless coding. Very recently, IntMDCT has been identified as one of the key components in audio lossless coding scheme [9]. It is then critical that a fast IntMDCT algorithm with low approximation error be developed and utilized to meet the industry requirements.

Manuscript received November 11, 2003; revised March 8, 2005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anamitra Makur.

The authors are with the Institute for Infocomm Research (I^2R), Singapore 119613, Singapore (e-mail: hhuang@i2r.a-star.edu.sg; rsusanto@i2r.a-star.edu.sg; rsyu@i2r.a-star.edu.sg; linxiao@i2r.a-star.edu.sg).

Digital Object Identifier 10.1109/TSP.2005.862942

As mentioned, the IntMDCT is derived from its prototype—the MDCT. In his book [10], Malvar gave an efficient realization of MDCT by cascading a bank of Givens rotations with a integer discrete cosine transform type IV (DCT-IV) block. It is well known that Givens rotation can be factorized into three lifting steps for mapping integers to integers [3]. With that, the realization of IntMDCT mainly relies on an efficient implementation of integer DCT-IV.

Integer transforms can be directly converted from their prototypes by replacing each Givens rotation with three lifting steps. For each lifting step there is one rounding operation, the total rounding number of an integer transform is three times the Givens rotation number of the prototype transform. For discrete trigonometric transforms, the number of Givens rotations needed is usually at $O(N \log_2 N)$, where N is the block size. Accordingly, the total rounding number is also at $O(N \log_2 N)$ for the family of directly converted integer transforms.

Because of the rounding, an integer transform can only approximate its floating-point prototype. Obviously, the approximation error increases with the number of rounding operations. In addition, low rounding number still does not guarantee low approximation error especially when matrix factorization is not well conditioned, which is usually the case when N is higher than 16. Thus, an IntMDCT with low rounding number and well-conditioned matrix factorization as well as low complexity is the motivation of this work which this correspondence addresses.

In this correspondence, a novel fast algorithm for realizing reversible integer DCT-IV is proposed. The fast algorithm has immediate application in improving the performance of a lossless audio coding system. The total rounding number of this algorithm is only $2.5N$ while the approximation error is shown to be far less than that of the directly converted integer transforms, which provides a solution when N is high $(N > 16)$. As a result, the lossless compression ratio of a lossless audio coding system employing the fast algorithm improves and at same time the computational complexity is reduced.

This paper is organized as follows. Section II describes the new algorithm. Section III presents discussions and numerical results on performance comparisons. Finally, in Section IV, we conclude the work.

II. Integer Type-IV DCT

The DCT-IV of a N-point real input sequence $x(n)$ is defined as follows [11]:

$$\mathbf{y} = \mathbf{C}_N^{\mathrm{IV}} \mathbf{x} \tag{1}$$

where $\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^{\mathrm{T}}$, $\mathbf{y} = [y(0), y(1), \ldots, y(N-1)]^{\mathrm{T}}$, and the $N \times N$ DCT-IV transform matrix $\mathbf{C}_N^{\mathrm{IV}}$ is given by

$$\mathbf{C}_N^{\mathrm{IV}} = \sqrt{\frac{2}{N}} \left[\cos\left(\frac{\left(m + \frac{1}{2}\right)\left(n + \frac{1}{2}\right)\pi}{N} \right) \right] \quad m = 0, 1, \cdots, N-1, \quad n = 0, 1, \cdots, N-1. \tag{2}$$

As the derivation of the decimation-in-time radix-2 fast Fourier transform (FFT) [12], we can factorize the DCT-IV matrix $\mathbf{C}_N^{\mathrm{IV}}$ as follows:

$$\mathbf{C}_N^{\mathrm{IV}} = \mathbf{R}_{\mathrm{po}} \begin{bmatrix} \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} & 0 \\ 0 & \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} \mathbf{D}_{\frac{N}{2}} \end{bmatrix} \mathbf{R}_{\mathrm{pr}} \mathbf{P} \tag{3}$$

where permutation matrix $\mathbf{P}$ reorders the components of vector $\mathbf{x}$ by separating the components which correspond to even indexes from the components which correspond to odd indices, such that

$$\begin{bmatrix} x(0) \\ x(2) \\ \vdots \\ x(N-2) \\ x(1) \\ x(3) \\ \vdots \\ x(N-1) \end{bmatrix} = \mathbf{P} \begin{bmatrix} x(0) \\ \vdots \\ x(N-1) \end{bmatrix}. \tag{4}$$

Matrix $\mathbf{R}_{\mathrm{pr}}$ is given by

$$\mathbf{R}_{\mathrm{pr}} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{I}_{\frac{N}{2}} \\ \mathbf{I}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix} \tag{5}$$

where $\mathbf{I}_{N/2}$ is the identity matrix of order $N/2$. $\mathbf{C}^{\mathrm{IV}}_{N/2}$ is the DCT-IV matrix of order $N/2$ and $\mathbf{D}_{N/2}$ is a diagonal matrix of order $N/2$

$$\mathbf{D}_{\frac{N}{2}} = \begin{bmatrix} 1 & & & & \\ & -1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & -1 \end{bmatrix}. \tag{6}$$

Let

$$c_n = \cos\left(\frac{n\pi}{4N}\right) \quad s_n = \sin\left(\frac{n\pi}{4N}\right) \quad n = 0, 1, \cdots, N-1 \tag{7}$$

matrix $\mathbf{R}_{\mathrm{po}}$ is represented as

$$\mathbf{R}_{\mathrm{po}} = \begin{bmatrix} c_1 & & & & & & & & s_1 \\ & c_3 & & & & & & s_3 & \\ & & \ddots & & & & \iddots & & \\ & & & c_{N-1} & -s_{N-1} & & & & \\ & & & s_{N-1} & c_{N-1} & & & & \\ & & \iddots & & & & \ddots & & \\ & -s_3 & & & & & & c_3 & \\ -s_1 & & & & & & & & c_1 \end{bmatrix}. \tag{8}$$

Combining the second and third matrices in the right-hand side of (3) into matrix $\mathbf{T}$

$$\mathbf{T} = \begin{bmatrix} \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} & 0 \\ 0 & \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \mathbf{D}_{\frac{N}{2}} \end{bmatrix} \mathbf{R}_{\mathrm{pr}} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} & \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \\ \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \mathbf{D}_{\frac{N}{2}} & -\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \mathbf{D}_{\frac{N}{2}} \end{bmatrix} \tag{9}$$

we can simplify (3) to

$$\mathbf{C}^{\mathrm{IV}}_{N} = \mathbf{R}_{\mathrm{po}} \mathbf{T} \mathbf{P}. \tag{10}$$

We further factorize $\mathbf{R}_{\mathrm{po}}$ and $\mathbf{T}$ as product of matrices. Matrix $\mathbf{T}$ is factorized in the following way:

$$\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3 = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{K}_1 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} -\mathbf{D}_{\frac{N}{2}} & \mathbf{K}_2 \\ 0 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{K}_3 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \tag{11}$$

where

$$\mathbf{K}_1 = -\left(\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \mathbf{D}_{\frac{N}{2}} + \sqrt{2}\mathbf{I}_{\frac{N}{2}}\right) \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \tag{12}$$

$$\mathbf{K}_2 = \frac{\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}}}{\sqrt{2}} \tag{13}$$

$$\mathbf{K}_3 = \sqrt{2}\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \mathbf{D}_{\frac{N}{2}} + \mathbf{I}_{\frac{N}{2}}. \tag{14}$$

Matrix $\mathbf{R}_{\mathrm{po}}$ is factorized as follows:

$$\mathbf{R}_{\mathrm{po}} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3 = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{H}_1 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{H}_2 \\ 0 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{H}_3 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \tag{15}$$

where

$$\mathbf{H}_1 = \mathbf{H}_3 = \begin{bmatrix} & & & & -\frac{s_{N-1}}{c_{N-1}} \\ & & & \iddots & \\ & & -\frac{s_3}{c_3} & & \\ -\frac{s_1}{c_1} & & & & \end{bmatrix} \tag{16}$$

$$\mathbf{H}_2 = \begin{bmatrix} & & & & s_1 \\ & & & s_3 & \\ & & \iddots & & \\ s_{N-1} & & & & \end{bmatrix}. \tag{17}$$

Now (10) can be expressed as

$$\mathbf{C}^{\mathrm{IV}}_{N} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3 \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3 \mathbf{P}. \tag{18}$$

Combing $\mathbf{R}_3$ and $\mathbf{T}_1$ by defining matrix $\mathbf{S}$

$$\mathbf{S} = \mathbf{R}_3 \mathbf{T}_1 = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{H}_3 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{K}_1 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & 0 \\ \mathbf{H}_3 + \mathbf{K}_1 & \mathbf{I}_{\frac{N}{2}} \end{bmatrix} \tag{19}$$

we obtain the following factorization for the DCT-IV matrix:

$$\mathbf{C}^{\mathrm{IV}}_{N} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{S} \mathbf{T}_2 \mathbf{T}_3 \mathbf{P}. \tag{20}$$

In the above factorization, we call the nontrivial submatrices $\mathbf{H}_1$, $\mathbf{H}_2$, $\mathbf{H}_3 + \mathbf{K}_1$, $\mathbf{K}_2$, and $\mathbf{K}_3$ in $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{S}$, $\mathbf{T}_2$, and $\mathbf{T}_3$, respectively, as critical submatrices because these submatrices determine the rounding error performance of the integer transform. We find that all these critical submatrices are well conditioned in the sense that rounding error magnification will not occur or only to a very limited degree with these matrices. In integer transforms, rounding error magnification will occur when the maximum norm of the row vectors in the critical submatrices exceeds 1. Hence, we can define this maximum norm as a condition index for indicating the degree of rounding error magnification associated with the critical submatrix. The condition index ξ of a $N \times N$ matrix $\mathbf{A}$ is defined as follows:

$$\xi(\mathbf{A}) = \max\left(\|\mathbf{a}_i\|\right) \quad i \in [0, N-1] \tag{21}$$

where $\mathbf{a}_i$ is the ith row vector of matrix $\mathbf{A}$, and $\|\mathbf{a}_i\|$ is the Euclidean norm of vector $\mathbf{a}_i$

$$\|\mathbf{a}_i\| = \sqrt{\sum_{j=0}^{N-1} a_{i,j}^2} \quad i = 0, 1, \ldots, N-1 \tag{22}$$

and $a_{i,j}$ is the element of matrix $\mathbf{A}$ at the ith row and jth column.

It is found that the condition index ξ provides a good indication of rounding error magnification as well as the final approximation error of the integer transform algorithm. The larger the condition index ξ is, the more the rounding error will be magnified. When ξ falls below 1, no rounding error magnification will occur.

The condition indexes of submatrices $\mathbf{H}_1$, $\mathbf{H}_2$, $\mathbf{H}_3 + \mathbf{K}_1$, $\mathbf{K}_2$, and $\mathbf{K}_3$ are, respectively

$$\begin{aligned} \xi(\mathbf{H}_1) &= \tan\frac{(N-1)\pi}{8N} < \tan\frac{\pi}{8} = 0.414 \\ \xi(\mathbf{H}_2) &= \sin\frac{(N-1)\pi}{4N} < \sin\frac{\pi}{4} = 0.707 \\ \xi(\mathbf{H}_3 + \mathbf{K}_1) &< 2.2 \quad N \geq 16 \\ \xi(\mathbf{K}_2) &= 0.707 \\ \xi(\mathbf{K}_3) &< 2 \quad N \geq 16. \end{aligned}$$

Therefore, all critical submatrices $\mathbf{H}_1$, $\mathbf{H}_2$, $\mathbf{H}_3+\mathbf{K}_1$, $\mathbf{K}_2$, and $\mathbf{K}_3$ are well conditioned in the proposed integer transform.

From (20), we can realize integer DCT-IV transform with five lifting stages. As these lifting stages have identical structures, we only need to provide the transform equations for one stage here. It is straightforward to write down the transform equations for other stages.

Let $\mathbf{x}=[x(0),x(1),\ldots,x(N-1)]^{\mathrm{T}}$, $y=[y(0),y(1),\ldots,y(N-1)]^{\mathrm{T}}$ be the input and output integer vectors of the first transform stage in (20), respectively. We have $\mathbf{y}=\mathbf{T}_3\mathbf{x}$. Partitioning vectors $\mathbf{x}$ and $\mathbf{y}$ into two halves, we rewrite the equation as

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}=\begin{bmatrix}\mathbf{I}_{\frac{N}{2}} & 0\\ \mathbf{K}_3 & \mathbf{I}_{\frac{N}{2}}\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}. \tag{23}$$

From (23), we obtain the forward integer transform, which maps $\mathbf{x}$ to $\mathbf{y}$

$$\begin{aligned}\mathbf{y_1}&=\mathbf{x_1}\\ \mathbf{y_2}&=\mathrm{round}(\mathbf{K_3x_1})+\mathbf{x}_2,\end{aligned} \tag{24}$$

and the corresponding inverse integer transform which reconstructs $\mathbf{x}$ from $\mathbf{y}$:

$$\begin{aligned}\mathbf{x_1}&=\mathbf{y_1}\\ \mathbf{x}_2&=\mathbf{y_2}-\mathrm{round}(\mathbf{K_3x_1})\end{aligned} \tag{25}$$

where round$(*)$ is the rounding to integer operator, which could be, but is not limited to, rounding to the nearest floor and ceiling. Clearly, perfect reconstruction does not depend on the rounding operator.

III. Discussion and Experimental Results

In (24) and (25), there are $N/2$ rounding operations in each lifting stage. Therefore, from (20), the total number of rounding operations for the proposed integer DCT-IV algorithm is five times $N/2$, or $2.5N$. From (20), it can be seen that the majority of computation is on the four $N/2$ point DCT-IV subroutines $\mathbf{C}^{\mathrm{IV}}_{N/2}$, when N is a large value, e.g., $N=1024$. It can be roughly estimated that the complexity of the proposed integer DCT-IV be twice that of the floating-point DCT-IV.

From the proposed Integer DCT-IV algorithm above, a new IntMDCT algorithm can be derived using the MLT algorithm in [10] and the IntMDCT algorithm in [3]. The proposed IntMDCT algorithm has two steps: 1) integer MDCT windowing operation and 2) integer DCT-IV transformation. In Step 1, the integer MDCT windowing operation is implemented by using three lifting steps for each Givens rotation. (Readers can refer to [3] for implementation details.) In Step 2, the integer DCT-IV transformation is implemented using our proposed integer DCT-IV algorithm.

The proposed IntMDCT algorithm is compared with the IntMDCT implementation used in the MPEG-4 lossless audio codec [13], which was adopted as reference model 0 (RM0) in July 2003. Fig. 1(a) and (b) shows the block diagrams for performance evaluation of the IntMDCT and integer inverse MDCT (IntIMDCT) transforms, respectively. Mean-squared error (MSE) is computed and the MPEG-4 lossless audio coding evaluation procedure is applied where the results are compared with the direct implementation method used in the MPEG-4 RM0, which is similar to [3]. The MSE for IntMDCT and IntIMDCT is defined as

$$\mathrm{MSE}=\frac{1}{L}\sum_{j=0}^{L-1}\frac{1}{N}\sum_{i=0}^{N-1}e_i^2 \tag{26}$$

where e is e_f for IntMDCT, and e_t for IntIMDCT. N is the DCT-IV block size, which is set to 1024 throughout the evaluation. L is the number of data blocks used in the evaluation.

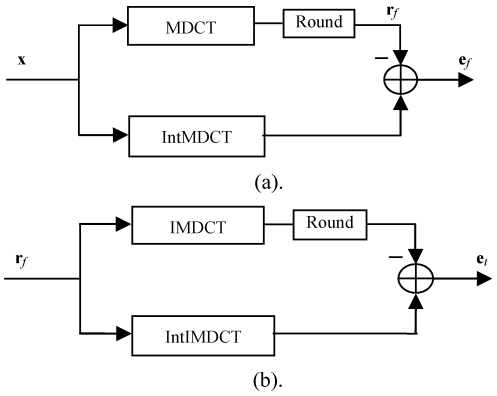

Fig. 1. Performance evaluation diagram for IntMDCT and IntIMDCT.

TABLE I
MSE Performance Comparison Under Different Block Lengths for the Proposed IntMDCT/IntIMDCT Algorithms and Those in MPEG-4 RM0

N	IntMDCT		IntIMDCT	
	RM0	Proposed	RM0	Proposed
16	0.802	0.538	0.803	0.538
32	0.964	0.542	0.964	0.541
64	1.131	0.543	1.132	0.545
128	1.295	0.545	1.296	0.544
256	1.460	0.546	1.462	0.545
512	1.625	0.547	1.628	0.545
1024	1.789	0.547	1.774	0.545
2048	1.954	0.546	1.960	0.545
4096	2.121	0.546	2.126	0.545

Table I shows a comparison of the proposed IntMDCT/IntIMDCT algorithms with those used in MPEG-4 RM0. Table II shows the performance comparison when the proposed algorithms are integrated with RM0 of the scalable lossless coding system.

From Table I, it is obvious that the MSE of the proposed method is lower than that of direct implementation method in MPEG-4 RM0. It can be seen that this MSE gap increases with the increment of the block length. It is found that for $N>16$, the MSE of the new algorithm does not depend on the block length of the transform. From Table II, it is further shown that the new algorithm demonstrates superior performance compared with the algorithm employed in RM0. The RM0 of MPEG-4 lossless audio coding system is shown in Fig. 2, which is the advanced audio zip (AAZ) technology adopted by MPEG recently [13].

In the AAZ encoder, the input audio frames are transformed losslessly by using the IntMDCT to generate the transform coefficients. These coefficients are scaled, quantized, and coded with perceptual weighting so that the noise introduced is best masked by the masking threshold of the human auditory system in the core AAC encoder. The resultant core-layer bit stream thus constitutes the minimum rate/quality unit of the final lossless bit stream. For optimal coding efficiency, an error-mapping procedure is employed to remove the information that has been coded in the core layer from the IntMDCT coefficients before they are noiselessly coded by bit-plane Golomb code (BPGC) [14] to form lossless bit stream. In the decoder, a reverse process is shown in the Fig. 2(b).

According to MPEG audio test requirement, 450 s with 15 different types of music files are used for each 48-kHz/16-b and 96-kHz/24-b test sets. It can be seen that the total compression ratios for both sequences

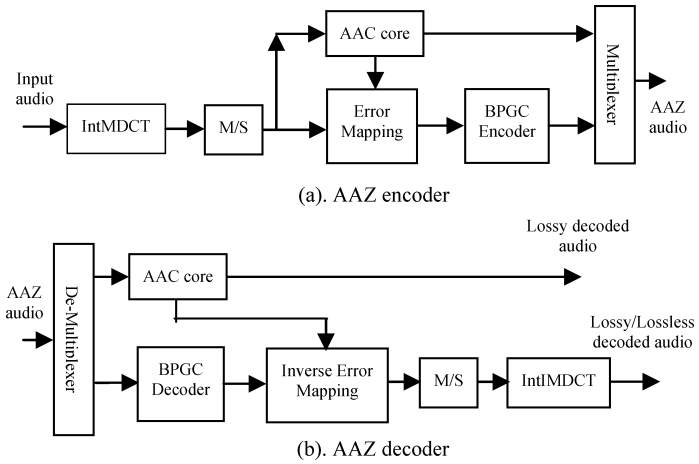

Fig. 2. AAZ scalable lossless audio encoder/decoder system diagram.

TABLE II
COMPRESSION PERFORMANCE COMPARISON WHEN INTEGRATED WITH LOSSLESS CODING SYSTEM FOR THE PROPOSED IntMDCT/IntIMDCT ALGORITHMS AND THOSE IN MPEG-4 RM0

	MSE		Compression Ratio Improvement
	RM0	Proposed	
48k/16bit test set			
IntMDCT	1.789	0.547	0.87%
IntIMDCT	1.774	0.545	
96k/24bit test set			
IntMDCT	1.785	0.546	0.42%
IntIMDCT	1.766	0.546	
Complexity[1]	47,616	34,816	

Count at MAC (multiply and accumulate with rounding, which is usually used in most DSPs).

are improved, and this is achieved without introducing any amendment from the original entropy coder and simultaneously the complexity has been reduced for more than 26%.

IV. CONCLUSION

In this paper, a new fast algorithm for realizing reversible integer type-IV DCT is proposed. This algorithm requires only $2.5N$ rounding operations for every block of N input samples. As a result, the approximation error is greatly reduced. The proposed algorithm is low in computational complexity and modular in structure. This algorithm is integrated and evaluated with MPEG-4 audio extension 3 lossless audio compression reference model 0. From the results, it is clear that the new algorithm contributes to an overall improvement of compression ratio computational complexity. The proposed method is also useful for applications requiring high-order integer DCT-IV transformation.

REFERENCES

[1] K. Komatsu and K. Sezaki, "Design of lossless LOT and its performance evaluation," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2000, pp. 2119–2122.

[2] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, "Integer fast Fourier transform (INTFFT)," *IEEE Trans. Signal Process.*, vol. 50, no. 3, pp. 607–618, Mar. 2002.

[3] R. Geiger, T. Sporer, J. Koller, and K. Brandenburg, "Audio coding based on integer transforms," presented at the 111th Audio Engineering Soc. Convention, New York, NY, 2001.

[4] R. Geiger, J. Herre, J. Koller, and K. Brandenburg, "INTMDCT—a link between perceptual and lossless audio coding," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, Orlando, 2002, pp. 1813–1816.

[5] J. Li, "A progressive to lossless embedded audio coder (PLEAC) with reversible modulated lapped transform," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, vol. III, 2003, pp. 221–224.

[6] J. Wang, J. Sun, and S. Yu, "1-D and 2-D transforms from integers to integers," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, vol. II, 2003, pp. 549–552.

[7] P. Hao and Q. Shi, "Matrix factorizations for reversible integer mapping," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2314–2324, Oct. 2001.

[8] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information, Part 7: Advanced Audio Coding (AAC)*, International Standard ISO/IEC 13 818-7, 1998.

[9] *Call for Proposals on MPEG-4 Lossless Audio Coding*, ISO/IEC JTC1/SC29/WG11, Oct. 2002.

[10] H. S. Malvar, *Signal Processing With Lapped Transforms*. Norwood, MA: Artech House, 1992, pp. 199–201.

[11] Z. Wang, "Fast algorithms for the discrete $\boldsymbol{w}$ transform and for the discrete Fourier transform," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-32, no. 4, pp. 803–816, Aug. 1984.

[12] J. G. Proakis, *Digital Signal Processing, Principles, Algorithms, and Applications*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996, pp. 456–457.

[13] *Advanced Audio Zip—A Scalable Perceptual and Lossless Audio Codec*, ISO/IEC JTC1/SC29/WG11 (m9134) MPEG2002 Awaji, Dec. 2002.

[14] R. S. Yu, C. C. Ko, S. Rahardja, and X. Lin, "Bit-plane Golomb coding for sources with Laplacian distribution," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, vol. IV, 2003, pp. 277–280.

3

Publication P3

H. Huang, X. Lin, S. Rahardja, R. Yu, Integer DFT with High Transform Accuracy, *Proc. 4th Int. Conf. on Information, Communications and Signal Processing (ICICS 2005)*, pp .1471-1474, Bangkok, Thailand, December 6-9, 2005.

Integer DFT with High Transform Accuracy

Huang Haibin, Lin Xiao, Rahardja Susanto, Yu Rongshan
Agency for Science, Technology and Research (A*STAR)
Institute for Infocomm Research
Singapore
hhuang@i2r.a-star.edu.sg

***Abstract*— An integer DFT (IntDFT) algorithm with high transform accuracy is presented. The algorithm is derived by firstly constructing an auxiliary block matrix using the normalized DFT matrix, then factorizing the auxiliary matrix into three lifting matrices. The proposed integer DFT algorithm exhibits low rounding number $3N$ (N is the transform block length), low complexity, and gives a highly accurate approximation of the floating-point DFT.**

***Keywords*—DFT, IntDFT, Integer Transfrom**

I. Introduction

Reversible integer transforms are important tools for signal lossless compression. Such a transform generates integer outputs for integer inputs. A corresponding inverse integer transform can perfectly reconstruct the original integer inputs.

Recently, we proposed two integer algorithms of type-IV discrete cosine transform (DCT-IV) [1][2], which have important applications in the current MPEG-4 audio lossless coding frame work [3]. In [1], we factorized the DCT-IV matrix into five lifting matrices, and performed integer to integer mapping in vector form. In [2] we constructed an auxiliary matrix using the DCT-IV matrix, and factorized the auxiliary into three lifting matrices. Both algorithms exhibit high and constant transform accuracy. In this paper, we apply the method of [2] to another important member of the discrete trigonometric transforms family - the discrete Fourier transform (DFT).

In literature, the number of works on integer DFT is quite limited. An integer DFT algorithm was developed from the split-radix FFT algorithm by substituting each Givens rotation with three 2×2 lifting steps [4]. This algorithm has rounding number $O(N \log_2 N)$, which leads to rapidly increased approximation error with the increment of the block length N [5][6]. Thus it is not suitable for applications requiring large N, e.g., audio compression. A LU-type matrix factorization method was provided in [7], which could be used to develop integer DFT algorithms. However this method requires exhaustive-search for the optimal permutation, which poses a prohibitive computational cost for large block length, e.g., $N = 1024$. Therefore, developing highly accurate integer DFT algorithms for large block length is still a challenging work.

In this paper, we propose an integer DFT algorithm which has high transform accuracy. The algorithm is derived by constructing an auxiliary matrix using the normalized DFT matrix, and then factorizing the auxiliary matrix into three lifting matrices. Similar to the integer DCT-IV algorithms in [1][2], the proposed integer DFT algorithm exhibits the following merits:

1) Rounding number at only $3N$, thus very low approximation error to the floating-point DFT;
2) Constant transform accuracy regardless of block length - suitable for both short and long blocks;
3) Low computational complexity - about 1.5 times that of the floating-point DFT;
4) Simple, modular structure ideal for real-world implementation.

This paper is organized in the following way. In Section II, we describe the new algorithm. Section III gives the computational complexity of the algorithm. Then the rounding error is analyzed in Section IV. We provide numerical results and discussions in Section V. Finally, Section VI summarizes the paper.

II. Integer DFT

The normalized DFT of an N-point discrete-time signal $x(n)$ is defined by

$$X(m) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) W_N^{mn} \qquad (1)$$
$$m, n = 0, 1, ..., N-1$$

where $W_N = e^{\frac{-j2\pi}{N}}$. In matrix-vector form, it is expressed as

$$\mathbf{X} = \mathbf{F}\mathbf{x} \qquad (2)$$

where

$$\mathbf{x} = [x(0), x(1), ..., x(N-1)]^T$$
$$\mathbf{X} = [X(0), X(1), ..., X(N-1)]^T$$

and $\mathbf{F}$ is the $N \times N$ normalized DFT matrix with elements $\mathbf{F}_{m,n} = \frac{W_N^{mn}}{\sqrt{N}}$ [8].

Assume we have $N \times 1$ input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Their normalizd DFT transforms are denoted as $\mathbf{X}_1$ and $\mathbf{X}_2$, respectively, where

$$\mathbf{X}_1 = \mathbf{F}\mathbf{x_1} \qquad (3)$$
$$\mathbf{X}_2 = \mathbf{F}\mathbf{x}_2 \ . \qquad (4)$$

Combining (3) and (4), we have

$$\begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{F} \\ \mathbf{F} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{x}_1 \end{bmatrix} . \qquad (5)$$

0-7803-9282-5/05/$20.00 ©2005 IEEE

Define an auxiliary matrix $\mathbf{T}$ as

$$\mathbf{T} = \begin{bmatrix} 0 & \mathbf{F} \\ \mathbf{F} & 0 \end{bmatrix} \tag{6}$$

which can be factroized as follows:

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{QF} & \mathbf{I} \end{bmatrix} \begin{bmatrix} -\mathbf{Q} & \mathbf{F} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{F} & \mathbf{I} \end{bmatrix} \tag{7}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix, $\mathbf{Q}$ is an $N \times N$ permutation matrix (which only reorders the elements of a vector) given by

$$\mathbf{Q} = \begin{bmatrix} 1 & \mathbf{0}_{N-1}^T \\ \mathbf{0}_{N-1} & \mathbf{J}_{N-1} \end{bmatrix}. \tag{8}$$

In (8), $\mathbf{0}_{N-1}$ is a column vector of $N-1$ zeros. $\mathbf{J}_{N-1}$ is the $(N-1) \times (N-1)$ counter identity matrix

$$\mathbf{J}_{N-1} = \begin{bmatrix} & & 1 \\ & 1 & \\ \cdots & & \\ 1 & & \end{bmatrix}. \tag{9}$$

The three matrices on the right hand side of (7) take the following forms:

$$\begin{bmatrix} \mathbf{P}_1 & \mathbf{A} \\ 0 & \mathbf{P}_2 \end{bmatrix}$$

or

$$\begin{bmatrix} \mathbf{P}_1 & 0 \\ \mathbf{A} & \mathbf{P}_2 \end{bmatrix}$$

where the two diagonal sub-matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ are $N \times N$ permutation matrices. Sub-matrix $\mathbf{A}$ is an arbitrary $N \times N$ matrix. The fourth sub-matrix contains only zeros. We call such matrices lifting matrices due to their analog to the well-known 2×2 lifting step [9]. With a lifting matrix, we can map integers to integers in vector form similar to the 2×2 lifting step. However, the rounding operation is applied each time to a vector instead of to a scalar.

Using (7), we can simultaneously compute integer normalized DFT for vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in the following three steps:

$$\mathbf{z} = \lfloor \mathbf{F}\mathbf{x}_2 \rfloor + \mathbf{x}_1 \tag{10}$$

$$\hat{\mathbf{X}}_1 = \lfloor \mathbf{F}\mathbf{z} \rfloor - \mathbf{Q}\mathbf{x}_2 \tag{11}$$

$$\hat{\mathbf{X}}_2 = -\lfloor \mathbf{QF}\mathbf{X}_1 \rfloor + \mathbf{z} \tag{12}$$

where $\mathbf{z}$ is an intermediate $N \times 1$ vector, $\hat{\mathbf{X}}_1$ and $\hat{\mathbf{X}}_2$ are the integer normalized DFT of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. In (10)-(12), $\lfloor * \rfloor$ denotes rounding of a vector. The rounding of a complex number means rounding to both its real and imaginary parts. For convenience, We refer to steps (10)-(12) as the forward integer DFT algorithm. Figure 1 gives the block diagram of the forward integer DFT algorithm.

Accordingly, we can obtain the integer inverse normalized DFT (IntIDFT) algorithm by simply reversing the signal flow (horizontal direction only) in Figure 1, and changing the +/- signs. The integer inverse DFT is given by

$$\mathbf{z} = \lfloor \mathbf{QF}\mathbf{X}_1 \rfloor + \hat{\mathbf{X}}_2 \tag{13}$$

$$\mathbf{x}_2 = \mathbf{Q}^T(\lfloor \mathbf{F}\mathbf{z} \rfloor - \hat{\mathbf{X}}_1) \tag{14}$$

$$\mathbf{x}_1 = -\lfloor \mathbf{F}\mathbf{x}_2 \rfloor + \mathbf{z}. \tag{15}$$

Figure 2 shows the block diagram of the IntIDFT.

III. Complexity

Let $\alpha(*)$ be the number of real additions, $\mu(*)$ be the number of real multiplications, and $\gamma(*)$ be the number of real roundings, respectively. From (10)-(12), we can calculate the complexity of the proposed IntDFT algorithm, which is given by

$$\alpha(\text{IntDFT}) = 6N + 3\alpha(\text{DFT}) \tag{16}$$

$$\mu(\text{IntDFT}) = 3\mu(\text{DFT}) \tag{17}$$

$$\gamma(\text{IntDFT}) = 6N. \tag{18}$$

It should be noted that results in (16)-(18) are the total complexity for processing two blocks of complex data, because the proposed algorithm computes integer DFT for two data blocks together. Thus for one block of data, the complexity is halved, which is given by

$$\alpha_1(\text{IntDFT}) = 3N + 1.5\alpha(\text{DFT}) \tag{19}$$

$$\mu_1(\text{IntDFT}) = 1.5\mu(\text{DFT}) \tag{20}$$

$$\gamma_1(\text{IntDFT}) = 3N \tag{21}$$

where α_1, μ_1, and γ_1 are the number of real additions, number of real multiplications, and number of real roundings, for one block of data, respectively.

From (19) and (20) we find that the complexity of integer DFT is about 1.5 times that of DFT. Therefore, the complexity of integer DFT depends on the DFT routine used. For DFT computation, if the split-radix FFT (SRFFT) algorithm [10] is used which has complexity

$$\alpha(\text{SRFFT}) = 3N\log_2 N - 3N \tag{22}$$

$$\mu(\text{SRFFT}) = N\log_2 N - 3N \tag{23}$$

the complexity of the proposed integer DFT algorithm for one block of data would be

$$\alpha_1(\text{IntDFT}) = 4.5N\log_2 N - 1.5N \tag{24}$$

$$\mu_1(\text{IntDFT}) = 1.5N\log_2 N - 4.5N. \tag{25}$$

IV. Rounding Error Analysis

Now, we analyze the rounding error for the proposed integer DFT. Let $\mathbf{\Delta}_1$, $\mathbf{\Delta}_2$, and $\mathbf{\Delta}_3$ be the $N \times 1$ complex rounding error vectors generated by the rounding operation $\lfloor * \rfloor$ in steps (10)-(12), respectively. Let $\boldsymbol{\epsilon}$ be the $2N \times 1$ rounding error vector at the output of step (12). From (7), we can derive the following equation

$$\boldsymbol{\epsilon} = \begin{bmatrix} \mathbf{F}\mathbf{\Delta}_1 + \mathbf{\Delta}_2 \\ -\mathbf{QF}\mathbf{\Delta}_2 + \mathbf{\Delta}_3 \end{bmatrix} \tag{26}$$

Without loss of generality, we assume that vectors $\mathbf{\Delta}_1$, $\mathbf{\Delta}_2$, and $\mathbf{\Delta}_3$ consist of independent, identically-distributed random variables with zero mean and variance σ^2 . Let ϵ_i be the i

th element of $\boldsymbol{\epsilon}$, by using properties of the normalized DFT matrix, we can derive the variance of ϵ_i as

$$E(\|\epsilon_i\|^2) = 3\sigma^2 \qquad 0 \le i \le 2N-1 \tag{27}$$

where $E(*)$ is the expectation operator, and $\|*\|$ represents the norm of a complex number.

From (27) we find that the rounding error of the proposed integer DFT exhibits a constant level of power, which is independent of the transform size N. If rounding-to-the-nearest-integer is used as the rounding operator in steps (10)-(12), the elements of vectors $\boldsymbol{\Delta}_1$, $\boldsymbol{\Delta}_2$, and $\boldsymbol{\Delta}_3$ will have uniform distribution between $-1/2$ to $1/2$ and $\sigma^2 = 1/12$. In this case, the integer DFT has rounding error which has zero-mean and variance $1/4$.

V. Numerical Results

The accuracy of IntDFT is indicated by the difference between its output to that of the floating-point DFT. Figure 3 and 4 illustrate the approximation error measurement systems. These systems were proposed by MPEG-4 lossless audio coding group during evaluation process for a new lossless audio coding standard [3]. In Figure 3 the forward approximation error e_f is calculated as the difference between the outputs of IntDFT and floating-point DFT. The inverse approximation error e_t is generated in a similar way as shown in Figure 4. In Table I, we list the transform accuracy of the proposed IntDFT algorithm in terms of the mean-squared-error (MSE).

The MSEs for IntDFT and IntIDFT are calculated by

$$MSE = \frac{1}{KN} \sum_{j=0}^{K-1} \sum_{i=0}^{N-1} \|e_i\|^2 \tag{28}$$

where e is the approximation error. For IntDFT $e = e_f$, for IntIDFT $e = e_t$. K denotes the total number of data blocks used in the evaluation. The evaluation uses the MPEG-4 audio 450-seconds 48kHz/16-bit test set which contains 15 different types of music files. Table I clearly shows that the proposed IntDFT/IntIDFT algorithms have very low, constant level of MSE regardless of the changes in transform block length N.

TABLE I
MSE of IntDFT/IntIDFT

N	IntDFT	IntIDFT
8	0.456	0.317
16	0.480	0.412
32	0.461	0.391
64	0.462	0.393
128	0.461	0.391
256	0.461	0.391
512	0.461	0.391
1024	0.460	0.391
2048	0.461	0.391
4096	0.461	0.391

The proposed integer DFT algorithm can be used in transform based lossless coders. For example, stereo audio signals can be transformed into the frequency domain by the IntDFT algorithm, then entropy coded. The IntDFT algorithm can also be used as a building block of other integer transforms, e.g., integer modified discrete cosine transform (IntMDCT), which is an important transform in audio signal processing.

VI. Conclusion

We propose an integer DFT algorithm with highly accurate approximation of the floating DFT. The proposed algorithm uses $3N$ real roundings for each block of N complex data samples. As a result, it has very low approximation error to the floating-point DFT. The IntDFT algorithm has transform accuracy independent of N, which makes it especially useful for applications requiring large block lengths, e.g., audio compression. Its computational complexity is about 1.5 times that of a floating-point DFT. The IntDFT algorithm also has simple, modular structure which makes it ideal for real-world implementation.

References

[1] H. Huang, S. Rahardja, R. Yu, and X. Lin, "A fast algorithm of integer MDCT for lossless audio coding," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP'04)*, Montreal, Apr. 2004.

[2] H. Huang, X. Lin, S. Rahardja, and R. Yu, "A method for realizing reversible integer type-IV discrete cosine transform (intDCT-IV)," in *Proc. IEEE Int. Conf. Signal Processing (ICSP'04)*, Beijing, Sept. 2004.

[3] "Coding of Moving Pictures and Audio: Workplan for evaluation of integer MDCT for FGS to lossless experimentation framework," ISO/IEC JTC1/SC29/WG11 N5578, Pataya, Thailand, Mar. 2003.

[4] S. Oraintata, Y. J. Chen, and T. Q. Nguyen, "Integer Fast Fourier Transform," *IEEE Trans. Signal Processing*, vol. 50, pp. 607–618, Mar. 2002.

[5] J. Li, "A progressive to lossless embedded audio coder (PLEAC) with reversible modulated lapped transform," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP'03)*, Hong Kong, Apr. 2003.

[6] G. Plonka and M. Tasche, "Invertible integer DCT algorithms," *Appl. Comput. Harmon. Anal.*, vol. 15, pp. 70–88, July 2003.

[7] P. Hao and Q. SHi, "Matrix factorizations for reversible integer mapping," *IEEE Trans. Signal Processing*, vol. 49, pp. 2314–2324, Oct. 2001.

[8] Z. Wang, "Fast algorithms for the Discrete W Transform and for the Discrete Fourier Transform," *IEEE Trans. Accoustics, Speech, and Signal Processing*, vol. ASSP-32, pp. 803–816, Aug. 1984.

[9] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, pp. 247–269, 1998.

[10] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Accoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 285–295, Apr. 1986.

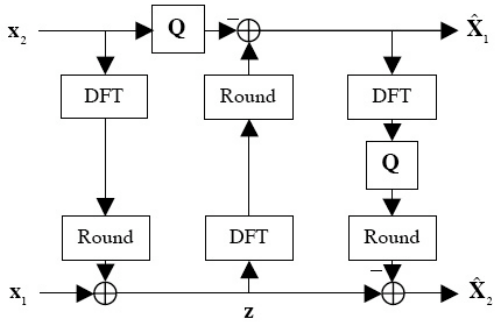

Fig. 1. Flow chart for IntDFT

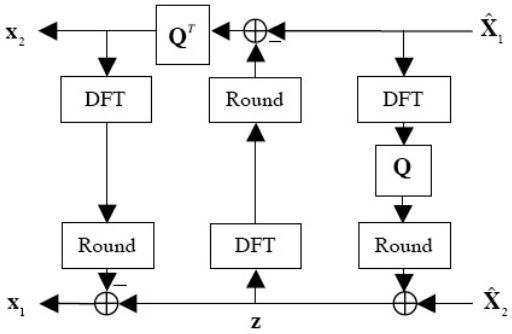

Fig. 2. Flow chart for IntIDFT

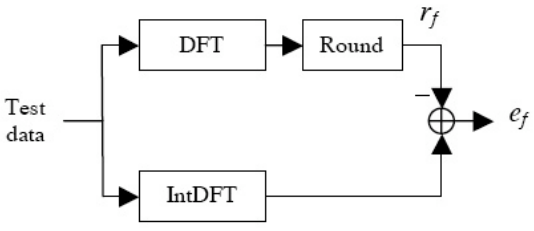

Fig. 3. Approximation error measurement for IntDFT

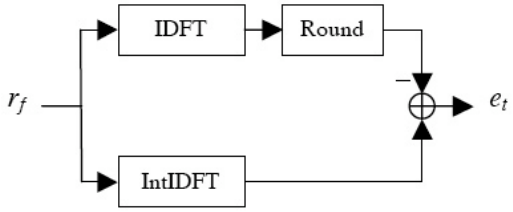

Fig. 4. Approximation error measurement for IntIDFT

4

Publication P4

H. Huang, P. Fränti, D. Huang, S. Rahardja, Cascaded RLS-LMS Prediction in MPEG-4 Lossless Audio Coding, *IEEE Trans. on Audio, Speech and Language Processing*, accepted for publication.

Cascaded RLS-LMS Prediction in MPEG-4 Lossless Audio Coding

Haibin Huang, Pasi Fränti, Dongyan Huang, *Member, IEEE,* and Susanto Rahardja, *Senior Member, IEEE*

***Abstract*—This paper describes the cascaded RLS-LMS prediction, which is part of the recently-published MPEG-4 Audio Lossless Coding international standard. The predictor consists of cascaded stages of simple linear predictors, with the prediction error at the output of one stage passed to the next stage as the input signal. A linear combiner adds up the intermediate estimates at the output of each prediction stage to give a final estimate of the RLS-LMS predictor. In the RLS-LMS predictor, the first prediction stage is a simple first-order predictor with a fixed coefficient value 1. The second prediction stage uses the Recursive Least Square algorithm to adaptively update the predictor coefficients. The subsequent prediction stages use the Normalized Least Mean Square algorithm to update the predictor coefficients. The coefficients of the linear combiner are then updated using the Sign-Sign Least Mean Square algorithm. For stereo audio signals, the RLS-LMS predictor uses both intra-channel prediction and inter-channel prediction, which results in a 3% improvement in compression ratio over using only the intra-channel prediction. Through extensive tests, the MPEG-4 Audio Lossless coder using the RLS-LMS predictor has demonstrated a compression ratio that is in par with the best lossless audio coders in the field. In this paper, the structure of the RLS-LMS predictor is described in detail and the optimal predictor configuration is studied through various experiments.**

I. Introduction

Lossless audio coding, as the name suggests, converts a digital audio signal from raw Pulse Code Modulation (PCM) format into a compressed format with a smaller file-size. The original audio signal can be perfectly re-constructed from the compressed file. Coupled with continually decreasing storage costs and the increasing growth of processor power, lossless audio compression started to gain popularity with the wide and rapid spread of broadband networks. Applications of lossless audio compression include digital music archival, network music downloading and broadcasting, personal music sharing, and mobile entertainment.

Over the years, various lossless audio compressors were developed by individuals, interested research groups, and commercial entities. For example, APE (by Monkey's Audio) [1] and FLAC (Free Lossless Audio Codec) [2] are popular ones in music file sharing over the internet. OptimFrog [3] and LA (Lossless Audio) [4] provide high compression ratios. Organizations like Apple, Microsoft, and Real Networks also developed their own lossless audio coders. Unfortunately, all these are proprietary coders, which are lacking in large-scale industrial adoption. In response to the industrial demand for a standardized lossless audio coding scheme, the *Motion Pictures Expert Group* (MPEG) issued a Call for Proposal in December 2002 [5]. Various parties responded and after three years of rigorous competition and productive collaboration, two schemes eventually emerged: the *Audio Lossless Coding* (ALS) and the *Scalable Lossless Coding* (SLS). Both ALS and SLS were adopted by MPEG because of their distinctive strengths: ALS provides a better compression performance, while SLS can be easily embedded with a lossy audio coder such as the MPEG-4 Advanced Audio Coding (AAC) [6]. The MPEG-4 ALS and SLS were formally published by the International Standard Organization as international standards in March 2006 [7], and June 2006 [8], respectively. Technical details of ALS and SLS are thoroughly explained in [9] and [10]. This paper focuses on the RLS-LMS predictor used in ALS.

In SLS, the input audio samples are first divided into blocks and then converted into transform coefficients by using the Integer Modified Discrete Cosine Transform (IntMDCT) [11]. The transform coefficients are scaled, quantized, and coded by the AAC encoder to generate a 'core' bit-stream, which constitutes the minimum quality/rate unit of the final lossless bitstream. For optimal coding efficiency, an error-mapping procedure is employed to remove the information that has already been coded in the core bit-stream from the transform coefficients. The residuals are subsequently coded by Bit-Plane Golomb Code [12] to form the final lossless bitstream. SLS provides fine-granular quality/rate scalability, and is well-suited for network music streaming services, where the bitstream can be dynamically truncated according to the available bandwidth.

In ALS, linear prediction is performed on the input audio samples to generate a residual signal, which has a smaller dynamic range than the input signal. The distribution of the residual signal can be closely modeled by a Laplacian (or two-sided geometric) distribution. The residual signal is entropy-coded with the Rice code [13]. For each block of audio samples, either all values can be coded by the same Rice code, or a single block can be further divided into four parts, each encoded with a different Rice code. Alternatively, the residual can also be coded by a more complex and efficient coding scheme called the block Gilbert-Moore code (BGMC) [14]. In BGMC, the residual distribution is further partitioned into three parts: a central region, flanked by two tail regions. The residuals in the tail regions are simply re-centered and coded with Rice codes. Residuals within the central region

H. Huang, D. Huang, and R. Susanto are with the Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, email: hhuang, huang, rsusanto@i2r.a-star.edu.sg.

P. Fränti is with the Department of Computer Science, University of Joensuu, Finland 80101, email: franti@cs.joensuu.fi

Manuscript submitted on March 1, 2007 and accepted on May 16, 2007;

are further split into those that belong to the Least Significant Bit (LSB) and the Most Significant Bit (MSB) parts. The LSB parts are directly transmitted using fixed-length codes without any processing, while the MSB parts are coded with the more efficient block Gilbert-Moore (arithmetic) code [15].

The MPEG-4 ALS has two different linear predictors: the *Linear Predictive Coding* (LPC) predictor [9], and the *RLS-LMS* predictor [16]. In the LPC predictor, the optimal predictor coefficients are computed using the Levinson-Durbin algorithm [17] for each block of samples. The audio samples pass through a linear predictor whose coefficients are the quantized version of the optimal coefficients. The quantized coefficients are coded together with the residual to form the lossless bitstream.

The RLS-LMS predictor provides an estimate of the current input sample using past input samples. The prediction residual is calculated as the difference between the current input sample and the derived estimate. Unlike the LPC predictor, only the residual is encoded and transmitted. There is no need to code the coefficients of the predictor, as an identical predictor runs in the decoder. The latter is guaranteed to always maintain the same states as that in the encoder. The RLS-LMS predictor consists of a cascade of stages made up of simple predictors. The residual of one predictor stage is passed on as the input of the next stage. The first stage is a first-order predictor with a fixed coefficient value 1. In the second stage, the predictor coefficients are updated using the *Recursive Least Square* (RLS) algorithm [18]. All the subsequent stages in the cascade use the *Normalized Least Mean Square* (NLMS) algorithm [18] in the updating of predictor coefficients. The estimate of the current input sample, or the final estimate, is generated by linearly combining the intermediate estimates generated by the cascaded predictor stages. The combiner coefficients are updated using the *Sign-Sign LMS* algorithm [18]. The residual signal is generated by subtracting the estimate from the current input sample, and is subsequently coded by the entropy coder to form the lossless bit-stream.

The development of the RLS-LMS predictor was motivated by prior work in cascaded prediction for lossless audio compression. In [19], Schuller proposed a cascaded predictor structure with three LMS prediction stages. The final estimate in that study was derived by a linear combination of the intermediate estimates from the three LMS predictors under the so-called Predictive Minimum Description Length Weighting. In the RLS-LMS predictor, the computation of the linear combiner coefficients is simplified by using the effective, low-complexity Sign-Sign LMS algorithm. In [20], Yu proposed a cascaded predictor using a low-order RLS predictor followed by a high-order LMS predictor. A final estimate was then given by directly adding the two intermediate estimates from the RLS and the LMS predictors. This predictor can be viewed as a special case of the RLS-LMS predictor that uses only two prediction stages. In Yu's work, the coefficients of the linear combiner take a fixed value 1. For stereo audio input, the RLS-LMS predictor can also perform *joint-stereo prediction*, which exploits the inherent correlation between the left and right audio channels. In this mode, the RLS prediction stage generates the estimate signal for each audio channel by using past samples from both channels. We find that joint-stereo prediction can bring about a 3% improvement in the compression ratio compared to the case where the predictor runs for the left and right channels independently.

The rest of the paper is organized as follows: The structure of the RLS-LMS predictor is introduced in Section II. The adaptive algorithms used for updating predictor coefficients are described in Section III, while the Joint-Stereo prediction is explained in Section IV. The optimal predictor configuration is determined through various experiments with results summarized in Section V. This section also provides comparison results with other lossless coders. A conclusion of the paper is given in the last section.

II. Cascaded RLS-LMS Prediction

The structure of the MPEG-4 ALS is shown in Figure 1, where the upper part shows the encoder and the lower half showing the decoder. In the encoder, an estimate of the current input sample is generated by the RLS-LMS predictor using past samples. This estimate is rounded to the nearest integer. A prediction residual is generated by subtracting the rounded estimate from the current input sample. The entropy coder subsequently encodes the residual with either the Rice code or the BGMC to form the lossless bit-stream.

In the decoder, the above process is reversed. The lossless bit-stream is first decoded into the prediction residual by the entropy decoder. The original audio data are then reconstructed by adding the residual to the rounded estimate. The RLS-LMS predictor in the decoder is identical to that in the encoder, and maintains the exact same states and coefficients as the latter at all times. Because of the lossless (noiseless) entropy encoding/decoding process, as well as the use of identical predictors in the encoder and the decoder, perfect-reconstruction of the original audio samples is guaranteed by ALS.

The structure of the cascaded RLS-LMS predictor is shown in Figure 3. The predictor consists of a cascade of simple prediction stages in the sequence of: a Differential PCM (DPCM) predictor, an RLS predictor, and a series of LMS predictors. The input samples pass through the prediction stages sequentially, with the residual of one stage serving as the input to the next stage. The estimates from each prediction stage are summed up by a linear combiner to generate the estimate of the current input sample according to

$$\hat{x}(n) = \sum_{k=1}^{K} c_k(n) y_k(n), \tag{1}$$

where $\hat{x}(n)$ is the estimate of the current input sample $x(n)$, n is the time index of samples, K is the number of prediction stages in the cascade, $c_k(n)$ are the coefficients of the linear combiner, k is the stage index, and $y_k(n)$ are estimates from the prediction stages.

In the k^{th} stage, defining the order of the predictor as M_k, the estimate $y_k(n)$ is given by

$$y_k(n) = \sum_{m=1}^{M_k} a_{k,m}(n) e_{k-1}(n-m), \tag{2}$$

where $a_{k,m}(n)$ are the coefficients of the predictor, m is the tap index, and $e_{k-1}(n)$ is the residual from the previous stage. The residual of the k^{th} stage is given by

$$e_k(n) = e_{k-1}(n) - y_k(n). \tag{3}$$

III. Adaptive Updating of Predictor Coefficients

The RLS-LMS predictor consists of a cascade of prediction stages and a linear combiner. All the prediction stages as well as the linear combiner update the coefficients adaptively, except for the first DPCM prediction stage, which uses a first-order predictor with a fixed coefficient value 1. The RLS algorithm is used in the second stage, and the NLMS algorithm is used in all the remaining stages in the cascade. For the linear combiner, the Sign-Sign LMS algorithm is used. The following sub-sections describe the adaptive algorithms used in each part of the RLS-LMS predictor.

A. DPCM Predictor

As the first predictor in the cascade, the DPCM predictor is a simple first-order predictor with the coefficient set to 1, i.e., the previous input sample is used as the estimate of the current input sample. The DPCM predictor is given by

$$\begin{aligned} y_1(n) &= x(n-1), & (4)\\ e_1(n) &= x(n) - y_1(n), & (5) \end{aligned}$$

where $y_1(n)$ is the estimate of the first prediction stage, $e_1(n)$ is the prediction residual, and $x(n-1)$ is the previous input sample.

B. RLS Predictor

The RLS predictor is the second predictor in the cascade. The RLS algorithm is used to adapt the predictor coefficients. The algorithm is initialized by setting the inverse autocorrelation matrix $\mathbf{P}$ as follows

$$\mathbf{P}(0) = \delta \mathbf{I},$$

where δ is a small positive number, $\mathbf{I}$ is an $M_2 \times M_2$ identity matrix, and M_2 is the order of the RLS predictor. The predictor coefficient vector $\mathbf{a}_2(n)$, defined as

$$\mathbf{a}_2(n) = [a_{2,1}(n), a_{2,2}(n), \ldots, a_{2,M_2}(n)]^T,$$

is initialized by

$$\mathbf{a}_2(0) = \mathbf{0}.$$

Here, the symbol T denotes the operation of vector transpose.

Define

$$\mathbf{e}_1(n) = [e_1(n-1), e_1(n-2), \ldots, e_1(n-M_2)]^T$$

as the RLS predictor input vector, for each instance of time, $n = 1, 2, \ldots$, the following calculations are made

$$\begin{aligned} \mathbf{v}(n) &= \mathbf{P}(n-1)\mathbf{e}_1(n), & (6)\\ m &= \begin{cases} \frac{1}{\mathbf{e}_1^T(n)\mathbf{v}(n)} & \text{if } \mathbf{e}_1^T(n)\mathbf{v}(n) \neq 0 \\ 1 & \text{else,} \end{cases} & (7)\\ \mathbf{k}(n) &= m\mathbf{v}(n), & (8)\\ y_2(n) &= \mathbf{a}_2^T(n-1)\mathbf{e}_1(n), & (9)\\ e_2(n) &= e_1(n) - y_2(n), & (10)\\ \mathbf{a}_2(n) &= \mathbf{a}_2(n-1) + \mathbf{k}(n)e_2(n), & (11)\\ \mathbf{P}(n) &= \text{Tri}\{\lambda^{-1}(\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{v}^T(n)\}. & (12) \end{aligned}$$

In (12), λ is the forgetting factor that is a positive value slightly smaller than 1. Tri$\{.\}$ denotes the operation of first computing the lower triangular part of $\mathbf{P}(n)$, and then copying the values in the lower triangular to the upper triangular according to

$$p_{i,j} = p_{j,i},$$

where $p_{i,j}$ is the element of matrix $\mathbf{P}(n)$ at the i^{th} row and the j^{th} column.

There is a slight difference between the above RLS algorithm and the standard version in [18]. The denominator in (7) is given by $\{\mathbf{e}_1^T(n)\mathbf{v}(n)\}$ instead of $\{\lambda + \mathbf{e}_1^T(n)\mathbf{v}(n)\}$ as defined in the standard RLS algorithm. The reason of why λ is neglected from the summation is that λ is several orders of magnitude smaller than $\{\mathbf{e}_1^T(n)\mathbf{v}(n)\}$ when the input signals are 16-bit PCM samples.

C. LMS Predictor

The RLS-LMS predictor has a series of LMS prediction stages. The NLMS algorithm [19] is used to adapt the coefficients of the LMS predictors. For the LMS predictor in the k^{th} stage, the coefficient vector

$$\mathbf{a}_k(n) = [a_{k,1}(n), a_{k,2}(n), \ldots, a_{k,M_k}(n)]^T$$

is initialized by

$$\mathbf{a}_k(0) = \mathbf{0},$$

where M_k is the order of the predictor.

Define

$$\mathbf{e}_{k-1}(n) = [e_{k-1}(n-1), e_{k-1}(n-2), \ldots, e_{k-1}(n-M_k)]^T$$

as the input vector to the k^{th} prediction stage, for each instance of time, $n = 1, 2, \ldots$, the following calculations are made

$$\begin{aligned} y_k(n) &= \mathbf{a}_k^T(n-1)\mathbf{e}_{k-1}(n), & (13)\\ e_k(n) &= e_{k-1}(n) - y_k(n), & (14)\\ \mathbf{a}_k(n) &= \mathbf{a}_k(n-1) + \frac{e_k(n)\mathbf{e}_{k-1}(n)}{1 + \mu_k \mathbf{e}_{k-1}^T(n)\mathbf{e}_{k-1}(n)} & (15) \end{aligned}$$

where $\mu_k \geq 1$ is the stepsize of the NLMS algorithm

D. Linear Combiner

The linear combiner multiplies a set of coefficients to the estimates from the DPCM, RLS, and LMS prediction stages. The results are summed up together to provide the estimate of the current input sample. The Sign-Sign LMS algorithm is used to adapt the coefficients of the linear combiner.

The coefficient vector is defined as

$$\mathbf{c}(n) = [c_1(n), c_2(n), \ldots, c_K(n)]^T,$$

where K is the number of prediction stages in the cascade. The input vector is given by

$$\mathbf{y}(n) = [y_1(n), y_2(n), \ldots, y_K(n)]^T.$$

The estimate of the RLS-LMS predictor is calculated as

$$\hat{x}(n) = \mathbf{c}^T(n)\mathbf{y}(n). \tag{16}$$

The linear combiner coefficients are updated according to

$$\mathbf{c}(n+1) = \mathbf{c}(n) + \alpha\,\mathrm{sgn}[\mathbf{y}(n)]\,\mathrm{sgn}[x(n) - \hat{x}(n)], \tag{17}$$

where the sgn function is defined as

$$\mathrm{sgn}[r] = \begin{cases} 1 & r > 0 \\ 0 & r = 0 \\ -1 & r < 0 \end{cases} \tag{18}$$

If the input to the sgn function is a vector, the output is also a vector that contains signs of each individual elements in that vector. The stepsize α takes a small positive value.

IV. Joint-Stereo prediction

For mono audio signals, the correlation that linear prediction tries to reduce is among audio samples within the same channel. This type of correlation is called *intra-channel* correlation. On the other hand, for stereo audio signals correlation also exists between samples in different channels. This type of correlation is referred to as *Inter-channel* correlation. Both intra-channel and inter-channel correlations are exploited by the RLS-LMS predictor through a joint-stereo prediction, where past samples from both L and R audio channels are used in estimating the current sample of each channel. This joint-stereo prediction is implemented in the second prediction stage as illustrated in Figure 2.

In Figure 2, the intra-channel predictor $\mathbf{a}_L$ generates an estimate for the L channel by using L channel samples. At the same time, the inter-channel predictor $\mathbf{b}_L$ generates another estimate by using samples in the R channel. The two estimates are added together to give the output estimate of the RLS prediction stage for the L channel. Let M_a and M_b be the orders of the intra-channel predictor $\mathbf{a}_L$ and inter-channel predictor $\mathbf{b}_L$, respectively, and the L channel estimate is given by

$$y_L(n) = \sum_{m=1}^{M_a} a_{L,m} x_L(n-m) + \sum_{m=1}^{M_b} b_{L,m} x_R(n-m), \tag{19}$$

where $a_{L,m}$ and $b_{L,m}$ are coefficients of predictor $\mathbf{a}_L$ and predictor $\mathbf{b}_L$, respectively. $x_L(n)$ and $x_R(n)$ are input samples in the L and R channels.

Similarly, the R channel estimate is given by

$$y_R(n) = \sum_{m=1}^{M_a} a_{R,m} x_R(n-m) + \sum_{m=0}^{M_b-1} b_{R,m} x_L(n-m), \tag{20}$$

where $a_{R,m}$ and $b_{R,m}$ are coefficients of the intra-channel predictor $\mathbf{a}_R$ and the inter-channel predictor $\mathbf{b}_R$, respectively. Note that the second summation term in (20) starts from 0 instead of 1 as in (19). The reason is that, in the ALS decoder, audio samples are re-constructed in the order of $[\cdots, x_L(n-1), x_R(n-1), x_L(n), x_R(n), x_L(n+1), x_R(n+1), \cdots]$. This order ensures that the L channel sample $x_L(n)$ is consistently decoded before the R channel sample $x_R(n)$, which motivated the use of $x_L(n)$ for the decoding of $x_R(n)$.

In joint-stereo prediction, the coefficients of the intra- and inter-channel predictors are updated using the RLS algorithm given in Section III-B. For the L channel, the input vector and coefficient vector are given by

$$\mathbf{x}_L(n) = [x_L(n-1), \ldots, x_L(n-M_a), x_R(n-1), \ldots, x_R(n-M_b)]^T$$

and

$$\mathbf{w}_L(n) = [a_{L,1}(n), \ldots, a_{L,M_a}(n), b_{L,1}(n), \ldots, b_{L,M_b}(n)]^T,$$

respectively. For the R channel, the input vector and coefficient vector are given by

$$\mathbf{x}_R(n) = [x_R(n-1), \ldots, x_R(n-M_a), x_L(n), \ldots, x_L(n-M_b+1)]^T$$

and

$$\mathbf{w}_R(n) = [a_{R,1}(n), \ldots, a_{R,M_a}(n), b_{R,1}(n), \ldots, b_{R,M_b}(n)]^T,$$

respectively. In joint-stereo prediction, the RLS routine is first called to update the L channel intra- and inter-channel predictors, and then called again to update the R channel predictors. There are also two $\mathbf{P}$ matrices, one for each channel.

V. Predictor optimization and Experimental Results

During the standardization process, extensive tests were conducted to optimize the parameters of the RLS-LMS predictor. To benchmark the performance of various lossless compressors, MPEG used a common test set [21] that consists of sampled waveforms of 15 different types of music. The sampling frequency/resolution used are: 48kHz/16bit, 48kHz/24bit, 96kHz/24bit, and 192kHz/24bit. Each waveform lasts 30 seconds, and the total playtime of the whole test set is 25 minutes. This test set was also used to run various experiments in this paper.

A. Predictor Signals and Residual Distribution

Figure 4 illustrates a segment of typical input and output signals of the RLS-LMS predictor. The typical probability density distributions of the residual signals are shown in Figure 5 for the following three predictor configurations:

a. DPCM (using only the first prediction stage)
b. DPCM + RLS (using the first and second prediction stages)

c. DPCM + RLS + LMS (using all the prediction stages)

Figure 5 shows that when the number of prediction stages increases, the residual distribution tends to concentrate towards the center. The entropies of the distributions in the figure are: a) 9.63, b) 6.54, and c) 6.08, respectively. The results show that the entropy is not reduced enough by the DPCM predictor alone, and significant improvement can be obtained by the subsequent RLS predictor. The LMS predictor reduces the entropy further by about 10%.

B. Various Predictor Configurations

The RLS-LMS predictor can be configured in a number of ways. A few intuitive configurations are listed in Table I. These configurations are compared in terms of compression ratio, which is defined as

$$\text{compression ratio} = \frac{\text{original filesize}}{\text{compressed filesize}}. \qquad (21)$$

TABLE I
COMPRESSION RATIOS FOR VARIOUS RLS-LMS PREDICTOR CONFIGURATIONS

No.	Predictor Configuration	Compression Ratio
1	DPCM + RLS + LMS	2.172
2	RLS + LMS	1.935
3	DPCM + LMS	2.126
4	DPCM + RLS	2.089
5	DPCM + mono RLS + LMS	2.109
6	DPCM + LMS + RLS	2.165

In Table I, six predictor configurations are compared. Among the six predictor configurations, Configuration 1 is the selected configuration with cascaded prediction stages in the sequence of DPCM, followed by RLS, and finally LMS. Configurations 2-4 contain only two stages in the cascade, with one stage turned off. Configuration 5 uses only intra-channel prediction (as inferred from the term 'mono') with no inter-channel prediction. In configuration 6, the position of the RLS and LMS prediction stages are swapped. To facilitate our analysis, the results listed in Table I are plotted in Figure 6. The results show that configuration 1 produces the highest compression ratio. This configuration is actually the one adopted by the standard. A 3% improvement in compression ratio is also found by comparing configuration 1 (using joint-stereo prediction) with configuration 5 (using only intra-channel prediction.

C. Configuration of LMS Prediction Stages

The RLS-LMS predictor contains a cascade of LMS prediction stages. Each stage is an LMS predictor of a certain order. Various configurations of the LMS predictor cascade are compared in Table II, where the second column lists the different combinations of LMS predictor orders. Each of the configurations in the table is comprised of three LMS stages, and has a total predictor order of 384.

In the first two configurations listed in Table II, the LMS cascade has predictor orders of descending values. Configuration 3 uses predictors of equal order, while in configuration 4 and 5, the predictor orders are increasing in sequence. The final two configurations show two alternative combination patterns of "short-long-short" and "long-short-long", respectively. For ease of comparison, the results of Table II are plotted in Figure 7. The results show that the highest compression ratio is provided by configuration 1, which is also the selected configuration of the RLS-LMS predictor. Configuration 1 confirms the observation in [19] that predictor orders should optimally be in a sequence of descending values. In addition, we find that a better compression ratio can be obtained when the LMS predictor orders are separated by wide margins, as demonstrated by the performances of configurations 1 and 2.

TABLE II
COMPRESSION RATIOS FOR VARIOUS CONFIGURATIONS OF LMS PREDICTION STAGES

No.	LMS Configuration	Compression Ratio
1	300 + 72 + 12	2.162
2	192 + 128 + 64	2.157
3	128 + 128 + 128	2.152
4	64 + 128 + 192	2.153
5	12 + 72 + 300	2.155
6	64 + 256 + 64	2.156
7	160 + 64 + 160	2.155

D. Number of LMS Prediction Stages

The RLS-LMS predictor can be configured to use different numbers of LMS prediction stages. Table III lists the configurations where one to six LMS stages are used. The total order of LMS predictors in each configuration is fixed at 512.

TABLE III
COMPRESSION RATIOS FOR CONFIGURATIONS WITH 1 TO 6 LMS PREDICTION STAGES

No.	Predictor Configuration	Compression Ratio
1	512	2.139
2	384 + 128	2.159
3	384 + 112 + 16	2.165
4	384 + 90 +30 + 8	2.165
5	360 + 100 + 36 + 12 + 4	2.164
6	300 + 120 + 60 + 22 + 8 + 2	2.162

In Table III, the LMS prediction stages are configured in a descending pattern of predictor orders, as suggested in Section V-C. The results are also shown in Figure 8. The compression ratio is found to peak at three LMS prediction stages, which suggests that the RLS-LMS predictor needs no more than three LMS stages.

E. Configuration of Linear Combiner

In the RLS-LMS predictor, the final estimate is generated by multiplying the intermediate estimates obtained from the prediction stages by the linear combiner coefficients, and summing up the results. The coefficients of the linear combiner are updated by the Sign-Sign LMS algorithm. we find that the best compression ratio can be obtained by updating only part of the combiner coefficients, while setting the others to 1. The configurations of the linear combiner coefficients are illustrated in Table IV, where the five central columns

list the values of the combiner coefficients. Each of these columns corresponds to the prediction stage that is indicated by the column header. Value '1' in the table indicates that the coefficient is fixed to 1, while a '*' means that the coefficient is adaptively updated. Six configurations of the combiner coefficients were tested, with the number of fixed, non-update coefficients increasing from zero to five. The corresponding compression ratios are plotted in Figure 9. As evident from the graph, the highest compression is achieved by fixing the first two combiner coefficients to 1, while adaptively updating the other three coefficients.

TABLE IV
COMPRESSION RATIOS FOR LINEAR COMBINER CONFIGURATIONS WITH 0 TO 5 NON-UPDATE COEFFICIENTS

No.	Configuration of Combiner Coefficients					Compression
	DPCM	RLS	LMS1	LMS2	LMS3	Ratio
0	*	*	*	*	*	1.985
1	1	*	*	*	*	2.158
2	1	1	*	*	*	2.172
3	1	1	1	*	*	2.171
4	1	1	1	1	*	2.169
5	1	1	1	1	1	2.167

F. Comparison of Lossless Compressors

A number of state-of-the-art lossless compressors were used to benchmark the performance of the RLS-LMS predictor. The experiments were run on a Pentium-4 2.4 GHz PC, and the results are summarized in Table V. The lossless compressors used are divided into two categories: standard coders from MPEG, and non-standard, proprietary ones. The first category includes the MPEG-4 SLS RM8 [22] and the MPEG-4 ALS RM18 [23] running in two predictor modes: the RLS-LMS mode and the LPC mode. Both these coders were implemented in 32-bit fixed-pointed C code. Coders for the second category are selected according to the following considerations: OptimFROG [3] is reported [24]–[26] to be one of the top lossless audio coders in terms of the compression ratio, and is therefore used as a major benchmark for our comparison. Because of the widespread popularity over internet, the Monkey's coder [1] is also included in the comparison. The compression ratio results are plotted in Figure 10.

TABLE V
COMPARISON OF LOSSLESS COMPRESSORS

Lossless Compressors		Compress. Ratio	Speed (× realtime)	
			Encode	Decode
Standard	ALS RM18 (RLS-LMS)	2.172	0.4	0.4
	ALS RM18 (LPC)	2.148	20	40
	SLS RM8	2.123	22	25
Non-standard	OptimFROG 4.600ex	2.174	1.2	2.1
	Monkey's Audio 3.99	2.097	6	6

Clearly, the highest compression ratio is provided by OptimFROG 4.600ex, closed followed by the ALS RM18 coder running in the RLS-LMS predictor mode. This suggests that the latter becomes one of the top performers in the field in terms of the lossless compression ratio. Among the five tested coders, the ALS RM18 coder running in the RLS-LMS predictor mode gives the slowest encoding/decoding speed. This slow speed of the coder results from the high complexity of the RLS-LMS predictor, which consists of a computationally intensive RLS filter, as well as large-order LMS filters.

The ALS reference software RM18 was implemented in 32-bit fixed-point arithmetics. In the RLS-LMS predictor, to guarantee convergence of the adaptive coefficients update recursions, computations inside the RLS recursions must be performed with a high numerical precision [18]. This requirement is generally not a problem for implementations done in double-precision floating-point, but remains a challenging issue for fixed-point implementations because of the much smaller dynamic range to represent values in fixed-point. In the RLS and NLMS recursions, each multiplication/division operation is coupled with necessary pre-scaling, post-scalings and range comparison instructions to keep the numerical precision high. As a result, a multiplication done in one floating-point step can only be achieved by several fixed-point steps. This overhead with fixed-point implementation also contributes to the high-complexity of the RLS-LMS predictor.

The MPEG-4 lossless audio compression standard provides a range of coders to handle different application scenarios. For example, the SLS coder provides lossless bit-stream that can be arbitrarily truncated to cater for different transmission bandwidth requirements. The ALS coder running in the LPC predictor mode has a very high encoding/decoding speed. Among all the coders in the standard, the ALS coder in the RLS-LMS predictor mode provides the highest level of audio compression.

VI. CONCLUSION

This paper provides a detailed description of the cascaded RLS-LMS predictor in the MPEG-4 ALS standard. The predictor consists of a cascade of linear prediction stages in the sequence of a DPCM predictor, followed by an RLS predictor, and finally a series of LMS predictors. A linear combiner then sums up the intermediate estimate signals from these prediction stages to generate the final estimate signal of the RLS-LMS predictor. Various configurations were experimented with the predictor to optimize the predictor settings. The results from these experiments provide valuable insights and guidelines to the field of cascaded adaptive filter design. The ALS coder running the RLS-LMS predictor demonstrates a compression ratio that is in par with the best lossless audio coders in the field. An important work in the future is to reduce the high computational complexity of the predictor. Some potential strategies are: optimizing parameters of the adaptive algorithms so that the orders of the RLS and LMS predictors can be kept low, and choosing adaptive algorithms that are less computationally intensive than the RLS and NLMS algorithms. The core portion of the RLS-LMS predictor is made up of a highly-efficient, fast-tracking cascaded adaptive filter, which also has a wide range of applications including: system identification, blind source classification and separation, channel equalization, beam-forming, and noise cancelation.

REFERENCES

[1] Monkey's Audio. [Online]. Available: http://www.monkeysaudio.com/
[2] FLAC. [Online]. Available: http://flac.sourceforge.net/
[3] OptimFROG. [Online]. Available: http://www.losslessaudio.org/
[4] Lossless Audio. [Online]. Available: http://www.lossless-audio.com/
[5] ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group, "Final Call for Proposals on MPEG-4 Lossless Audio Coding," N5208, Shanghai, China, Oct. 2002.
[6] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–814, 1997.
[7] ISO/IEC 14496-3:2005/Amd 2:2006, "Audio Lossless Coding (ALS), new audio profiles and BSAC extensions," Mar. 2006.
[8] ISO/IEC 14496-3:2005/Amd 3:2006, "Scalable Lossless Coding (SLS)," Jun. 2006.
[9] T. Liebchen, T. Moriya, N. Harada, Y. Kamamoto, and Y. Reznik, "The MPEG-4 Audio Lossless Coding (ALS) standard - technology and applications," in *119th AES Convention*, New York, USA, Oct. 2005.
[10] R. Yu, R. Geiger, S. Rahardja, J. Herre, X. Lin, and H. Huang, "MPEG-4 scalable to lossless audio coding," in *117th AES Convention*, San Francisco, USA, Oct. 2004, preprint 6183.
[11] Y. Yokotani, R. Geiger, G. D. T. Schuller, S. Oraintara, and K. R. Rao, "Lossless audio coding using the IntMDCT and rounding error shaping," *IEEE Trans. Audio Speech Language Processing*, vol. 14, no. 6, pp. 2201–2211, Nov. 2006.
[12] R. Yu, S. Rahardja, X. Lin, and C. C. Ko, "A fine granular scalable to lossless audio coder," *IEEE Trans. Audio Speech Language Processing*, vol. 14, no. 4, pp. 1352–1363, Jul. 2006.
[13] R. F. Rice, "Some practical universal loiseless coding techniques," *JPL Tech. Reps 79-22*, 1979.
[14] Y. A. Reznik, "Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS)," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'04)*, vol. 3, Montreal, Canada, May 2004, pp. 1024–7.
[15] E. N. Gilbert and E. F. Moore, "Variable-length binary encodings," *Bell Syst. Tech. J. 38*, pp. 932–967, Jul. 1959.
[16] H. Huang, S. Rahardja, X. Lin, R. Yu, and P. Franti, "Cascaded RLS-LMS prediction in MPEG-4 lossless audio coding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP'06)*, vol. 5, Toulouse, France, May 2006, pp. 181–184.
[17] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
[18] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1999.
[19] G. D. T. Schuller, B. Yu, D. Huang, and B. Edler, "Perceptual audio coding using adaptive pre- and post-filters and lossless compression," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 6, pp. 379–390, Sep. 2002.
[20] R. Yu and C. C. Ko, "Lossless compression of digital audio using cascaded RLS-LMS prediction," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 532–537, Nov. 2003.
[21] Audio Research Labs. [Online]. Available: http://www.audioresearchlabs.com/
[22] MPEG-4 SLS Reference Software. [Online]. Available: ftp://vpsp.i2r.a-star.edu.sg
[23] MPEG-4 ALS Reference Software. [Online]. Available: http://www.nue.tu-berlin.de/forschung/projekte/lossless/mp4als.html
[24] Hydrogen Audio. [Online]. Available: http://wiki.hydrogenaudio.org/
[25] Performance comparison of lossless audio compressors. [Online]. Available: http://members.home.nl/w.speek/comparison.htm/
[26] Compression and speed of lossless audio formats. [Online]. Available: http://web.inter.nl.net/users/hvdh/lossless/lossless.htm

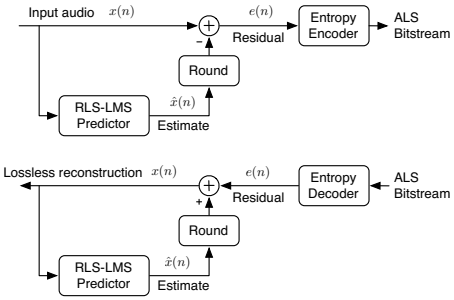

Fig. 1. MPEG-4 ALS encoder (top) and decoder (bottom) with the RLS-LMS predictor

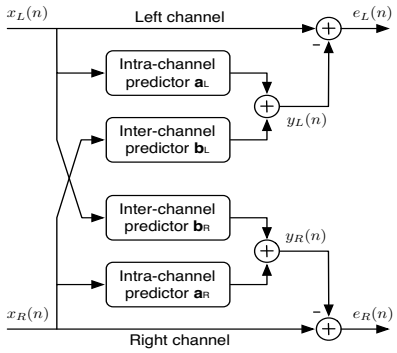

Fig. 2. Joint-stereo prediction

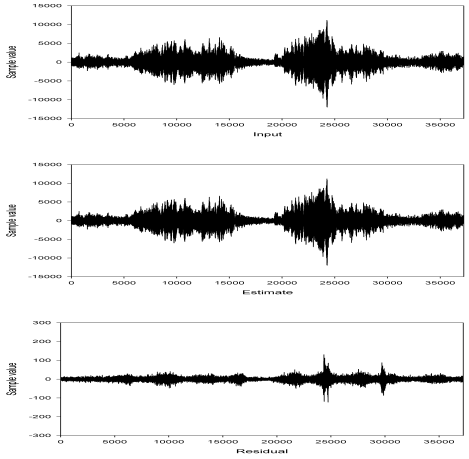

Fig. 4. Input and output signals of the RLS-LMS predictor

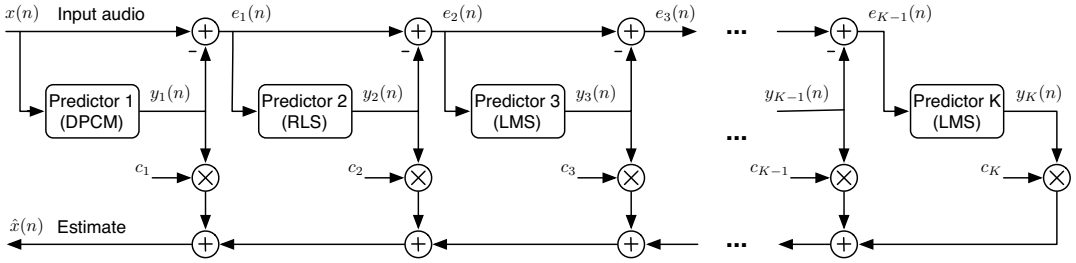

Fig. 3. Structure of the cascaded RLS-LMS predictor

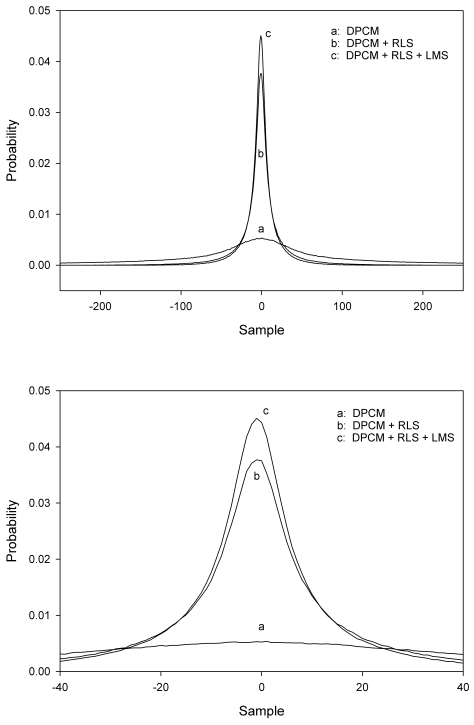

Fig. 5. Residual distributions for three predictor configurations (above), and a close-up of the central region of the distributions (below).

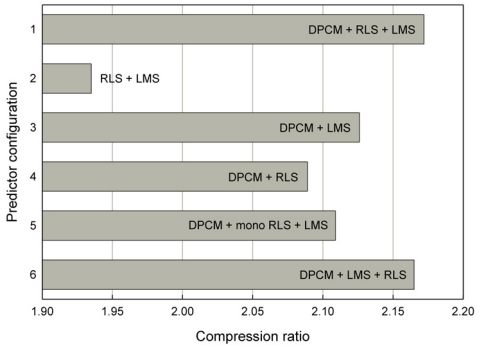

Fig. 6. Compression ratios for various RLS-LMS predictor configurations

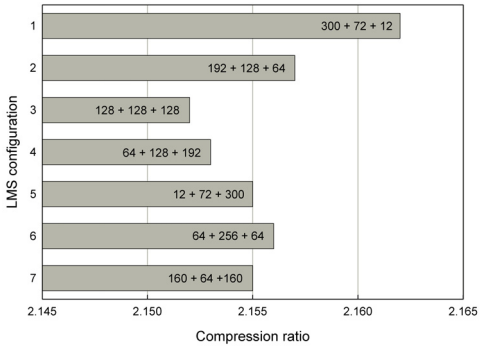

Fig. 7. Compression ratios for various configurations of LMS predictor stages

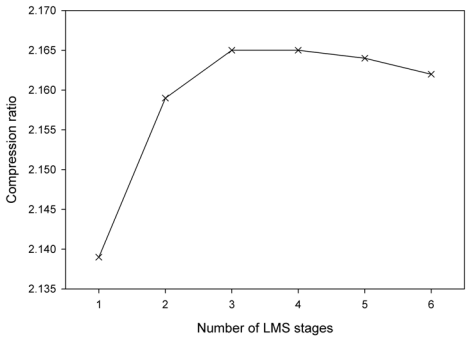

Fig. 8. Compression ratios for configurations with 1 to 6 LMS prediction stages

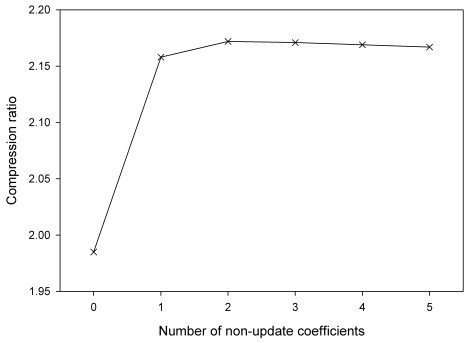

Fig. 9. Compression ratios for linear combiner configurations with 0 to 5 non-update coefficients

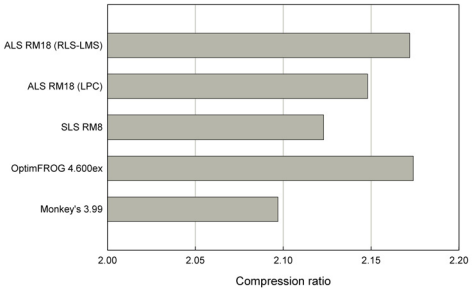

Fig. 10. Comparison of lossless compressors

5

Publication P5

R. Yu, X. Lin, S. Rahardja, C. C. Ko, H. Huang, Improving Coding Efficiency for MPEG-4 Audio Scalable Lossless Coding, *Proc. 2005 Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, pp. III169-III172, Philadelphia, USA, March 18-23, 2005.

IMPROVING CODING EFFICIENCY FOR MPEG-4 AUDIO SCALABLE LOSSLESS CODING

[1]R. Yu, [1]X. Lin, [1]S. Rahardja, [2]C. C. Ko and [1]H. Huang
[1]Institute for Infocomm Research (I^2R), A*STAR,
21 Heng Mui Keng Terrace, Singapore 119613, email: *{rsyu,linxiao,rsusanto,hhuang}@i2r.a-star.edu.sg*
[2]Department of Electrical and Computer Engineering,
National University of Singapore, Singapore 119260, email: elekocc@nus.edu.sg

ABSTRACT

The recently introduced MPEG standard for lossless audio coding (MPEG-4 Audio Scalable to Lossless (SLS) coding technology) provides a universal audio format that integrates the functionalities of lossy audio coding, lossless audio coding and fine granular scalable audio coding in a single framework. In this paper, we propose two coding methods that improve the coding efficiency of the SLS, namely, the context-based arithmetic code (CBAC) method and the low energy mode code method. These two coding methods work harmonically with the current SLS framework and preserve all its desirable features such as fine granular scalability, while successfully improving its lossless compression ratio performance.

1. INTRODUCTION

During the last two decades, considerable efforts have been devoted to the development of audio compression technologies, whose primary objective is the transmission of high-quality digital audio over communication channels with limited bandwidth. These efforts have led to fruitful results. Nowadays, most audio compression algorithms, such as the MPEG-1 Layer III (mp3) [1] or MPEG-4 AAC [2], can deliver "perceptually transparent" CD quality (48kHz, 16bit/sample) audio at bit-rates from 48 ~ 64 kbps per channel, or 7 ~ 14 times compression compared with the uncompressed audio.

Recently, with advances in broadband access networking and storage technologies, an increasing number of digital audio applications have emerged to provide high quality audio services with high sampling rate, high amplitude resolution (e.g., 96 kHz, 24 bit/sample) audio at lossless quality. Meanwhile, there will still be many applications that require highly compressed digital audio. A solution that provides interchangeability across these two application domains would greatly simplify the problem of migrating audio content between these domains, and facilitate the transition from lossy to lossless digital audio service. In response to this need, the international standard body ISO/IEC JTC1/SC29/WG11, also known as the Moving Picture Experts Group (MPEG), has recently issued a Call for Proposal (CfP) [3] on lossless audio coding. Contributions for a solution for scalable to lossless (SLS) audio compression are invited. Subsequently, the Advanced Audio Zip [4] has been adopted as the Reference Model (RM) [5] for this work at the 65th MPEG in July 2003.

AAZ adopts the transform coding approach where the Integer MDCT (IntMDCT) [6], an integer implementation for the Modified Discrete Cosine Transform (MDCT) [7] is used as its filterbank for lossless coding. In AAZ, the IntMDCT spectral data of the input audio are first coded with an MPEG-4 AAC encoder to generate an embedded AAC bit-stream, while the residual spectrum between the IntMDCT spectral data and the their quantized value by the embedded AAC encoder is subsequently coded with Bit-Plane Golomb Code (BPGC) [8] to produce the fine granular scalable lossy to lossless portion of the final lossless bit-stream.

In this paper, we propose two coding methods that further refine the BPGC coding process to improve its coding efficiency for lossless compression of IntMDCT residual. The first one is Context-Based Arithmetic Code (CBAC), which uses context modeling technology to capture the statistical dependencies of the probability distribution of the IntMDCT residual signal on its frequency location, amplitude of its adjacent spectral lines, and the core layer AAC quantizer. The second one is the so-call low energy mode coding method, which is used to replace the BPGC process for IntMDCT spectral data from time/frequency (T/F) regions with extremely low energy level. Since these spectral data are dominated by the rounding errors of the IntMDCT algorithm, they have a probability distribution that is far away from the Laplacian distribution for which BPGC/CBAC would lead to suboptimal compression results. The structure of the MPEG-4 SLS codec integrated with the proposed CBAC and low energy mode coding methods is illustrated in Fig. 1.

The CBAC and the low energy mode coding methods have been previously proposed to MPEG as Core Experiments (CE) [11][12] to improve the performance of MPEG-4 SLS. It is found that they successfully improve the compression ratio performance of SLS, while preserving all its other features such as fine granular scalability. As a result, these technologies have been adopted by MPEG, and incorporated into the RM for this work.

2. BIT-PLANE CODING WITH CBAC

The BPGC coding process used in MPEG-4 SLS is basically a bit-plane coding scheme where the bit-plane

symbols are arithmetic coded with a structural frequency assignment rule. Consider an input data vector $\mathbf{e} = \{e[0],...,e[N-1]\}$, for which N is the dimension of $\mathbf{e}$. Each element $e[k]$ in $\mathbf{e}$ is first represented in a binary format as

$$e[k] = (2s[k]-1)\cdot\sum_{j=0}^{M-1} b[k,j]\cdot 2^j, k = 0,...,N-1,$$

which comprises of a sign symbol

$$s[k] \overset{\Delta}{=} \begin{cases} 1 & , e[k] \geq 0 \\ 0 & , e[k] < 0 \end{cases}, \; k = 0,...,N-1$$

and bit-plane symbols $b[k,j] \in \{0,1\}$, $i = 1,...,k$. Here, M is the Most Significant Bit (MSB) for $\mathbf{e}$ that satisfies $2^{M-1} \leq \max\{|e[k]|\} < 2^M, k = 0,...,N-1$. The bit-planes symbols are then scanned and coded from the MSB to the Least Significant Bit (LSB) over all the elements in $\mathbf{e}$, and coded by using arithmetic code with a structural frequency assignment $Q(j)$ given by

$$Q(j) = \begin{cases} \dfrac{1}{1+2^{2^{j-L}}} & , j \geq L \\ \dfrac{1}{2} & , j < L \end{cases}, \qquad (1)$$

where the Lazy plane parameter L can be selected using the adaptation rule

$$L = \min\{L' \in \mathbb{Z} \mid 2^{L'+1} N \geq A\}. \qquad (2)$$

Here A is the absolute sum of the data vector $\mathbf{e}$.

Although the above BPGC coding process delivers excellent compression performance for data that are issued from independent and identically distributed (iid) source with Laplacian distribution [8], it lacks the capability to explore the statistical dependencies that may exist in certain sources to achieve better compression performance. These correlations can be very effectively captured by incorporating context-modeling technology into the BPGC coding process, where the frequency assignment for arithmetic coding of bit-plane symbols is not only depended on the distance of the current bit-plane to the Lazy plane parameter as in the frequency assignment rule (1), but also on other possible elements that may effect the probability distribution of these bit-plane symbols.

In the context of lossless coding of IntMDCT spectral data for audio, elements that possibly effect the distribution of the bit-plane symbols include the frequency locations of the IntMDCT spectral data, amplitude of the adjacent spectral lines, and status of the AAC core quantizer. In CBAC, these correlations are effectively captured by using several types of contexts. The guide in selecting these contexts is to try to find

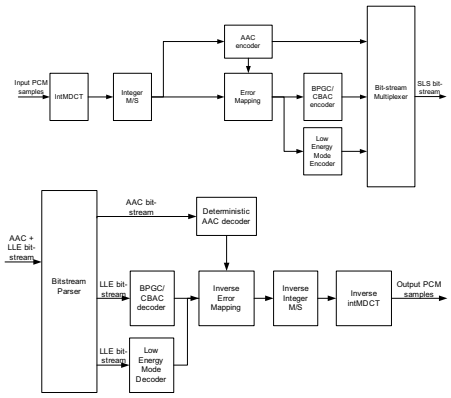

Fig. 1 Block diagram of the MPEG-4 SLS encoder and decoder integrated with CBAC and low energy mode coding

those contexts that are "most" correlated to the distribution of the bit-plane symbols. In addition, cares have to be taken to avoid overpopulating the number of the contexts, which may eventually deteriorate the coding efficiency performance as a result of modeling cost as well as introducing unnecessary implementation burdens.

In CBAC, three types of contexts, namely, the *frequency band (FB)* context, the *distance to lazy (D2L)* context, and the *significant state (SS)* context are used. The detailed context assignments are summarized as follows:

- Context 1: *frequency band (FB)* – It is found in our experiments that the probability distribution of bit-plane symbols of IntMDCT varies for different frequency bands. Therefore, in CBAC the IntMDCT spectral data is classified, empirically, into three different *FB* contexts, namely, Low Band (0 ~ 4 kHz, *FB* = 0), Mid Band (4 kHz ~ 11 kHz, *FB* = 1) and High Band (above 11 kHz, *FB* = 2).
- Context 2: *distance to lazy (D2L)* – The D2L context is defined as the distance of the current bit-plane to the BPGC Lazy plane parameter *L*. The introduction of this context follows the same rationale behind the BPGC frequency assignment rule (1), which is based on the fact the skew of the probability distribution of the bit-plane symbols from a source with Laplacian or near-Laplacian distribution tends to decrease as the number of *D2L* decrease [8]. To reduce the total number of the *D2L* context, all the bit-planes with $D2L < -2$ are grouped into one context where all the bit-plane symbols are coded with probability 0.5 since the probability skew in these contexts is so small that the coding gain is negligible if they are arithmetic coded.

- Context 3: *significant state (SS)* – The *SS* context tries to group the factors that may correlate with the distribution of the amplitude of the IntMDCT residual in one place. These include the amplitude of the adjacent IntMDCT spectral lines and the quantization interval of the AAC core quantizer if it has previously quantized in the core encoder. The detailed configuration of the *SS* context can be found at [12].

3. LOW ENERGY MODE CODING

The BPGC/CBAC coding process described above provides excellent compression performance only for source with Laplacian or near-Laplacian distribution. However, in many audio signals, it is found that there exist some "silence" T/F regions, such as the high frequencies regions for some single instrumental music, or silence portion of the music, from which the IntMDCT spectral data are in fact dominated by the rounding errors accumulated from the rounding operation in the IntMDCT algorithm. The distribution of these rounding errors is far away from the Laplacian distribution, and hence the use of BPGC/CBAC results only in suboptimal compression performance. In order to improve the coding efficiency, we replace the BPGC/CBAC coding process with a different coding method, namely, the low energy mode coding for IntMDCT spectral data from those regions.

The low energy mode coding is performed on scale factor bands (sfb) for which the BPGC parameter L is smaller or equal to 0. Here L in effect signals the energy level of the IntMDCT spectrum. At low energy mode coding, the amplitude of the residual spectral data $e[k]$ is first converted into unitary binary string $\mathbf{b} = \{b[0], b[1], \ldots, b[pos], \ldots\}$ as illustrated in table 1:

Amplitude of $e[k]$	Binary string $\{b[pos]\}$
0	0
1	1 0
2	1 1 0
…	…
$2^M - 2$	1 1 … … … 1 0
$2^M - 1$	1 1 … … … 1 1
pos	0 1 2 3 …

Table 1. Binarization of IntMDCT error spectrum at low energy mode. Here M is the maximum bit-plane.

It can be seen that the probability distributions of these symbols are jointly determined by their position *pos*, and the distribution of $e[k]$:

$$\Pr\{b[pos]=1\} = \Pr\{e[k] > pos \mid e[k] \geq pos\}$$
$$0 \leq pos < 2^M .$$

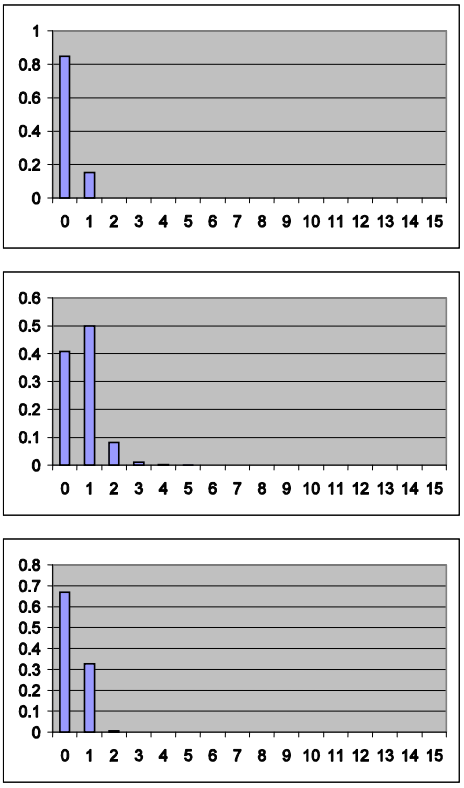

Fig. 2 Histogram of the IntMDCT residual signal for L=0 (Top), L=-1 (Middle), and L=-2 (Bottom) from a piece of audio training set.

In addition, the distribution of $e[k]$ is directly related to the Lazy plane parameter L (see Fig. 2 for an example). Therefore, in low energy mode coding $b[pos]$ is arithmetic coded conditioned on its position *pos* and the BPGC parameter L with a trained frequency table. The sign bit for non-zero $e[k]$ is also coded with probability 0.5 after its amplitude is coded.

4. PERFORMANCE

We compare the compression ratio performances of the MPEG-4 SLS codec and its improved version that integrates the CBAC and low energy mode coding methods. The audio sequences used in our comparison are from the MPEG lossless coding task group [3], which include two popular combinations of sampling rate and quantization word lengths, i.e. 48 kHz/16 bit and 96 kHz/24 bit. Detailed results are listed in Table. 2.

Evidently, these results suggest that the proposed technologies successfully improve the performance of the MPEG-4 SLS encoder, which have improved the compression ratio performance of MPEG-4 SLS encoder by 2.61% and 1.52% for 48/16 testing set and 96/24 testing set respectively.

5. CONCLUSION

As the latest member in the MPEG audio coding family, the MPEG-4 Audio Scalable Lossless (SLS) coding provides fine granular scalable lossy to lossless coding, a functionality that complements the existing MPEG audio coding tools. In this paper we introduce two coding methods to further improve the coding efficiency of MPEG-4 SLS. It is found that the proposed technologies can be readily integrated within the framework of the MPEG-4 SLS, leading to significantly improvement in terms of lossless compression ratio performance as shown in simulation results. The proposed technologies have been adopted by MPEG as successful Core Experiments for MPEG-4 SLS, and they have been incorporated with the RM for this work.

Item	MPEG-4 SLS	Improved MPEG-4 SLS	Improve-ment
avemaria	2.44	2.53	3.66%
blackandtan	1.73	1.75	1.49%
broadway	1.93	1.97	1.72%
cherokee	1.79	1.82	1.72%
clarinet	2.04	2.08	2.24%
cymbal	2.75	2.99	8.66%
dcymbals	1.59	1.60	1.22%
etude	2.30	2.37	2.92%
flute	2.38	2.46	3.04%
fouronsix	2.03	2.08	2.56%
haffner	1.77	1.80	1.93%
mfv	3.06	3.25	6.20%
unfo	1.86	1.90	2.16%
violin	2.01	2.05	2.37%
waltz	1.81	1.84	1.77%
Overall	2.03	2.09	2.61%

Item	MPEG-4 SLS	Improved MPEG-4 SLS	Improve-ment
avemaria	1.95	1.96	0.54%
blackandtan	2.06	2.10	2.03%
broadway	1.71	1.72	0.68%
cherokee	2.09	2.13	2.13%
clarinet	2.20	2.25	2.27%
cymbal	2.13	2.14	0.61%
dcymbals	1.63	1.64	0.61%
etude	1.89	1.90	0.52%
flute	2.23	2.28	2.22%
fouronsix	2.26	2.32	2.67%
haffner	1.95	1.99	1.98%
mfv	1.97	1.98	0.58%
unfo	2.14	2.19	2.55%
violin	2.08	2.13	2.10%
waltz	2.11	2.15	2.26%
Overall	2.01	2.04	1.52%

Table 2. Comparison of compression ratio performance (Top: 48kHz/16bit; Bottom: 96kHz/24bit)

REFERENCE

[1] ISO/IEC JTC1/SC29/WG11, "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part3: Audio," IS 11172-3, 1992.

[2] Information Technology – Coding of Audiovisual Objects, Part 3. Audio, Subpart 4 Time/Frequency Coding, ISO/JTC1/SC29/WG11, 1998.

[3] ISO/IEC JTC1/SC29/WG11 N5040, "Call for Proposals on MPEG-4 Lossless Audio Coding," Awaji Island, Japan, 2002.

[4] R. Yu, X. Lin, S. Rahardja and C. C. Ko, "A scalable lossy to lossless audio coder for MPEG-4 audio scalable lossless coding," Proceeding of ICASSP 2004.

[5] ISO/IEC JTC1/SC29/WG11 N5720, "Workplan for Audio Scalable Lossless Coding (SLS)," Trondheim, Norway, 2003.

[6] R. Geiger, T. Sporer, J. Koller and K. Brandenburg, "Audio Coding based on Integer Transforms," 111th AES Convention, Sep. 2001.

[7] J. Princen, A. Johnson, A. Bradley, "Subband/transform coding using filter bank designs based on time domain aliasing cancellation", *Proc. of the ICASSP 1987*, pp. 2161 – 2164, 1987.

[8] R. Yu, C.C. Ko, S. Rahardja and X. Lin, "Bit-plane Golomb code for sources with Laplacian distributions," Proc. of the ICASSP 2003, pp. 277 – 280, 2003.

[9] R. Yu, X. Lin, S. Rahardja and H. Huang, "Technical Description of I2R's Proposal for MPEG-4 Audio Scalable Lossless Coding (SLS): Advanced Audio Zip (AAZ)," ISO/IEC JTC1/SC29/WG11, MPEG2003/M10035, October 2003, Brisbane, Australia

[10] R. Yu, C.C. Ko, S. Rahardja and X. Lin, "Bit-plane Golomb code for sources with Laplacian distributions," Proceeding of ICASSP 2003.

[11] R. Yu, X. Lin, S. Rahardja, and H. Haibin, "Proposed Core Experiment for improving coding efficiency in MPEG-4 audio scalable coding (SLS)", ISO/IEC JTC1/SC29/WG11, M10136, Oct. 2003, Brisbane, Australia.

[12] R. Yu, X. Lin, S. Rahardja, and H. Haibin, "Proposed Core Experiment for improving coding efficiency in MPEG-4 audio scalable coding (SLS)", ISO/IEC JTC1/SC29/WG11, M10683, March. 2004, Munich, Germany.

Dissertations at the Department of Computer Science and Statistics

Rask, Raimo. Automatic Estimation of Software Size during the Requirements Specification Phase - Application of Albrecth's Function Point Analysis Within Structured Methods. Joensuun yliopiston luonnontieteellisiä julkaisuja, 28 - University of Joensuu. Publications in Sciences, 28. 128 p. Joensuu, 1992.

Ahonen, Jarmo. Modeling Physical Domains for Knowledge Based Systems. Joensuun yliopiston luonnontieteellisiä julkaisuja, 33. 127 p. Joensuu, 1995.

Kopponen, Marja. CAI in CS. University of Joensuu, Computer Science, Dissertations 1. 97 p. Joensuu 1997.

Forsell, Martti. Implementation of Instruction-Level and Thread-Level Parallelism in Computers. University of Joensuu, Computer Science, Dissertations 2. 121 p. Joensuu 1997.

Juvaste, Simo. Modeling Parallel Shared Memory Computations.U niversity of Joensuu, Computer Science, Dissertations 3. 190 p. Joensuu 1998.

Ageenko, Eugene. Contex-based Compression of Binary Images. University of Joensuu, Computer Science, Dissertations 4. 111 p. Joensuu 2000.

Tukiainen, Markku. Developing a New Model of Spreadsheet Calculations: A Goals and Plans Approach. University of Joensuu, Computer Science, Dissertations 5. 151 p. Joensuu 2001.

Eriksson-Bique, Stephen. An Algebraic Theory of Multidimensional Arrays. University of Joensuu, Computer Science, Dissertations 6. 278 p. Joensuu 2002.

Kolesnikov, Alexander. Efficient Algorithms for Vectorization and Polygonal Approximation. University of Joensuu, Computer Science, Dissertations 7. 204 p. Joensuu 2003.

Kopylov, Pavel. Processing and Compression of Raster Map Images. University of Joensuu, Computer Science, Dissertations 8. 132 p. Joensuu 2004.

Virmajoki, Olli. Pairwise Nearest Neighbor Method Revisited. University of Joensuu, Computer Science, Dissertations 9. 164 p. Joensuu 2004.

Suhonen, Jarkko A Formative Development Method for Digital Learning Environments in Sparse Learning Communities. University of Joensuu, Computer Science, Dissertations 10. 154 p. Joensuu 2005.

Xu, Mantao K-means Based Clustering and Context Quantization. University of Joensuu, Computer Science, Dissertations 11. 162 p. Joensuu 2005.

Kinnunen, Tomi Optimizing Spectral Feature Based Text-Independent Speaker Recognition. University of Joensuu, Computer Science, Dissertations 12. 156 p. Joensuu 2005.

Kärkkäinen, Ismo Methods for Fast and Reliable Clustering. University of Joensuu, Computer Science, Dissertations 13. 108 p. Joensuu 2006.

Tedre, Matti The Development of Computer Science: A Sociocultural Perspective. University of Joensuu, Computer Science, Dissertations 14. 502 p. Joensuu 2006.

Akimov, Alexander Compression of Digital Maps. University of Joensuu, Computer Science, Dissertations 15. 116 p. Joensuu 2006.

Vesisenaho, Mikko Developing University-level Introductory ICT Education in Tanzania: A Context Approach. University of Joensuu, Computer Science, Dissertations 16. 199 p. Joensuu 2007.

Huang, Haibin Lossless Audio Coding for MPEG-4. University of Joensuu, Computer Science, Dissertations 17. 86 p. Joensuu 2007.

Publisher

University of Joensuu
Department of Computer Science and Statistics

Exchanges

Joensuu University Library / Exchanges
P.O. Box 107, FI-80101 Joensuu, FINLAND
Tel. +358-13-251 2677, fax +358-13-251 2691
e-mail: vaihdot@joensuu.fi