# LOSSLESS COMPRESSION OF ONE-BIT AUDIO

*Erwin Janssen[1], Eric Knapen[2], Derk Reefman[1], Fons Bruekers[1]*

[1]Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
[2]Philips Digital Systems Laboratories, Glaslaan 2, 5616 LW Eindhoven, The Netherlands

## ABSTRACT

In this paper, the coding technique that is used by the Super Audio CD system to losslessly compress 64 times oversampled one-bit audio data is introduced. The individual steps in the encoding and decoding process are detailed. The performance of the lossless compression algorithm, as function of various genres of music, is discussed. It is shown that 74 minutes of both stereo and multi-channel music, which translates to 12.5GB of raw data, fit on the 4.7GB Super Audio disc after compression. An extension to the algorithm for future professional use of higher sampling rates is made. The lossless compression performance is demonstrated with wide-band 256 Fs recordings, and 128 and 64 Fs down converted versions of these, demonstrating the scalability of the algorithm.

## 1. INTRODUCTION

With the advent of high-capacity storage media in the early nineties, the interest for high-resolution audio home delivery has significantly increased. This trend has been recognized by the music industry, and has seeded the conception of Super Audio CD, an audio delivery format that combines the desired ultra high audio quality with the desire to reproduce both stereo and multi-channel audio recordings. While a one-bit digital storage format has been found to comply with the most demanding consumer requirements with respect to audio quality, a separate stereo and six-channel (often referred to as multi-channel, MC) area has been found necessary to deliver the optimum sound quality both in multi-channel as well as in stereo. While the physical storage capacity of the disc is $4.7 \cdot 10^9$ bytes, this still proves to be insufficient to store the necessary data for a 74-minute recording. The one-bit coding employed [1] (called "DSD", Direct Stream Digital), displays a data rate of 64 times 44.1 kS/s ("64 Fs"), roughly equaling 2.8 MS/s. For both a stereo and MC area, this translates to 12.53 Gbyte that needs to be stored. This clearly calls for a lossless compression technique, to store the data on the physical disc.

Lossless compression techniques for audio have been around for quite a while, and have all focused on the compression of PCM as this has been the digital format employed on the CD [2], [3], [4]. Because the compression is lossless, the bit-rate will vary over time, leading to playing time uncertainty. So-called "lossy" compression techniques, such as AC3 and MP3, guarantee a fixed data rate, and thus a predictable playing time. The price to pay for this feature is a time variable audio quality, which is in contradiction with the goal of Super Audio CD. Therefore Super Audio CD employs a lossless coding technique for its one-bit data, which has been coined "DST" (Direct Stream Transfer) [5].

The encoding and decoding process will be discussed in Section 2. Section 3 will show the performance of the algorithm. Finally, in Section 4, a summary is presented.

## 2. ENCODING AND DECODING

For a good understanding of the lossless encoding and decoding scheme, it is useful to distinguish the three stages [6], [7] of framing, prediction, and entropy coding, which will be discussed separately. To complete the discussion on the encoding process, the final stage of multiplexing is briefly investigated.

### 2.1. Framing

The framing process divides the original one-bit audio stream consisting of samples $b \in \{0, 1\}$ into frames of length 37,632 bits, corresponding to 1/75 of a second, assuming a sampling rate of 2.8MS/s. The purpose of framing is two-fold. Firstly, framing is necessary to provide easy, "random" access to the audio data during playback. For the same reason, each frame needs to be independently encoded, which enables the player to decode separate frames without knowledge about preceding frames. Secondly, framing allows the audio contents in a frame to be regarded as stationary (or at least, quasi-stationary). This is the underlying assumption in the prediction process. The framing rate is chosen such that the assumption of quasi-stationary audio is reasonable, while it still does not result in excessive overhead.

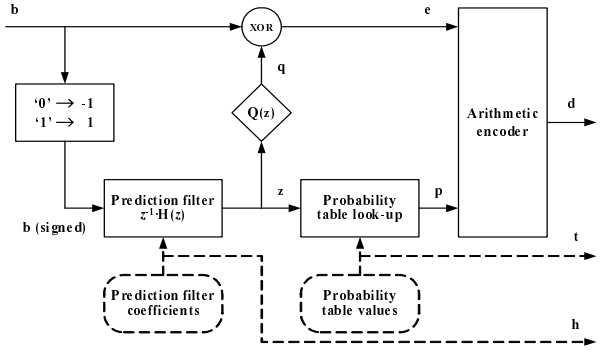


**Figure 1 Schematic overview of the encoder.**

### 2.2. Prediction

Prediction filtering is the first necessary step in the process of (audio) data compression. The prediction filtering step, shown in more detail in Figure 1, attempts to remove redundancy from the

audio bit stream b, by creating a new bit stream e, which is not redundant. Together with the prediction filter coefficients h, error stream e carries the same information as b. The prediction filter is denoted as $z^{-1}H(z)$, to emphasize the fact that the filter transfer contains a delay, which is mandatory to create an encoder that can be time-reversed (thus creating the decoder). The FIR prediction filter can be designed according to standard methods, the most well-known based on Minimum Mean Squared Error (MMSE, [3]).

Application of the MMSE criterion leads to the prediction error equality that needs to be minimized :

$$\sum_{n=1}^{M} \varepsilon^2[n] = \sum_{n=1}^{M} \left( \sum_{i=1}^{L} h[i] \cdot b[n-i] - b[n] \right)^2 \qquad (1)$$

where M is the number of bits per frame, and L the number of prediction coefficients. After straightforward manipulation [8], [3] this results in the coefficients h. In general, the FIR filter found in this way will be a minimum phase filter. To obtain the optimum balance between prediction accuracy and the number of bits taken by the prediction filter description, the prediction filter coefficients are quantized to 9-bit fixed-point numbers. Referring back to Figure 1, it is clear that the prediction signal z is multi-bit. The prediction bits q are derived from the multi-bit values z by simple truncation, indicated by the block labeled Q(z). It should be noted that the error between bit-stream b and multi-bit signal z is minimized, whereas ideally the difference between b and the q, the one-bit quantized version of z, would be minimized (see Figure 1). This however results in intractable mathematics.

Finally, the error signal e is calculated by an exclusive-or (XOR) operation between b and q. The purpose of the prediction filter is to create as many zeroes in e as possible, as this will enable significant data reductions by entropy encoding (see Section 2.3).

As becomes clear from the decoder diagram (Figure 2), the computationally demanding design of the prediction filter is only required in the encoder. The player only has to perform the, much less demanding, decoding process, where the most expensive operation is the FIR filtering process. Since the filtering needs to be performed on a one-bit signal, the implementation is straightforward and does not pose any problems.

To enable complete reconstruction of the original bit stream on the decoder side, the prediction filter coefficients and the error bits have to be transferred for each frame. The decoder calculates the original bit stream from the error bits and the predictions (which are calculated exactly in the same way as in the encoder):

$$b[n] = \begin{cases} 0, \; if \; e[n] = q[n] \\ 1, \; if \; e[n] \neq q[n] \end{cases} \qquad (2)$$

So far, no attention has been paid to the maximum number of prediction filter coefficients. While, at least theoretically, the prediction results turn out to be better for increased filter length, the overhead of the extra coefficients that need to be stored on the disc increases. This is indicated in Figure 3, where the compression ratio is plotted as a function of the prediction filter length, with and without taking the overhead of the prediction filter coefficients into account. We see that around a filter length of 130, the optimum compression ratio is achieved. While these graphs will vary over, *e.g.*, the different genres of music, it turns out that a maximum filter length of 128 typically results in close-to-optimal performance. Although for a classic prediction filter, a length of 128 seems long, it should be realized, however, that the input signal is a one-bit signal, and that the filtering operation therefore is not demanding at all.

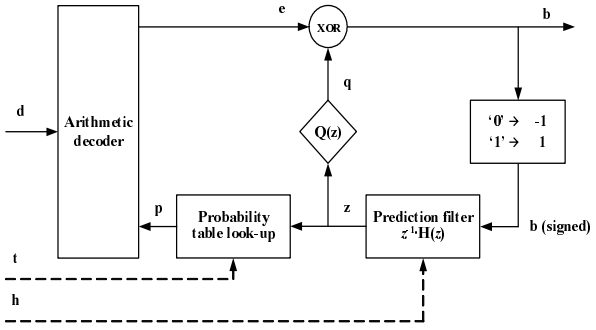


**Figure 2 Decoder overview. The decoder needs to receive the prediction filter coefficients h, encrypted data-stream d, and probability table information.**

It is also interesting to note that the compression ratio seems to increase in a step-wise manner with the increase of the prediction filter length. This feature occurs for one-bit audio with virtually all genres of audio, and some times results in the fact that a plateau in compression ratio is achieved for a filter length less than 128, after which no further improvement in compression ratio is observed. In these cases, the prediction filter length is chosen to be less than 128. In most practical applications, the optimal filter length has been found to vary between 40 and 128.

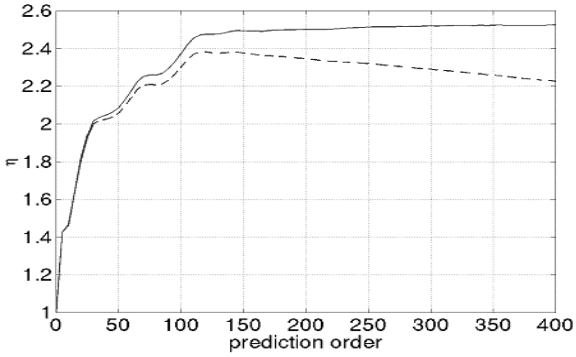


**Figure 3 Compression ratio, with (dashes) and without (solid line) overhead taken into account, as a function of the filter length. For higher orders, the filter coefficient overhead generally grows faster than the gain in coding efficiency.**

## 2.3. Entropy coding

When proper prediction filters are used, the signal e will consist of more zeroes than ones and can thus result in a possible compression gain. Suppose that the probability of a '1' in e is denoted by p, then the probability of a '0' equals (1-p). The

minimum number of bits $N_{bits}$ with which, on average, a single bit of the stream e can be represented then equals:

$$N_{bits} = -( p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p) ) \quad (3)$$

Suppose 90% of all predictions are correct, then p = 0.1 and $N_{bits}$ = 0.47. A compression of about a factor 2 is possible. While this calculation based on entropy calculations presents an upper limit to the achievable compression, an algorithm that under practical circumstances approaches this limit is the arithmetic coding algorithm [9], [10].

Arithmetic encoding methods can only be used successfully when accurate information on the probabilities of the symbols "0" or "1" is available. In Figure 4, a histogram (measured over 188,160 samples taken from a 6-channel pop-recording) of the occurrence of the different values of |z| is given, indicated by diamonds. Also shown in squares, is the histogram for the occurrence of |z|, given the fact that the prediction of the sign of z is correct, *i.e.*, e=0. In triangles, the histogram is shown for the occurrence of |z|, given the fact that the prediction is wrong (e=1). These plots show that there is a very strong relationship between the value of z and the reliability of the prediction, which can be exploited in the arithmetic encoding.

The symbol probabilities needed for arithmetic coding are calculated by making a histogram (or table) as shown in Figure 4. Denoting the probability that a prediction is correct by P(e=0), we see that since P(e=0) = 1 - P(e=1), it is not necessary to calculate two tables: only the error probability table t, for P(e=1), is used for arithmetic encoding and transferred to the decoder.

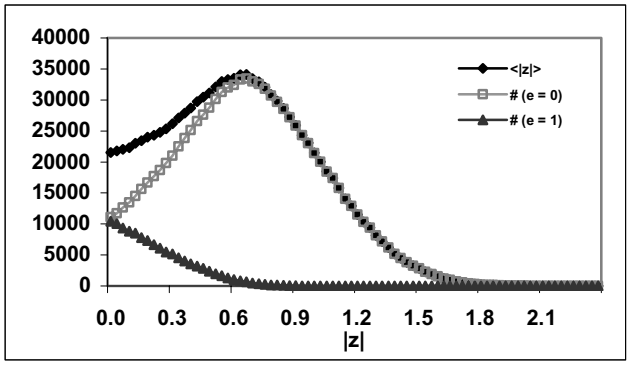


**Figure 4 Distribution of |z| (diamonds) and the corresponding histograms for the number of right predictions (e=0, squares) and wrong predictions (e=1, triangles).**

### 2.4. Channel multiplexing

In the previous sections the "source model" [9], consisting of the prediction filter and probability table, has been discussed for one channel. In a full encoder, every channel has its own source model, whereas only a single arithmetic encoder is used. To exploit the correlation between channels, however, it is also possible to let channels share prediction filters and/or probability tables. Sharing filters or probability tables is profitable when the decrease in number of metadata bits, necessary to transfer the filter or table information from the encoder to the decoder, is higher than the increase in number of arithmetic code bits. The latter number will typically be somewhat larger, since it is not always possible to construct a prediction filter (or probability table) that leads to optimal arithmetic encoding for all channels that are using it.

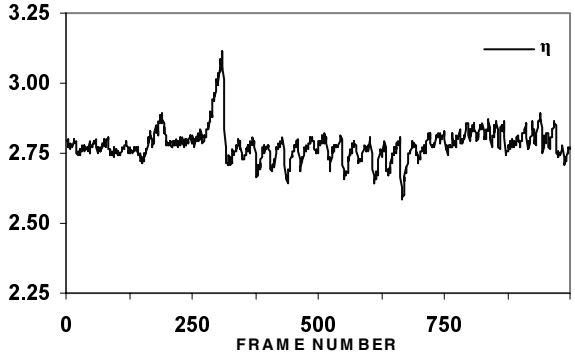


**Figure 5 Compression ratio for 1000 frames of recording A (classical music, 6 channels).**

## 3. RESULTS

To illustrate the performance of the algorithm, a few typical music excerpts have been selected from commercially available recordings, labeled A, B, C, and D. A and B represent a typical classical piece of music, excerpt C is a piece of jazz, and D represents a typical pop music recording. In Figure 5, for excerpt A, the compression ratio η is depicted over a length of 1000 frames. It can clearly be seen that the compression ratio varies significantly over time. Musical events, such as a sudden intense percussion, have a significant impact on the compression ratio. Typically, during periods of silence, the compression ratio increases, whereas during extremely loud passages the compression ratio drops.

| Title | Description | #ch | η | Achievable playing time |
|---|---|---|---|---|
| A | Classical music | 6<br>2 | 2.795<br>2.777 | 1h 17min 26s |
| B | Classical music | 6<br>2 | 2.826<br>2.820 | 1h 18min 23s |
| C | Jazz music | 6<br>2 | 2.728<br>2.696 | 1h 15min 29s |
| D | Pop music | 6<br>2 | 2.688<br>2.636 | 1h 14min 14s |

**Table 1 Average compression ratio for recordings A – D.**

Table 1 shows the average compression ratios as obtained on the full music titles, from which the excerpts have been taken. All the tabulated discs comprise a stereo as well as a 6-channel

program. The maximum possible playing time for a disc containing the stereo and multi-channel programs (equal length) has been calculated and tabulated in Table 1. It has been assumed that the average compression ratio stays constant and that the data has to fit on one layer of a Super Audio CD with a capacity of $4.7 \cdot 10^9$ bytes.

An interesting observation is that, typically, the MC part compresses slightly better than the stereo part. In most circumstances this is due to the fact that especially the rear channels of the MC recording are quieter than the stereo channels. As remarked earlier, the compression ratio increases slightly for quieter recordings, explaining the higher compression ratio for the MC part.

The lossless compression algorithm, although initially developed for a 64 Fs sampling rate, can be easily extended to higher sampling rates for professional use, *e.g.,* 128 Fs and 256 Fs. In order to achieve the best compression gain for these higher rates, the frame length duration is kept constant at 1/75 of a second, resulting in frame sizes of double or quadruple the size used at 64 Fs. Other parameters, *e.g.* the maximum prediction order or probability table length, stay unchanged.

Table 2 lists for three example wideband recordings the coding gain. The recordings have been performed at 256 Fs, from which downsampled versions at 64 and 128 Fs have been constructed. Fragment E is a typical classical recording in stereo. Fragment F consists out of 6 channel pop music. Fragment G contains a solo intermezzo of a violoncello, recorded using 5 microphones. From the table it becomes immediately clear that the type of music has strong influence on compression gain. Higher sampling rates result in higher compression gains. However, a clear relationship between the increase in compression gain for increasing sampling frequency is absent. Recording at a higher sampling rate increases the amount of information, which, as expected, in spite of the increase in compression gain, increases the amount of required storage.

| Title | Description | #ch | η | | |
|---|---|---|---|---|---|
| | | | 64 Fs | 128 Fs | 256 Fs |
| E | Classical music | 2 | 2.66 | 3.03 | 4.24 |
| F | Pop music | 6 | 2.76 | 3.05 | 3.59 |
| G | Classical music (solo) | 5 | 2.98 | 3.20 | 3.51 |

**Table 2 Compression gain for three 64, 128, and 256 Fs recordings.**

## 4. SUMMARY

We have demonstrated the feasibility of lossless compression for one-bit coded signals. The compression algorithm is based on a combination of linear prediction and arithmetic encoding. Although the encoding algorithm is computationally rather demanding, the decoding algorithm is much less so, and can easily be implemented using standard ASIC technology. As a result of the lossless nature of the compression, the compression ratio varies, and is dependent on the program material. Quieter material will result in a slightly improved compression, and hence a longer maximum playing time. On average, it is possible to achieve a playing time of 74 minutes for a disc containing both a stereo and multi-channel recording.

An extension to the algorithm, to efficiently deal with higher sampling rates, has been made. It has been shown, that with a minimal change to the algorithm, also higher sampling rates, *e.g.*, 128 Fs and 256 Fs, show excellent compression ratios. This extension enables the use of higher sampling rates for future professional use. As a result, one-bit coding technology combines audiophile audio quality with a realistic data-rate requirement, offering the best of high-resolution audio, for both consumer and professional use.

## 5. ACKNOWLEDGEMENTS

The authors wish to thank all their Philips colleagues who did pioneering work in the field of one-bit lossless compression.

## 6. REFERENCES

[1] Derk Reefman and Erwin Janssen, “One-bit audio: an overview”, JAES, vol. 52, no. 2, February 2004.

[2] Claude Cellier and Pierre Chênes, “Lossless audio data compression for real-time applications”, AES preprint 3780, presented at the 95th AES Convention, New York, U.S.A. (1993 October 7-10).

[3] N.S. Jayant and P.I. Noll, “Digital Coding of Waveforms: Principles and Applications to Speech and Video”, Englewood Cliffs, NJ: Prentice Hall 1984.

[4] R. F. Rice, “Some practical universal noiseless coding techniques”, Tech. Rep. 79-22, Jet Propulsion Laboratory, Pasadena, CA, Mar. 1979.

[5] Eric Knapen, Derk Reefman, Erwin Janssen, and Fons Bruekers, “Lossless compression of one-bit audio”, JAES, vol. 52, no. 2, February 2004.

[6] Fons Bruekers, Werner Oomen, René van der Vleuten and Leon van de Kerkhof, “Lossless coding of 1-bit audio signals” presented at the AES 8th Japanese Regional Convention, Tokyo, Japan (1997 June 25-27).

[7] Fons Bruekers, Werner Oomen, René van der Vleuten and Leon van de Kerkhof, “Improved lossless coding of 1-bit audio signals”, presented at the 103rd AES Convention, New York, U.S.A. (1997 September 26-29).

[8] William H. Press et al., “Numerical recipes in C: the art of scientific computing”, Second edition, Cambridge University Press, Cambridge, England, 1992.

[9] I.H. Witten, R.M. Neal and J.G. Cleary, “Arithmetic coding for data compression”, Communications ACM, vol. 30, pp. 520-540, June 1987.

[10] P.G. Howard and J.S. Vitter, “ Arithmetic coding for data compression”, Proc. IEEE, vol. 82, no. 6, pp. 857-865, June 1994.