Decoder Scheduling of Hybrid Turbo Codes

Neele von Deetzen and Werner Henkel
International University Bremen (IUB)
Campus Ring 1
D-28759 Bremen, Germany
Email: {n.vondeetzen, w.henkel}@iu-bremen.de

***Abstract*— This paper describes the Turbo decoding of hybrid concatenated codes with interleavers and presents a new analysis of the decoding process including information processing and convergence. From this analysis, the decoding can be scheduled with respect to optimization issues like computational complexity and convergence behavior.**

I. Introduction

Turbo codes are powerful codes presented 1993 in [1]. They consist of two or more parallel concatenated convolutional codes separated by interleavers. These well-known codes can approach the Shannon limit very closely with moderate decoding complexity. The decoding of concatenated codes is done by activating decoders of the component codes iteratively and exchanging extrinsic information. An alternative to the parallel concatenation is to concatenate convolutional codes in series. These codes have a much lower error floor than parallel concatenated codes but have worse performance for smaller SNR. Decoding works similar to parallel concatenations but with slight differences in information processing between the two decoders. A further form of concatenation, which we call hybrid, is a mixture of a parallel and a serial concatenation, i.e., parallel concatenated serial concatenations. The decoding of hybrid concatenated codes is more complicated than of parallel or serial concatenations and is a combination of both. However, it is not clear in which order the decoders of the component codes have to be activated to achieve maximum possible mutual information or minimum decoding complexity. These hybrid concatenated codes with interleavers and Turbo-decoding will be called hybrid Turbo codes for the rest of the paper.

This paper is organized as follows. The system model of the encoding and decoding of hybrid concatenated codes is given in Section II. The correct arrangement of all single decoders will be shown. Section III presents a detailed description of the information extraction and processing between all component decoders. In Section IV, we describe the optimization issues. Furthermore, this section contains a description of the scheduling optimization obtained by multiple EXIT charts. Finally, conclusions are summarized in Section V.

II. System Model

A hybrid concatenation is defined by a combination of a parallel and a serial concatenation, i.e., a parallel concatenation of serial concatenated codes. Figure 1 presents a simple example with two parallel branches with two serial concatenated component codes each. The parallel branches as well as the serial concatenated component codes are each separated by interleavers of appropriate size. In this example, the outer codes are of rate $R_{11} = R_{21} = 1/2$ and the inner code rates are $R_{12} = R_{22} = 2/3$.

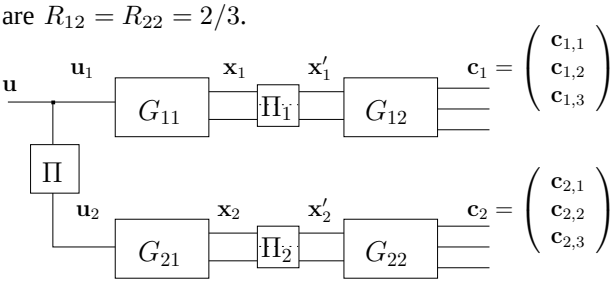

Fig. 1. Encoder structure of a hybrid concatenated code with interleavers

In the following, all component codes are assumed to be recursive systematic convolutional (RSC) codes. We define the interleavers in the upper and lower branch, Π_1 and Π_2, not to mix their upper and lower input bit streams, such that the whole Hybrid Turbo code is systematic and the information bits only have to be transmitted once. The coded bit stream is composed as follows

$$\begin{aligned}\mathbf{c} \;=\; &\big(c_{1,1}(1)\; c_{1,2}(1)\; c_{1,3}(1)\; c_{2,2}(1)\; c_{2,3}(1)\\ &\quad c_{1,1}(2)\; c_{1,2}(2)\; c_{1,3}(2)\; c_{2,2}(2)\; c_{2,3}(2)\;\; \ldots\big)\,,\end{aligned}$$

where $c_{1,1}(1) = u(1)$, $c_{1,1}(2) = u(2)$ and so on. The decoding structure of such hybrid Turbo codes is complicated and represents a combination of the decoding structures of parallel and serial concatenations. Thus, we will first present the decoding structures of these simple concatenations and later explain how to combine them.

For the parallel concatenation with two component encoders and an interleaver Π_p in between, we define the information sequence of the whole encoder to be $\mathbf{u}$ and the information sequences of the component encoders as $\mathbf{u}_1 = \mathbf{u}$ and $\mathbf{u}_2 = \mathbf{u}(\Pi_p)$. The respective coded sequences are called $\mathbf{c}_1$ and $\mathbf{c_2}$. As component decoders are used APP decoders (A-Posteriori Probability, e.g. BCJR, Log-MAP) which have two inputs and two outputs in form of Log-Likelihood ratios or L-values. In Figures 2 and 3, these decoders are represented by boxes with two inputs at the left and two outputs at the right. The decoders of the parallel concatenation receive information from the channel, called intrinsic information. This intrinsic information can be interpreted as a-priori information concerning the coded

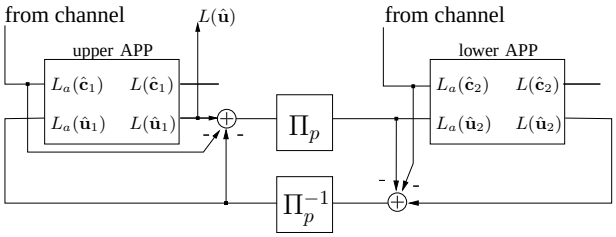

Fig. 2. Decoder structure of a parallel concatenated code with an interleaver

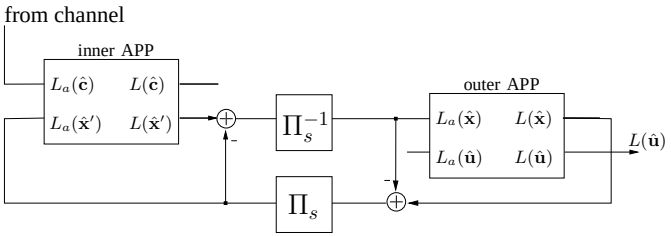

Fig. 3. Decoder structure of a serial concatenated code with an interleaver

bit stream and is, thus, called $L_a(\hat{\mathbf{c}}_i)$ (upper input). The second (lower) input represents the a-priori information concerning the uncoded bit stream, denoted by $L_a(\hat{\mathbf{u}}_i)$. Accordingly, the decoder outputs two Log-Likelihood ratios corresponding to the coded (upper output) and the uncoded (lower output) bit streams denoted by $L(\hat{\mathbf{c}}_i)$ and $L(\hat{\mathbf{u}}_i)$, respectively. For systematic encoders, the output values are composed of the two a-priori values and some extrinsic information, which has been gained by the decoding process. In order to avoid statistical dependencies, the two decoders only exchange the extrinsic L-values corresponding to the uncoded bit stream $L_e(\hat{\mathbf{u}}_i)$. Thus, the a-priori values are subtracted from the estimated values before passing them as a-priori information to the next decoder. Figure 2 shows the decoding structure of such a parallel concatenation.

For a serial concatenation with interleaver Π_s, let $\mathbf{u}$ and $\mathbf{x}$ be the input and output of the outer encoder, and $\mathbf{x}'$ and $\mathbf{c}$ be the input and output of the inner encoder, respectively, similar to the notations of the upper or lower branch in Fig. 1. The decoder structure is similar to that of the parallel concatenation, except for two differences. The first difference is the input and output of the outer decoder. The second, i.e., outer decoder receives no intrinsic information directly from the channel but from the estimated values of the inner decoder. The a-priori information concerning the uncoded input of the outer decoder is zero all the time, since there is no information from this side of the decoder. Furthermore, the outer decoder does not pass information corresponding to the uncoded but to the coded bits to the inner decoder, since the (interleaved) coded output of the outer encoder corresponds to the uncoded input of the inner encoder. The second difference is that the inner and the outer decoder only subtract the a-priori knowledge about the uncoded and the coded data, respectively. All in all, the inner decoder estimates the uncoded data and the outer one estimates its coded data. Both decoders only subtract from their estimated values what they received from the other decoder.

Like hybrid concatenated encoders are a mixture of a parallel and a serial concatenation, also the decoder of a hybrid structure is a combination of both decoders. In this case, the single APP decoders in Fig. 2 denoted by 'upper APP' and 'lower APP' will each contain a whole serial decoding structure like in Fig. 3. Thereby, the uncoded a-priori input of the parallel decoder, $L_a(\hat{\mathbf{u}}_i)$, will be connected to the uncoded input of the outer decoder of Fig. 3, then denoted by $L_a(\hat{\mathbf{u}}_i)$. The corresponding coded input in Fig. 2 will be connected to the intrinsic input of Fig. 3. The two outputs of the respective APP decoders in Fig. 2 are connected to the coded output of the inner component decoder and to the uncoded output of the outer component decoder of Fig. 3. These connections are depicted in Fig. 4, which presents the serial decoding inside the upper APP decoder of a parallel concatenation.

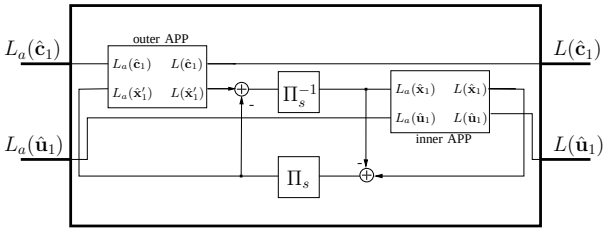

Fig. 4. Decoder structure of a serial concatenation inside an APP decoder

III. Information Transfer of Decoding

In the last section, we described how the component decoders are connected to each other. In this section, we explain the information processing during the decoding process. The decoding can most clearly be imagined using the encoding structure, see Fig. 5. The channel provides information about the outputs of the two inner encoders, called $L_a(\hat{\mathbf{c}}_1)$ and $L_a(\hat{\mathbf{c}}_2)$. We assume the upper branch to be decoded first. Thus, the upper inner decoder starts by estimating $\mathbf{x}'_1$ and passing the estimated L-values to the upper outer decoder. This decoder subtracts its a-priori L-values $L_a(\hat{\mathbf{x}}_1)$ (provided by the upper inner decoder) from its estimated L-values $L(\hat{\mathbf{x}}_1)$ and passes them again to the upper inner decoder. This proceeding is performed for a certain number of iterations, denoted by $n_{it,1}$. The last iteration is only performed half such that the decoding of the upper branch stops with an activation of the outer decoder. The upper branch now subtracts the initial incoming channel information $L_a(\hat{\mathbf{c}}_1)$ from the estimated input sequence $L(\hat{\mathbf{u}}_1) = L(\hat{\mathbf{u}})$ and passes it to the lower branch.

The decoding of the lower branch starts with the activation of the outer decoder, which receives a-priori information from the upper branch $L_a(\hat{\mathbf{u}}_2)$ and from the channel $L_a(\hat{\mathbf{x}}_2)$ (note that the inner encoder is systematic and the channel information can be passed to the outer decoder without

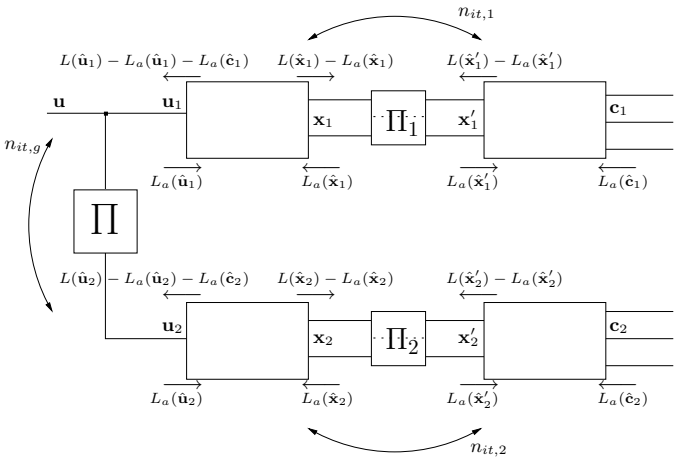

Fig. 5. Decoder structure of a hybrid Turbo decoder

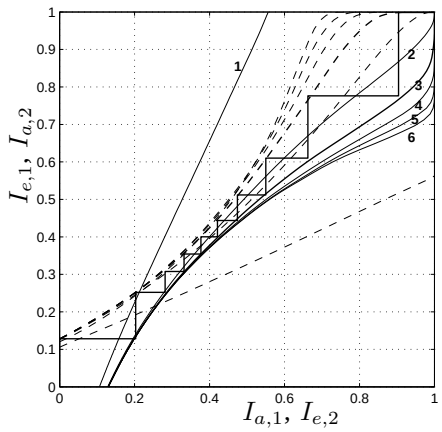

Fig. 6. Multiple EXIT chart for various numbers of local iterations at $E_b/N_0 = 0.17dB$

activation of the inner decoder). The decoder subtracts the a-priori values $L_a(\hat{\mathbf{x}}_2)$ from its estimated values $L(\hat{\mathbf{x}}_2)$ and passes the information to the lower inner decoder. This decoder subtracts the a-priori values $L_a(\hat{\mathbf{x}}'_2)$ from its estimated values $L(\hat{\mathbf{x}}'_2)$. The lower branch is decoded with $n_{it,2}$ iterations ending with an activation of the outer decoder, which subtracts the initial channel information $L_a(\hat{\mathbf{c}}_2)$ and the a-priori values coming from the upper branch $L_a(\hat{\mathbf{u}}_2)$ from its estimation $L(\hat{\mathbf{u}}_2)$. This whole process is executed for $n_{it,g}$ iterations, called global iterations. Each decoding of a branch is performed as described above for the lower branch, since this is the case where the decoding of a branch starts decoding with non-zero a-priori knowledge from the other branch. The iterations within the branches are called local iterations.

For the analysis of the decoding, it is possible to construct EXIT charts [2] corresponding to the local as well as to the global iterations. Although these different EXIT charts interact, the connections cannot easily be seen. Thus, we propose to use only the global EXIT chart. Hereto, we consider each serial decoding structure of a branch as one component decoder in a parallel concatenation. The global EXIT chart contains mutual information concerning the a-priori values $L_a(\hat{\mathbf{u}}_i)$ and the extrinsic values $L_e(\hat{\mathbf{u}}_i) = L(\hat{\mathbf{u}}_i) - L_a(\hat{\mathbf{u}}_i) - L_a(\hat{\mathbf{c}}_i)$. For finitely long sequences of length N, mutual information between some data sequence $\mathbf{u}$ and the corresponding L-values $L_{a/e}(\hat{\mathbf{u}})$ can be calculated by the following two equations.

$$I_a = I(\mathbf{u}; L_a(\hat{\mathbf{u}})) = \mathrm{E}\left\{1 - \log_2\left(1 + e^{-u_l \cdot L_a(\hat{u}_l)}\right)\right\} \quad (1)$$

and

$$I_e = I(\mathbf{u}; L_e(\hat{\mathbf{u}})) = \mathrm{E}\left\{1 - \log_2\left(1 + e^{-u_l \cdot L_e(\hat{u}_l)}\right)\right\} . \quad (2)$$

These expressions are then computed for each parallel branch, where the extrinsic information corresponds to the L-values achieved after the whole number of local iterations $n_{it,1/2}$.

An important new approach to the analysis is not only including the transfer curves of a fixed number of local iterations in the EXIT chart but for a range of numbers of local iterations. This means, the EXIT chart contains one curve per parallel branch for the information transfer corresponding to one local iteration, another curve representing the information transfer after the second local iteration etc.
Figure 6 depicts such a multiple EXIT chart at $E_b/N_0 = 0.17$ dB for example codes given by

$$G_{11} = G_{21} = \begin{pmatrix} 1 & \frac{D^2}{1+D+D^2} \end{pmatrix} \quad (3)$$

and

$$G_{12} = G_{22} = \begin{pmatrix} 1 & 0 & \frac{1+D+D^2}{1+D^2} \\ 0 & 1 & \frac{1}{1+D^2} \end{pmatrix} . \quad (4)$$

The EXIT chart contains the transfer curves for $n_{it,1} = 1, \ldots, 6$ as dashed lines and $n_{it,2} = 1, \ldots, 6$ as solid lines. We additionally inserted the decoding trajectories for a hybrid decoding scheme with $n_{it,1} = n_{it,2} = 3$ and $n_{it,g} = 10$. It is obvious that the edges of the trajectories lie on the transfer curves corresponding to $n_{it,1} = n_{it,2} = 3$, marked as bold lines.

IV. Scheduling Optimization

In the last section, we described the decoding of a hybrid Turbo code for fixed numbers of local and global iterations $n_{it,1}$, $n_{it,2}$, and $n_{it,g}$. From the EXIT chart in Fig. 6, we can see the number of local iterations $n_{it,1}$ and $n_{it,2}$ which are required in order to pass the bottleneck between the two transfer curves. For $n_{it,1} = n_{it,2} = 1$, the mutual information would get stuck at $(I_{a,1}; I_{e,1}) = (I_{a,2}; I_{e,2}) \approx 0.18$ and for $n_{it,1} = n_{it,2} = 2$, the maximum achievable mutual information would be $(I_{a,1}; I_{e,1}) = (I_{a,2}; I_{e,2}) \approx 0.36$. However, when choosing $n_{it,1}$ and $n_{it,2}$ differently, e.g. $n_{it,1} = 3$ and $n_{it,2} = 2$, the bottleneck just opens at 0.17 dB.
Of course, we can also vary the number of local iterations each time, one of the branches is decoded. We could, e.g.,

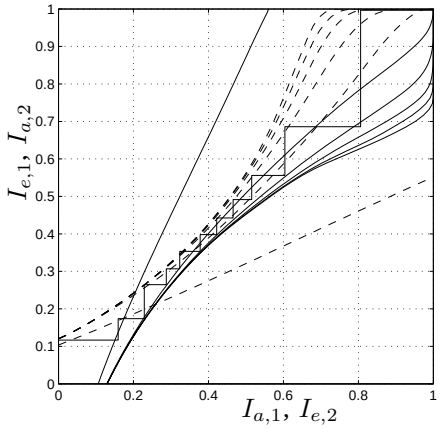

Fig. 7. Multiple EXIT chart for varying numbers of local iterations

decode the upper branch with $n_{it,1} = 3$ iterations, jump to the lower branch and decode it with $n_{it,2} = 4$ iterations, jump back to the upper branch and decode with $n_{it,1} = 2$ iterations etc. Thus, the numbers of local iterations have to be described by vectors of length $n_{it,g}$ where the ith component represents the number of local iterations during global iteration i. The information transfer of such a variable decoding process can also be visualized in a multiple EXIT chart introduced in Fig. 6. The edges of the trajectories now do not touch only one of the curves on each side but jump between several of the transfer curves. The decoding of the hybrid Turbo code example from above is shown in Fig. 7 for the number of global iterations $n_{it,g} = 11$ and the local iteration vectors

$$n_{it,1} = [1\ 1\ 2\ 2\ 3\ 3\ 3\ 3\ 3\ 3\ 4]$$

and

$$n_{it,2} = [1\ 2\ 2\ 2\ 3\ 3\ 3\ 3\ 3\ 3\ 2]\ .$$

When considering the EXIT chart in Fig. 7, we can also minimise the computational complexity of decoding. Once through the bottleneck, there are several possibilities of reaching the point $(I_a; I_e) = 1$. By applying a search algorithm, we would be able to find the decoding schedule which converges with lowest complexity. Hereto, we consider the complexity estimation from [3], where the number of equivalent additions is given for the decoding of a convolutional code with m memory elements. For a Log-MAP decoder, it has been computed as

$$N_{conv}(m) = 48 \cdot 2^m - 13\ . \tag{5}$$

For a complete decoding of a hybrid Turbo code, we perform $n_{it,g}$ global decoding iterations, where each iteration contains the decoding of both branches, which in turn consist of $n_{it,1/2}$ and $n_{it,1/2} - 1$ decodings of the outer and inner codes, respectively, plus one additional decoding of the very first inner decoder. Thus, the overall complexity is given by

$$\begin{aligned} N_{TC} \quad = \quad & n_{it,g} \cdot \Big(n_{it,1} \cdot N_{conv}(m_{11}) + (n_{it,1} - 1) \cdot \ldots \\ & N_{conv}(m_{12}) + n_{it,2} \cdot N_{conv}(m_{21}) + (n_{it,2} - 1) \cdot \ldots \\ & N_{conv}(m_{22}) \Big) + N_{conv}(m_{11})\ , \end{aligned} \tag{6}$$

where m_{ij} denotes the number of memory elements of encoder G_{ij}. For the above example codes (3) and (4), the numbers of memory elements are $m_{11} = m_{21} = m_{12} = m_{22} = 2$ which leads to

$$N_{TC} = n_{it,g} \cdot 358 \cdot (n_{it,1} + n_{it,2} - 1) + 179 \tag{7}$$

for fixed $n_{it,1}$ and $n_{it,2}$. For the same codes and variable numbers of local iterations, the number of equivalent additions can be determined by

$$N_{TC} = \Big(1 + \sum_{i=1}^{n_{it,g}} 2 \cdot n_{it,1}(i) + 2 \cdot n_{it,2}(i) - 2 \Big) \cdot 179\ . \tag{8}$$

We now compare the complexity of several decoding schedules that lead to a converging decoding process. First, we give the complexity of a decoding with constant and equal numbers of local iterations, i.e., $n_{it_1} = n_{it,2} = 3$. From Fig. 6, we can see that $n_{it,g} = 10$ is required to converge. With Eq. (7), we compute $N_{TC,3-3} = 18079$ or equivalently, the number of decoder activations is 101. When improving the decoding by applying the decoding schedule given in Fig. 7, the number of equivalent additions is $N_{TC,var} = 15931$ and the decoding only needs 89 decoder activations. Thus, we are able to reduce the decoding complexity of such a hybrid Turbo decoding by means of the new presented decoding analysis. We found that the bit-error rate is not increased by this improvement.

V. Conclusions

We defined a special concatenation of convolutional codes, which we call hybrid, consisting of a mixture of parallel and serial concatenations. We described the information processing of the iterative decoder in detail and developed a new analysis of the decoding process using multiple EXIT charts. By means of this analysis, we can determine the number of decoding iterations required to pass the bottleneck in the EXIT chart at critical SNR and to schedule the activation order of the component decoders in order to reduce the decoding complexity while keeping the convergence properties in the waterfall region.

Acknowledgment

This work is part of the FP6 / IST project M-Pipe and is co-funded by the European Commission.

References

[1] Berrou, C., Glavieux, A., Thitimajshima, P., "Near Shannon limit error-correcting coding and decoding: turbo codes", *Proc. IEEE International Conference on Communication (ICC)*, May 1993, Geneva, Switzerland, May 1993, pp. 1064-1070.

[2] ten Brink, S., "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. on Communications*, Vol. 49, No. 10, pp. 1727-1737, Oct. 2001.

[3] Chatzigeorgiou, I.A., Rodrigues, M.R., Wassell, I.J., Carrasco, R., "A Comparison of Convolutional and Turbo Coding Schemes for Broadband FWA Systems", *12th International Conference on Telecommunications*, May 2005, Cape Town, South Africa.