DEVELOPMENTS WITH A ZEROTREE AUDIO CODEC

BEN LESLIE[1], CHRIS DUNN[2] AND MARK SANDLER

Department of Electronic Engineering, King's College,
Strand, London WC2R 2LS.
{ben.leslie, chris.dunn, mark.sandler} @kcl.ac.uk

This paper describes the use of zerotree quantisation within audio coders. We introduce the EZK algorithm, a version of zerotree quantisation developed by the authors, and describe why it provides superior compression performance compared to other tree algorithms. We describe how incorporating psychoacoustic weighting to the core algorithm can improve perceptual performance

INTRODUCTION

The concept of embedded-zerotree quantisation was first introduced by Shapiro [1] as Embedded Zerotree Wavelet (EZW) coding for image compression. This was shown to provide compression performance equal to or better than alternative techniques, with very low algorithmic complexity. In addition, the process produces a fully embedded code, where bits in the bitstream are produced in order of importance. The coding process may thus be terminated at any point, yielding a reconstruction quality that is roughly proportional to the length of bitstream produced. An alternative way to consider an embedded code at a certain rate is that it includes all lower-rate codes. This property is very useful for loss recovery and scalability, as shown in [1].

More recently Said and Pearlman [3] have developed an improved zero tree algorithm giving greater compression performance, which they termed Set Partitioning in Hierarchical Trees (SPIHT). Srinivasan and Jamieson have also shown that zero tree quantisation may be used in audio coders with good results [4].

EZW and SPIHT may be characterised by the following steps:

1. Wavelet Transform: correlation between the original image pixels is removed using a hierarchical wavelet decomposition. The transform coefficients are arranged in a two-dimensional matrix.

2. Embedded Zerotree (EZ) Quantisation: transform coefficients are effectively transmitted as bit-planes. However, the EZ process recognises that coefficients at low scales (frequency bands) are generally of greater magnitude than those at higher scales. It uses this property to significantly reduce the number of zero bits which would be transmitted if pure bit-plane encoding were used. The EZ quantisation process is composed of two stages which are repeated successively until the target bit-rate has been met:

- **a. Significance**: compares the coefficient matrix with a threshold level, and produces a sequence of bits or symbols which identifies which coefficients are significant with respect to (ie greater than) the threshold. The threshold is halved following each pass.
- **b. Refinement**: produces a series of bits which halves the uncertainty interval of coefficients which have already been found to be significant.

3. Entropy Coding: the sequence of symbols produced is losslessly compressed using arithmetic coding.

In Section 1, we describe how the differences between images and audio signals affect zerotree coding, and compare the EZW and SPIHT algorithms. In Section 2 we describe the EZK algorithm and show it to have improved performance relative to EZW and SPIHT for uniform decompositions. We also show here how psychoacoustic modelling may be incorporated to improve perceptual performance. Finally we present our conclusions and suggestions for further work in Section 3.

[1] Supported by British Telecommunications PLC.
[2] Supported by the EPSRC.

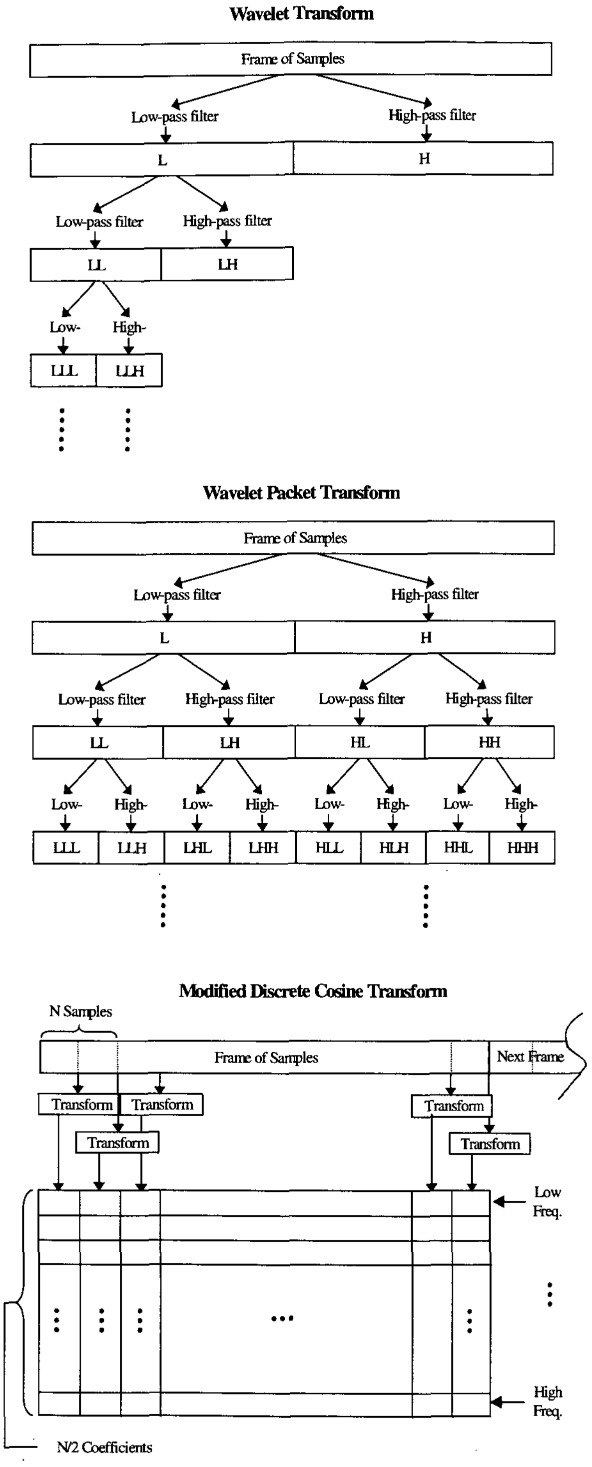

Figure 1: Alternative transforms

1. ZEROTREE ALGORITHMS FOR AUDIO

1.1 Embedded-Zerotree Quantisation

Embedded zerotree algorithms such as EZW and SPIHT were originally developed for image coding using a wavelet transform. In order to use the algorithms in audio coders with Wavelet Packet (WP) or Modified Discrete Cosine Transforms (MDCT), the main differences to consider are as follows:

1. Uniform vs. Non-Uniform Decomposition

The wavelet transform uses non-uniform subband decomposition where the low-pass results of previous half-band filtering operations are further decomposed, but the high-pass coefficients are not subject to further decomposition. By contrast, the wavelet packet approach involves further decomposition of both low- and high-pass filter coefficients at each stage, yielding an overall uniform decomposition. The MDCT also achieves a uniform decomposition but does not take a hierarchical approach, instead providing a transform-domain representation directly for each block of samples within the frame [5]. The differences are illustrated in Figure 1.

Images are two-dimensional, and require a 2-D transform to yield a matrix of transform coefficients in which both dimensions reflect space and scale. This is illustrated in Figure 2, where the highest-frequency subbands are at the bottom right, and the lowest-frequency subbands at the top left.

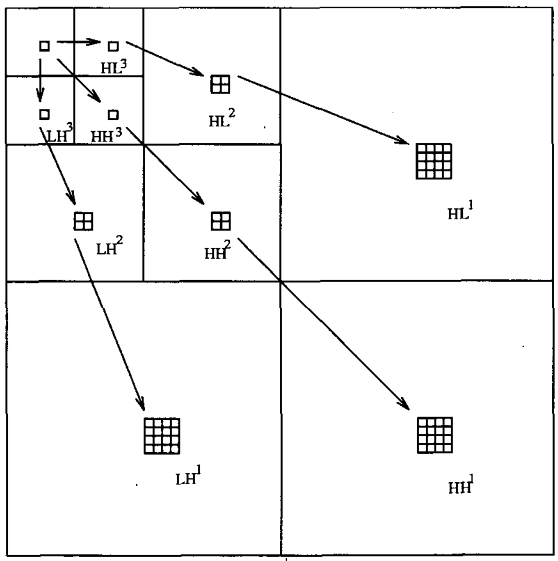

Figure 2: Wavelet transform of an image.

2. Two Dimensions vs. One

All cells denote transform coefficients which represent the same spatial area in the original image, with the arrows indicating 'heredity' - the low-frequency subbands are the 'parents' of the high-frequency subbands. By contrast an audio signal is one-dimensional (in time), and transformation yields a one-dimensional vector of transform coefficients reflecting scale/frequency. However, it is possible to arrange the frequency domain coefficients from block transforms across several time slots in a 2D matrix (Figure 3).

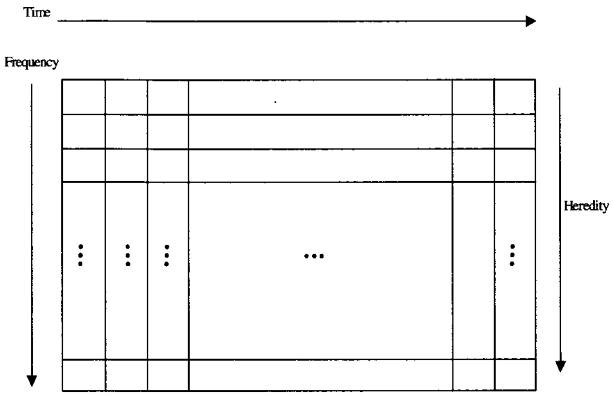

Figure 3: Audio transform coefficients rearranged to show heredity.

Here heredity applies in the frequency domain alone, where all coefficients in the table except the top row (lowest frequency) are 'children' of those at lower frequencies.

The EZW and SPIHT algorithms were adapted for such a coefficient matrix derived from uniform transforms and are now described in turn using pseudo-code.

1.2 EZW Algorithm

EZW codes significance of the matrix of coefficients with respect to a threshold which is halved after each pass. This process produces a stream of symbols from a 4-symbol alphabet. The four symbols are:

- POS: The coefficient is significant with respect to the threshold, and positive.
- NEG: The coefficient is significant with respect to the threshold, and negative.
- IZ: Isolated Zero - the coefficient is not significant, but has descendants which are.
- ZTR: Zero Tree - the coefficient is not significant, and neither are any of its descendants. There is therefore no need to code a symbol for any of these descendants.

The pseudo-code for EZW is:

```
Max=maximum value in coefficient array
n=⌊log2(max)⌋
while n>=0
    thresh=2^n
    Significance Stage (Dominant Pass)
    For each time slot
        If timeslot does not yet contain a zerotree symbol
            For each frequency band, from low to high
                If coefficient magnitude for current band and
                timeslot > thresh
                    Transmit POS or NEG according to sign of
                    coefficient
                    Put value in subordinate list
                    Set matrix value to zero
                Otherwise, if any descendants are > thresh
                    Transmit IZ
                Otherwise,
                    Transmit ZTR
                End
            End
        End
    End

    Refinement Stage (Subordinate Pass)
    For all values in subordinate list,
        Transmit (n-1)th bit of value (where 0th bit is LSB)
    End

    n=n-1
End.
```

1.3 SPIHT Algorithm

The SPIHT algorithm [3] codes significance with a stream of bits rather than symbols, by using a number of lists of pointers to coefficients in the matrix. These are:

List of Significant Pixels (LSP), containing pointers to coefficients which have been found to be significant. This list is initially empty.
List of Insignificant Pixels (LIP), containing pointers to coefficients which are not significant but which have significant descendants. This list initially contains pointers to the top row of the matrix (the parents of all other coefficients).
List of Insignificant Sets (LIS), containing pointers to coefficients the significance of whose *descendants* we wish to test. Its members are flagged as being of type A if we wish to test the significance of its *children* and lower generations, and of type B if we wish to test the significance of its *grandchildren* and lower generations. This list initially contains pointers to the top row of the matrix, and these are all flagged as being of type A.

The algorithm is then:

```
Max=maximum value in coefficient array
n = ⌊log2(max)⌋
while n>=0
    thresh=2^n
    For all members of LIP,
        If coefficient magnitude is significant (> thresh)
            Transmit 1
            Transmit sign bit
            Move pointer to end of LSP
        Otherwise
            Transmit 0
        End
    End

    Significance Stage
    For all members of LIS,
        If type A,
            If coefficient has any significant descendants
                Transmit 1
                If immediate descendant (child) is significant
                    Transmit 1
                    Transmit sign bit
                    Add child to LSP
                Otherwise
                    Transmit 0
                    Add child to LIP
                End
                If coefficient has no grandchildren
                    Remove it from LIS
                Otherwise (significant coefficient must be
                grandchild or lower)
                    Move coefficient to end of LIS and change
                    to type B
                End
            Otherwise
                Transmit 0
            End
        If type B,
            If coefficient has any significant grandchildren,
                Transmit 1
                Remove coefficient from LIS and add child to
                end of LIS as type A
            Otherwise
                Transmit 0
            End
        End
    End

    Refinement Stage
    For all coefficients in LSP except those included in last
    significance pass
        Transmit nth bit of value (where 0th bit is LSB)
    End

    n=n-1
End.
```

1.4 A Comparison of EZW and SPIHT

The main differences between the two algorithms are:

- For EZW, the four symbols produced by the significance stage are POS, NEG, ZTR and IZ. The refinement stage then produces a sequence of bits. SPIHT produces a sequence of bits both to convey significance and for the refinement stage.
- In EZW, for each new value of n (halving of the threshold), scanning for significant coefficients starts again at the top of the matrix (lowest frequency). This is so that coefficients which were previously insignificant can be tested at the new threshold. Conversely the SPIHT algorithm avoids returning to the top of the matrix by use of the LIP, each new loop with a new threshold beginning with a scan of the LIP. Each scan of the coefficient matrix can then commence from the position where the scan at the previous threshold finished.

2. EZK ALGORITHM

The motivation for the development of the EZK algorithm was to improve on the compression performance of SPIHT with uniform transform coders (WP and MDCT). In particular, the stream of bits produced by SPIHT when coding a significant coefficient with insignificant parents seemed longer than necessary. We may illustrate this by example, where the timeslot indicated by the arrow in Figure 4 shows three 'generations' of insignificant coefficients (denoted '-'), followed by a significant coefficient (denoted '*').

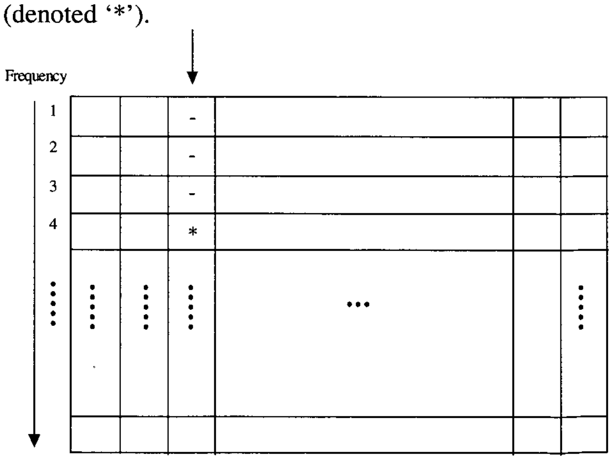

Figure 4: Coding insignificance: an example.

The first coefficient appears in the LIP and also in the LIS as type A. Following the SPIHT algorithm detailed above, initially a 0 is transmitted because row 1 is insignificant in the scanning of the LIP. Then the LIS is scanned, and 1 is transmitted to show that the coefficient in row 1 has significant descendants. Zero is then transmitted to show that the coefficient in row 2 is not significant, and it is added to the LIP. The coefficient in row 1 is then moved to the end of the LIS as type B. When this new entry in the LIS is scanned, a 1 is transmitted because the coefficient in row 1 has significant grandchildren, and the coefficient in row 2 is added to the LIS as type A. This process continues, with coefficients being added to the LIS as type A and type B alternately, until finally the coefficient in row 3 is tested as type A, its child is found to be significant, and a 1 is transmitted to show significance, followed by the sign bit. The code for this process is then '010110111X' (where X is the sign bit for the significant coefficient in row 4), a total of 10 bits.

The complexity of SPIHT may be necessary for WT coding of images, where use of nonuniform transforms leads to 'family trees' where each coefficient can have more than one descendant. However, for our purposes using uniform transforms, a simplified approach may be used. We may apply the tests on the coefficient in row 1 to determine whether it and its descendants are significant, as before. However, the location of a significant coefficient can be more simply conveyed with a run of 0s followed by a 1. Using this approach, the code for the example in Figure 4 becomes '01001X', only 6 bits long. This may be broken down as shown in Table 1.

Bit	Explanation
0	Coefficient in row 1 is not significant
1	Coefficient in row 1 has significant descendants
0	Coefficient in row 2 is not significant
0	Coefficient in row 3 is not significant
1	Coefficient in row 4 is significant
X	Sign bit for coefficient in row 4

Table 1: Coding of insignificance using EZK

The lists employed by EZK are:

- Sig_coeffs: this is analogous to the LSP in SPIHT. It contains pointers to coefficients which have been found to be significant, and is initially empty.
- Insig_coeffs: analogous to the LIP in SPIHT. It contains pointers to coefficients which are not significant, but which have significant descendants. It is initially filled with pointers to the top row of the matrix (the parents of all other coefficients).
- Ts_ptr. This is analogous to the LIS in SPIHT. It contains a pointer for each timeslot (column) in the coefficient matrix. Each one points to the coefficient in that column among whose descendants we shall next be checking for significance. In general, these will point to progressively lower rows (higher-frequency coefficients) as the threshold decreases. We define the *remainder* of a column as the elements in that column from the member pointed to by Ts_ptr downwards.

The pseudo-code for the entire EZK algorithm is then:

```
Max=maximum value in coefficient array
n = ⌊log2(max)⌋
while n>=0
    thresh=2^n
    For all members of Insig_coeffs,
        If coefficient magnitude is significant (> thresh)
            Transmit 1
            Transmit sign bit
            Move pointer to end of Sig_coeffs
        Otherwise
            Transmit 0
        End
    End

    Significance Stage
    For each time slot (column) of coefficient matrix,
        While remainder of column has any significant
        members,
            Transmit 1
            For each member of column remainder
                If member is significant
                    Transmit 1
                    Transmit sign bit
                    Add member to Sig_coeffs
                    Set Ts_ptr(column) to point to row below
                    member
                Otherwise
                    Transmit 0
                    Transfer to Insig_coeffs
                End
            End
        End
        Transmit 0  (no more significant coefficients in this
        column)
    End

    Refinement Stage
    For all coefficients in Sig_coeffs,
        Transmit nth bit of value
    OR Transmit (n-1)th bit of value
    (Allows EZW-style or SPIHT-style refinement)
    End

    n=n-1
End.
```

2.1 Compression Performance

All three algorithms have been implemented in a mathematical modelling program, and their compression performance compared on a range of real audio segments. For EZW we have used a 2-bit code for the four symbols POS, NEG, ZTR and IZ, with no entropy coding [1].

The results in Table 2 show compressed file sizes (in bits) for a sample audio frame resulting from the EZW, SPIHT and EZK algorithms. The 'No. of Passes' column indicates how many times the main loop has iterated, a higher value indicating a smaller final threshold value and hence more accurate coding of matrix coefficients. These results show that the EZK algorithm gives the best compression performance of the three algorithms.

No. of Passes	EZW	SPIHT	EZK
14	21181	12246	9314
10	12458	7863	5146
5	1527	1384	1073

Table 2: Compressed file sizes in bits for alternative EZ algorithms.

2.2 Psychoacoustic Weighting

The EZK quantisation algorithm has been incorporated into a simple audio codec, as described in [2]. This employs an MDCT or wavelet packet transform, and EZK quantisation of the resulting coefficient matrix. The algorithm allows transform lengths from 64 to 1024-points within an overall 1024-sample frame length, thus yielding coefficient matrices with different aspect ratios. We found in [2] that coding performance using an MDCT with a fixed transform length is generally optimised with a 256-point transform, corresponding to a coefficient matrix with 128 rows and 8 columns.

The performance of this codec was found in [2] to be superior to MPEG-Audio Layer I and comparable to Layer II, despite the much lower complexity of EZK. Both MPEG alternatives employ psychoacoustic modelling to perform bit allocation such that quantisation noise is least perceptible. Conversely the raw EZK algorithm effectively allocates bits to the highest-magnitude coefficients, the effect of which is to distribute quantisation noise uniformly among frequency bands (ie the quantisation noise tends to a white spectrum). Such a noise distribution may be suboptimal from a psychoacoustic perspective.

It is clearly not possible to explicitly allocate bits in the manner of MPEG coders [8]. However, it is possible to use a psychoacoustic model to guide a weighting of the frequency bands, as suggested in [7]. The encoder block diagram using this idea is given in Figure 5. This scheme ensures that the coefficients presented to the EZK algorithm are reduced in relative magnitude where the masking effect is greatest, and increased in relative magnitude where masking is least. When the coefficients are subject to an inverse weighting at the decoder, the overall quantisation noise characteristic will then follow the masking threshold rather than being uniform with frequency.

The psychoacoustic model employed in our experimental coder is broadly guided by the principles outlined in [9]:

- an FFT is taken of the audio frame, and this is converted into a power spectrum
- a critical band analysis is performed to determine the energy in each critical band
- a spreading function is applied in order to take account of masking across critical bands
- the result is compared to the absolute threshold function across all frequencies, the maximum of which is output as the overall masked threshold function.

Psychoacoustic weighting was implemented as shown in Figure 5 and tested using a number of signals across a range of bitrates.

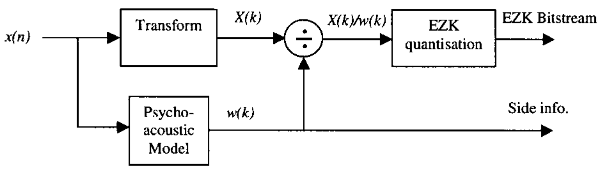

Figure 5: EZK encoder with psychoacoustic weighting.

The results for a number of MDC transform lengths at a bitrate of 64 kbs^{-1} are shown in Table 3, where a '√' indicates the reconstruction with psychoacoustic weighting sounds better than that without (uniform noise), while a 'X' indicates that it sounds worse, and '-' that it sounds about the same.

Signal	64-point	256-point	512-point
Castanets	X	√	√
Voice	X	√	√
Harpsichord	X	√	-
Pitch Pipe	X	X	X

Table 3: Results for EZK codec with psychoacoustic weighting.

The results show that psychoacoustic weighting works well for the 256-point transform, although results at other transform lengths are not as good as expected. This has led to the supposition that the psychoacoustic weighting process is somehow acting to make the EZK process less efficient. In particular, if the weighting has the effect of increasing the magnitude of higher frequency coefficients relative to lower frequency coefficients, then weighting will tend to increase the average length of runs of zeroes when coding insignificance in the coefficient matrix. The EZK algorithm therefore becomes less efficient.

This hypothesis was tested by measuring the proportion of the total coded file size taken up by runs of zeroes used to code insignificance. The results are presented in Table 4, where the results from Table 3 are repeated under the 'Auditn' columns. The 'White' columns correspond to percentages of the coded files which are taken up by zero runs for coders without psychoacoustic weighting, while 'PW' data corresponds to percentages for coders with psychoacoustic weighting. The results indicate that the amount of the file dedicated to runs of zeroes is greater on average with psychoacoustic weighting, thereby impairing the compression performance of the EZK algorithm.

	64-point		
Signal	Better	No pa	With
Castanets	X	12.1	12.5
Voice	X	14.9	16.8
Harp	X	10.6	15.5
Pitch pipe	X	5.8	8.1
	256-point		
	Better	No pa	With
Castanets	√	29.5	29.1
Voice	√	36.1	37.0
Harp	√	21.8	27.2
Pitch pipe	X	28.9	27.3
	512-point		
	Better	No pa	With
Castanets	√	29.7	30.3
Voice	√	39.1	39.4
Harp	-	23.0	30.7
Pitch pipe	X	29.7	34.5

Table 4: Percentages of coded files taken up by zero runs.

3. CONCLUSIONS AND FURTHER WORK

In this paper we have reviewed the differences between image and audio coding when using zerotree quantisation algorithms such as EZW and SPIHT. We described how EZW and SPIHT may be applied to an audio signal that has been transformed using a uniform decomposition, and introduced a more efficient algorithm (EZK) for use with such uniform decomposition coders. Central to the new algorithm is the use of zero runs to convey insignificance, and such a technique may also yield interesting results in non-uniform decompositions such as the WT.

We have shown that although zerotree algorithms do not permit explicit bit allocation, it is possible to apply a psychoacoustic weighting which results in a significant improvement in perceptual quality for some signals, although the weighting was also shown in general to make the EZK algorithm less efficient. One possibility to overcome this problem would be to modify the weighting function applied in order to bound any efficiency loss incurred.

REFERENCES

[1] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", IEEE Trans. on Sig. Proc., vol. 41, no. 12, pp. 3345-3462 (1993 Dec.).

[2] B. Leslie and M. Sandler, "Joint Source and Channel Coding for Internet Audio Transmission", presented at the 106th AES Convention, Munich, May 1999 (preprint 4932).

[3] A. Said and W. A. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Trans. Circuits Sys. Video Tech., vol. 6, no. 3, pp. 243-250 (1996 June).

[4] P. Srinivasan and L. H. Jamieson, "High-Quality Audio Compression Using an Adaptive Wavelet Packet Decomposition and Psychoacoustic Modeling", IEEE Trans. Signal Proc., vol. 46, no. 4, pp. 1085-1093 (1998 Apr.).

[5] J. P. Princen and A. B. Bradley, "Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation", IEEE Trans. Acoust. Speech Sig. Proc., vol. 34, no. 5, pp. 1153-1161 (1986 October)

[6] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic Coding for Data Compression", Communications of the ACM, vol.20 (1987 June).

[7] H. Malvar, "Enhancing the Performance of Subband Audio Coders for Speech Signals", IEEE Int. Symp. Circuits Sys. (1998).

[8] "Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbps – Part 3:Audio" ISO-IEC 11172-3 (1993 Aug.).

[9] J. D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria", IEEE Journal on Selected Areas in Communications, Vol. 6, No. 2, pp. 314-323 (1988 Feb.).