DOI: 10.15439/2018F181
ISSN 2300-5963 ACSIS, Vol. 17

Analysis of inter-channel dependencies in audio lossless block coding

Cezary Wernik
West Pomeranian University of Technology in Szczecin,
al. Piastów 17, 70-310 Szczecin, Poland
Email: cwernik@wi.zut.edu.pl

Grzegorz Ulacha
West Pomeranian University of Technology in Szczecin,
al. Piastów 17, 70-310 Szczecin, Poland
Email: gulacha@wi.zut.edu.pl

Abstract— **In this paper the basics of data predictive modeling (using the method of minimization mean square error) for lossless audio compression are presented. The described research focuses on inter-channel analysis and setting range of prediction in dependencies of frame size. In addition, the concept of data flow using inter-channel dependencies and an authorial, effective and flexible method of saving prediction coefficients are presented.**

I. Introduction to lossless audio data compression

MINIMIZATION of storage and transmission data cost are one of most important issues of teleinformatics. Tool for simplification of reduction of this cost is data compression. A lot of such compression algorithms exist and the most effective ones are adapted to the specific data type. Compression methods can be divided into lossy and lossless, and this research focuses on the latter, limited to the coding of audio data.

Important purposes of lossless audio compression include recording storage, saving of records with high-quality sound on commercial media (e.g. DVDs, Blu-Ray), and selling songs in online music stores for more demanding customers who are not satisfied with the quality of mp3 format [5]. Moreover, lossless mode is often required at the stage of music processing in a studio, advertising materials and in the production of radio and television programs, films (post-production [1]) etc. In such case, no lossy coding is used, which at each iteration of sound editing may cumulate additional distortions.

In the beginning of the 21st century, many effective proposals for the MPEG-4 Lossless Audio Coding standard were developed [4], but we cannot ignore the fact that a branch of amateur solutions develops in an independent way, which use algorithms that are not fully presented in scientific publications. For example OptimFrog [14] and Monkey's Audio[15] belong to the top of most efficient programs for lossless audio compression.

In modern compression methods usually two steps are used: data decomposition, and then compression by one of the efficient entropy methods. The most effective ones are arithmetic coding, Huffman coding [10] and its variations such as Golomb [3] and Rice [8] code. In the case of audio coding, a Gilbert-Moore block code, which is a variation of the arithmetic code [7] is also used.

Publications in the field of lossless compression of audio are mainly focused on the stage of data decomposition. There are two basic types of modeling. The first is the use of linear prediction [2], [9] or non-linear (e.g. using neural networks [6]). The second type is the use of such transformations as DCT (MPEG-4 SLS [12]) or wavelet, but the current research shows that this type is slightly less efficient in the case of lossless coding. Therefore, predictive methods are used in most cases.

In Sections II and III the basics of audio modeling, including calculation (based on the mean square error minimization method) and a typical method of saving prediction coefficients are presented. In Section IV the authorial solution of flexible and effective method saving prediction coefficients are presented, afterwards In Section V presented the analysis of inter-channel dependencies. In Section VI was described the effective way of coding prediction errors (Golomb adaptive code), while in Section VII presented the summary and comparison of the efficiency of the proposed method against other known solutions.

II. Basics of audio modeling

In lossless audio codec's, typical linear predictor of the order r is used for modeling, which is the predicted value of the currently coded sample $x(n)$ based on weighted average of the r previous signal samples. The simplest predictive models are those with fixed coefficients, including, the DPCM constant model using the previous sample $\hat{x}(n) = x(n-1)$. The use of a linear predictor allows to encode only prediction errors, i.e. differences $e(n)$ between the actual and predicted value (rounded to the nearest integer), which are most often small values oscillating near zero:

$$e(n) = x(n) - \hat{x}(n) \qquad (1)$$

In this way we obtain a difference signal in which the distribution of errors $e(n)$ has a character similar to Laplace distribution, which allows for efficient coding using one of static or adaptive entropy methods, such as Golomb-Rice code or arithmetic code (see Fig. 1 in chapter V).

Usually a method called forward adaptation is used, as the encoder must have access to the whole frame before encoding, and this means that the calculated coefficients should also be sent to the decoder. For this reason, when developing the method with forward adaptation, one should calculate the effective way of choosing the frame size, the order of prediction, as well as the accuracy of saving the prediction coefficients. This is an asymmetrical method temporarily, since the decoder works relatively quickly, downloading only the head information associated with the given frame and decoding based on the formula (1). The disproportion of time in the method of forward adaptation (characterized by a longer coding time in relation to decoding) plays a significant role, since the coding operation is most often carried out once, and decoding many times.

A typical forward adaptation solution is used in the MPEG-4 ALS, where the frame is approximately 43 ms long (depending on the sampling frequency, the frame length counted in the samples is different and the maximum at $f_p = 192$ kHz is $N = 8192$).

We can also introduce the term *long term* of the frame (which is a group of frames) called *super-frame* here, in which the number N can be hundreds of thousands of samples. Although MPEG-4 ALS uses frames with a maximum size of $N = 2^{13}$, there is no obstacle in increasing length of frames in newer solutions. Their length may be limited by the principle of free access to data in real time, which results from the needs of e.g. studio sound processing. For this purpose, MPEG-4 ALS proposes independent access to the frame set every 500 ms (no need to decode previous data to correctly decode any frames), which introduces a limit of $N_{\max}$ = 24 000 samples in one super-frame when sampling frequency is 48 kHz. Above this value one should give up the possibility of free access. However, this is not a problem for archiving and transmission applications, e.g. when someone wants to purchase the whole artist's album from the online store.

In this paper, we consider th application of super-frames with a length of 20 seconds. The analysis was made on 16 dozen-second fragments of recordings (stereo, 16-bit samples, 44 100 samples per second) of various genres of music, men's and women's speech recordings available in base [16].

III. Calculation of prediction coefficients using MMSE method

It has been widely accepted that for audio signals, the calculation of prediction coefficients using the Mean Square Error Minimization (MMSE) method gives very good results.

To calculate the prediction coefficients in practice, the Autocorrelation Levinson-Durbin method is most often used [13], which by simplification does not require the calculation of the inverse matrix, but it is able to calculate the model coefficients in an iterative manner for subsequent orders of prediction. In this way, we reduce computational complexity from $0(n^3)$ to $0(n^2)$. An additional advantage is that the reflection coefficients $\mathbf{k} = [k_1, k_2, ..., k_r]^T$ often referred to as PARCOR (partial correlation coefficients) belong to the compartment (-1; 1) and they can be effectively coded (they are subjected to a quantization process, e.g. using 7 bits). Using this, the size of the header containing the set of coefficients of a given model is not large, even if quite high orders of prediction are used [5].

In contrast to the PARCOR format, which assumes the stationarity of the audio signal the new approach proposed allows for coding efficiently prediction coefficients obtained in any way (e.g. thanks to the use of different types of suboptimal algorithms that minimize the entropy value or a bitwise average in each coded frame) [10].

Rejecting the assumption about the stationarity of audio signal should be used the autocovariance method, wherein the vector of prediction coefficients $\mathbf{w} = [w_1, w_2, ..., w_r]^T$ is calculated from the matrix equation [10]:

$$\mathbf{w} = \mathbf{R}^{-1} \cdot \mathbf{p} \qquad (2)$$

where $\mathbf{R}$ is a square matrix with dimensions $r \times r$ elements $R_{(j,i)}$ such that:

$$R_{(j,i)} = \sum_{n=0}^{N-1} x(n-i) \cdot x(n-j), \qquad (3)$$

while $\mathbf{p}$ is a vector with size $r \times 1$ elements $p_{(j)}$ such that:

$$p_{(j)} = \sum_{n=0}^{N-1} x(n) \cdot x(n-j), \qquad (4)$$

where N is the number of samples in frame. It is assumed that for the first r samples in the first frame of super-frame a simplified prediction model of the lower order is used.

IV. Encoding prediction coefficients method

The method of saving header data proposed in this work assumes that from the set of prediction coefficients $\mathbf{w}$ we choose the one with the highest absolute value. We mark his position in the vector as $i_{\max}$, and the value of this coefficient as $w_{\max}$. Value of $w_{\max}$ is initially projected onto a 32-bit float type, and the index $i_{\max}$ is saved as an 8-bit integer (at $r \le 256$).

All other coefficients are coded on $b + 1$ bits after normalizing their value in relation to $w_{\max}$ and appropriate scaling, in a manner consistent with the formula:

$$\overline{w}_i = \left\lfloor \left(\frac{w_i}{w_{\max}} + 1 \right) \cdot \left(2^b - \frac{1}{2} \right) + \frac{1}{2} \right\rfloor, \qquad (5)$$

Reconstruction of the original prediction coefficients from the encoded header follows the use of the formula:

$$\overset{=}{w_i} = \left(\frac{w_i}{2^b - \frac{1}{2}} - 1 \right) \cdot w_{\max} . \tag{6}$$

The length of the header in bits for each frame is calculated by formula $\text{float}_{\text{size}} + \lceil \log_r r \rceil + (r-1)\cdot(b+1)$, where $\text{float}_{\text{size}}$ is the size of a variable compliant with the standard float32, intended for saving the $w_{\max}$ coefficient, which accuracy was experimentally reduced from 32 to 21 bits.

Theoretically, the higher the order of prediction we use, the more effective the prediction model we will get. Unfortunately higher r is the reason for the increase in the size of the frame header. At the same time it can be noticed that the increase in the order of predictions also increases the required accuracy of the vector **w** coefficients. Both of these parameters indicate that it may be more profitable to use smaller order thus reducing the size of the frame. If frames are shorter, the more accurate matching of predictive models to the variability in time of the audio characteristics is. On the other hand, too short frames cause significant increase of the header, which in turn leads to the need to further reduct the order of prediction and the value of b. Based on conducted experiments, specific b values are set for ranges of prediction orders r, at given frame lengths 2^q as presented in Table I.

TABLE I.
OPTIMAL b-VALUES RELATIVE TO THE ORDER OF PREDICTION r WITH 2^q FRAME LENGHT

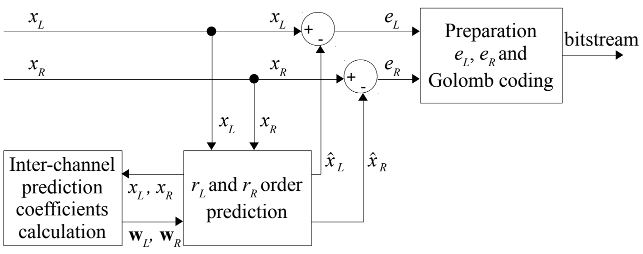
Q	b	
9	11, when $r < 24$	10, when $r \geq 24$
10	11	
11	12, when $r < 12$	11, when $r \geq 12$
12	12, when $r < 192$	11, when $r \geq 192$
13 and 14	12	

V. EXPLOITING INTER-CHANNEL DEPENDENCIES

Existing dependencies between channels allows to use in r-predictive models samples from a both channels, left $x_L(n-i)$ and right $x_R(n-j)$.

$$\begin{aligned} \hat{x}_L(n) &= \sum_{i=1}^{r_L} a_i^{(L)} \cdot x_L(n-i) + \sum_{j=1}^{r_R} b_j^{(L)} \cdot x_R(n-j), \\ \hat{x}_R(n) &= \sum_{j=1}^{r_L} b_j^{(R)} \cdot x_R(n-j) + \sum_{i=0}^{r_R-1} a_i^{(R)} \cdot x_L(n-i) \end{aligned} \tag{7}$$

There are vectors of prediction coefficients for the left channel $\mathbf{w}_L = \left[a_1^{(L)}, a_2^{(L)}, \ldots, a_{r_L}^{(L)}, b_1^{(L)}, b_2^{(L)}, \ldots, b_{r_R}^{(L)}\right]^T$ and $\mathbf{w}_R = \left[b_1^{(R)}, b_2^{(R)}, \ldots, b_{r_L}^{(R)}, a_0^{(R)}, a_1^{(R)}, \ldots, a_{r_R-1}^{(R)}\right]^T$ the right channel. Fig. 1. shows the data flow and existing dependencies between them. The formulas are two because by coding (decoding) the value of the right channel sample $x_R(n)$, there is already access to the current sample of the left channel $x_L(n)$.



x_L
x_R
x_L
x_R
e_L
e_R
+
-
Preparation e_L, e_R and Golomb coding
bitstream
x_L
x_R
$\hat{x}_L$
$\hat{x}_R$
Inter-channel prediction coefficients calculation
x_L, x_R
r_L and r_R order prediction
$\mathbf{w}_L$, $\mathbf{w}_R$

Fig. 1 Schema of proposition our algorithm

The result of the bit average can be influenced by the selection of which channels are coded in the first order. It should be clearly noted that the r_L ordering in both cases concerns the set of samples of the currently coded channel, while r_R is number samples of the opposite channel, in addition $r = r_L + r_R$.

There is a problem with the selection of the universal r_R/r_L ratio. The most common proportions in the literature are 1:1 and 1:2, but after completing the bit-minimizing experiments it turned out that depending on the size of frame (2^q), the best order of prediction r is changing (within the test database). Also the increase order of the prediction r leads to a decreasing value of the ratio r_R/r_L. The results of experiments are presented in Table II.

TABLE II.
EXAMINATION OF THE OPTIMAL RATIO r_R/r_L IN RELATION TO THE LENGTH OF THE 2^q FRAME

q_r	R	r_L	r_R	$\frac{r_R}{r_L}$	L_{avg}
9	12	6	6	1,000	9,369
10	15	8	7	0,875	9,249
11	19	11	8	0,727	**9,202**
12	34	22	12	0,545	9,246
13	58	42	16	0,381	9,335
14	57	44	13	0,295	9,438

On Fig. 2. was show the average level of Pearson's correlation coefficient (for the whole test base), using the best settings $\{q, r_L, r_R\} = \{11, 11, 8\}$, between the coded sample $x_L(n)$ and 11 adjacent samples $x_L(n-i)$ left channel and 8 samples (graph bars (lag) with indexes 12 to 19) of the right channel $x_R(n-j)$. The shape of the chart for individual files does not differ significantly from the one shown in Fig. 2. This can not be said about the charts of the average level of absolute value of prediction coefficients, which differ significantly for individual files. The ATrain and TomsDiner files were selected from the test database, for which the average level of absolute values of prediction coefficients are presented in Fig. 3 and Fig. 4, respectively. For both of these files, a bit average analysis was made depending on the proportion of r_L to r_R with a constant value of r = 19 (see Fig. 5). The best bit averages was obtained with $\{r_L, r_R\} = \{18, 1\}$ for file ATrain and with $\{r_L, r_R\} = \{7, 12\}$ for file TomsDiner. However, these optimal settings can not be deduced only from analysis Fig. 2-4, and the procedure for scanning all settings r = 19 (resulting in the data from Fig. 5) leads to a significant complexity of the implementation of the encoder. However, these optimal settings can not be deduced solely from based on the analysis of Fig. 2-4. For this reason, use the scanning procedure all settings (r=19) lead to a significant complexity of the implementing coder. This shows that the proportions r_L to r_R differ significantly from the common for the whole base of the best compromise pair $\{r_L, r_R\} = \{11, 8\}$.

VI. Golomb Code Aplication for Prediction Error Coding

Golomb code [3] is a specific version of the Huffman code [10] which is a prefix code for a source with an infinite symbol alphabet. It is used to represent integers consistent with the geometric distribution. It works well for encoding audio after their predictive modeling.

The main advantage of the Golomb code is that it does not require the use of code tables and is relatively simple in hardware implementation. Golomb's code with forward adaptation was used in this work, calculating the individual parameter m of this code for each frame with a length of 512 samples. Its saved on 13 bits is sufficient to describe the local probability distribution.

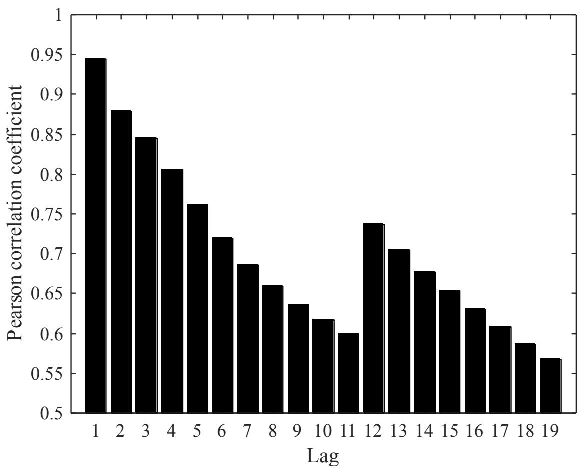


Fig. 2 Inter-channels mean of absolute value Pearson correlation coefficients

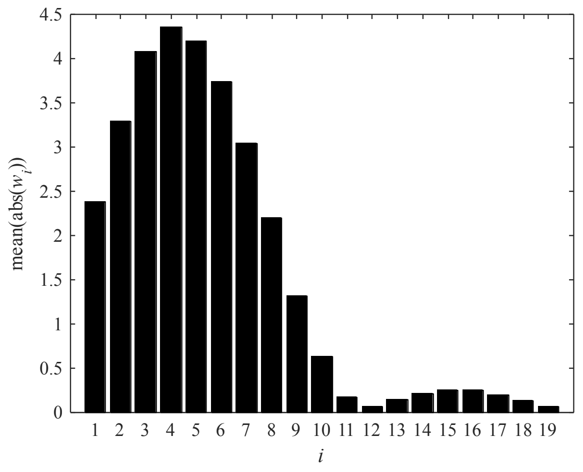


Fig. 3 Mean value of absolute prediction coefficients for ATrain file

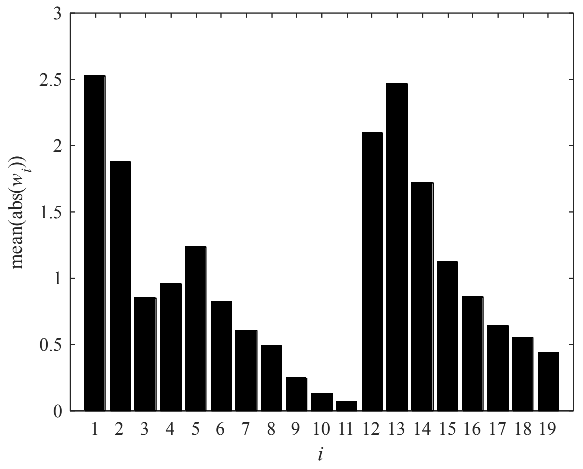


Fig. 4 Mean value of absolute prediction coefficients for TomsDiner file

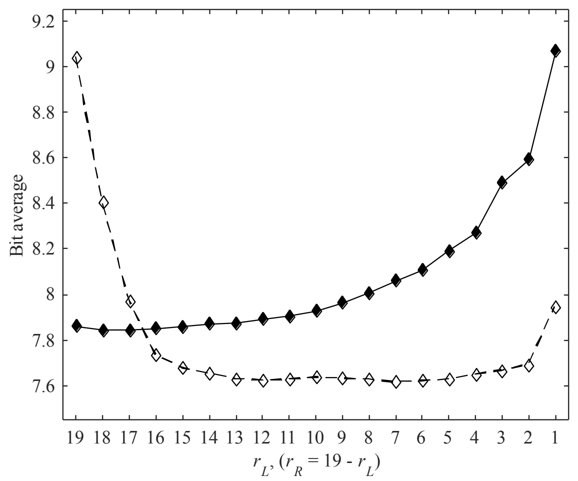


Fig. 5 Bit average for ATrain (continuous line) and TomsDiner (long dashed line) files in dependencies of r_L to r_R proportions

VII. SUMMARY

Using the proposed method (MMSE) for the universal parameter set $\{q, r_L, r_R\} = \{11, 11, 8\}$ the bit average for the test database (used in [11]) was better by 14.95% compared to the universal archiving tool RAR 5.0, as well as 11.81% better than the dedicated SHORTEN 3.6.1 solution. In compare to default mode of MP4-ALS-RM23 our algorithm are better about 2.11%, but still, the results are worse than the best published solutions, such as MP4-ALS-RM23 turned in the best (but slow) mode by using special switches, which is presented in Table III.

TABLE I.
COMPARISON OF CODES FOR BASE 16 AUDIO FILES [11]

Codec	RAR 5.0	Shorten 3.6.1	MP4-ALS-RM23 (default mode)	WavPack 5.1 [17]	Our proposition	MP4-ALS-RM23 (the best mode)
bit average	10.820	10.434	9.352	9.208	9.202	8.718

The main purpose of this work was to show directions of optimization of predictive models, such as choosing the order of coded channels or individual r_R/r_L ratio (for a given file or even each frame).

In the MP4-ALS-RM23 standard, not everything has been fully developed. The proposal of our algorithm is similar in construction to MP4-ALS in basic ALS version without RLSLMS and BGMC mode, but building the next steps of the algorithm we fill the gaps that were omitted in MP4, which will increase overall efficiency in the future. Our algorithm loses with MP4 (the best mode in Table III) because we tested MP4 in the best mode using switches: -z3, -p, -b, providing respectively: RLSLMS mode (in best configuration), long-term prediction, using BGMC codes for prediction residual (instead Rice code). In the current version of our algorithm, using the Golomb code we have a faster implementation than we would use arithmetic coding used in MP4 with BGMC mode. MP4-ALS-RM23 have many switches, which were chosen best for effective compression. Our proposal has a static configuration that is flexible ad-hoc.

Further work using the approach proposed in this paper, with the use of ideas employed, among others, in [11] (introducing cascading combining of successive blocks to minimize prediction errors) will allow to significantly improve the efficiency of forward coder. It is also expected to introduce a better than MMSE technique for the selection of prediction coefficients as well as a more accurate individual selection of the parameters $\{q, r_L, r_R\}$ described here for each super-frame, as in case of MPEG4-ALS.

REFERENCES

[1] S. Andriani, G. Calvagno, T. Erseghe, G. A. Mian, M. Durigon, R. Rinaldo, M. Knee, P. Walland, M. Koppetz, Comparison of lossy to lossless compression techniques for digital cinema, *Proceedings of International Conference on Image Processing ICIP'04*, 24-27 Oct. 2004, vol. 1, pp. 513-516.

[2] C. D. Giurcaneau, I. Tabus, J. Astola, Adaptive context based sequential prediction for lossless audio compression, *Proceedings of IX European Signal Processing Conference EUSIPCO 1998*, Rhodes, Greece, Sept. 1998, vol. 4, pp. 2349-2352.

[3] S. W. Golomb, Run-length encoding, *IEEE Transactions on Information Theory*, July 1966, vol. 12, pp. 399-401.

[4] H. Huang, P. Fränti, D. Huang, S. Rahardja, Cascaded RLS-LMS prediction in MPEG-4 lossless audio coding, *IEEE Trans. on Audio, Speech and Language Processing*, March 2008, vol. 16, no. 3, pp. 554-562.

[5] T. Liebchen , Y. A. Reznik, Improved Forward-Adaptive Prediction for MPEG-4 Audio Lossless Coding, *in 118th AES Convention*, 28-31 May 2005, Barcelona, Spain, pp. 1-10.

[6] E. Ravelli, P. Gournay, R. Lefebvre, A Two-Stage MLP+NLMS Lossless coder for stereo audio, *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)*, Toulouse, France, 14-19 May 2006, vol. 5, pp. V_177-180.

[7] Y. A. Reznik, Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS), *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, Montreal, Quebec, Canada, 17-21 May 2004, vol. 3, pp. III_1024-1027.

[8] R. F. Rice, *Some practical universal noiseless coding techniques*, Jet Propulsion Labolatory, JPL Publication 79-22, Pasadena, CA, March 1979.

[9] T. Robinson, SHORTEN: Simple lossless and near-lossless waveform compression, Cambridge Univ. Eng. Dept., Cambridge, UK, Tech. Rep. 156, 1994, pp. 1-17.

[10] K. Sayood, *Introduction to Data Compression*, 2nd edition, Morgan Kaufmann Publ., San Francisco, 2002.

[11] G. Ulacha, R. Stasiński, Entropy Coder for Audio Signals, International journal of electronics and telecomunications, Vol. 61, No. 2, Poland, pp. 219-224, 2015

[12] R. Yu, S. Rahardja, C. C. Ko, H. Huang, Improving coding efficiency for MPEG-4 Audio Scalable Lossless coding, *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, Philadelphia, PA, USA, 18-23 March 2005, vol. 3, pp. III_169-172.

[13] Yuli You, Audio coding. Theory and applications, 1st edition, Springer, New York 2010.

[14] http://www.losslessaudio.org/

[15] http://www.monkeysaudio.com/

[16] http://www.rarewares.org/test_samples/

[17] http://www.wavpack.com/