The FlexCode Source-Channel Coding Approach for Audio and Speech Transmission

Laurent Schmalen, Thomas Schlien, and Peter Vary
Institute of Communication Systems and Data Processing (ind)
RWTH Aachen University, Germany
{schmalen|schlien|vary}@ind.rwth-aachen.de

***Abstract*— The *FlexCode* project is a joint research project of KTH Stockholm, RWTH Aachen University, Ericsson AB, Nokia, and Orange/France Telecom under the umbrella of the sixth framework programme of the European Commission (`http://www.flexcode.eu`). In this paper we present the generic channel coding platform developed within the *FlexCode* project. This channel encoder provides iterative source-channel decoding at the receiver in order to achieve near-capacity transmission of the source coder parameters. The structure of the encoder allows to flexibly adapt instantaneously the source and channel coding rates according to varying conditions. The approach handles both main operating modes of the *FlexCode* project: constrained resolution scalar quantization and constrained entropy scalar quantization which lead to either constant or variable bit rates.**

I. Introduction

The increasing heterogeneity of communication networks and the variability in user requirements form a challenge for source and channel coding algorithms. To address this challenge, the aim of the *FlexCode* project is to create a speech and audio coder that can adapt instantaneously to network and user requirements. The channel coder can flexibly select the required coding rate depending on the current network conditions, the available data rate and the computational power available at the terminal or base station. Thus the source coder can be set up according to the required quality of service.

With the discovery of Turbo codes, channel coding close to the Shannon limit has become possible with reasonable computational complexity. In the past years, the Turbo principle of exchanging extrinsic information between separate channel decoders has also been extended to other receiver components. In a Turbo-like process the residual redundancy of source codec parameters such as scale factors or predictor coefficients for speech, audio, and video signals can be exploited by *iterative source-channel decoding* (ISCD) [1], [2]. This residual redundancy occurs due to imperfect source encoding resulting for instance from delay and complexity constraints. It can be utilized by a *soft decision source decoder* (SDSD) [3] which exchanges extrinsic reliabilities with a channel decoder.

A joint source-channel coding approach with iterative decoding has been selected for the *FlexCode* concept. However, several major modifications were needed to adapt the channel coder to the specific source coder. These modifications will be explained and highlighted within this paper. It will be shown how the system with iterative-source channel decoding can be applied for all *FlexCode* scenarios and how a generic, very flexible source-channel coding approach is obtained.

This work was financed by the European Union under Grant FP6-2002-IST-C 020023-2 *FlexCode*

II. The *FlexCode* Source-Channel Coding Approach

The design goal of the *FlexCode* source and channel coding platform was to realize a system that is flexible in rate and can instantly adapt to varying channel and network conditions. Therefore, a flexible source encoder has been designed without utilizing fixed code books. In contrast to, e.g., the AMR-WB codec [4], which uses nine different operating modes to select between nine possibly available rates, the *FlexCode* system offers ultimate flexibility: any rate larger than a minimum rate can be selected on a quasi continuous scale. The rate adaptation in the *FlexCode* system works as follows: a certain gross rate is given by the network; using the knowledge of the channel quality, the *FlexCode* rate adaptation mechanism automatically allocates the source and channel coding rate such that possible transmission errors are minimized while maintaining a maximum speech quality.

The baseline *FlexCode* source coding concept is described in [5], [6], and [7]. A simplified block diagram is given in Fig. 1. In fact, the *FlexCode* source encoder is a transform-based speech and audio coder. For each frame, the source encoder provides a set of parameters that can be grouped into two main parts: model parameters and transform coefficients. The model parameters include the LP coefficients describing the spectral envelope of the signal, gain factors, and parameters describing the pitch. From the model parameters the *Karhunen-Loève Transform* (KLT) is derived, which delivers the transform coefficients. The model parameters are quantized with a fixed rate. It has been shown in [8] that for the model

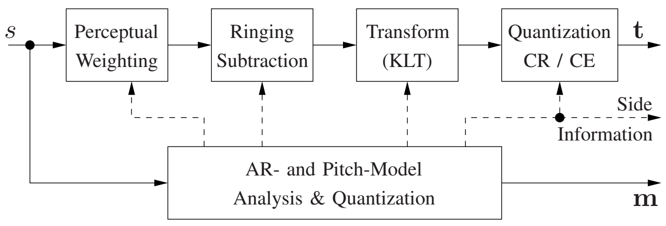

Fig. 1. The *FlexCode* source coding approach, simplified block diagram
CR: constrained resolution (fixed rate)
CE: constrained entropy (variable rate, fixed mean rate)

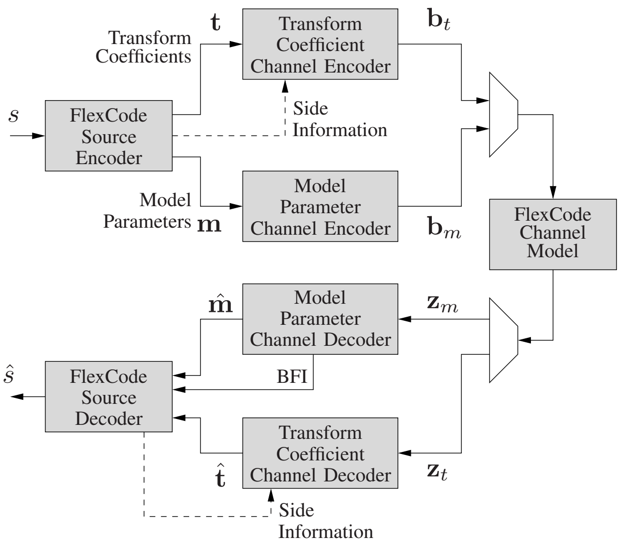

Fig. 2. The basic *FlexCode* source-channel coding approach

parameters, a minimum constant bit rate, which is independent of the total overall rate, is required and that the remainder of the bit rate shall be distributed to the transform coefficients. As an alternative to the KLT, an MDCT can be employed to realize a system with reduced complexity (but with somewhat lower overall performance).

Using the model parameters, the source encoder determines the quantizer setup for the transform coefficients which are quantized using either constrained entropy or constrained resolution scalar quantization [9], [10]. In the case of constrained resolution (CR) quantization, the source encoder determines the bit allocation of the transform coefficients, i.e., the number of quantization levels to be used for the considered parameter resulting in a fixed bit rate. In the case of constrained entropy (CE) quantization, the source encoder implicitely uses the model parameters to determine the distribution of the transform coefficients and the step size of the uniform quantizer. Using this information, an entropy coder (for example an arithmetic coder [11]) can efficiently generate a compressed bit stream of variable but rate, with a fixed mean rate however.

The channel coding concept had to be adapted to the basic structure of the source encoder. It has been found out that it is difficult to employ the envisaged *iterative source-channel decoder* for the transform coefficients unless the model parameters are available at the receiver. The source-channel decoder requires knowledge about the model in order to determine the encoding parameters of the transform coefficients like bit allocation and step sizes. Therefore, we decided for a separate channel coding of the model parameters and the transform coefficients as depicted in Fig. 2 [12].

The *FlexCode* source encoder outputs model parameters $\mathbf{m}$ and transform coefficients $\mathbf{t}$ which are applied to independent channel encoders in two branches. In order to encode the transform coefficients, side information (which depends on the utilized quantization mode) resulting from the model parameters is required. After separate encoding of both model parameters and transform coefficients, the resulting bit streams $\mathbf{b}_m$ (model) and $\mathbf{b}_t$ (transform coefficients) are multiplexed and transmitted over the *FlexCode* channel model [13]. The main parameters for the *FlexCode* source coding setup are summarized in Table I for both possible transforms. In the remainder of this paper, we only consider the *FlexCode* KLT source coder.

TABLE I
DETAILS OF THE *FlexCode* SOURCE CODING SETUPS

	KLT	MDCT
Sampling rate	12.8 kHz	16 kHz
Frame length	20 ms (256 samples)	16 ms (256 samples)
Subframes	4	2
LP model order	16	16
Transform size	64	128
Bit rate for model	5.6 kbps (constant)	≈ 3.6 kbps (constant)
Source coding	10 kbps – 64 kbps	
Channel coding	0 kbps (no coding) – 256 kbps	

At the receiver, this bit stream is demultiplexed and the model parameters are channel decoded using the received values $\mathbf{z}_m$. Additionally a *Bad Frame Indicator* (BFI) is computed by means of error detection. Such a BFI can be used for instance at the source decoder to perform *frame erasure concealment*. Using the model parameters the source decoder computes the side information required by the transform coefficient channel decoder. Without this side information it is not possible to reconstruct the transform coefficients from the received values $\mathbf{z}_t$. Therefore the BFI is computed from the model parameters as an erroneously decoded model would result in a complete decoding failure of the remainder of the packet. On the other hand, if the transform coefficients cannot be recovered, a signal with acceptable quality can be recovered as the spectral envelope and the gains are available in the model and can be used to approximate the signal. With fully recovered side information, the transform coefficients can be reconstructed and the source decoder can compute the signal $\hat{s}$.

In the following two Sections, both channel coders will be described in detail. First, the model channel encoder is explained in Sec. III while the transform coefficient channel coder is discussed in Section IV.

III. Model Parameter Channel Coding

The model parameters are grouped and quantized followed by mapping the indices to bit patterns. The grouped model bit stream is encoded using a strong channel code. The bit rate for transmitting the model parameters is rather small and more or less fixed (around $5-6$ kbit/s, see [8]). For details on the model parameters we refer the reader to [7] and [14]. The model parameter output of the source encoder consists of two types of information: the vector of quantizer indices $\mathbf{m}$ as well as the number of quantizer reproduction levels used by the bit mapper to generate an uncoded bit stream.

This bit stream is then encoded by the model channel code. Depending on the channel quality, the model channel code is one out of four possible LDPC codes [15], [16], [17]. Although several possibilities have been studied throughout the project for encoding the model bit stream, we have selected LDPC codes for two main reasons. The complexity of the LDPC decoder, based on belief propagation [16], scales with the channel quality. In good channel conditions, a small number of iterations is sufficient to successfully decode the block while the number of necessary iterations increases if the channel quality becomes worse. The LDPC code offers furthermore built in error detection capabilities. If at least one parity check equation is not satisfied after decoding, the *Bad Frame Indicator* (BFI) is set. In this case, the source decoder can take appropriate measures for frame erasure concealment.

Four different LDPC codes of different rates are included in the final *FlexCode* channel code. During the operation, the coder can select either of these three, depending on the assumed transmission quality. The three codes are all regular LDPC codes with column weight 3 constructed using MacKay's algorithm [17]:

- LDPC code $\mathcal{C}_1(170, 120)$, generating 50 parity bits
- LDPC code $\mathcal{C}_2(220, 120)$, generating 100 parity bits
- LDPC code $\mathcal{C}_3(420, 120)$, generating 300 parity bits
- LDPC code $\mathcal{C}_4(720, 120)$, generating 600 parity bits

As the number of input bits $N_B^{[\text{Model}]}$ may vary from one codec configuration to the other, the code has to be shortened. Therefore, the input bit stream of length $N_B^{[\text{Model}]}$ is padded with zeros and encoded. After encoding, the padded zeros are removed again. At the receiver, the knowledge of that part of the input bit stream that are zeros can be used as *a priori* information in the decoder. Therefore, the rates of the four LDPC codes are $\frac{N_B^{[\text{Model}]}}{N_B^{[\text{Model}]}+50}$, $\frac{N_B^{[\text{Model}]}}{N_B^{[\text{Model}]}+100}$, $\frac{N_B^{[\text{Model}]}}{N_B^{[\text{Model}]}+300}$, or $\frac{N_B^{[\text{Model}]}}{N_B^{[\text{Model}]}+600}$, respectively. Fine-tuning of the rate by puncturing the parity bits of the LDPC code may be necessary.

The decoder of the model parameter transmission chain performs first belief propagation decoding of the LDPC code and then reconstructs the quantization indices of the different model parameters from the decoded bit stream. Figure 3 shows the bit error rate (BER) results of the utilized codes after transmission over an AWGN channel with $E_\text{s}/N_0 = \frac{1}{2\sigma_n^2}$. Note that we use E_s/N_0 instead of E_b/N_0 as the overall total gross bit rate (model parameters and transform coefficients) does not change. The different codes are not shortened in this example, i.e., $N_B^{[\text{Model}]} = 120$, and 60 belief propagation iterations are employed at the receiver. From Fig. 3, the channel qualities E_s/N_0 at which the codes have to switched can easily be read off, given a specific target BER (for instance 10^{-5}).

In very good channel conditions, no channel coding is necessary, however, error detection is mandatory as a rapid decrease of the transmission quality might affect the model parameters. As errors in the model can cause very annoying audible distortions, concealment based on a BFI is necessary and therefore, the uncoded bit stream is protected by a CRC code [18] if LDPC coding is switched off.

IV. Transform Coefficients Channel Coding

The KLT (or MDCT) transform coefficients on the other hand are encoded using an iterative source-channel coding system. For implementational details of this joint source-channel coding approach with iterative decoding, we refer the reader to the literature, e.g., [19]. As the approach depends on

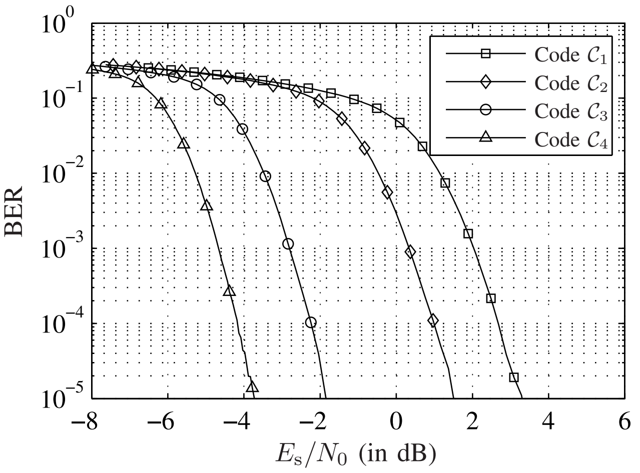

Fig. 3. BER results of the utilized model codes on an AWGN channel.

the type of quantization (constrained resolution or constrained entropy) two cases and two different coding schemes have to be considered. The compatibility and interoperability for both quantization modes was a required constraint.

The basic concept is identical for both quantization modes: a bit stream is generated and an outer encoder (a block or convolutional encoder) adds a certain amount of artificial redundancy (depending on the overall coding rate) to the bit stream. This bit stream is interleaved using the interleaver presented in Sec. IV-A and then encoded by an inner convolutional encoder. At the receiver, an inner MAP channel decoder and a *Soft Decision Source Decoder* (SDSD) (which exploits the redundacy added by the outer encoder and may also exploit the residual redundancy of the transform coefficients, if available) iteratively exchange extrinsic information according to [2], [1]. After a certain number of iterations have been carried out, the transform coefficients are estimated using the MAP rule.

The side information (see Fig. 2) about the bit allocation (constrained resolution quantization) or the quantizer step sizes and parameter distribution (constrained entropy quantization) is derived by the *FlexCode* source decoder from the model parameters which are decoded first. This information is required by the SDSD for generating the extrinsic information in the ISCD process.

A. Flexible Interleavers

The *FlexCode* channel decoder uses the Turbo principle of exchanging extrinsic information between two component codes. An essential element of a transmission or storage system employing the Turbo principle is the interleaver. As the frame size of the data to be channel coded might be subject to frequent changes, interleavers which can change their size on the fly with moderate computational complexity are needed. Such interleavers are called *prunable* interleavers [20]. The *FlexCode* interleaver is based on [21] which extends an S-random interleaver by adding additional entries such that the S-condition remains fulfilled. The pruning can be easily performed on the fly during (de-)interleaving. For implementational details, we refer the reader to [22] and [14].

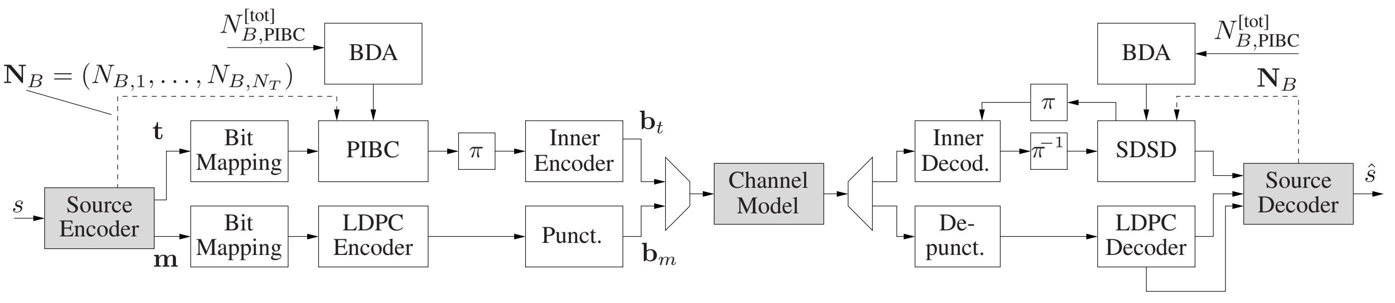

Fig. 4. Transmitter and receiver of the *FlexCode* channel coding platform for the case of Constrained Resolution (CR) quantization

B. Constrained Resolution Quantization

In the case of CR quantization, the source encoder determines the number of quantization levels L_ℓ (and thus also the number of required bits $N_{B,\ell}$) for each transform coefficient t_ℓ of the frame $\mathbf{t} = (t_1, \ldots, N_T)$, with N_T being the number of transform coefficients per frame. The natural binary representation of the transform coefficient quantization index t_ℓ (which has been quantized using $L_\ell = 2^{N_{B,\ell}}$ levels) gives the bit pattern $\mathbf{x}_\ell^{\mathbf{t}} = (x_{\ell,1}^{\mathbf{t}}, \ldots, x_{\ell,N_{B,\ell}}^{\mathbf{t}})$. This operation is performed in the block *Bit mapping* in the block diagram of the constrained resolution (CR) channel coder given in Fig. 4. If, due to adverse channel conditions, additional channel coding redundancy is required, one or several parity check bits are added to each transform coefficient. This is performed by the outer channel code, i.e. the block *Parameter Individual Block Code* (PIBC) in Fig. 4. PIBCs have been employed successfully in the context of error-resilient speech transmission in, e.g., [23]. The *Bit Distribution Algorithm* (BDA) determines the selection of the PIBCs: knowing the source encoder bit rate and the gross bit rate on the channel, the rate control mechanism can compute the number $N_{B,\text{PIBC}}^{[\text{tot}]}$ of parity bits that it is allowed to add to each frame. Furthermore, let $N_B^{[\text{tot}]} = \sum_{\ell=1}^{N_T} N_{B,\ell}$ be the number of bits utilized by the source encoder for the transform coefficients (TC). The task of the BDA is to distribute the $N_{B,\text{PIBC}}^{[\text{tot}]}$ parity bits to the different transform coefficients such that each transform coefficients uses an amount of $N_{B,\text{PIBC},\ell}$ parity bits. The BDA operates as follows: first, $\left\lfloor \frac{N_{B,\text{PIBC}}^{[\text{tot}]}}{N_T} \right\rfloor$ bits are distributed to each transform coefficient and then one additional bit is distributed to the first $N_{B,\text{PIBC}}^{[\text{tot}]} - N_T \cdot \left\lfloor \frac{N_{B,\text{PIBC}}^{[\text{tot}]}}{N_T} \right\rfloor$ transform coefficients. This algorithm has been selected after a simulative comparison between several different BDA algorithms.

The overall performance of the system can basically be controlled by a careful selection of the PIBCs. As each transform coefficient can be quantized using a different number of bits due to the flexibility of the source encoder and as the number of additional parity bits depends on the gross bit rate available on the channel (the number of parity bits per parameter is determined by the BDA), the codes that are utilized should have a certain structure and the parity bits should be determinable by simple operations. Due to these constraints, it is not feasible to store a large number of optimized codes but it is useful to store only a certain structure of the code. Following [24] the utilized PIBCs are built using the following design rule: Let $\mathbf{y}_\ell^{\mathbf{t}} = (y_{\ell,1}^{\mathbf{t}}, \ldots, y_{\ell,N_{B,\ell}+N_{B,\text{PIBC},\ell}}^{\mathbf{t}})$ denote the bit pattern of the ℓ'th transform coefficient after encoding. The code is systematic, i.e., $y_{\ell,i}^{\mathbf{t}} = x_{\ell,i}^{\mathbf{t}}$, $\forall i \in \{1, \ldots, N_{B,\ell}\}$. The first parity bit is obtained by performing a parity check over all bits of $\mathbf{x}_\ell^{\mathbf{t}}$, i.e.,

$$y_{\ell,N_{B,\ell}+1}^{\mathbf{t}} = x_{\ell,1}^{\mathbf{t}} \oplus \cdots \oplus x_{\ell,N_{B,\ell}}^{\mathbf{t}} = \sum_{j=1}^{N_{B,\ell}}\!\!\oplus\; x_{\ell,j}^{\mathbf{t}} \,. \tag{1}$$

The following parity check bits then exclude one of the systematic bits, i.e., they are obtained by

$$y_{\ell,N_{B,\ell}+1+i}^{\mathbf{t}} = \sum_{\substack{j=1 \\ j \neq 1+((i-1) \bmod N_{B,\ell})}}^{N_{B,\ell}}\!\!\oplus\; x_{\ell,i}^{\mathbf{t}} \,. \tag{2}$$

For the example of $N_{B,\ell} = 4$ and $N_{B,\text{PIBC},\ell} = 7$ the generator matrix looks like follows:

$$\mathbf{G}_{\text{OPT},\ell}^{[4,7]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} . \tag{3}$$

Due to the special structure of this code the parity bits can easily be generated using simple operations and the generator matrices do not need to be explicitly stored.

After PIBC, the resulting bits of a complete frame are interleaved and encoded by an inner channel encoder, which is a recursive systematic rate $1/2$ convolutional encoder with octal generator polynomials $\left(1, \frac{10}{17}\right)$. By puncturing the systematic bits, this code can be rendered into a rate-1 code which has been proven to give good results for iterative source-channel decoding [19]. The coding redundancy is only added by the outer code component, i.e., the PIBC. After inner channel coding, the bit stream of the transform coefficients is concatenated with the bit stream of the model parameters and the frame is transmitted over the *FlexCode* channel model [13].

At the receiver, first the model parameters are decoded and fed together with the BFI to the source decoder. Second, the transform coefficients are decoded using iterative source-channel decoding (ISCD). The SDSD gets $N_{B,\text{PIBC},\ell}$ from the BDA and can decode the PIBC using the SDSD technique. The SDSD can exploit all available statistical side information on the transform coefficients such as an unequal distribution or correlation as well as the artificial redundancy added by the PIBC [19]. For detailed simulation results, a comparison of different code families for PIBC, and a comparison of different BDAs, we refer the reader to [14].

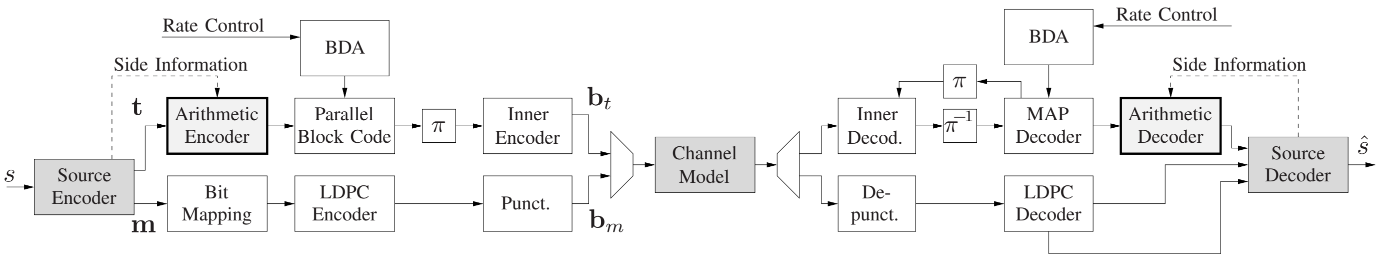

Fig. 5. Transmitter and receiver of the *FlexCode* channel coding platform for the case of Constrained Entropy (CE) quantization and arithmetic coding

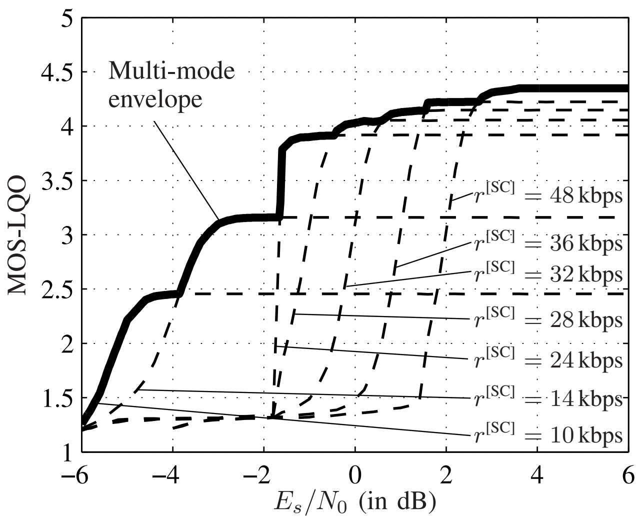

Fig. 6. Performance of the *FlexCode* system with constrained resolution quantization (CR), gross bit rate of $r^{[\text{tot}]} = 64$ kbps, different source coding rates, 10 ISCD iterations

In order to evaluate the overall performance of the *FlexCode* source-channel coding scheme, we employ the PESQ speech quality measurement tool [25]. Figure 6 depicts the MOS-LQO (*Mean Opinion Score - Listening Quality Objective*) values measured by PESQ for a *FlexCode* application example, recorded with a development version of the source coder. The gross transmission rate is fixed to $r^{[\text{tot}]} = 64$ kbps and the utilized channel is an AWGN channel. Different source coding bit rates $r^{[\text{SC}]}$ have been chosen; the remainder of the bit rate $r^{[\text{CC}]} = r^{[\text{tot}]} - r^{[\text{SC}]}$ is used for channel coding. The model is protected as described in Sec. III. The LDPC switching points are selected using Fig. 3 and a target BER of 10^{-5}. However, the low rate LDPC codes are only used if $r^{[\text{CC}]}$ is not exceeded by the LDPC code only. The ISCD receiver utilizes 10 ISCD informations and exploits knowledge on the distribution of the transform coefficients. For very bad channels with $E_s/N_0 < -1.7$ dB, the fourth LDPC code $\mathcal{C}_4(720, 120)$ is used. In that case however, the remaining bit budget for protecting the transform coefficients is not sufficient, resulting in the steep degradation in Fig. 6. It can be seen in Fig. 6 that, if for a certain channel quality the best choice of the considered source and channel coding rate compromise is selected, we obtain a multi-mode envelope (thick solid line). This scheme gives superior performance in all cases and is a guideline on the setup of the source and channel coding rates given the channel quality. It features a high listening quality for good channels as well as error robustness for bad channel conditions.

C. Constrained Entropy Quantization

In the case of CE quantization, the source encoder utilizes a uniform scalar quantizer thus leaving a certain amount of residual redundancy in the quantization indices. This redundancy is removed by an entropy coder generating a variable-length bit stream using the statistical information on the transform coefficients. We use an arithmetic coder, which is known to compress the source close to the entropy. The *FlexCode* system with ISCD however shall still be used as a universal channel coding system. The resulting system is depicted in Fig. 5.

The transform coefficients $\mathbf{t}$ are encoded by an arithmetic coder [11] to a (variable length) bit stream using the side information on the pdf of the different coefficients provided by the source encoder. The arithmetic encoder can compress the transform coefficients close to the entropy with low computational complexity. For channel coding, the resulting bit stream is partitioned into several groups of bits. Each of these groups is then considered to be a parameter. Each of these groups is then encoded by an individual block code, as presented above. Thus, the same structure as used for CR quantization can be applied. The BDA determines a selection of block codes such that the desired channel coding rate is achieved while maintaining decodability. For instance, the assignment of block codes to the groups of bits can be optimized using the concept of irregular codes and index assignments [14], [26], [27]. After interleaving the *inner channel encoder* performs a second channel coding step. This inner channel code is a recursive convolutional code, which is usually punctured to rate-1 if iterative decoding is employed.

At the receiver, the outer channel decoder consists then of an SDSD which does not exploit any statistical properties as the bits after arithmetic coding are assumed to be equiprobable. The SDSD reduces in this case to a MAP decoding of the single block codes. As the single block codes are assumed to be small codes, the computational complexity of their MAP decoding is also reasonably low. After performing the iterative decoding, the bit stream is reconstructed, arithmetically decoded using the side information on the pdfs. The decoded transform coefficients are fed to the source decoder. Note that the transmission of the model parameters is performed as described in Section III.

It shall further be noted that by a proper selection of the inner and outer codes, the channel coding part can be completely disabled and the arithmetically coded bit stream is directly fed to the channel. This is especially important if no

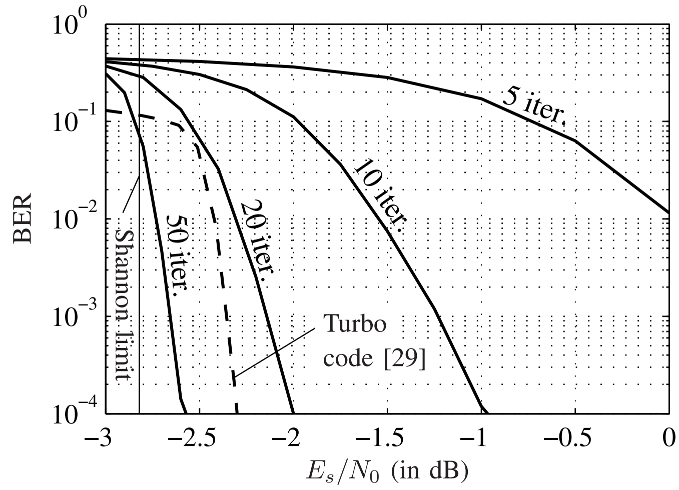

Fig. 7. Bit error rate performance of the generic *FlexCode* transform coefficient channel coder, CE (constrained entropy) case, for blocks of 20000 data bits, overall coding rate $1/2$

bit errors are expected on the transmission link.

In a simulation example, the generic *FlexCode* channel coder of Fig. 5 is utilized for the transmission of an arithmetically coded bit stream. For demonstrating the concept, we use for this simulation a Bernoulli source with blocks of 20000 equiprobable data bits. The utilized channel code is the rate-1 code taken from [28]. The overall coding rate has been selected to $1/2$. The size of the utilized codes are $(n_i, k_i) \in \{(10, 1); (6, 2); (5, 2); (4, 2); (3, 2)\}$. The optimization according to [26] gives a guideline how to assign the codes to the different bit groups. The simulation results in terms of bit error rate are depicted in Fig. 7. Within 50 iterations, the channel code is able to closely reach the Shannon bound and outperforms the rate-$1/2$ Turbo code of [29] by $\approx 0.2\,\text{dB}$ in terms of E_s/N_0. The gap to the theoretical limit is mainly due to the finite block size of 20000 bit. Note that block sizes of 20000 bit are usually not given in speech and audio transmission schemes, however, they can easily occur in video transmission scenarios, which are also targeted by *FlexCode* [30]. Further information and results for the constrained entropy case can be found in [14].

V. Conclusion

In this paper we have presented the final channel coding approach utilized in the *FlexCode* project. The proposed channel code has been developed in close cooperation with the source codec and the technique of iterative source-channel decoding has been applied. We have proposed a flexible channel codec which can instantaneously adapt to different channel conditions and change the coding rate on the fly. Furthermore, the channel coder can adapt to both quantization modes of the source encoder. Implementational details of the developed approach are given and the performance of the system in one operating mode has been shown by a simulative example.

Acknowledgments

The authors would like to thank Janusz Klejsa and W. Bastiaan Kleijn from KTH Stockholm for many fruitful discussions.

References

[1] M. Adrat, P. Vary, and J. Spittka, “Iterative Source-Channel Decoder Using Extrinsic Information from Softbit-Source Decoding,” in *IEEE ICASSP*, Salt Lake City, USA, May 2001.

[2] N. Görtz, “On the Iterative Approximation of Optimal Joint Source-Channel Decoding,” *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, pp. 1662–1670, Sept. 2001.

[3] T. Fingscheidt and P. Vary, “Softbit Speech Decoding: A New Approach to Error Concealment,” *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 240–251, Mar. 2001.

[4] 3GPP TS 26.190, “AMR Wideband Speech Codec; Transcoding Functions,” 2001, available at http://www.3gpp.org.

[5] S. Bruhn, V. Grancharov, W. B. Kleijn, J. Klejsa, M. Li, J. Plasberg, H. Pobloth, S. Ragot, and A. Vasilache, “The FlexCode Speech and Audio Coding Approach,” in *ITG Conference on Speech Communication*, Aachen, Germany, Oct. 2008.

[6] FlexCode, “Deliverable D-1.1: Baseline Source Coder,” European Union, Tech. Rep., Feb. 2008, available at http://www.flexcode.eu.

[7] FlexCode, “Deliverable D-1.2: Final Source Coder,” European Union, Tech. Rep., Oct. 2008, available at http://www.flexcode.eu.

[8] W. B. Kleijn and A. Ozerov, “Rate Distribution between Model and Signal,” in *Proc. of IEEE WASPAA*, New Paltz, NY, USA, Nov. 2007.

[9] A. Gersho, “Asymptotically Optimal Block Quantization,” *IEEE Trans. Inform. Theory*, vol. IT-25, no. 4, pp. 373–380, July 1979.

[10] R. M. Gray and D. L. Neuhoff, “Quantization,” *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2352–2383, Oct. 1998.

[11] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Prentice Hall, Inc., 1990.

[12] L. Schmalen, B. Schotsch, M. Jeub, P. Vary, and T. Clevorn, “The FlexCode Channel Coding Approach,” in *ITG Conference on Speech Communication*, Aachen, Germany, Oct. 2008.

[13] FlexCode, “Deliverable D-2.1: BISO Channel Model,” European Union, Tech. Rep., July 2007, available at http://www.flexcode.eu.

[14] FlexCode, “Deliverable D-2.3: Final Channel Coder,” European Union, Tech. Rep., Oct. 2008, available at http://www.flexcode.eu.

[15] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: M.I.T. Press, 1963.

[16] D. J. C. MacKay and R. M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes,” *Electron. Lett.*, Aug. 1996.

[17] D. J. C. MacKay, “Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Trans. Inform. Theory*, pp. 399–431, Mar. 1999.

[18] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson Higher Education, 2003.

[19] M. Adrat, T. Clevorn, and L. Schmalen, “Iterative Source-Channel Decoding & Turbo-DeCodulation,” in *Advances in Digital Speech Transmission*. John Wiley & Sons, Ltd., Jan. 2008, ch. 13.

[20] A. Tarable, L. Dinoi, and S. Benedetto, “Design of Prunable Interleavers for Parallel Turbo Decoder Architectures,” *IEEE Commun. Lett.*, vol. 11, no. 2, Feb. 2007.

[21] M. Ferrari, F. Scalise, and S. Bellini, “Prunable S-Random Interleavers,” in *Proc. of International Conference on Communications (ICC)*, New York City, NY, USA, Apr. 2002.

[22] FlexCode, “Deliverable D-2.2: Baseline Channel Coder,” European Union, Tech. Rep., Feb. 2008, available at http://www.flexcode.eu.

[23] T. Fingscheidt, S. Heinen, and P. Vary, “Joint Speech Codec Parameter and Channel Decoding of Parameter Individual Block Codes (PIBC),” in *Proc. of IEEE SCW*, Porvoo, Finland, June 1999.

[24] T. Breddermann, L. Schmalen, and P. Vary, “AMR-NB Speech Transmission Quality Beyond UMTS by Iterative Source-Channel Decoding,” in *submitted to VTC-Fall 2010*, Taipei, Taiwan, 2010, unpublished.

[25] *ITU-T P.862 Perceptual Evaluation Of Speech Quality (PESQ): An Objective Method for End-To-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, ITU-T, February 2001.

[26] M. Tüchler and J. Hagenauer, “EXIT Charts of Irregular Codes,” in *Proc. of CISS*, Princeton University, Mar. 2002.

[27] L. Schmalen, P. Vary, T. Clevorn, and B. Schotsch, “Efficient Iterative Source-Channel Decoding Using Irregular Index Assignments,” in *Proc. of ITG Conf. on Source and Channel Coding*, Ulm, Germany, 2008.

[28] S. ten Brink, “A Rate One-Half Code for Approaching the Shannon Limit by 0.1dB,” *IEE Electron. Lett.*, vol. 36, no. 15, July 2000.

[29] C. Berrou and A. Glavieux, “Near Optimum Error Correcting Coding and Decoding: Turbo-Codes,” *IEEE Trans. Commun.*, Oct. 1996.

[30] FlexCode, “Deliverable D-1.3: Video Coder Pilot Study,” European Union, Tech. Rep., Mar. 2009, available at http://www.flexcode.eu.