



Audio Engineering Society Convention Paper

Presented at the 117th Convention
2004 October 28–31 San Francisco, CA, USA

This convention paper has been reproduced from the author's advance manuscript, without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Additional papers may be obtained by sending request and remittance to Audio Engineering Society, 60 East 42nd Street, New York, New York 10165-2520, USA; also see www.aes.org. All rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Ultra Low Delay audio coding with constant bit rate

Ulrich Krämer, Gerald Schuller, Stefan Wabnik, Juliane Klier, and Jens Hirschfeld

Fraunhofer Institute for Digital Media Technology, Langewiesener Strasse 22, 98693 Ilmenau, Germany

Correspondence should be addressed to Gerald Schuller (shl@idmt.fraunhofer.de)

ABSTRACT

The Ultra Low Delay (ULD) codec developed at the Fraunhofer IDMT is based on a versatile perceptual audio coding method that achieves very low encoding/decoding delay and is nevertheless capable of high compression ratios. Utilizing a perceptual model for irrelevance reduction, the ULD codec is in principle a variable bit rate codec. To achieve coding with constant bit rate, the use of bit reservoir techniques would result in additional coding delay. This paper presents a rate loop which ensures constant bit rate coding without increasing coding delay. It is shown that this technique does not decrease the decoded audio quality significantly.

1. INTRODUCTION

The use of digital audio coding in professional audio productions is nowadays common practice. For some production steps, very low encoding and decoding delay has become an essential prerequisite. In live productions with wireless microphones and simultaneous monitoring or in distributed productions where artists perform simultaneously in different studios the tolerable total delay time is less than ten milliseconds. Such a threshold can hardly be reached by means of standard audio coding schemes like MPEG-1 layer 3 (MP3) [1], MPEG-2 Advanced Audio Coding (AAC) [2] and MPEG-2/4 Low Delay [3], of which the delays range from 20 milliseconds

to several hundred milliseconds [4].

The ULD Coder achieves a total encoding/decoding delay of 5.33 to 8 milliseconds with sampling frequencies from 32 kHz to 48 kHz by separating the two aims of irrelevance and redundancy reduction and assigning them to different functional units [5][6]. Fig. 1 shows the main functional blocks of the ULD Coder.

A psycho-acoustically controlled adaptive linear filter is used on the input audio signal $s(n)$ for the irrelevance reduction. In our current implementation, the perceptual model incorporates a DFT based on 256 samples with 50% overlap causing a delay of 128 samples. To synchronize the prefiltering step with

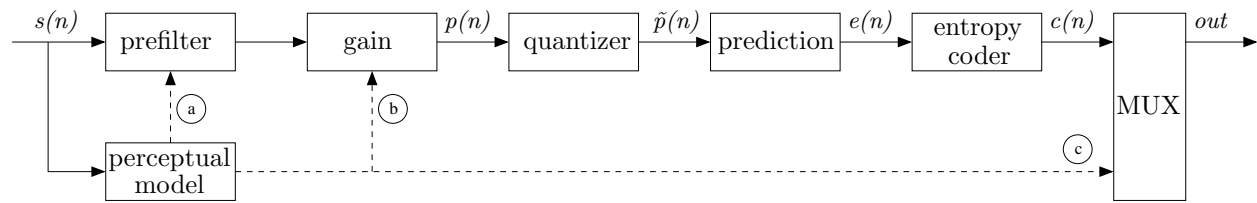


Fig. 1: Basic structure of the ULD Encoder: a) filter coefficients, b) gain factor, c) side info including a and b and additional signaling and initialization information

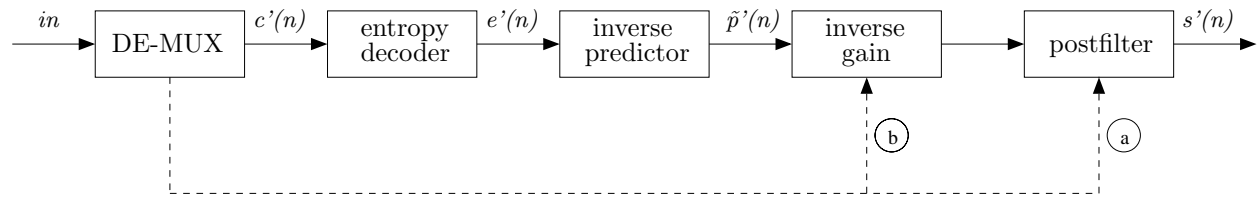


Fig. 2: Basic structure of the ULD Decoder: a) filter coefficients, b) gain factor

the perceptual model, a corresponding delay is introduced in the signal path. The output of the perceptual model is an estimation of the masking threshold. This estimation is then transformed into filter coefficients and a gain factor via the Levinson-Durbin algorithm. Together, the filter operation and the following filter gain (see Fig. 1) can be interpreted as a normalization of the input signal $s(n)$ to the masking threshold. The resulting prefiltered signal $p(n)$, which is much smaller than $s(n)$, is then uniformly quantized into $\tilde{p}(n)$ via a simple rounding operation. The decoder (see Fig. 2) contains a postfilter, which is the inverse of the prefilter, and hence has a frequency response like the masking threshold. Thus, the added quantization noise remains at or below the masking threshold, as desired.

The redundancy reduction employs a predictive lossless coding scheme [7], using backward prediction and entropy coding. The respective signals are the prediction error signal $e(n)$ and the entropy code words $c(n)$. Both processing steps provide minimal coding delay and are lossless. However, both steps inherently produce a variable bit rate, because $\tilde{p}(n)$ contains a variable amount of information.

In a final stage, side information and entropy coded audio data are packed into a bit stream; in Fig. 1

this step is visualized as a multiplexer.

2. GOAL

For fixed bit rate channels it is necessary to smooth the bit rate of the coder. A common approach for bit rate smoothing is to use a buffer [8]. The size of the buffer depends on the magnitude of the variation of the bit rate. Especially if the variations are large, this results in a large buffer size, which leads to a high encoding/decoding delay. An example are coders where there is a switch between different numbers of bands, for example 1024 and 128 bands, as in the MPEG-AAC coder. The MPEG AAC Low Delay coder avoids the block switching and hence also the high bit rate peaks. This leads to smaller buffers with a correspondingly smaller delay.

The Ultra Low Delay coder avoids switching, and thus high bit rate peaks. As a result, only a smaller buffer with a corresponding smaller delay would be necessary. But it is desirable to completely avoid the delay associated with buffering by avoiding having a buffer.

3. NEW APPROACH FOR BIT RATE SMOOTHING

3.1. Outline

Instead of a buffer, a rate loop (Fig. 3) is constructed which limits the bit demand to a user defined setting. This allows a smooth approximation of the upper bit rate limit without increasing coding delay. The reason why such a technique without delay is not widely used is, that it can lead to fast variations in decoded audio quality, which can result in perceivable audible distortions. The goal is to design a control loop in such a way, that audible distortions are minimized, without adding any buffer.

3.2. New functional units

For the novel bit rate loop, two more steps are added to the encoding process. The first step is to introduce a controllable scaling factor between the gain and the quantizer. Depending on the scale factor, the quantization noise can be varied, corresponding to a shift parallel to the masking threshold for consecutive blocks of 128 samples. A scale factor of 1.0 leaves the prefiltered signal $p(n)$ unchanged, so the quantization noise will be at the estimated masking threshold.

If the scale factor is smaller, $p(n)$ is quantized more coarsely, so quantization noise may become audible. But $\tilde{p}(n)$ is better predictable, which leads to a smaller bit demand of the following entropy coding stage. Scale factors greater than 1.0 provide additional quality headroom, because the scaling reduces the quantization noise power. This is useful, because the perceptual model is cannot be perfect. These high scale factors are used to increase the data rate in blocks that demand fewer bits than available with scale factor 1.0. In order to reverse the effect of the scaling, the scale factor has to be signaled to the decoder.

The next step to implement a rate loop is to identify the bit demand of the current block. This task is fulfilled by the bit counting module, as can be seen in Fig. 3. All information about bit consumption of the side information, signaling and audio data is collected here. For each block of 128 samples the bit rates for different scale factors are measured, and the one which best fulfills the user-defined bit rate

constraint is chosen. On the decoder side, only one pass is necessary to decode the data, because the correct scale factor is transmitted to the decoder.

Some minor modifications have to be made to the prediction and the entropy coder. Their internal memories – and in case of the predictor also the filter coefficients – have to be stored after the rate loop is finished. This ensures that all iteration steps of the next block have access to the unchanged memories of predictor and entropy coder of the preceding block. If this is not done, every iteration step would alter the memories and the filter coefficients. This would not only be suboptimal for the encoder, but the bit stream would also be impossible to decode, as the decoder only performs one iteration with the transmitted scale factor.

In addition to this, we also provide memory for the results of the iteration step that was so far closest to the target data rate, yet still below it. Scale factor, predictor memory and filter coefficients, entropy coder memory as well as the coded audio data are saved. By storing these results, we avoid having to recalculate the best iteration step after the best scale factor was determined. Section 3.3 describes a search algorithm for the optimal scale factor, where the storage of these intermediate results will be essential.

3.3. Rate loop with fixed number of iterations

For most DSP applications, the amount of iterations has to be limited, depending on the processing power of the DSP. On the other hand, more iterations would enhance audio quality, as the target data rate can be approximated more precisely. So a certain trade-off between complexity and quality has to be made, depending on the application.

The first step to limit the amount of necessary iterations is to fix the range of possible scale factors. Extensive tests with different audio material showed that the scale factors would seldom drop below 0.1, so this value was chosen as a lower limit. To achieve a bit rate as close to the target data rate as possible and to provide some quality headroom for certain blocks, some scale factors above 1.0 should be present. For our tests we decided for an upper limit of 6.0.

While testing scale factors that are quite close to each other, we made an interesting observation:

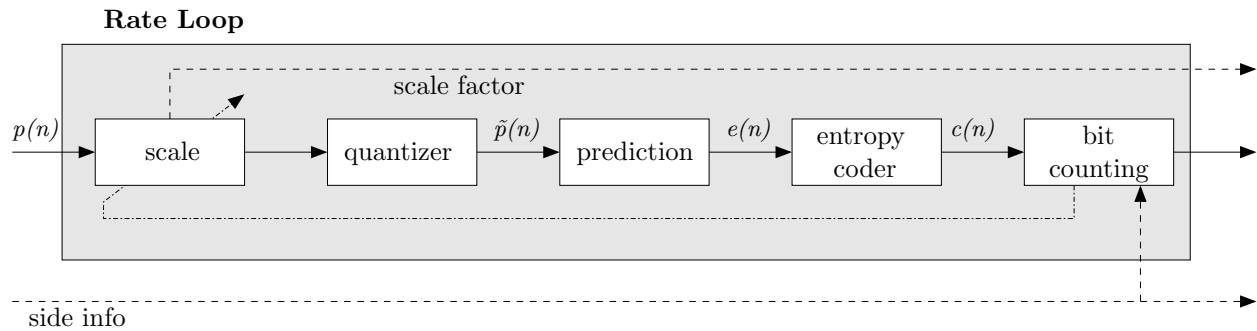


Fig. 3: Design of the rate loop

slight increases of the scale factor do not necessarily lead to higher data rates; the data rate can even drop. The same is true for small decreases of the scale factor; in some cases data rate might increase. This effect can be explained with the interaction between quantization, prediction and entropy coding. A small change of the scale factor might toggle a prefiltered audio sample from one value to the next higher or lower. As the predictor's estimate is a rounded integer value, too, it can happen that the new quantized prefiltered value is closer to the predictor's approximation. This results in a smaller prediction error and finally in a smaller amount of coded bits. Although the effect is not very strong, it must be taken into consideration in our further investigations.

The next step was to decide for a strategy to reach the best possible scale factor for a block as fast as possible. Experiments showed, that – depending on the step size – linear search with a fixed step size would either result in a high number of iterations needed to reach the scale factors at the boundaries or in reduced audio quality.

Instead of assuming a scale factor of 1.0 for the first iteration step of each block, we also tested using the scale factor of the previous block. This way, the average number of iterations can be reduced considerably, but in the worst case even more iterations would be needed to find the best scale factor.

Adjusting the step size for each iteration step according to the distance to the target data rate improved performance much, but still the results were not satisfactory for some few blocks of the signal. Especially sudden changes between silence and loud

passages and vice versa proved to be problematic, because the starting point was far away from the ultimately chosen scale factor. In addition to this, the not-monotone relation between scale factor and data rate complicates the termination of the rate loop, because the step size could be adapted into the wrong direction.

We decided to reduce the number of possible scale factors by designing a table lookup, so the maximum amount of iterations is predetermined. The goal was to create the table in such way, that audio quality does not degrade, although the set of scale factors is limited. An intuitive approach is to distribute the available scale factors so that their probability of occurrence is equal for each of them. This means that most scale factors could be found in the vicinity of 1.0, and few at the edges of the range. Hence the difference between adjacent scale factors is high near 0.1 and 6.0, and smaller around 1.0. Psycho-acoustically however, it is more desirable to enhance the resolution especially for smaller scale factors. This helps to better approximate the signal in already critical blocks. On the other hand, scale factors above 1.0 are psycho-acoustically not very important, so it is straightforward to shift scale factor resolution from the region beyond 1.0 to the region below.

Based on these findings, we optimized a set of 31 scale factors with respect to the resulting audio quality. The reason for this number will be explained later in this paper. Informal listening tests show, that the audio quality does not change significantly if only this limited set of scale factors is used. Measurements with the PEAQ tool [9] show just a

marginal degradation up to 0.15 objective difference grades (ODG).

The starting point for the linear search was chosen to be 6.0. By starting at the upper boundary, always the highest scale factor is found that fulfills the data rate constraint. Unfortunately, in the worst case we need $n = 31$ iterations to find the result, n being the amount of iterations in the worst case. If we choose the middle scale factor as the starting point, the rate loop has to iterate only $n = 16$ times, assuming that the relation between scale factor and data rate is monotone. In this case, the highest possible scale factor is not always found, but that does not affect audio quality.

To speed up the search, we decided to use binary search instead. The advantage compared to linear search is clear: a binary search over $2^n - 1$ scale factors needs exactly n iterations. That means, the given table can be searched in $n = 5$ iterations. Again, not the highest possible scale factor is found. In fewer than 1% of the blocks, a smaller scale factor is chosen. But listening test showed that this effect was not audible compared to the linear search beginning from 6.0. The differences in PEAQ measurements showed fluctuations in the range of only 0.01 ODG, which confirms the results of the listening tests.

For the transmission of the scale factor only 5 bits are needed, as the set is fixed to $2^5 - 1 = 31$ factors. The remaining table entry is used to signal a fall-back strategy, which will be described later. For all channels of a multi-channel signal, the same scale factor is used. Thus only one scale factor has to be transmitted. For multichannel files, the sum of the entropy coded bits of all channels plus additional side info has to be below the bit rate limit. As all channels use the same scale factor, another useful effect can be observed: the available audio bits are automatically distributed with respect to the bit demand of each channel. This leads to a perceptually much better result than to distribute the audio bit rate equally to all channels, as channels with low bit demand act as a kind of buffer for those with high bit demand.

Additionally, a fall-back strategy was developed, in case the smallest scale factor of 0.1 still does not yield a data rate below the target rate. In this case,

we use an m -bit quantizer on the prefiltered signal, where

$$m = \left\lfloor \frac{\text{min. number of audio bits per block}}{\text{block size}} \right\rfloor.$$

Prediction and entropy coding are skipped and the quantized signal $\tilde{p}(n)$ is passed on to the multiplexer. This strategy is signaled to the decoder by using the remaining entry in the scale factor look-up table.

In case the last iteration with the highest scale factor does not reach the data rate limit or if the target data rate is not met exactly, the remaining bits are padded with fill bits. Alternatively, this space could be used to transmit additional information.

3.4. Optimizations

The basic structure of the rate loop described in the previous section can be further refined to achieve better audio quality and performance. For example, average workload can be reduced by exiting the rate loop as soon as the target rate is exactly met or if a block contains silence. In both cases, further iterations do not lead to better results. If quantization, prediction and entropy coding are performed sample by sample, it is also possible to end the current iteration as soon as the bit limit is exceeded.

For better audio quality, the predictor's memory should be scaled every iteration step. To re-scale the memory appropriately, it has to be multiplied by the inverse scale factor of the last block and by the scale factor of the current iteration step. If that is not done, the predictor's estimates would be too big or too small, depending on the scale factor of the last block. The benefit of this optimization is a smaller error signal and thus lower bit demand of the entropy coder. Altogether, higher scale factors can be used to transmit the signal.

Further quality enhancement can be achieved if scale factor tables especially designed for the target data rate and the maximum amount of iterations are used. This way, the distribution of the scale factors can be tuned more accurately, which leads to a smoother data rate with less fill bits and higher quality.

4. RESULTS

In this section we will present the results of the subjective listening test according to the MUSHRA standard [10] and results of PEAQ measurements [9].

4.1. Test conditions

The listening tests were performed in a reference listening room¹. The MUSHRA test was implemented on a Laptop with external DA-converter and STAX amplifier/headphones. The group of the nine test listeners consists of expert and non-expert listeners. Before the subjects started with the listening test, they had the possibility to listen to a test set.

4.2. Listening test results

The tests were accomplished with 12 stereo audio files, which were coded with 96 kbps/channel . The bit rate of the variable bit rate (vbr) mode was chosen such that the average bit rate per audio file was equal to the constant bit rate (cbr) mode. In Fig. 4 the results of the PEAQ measurements are shown and in Fig. 5 the results of the MUSHRA test are presented. Both use the MPEG test suite audio files. The left bars show the values for the cbr mode and the right bars present the vbr mode. The detailed results of the MUSHRA listening test, including confidence intervals, reference file and anchors, can be found in Fig. 8.

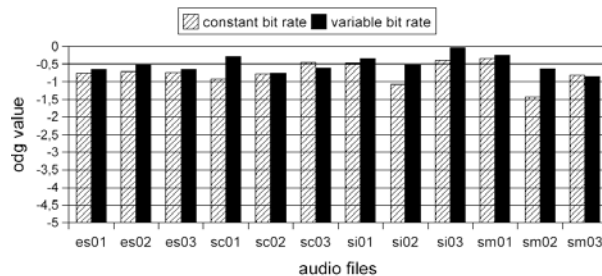


Fig. 4: PEAQ Objective Difference Grades (ODG) values of the ULD with variable and constant bit rate at 96 kbps/chan

One goal of the new approach was that the quality does not degrade too much in comparison to the vbr mode. As you can see in Fig. 4 and Fig. 5, both vbr

¹ITU-R BS. 1116 and EBU Tech. 3276

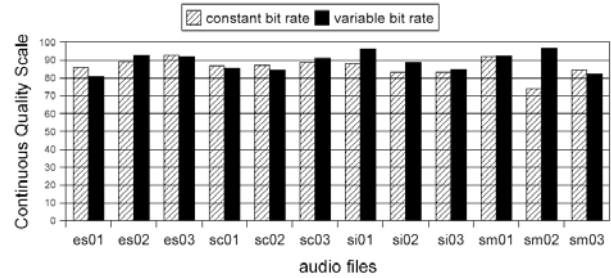


Fig. 5: MUSHRA listening test results of the ULD with variable and constant bit rate at 96 kbps/chan

and cbr have about the same quality values. The difference between the two cases is relatively small in comparison to the magnitude of the value. In some cases the cbr coded file is even slightly better than the vbr coded file. These files benefit from the additional quality headroom that is provided by the scale factors greater than 1.0.

Fig. 8 shows the results of our MUSHRA test including the 95%-confidence intervals as bars. As long as the confidence intervals overlap, there is no statistical significant difference in the grading. As can be seen, there is no significant difference for most of the items and also for the overall score (all items). There is only one item with a significant difference, sm02 (Glockenspiel), with strong tonal attacks leading to bit rate peaks.

Some test items like es01 (Suzanne Vega) show, that the ranking between cbr and vbr coded files differ for the two test methods. For the MUSHRA test the cbr coded file is better than the vbr coded file. The PEAQ measurements for this test item are reversed. This effect can be explained with the functionality of PEAQ. The used algorithm is only an approximation of the human auditory perception. Therefore, subjective listening tests are commonly more reliable than PEAQ measurements.

Fig. 6 and Fig. 7 show the measured bit rate per block (128 samples) for both vbr and cbr. *max bits per block* describes the maximum available bits per block, which can be used to encode the samples. This value is calculated from the chosen specifications, e.g. 768 bits per block for 32 kHz stereo files coded with 96 kbps/channel . *used bits per block* and *vbr bits per block* mark the actually required bits per block.

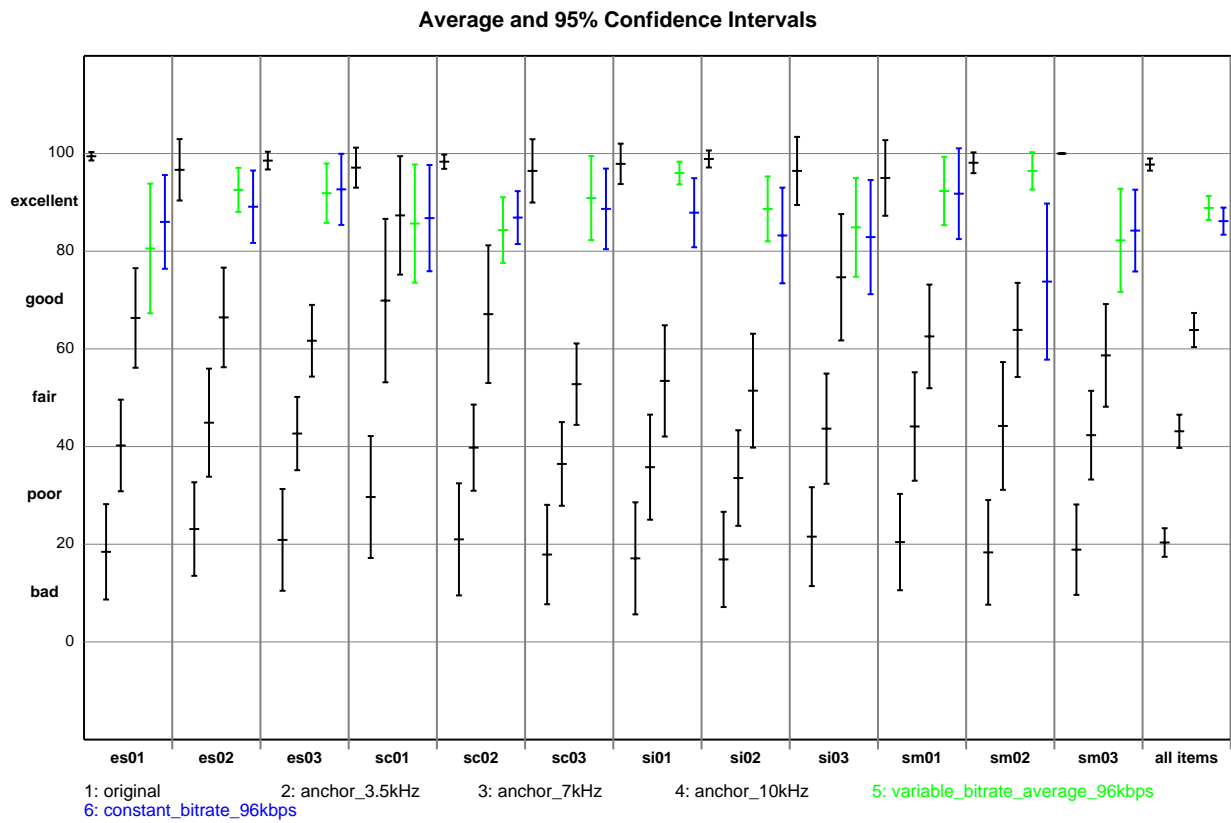


Fig. 8: Detailed results of the MUSHRA listening test

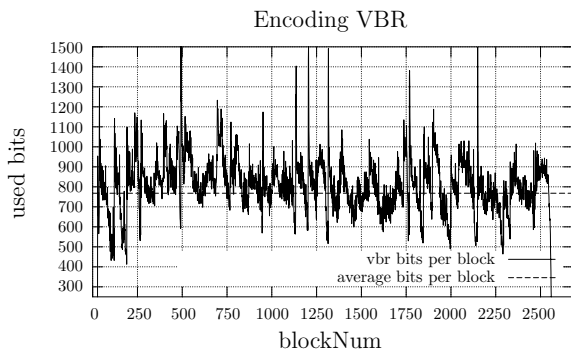


Fig. 6: es01 with variable bit rate mode: used bits per block and average bit rate per block at bit rate 96 kbps/chan

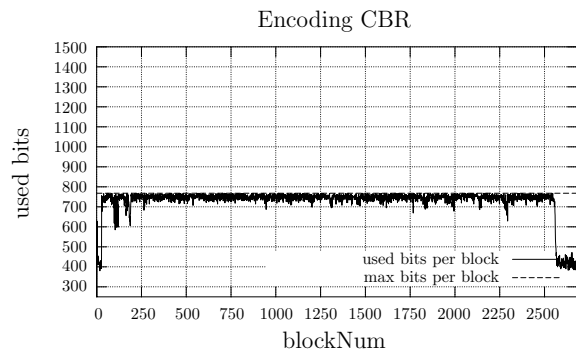


Fig. 7: es01 with constant bit rate mode: used bits per block and maximum available bits per block at bit rate 96 kbps/chan

Fig. 6 shows that the needed bits in the vbr case vary between 400 and 1500 bits. The bit usage is calculated blockwise. The result of the tested iteration loop is shown in Fig. 7, where the bit rate is flat and almost always close to the bit rate limit *max bits per block*. The increase, respectively the decrease at the beginning and end of the test items are caused by silence in the audio file.

5. CONCLUSION

The predictive coding scheme, on which the Ultra Low Delay audio coder is based, has a competitive compression ratio at a very low encoding/decoding delay. To keep this delay as low as possible also for the case of a constant bit rate channel, we introduced a new bit rate control technique, which does not need a bit rate buffer. Our subjective listening test shows that our bufferless scheme performs as good as a ULD scheme with a variable bit rate (which can be seen as with a buffer as big as one sound item) on most items. The MUSHRA test shows that there are no significant differences between constant and variable bit rate mode except for one test item (sm02, Glockenspiel), which has a difference of about 0.8 grades. At the same time our scheme eliminates the delay caused by using a bit rate buffer.

6. REFERENCES

- [1] ISO/IEC JTC1/SC29/WG11 (MPEG), International Standard ISO/IEC IS 11172-3 “Generic Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s,” 1993
- [2] ISO/IEC JTC1/SC29/WG11 (MPEG), International Standard ISO/IEC IS 13818-7 “Generic Coding of Moving Pictures and Associated Audio: Advanced Audio Coding,” 1997
- [3] Eric Allamanche, Ralf Geiger, Jürgen Herre and Thomas Sporer, “MPEG-4 Low Delay Audio Coding based on the AAC Codec,” presented at the AES 106th convention, Munich, Germany, 1999 May 8–11
- [4] Manfred Lutzky, Gerald Schuller, Marc Gayer, Ulrich Krämer and Stefan Wabnik, “A guideline to audio codec delay,” presented at the AES 116th convention, Berlin, Germany, 2004 May 8–11
- [5] Bernd Edler, Christof Faller and Gerald Schuller, “Perceptual Audio Coding Using a Time-Varying Linear Pre- and Post-Filter,” presented at the AES 109th convention, Los Angeles, CA, USA, 2000 September 22–25
- [6] Gerald Schuller, Bin Yu, Dawei Huang and Bernd Edler, “Perceptual Audio Coding using Adaptive Pre- and Post-Filter and Lossless Compression,” IEEE Transactions on Speech and Audio Processing, September 2002, pp. 379–390
- [7] Gerald Schuller and Aki Harma, “Low Delay Audio Compression using Predictive Compression,” IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Orlando, FL, USA, 2002 May 13–17
- [8] Vijay K. Madisetti and Douglas B. Williams, “The Digital Signal Processing Handbook”
- [9] ITU-R BS.1387-1, “Method for objective measurements of perceived audio quality,” November 2001
- [10] ITU-R BS.1534-1, “Method for the subjective assessment of intermediate quality levels of coding systems,” January 2003