A Pipelined Turbo Decoder with Random Convolutional Interleaver

Werner Henkel
University of Applied Sciences
Neustadtswall 30
D-28199 Bremen, Germany
Email: Werner.Henkel@ieee.org
(formerly ftw.)

Laith Jusif and Jossy Sayir
Telecommunications Research Center (ftw.)
Donau-City-Str. 1
A-1220 Vienna, Austria
Email: {jusif, sayir}@ftw.at

***Abstract*— This paper describes a pipelined iterative decoder ("Turbo" decoder) for parallel concatenated codes offering delay advantages when the decoding speed of the component decoders is not significantly higher than the channel data rate. The key component of the decoder is a convolutional interleaver replacing the usual random block interleaver. This interleaver is randomized by changing the switching sequence according to a pseudo-random sequence known at the transmitter and receiver.**

I. Introduction

SINCE the introduction of "Turbo" codes in 1993 [1] the coding community has put much effort in approaching the Shannon limit even further. Practically, however, the delay is much more an issue that can disqualify Turbo-like codes. Usually, big interleavers of at least 2000 bits are proposed and the codes that approach the Shannon limit very closely only do this with much bigger interleavers and an increased number of iterations. Our focus is on the reduction of the latency by using convolutional-type interleavers and codes that converge in only a few iterations, even when we do not approach the Shannon limit too tightly. We adopt the original parallel concatenation. The decoder, however, is pipelined to allow for higher and continuous data throughput. This is typically required for high data rates as, *e.g.*, in optical communications. The pipelined structure requires sequential operations at every stage. This means not only a convolutional interleaver but also a decoding algorithm that works sequentially, like the SOVA [2], [3] or a sliding-window BCJR algorithm. These algorithms do not require a blocked, terminated trellis like the original BCJR algorithm [4] but operate with a limited decoding delay like the original Viterbi algorithm does.

A standard convolutional interleaver is very structured and will thus be inferior to the random interleavers usually applied in Turbo coding. However, in the following section it will become clear that randomization of a convolutional interleaver is straight forward. Section 3 will then present the pipelined Turbo decoder and Section 4 contains performance results together with the so-called EXIT charts [5] that offer an insight into the convergence of the iterative decoding algorithm at a certain signal-to-noise ratio.

II. A Randomized Convolutional Interleaver

The interleaver is *the* key element of a Turbo coding scheme. It determines the performance and causes the delay. The interleaver type that is usually chosen for non-Turbo applications is a convolutional interleaver due to its lower latency. We now adopt these interleavers for Turbo codes as well.

Convolutional interleavers have especially been published by Ramsey and Forney in [6] and [7], respectively. We follow Forney's approach which is shown in Fig. 1. In there, D_i and μ denote the interleaving depth and the length of the delay elements, respectively. When comparing with a standard block interleaver, the second dimension apart from the interleaving depth, which is usually chosen to be the code length N of the outer code, corresponds to $N = \mu \cdot D_i + 1$. Due to the triangular structure of the delay registers with its complementary use in transmitter and receiver, the total delay (and memory requirement) is half the one that would be needed for a block interleaver. The random interleaver of a Turbo code is actually a block interleaver, although the order is not regular.

In iterative decoding, the (de-)interleaving has to ensure (almost) statistical independence in the estimations of the two decoders. A random (or S-random) interleaver thus has advantages compared to very regular ones. The convolutional interleaver in Fig. 1 can easily be randomized by just changing the order of the switches according to a random sequence which is modified after each full (randomized cycle). The sequence needs, of course, to be known at the receiver to allow for synchronous switching.

The convolutional interleaver is also a key element of the pipelined Turbo decoder which will be discussed in the following section.

III. A Pipelined Turbo Decoder

The well-known Turbo encoder and decoder structures are redrawn in figs. 2 and 3, respectively. A pipelined architecture is achieved by duplicating the decoders, interleavers, and deinterleavers as in Fig. 4, thus realizing the iterations at different stages of the pipeline. A fully pipelined design requires sequential operations. One such sequential component is the convolutional interleaver that has already been described. As

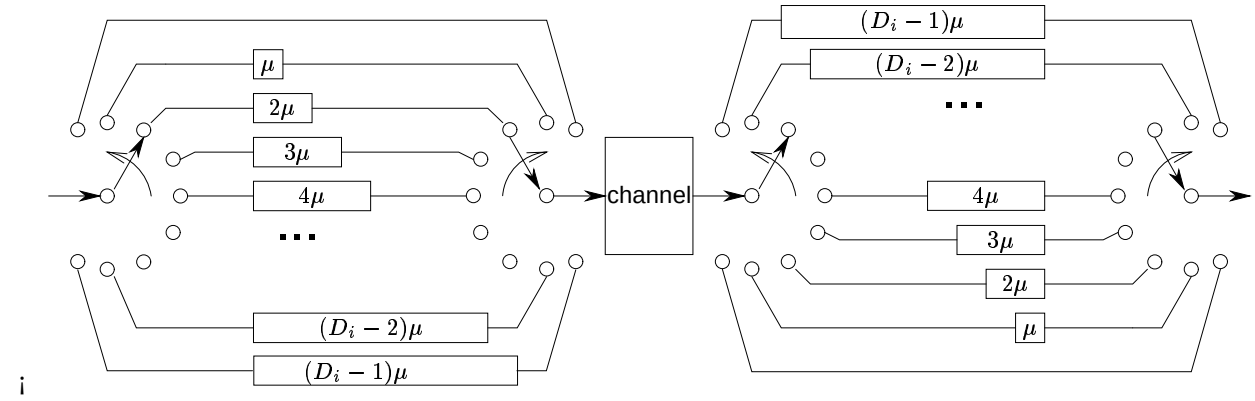

Fig. 1. Convolutional interleaver according to [7]

a sequential[1] decoder we implemented a sliding-window BCJR algorithm. This is essentially the same as the original BCJR algorithm as far as the forward recursion and the current transition are concerned (α and γ, respectively). For the backward recursions, an approximation is applied, namely initializing the β's with the α's of the forward recursion at some depth in the trellis. As long as this depth, *i.e.*, the window is big enough, no significant performance degradation is expected.

IV. Performance Results

As component codes we chose very simple ones, namely the well-known $(7,5)_8$ code. It is especially important to select codes with low constraint length, since we are concentrating on very small interleavers in the order of a few hundred bits. A high constraint length together with a small interleaver very easily produces cycles that quickly stop the iterative improvement of the likelihoods (*i.e.*, log-likelihood ratios).

The convergence can be observed by means of the so-called EXIT charts [5], where the function between the mutual information between the transmitted data and the a-priory knowledge and the mutual information between the transmitted and the extrinsic information after the component decoding is depicted. Thus, they unveil the effect of the decoding of a component code on the mutual information, *i.e.*, the function $I_A = f(I_E)$. Since the extrinsic information serves as an a-priory information for the next decoding stop, there is a similar relation for the second decoder. This can be shown as a second curve with swapped axis. In [5] a bit-error scaling was added that we also include in our figures. Fig. 5 shows an exemplary EXIT chart with a possible trajectory for our coding scheme at an E_b/N_0 of 1.5 dB. One can see that due to the shape of the mutual information functions, the convergence is extremely fast. In order to approach capacity, however, one would design curves that only leave a small corridor.

We now compare the bit-error rate curves of a random block interleaver with the ones of a randomized convolutional interleaver obtained by simulations. We draw separate curves for

[1] 'Sequential' decoder should not be understood in the classical sense standing for algorithms like Fano, Stack, and alike.

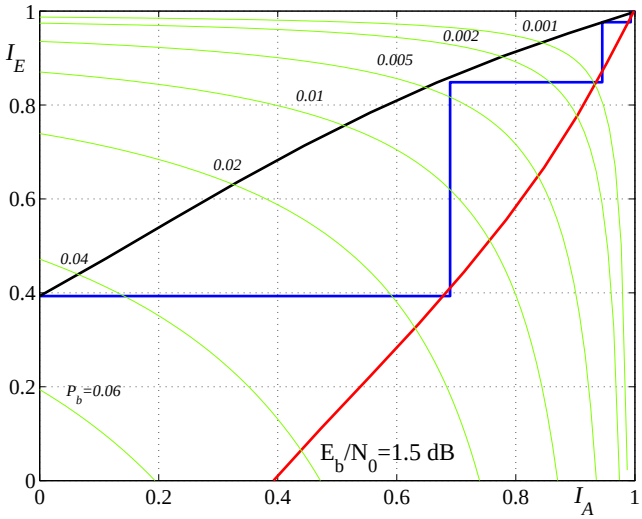

Fig. 5. EXIT chart for (7,5) component codes at $E_b/N_0 = 1.5$ dB

every iteration, meaning the outcome of a cycle of two decoding steps (decoders I and II). The result is shown in Fig. 6. To allow for a halfways fair comparison, we chose the delay of both interleavers to be the same. The delay of convolutional interleaving plus deinterleaving is $(D_i - 1) \cdot \mu \cdot D_i \approx D_i^2 \mu$. The size of a random block interleaver can only be half of this value to make the delay equal. This causes the poorer performance of the random block interleaver in Fig. 6. The actual values in our simulations were $D_i = 13$, $\mu = 2$ with a delay of $12 \cdot 2 \cdot 13 = 312$. The corresponding random block interleaver would thus only have a size of 156, which is indeed very small. Note that we found acceptable performances with our very small randomized convolutional interleaver, though.

V. Conclusions

We presented a fully pipelined Turbo decoder featuring randomized convolutional interleaver/deinterleaver pairs and a sliding-window BCJR decoding algorithm for the component codes. We have obtained acceptable performances at very small

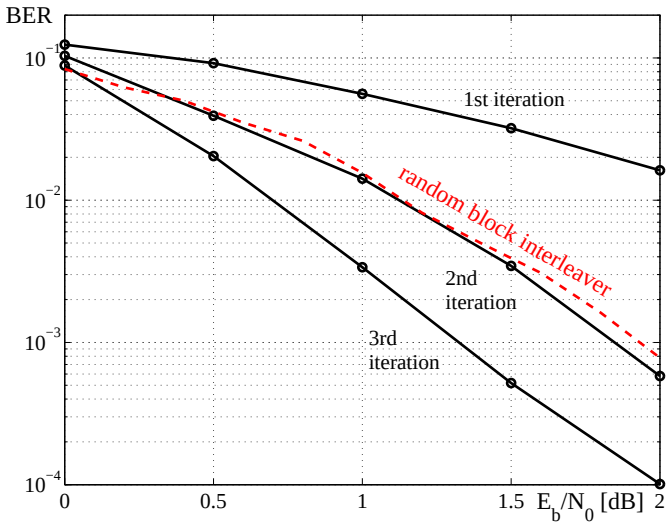

Fig. 6. Bit-error rates as a function of E_b/N_0 for a randomized convolutional interleaver (first three iterations) and a random block interleaver (only 3rd iteration shown) of equal delay.

delay and fast convergence. This scheme, however, is only suited when the decoding speed of the component decoders is in the order of the data rate on the channel. Note that the delay of the convolutional (de-)interleavers accumulate in the pipeline, which seems to be unavoidable. If the decoder speed should be much higher than the data rate, a conventional random block interleaver is preferred, since the decoders can operate on the block of data much faster at later iterations independent of the data rate on the channel.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," *Proc. ICC*, May 1993, pp. 1064-1070.

[2] J. Huber, and A. Rüppel, "Zuverlässigkeitsschätzung für die Ausgangssymbole von Trellis-Decodern," *Proc. AEÜ*, 1990, Vol. 44, No. 1, pp. 8-21. (in German)

[3] J. Hagenauer, P. Höher: "A Viterbi algorithm with soft-decision outputs and its applications", *Proc. Globecom 1989.*

[4] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Tr. on Inf. Th.*, Vol. IT-20, March 1974, pp. 284-287.

[5] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Tr. on Comm.*, Vol. 49, No. 10, October 2001, pp. 1727-1737.

[6] J.L. Ramsey, "Realization of optimum interleavers", *IEEE Tr. on Inf. Th.*, Vol. IT-16, May 1970, pp. 338-345.

[7] G.D. Forney, "Burst-correcting codes for the classic bursty channel", *IEEE Tr. on Comm. Techn.*, Vol. COM-19, Oct. 1971, pp. 772-781.

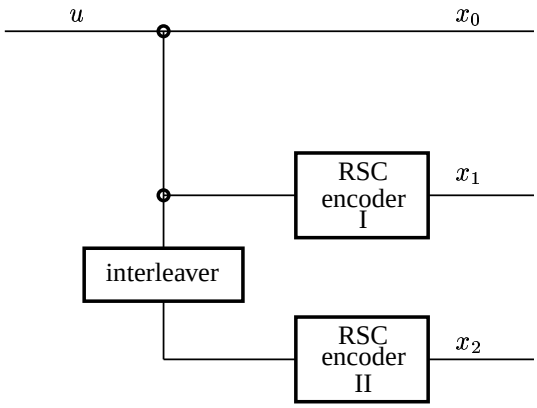

Fig. 2. Turbo encoder of rate 1/3

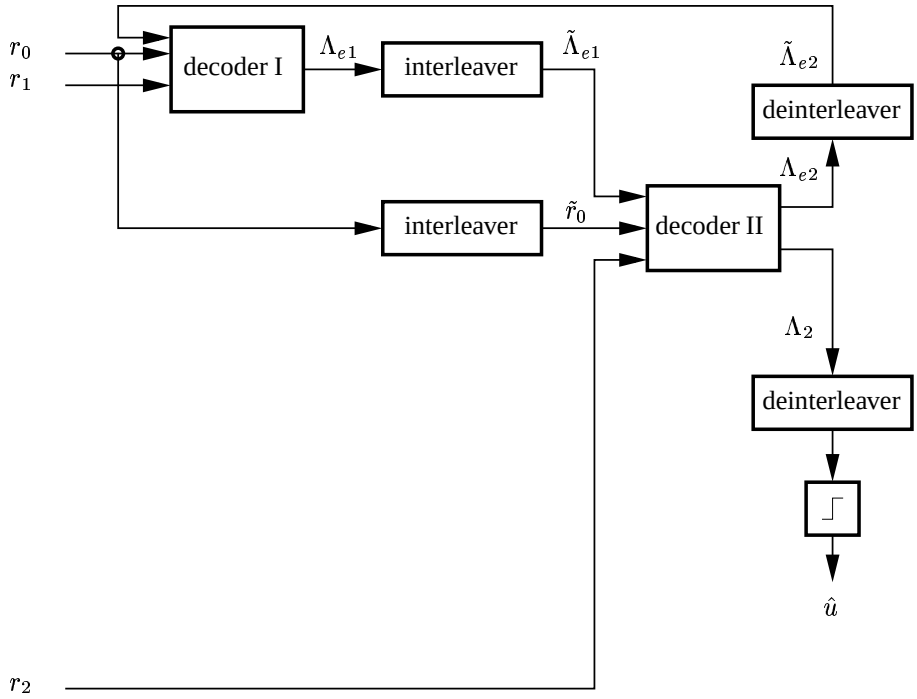

Fig. 3. Turbo decoder $(R = 1/3)$

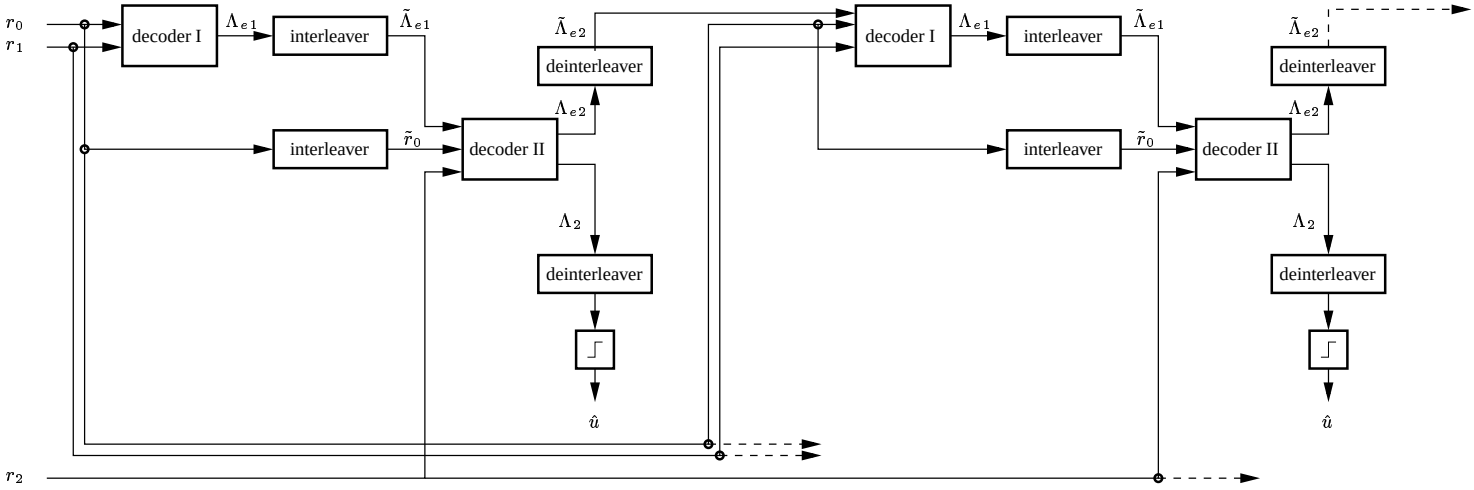

Fig. 4. Pipelined Turbo decoder $(R = 1/3)$