Grzegorz ULACHA, Tomasz MĄKA, Piotr DZIURZAŃSKI
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY,
Żołnierska 49, 71-210 Szczecin

On hardware implementation of the Context-based Lossless Audio Codec

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 assistant professor in Faculty of Computer Science & Information Technologies, West Pomeranian University of Technology, Szczecin. His scientific interests are mainly linked with lossless and lossy image and audio coding.

e-mail: gulacha@wi.zut.edu.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.zut.edu.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.zut.edu.pl

Abstract

A modification of the most efficient version of MPEG4 Lossless Audio with extension of the RLS (Recursive Least Square) and NLMS (Normalized Least-Mean-Square) blocks is described in the paper. Moreover, a segmentation block influencing the selection of proper predictive modelling parameters is introduced. These blocks have been implemented in hardware description language ImpulseC and synthesised into a reprogrammable device from the Xilinx Virtex5 family.

Keywords: lossless compression, linear prediction, LMS, RLS, ImpulseC.

Sprzętowa realizacja bezstratnej kompresji audio z przełączanym modelem predekcyjnym

Streszczenie

W pracy zaprezentowano rozwinięcie najwydajniejszej wersji MPEG4 Lossless Audio przez rozbudowanie bloków RLS (Recursive Least Square) i NLMS (Normalized Least-Mean-Square), wprowadzając przy tym blok segmentacji wpływający na dobór odpowiednich parametrów modelowania predykcyjnego. Zwiększono nie tylko rząd predykcji w poszczególnych blokach modelowania, ale też rozwinięto metodę NLMS do ES-NLMS i dobrano eksperymentalne wartości współczynników uczących, a także odpowiednie proporcje liczby współczynników predykcji w trybie stereo. Ponadto opracowano własny blok adaptacyjnego kodera arytmetycznego, w którym wykorzystano adaptacyjne kodowanie Golomba-Rice'a. Każdy z tych bloków został przygotowany do potrzeb implementacji sprzętowej. Bloki RLS i NLMS wykorzystują dane pochodzące z modułu segmentacji, co ma pozytywny wpływ na efektywność kompresji. Głównym zadaniem bloku segmentacji jest wydzielenie segmentów różniących się zawartością akustyczną. Wykorzystano na tym etapie dwa podejścia do segmentacji – pierwsze z nich realizuje podejście polegające na porównywaniu sąsiednich ramek sygnału w przestrzeni cech składającej się z 12 współczynników MFCC (Mel-Frequency Cepstral Coefficients) i drugie polegające na ocenie dwóch modeli w przestrzeni cech w użyciem typowego podejścia opartego o Bayesowskie kryterium informacyjne. Wyniki uzyskane z obu technik są następnie łączone w celu kompensacji potencjalnych błędów określających granice segmentów. Dla każdego z uzyskanych segmentów wyznaczany jest uśredniony wektor cech MFCC, który dostarczany jest do bloków RLS i NLMS jako źródło do określania kontekstu. Bloki funkcjonalne zostały zaimplementowane w języku opisu sprzętu ImpulseC oraz dokonano syntezy do układu reprogramowalnego z rodziny Xilinx Virtex5.

Słowa kluczowe: bezstratna kompresja, predykcja liniowa, LMS, RLS, ImpulseC.

1. Introduction

Despite the omnipresence of lossy audio compression, lossless techniques are still applied in a number of domains, including recording archiving, post-production in recording studios, or even selling recordings for demanding customers, not satisfied with the MP3 format quality.

In Fig. 1, a scheme of the proposed extension of the most efficient MPEG4 Lossless Audio [1] technique is shown. There is presented a simplified data flow, where the DPCM (Differential Pulse Code Modulation) block is an introductory derivative block, followed by subsequent predictive blocks determining the lowest prediction error. The proposed extended versions of RLS (Recursive Least Square) and NLMS (Normalized Least-Mean-Square) benefit from a segmentation block influencing selection of appropriate predictive modelling parameters. In the last stage, the encoding of prediction errors is performed with use of the proposed adaptive arithmetic encoder employing arithmetic Golomb-Rice coding. These blocks were prepared in a way to be well implemented in hardware by enhancing parallelism and introducing pipeline stages.

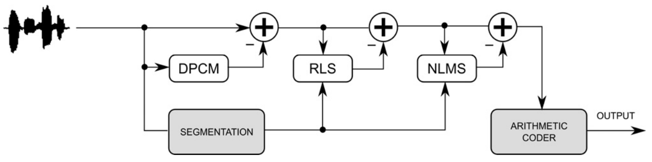

Fig. 1. Structure of the proposed lossless audio codec
Rys. 1. Struktura proponowanego bezstratnego kodeka dźwięku

2. Audio segmentation module

The segmentation module involves two segmentation techniques and a context generation stage. The determination of audio segment boundaries is performed using metric and model based techniques run in parallel. In the metric-based approach, segment boundaries are calculated by the content similarity between adjacent frames. For each frame, a feature vector using $d = 12$ coefficients of MFCC (Mel-Frequency Cepstral Coefficients) [2] feature is computed and then compared with the feature vector of the neighbouring frame using the Euclidean distance. The obtained similarity trajectory is thresholded adaptively in order to define change points in the input signal. The possible change points are determined for all positive peaks where their values are greater than the threshold value.

In the same time, a segmentation technique based on the delta-BIC (Bayesian Information Criterion) approach [3] is executed.

The delta-BIC method compares two models: 1) data from a single Gaussian distribution; 2) data from two Gaussians.

The data is obtained by splitting the input data at position i to the left and right-side windows. The comparison is performed by subtracting BIC values of these two models (for both windows) $\Delta BIC(i) = N_1 \log|\Sigma_1| - N_2 \log|\Sigma_2| - N \log|\Sigma| - P$, where: i is the position of a possible change-point, $i = 1, ..., N$, N_1, N_2, N denotes the size of the left-side, right-side and the whole data windows, $|\Sigma_1|, |\Sigma_2|, |\Sigma|$ are the determinants of covariance matrices of the respective data windows and $P = 45\log N$ (calculated for $d = 12$) [3]. If the value of expression (2) is positive, then the model of two Gaussians is favored. The possible change point exists at position i if the inequality $\max_i \Delta BIC(i) > 0$ is satisfied.

The resulting sets of the change points obtained from both techniques were fused together utilizing a simple procedure presented in [4]. Finally, context information was generated for each segment. The previously calculated MFCC vectors in each segment were averaged and send to the RLS and NLMS blocks as the context information.

3. RLS Module and prediction error encoding

RLS is an adaptive method for obtaining a linear predictive model using minimization of a mean square error based on the samples encoded previously. For each subsequently encoded sample, an adaptation of prediction coefficients is performed, which makes the prediction model take into account local signal alterations.

Utilising the idea presented in [5] to increase the accuracy of the RLS method, an input data stream is comprised of the elements being a difference between subsequent samples, i.e., $g(n-j) = x(n-j) - x(n-j-1)$ (an introductory DPCM module). Assuming that the left channel is encoded before the right one, the input vector at time stamp n is of the following form for the left channel: $\mathbf{g}(n) = [g(n-1), g(n-2), ..., g(n-r)]^{\mathrm{T}} = [x_L(n-1) - x_L(n-2), x_L(n-2) - x_L(n-3), ..., x_L(n-r_L) - x_L(n-r_L-1), x_R(n-1) - x_R(n-2), x_R(n-2) - x_R(n-3), ..., x_R(n-r_R) - x_R(n-r_R-1)]^{\mathrm{T}}$, whereas for the right channel it can be described with formula $\mathbf{g}(n) = [x_R(n-1) - x_R(n-2), x_R(n-2) - x_R(n-3), ..., x_R(n-r_L) - x_R(n-r_L-1), x_L(n) - x_L(n-1), x_L(n-1) - x_L(n-2), ..., x_L(n-r_R+1) - x_L(n-r_R)]^{\mathrm{T}}$.

Due to the characteristic of the $\mathbf{g}(n)$ vector, the predicted values are determined using equations: $\hat{x}_L(n) = x_L(n-1) + \mathbf{w}_L^T(n) \cdot \mathbf{g}_L(n)$ and $\hat{x}_R(n) = x_R(n-1) + \mathbf{w}_R^T(n) \cdot \mathbf{g}_R(n)$. During the coefficient adaptivity process, operations on matrix $\mathbf{K}(n)$ are performed, which is initialised with identity matrix $\mathbf{I}$ multiplied by constant $\varepsilon \ll 1$. In the proposed version of RLS, the value of ε was determined in an experimental way as $\varepsilon = 0.0002$ and matrix $\mathbf{K}(n)$ is initialised as $\mathbf{K}(0) = 0.0002 \cdot \mathrm{diag}(\overline{\mathbf{d}})$, where vector $\overline{\mathbf{d}} = [\overline{d}_1, \overline{d}_2, ..., \overline{d}_r]^T$ includes a set of weight $\overline{d}_j$ obtained according to the formula $\overline{d}_j = 1/\sqrt{j}$, $\mathrm{diag}(\overline{\mathbf{d}})$ denotes a square matrix $r \times r$, whose main diagonal includes elements from vector $\overline{\mathbf{d}}$ and the remaining elements are zeros. The prediction coefficient vector is initialised with 0s: $\mathbf{w}(0) = \mathbf{0}$. During each sample encoding, the first step is to determine the expected value, which allows us to calculate a current prediction error. The second step is to compute vector $\mathbf{u}(n+1)$ [5]: $\mathbf{u}(n+1) = \mathbf{K}(n) \cdot \mathbf{g}(n)$. Next, a learning rate, μ, is determined $\mu = 1/(q_+ + \mathbf{g}(n)^T \cdot \mathbf{u}(n+1))$.

In the next step, a matrix adaptation of $\mathbf{K}(n)$ is performed $\mathbf{K}(n+1) = f_{\mathrm{adapt}} \cdot (\mathbf{K}(n) - \mu \cdot \mathbf{u}(n+1) \cdot \mathbf{u}(n+1)^T)$, whose value is substituted to equation for computing adaptation for prediction coefficient vector $\mathbf{w}(n) = [w_1, w_2, ..., w_r]^T$: $\mathbf{w}(n+1) = \mathbf{w}(n) + \widetilde{e}(n) \cdot \mathbf{K}(n+1) \cdot \mathbf{g}(n)$, where $\widetilde{e}(n)$ is a current prediction error with limit $\varphi = 2$ (chosen experimentally to prevent oscillations of the prediction value) and is computed in the following manner: $\widetilde{e}(n) = \mathrm{sgn}(e(n)) \cdot \min\{|e(n)|, \varphi\}$.

The coefficient $f_{\mathrm{adapt}} = 1.0005$ was determined experimentally, using experiments from [6]. Similarly the parameter q_+ was determined as $q_+ = 150$. The best results were obtained for the range $r = 39$, where inter-channel encoding was used, for $r_L = 18$, $r_R = 11$.

Tab. 2. The number of high-level resource of various types required for realising particular functional blocks

Tab. 2. Liczba zasobów wysokiego poziomu wymaganych do realizacji poszczególnych bloków funkcjonalnych

	Delta-BIC segmentation	DPCM + RLS	NLMS	Arithmetic coder
Adder(s)/Subtractor(s) (1 bit)	0	0	0	14
Adder(s)/Subtractor(s) (5 bit)	6	0	0	2
Adder(s)/Subtractor(s) (6 bit)	0	0	2	11
Adder(s)/Subtractor(s) (7 bit)	0	0	25	0
Adder(s)/Subtractor(s) (8 bit)	0	0	1	0
Adder(s)/Subtractor(s) (9 bit)	6	0	0	0
Adder(s)/Subtractor(s) (11 bit)	0	14	0	0
Adder(s)/Subtractor(s) (13 bit)	0	0	0	4
Adder(s)/Subtractor(s) (14 bit)	0	0	0	22
Adder(s)/Subtractor(s) (16 bit)	0	7	2	27
Adder(s)/Subtractor(s) (32 bit)	38	9	16	111
Multiplier(s) (9 bit)	0	0	0	2
Multiplier(s) (17 bit)	0	0	0	18
Multiplier(s) (32 bit)	13	7	7	11
Multiplier(s) (64 bit)	0	0	2	0
Multiplier(s) (96 bit)	0	0	3	0
Divider(s) (32 bit)	6	1	3	11
Comparator(s) (1 bit)	0	0	0	9
Comparator(s) (2 bit)	1	1	1	1
Comparator(s) (9 bit)	0	0	2	0
Comparator(s) (16 bit)	0	0	0	4
Comparator(s) (17 bit)	0	8	4	20
Comparator(s) (32 bit)	23	3	4	62
Total	93	50	72	329

Tab. 3. Zużycie zasobów układu z rodziny Virtex5 w zaimplementowanych blokach funkcjonalnych

Tab. 3. Resource utilization of a Virtex5 family device for the implemented functional blocks

Block	FF Used	LUT Used
Delta-BIC segmentation	927	20791
NLMS	840	1253
DPCM + RLS	397	652
Arithmetic coder	2544	7102

4. RLS-NLMS cascade connection

Owing to the cascade connection of the prediction methods which has been proposed in [1] and our modifications, it is possible to get the result where the average bit rate is lower for NLMS prediction higher ranges in the case of a majority of testing files (which is not typical in case of the non-cascade NLMS method). This property is important not only due to the possibility of uniformity of the r value for an arbitrary audio file. It also allows us to remove mutual dependencies between distant samples for, e.g., $r = 500$ (which has not been achieved in RLS). Moreover, in this configuration the NLMS block does not require the usage of the multi-channel version, since inter-channel dependencies are removed in the RLS block, which becomes an initial block performing an introductory data modelling in this cascade solution. Its scheme is presented in Fig. 1.

In this solution, $\mathbf{v}(n)$ becomes an input data for the NLMS method, whose values originate from the output of the RLS method, i.e., $\mathbf{v}(n) = [v(n-1), v(n-2), ..., v(n-r)]^{\mathrm{T}} = [x(n-1) - x(n-2) - x_{RLS}(n-1), x(n-2) - x(n-3) - x_{RLS}(n-2), ..., x(n-r) - x(n-r-1) - x_{RLS}(n-r)]^{\mathrm{T}}$. It is possible to improve the learning rate by introducing the meaning of the signal samples that is non-decreasing with their distance. It is guaranteed thanks to the usage of the experimentally selected scaling coefficient $1/(i+1)^{0,8}$:

$$\mu(n) = \left(\mu \cdot \sum_{i=0}^{r-1} \frac{1}{(i+1)^{0,8}}\right) \Bigg/ \left(1 + \sum_{i=0}^{r-1} \frac{1}{(i+1)^{0,8}} \cdot v^2(n-i)\right).$$

The basic equation for learning rate adaptivity can be extended similarly by introducing scalar coefficients, which are situated at the diagonal of matrix $\mathbf{C}$ = diag([0.995^1 0.995^2 ... 0.995^r]), to the formula for determining the prediction coefficient adaptivity (this idea is taken from the Exponentially weighted Stepsize NLMS, ES-NLMS, method) [7]. Its value (0.995) was determined in an experimental way: $\mathbf{w}_{NLMS}(n+1) = \mathbf{w}_{NLMS}(n) + \mu(n) \cdot e(n) \cdot \mathbf{C} \cdot \mathbf{v}(n)$. The initial value of the prediction coefficient vector is $\mathbf{w}_{NLMS}(0) = [1, 0, \dots, 0]^T$. The value of parameter μ was found experimentally and for the encoded 16-bit samples is equal to $\mu = 2^{-8}$. The cascade connection gives even better results, a number of the NLMS blocks are connected, where the output data of the k-th NLMS block becomes an input data of the (k+1)-th block (schematically, in Fig. 1 additional NLMS should be added, as shown in [1]).

The majority of the audio signal prediction error encoding is based on modifications of the Golomb-Rice [8] codes. To obtain higher efficiency than the classic Golomb code, an adaptive version of an arithmetic encoder should be implemented [9]. It allows us to suit currently encoded samples to a proper probability distribution more accurately thanks to the context switching technique.

In the proposed codec an adaptive Golomb code is used. Its input data is additionally compressed using two adaptive arithmetic encoders (tuned separately to encode the values of u_G and v_G). A split into 9 classes is applied, where in each class the best among 38 defined probability distributions is selected adaptively after each consecutive sample encoding.

5. Implementation

The algorithms described earlier in this paper were implemented in C language. Since manual rewriting of these codes into any hardware description language would be a time-consuming process prone to errors, we decided to investigate into C-based HDLs. Taking into account our previous negative experience with SystemC [10], we preferred to use a less complex language, whose genesis was purely hardware implementation, without simulation or system-level-oriented overhead. We chose ImpulseC [11], an extension of ANSI C with hardware-oriented data types and new functions and directives for steering the hardware implementation. In this language, the code is split into the so-called processes, which can communicate each other using streams. This property is of particular importance for our task due to the streaming nature of the proposed system architecture.

The hardware implementation benefits from various ImpulseC optimization techniques, such as loop unrolling or pipelining. Especially the first technique was used extensively in the proposed system, due to independence of data in numerous loop bodies. In a few cases, additional loop optimisation techniques (especially loop inversion, loop interchange, and loop reversal) have to be applied manually to enhance fine-grain parallelism. However, where a particular type of data dependence prohibited parallel loop realisations, the code was split into a series of blocks for pipeline processing.

However, the target reprogrammable chip (XC5VSX50T in Virtex 5 ML506 Evaluation Platform) is equipped with insufficient resources to fit the purely parallel version of the code. During a trial-and-error search some loops were chosen to be executed serially. Since this transformation is a typical trade-off between computational time and resource utilization, we analysed the impact of each modification 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 a quite handy Stage Master Explorer tool from the ImpulseC CoDeveloper package. This tool computes two parameters, Rate and Max Unit Delay (MUD), which approximates the performance of future hardware implementation. It is worth stressing that these parameters are computed instantly, in contrast with long-lasting hardware implementation. The final high-level resource utilization, indicated by ImpulseC CoDeveloper package, is presented in Tab. 2.

At the last stage, we used Xilinx EDK to perform an implementation of the core in Virtex5 FPGA device and got the device utilization presented in Tab. 3. These blocks use about 90% of the chip's resources. Note, however, that in the final implementation we enhanced parallelism, which led to increasing the resource utilization. The target chip is also perceived as a rather limited one in terms of available resources, not mentioning its a bit obsolete architecture.

6. Conclusions

The proposed extension of MPEG4 Lossless with modification of the RLS and NLMS blocks aims at increasing the predictive modelling quality. Other factors improving the prediction are the introducing of a segmentation block influencing the selection of appropriate predictive modelling parameters and an adaptive arithmetic encoder employing the adaptive Golomb-Rice code. These blocks were realised in a hardware description language and fit into rather limited resource of the target FPGA device. A number of decision decreasing the target chip area, but also leading to slower work of the target device were made. These trade-off decisions were aided by handy tools from the ImpulseC CoDeveloper package, allowing us to analyse a vast subset of design space quite rapidly.

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

7. References

[1] 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.

[2] Rabiner L., Schafer R.: Theory and Applications of Digital Speech Processing, Pearson Higher Education, Inc., 2011.

[3] Chen S. and Gopalakrishnan P.: Speaker, environment and channel change detection and clustering via the bayesian information criterion, In In Proc. DARPA Broadcast News Transcription and Understanding Workshop, 1998.

[4] Maka T., Dziurzanski P.: Feature Contours Fusion for Determining Segment Boundaries in Audio Data, International Conference on Systems, Signals and Image Processing - IWSSIP 2014, 12-15 May, Dubrovnik, Croatia, 2014.

[5] 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.

[6] Bekkouche H., Barret M.: Adaptive multiresolution decomposition: applications to lossless image compression, Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02), Orlando, Florida, USA, May 2002, vol. 4, pp. IV_3533-3536.

[7] Makino S., Kaneda Y., Koizumi N.: Exponentially weighted stepsize NLMS adaptive filter based on the statistics of a room impulse response, IEEE Transactions on Speech and Audio Processing, Vol. 1, No. 1, 1993, pp. 101-108.

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

[9] Giurcaneau C. D., Tabus I., Astola J.: 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.

[10] Dziurzański P.: Pros and cons of system-level synthesis of data-dominated algorithms, Electronics - Constructions, Technologies, Applications, vol. 51, no. 10, pp. 169-171, 2010.

[11] Impulse Accelerated Technologies, Accelerating HPC and HPEC Applications Using Impulse C, Reconfigurable Systems Summer Institute (RSSI), Urbana, IL, July 17-20, 2007.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*