Grzegorz ULACHA, Tomasz MĄKA, Piotr DZIURZAŃSKI
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE, WYDZIAŁ INFORMATYKI,
ul. Żołnierska 49, 71-210 Szczecin

Lossless audio compression with a switched prediction model

Dr inż. Grzegorz ULACHA

Grzegorz Ulacha (M'2000, PhD'2004) graduated from the Szczecin University of Technology, where he also defended his PhD thesis. He is working now as an Associate Professor at the Institute of Computer Architecture and Telecommunications of the West Pomeranian University of Technology, Szczecin. His scientific interests are mainly linked with lossless and lossy image and audio coding.

e-mail: gulacha@wi.ps.pl

Dr inż. Tomasz MĄKA

He received the MSc and PhD degrees in computer science from Szczecin University of Technology in 2000 and 2005, respectively. He is currently working as an assistant professor in Faculty of Computer Science & Information Technologies, West Pomeranian University of Technology, Szczecin. His scientific interests include hardware realization of digital signal processing systems and acoustic signal processing techniques.

e-mail: tmaka@wi.ps.pl

Dr inż. Piotr DZIURZAŃSKI

He received the MSc and PhD degrees in computer science from Szczecin University of Technology in 2000 and 2003, respectively. He is currently working as an assistant professor in Faculty of Computer Science & Information Technologies, West Pomeranian University of Technology, Szczecin. His scientific interests include hardware-software co-synthesis, high level synthesis and formal verification.

e-mail: pdziurzanski@wi.ps.pl

Abstract

In this paper there is described a possibility of context switching into a lossless compression system. The context is determined based on the features of the previous signal samples. Each context is associated with an individual predictor. The idea of context switching allows us to choose one of the set of a few predictor models individually for each sample instead of each frame. Consequently, the system adjusts fast in case of rapid signal changes. The system was implemented using the ImpulseC hardware description language and implemented on an FPGA platform.

Keywords: lossless compression, linear prediction, contextual split, ImpulseC.

Bezstratna kompresja audio z przełączanym modelem predekcyjnym

Streszczenie

W nowoczesnych metodach kompresji audio wykorzystuje się zwykle dwa etapy: dekompozycję danych, a następnie kompresję jedną z wydajnych metod entropijnych. Najczęściej do modelowania służy typowy predyktor liniowy rzędu *r*, który jest wartością przewidywaną aktualnie kodowanej próbki na podstawie *r* poprzednich próbek sygnału. Kluczową rolę odgrywa tu sposób doboru współczynników danego modelu. Mogą być one ustalone na stałe, statyczne w obrębie jednej kodowanej ramki, jak i w pełni adaptacyjne. Główną propozycją wzrostu efektywności kompresji zaprezentowaną w tej pracy jest wprowadzenie przełączania kontekstów, które wyznacza się na podstawie cech sygnału poprzednich próbek. Każdemu kontekstowi przypisany jest indywidualny predyktor. W artykule przedstawiono podział na 2 oraz 3 konteksty (tab. 1). Przedstawiono metodę statyczną uwzględniającą zależności międzykanałowe, a także kodowanie międzykanałowe z przełączaniem kontekstów. Aby sprawdzić możliwości uogólnienia i uproszczenia pomiarów, wybrano zestaw utworów muzycznych. Proponowana metoda w 60% przypadków skutkowała zmniejszeniem średniej bitowej. Dysponując pełnym zestawem wyników użycia 140 deskryptorów dla wybranych utworów, można spróbować wybrać kilka deskryptorów dających najlepsze rezultaty, a następnie zastosować je do innych utworów testowych. Zaproponowany algorytm został zaimplementowany w układzie FPGA z rodziny Virtex 5 wykorzystując język opisu sprzętu ImpulseC (tab. 3).

Słowa kluczowe: bezstratna kompresja, predykcja liniowa, podział kontekstowy, ImpulseC.

1. Introduction

In modern compression method techniques, two stages are usually utilized. The first of them is data decomposition, which is followed by one of the efficient entropy methods. Among these methods the most effective are arithmetic encoding, Huffman encoding and its modifications, such as Golomb and Rice codes. In case of lossless audio encoding, the Gilbert-Moore block code, which is a combination of arithmetic with Golomb-Rice codes, is often used [1].

There are two basic directions in the data modelling domain. In the first of them, linear or non-linear prediction is used. The second type utilizes DCT (MPEG-4 SLS [2]) or wavelet transforms.

Usually, a typical linear predictor of order *r* is used. It is the expected value of the sample to be encoded based on the previously encoded *r* signal samples. The usage of the linear predictor allows us to encode only the predictor errors, i.e., the differences *e* between the real and expected values (rounded to an integer value), which usually are values close to zero. In consequence, we obtain a differential signal, whose error distribution is similar to the Laplace distribution. Then it may be effectively encoded using one of static or adaptive entropy methods. It is crucial to choose favourable coefficients of the given model. They may be constants (constant predictors), static inside a single encoded frame (changing during transition to a consecutive frame), or fully adaptive (their modification can take place even after each sample encoding).

In a static method, increase of the prediction order decreases the mean-square error, but there is no guarantee of compression effectiveness increase. This is mainly caused by the increase of the coefficient number placed in the header. It is thus recommended to find, e.g., one value *r* being a compromise for the whole file or to measure the entropy or the encoded frame length for consecutive prediction ranges. This solution is used in MPEG-4 ALS, where a frame of the length equal to about 43 ms was chosen.

2. Contextual split

In this paper the main proposal of the compression effectiveness increase consists in introduction of a context switching, which is determined based on the features of the previous signal samples. Each context is associated with an individual predictor. The idea of context switching allows us to choose one of the set of *h* predictor models individually for each sample instead of each frame. As a result, we obtain a possibility of fast adaptivity for rapid signal alterations.

The first proposed version of this technique utilizes only $h = 2$ contexts for long-term frames (in our experiments the frame length was equal to the samples number in one channel in the encoded file); the bit average can be then computed with the following formula:

$$L = H(S) + \frac{m \cdot r \cdot h}{N}, \tag{1}$$

where $H(S)$ is the zeroth-order entropy of the prediction error signal, r is the prediction length, and m determines the number of bits used for storing the prediction coefficient. The first stage is to determine the average value from the absolute value of the differences between neighbouring samples in a frame:

$$S_{\text{ave}} = \frac{1}{N-1}\sum_{n=2}^{N}\left|x(n)-x(n-1)\right|. \tag{2}$$

After rounding to an integer, this value is transferred to a decoder. This criteria for context switching is based on a level of local changes between the closest samples. During the encoding stage, for each sample x(n) the following q value is computed:

$$q = \max_{j=\{1,\ldots,9\}}\left\{\left|x(n-j)-x(n-j-1)\right|\right\}, \tag{3}$$

Then for each value of q a two-stages quantizer is used with the threshold set to $\alpha \cdot S_{\text{ave}}$, where α = 2.5 was selected in an experimental way. In this way samples of each channel are split into two subsets and they are used as a basis to determine predictors different for each context using the MMSE. Despite the fact that it requires the doubled number of coefficients per channel, it produces better results than doubling the prediction range using a single model. Additionally, it is possible to measure the average eight time, choosing the best result for eight values α_i = {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4}. Then the average bit rate slightly decreases. To simplify the average bit rate computation, in our experiments the prediction coefficient is written in m = 32 bits.

Increasing the number of contexts to three, it is required to provide two quantization thresholds, q, computed with (3), the remaining computations are similar as in the case of two contexts, described above. In an experimental manner, pair $\{\alpha_1; \alpha_2\}$ = {1.5; 4} was chosen as the best trade-of. 16 fragments of the testing files were measured for r = 200, obtaining the bit average equal to 10.413. The further efficiency increase is possible when instead of one measurement we perform it eight times., selecting the best quantization pair thresholds. Based on the experimental results, we chose 8 best pairs $\{\alpha_{1i}; \alpha_{2i}\}$. Choosing the best measurement out of 8, for tested benchmarks we managed to decrease their bit-average (for r = 200) to 10.396.

Tab. 1. Comparison of the bit-averages for statistic methods with 600 coefficients written in the file header
Tab. 1. Porównanie średnich bitowych dla metod statycznych o 600 współczynnikach zapisywanych w nagłówku pliku

Test name	r = 600, context-free	r = 300, 2 contexts	r = 300, 2 contexts (8 measure-ments)	r = 200, 3 contexts	r = 200, 3 contexts (8 measure-ments)
ATrain	8.47830	8.40454	8.39622	8.40549	8.39570
BeautySlept	10.04539	10.27163	10.26925	10.41231	10.40850
chanchan	10.79826	10.72185	10.70160	10.68729	10.67666
death2	10.47433	7.92506	7.69276	7.65953	7.65932
experiencia	12.26870	12.21207	12.20447	12.20256	12.19955
female_speech	9.13827	8.31041	8.29872	8.22219	8.21096
FloorEssence	11.67929	11.66518	11.65198	11.62388	11.62388
ItCouldBeSweet	11.40019	11.34135	11.33980	11.33286	11.32874
Layla	11.20194	10.83219	10.82088	10.79939	10.79495
LifeShatters	11.35826	11.35419	11.35343	11.35632	11.35483
macabre	10.06908	10.03100	9.96025	9.99353	9.99334
male_speech	8.18880	8.02271	8.01902	7.98155	7.96473
SinceAlways	12.46821	12.31252	12.30736	12.30567	12.28545
thear1	12.10544	12.11046	12.11046	12.12290	12.11858
TomsDiner	9.83010	9.74519	9.74135	9.70890	9.69194
velvet	11.57260	11.64013	11.57596	11.79359	11.63399
Average	10.69232	10.43128	10.40272	10.41300	10.39632

In the next experiment, we analysed the influence of the context number, maintaining the constant total number of 600 prediction coefficient per channel. The results are provided in Tab. 1. We measured the context-free method for r = 600 (column 2), and then the method with two-context split for r = 300 and the quantisation threshold α = 2.5 (column 3) and with the selection of the best out of 8 quantization thresholds (column 4). The last two measurements were made for the method with three-context split, r = 200, and quantization thresholds $\{\alpha_1; \alpha_2\}$ = {1.5; 4} (column 5) and with the best quantization threshold pair out of 8 analysed (column 6). The higher is context number, the lower is bit-average L. The improvement between three- and two-context versions is lower than the one between two-context and the context-free ones.

Since the increase of the context number makes selection of the proper thresholds more difficult, from the experimental results we can conclude that three-context solution can be treated as a good trade-off. The idea of the contextual split may be implemented in the existing approaches, e.g., in the RLS block in five-stage cascade method described in [3].

3. Inter-channel encoding with context switching

Since some dependences can be usually observed between channels, it is possible to benefit from this fact, using two different models for the left (prior encoded) and the right channels. According to the experimental results, it is beneficial to use the proportion range r_L:r_R equal to 2:1, where L and R denotes the left and the right channel, respectively. The first parameter indicates which channel is prior encoded. At the second stage, the quantization thresholds are selected (3-context split version as described in section 2). The last stage consists of the selection of prediction order in the interchannel mode: $r = r_L + r_R = 120$. As a result, the average was decreased to 9.656 bit per sample.

4. Other context switching techniques

It is worth considering whether the feature described in Section 2 with equation (3), which computes number q, can be replaced by other rule, which would allow us to select context based on a simple rule using 3-state quantizer with two thresholds equal to $(\alpha_1 \cdot S_{\text{ave}}, \alpha_2 \cdot S_{\text{ave}}\}$, where S_{ave} is treated as the bit-average of all values q measured for the encoded signal.

In the following experiments, a set of 140 descriptors typical for the sound classification domain was used. For each encoded sample, it is necessary to determine the value of q, using the rule of measurement the previous frame (in the experiment, frames of the length 30ms have been used). The complexity of this process is worth stressing: 140 various descriptors are determined, and for each of them 32 sets of quantization thresholds is checked (it is necessary due to various features of the normalized descriptor values) and for each of these cases an encoding of the test is performed. Finally, there is chosen the descriptor leading to the shortest bit-average of the test.

To check the possibility of the measurement generalization and simplification, 5 tests out of 16 were chosen that varied from each other to a considerable extent (speech, classical music, rock music, music with vocal). After the measurements, in 3 out of 5 cases the decrease of the bit-average was observed. In the remaining 2 cases (experiencia, male_speech), where a worse result was obtained, the basic technique described in Section 2 was still used. As a result, for 5 tests the average result decreased from 9.416 (in basic method) to 9.389 bit per sample in method with the best descriptor selection. Tab. 2 includes the results of these experiments.

Tab. 2. Bit-average measurements with 3 contexts and the best descriptor (out of considered 140) selection
Tab. 2. Pomiar średniej bitowej z wykorzystaniem 3 kontekstów i doborem najlepszego deskryptora spośród 140 badanych

Test name	Bit average	Method	Quantization thresholds
ATrain	**8.31331**	s_1	0.3; 1.0
experiencia	11.69804	s_2	0.3; 1.0
LifeShatters	**11.07703**	s_3	1.0; 2.5
macabre	**9.96893**	s_8	0.8; 2.5
male_speech	5.95346	s_4	0.3; 1.5

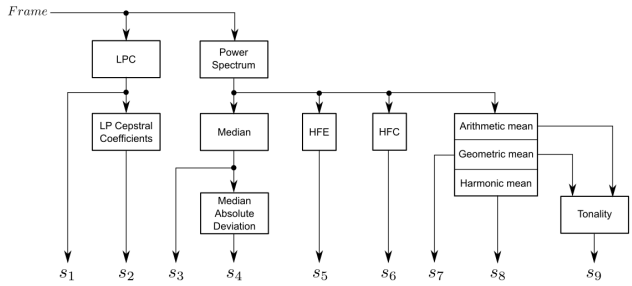

Fig. 1. Process of features vector generation for context switching
Rys. 1. Proces generowania wektora cech do przełączania kontekstów

Having all the results of the 140 descriptors utilization for these 5 selected tests, it is possible to select a few descriptors leading to the best average results, and then to apply them to the whole set of 16 tested tests. Using this technique we selected 5 descriptors: s_5, s_6, s_7, s_8, s_9. The features vector extraction process is depicted in Fig. 1. The vector is calculated for each frame extracted from the source signal with one sample step between consecutive frames. After extensive experimentation we found nine features usable in context switching. Two features were extracted in the time domain - linear prediction coefficients and linear prediction cepstral coefficients [4]. In our case, the 10th order filter was used, where 7th and 2nd element of the obtained vectors were exploited as s_1 and s_2 features, respectively. Other features were calculated in the frequency domain, where in addition to selected statistical properties of the power spectrum (s_3, s_4, s_7 and s_8), three features were extracted: HFE (s_5), HFC (s_6) and Tonality (s_9). The high-frequency energy HFE feature is calculated as the energy of frequency components above 4kHz [5]. Similarly, the HFC descriptor also describes the energy of higher frequencies as it is defined by a weighted energy function, where weights increase with increasing frequencies of the power spectrum [6]. The last descriptor of the feature vector, Tonality, denotes whether a signal is tone- or noise-like and is calculated using a ratio of spectral flatness expressed in decibels to the reference level [7]. Applying these descriptors to the technique described in this paper, the average bit rate decreases from 9.656 to 9.641.

5. Implementation

Since the original algorithms described in the previous section were realized in C language, we chose one of the existing C-based hardware description language to implement the hardware-targeted counterpart of the source code, namely ImpulseC [8]. It is an extension of ANSI C with new data types, aimed at hardware synthesis, and new functions and directives for steering the hardware implementation. The code is executed in the so-called processes, which communicate with other processes using streams, signals, shared memory and semaphores. As the processes are to be realized in hardware, they may benefit from various ImpulseC optimization techniques, such as loop unrolling or pipelining. The first of these techniques are quite important in our system, as numerous computations in data-dominated algorithms are independent from each other. However, there are usually no enough resources for generating hardware for each iteration, thus some kind of clustering is necessary. This trade-off between the computation time and target chip area can be established during a series of experiments. This is also the path followed by the authors. After some code modifications, aiming at improving its hardware realization, we obtained our final code to be implemented in hardware. These modifications included division of the code into coarse-grain partitions to be implemented in parallel. The transformation is always a trade-off between the computational time and resource utilization, hence we analyzed the impact of each module onto the final realization in terms of particular functional blocks, such as adders, multipliers etc., estimated DSP blocks (present in our target FPGA chip) and number of computational stages. To estimate the impact of these modifications into the target chip parameters, we used Stage Master Explorer tool from the ImpulseC CoDeveloper package. This tool computes two parameters, Rate and Max Unit Delay (MUD), which approximate the performance of future hardware implementation. It is worth stressing that these parameters are computed instantly, in contrast with long-lasting hardware implementation.

At the last stage, we used Xilinx ISE to perform an implementation of the core in Virtex5 FPGA device (XC5VSX50T, Virtex 5 ML506 Evaluation Platform) and got the following device utilization: 2804 Slice Registers, 7707 Slice LUTs, 2032 fully used LUT-FF pairs, 64 DSP48Es.

The above assignments mean that less than 20 percent of the device were used, leaving more than 80% for the router and the remaining cores. The more detailed resource requirements and the elementary operators used for the implementation of the presented algorithms are given in Tab. 3.

Tab. 3. Number of the synthesized resource of various types
Tab. 3. Liczba zsyntetyzownych zasobów różnych typów

Resource type	Number of used resource
Adder(s)/Subtractor(s) (8 bit)	32
Adder(s)/Subtractor(s) (32 bit)	114
Multiplier(s) (32 bit)	25
Multiplier(s) (64 bit)	3
Divider(s) (32 bit)	10
Comparator(s) (2 bit)	1
Comparator(s) (32 bit)	44

6. Conclusions

The idea of contextual split, presented in the paper, can be implemented in the most effective, known from the literature, sound compression systems, e.g. in the RLS system in five-stage cascade method described in [5]. It allows us, together with other solutions, to increase the compression level. The proposed algorithm is also easily implementable in hardware, leading to a solution requiring relatively low resources.

The research work presented in this paper was supported by Polish National Science Centre (grant no. N N516 492240).

7. References

[1] Reznik Y.A.: 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.
[2] Yu R., Rahardja S., Ko C. C., Huang H.: 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.
[3] Huang H., Fränti P., Huang D., Rahardja S.: 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.
[4] Huang X., Acero A., Hon H.: Spoken Language Processing - A Guide to Theory Algorithms and System Development, Prentice Hall PTR, 2001.
[5] Monson B., Lotto A., Ternstrom S.: Detection of high-frequency energy changes in sustained vowels produced by singers, J. Acoust. Soc. Am. 129 (4), April, 2011.
[6] Masri P., Bateman A.: Improved modelling of attack transient in music analysis-resynthesis. In: Proceedings of the International Computer Music Conference, Hong-Kong, 1996.
[7] Johnston J.: Transform Coding of Audio Signals Using Perceptual Noise Criteria, IEEE Journal On Selected Areas In Communications, Vol. 6, No. 2, February, 1988.
[8] Impulse Accelerated Technologies, Accelerating HPC and HPEC Applications Using Impulse C, Reconfigurable Systems Summer Institute (RSSI), Urbana, IL, July 17-20, 2007.

otrzymano / received: 20.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*