Audio Engineering Society

Convention Paper 6749

Presented at the 120th Convention
2006 May 20–23 Paris, France

Scalable Bitplane Runlength Coding

Chris Dunn

Scala Technology Ltd, London, UK

chris.dunn@scalatech.co.uk

ABSTRACT

Low-complexity audio compression offering fine-grain bitrate scalability can be realised with bitplane runlength coding. Adaptive Golomb codes are computationally simple runlength codes that allow bitplane runlength coding to achieve notable coding efficiency. For multi-block audio frames, coefficient interleaving prior to bitplane runlength coding results in a substantial increase in coding efficiency. It is shown that bitplane runlength coding is more compact than the best known SPIHT arrangement for audio coding, and achieves coding efficiency that is competitive with fixed-rate quantisation.

1. INTRODUCTION

Audio coding algorithms with bitrate scalability allow an encoder to transmit or store compressed audio data at a high bitrate and decoders to successfully decode a lower-rate bitstream contained within the high-rate code. Scalability has become an important aspect of low bitrate audio coding, particularly for multimedia applications where a range of coding bitrates may be required, or where channel bitrate fluctuates. Fine-grain scalability, where useful increases in coding quality can be achieved with small increments in bitrate, is particularly desirable.

Audio coding with fine-grain bitrate scalability allows real-time streaming with low buffer delay and uninterrupted service in the presence of channel congestion, and yields the most efficient use of available channel bandwidth. Scalability is also useful in archiving and personal media storage, where a program item may be coded at the highest bitrate required and stored as a single file, rather than storing many coded versions across the range of required bitrates. The potential reduction in overall storage requirement achieved by using scalable formats can be significant when maintaining large song libraries. As well as the saving in storage, bitrate scalability avoids cumulative archiving degradation that can occur due to recoding.

While fine-grain bitrate scalability can be useful, it is important that it is achieved without significant coding efficiency penalty relative to fixed-bitrate systems, and with low computational complexity. In this paper, bitplane coding is examined as a simple and effective method of achieving bitrate scalability. Bitplane coding systems where runlength codes identify newly-significant transform coefficient locations within

bitplanes are investigated in detail. Bitplane runlength coding with adaptive Golomb runlength codes is shown to offer coding efficiency that outperforms previously reported SPIHT-based bitplane implementations, and is competitive with the most effective fixed-rate quantisation approaches. The results of listening tests designed to assess the coding efficiency of a practical demonstration codec employing bitplane runlength coding are reported.

2. BITPLANE CODING

Audio compression algorithms typically use some form of transform coding where the time-domain audio signal is transformed to the frequency domain before quantisation, entropy coding and frame packing to a bitstream (Fig. 1). A psychoacoustic model determines a target noise shaping profile, which is used to allocate bits to transform coefficients such that quantisation errors are least audible to the human ear. In a conventional fixed-bitrate encoder the bit allocation is typically achieved with a recursive algorithm that attempts to meet the noise-shaping requirement within a bitrate constraint [1]. It is the bit allocation and quantisation stages of the coding process that are most affected by scalability issues - the time-frequency transform and psychoacoustic model in a scalable coder can be similar to a fixed-rate design.

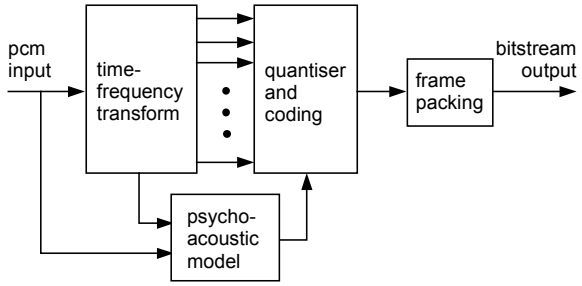

Fig. 1. Perceptual transform encoder for audio compression.

One approach to achieving scalability is the 'error-feedforward' arrangement where a core coder produces the lowest embedded bit rate, and subsequent layers progressively reduce the error due to the core [2]. However, a significant amount of side information is associated with each layer which can reduce coding efficiency, and the number of possible decoding rates is limited to the number of layers.

An alternative approach to achieving scalability is ordered bitplane coding of transform coefficients, where in each frame coefficients are arranged in sign-magnitude format and magnitude bits are coded in order of significance, beginning with the most-significant bits (MSB's) and progressing to the LSBs. This results in fully embedded coding where the bitstream at a certain rate contains all lower-rate codes, allowing low-complexity bitrate scaling by simply truncating coded frames. A related advantage is that variable-bitrate coding can be easily implemented by truncating each coded frame at an appropriate point before packing the bitstream. Bitplane coding exhibits fine-grain scalability in contrast to the coarse granularity offered by error-feedforward systems, and generates an ordered bitstream syntax which can simplify source-coding efforts to increase error resilience. In addition to these benefits, bitplane coding can yield a significant increase in encoding speed since quantisation typically requires a single scan through the transform coefficients for each frame, as opposed to the recursive bit allocation search executed in fixed-rate coding.

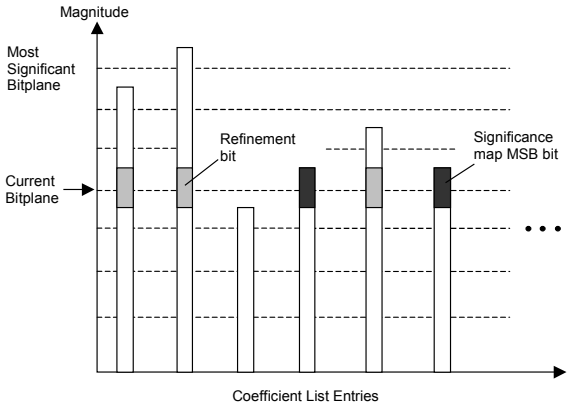

Fig. 2. Two-stage bitplane coding distinguishes significance map MSB bits from refinement LSB bits.

Modern bitplane coding techniques have their routes in image coding. Shapiro [3] described a bitplane coding algorithm for two-dimensional wavelet transform coefficients termed zero-tree quantisation, where each bitplane is coded in two stages - a significance map which identifies coefficients with MSBs positioned within the current bitplane, and a refinement stage which outputs LSB information for previously identified significant coefficients (Fig. 2). In Shapiro's algorithm the refinement bits are coded relative to a moving threshold that is lower than that used to code the significance map. The two-stage approach was later

refined by Said and Pearlman [4] with the SPIHT algorithm, where the significance map and refinement bits of each bitplane pass are coded with respect to the same threshold. This is the preferred approach, since with a common relative threshold quantisation error intervals tend to be reduced more uniformly as bitplane coding progresses, and is also preferred from a rate-distortion perspective [5, Section 2].

Fig. 3 shows a general two-stage bitplane encoding process, which can be used to code audio transform coefficients on a frame-by-frame basis. Because the quantisation error due to bitplane coding tends to a white spectrum, perceptually-appropriate quantisation error shaping can be achieved by weighting banded coefficients prior to bitplane coding (step s0). Banded-weight side information is output for each frame to enable a complementary inverse weighting at the decoder. At step s1 a bit allocation variable is initialised to the required size of the coded frame, and is subsequently updated as coding progresses. Scaled and weighted floating-point transform coefficients $x(k)$ are represented in sign-magnitude format, and at step s2 the largest coefficient magnitude within the frame $|x|_{\max}$ is determined and an initial threshold level T set such that

$$T \leq |x|_{\max} < 2T. \qquad (1)$$

T determines the current bitplane level in the encoding process, and the initial threshold value is output as side information so that a decoder can begin decoding at the correct bitplane level.

For each bitplane, coefficients are scanned at step s3 to locate those with magnitudes equal to or exceeding T – these coefficients are significant with respect to the current threshold. When a significant coefficient is located, the component of the significance map describing the location is coded and output to a buffer, followed by a coefficient sign bit. When significance map coding is complete, a refinement stage s4 is executed where refinement bits corresponding to the current threshold level are output to the buffer for all significant coefficients previously identified in more-significant bitplanes. T is halved at step s5 and coding progresses to the next bitplane. This process is repeated for progressively less significant bitplanes until the bit allocation for the frame is reached, at which point coding terminates (step s6), and at step s7 coded frame data in the output buffer is written to the bitstream.

Since in the refinement stage the probability of coding LSBs as '1' or '0' is approximately equal, refinement bits can be output directly without additional entropy coding, and most of the measurable differences in coding efficiency that arise between different bitplane coding algorithms occur due to differences in the way the significance map is coded. An efficient coding process will identify the MSB locations in each bitplane using as few bits as possible.

A general bitplane decoding algorithm has a similar structure, for each bitplane alternating between significance map decoding to identify the positions of newly-significant coefficient locations, and a refinement stage which decodes LSB bits for previously-identified significant coefficients.

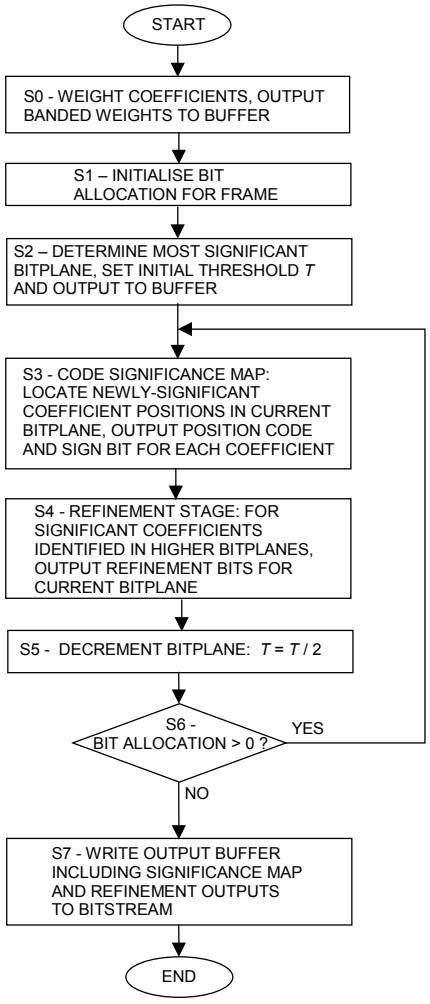

Fig. 3. Bitplane encoding process.

An early example of audio bitplane coding is provided by Karelic and Malah [6] who adapt Shapiro's zero-tree approach for use with a non-uniform wavelet packet decomposition. Bitplane coding is also the basis of Bit-Sliced Arithmetic Coding (BSAC) [7], where effectively arithmetic coding is used for significance map coding. More recent examples of bitplane-based audio coding have been described by Lu and Pearlman [8], Dunn [9], Zhou et al. [10], and Li [11].

3. CODING EFFICIENCY

A useful measure of coding efficiency for any quantisation scheme, fixed-rate or scalable, is to record the proportion of total bitrate allocated to directly coding coefficients that have been quantised to non-zero values. Consider a coder with a frame length of K samples; following time-to-frequency transformation with critical sampling, the quantiser is presented with a set of K coefficients to quantise and code for each frame. In Section 2 we noted the significance map identifies the positions of non-zero coefficient MSBs within each bitplane. Let the total number of bits allocated to code a frame = BA, the number of bits used to code the significance map = SM, with the remaining bits used to code non-zero coefficients NZ:

$$BA = SM + NZ. \qquad (2)$$

Suppose that *nsig* coefficients are quantised to non-zero integer values $q_{\mathrm{NZ}}(i)$, that is *nsig* out of K coefficients have been found to be significant. Noting that NZ consists of magnitude and sign information for non-zero coefficients, and does not include information regarding the *positions* of $q_{\mathrm{NZ}}(i)$ within K, the minimum number of bits required to code $q_{\mathrm{NZ}}(i)$ is given by

$$NZ = \sum_{i=0}^{nsig-1} \left\lceil \log_2\left(2\left|q_{\mathrm{NZ}}(i)\right|+1\right)\right\rceil \quad . \qquad (3)$$

This expression for NZ assumes no entropy coding of significant coefficient values, and is similar to Johnston's description of perceptual entropy [12, Sec. 2.4.3].

Efficient coding requires compact significance map coding, leaving as many bits as possible from the available bit allocation for the frame to code non-zero coefficients, and hence maximise *nsig*. One measure of coding efficiency η is therefore obtained by calculating the proportion of total bitrate allocated to NZ:

$$\eta = \frac{1}{BA} \sum_{i=0}^{nsig-1} \left\lceil \log_2\left(2\left|q_{\mathrm{NZ}}(i)\right|+1\right)\right\rceil \quad . \qquad (4)$$

η indicates how efficiently quantised coefficient data is 'supported' within the generated bitstream. Shapiro has shown that at low bitrates a large proportion of bitrate must be allocated to the significance map, even with the most efficient coding scheme possible [3, Sec. 3], hence we would expect η to be lower than 0.5. In practice η will tend to vary from frame to frame, and averaging η over the duration of a test piece provides a useful measure of algorithm coding efficiency that can be used to compare the relative efficiencies of different quantisation schemes.

While graphical plots of η as a function of bitrate are effective at highlighting differences between quantisation algorithms, perhaps a more relevant measure of coding efficiency is obtained by computing the average number of significant coefficients *nsig* identified in each frame. The maximum possible value for *nsig* is clearly equal to K, but typically less than 50% of all available coefficients are coded to non-zero values, even at higher bitrates. For bitplane quantisers *nsig* tends to increase almost linearly with bitrate, and so comparing the number of significant coefficients coded by alternative quantisation algorithms allows equivalent bitrate advantages / penalties to be computed.

By considering the different quantiser functions for scalable bitplane coding and conventional fixed-rate coding, we now show that *nsig* can also be used to directly compare coding efficiencies of bitplane and fixed-rate systems. The general bitplane encoding algorithm described in Section 2 implements uniform quantisation with a dead-zone around zero, resulting in integer quantised coefficient values given by

$$q(i) = \mathrm{sgn}(x(i)) \left\lfloor \frac{x(i)}{T_{\mathrm{F}}} \right\rfloor \quad , \qquad (5)$$

where T_{F} is the final threshold value used to encode each coefficient $x(i)$. In the bitplane decoder, for each significant coefficient identified there exists a range of uncertainty concerning the reconstructed value, which depends on the threshold T_{F} corresponding to the final significant bit decoded. A simple reconstruction

approach is to set each significant coefficient to the center of its uncertainty interval:

$$\overline{x(i)} = \begin{cases} 0, & q(i)=0 \\ T_F[q(i)+0.5], & q(i)>0 \\ T_F[q(i)-0.5], & q(i)<0 \end{cases} . \quad (6)$$

The combined bitplane encoder / decoder quantiser function is shown in Fig. 4(a), where the width of the zero-amplitude bin is twice that of other bins, in contrast to the uniform bin width distribution of the mid-tread quantiser typical of fixed-bitrate coding [Fig. 4(b)].

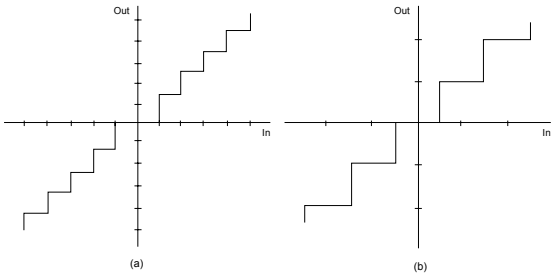

Fig. 4. Quantiser functions with equal zero-bin widths: (a) bitplane quantiser, and (b) uniform mid-tread quantiser.

If on average a bitplane coder identifies the same number of significant (non-zero) coefficients as an otherwise-identical fixed-rate coder with uniform quantisation, then the average zero bin width for the two quantisers will be equal. Due to the reduced non-zero bin width, the bitplane quantiser will introduce a smaller average quantisation error for non-zero coefficients. Hence we can say that if a bitplane quantiser identifies the same number of non-zero coefficients *nsig* as a fixed-rate quantiser with uniform quantisation, then the bitplane quantiser will code to a precision at least as good as the fixed-rate quantiser.

4. FIXED-RATE REFERENCE CODER

We can use the evaluation framework established above to simulate the coding efficiency of a fixed-bitrate coder, and use this as a reference to gauge the performance of scalable quantisation algorithms. The fixed-rate reference coder follows the generic structure outlined in Fig. 1. A fixed-length modified discrete cosine transform (MDCT) is used with a frame length K of 1024 samples and a 50% overlapping window length of 2048 samples; this transform arrangement is similar to the long block mode of MPEG-2/4 AAC [13]. The transform output is partitioned into 32 bands where the band boundaries follow a compressed critical-band law. Coefficients in each band are weighted using banded scalefactors prior to quantisation in order to achieve a perceptually-appropriate shaping of quantisation noise. The scalefactors are calculated using a custom psychoacoustic model and quantised in steps of 3 dB.

To calculate coding efficiency η for this fixed-rate system we use a result derived by Watson and Truman [14] who measure the behaviour of the multi-dimensional Huffman coding algorithm used in AAC [13], and show the average bitrate requirement for quantised coefficients in each frame is closely approximated by the 1-dimensional entropy of the quantised coefficient values. To simulate this system and find the quantised coefficient values for each frame of a given signal at a target bitrate, a global gain factor applied to the band weights is set recursively on a frame-by-frame basis (Fig. 5). Using this approach, coding efficiency is estimated as a function of bitrate by averaging η at each bitrate across 3 single-channel 44.1 kHz sampled test pieces (harpsichord, pitchpipe, and female voice with background music). The results are shown in Fig 6 marked 'AAC' - as expected, coding efficiency is less than 50 % even at higher bitrates.

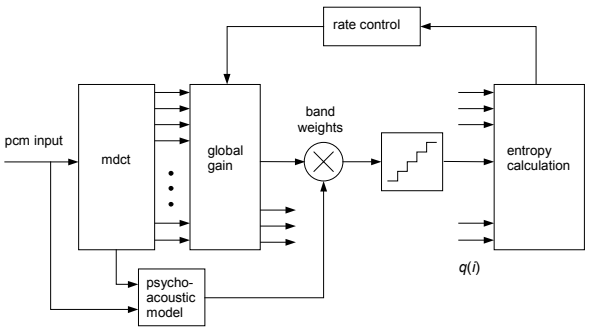

Fig. 5. Simulating AAC coefficient values for coding efficiency estimation.

5. SPIHT

In the bitplane coding algorithm pioneered by Shapiro for use in image coding [3], a zero-tree hierarchy is defined for significance map coding. Coefficients are scanned sequentially following the order of the hierarchy, where each coefficient at a given frequency (the *parent*) can be related to a set of coefficients at higher frequencies (the *children*), and no child node is scanned before its parent. For two-dimensional image

coding each parent coefficient typically has four children, and the zero-tree hierarchy can efficiently code significance maps because if a low-frequency coefficient is found to be insignificant, then many higher-frequency coefficients with the same spatial location are also likely to be insignificant.

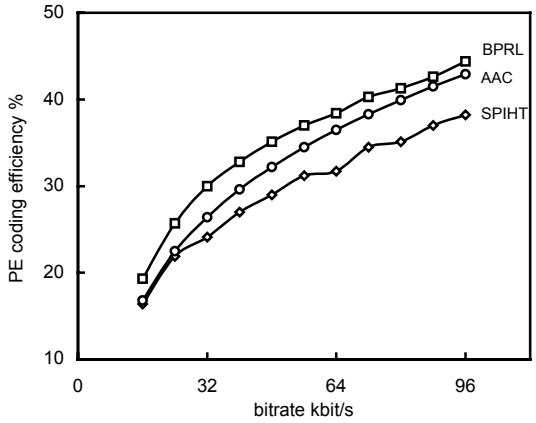

Fig. 6. Coding efficiency η as a function of bitrate for 3 coding schemes.

Set Partitioning in Hierarchical Trees (SPIHT) due to Said and Pearlman [4] is a refinement of the zero-tree algorithm, where again compact significance map coding is dependent upon each parent in the hierarchy having many children. Effective use of SPIHT for audio bitplane coding requires a solution to the problem of how to map a one-to-many hierarchy to a 1-dimensional transform array typical with audio coding. Fig. 7 shows one possible hierarchy where each parent coefficient has 4 child coefficients clustered together in frequency, with the exception of the dc coefficient which has no offspring. This arrangement was first reported in [9], and later studied in [15]. Here we can relate frequency indices for the 4 child coefficients $c_0 \ldots c_3$ to their parent frequency index p:

$$\begin{aligned} c_0(p) &= 4p \\ c_1(p) &= 4p+1 \\ c_2(p) &= 4p+2 \\ c_3(p) &= 4p+3 \end{aligned} \qquad (7)$$

$$\text{for } 1 \le p < \frac{K}{4}.$$

More generally for a hierarchy 'fanout' N, where N is an integer power of 2,

$$c(p) = Np + \{0,\ N-1\} \qquad (8)$$

$$\text{for } 1 \le p < \frac{K}{N}.$$

Using otherwise identical coding conditions, simulation results shown in Fig. 6 suggest SPIHT-based scalable bitplane coding with $N = 4$ achieves coding efficiency somewhat lower than the fixed-rate AAC reference. Direct comparisons of the average number of significant coefficients *nsig* coded across the bitrate range 32 - 96 kb/s (sampled in steps of 8 kb/s) indicates SPIHT suffers an equivalent bitrate penalty of 9% relative to AAC (Table 1). This result is consistent with previously reported subjective comparisons between AAC and SPIHT [16].

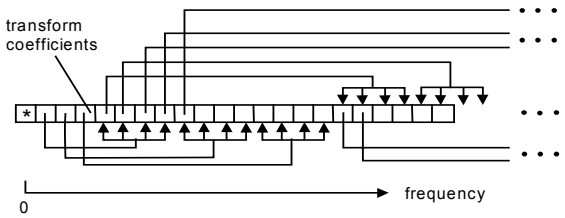

Fig. 7. SPIHT hierarchy for use with uniform decomposition audio transforms, where each parent coefficient has 4 offspring.

Coding Scheme	Fixed-rate / Scalable	Relative coding efficiency %
SPIHT	Scalable	91
AAC	Fixed-rate	100
BPRL	Scalable	104

Table 1. Relative coding efficiencies averaged across bitrate range 32 - 96 kb/s.

6. BITPLANE RUNLENGTH CODING

For audio bitplane coding an effective low-complexity alternative to SPIHT coding of the significance map is to runlength code the MSB locations within each bitplane. Several references to bitplane runlength coding can be found in the literature.

Delogne and Macq [17] describe a pseudo-bitplane coding scheme for video applications, where MSB locations within each bitplane are coded using runlength codes and all LSB bits for a newly-identified significant coefficient are coded immediately following the MSB. Because refinement bits are not output progressively on a bitplane-by-bitplane basis, the resultant bitstreams are not optimally embedded.

The JPEG image compression standard [18] includes a progressive coding mode where an initial coefficient scan codes the 4 most significant bitplanes, followed by refinement scans which increase the coded image resolution in single-bitplane steps. MSB locations within the refinement bitplanes are identified using runlength codes which are then Huffman encoded, and this significance map data is interleaved with refinement bits of significant coefficients identified in more significant bitplanes.

Shapiro [3] briefly considers runlength coding of the significance map in two-stage bitplane coding, but notes that for image wavelet transform coefficients zero-tree coding results in more compact significance map representations.

Ordentlich and co-workers [5] provide a more detailed study of bitplane runlength coding of image wavelet coefficients. Here adaptive Golomb codes are used to code the zero-run lengths between bitplane MSB locations. Malvar [19] further investigated the use of adaptive Golomb codes with image bitplane coding, focusing on the adaptive runlength code due to Langdon Jr [20].

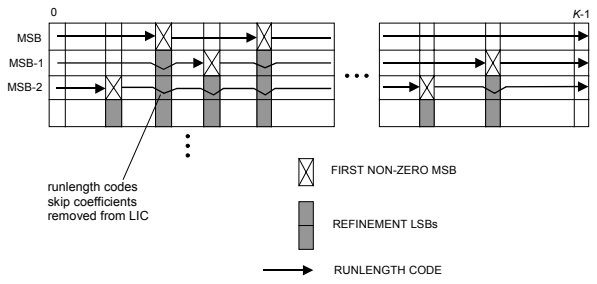

Fig. 8. Bitplane runlength coding.

Given a frame of K audio transform coefficients $x(k)$ arranged in sign-magnitude format, a two-stage bitplane runlength coding algorithm includes the following steps (Fig. 8):

- form two coefficient lists: a list of insignificant coefficients (LIC, initially containing all coefficients in the frame), and a list of significant coefficients (LSC, initially empty)
- code the most-significant bitplane level for all LIC members = $\lfloor \log_2 |x|_{max} \rfloor$
- for each bitplane, beginning with the most significant bitplane:
  - code the significance map:
    - runlength code the positions of newly-significant LIC members, ie those coefficients whose MSB is located within the current bitplane
    - when a LIC member is found to be newly significant:
      - output its sign
      - remove this coefficient from the LIC and add to the LSC
  - code refinement bits:
    - for all LSC members added in previous more-significant bitplanes, output the LSB corresponding to the current bitplane
- terminate coding when either the bit allocation for the frame is used or target coding resolution achieved

6.1. Adaptive Golomb Codes

A useful runlength code is the Golomb code with parameter g, where non-negative runlength r is coded as 2 components – a prefix $\lfloor r / g \rfloor$ coded in unary, followed by suffix [r mod g] coded in binary [21]. A particularly simple form of Golomb code, sometimes known as Rice codes, occurs when $g = 2^w$ for some integer wordlength $w \geq 0$ - here r can be coded by removing the w least-significant bits from r, coding the remainder as a unary prefix, and appending w binary LSB's. For example, if $r = 9$ and $w = 2$, then the Golomb-Rice code for r is '00101' – here the prefix is '001' = 8, and the remainder is '01' = 1.

The coding efficiency achieved using Golomb-Rice codes to runlength code significant coefficient locations in a bitplane depends on the code wordlength w and the runlength distribution. w can be set to a fixed value which on average results in the most compact code across many frames of a test item. Alternatively w can be optimised for each frame, and sent as side information at the start of the frame so that a decoder can correctly interpret the coded bitplane data. Yet another approach [17] is to optimise w for each bitplane

of each frame, and send the appropriate side information at the start of each bitplane.

An alternative to explicitly coding wordlength side information in the bitstream is to make the Golomb-Rice code adaptive in the sense that w varies as a function of previously coded bitplane data – that is, backwards-adaptive runlength coding. Malvar [19] uses the adaptive code due to Langdon Jr [20] for bitplane runlength coding of image transform coefficients, where each '0' in the unary-coded prefix causes the wordlength w to increment, and w decrements following the binary-coded suffix. Many other adaptation strategies exist [5]. In general the advantages of using adaptive Golomb-Rice codes to scan for significant list entries includes the simplicity and computational efficiency of the codes, and also the efficiency with which the codes can adapt to changing runlength statistics within a list. This results in relatively compact bitplane coding without the use of wordlength side information.

For audio transform coefficient bitplanes the average spacing between significant LIC entries tends to increase with frequency, particularly for more-significant bitplanes coded earlier in the frame. When using adaptive Golomb-Rice coding to runlength code bitplane MSB locations, coding efficiency is significantly improved by resetting the Golomb-Rice wordlength to a small value at the beginning of each bitplane scan. In practice with typical frame lengths, resetting w to 0 or 1 can increase overall bitplane runlength coding efficiency by approximately 10% compared to coding with an unconstrained wordlength.

The effectiveness of adaptive Golomb-Rice codes is indicated when comparing bitplane runlength coding efficiency with an optimised adaptive code against a Golomb-Rice code with a wordlength that is fixed (to an optimal value) across all bitplanes and frames. With 44.1 kHz sampled test items, allowing the Golomb-Rice wordlength to adapt improves average coding efficiency across the bitplane range 32 - 96 kb/s by a substantial 40%.

6.2. End-of-Run Codes

When fixed- or adaptive- Golomb-Rice codes are used to scan bitplanes for newly-significant coefficients, coding the end of the bitplane following the final significant coefficient can be simply achieved by outputting repeated prefix '0's until the end of the LIC is passed. When a decoder receives coded data and the current LIC position passes the known list length, all remaining coefficients following the last significant position are marked as insignificant and decoding of the current bitplane terminates. These end-of-run codes are particularly compact when an adaptive Golomb-Rice runlength code is used. For example, with the runlength wordlength adaptation rule due to Langdon Jr [20] and a maximum list length of 1024 entries, repeated-prefix end-of-run codes require a maximum of eleven (and typically far fewer) prefix bits.

An alternative end-of-run coding method described by Leslie et al. [22] is to output a single-bit flag following each MSB location coded, to indicate the existence of further newly-significant coefficients within the bitplane. When '0' is output to indicate no further MSB locations within the bitplane, significance map coding for the current bitplane ceases. The motivation behind such a scheme is that at lower bitrates most significant coefficients are located at lower frequencies, and long runs of insignificant high-frequency coefficients can be efficiently coded by outputting a single bit. However, a simple calculation indicates that under typical coding conditions this technique is less efficient than the repeated-prefix method. At a typical bitrate of 64 kb/s, an efficient coder operating in long block mode (K = 1024) at 44.1 kHz sample rate will on average identify approximately 170 significant coefficients within 8 bitplanes. Hence on average each bitplane contains 20 newly-significant coefficients, and single-bit flags collectively require 20 bits to code the end-of run code for the bitplane.

6.3. RVLCs

An interesting aspect of fixed-wordlength and some adaptive Golomb-Rice codes is that with minor modification they can form reversible variable length codes (RVLCs), where the code prefix can be decoded in either a forward or reverse direction. Favourable code length distributions allow bitplane runlength coding with RVLCs to achieve good coding efficiency, while bi-directional decoding can confer advantages of improved coding robustness with error-prone transmission channels [23]. An audio bitplane coder where RVLCs are used to code bitplane runlengths is described by Zhou et al. [10].

6.4. Short Block Interleaving

When the time-frequency transform outputs multiple coefficient blocks across a frame, for example with a block-switched coder operating in short-block mode under transient signal conditions, a reordering process prior to bitplane runlength coding which interleaves coefficients in a data-independent manner can significantly increase coding efficiency.

The reordering step groups together coefficients with the same frequency index within the LIC. Since coefficients with the same frequency index tend to be of similar magnitude, reordering has the effect of clustering significant coefficient locations within LIC scans. This results in longer runs of insignificant coefficients which can improve coding efficiency when runlength coded, particularly when using adaptive runlength codes. A similar process is described by Malvar [19] for reordering image wavelet transform coefficients prior to bitplane coding, while short-block interleaving is also used in fixed-rate MPEG-2 AAC audio coding [24].

Consider a coder with a frame length of K time-domain input samples and an MDCT window length of $2M$. The number of coefficient blocks output for each frame $B = K / M$, each block containing M unique coefficients ranging from dc to half the sampling frequency. Addressing the transform coefficients with a time index b, and frequency index m,

$$\begin{aligned} \text{MDCT output} &= x[b][m], \\ \text{where} \quad b &= 0 \ldots B-1 \\ m &= 0 \ldots M-1. \end{aligned} \tag{9}$$

With a frame length of 1024 samples, a typical short-block frame contains $B = 8$ blocks each with $M = 128$ coefficients.

The reordering process prior to bitplane coding can be described as

$$\begin{aligned} LIC(k) &= x[b][m], \\ \text{where} \quad k &= 0 \ldots K-1, \\ b &= k \bmod B, \\ m &= \left\lfloor \frac{k}{B} \right\rfloor. \end{aligned} \tag{10}$$

When a frame contains only a single block of MDCT coefficients (long block mode, $B = 1$), the reordering operation is the trivial task of copying the coefficients to the LIC in frequency order,

$$LIC(k) = x[0][k]. \tag{11}$$

For bitplane runlength coding using adaptive Golomb-Rice codes, reordering results in a substantial increase in coding efficiency for short-block MDCT frames. With 44.1 kHz sampled test items, the average improvement in coding efficiency is approximately 30% across the bitrate range 32 - 96 kbit/s compared to the non-interleaved case.

Note that a similar reordering can be made with non-uniform transform decompositions, for example from a wavelet packet transform, grouping together all coefficients with the same subband frequency index within the LIC prior to bitplane coding.

6.5. Sub-Sequence Formation

The coding efficiency of the bitplane runlength coding method described above can be enhanced by extracting LIC coefficients to form a sub-sequence, which for each bitplane significance scan is coded before the remaining LIC coefficients. This technique was originally investigated by Ordentlich et al [5], [25] for bitplane coding of image wavelet coefficients.

The sub-sequence is coded using a method similar to that used to code the main coefficient list (LIC), ie identify the significant list entries using fixed or adaptive Golomb runlength codes. The addition of sub-sequence coding requires the significance map to be coded in three stages, which collectively identify coefficients with MSBs in the current bitplane:

(a) extract sub-sequence coefficients from LIC;

(b) scan sub-sequence(s) for significant entries, output runlength codes and sign bits, move significant entries to LSC;

(c) scan LIC for significant entries, output runlength codes and sign bits, move significant entries to LSC.

The criteria used to extract coefficients from the LIC to form a sub-sequence should be that the extracted

coefficients have a higher expected probability of becoming significant in the pending bitplane scan than those coefficients that remain in the LIC [5]. Suitable contexts for selecting coefficients to form the sub-sequence could include:

- coefficients that are frequency-domain neighbours to significant coefficients with the same time index
- for frames containing more than one transform block ($B > 1$), coefficients that are time-domain neighbours to significant coefficients with the same frequency index
- the significant neighbour 'age', or the bitplane difference between the significant neighbour MSB bitplane and the current bitplane
- coefficients that have some harmonic relationship to significant coefficients.

Note that it is possible for coefficient extraction to take place at any point(s) within a bitplane scan, although in practice the sub-sequence is conveniently formed at the start of each bitplane scan. Multiple sub-sequence formation and coding using a number of extraction contexts is also possible.

Although sub-sequence formation and coding increases the complexity of the bitplane runlength coding algorithm, it can confer modest increases in coding efficiency compared to simple single-list scans. For example, with 44.1 kHz sampled test items, forming a single sub-sequence with a simple frequency-neighbour extraction context improves overall coding efficiency by an average of 5 % across the bitrate range 32 - 96 kb/s.

6.6. Layered Coding

The full-bandwidth coding algorithms described so far address all coefficients in a bitplane before coding lower-significance bitplanes, and coding bandwidth is invariant with bitrate. While full-bandwidth coding results in good subjective quality at higher bitrates, coding quality can decrease at lower bitrates where on average too few bits are available to code each significant coefficient. At lower bitrates improved subjective quality can be achieved by limiting the bandwidth of each bitplane scan, allocating more bits on average to each significant coefficient coded. Ideally the coding bandwidth should be constrained to a fixed value within a defined bitrate range, so that consecutive frames decoded at the same bitrate have the same bandwidth; this avoids decoding consecutive frames to different bandwidths, which can result in uncancelled transform alias products.

Defining a number of bitrate ranges where encoder bitplane scans are constrained to a limited range of coefficient frequencies results in a 'layered' bitstream where coding bandwidth increases with bitrate, and fine-grain scalability is maintained within each coded layer. Each layer is defined with a bit allocation and a bandwidth limit. Referring to Fig. 9, following coding of the base layer with the lowest bandwidth, each enhancement layer codes coefficients to a higher bandwidth limit, and can also code uncoded coefficient data from previous layer bandwidth limits.

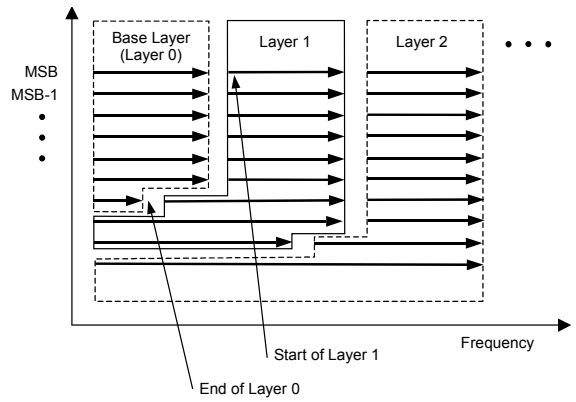

Fig. 9. Layered audio bitplane coding with variable coding bandwidth a function of bitrate.

An example of layered bitplane coding has previously been described by Park and co-workers [7], [26] where arithmetic coding is used to identify newly-significant coefficient locations within each bitplane. Runlength coding can also be used to code layered bitplanes, embedding list scans within an outer layer loop in order to achieve a layered bitstream format. The coding efficiency of a layered bitplane runlength coder can be gauged by comparing the high-bitrate performance with an equivalent full-bandwidth (non-layered) coder. Typically the overall loss in coding efficiency relative to the non-layered reference can be limited to less than 5% by addressing various design aspects of the layered system. For example, if fixed-wordlength Golomb-Rice runlength codes are used to code layered bitplanes, the runlength code wordlength can be optimised for each frame, or each layer, or each bitplane within each layer,

and output as bitstream side information as appropriate. If an adaptive Golomb-Rice runlength code is used, then the wordlength should be reset to a small value at the start of each bitplane within each layer. End-of-run's can be efficiently coded by outputting consecutive prefix '0's until the layer bandwidth limit is passed.

6.7. Results

Coding efficiency results are shown in Fig. 6 for a low-complexity bitplane runlength (BPRL) coding algorithm with the following features:

- full bandwidth coding (non-layered)
- simple LIC scans (no sub-sequence formation)
- adaptive Golomb-Rice runlength coding
- repeated prefix end-of-run codes.

The results measured with 44.1 kHz sampled test items and 1024-sample single-long-block frames suggest scalable bitplane runlength coding achieves coding efficiency slightly higher than the fixed-rate AAC reference. Direct comparisons of the average number of significant coefficients coded across the bitrate range 32 - 96 kb/s indicates the bitplane runlength algorithm achieves an equivalent bitrate advantage of 4% relative to AAC (Table 1).

7. SUBJECTIVE RESULTS

7.1. Demonstration Codec

A demonstration codec featuring a scalable quantisation stage with bitplane runlength (BPRL) coding was constructed. The BPRL encoder design follows the generic structure outlined in Fig. 1, with a time-frequency transform similar to the block-switched MDCT used in MPEG-AAC [13]. With a frame length of 1024 samples, combined BPRL encoder-decoder delay is approximately 1600 samples including lookahead, equivalent to 36 ms at typical sample rates.

A custom psychoacoustic model is used to spectrally shape quantisation errors so as to be minimally audible. Most of the computational requirement in the encoder is due to the psychoacoustic model. Because psychoacoustic model calculations are confined to the encoder, decoder complexity is low.

The BPRL quantisation stage features design details outlined in Section 6.7. Inter-frame prediction is not used, and each frame of data received by the decoder can be decoded independently without any knowledge of previous frames. Optional source coding techniques can be used to produce bitstreams that are robust against transmission errors. The bitstream syntax adopted restricts bitrate granularity to 16-bit atoms, which at typical sample rates results in bitrate granularity of 0.7 kbit/s.

A Win32 BPRL decoder and scalable bitstreams are available for demonstration purposes at http://www.scalatech.co.uk/download.htm

7.2. Subjective Comparisons

Informal but carefully controlled listening tests compared perceptual transparency bitrates achieved with the BPRL demonstration codec against two fixed-bitrate and two scalable reference codecs:

- MPEG-1 Layer 3 (MP3) - FhG mp3 codec - fixed bitrate
- MPEG-4 AAC Low Complexity Profile (AAC) [13] - Apple Quicktime Pro 6.5 - fixed bitrate
- MPEG-4 Bit Sliced Arithmetic Coding (BSAC) [7], [27] - scalable
- Microsoft Embedded Audio Coder (EAC) [11] - scalable

A measure of transparency was obtained for each codec by averaging transparency bitrates across four demanding 44.1 kHz-sampled test pieces. The tests were conducted using both single-channel and stereo material.

Results for the BPRL codec indicate average transparency bitrates of 80 kbit/s for mono signals, and 132 kbit/s for stereo material (Fig. 10). This represents an improvement in coding efficiency compared to fixed-bitrate MP3, and also the two scalable reference codecs. While the BPRL mono result is close to optimised AAC performance, the shortfall in stereo performance compared to AAC may relate to the absence of intensity stereo coding with the tested BPRL implementation

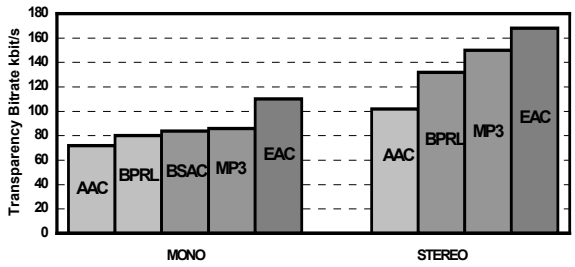

Fig. 10. Transparency bitrates for comparison codecs.

8. CONCLUSIONS

Bitplane coding has been considered as a general method of achieving fine-grain bitrate scalability suitable for audio coding. Analysing bitplane coding as a two-stage process comprising significance map and refinement stages allows coding efficiency metrics to compare alternative bitplane coding algorithms. Compact significance map coding is shown to relate to perceptual entropy coding efficiency, while the number of significant coefficients coded in each frame can be used to directly compare the coding efficiencies of fixed-rate and scalable algorithms.

Bitplane runlength (BPRL) coding, where runlength codes identify newly-significant coefficient locations in each bitplane, is shown to achieve efficient, low-complexity scalable audio coding. BPRL coding is particularly effective when using adaptive Golomb-Rice runlength codes. The coding efficiency observed with multi-block frames is substantially improved by interleaving coefficients prior to bitplane runlength coding.

BPRL coding efficiency comfortably exceeds that achieved with currently-known SPIHT hierarchies. Moreover, BPRL coding shows a small theoretical coding efficiency advantage compared to fixed-rate MPEG-AAC coding. Subjective comparisons between a demonstration BPRL codec and various reference codecs indicates practical audio bitplane runlength coding achieves fine-grain scalability while remaining competitive in terms of coding efficiency with fixed-rate codec designs.

9. ACKNOWLEDGEMENTS

This work was initially supported by the UK Department of Trade and Industry (DTI). Further support has been received from the National Endowment for Science, Technology and the Arts.

10. REFERENCES

[1] J. D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria," *IEEE J. Select Areas in Communications*, vol. 6, pp. 314 - 323 (1988 Feb.).

[2] J. Herre et al., "The Integrated Filterbank Based Scalable MPEG-4 Audio Coder," presented at the 105th Convention of the Audio Engineering Society, San Francisco, 1998 (preprint 4810).

[3] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Trans. Sig. Proc*., vol. 41, pp. 3445 - 3462 (1993 Dec.).

[4] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circuits and Sys. Video Tech*., vol. 6, pp. 243 - 250 (1996 June).

[5] E. Ordentlich et al., "A Low-Complexity Modelling Approach for Embedded Coding of Wavelet Coefficients," Proc. 1998 IEEE Data Compression Conference, Snowbird, Utah, pp. 408 - 417 (1998 Mar.).

[6] Y. Karelic and D. Malah, "Compression of High-Quality Audio Signals using Adaptive Filterbanks and a Zero-Tree Coder," Proc. 18th IEEE National Convention, Tel Aviv, Israel (1995 Mar.).

[7] S. H. Park et al., "Multi-Layer Bit-Sliced Bit Rate Scalable Audio Coding," presented at the 103rd Convention of the Audio Engineering Society, New York, Sep. 1997 (preprint 4520).

[8] Z. Lu and W. A. Pearlman, "High Quality Scalable Stereo Audio Coding" (1999), available at http://www.cipr.rpi.edu/~pearlman/papers/scal_audio.ps.gz

[9] C. Dunn, "Efficient Audio Coding with Fine-Grain Scalability, " presented at the 111th Convention of the Audio Engineering Society, New York, 30 November - 3 December 2001, *J. Audio Eng. Soc. (Abstracts)*, vol. 49, p. 1235 (2001 Dec.), preprint 5492.

[10] J. Zhou et al., "Error Resilient Scalable Audio Coding (ERSAC) for Mobile Applications," IEEE

2001 Workshop on Multimedia Signal Processing, Cannes, France (2001 Oct.).

[11] J. Li, "Embedded Audio Coding (EAC) with Implicit Psychoacoustic Masking", ACM Multimedia 2002, pp. 592 - 601, Nice, France, Dec.1 - 6, 2002.

[12] J. D. Johnston, "Estimation of Perceptual Entropy Using Noise Masking Criteria," Proc. ICASSP 1988, pp. 2524 - 2527.

[13] M. Bosi et al., "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc*., vol. 45, pp. 789 - 812 (1997 Oct.).

[14] M. A. Watson and M. Truman, "Analyzing the Performance of Lossless Coding Techniques Used in Audio Coders," presented at the 109th Convention of the Audio Engineering Society, Los Angeles, Sept. 2000 (preprint 5275).

[15] M. Raad, A. Mertins and I. Burnett, "Audio Compression using the MLT and SPIHT," Proc. 6th Int. Symposium on Digital Signal Processing for Communication Systems, Sydney, Australia, pp. 128 - 132 (2002 Jan.).

[16] M. Raad, A. Mertins and I. Burnett, "Scalable to Lossless Audio Compression Based on Perceptual Set Partitioning in Hierarchical Trees (PSPIHT)," Proc. IEEE Int. Conf. Acoustics Speech Sig. Proc (ICASSP) (2003).

[17] P. Delogne and B. Macq, "Universal Variable Length Coding for an Integrated Approach to Image Coding," *Ann. Telecommun*., vol. 46, no. 7 - 8 (1991 July).

[18] Digital Compression and Coding of Continuous-Tone Still Images, Part 1, Requirements and Guidelines: Annex G - Progressive DCT-Based Mode of Operation, ISO/IEC International Standard 10918-1 (1992 Sept.).

[19] H. S. Malvar, "Fast Progressive Wavelet Coding," Proc. 1999 IEEE Data Compression Conference, Snowbird, Utah (1999 Mar.)

[20] G. G. Langdon Jr., "An Adaptive Run-Length Coding Algorithm," *IBM Technical Disclosure Bulletin*, vol. 26, no. 7B (1983 Dec.).

[21] S. W. Golomb, "Run-length Encodings," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 399 - 401 (1966 July).

[22] B. Leslie, C. Dunn and M. Sandler, "Developments with a Zero Tree Audio Codec," Proc. AES 17th International Conf. 'High Quality Audio Coding', Florence, pp. 251 - 257 (1999 Sep.).

[23] J. Wen and J. D. Villasenor, "Reversible Variable-Length Codes for Efficient and Robust Image and Video Coding," Proc. 1998 IEEE Data Compression Conference, Snowbird, Utah, pp. 471 - 480 (1998 Mar.).

[24] S. R. Quackenbush and J. D. Johnston, "Noiseless Coding of Quantized Spectral Components in MPEG-2 Advanced Audio Coding," Proc. IEEE ASSP Workshop on Apps. Sig. Proc. to Audio and Acoustics, Mohonk (1997).

[25] E. Ordentlich et al., "Context-Based Ordering and Coding of Transform Coefficient Bitplanes for Embedded Bitstreams," US6263109 (2001 July).

[26] S. H. Park, "Scalable Audio Coding / Decoding Method and Apparatus," EP0884850A3 (2000 Mar.), see also US6122618 (2000 Sep.).

[27] ISO/IEC JTC1/SC29/WG11 N2803, "MPEG-4 Audio Version 2 (Final Committee Draft 14496-3 AMD1)", Vancouver, Canada (1999 July).