Syndrome Based Block Decoding of Convolutional Codes

Jan Geldmacher, Klaus Hueske, Jürgen Götze

Information Processing Lab
Department of Electrical Engineering, Dortmund University of Technology
Otto-Hahn-Strasse 4, 44227 Dortmund, Germany
jan.geldmacher@tu-dortmund.de

***Abstract*—A block processing approach for decoding of convolutional codes is proposed. The approach is based on the fact that it is possible for Scarce-State-Transition decoding and syndrome decoding to determine the probability of a certain trellis state before the actual decoding happens. This allows the separation of the received sequence into independant blocks with known initial and final states, thus making overlapping or modifications of the encoder or the information stream unnecessary. The proposed scheme offers potentials for both parallelization and reduction of power consumption.**

I. Introduction

Partitioning a convolutional encoded sequence into blocks is a widely used approach for both parallel decoding and decoding in block processing systems.

One option to achieve this is to modify the encoder and to periodically force it into a certain state, by either inserting zero sequences into the stream of information bits or just resetting the encoder to a certain state (zero-state)[1]. If the intervals are known on decoder side, the received sequence can be separated into blocks accordingly. However, these approaches lead to a reduction of the code rate or a degradation of the decoding performance, respectively. Additionally, in the second approach a modification of the encoder is required.

Another approach for block decoding is to separate the received sequence into overlapping blocks and to decode these blocks independently. This general principle has been called sliding block decoding [2], overlap-add Viterbi algorithm [3] or sliding block Viterbi decoder [4]. It is also applied in the minimized method presented in [5]. In this work an approach for block decoding of convolutional codes is proposed, which does not require modifications of the input stream, modifications of the encoder or overlapping.

This paper is organized as follows: Section II summarizes previous work on decoding of convolutional codes using overlapping techniques and presents the basic idea of the proposed block processing approach. In the following section two error decoding algorithms are compared in terms of their suitability for the proposed approach. Section IV describes the proposed block processing scheme and finally in Section V simulation results are discussed.

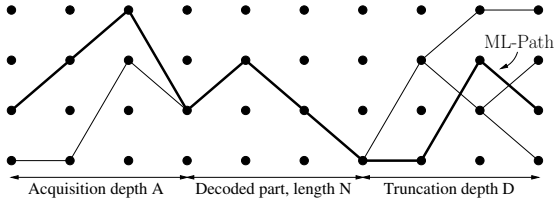

Fig. 1. Composition of a trellis block.

II. Block processing of convolutional codes

The sliding block decoder has been proposed to achieve fast decoding of convolutional codes [2]. The basic idea is, to "cut" blocks out of the received sequence and determine a decoded symbol for every block by looking it up in a pre-calculated table. However, it is known that the blocks have to overlap to a certain extend to make them independent from each other and to achieve optimal decoding performance in maximum likelihood sense. The required minimum overlapping length is governed by two parts: the acquisition depth and the truncation depth. Fig. 1 illustrates the composition of a block with overlapping parts:

- *Acquisition part.* Because of the unknown initial state and metric, the first part of the block consists of an acquisition part of length A where the state metrics are dependent of the unknown initial metrics. After A steps the metric can be assumed to be independent from the initial metrics. This is likely the case for $A = 5\nu$ [6], where ν is the constraint length.
- *Decoded part.* The middle part of length N is the part where symbol estimates are delivered.
- *Truncation part.* When starting the traceback from the end of the block with unknown final state, the traceback length $D = 5\nu$ is the number of steps after which all paths have most likely merged [7].

Thus the additional minimum number of received symbols required to decode N symbols from an encoded sequence is $A + D = 10\nu$. Therefore it is clear, that an implementation of the sliding block decoder ($N = 1$) with table look-up is infeasible for higher constraint lengths ν because of huge memory requirements.

Approaches which overcome this problem are based on the same idea of extracting overlapping blocks from the input sequence [3], [4], [5]: However, in these approaches instead of a table-look-up the Viterbi decoder (VD) is used to deliver symbol estimates for each block. This enables the usage of an arbitrary number of parallel VDs. Thus arbitrary speed-ups through parallelization and high-data rates, especially in hardware realizations, are possible. A drawback of these procedures is, that many operations have to be carried out just for the state identification and without actually delivering decoded symbols. It is therefore interesting to investigate how the necessity of overlapping could be removed without modifying the encoder.

This can be achieved if the encoded sequence is separated into blocks with known initial and final states. For Viterbi Decoding this is impossible, because the VD decodes a maximum likelihood (ML) path, which passes through all trellis states with the same probability and it is impossible to decide which state is on the ML path at a certain time without actually decoding this part. However, there are other decoding algorithms based on the Viterbi algorithm (VA), which pass a certain state with higher probability. These algorithms are based on the idea of estimating error sequences instead of the code sequences and using this estimate to correct the transmission errors. This leads to decoding with unbalanced state probabilities because the state probabilities now depend on the transmission errors and no longer on the information sequence. For these algorithms in error-free parts the decoding path traverses a certain trellis state with high probability. This state with register contents all zero and a leaving edge with input/output bits all zero is refered to as the "zero-state" in the following sections. In the block processing approach proposed in this paper, error-free parts are identified before decoding and the sequence is separated at these parts into bocks with initial and final state being the zero-state.

III. Approaches to error decoding

Several approaches for decoding error sequences have been proposed, as e.g. syndrome decoding (SD) [8], syndrome decoding using the general syndrome equation (GSD) [9] and Scarce-State-Transition decoding (SST) [10]. In the following these approaches are summarized in brief, and it is shown that SST and GSD are actually identical algorithms. Further on SD and SST are compared in terms of the probability of the zero-state. This should give an indication on how often a separation of the received sequence is possible.

In the following matrices and vectors have polynomial elements in D with coefficients from $GF(2)$. The addition/subtraction-operator is denoted by $\oplus$.

A. Viterbi decoding

An information sequence $\mathbf{u}$ is encoded with a generator matrix $\mathbf{G}$ and transmitted over a channel. With $\mathbf{e}$ representing the channel error, the received sequence can be expressed as

$$\mathbf{r} = \mathbf{uG} \oplus \mathbf{e} = \mathbf{v} \oplus \mathbf{e}, \tag{1}$$

where $\mathbf{v}$ is the encoded sequence. The decoding problem can then be written in terms of an optimization problem:

$$\min\|\hat{\mathbf{e}}\|, \tag{2}$$

i.e. to find an estimation of the information sequence $\hat{\mathbf{u}}$ with minimum estimation error $\hat{\mathbf{e}}$, where

$$\hat{\mathbf{e}} = \mathbf{r} \oplus \hat{\mathbf{u}}\mathbf{G} = \mathbf{r} \oplus \hat{\mathbf{v}}. \tag{3}$$

In Viterbi decoding [11] the problem (2) may be formulated as

$$\min_{\hat{\mathbf{v}}}\|\mathbf{r} \oplus \hat{\mathbf{v}}\| = \min_{\hat{\mathbf{u}}}\|\mathbf{r} \oplus \hat{\mathbf{u}}\mathbf{G}\|, \tag{4}$$

where $\hat{\mathbf{e}}$ has been substituted by (3). This optimization problem is solved by applying the VA to search the trellis of the encoder $\mathbf{G}$ for the code sequence $\hat{\mathbf{v}} = \hat{\mathbf{u}}\mathbf{G}$ with minimum distance to the received sequence $\mathbf{r}$. As usual, it is assumed that all information sequences have equal probability. Therefore, all code sequences and accordingly all paths through the encoder trellis and conseuently all trellis states will have equal probability, too. Hence in the decoding process all trellis states will appear with equal probability 2^{ν} on the ML path.

B. Syndrome decoding

In syndrome decoding [8], instead of searching for $\hat{\mathbf{v}}$, one searches for a sequence $\hat{\mathbf{e}}$, which corrects the errors in $\mathbf{r}$. The according constrained optimization problem can be stated as

$$\min_{\hat{\mathbf{e}}}\left\{\|\hat{\mathbf{e}}\| \mid \mathbf{r}\mathbf{H}^T = \hat{\mathbf{e}}\mathbf{H}^T\right\}, \tag{5}$$

where $\mathbf{H}^T$ is called the syndrome former matrix and $\mathbf{r}\mathbf{H}^T = \mathbf{b}$ is called the syndrome of $\mathbf{r}$. The equivalence to (4) can easily be verified by inserting (3) into (5). The syndrome former $\mathbf{H}^T$ is defined to be orthogonal to the generator matrix and thus to all code sequences. So it holds that

$$\mathbf{b} = \mathbf{r}\mathbf{H}^T = (\mathbf{v} \oplus \mathbf{e})\mathbf{H}^T = \mathbf{u}\mathbf{G}\mathbf{H}^T \oplus \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

and it is obvious that the syndrome only depends on the error sequence. Thus error-free periods in the received sequence can be identified by detecting parts of consecutive zeros with sufficient length in the syndrome sequence. This allows the detection of error-free periods before the actual decoding happens. A similar idea has recently been used to avoid the search for the best metric which is necessary when applying the T-algorithm [12]. We note that the calculation of the syndrome is realised by simple XOR-operations and thus is of negligible complexity.

The optimization problem (5) is solved by applying the VA to search the trellis of the syndrome former $\mathbf{H}^T$ for the error sequence $\hat{\mathbf{e}}$, which satisfies $\hat{\mathbf{e}}\mathbf{H}^T = \mathbf{b}$ and has minimum weight. The estimated error sequence $\hat{\mathbf{e}}$ is then used to correct the transmission errors in $\mathbf{r}$. Finally the estimation of the information sequence is obtained by multiplication with the right inverse of the generator matrix $\mathbf{G}^{-1}$, $\hat{\mathbf{u}} = (\mathbf{r} \oplus \hat{\mathbf{e}})\mathbf{G}^{-1}$.

C. Scarce-State-Transition decoding

The SST decoding algorithm [10] searches for a sequence $\hat{\mathbf{a}} = \hat{\mathbf{e}}\mathbf{G}^{-1}$, which corrects the channel errors in the predecoded information sequence $\mathbf{r}\mathbf{G}^{-1}$. By rearranging (3) to $\mathbf{r} = \hat{\mathbf{e}} \oplus \hat{\mathbf{v}}$ and modifying it in the following way

$$\begin{aligned} \mathbf{r}\mathbf{G}^{-1} &= \hat{\mathbf{e}}\mathbf{G}^{-1} \oplus \hat{\mathbf{u}} && \text{predecoding} \\ \mathbf{r}\mathbf{G}^{-1}\mathbf{G} &= \hat{\mathbf{e}}\mathbf{G}^{-1}\mathbf{G} \oplus \hat{\mathbf{v}} && \text{reencoding} \\ \mathbf{r}\mathbf{G}^{-1}\mathbf{G} &= \hat{\mathbf{e}}\mathbf{G}^{-1}\mathbf{G} \oplus \hat{\mathbf{e}} \oplus \mathbf{r} && \text{using (3)} \\ \hat{\mathbf{e}} &= \mathbf{r}\mathbf{G}^{-1}\mathbf{G} \oplus \mathbf{r} \oplus \hat{\mathbf{a}}\mathbf{G} && \text{rearranging} \end{aligned}$$

the according optimization problem is found as

$$\min_{\hat{\mathbf{a}}} \|\mathbf{r}\mathbf{G}^{-1}\mathbf{G} \oplus \mathbf{r} \oplus \hat{\mathbf{a}}\mathbf{G}\|.$$

We note that by using the relation $\mathbf{G}^{-1}\mathbf{G} \oplus \mathbf{I} = (\mathbf{H}^T)^{-1}\mathbf{H}^T$, which can be derived from the invariant factor decomposition of $\mathbf{G}$ [13], this is seen to be equivalent to the GSD algorithm:

$$\min_{\hat{\mathbf{a}}} \|\mathbf{r}\mathbf{H}^T \left(\mathbf{H}^T\right)^{-1} \oplus \hat{\mathbf{a}}\mathbf{G}\| = \min_{\hat{\mathbf{a}}} \|\mathbf{b} \left(\mathbf{H}^T\right)^{-1} \oplus \hat{\mathbf{a}}\mathbf{G}\|$$

In both cases $\hat{\mathbf{a}}$ is found by applying the VA on the trellis of the encoder and the estimation of the information sequence is finally delivered from $\hat{\mathbf{u}} = \mathbf{r}\mathbf{G}^{-1} \oplus \hat{\mathbf{a}}$.

D. Comparison of the decoding algorithms

The decoding complexity of SD and SST in terms of states in the trellis and required ACS operations of the VA is identical to VD: SST operates on the same trellis as VD and the trellis of the syndrome former is of the same complexity as the corresponding encoder trellis for minimal generators [13]. However, the difference between VD and SD/SST is that for the latter the state probabilities are unbalanced, with probabilities depending on the probabilities of the error sequences and not on the probabilities of the code sequences: The better the transmission conditions the more zeros will be in the error sequence and the more often the ML path will merge back into the zero-state.

To decide whether SD or SST are better suited for the proposed block decoding approach, SST and SD are compared in terms of probability of the zero-state being on the ML path. Although both decoding algorithms are known to be equivalent [14], it is expected that this probability will differ, because the algorithms use different trellises for decoding. The probability of the zero-state should give an indication on how long it takes for a trellis path to merge back into the zero-state after an error event. Furthermore, the more often the zero-state appears on the decoded path, the more often a separation of the sequence is possible and the shorter are the resulting blocks.

Fig. 2 shows simulation results for the probability of the zero-state for several codes of constraint lengths $\nu = 2$, $\nu = 4$, $\nu = 6$ and $\nu = 8$. It can be noticed that the probability of the zero-state depends on

- *Signal-to-Noise Ratio (SNR).* The higher the SNR the more often the zero-state is on the ML-path because higher SNR leads to less transmission errors and therefore more *0*s in the decoded path.
- *Constraint length.* The higher the constraint length the less likely the zero-state because it takes longer for the decoded path to merge back into the zero-state after an error-event.
- *Trellis.* Clearly, the probability depends on the trellis structure. When comparing two codes with the same constraint length the zero-state probability differs for the SST, although both (encoder) trellises have the same number of states.This effect does not hold for the SD, where the probability is seen to be independant of the (syndrome former) trellis structure for the simulated codes.
- *SD or SST.* Comparing SD and SST, the zero-state probability of the SD is higher than for the SST for all simulated codes.

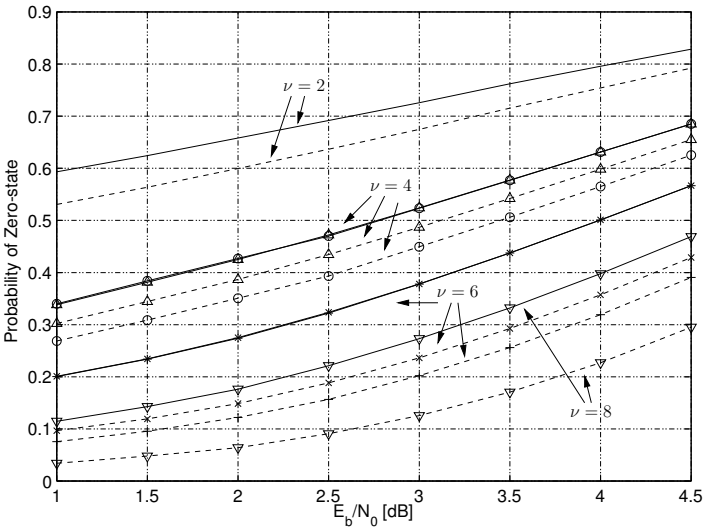

Fig. 2. Probability of the zero-state as function of the SNR for SST ($--$) and SD ($-$). Simulations for the codes with generator matrices $\mathbf{G} = (5_8, 7_8)$ ($-$), $\mathbf{G} = (23_8, 35_8)$ ($\triangle$), $\mathbf{G} = (27_8, 31_8)$ ($\circ$), $\mathbf{G} = (133_8, 171_8)$ ($+$), $\mathbf{G} = (117_8, 155_8)$ ($\times$), $\mathbf{G} = (561_8, 491_8)$ ($\triangledown$) .

These simulations suggest that the SD is more suitable for the proposed block decoding approach because it has a higher zero-state probability than the SST and is also less depending on the trellis structure than the SST decoder. Therefore, in the following section SD is applied for the proposed block processing approach and the corresponding algorithm is called block syndrome decoder (BSD).

IV. Block syndrome decoder

The principle of the BSD for parallel decoding is illustrated in Fig. 3. First the syndrome sequence is analyzed and parts with a sufficient number of consecutive zeros are identified ("all-zero parts"). Using this information, the received sequence is separated into blocks with initial and final state being the zero-state. The resulting blocks can then be distributed to the decoders. In this example the number of blocks equals the number of decoders, but one could clearly use an arbitrary (smaller) number of decoders along with a simple scheduling scheme to distribute the resulting blocks to the decoders.

Compared to the overlapping schemes described in Section II, in the proposed method the block length is not fixed and cannot be determined before decoding. The block length in fact

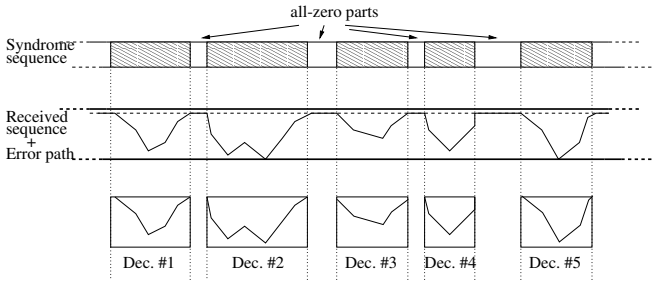

Fig. 3. Overview of the proposed block decoding approach.

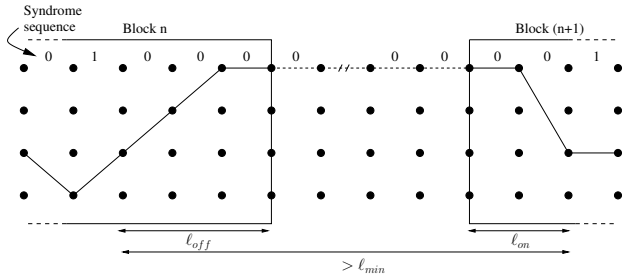

Fig. 4. Design parameters of the block processing scheme.

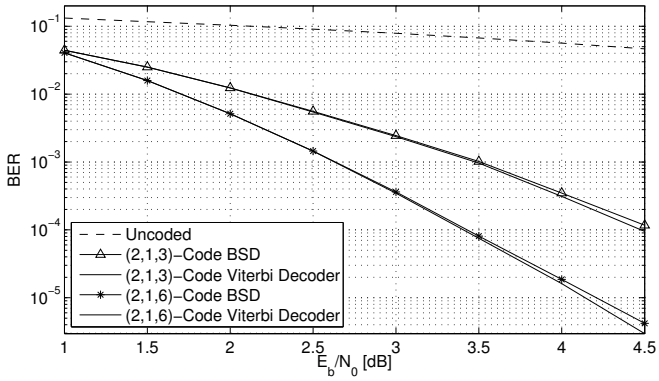

Fig. 5. BER as a function of the SNR for an AWGN channel.

depends on the channel error, i.e. the transmissions conditions: Good transmission conditions will allow separations more frequently and thus result in shorter block lengths, while difficult transmission conditions (more errors) will result in longer blocks.

In a practical realization of the proposed scheme one would obviously have to limit the block length to a suitable upper bound. This can easily be achieved by implementing overlapping as a fallback. This extension of the BSD is refered to as BSD/OL in the following. For the BSD/OL the block length is limited by partitioning blocks with length greater than a certain upper bound into overlapping subblocks. To avoid degradation of bit-error-rate in this scheme the overlapping length between consecutive subblocks is set to 5ν, as discussed in Section II.

As error-free and erroneous sequences can be identified it is reasonable to feed only those parts of the received sequence into the decoders, which are actually erroneous. For the error-free parts no error sequence has to be determined. This will result in an additional decoding speed-up, which however depends on the transmission conditions. A good SNR will result in longer error-free sequences and thus allows for higher speed-ups, while lower SNR will result in shorter error-free sequences and thus less speed-up. The additional speed-up could also be interpreted as a reduction of decoding complexity [15].

V. Simulations

In this section simulation results for BSD and BSD/OL are presented. In the following a code with rate $R = n/k$ and constraint length ν is denoted as (n, k, ν)-code. The codes used in the simulations are the (2,1,3)-code with generator matrix $\mathbf{G} = (13_8, 17_8)$ and the (2,1,6)-code with $\mathbf{G} = (133_8, 171_8)$.

A. Design Parameters

A critical part of the BSD is how to decide from the syndrome sequence when a separation of the received sequence is possible. Therefore, the following design parameters are defined: The parameter ℓ_{min} is defined to be the minimum number of consecutive *0*s in the syndrome sequence required for a separation. That is, whenever a sequence of consecutive *0*s with length greater or equal to ℓ_{min} is detected, the received sequence is separated at this point. The parameters ℓ_{off} and ℓ_{on} define where exactly the previous block ends and where the next block starts: ℓ_{off} denotes the number of *0*s in the syndrome sequence at the end of a block, i.e. the number of stages until it can be assumed with sufficient probability that the ML path is returned to the zero-state. The third parameter ℓ_{on} is defined to be the number of *0*s in the syndrome sequence from the beginning of a block to the first *1*. Fig. 4 illustrates the meaning of these parameters for a four state trellis, where values are chosen as $\ell_{off} = 3$ and $\ell_{on} = 2$.

The selection of the parameter values has an impact on the performance in terms of bit-error-rate (BER) and parallelization speed-up: Choosing too small values for ℓ_{on}, ℓ_{off} or ℓ_{min} will result in a degradation of the BER. On the other hand, large values for ℓ_{min} reduce the amount of possible separation points, resulting in longer blocks, and large values for ℓ_{on} and ℓ_{off} reduce the speed-up, that results from avoiding the decoding of the error-free parts between two consecutive blocks. Suitable parameter values can be determined from simulations of parameter settings versus bit-error-rate.

B. Simulation results and discussion

For a setup with two parallel decoders, which alternately decode the resulting blocks, simulations of the BER, the average block length and the speed-up factor for the BSD and the BSD/OL are given in Fig. 5-7. The design parameters have been set to $\ell_{on} = \ell_{off} = 3$ and $\ell_{min} = 10$ for the (2,1,3)-code and $\ell_{on} = \ell_{off} = 6$ and $\ell_{min} = 16$ for the (2,1,6)-code.

The decoding performance in terms of the BER given in Fig. 5 can be seen to be almost as good as the decoding performance of the standard decoder. For the (2,1,3)-code there is a loss of less then 0.1dB for high SNR, while the (2,1,6)-code shows a loss of about 0.1dB. Clearly, this loss of coding

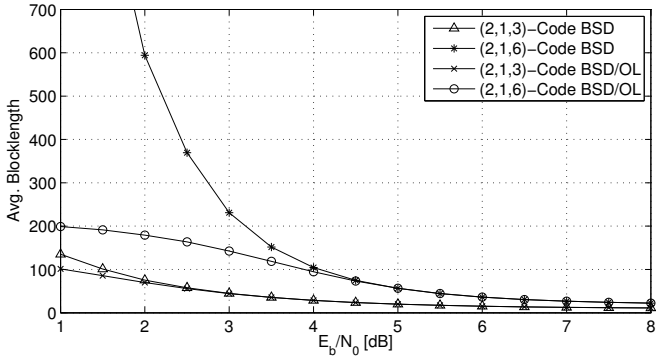

Fig. 6. Average length of decoded blocks as a function of the SNR.

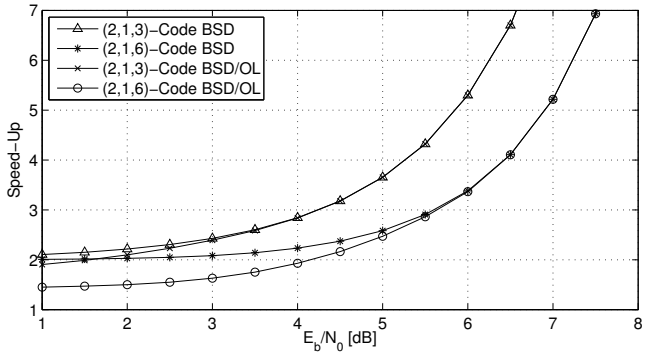

Fig. 7. Speed-Up factor for 2 decoders as function of the SNR.

gain could be compensated by choosing higher values for ℓ_{min}, which however would lead to a lower speed-up factor. The average block length as a function of the SNR is depicted in Fig. 6. The block length decreases with increasing SNR because the better the transmission conditions, the more error-free periods with sufficient length and the more separation possibilities occur. Obviously the choice of a smaller ℓ_{min} for the (2,1,3)-code, results in significant smaller block lengths for this code, compared to the (2,1,6)-code. For the BSD/OL the block length has been limited to $N = 150$, which means that it will converge to $N + 10\nu$ for low SNR, due to the involved overlapping. Fig. 7 finally shows the speed-up of the BSD and BSD/OL for the given setup with two decoders. For the BSD the speed-up S_{BSD} would be 2, when considering received sequences with infinite length. However, for medium to high SNR longer error-free sequences, which do not require any decoding, occur, thus resulting in an additional speed-up, that increases with SNR. It can be noticed that because of the different settings of the parameter ℓ_{min} the (2,1,3)-code achieves a higher additional speed-up than the (2,1,6)-code. Considering the BSD/OL the speed-up $S_{\text{BSD/OL}}$ is smaller than S_{BSD}, because of the limited block length and the involved overlapping overhead. With increasing SNR $S_{\text{BSD/OL}}$ converges towards S_{BSD}. Thus $S_{\text{BSD/OL}}$ may be expressed as

$$S_{\text{BSD/OL}} = \frac{NP}{N+\Delta} \text{ with } \Delta = \begin{cases} A+D & \text{for SNR} \to -\infty \\ -N & \text{for SNR} \to \infty \end{cases}$$

and with P the number of decoders and N the block length. We note that the additional speed-up can be traded against a reduction of the power consumption by just switching off the decoders periodically.

VI. Conclusion

An approach for block parallel decoding of convolutional encoded sequences has been presented. The key idea is to use the syndrome of the received sequence to identify error-free parts, where the ML path is known to merge back into the zero-state. The sequence can be separated at these points with insignificant performance degradation. Using this block processing approach, modifications of the encoder or the input stream as well as the use of overlapping techniques, which imply redundant operations, can be avoided. To bound the lengths of the resulting blocks the scheme can be extended to use overlapping as a fallback for low SNR. The speed-up of the resulting scheme equals the speed-up of the well-known overlapping VD, but increases significantly with increasing SNR.

References

[1] H.-D. Lin and D. Messerschmitt, "Algorithms and architectures for concurrent viterbi decoding," in *IEEE International Conference on World Prosperity Through Communications (ICC89)*, vol. 2, June 1989, pp. 836–840.

[2] K.-H. Tzou and J. Dunham, "Sliding block decoding of convolutional codes," *IEEE Transactions on Communications*, vol. 29, no. 9, pp. 1401–1403, Sept. 1981.

[3] P. Black and T.-Y. Meng, "A hardware efficient parallel viterbi algorithm," in *International Conference on Acoustics, Speech, and Signal Processing, 1990. ICASSP-90.*, vol. 2, April 1990, pp. 893–896.

[4] ——, "A 1-Gb/s, four-state, sliding block viterbi decoder," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 797–805, June 1997.

[5] G. Fettweis, H. Dawid, and H. Meyr, "Minimized method viterbi decoding: 600 Mbit/s per chip," in *IEEE Global Telecommunications Conference.*, vol. 3, Dec. 1990, pp. 1712–1716.

[6] G. Fettweis and H. Meyr, "Feedforward architectures for parallel viterbi decoding," *The Journal of VLSI Signal Processing*, vol. 3, no. 1-2, pp. 105–119, June 1991.

[7] S. Lin and D. J. Costello Jr., *Error Control Coding*, 2nd ed. Prentice Hall, 2004.

[8] J. Schalkwijk and A. Vinck, "Syndrome decoding of binary rate-1/2 convolutional codes," *IEEE Transactions on Communications*, vol. 24, no. 9, pp. 977–985, Sept. 1976.

[9] I. Reed and T. Truong, "Error-trellis syndrome decoding techniques for convolutional codes," *IEE Proceedings Pt. F*, vol. 132, no. 2, pp. 77–83, April 1985.

[10] T. Ishitani, K. Tansho, N. Miyahara, S. Kubota, and S. Kato, "A scarce-state-transition viterbi-decoder VLSI for bit error correction," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 4, pp. 575– 582, Aug. 1987.

[11] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[12] J. Jin and C. ying Tsui, "Low-power limited-search parallel state viterbi decoder implementation based on scarce state transition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1172–1177, Oct. 2007.

[13] G. Forney Jr., "Convolutional codes I: Algebraic structure," *IEEE Transactions on Information Theory*, vol. 16, no. 6, pp. 720–738, Nov. 1970.

[14] M. Tajima, K. Shibata, and Z. Kawasaki, "On the equivalence between scarce-state-transition viterbi decoding and syndrome decoding of convolutional codes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. Vol.E86-A, no. 2003/08/01, pp. 2107–2116, 2003.

[15] K. Hueske, J. Geldmacher, and J. Goetze, "Adaptive decoding of convolutional codes," in *Advances in Radio Science*, vol. 5, 2007, pp. 209–214.