PERFORMANCE OF SERIAL CONCATENATED CODES UNDER ITERATIVE DECODING AND DIFFERENT UPDATE MODES

Anne Wolf, Jochen Ertel, Adolf Finger

Dresden University of Technology, Germany

ABSTRACT

Concatenated codes are typically decoded using iterative methods. The sum-product algorithm can be shown to be optimal for cycle-free codes. In this paper two commonly used versions of the sum-product algorithm are investigated with respect to their performance for concatenated codes with cycles. Various serial concatenated codes with an inner accumulator are studied. Simulation results for both update modes are presented for Repeat-Accumulate codes, Accumulate-Repeat-Accumulate codes, and Product-Accumulate codes.

1. INTRODUCTION

An accumulator (the simplest rate-1 recursive convolutional code) is an often used subcode for the construction of concatenated codes. In the last years, many serial concatenated codes with an accumulator as inner code have been presented, e.g., Repeat-Accumulate codes (RA) [1][2], Accumulate-Repeat-Accumulate codes (ARA) [3], or Product-Accumulate codes (PA) [4].

This work was initialized by an observation for the well known Repeat-Accumulate codes: the two different iterative decoding modes for an inner accumulator (serial and parallel update) result in different behaviors of the overall bit-error rate. Therefore, the focus in this paper is set on codes with a similar structure, i.e., serial concatenated codes with an inner accumulator. An important issue is the interrelationship of the accumulator decoding with the capability of the concatenated codes to correct errors, since the utilized iterative decoding – implemented by message-passing – is not optimal for codes with cycles [5].

This paper is organized as follows. In the next section, the method of iterative decoding for concatenated codes and its realization with message-passing are described briefly. The issues that result from code graphs with cycles are discussed. In Section 3 the two commonly used update modes for the accumulator decoding are described. Some simulation results for Repeat-Accumulate, Accumulate-Repeat-Accumulate, and Product-Accumulate codes are presented in Section 4 and discussed in Section 5.

2. ITERATIVE DECODING AND CYCLES IN THE CODE GRAPH

It is well known that concatenated codes should consist of relatively simple component codes, in order to allow simple local decoding. The latter is required for an efficient iterative decoding of the overall code. Moreover, the performance of a concatenated code relies on sufficient randomness, which can be easily achieved by a randomly generated interleaver pattern. The structure of the code ensures a manageable complexity for the encoding and the decoding of the code, whereas the introduced randomness permits a good error correcting capability [6].

Iterative decoding is based on the replacement of the decoding of the received word as a whole by separate decodings of the component codes and a frequent interchange of extrinsic information between them. The procedure is repeated for several times (iterations).

A Tanner graph is a simple graphical visualization of the structure of a concatenated code. It consists of variable nodes (○), which stand for information or parity bits, check nodes (⊞), representing parity checks of all neighboring variable nodes, and edges between the nodes. A Tanner graph can be used for an efficient implementation of the iterative decoding process, denoted message-passing or sum-product algorithm [5]. The nodes act as independent processing units. They compute outgoing messages at every edge from the incoming messages of all other incident edges of this node. The messages within the graph are typically log-likelihood ratios (LLRs) corresponding to the values of the variable nodes.

When the incoming information is processed, it is typically assumed that the information on different edges is independent of each other. As long as the neighborhood of every node is tree-like, the required independence of the incoming messages is guaranteed [6]. But if a message can return to its origin via several nodes and edges, the code graph will contain so-called cycles. Only on cycle-free code graphs message-passing decoding will converge to the optimal solution [7][8]. Message-passing algorithms applied to concatenated codes despite their cycles generally achieve no longer optimum decoding but they are still pretty good. Moreover, iterative decoding has an excellent complexity vs. performance trade-off [6].

The Tanner graph of a short Repeat-Accumulate code is shown in Fig. 1. It is obvious that such a concatenated code must possess cycles. The construction of good codes using serial concatenation inevitably results in the presence of cycles in the code graph. Very short cycles in the code graph impair the performance of iterative decoding. But their probability extremely decreases for longer block lengths.

It is apparent that it depends on the number of completed iterations whether the neighborhood of each node is still tree-like, and thus, whether the message-passing decoding is still cycle-free.

3. DECODING OF THE ACCUMULATOR

For a single accumulator, there is no need to apply iterative decoding, since a simple accumulator can be decoded in one step. However, if an accumulator is one component code of a concatenated code, it will be decoded in each iteration, and messages will be exchanged with the other subdecoders.

The following parts of this section present two typically used update modes for the decoding of the accumulator that base on the representation of the accumulator by its Tanner graph.

3.1. Serial Accumulator Update

The serial update of the messages in the accumulator graph corresponds to an optimum decoding for an accumulator [7][8]. Messages are passed forward and backward through the accumulator graph, so that they finally contain all the information available for every variable node from all other nodes (cf. Fig. 2).

The values calculated by the outer subdecoder in the previous iteration are included as so-called a-priori information $L_o(x_i)$ in the computation of the messages in the code graph of the accumulator (the inner subdecoder). The information arriving at variable node y_i from the forward and backward path is denoted $L_{ef}(y_i)$ and $L_{eb}(y_i)$. In the μ-th iteration their values can be computed by

$$L_{ef}^{(\mu)}(y_i) = L_o^{(\mu-1)}(x_i) \boxplus \left[L_{ch}(y_{i-1}) + L_{ef}^{(\mu)}(y_{i-1}) \right] \quad (1)$$

$$L_{eb}^{(\mu)}(y_i) = L_o^{(\mu-1)}(x_{i+1}) \boxplus \left[L_{ch}(y_{i+1}) + L_{eb}^{(\mu)}(y_{i+1}) \right], \quad (2)$$

where $L_{ch}(y_i)$ denotes the LLR at the output of the channel [7][8]. The check operation $\boxplus$ is defined as [9]

$$c = a \boxplus b = 2 \cdot \operatorname{atanh}\left(\tanh\left(\tfrac{a}{2}\right) \cdot \tanh\left(\tfrac{b}{2}\right) \right). \quad (3)$$

The extrinsic information of check node x_i in the μ-th iteration is determined by

$$L_e^{(\mu)}(x_i) = \left[L_{ch}(y_{i-1}) + L_{ef}^{(\mu)}(y_{i-1}) \right] \boxplus \left[L_{ch}(y_i) + L_{eb}^{(\mu)}(y_i) \right], \quad (4)$$

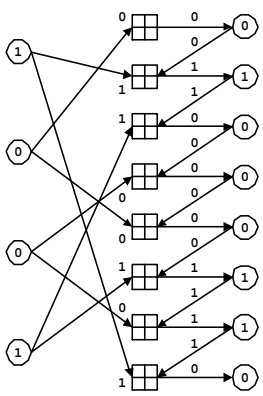

Fig. 1 Tanner graph of a short RA code

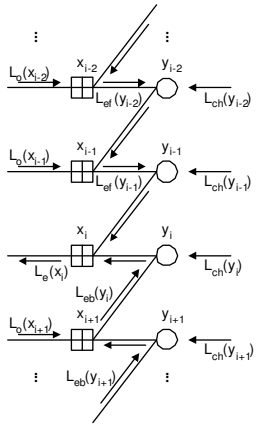

Fig. 2 message flow in the code graph of the accumulator

and finally passed to the outer subdecoder. Refer to [7][8] for more information about the serial update, initial values, and boundary conditions.

3.2. Parallel Accumulator Update

In the context of Repeat-Accumulate codes Jin [10] proposed a simple parallel decoding scheme that replaces the serial message flow in the accumulator with only little compromise in performance. The approach works as follows. Instead of performing one forward and one backward recursion and updating the outgoing messages of each node serially, the messages are updated in parallel at the same time by using the values of the previous iteration. Mathematically, this can be expressed as

$$L_{ef}^{(\mu)}(y_i) = L_o^{(\mu-1)}(x_i) \boxplus \left[L_{ch}(y_{i-1}) + L_{ef}^{(\mu-1)}(y_{i-1}) \right] \quad (5)$$

$$L_{eb}^{(\mu)}(y_i) = L_o^{(\mu-1)}(x_{i+1}) \boxplus \left[L_{ch}(y_{i+1}) + L_{eb}^{(\mu-1)}(y_{i+1}) \right]. \quad (6)$$

According to [8] this parallelization of the relations in (1) and (2) influences the convergence of the algorithm and the error correcting capability of the code only marginally for sufficient iterations. But this algorithm represents no longer an optimum decoder for the accumulator. This simplification will have an impact on the convergence of the iterative decoder and the global decoding result of the complete word. It is the aim of the next section to study this issue for several serial concatenated codes with an inner accumulator.

4. SIMULATION RESULTS FOR SOME SERIAL CONCATENATED CODES

We investigated the performance of RA, ARA and PA codes. For each code two simulations were performed using the same structure of the code graph, the same parameters, and the same randomly generated interleaver pattern(s). They only differed in the chosen update mode for

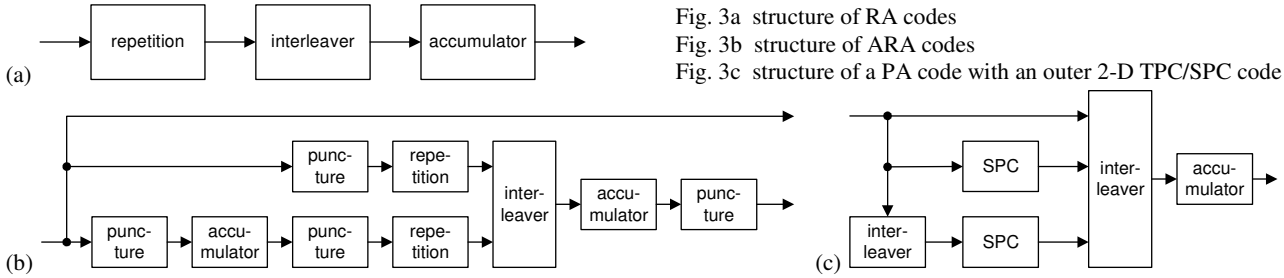

Fig. 3a structure of RA codes
Fig. 3b structure of ARA codes
Fig. 3c structure of a PA code with an outer 2-D TPC/SPC code

the decoding of the accumulator(s). The received words were decoded iteratively using message-passing. The algorithm was implemented with consecutively working sub-decoders and information exchange after the decoding of every component code. The transmission was modeled using a channel with Additive White Gaussian Noise (AWGN) with Binary Phase Shift Keying (BPSK) modulation. All applied interleaver patterns were generated by an S-Random-Interleaver. For the simulation the block- and the bit-error rate were evaluated after every iteration up to a maximum number of 30 iterations. Only errors in the information word were considered to identify the number of block and bit errors.

4.1. Repeat-Accumulate Codes

The Repeat-Accumulate codes (RA) were defined by Divsalar, Jin, and McEliece in 1998 [1]. Fig. 3a shows their structure in principle. Fig. 4 and Fig. 5 present the simulation results for a non-systematic RA code of rate 1/3 with information block length k=1024 and repetition 3 for the serial and the parallel update mode.

As expected, the block-error rates decrease with more iterations for both update modes. The serial update mode, which processes the information of a higher number of nodes per iteration, yields faster convergence than its approximation, the parallel update mode. But in some cases,

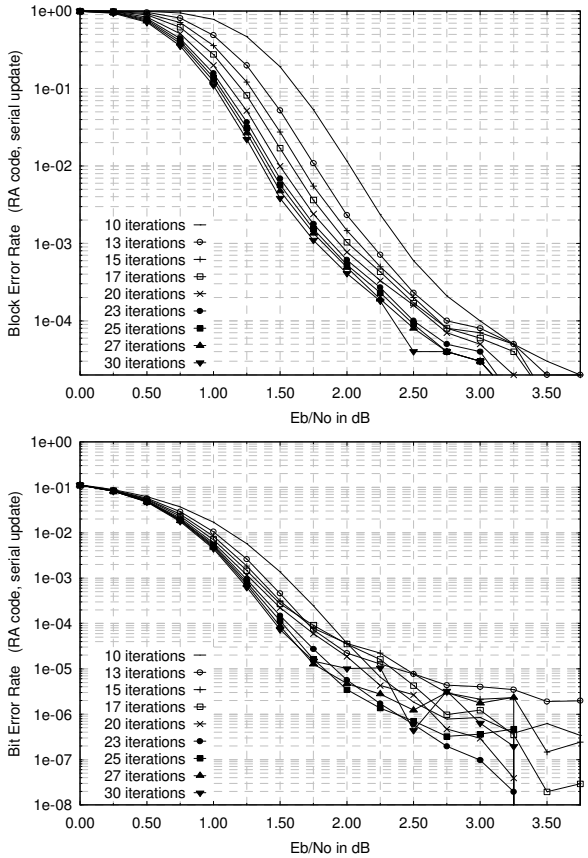

Fig. 4 RA codes: block- and bit-error rates for the serial update mode

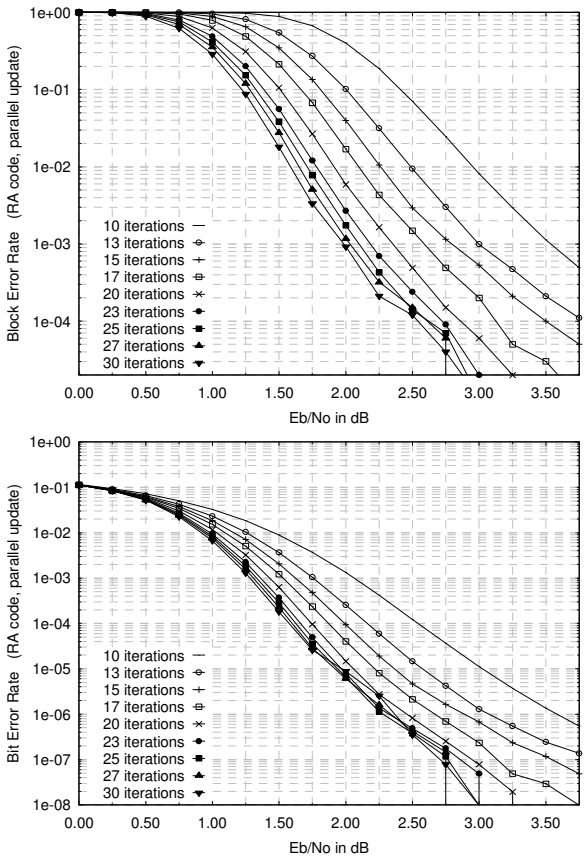

Fig. 5 RA codes: block- and bit-error rates for the parallel update mode

when after several iterations only a little fraction of all words is still incorrect, the following effect can be observed. Especially for the serial update mode, a continued iterative decoding of these words generates more and more bit errors. The not converging decoding process of these few words can cause a dramatically increase of the bit-error rate for a higher number of iterations. Since the serial update mode is more susceptible to this problem, it can be finally outperformed by the parallel update mode. This effect can be observed similarly for the min-sum algorithm [8], a version of the message-passing algorithm with an approximated operation at the check nodes. Fig. 6 shows the block- and bit-error rates with the number of iterations for a signal-to-noise-ratio $E_b/N_0 = 2.75$dB and both update modes.

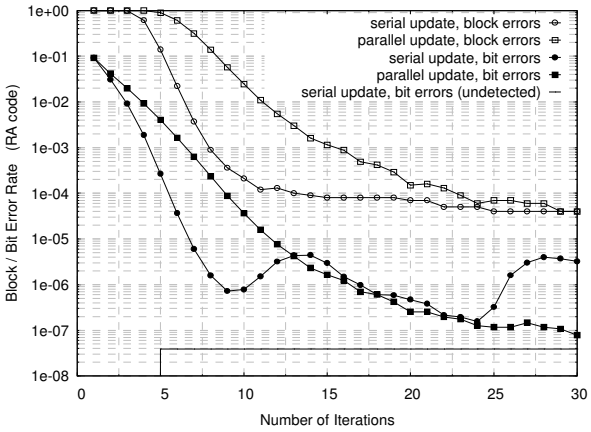

Fig. 6 RA codes: comparison of block- and bit-error rates for both update modes

4.2. Accumulate-Repeat-Accumulate Codes

Accumulate-Repeat-Accumulate codes (ARA), whose structure is depicted in Fig. 3b, were first proposed by Abbasfar, Divsalar, and Kung [3]. They can be viewed as precoded Repeat-Accumulate codes with puncturing. For detailed information and puncture patterns it is referred to [3]. Simulations were done for a systematic ARA code with information block length k=1000, repetition 3, and the puncture patterns proposed in [3] for code rate equal to 2/3. The resulting bit-error rates are presented in Fig. 7. The effect of the accumulator decoding mode described in the context of RA codes can be similarly observed for ARA codes. However, the differences between the two modes are not as large as for the RA codes.

4.3. Product-Accumulate Codes

Product-Accumulate codes (PA) [4], sometimes also denoted as Parity-Accumulate codes, are a class of interleaved serial concatenated codes, where the outer code is a parallel concatenation of two Single Parity-Check (SPC) codes. According to their structure (cf. Fig. 3c) the check nodes can be divided into two groups to determine the update order in the outer subdecoder. Every SPC code corresponds to a cycle-free part of the Tanner graph and can be decoded separately. Simulations were performed for a non-systematic PA code with information block length k=1000. Every block was divided into 500 parts of length t=2 to calculate the single parity checks. Hence, the resulting code rate was equal to 1/2. The bit-error rates for both update modes show a similar behavior as those of RA codes (cf. Fig. 8).

5. DISCUSSION AND CONCLUSION

Message-passing is an efficient decoding algorithm for serial concatenated codes. It yields good performance for these codes while requiring a relatively low decoding complexity. It was shown that for several serial concatenated codes with an inner accumulator the parallel update mode for the accumulator decoding leads to a more steadily convergence of the bit-error rates. Despite its slower convergence, iterative decoding with the parallel update mode can finally lead to a lower bit-error rate than the serial update mode. This is a remarkable observation, since the parallel update mode was originally introduced as an approximation for the serial update mode, which represents the optimum decoding for the inner accumulator.

The chosen update mode determines the number of nodes considered in the updating process at the nodes in every iteration. Therefore, it influences how many iterations the neighborhood of each node is tree-like and the decoding process cycle-free, i.e. optimal. The probability of very short cycles in a code graph with a randomly generated interleaver pattern extremely decreases for long block lengths. Hence, this effect mainly occurs for short or moderate block lengths.

The result of the iterative decoding process can be controlled by the constraints given in the code graph. It can be examined whether the parity sums at all check nodes are satisfied by the calculated bits. The increasing number of bit errors can be ascribed to only a few words whose decoding process does not converge. These words can be detected by a following control of all check nodes, but this involves further decoding complexity.

Besides the bit-error rates for both update modes Fig. 6 shows for the serial update mode how many bit errors remain undetected after this control process by the check nodes. This illustrates that for a decoding process with a fixed number of iterations, e.g. 30, the parallel update mode yields almost the same performance like iterative decoding with serial update and following detection but with less decoding complexity. Implementing an iterative decoder for such serial concatenated codes with short or moderate block lengths this fact can be utilized.

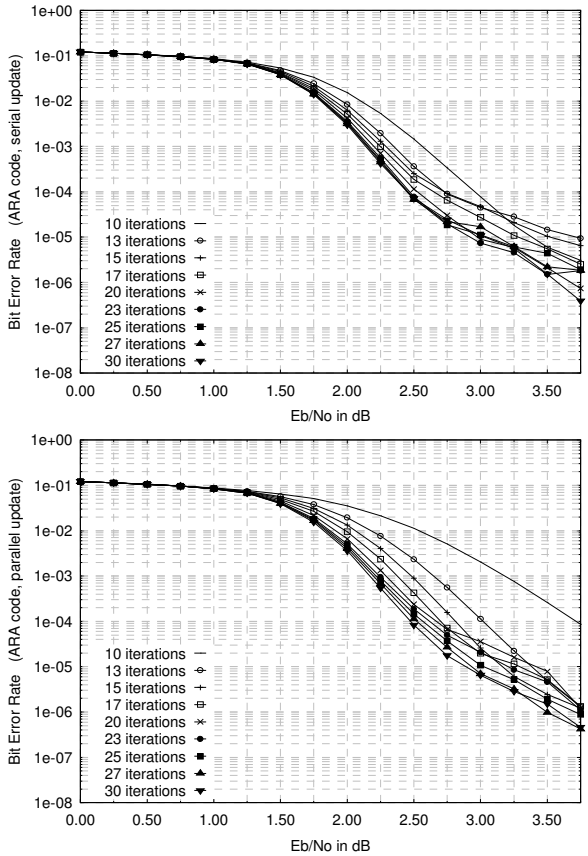

Fig. 7 ARA codes: bit-error rates for both update modes

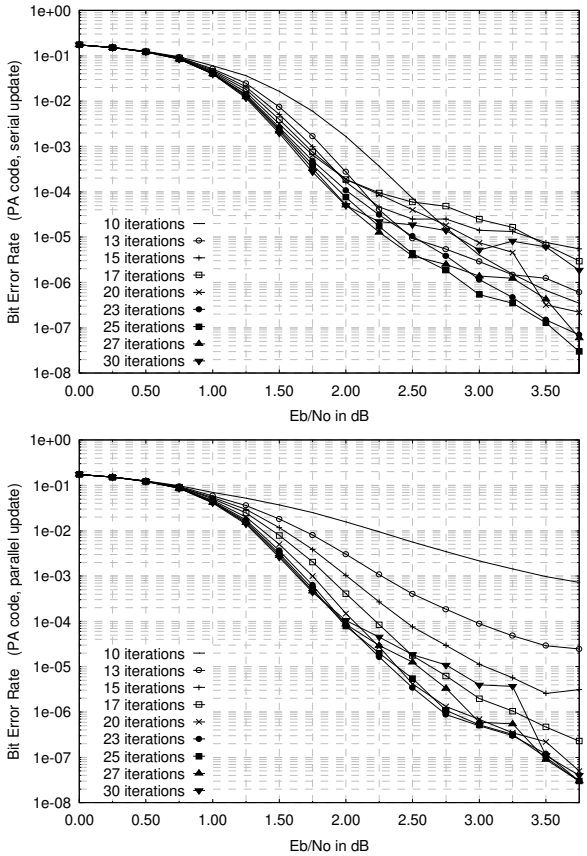

Fig. 8 PA codes: bit-error rates for both update modes

6. REFERENCES

[1] D. Divsalar, H. Jin, and R. J. McEliece, "Coding Theorems for 'Turbo-Like' Codes," 36th Allerton Conf. on Communication, Control and Computing, September 1998, pp. 201-210.

[2] H. Jin, A. Khandekar, and R. J. McEliece, "Irregular Repeat-Accumulate Codes," 2nd Int. Symp. Turbo Codes and Related Topics, Brest, France, September 2000.

[3] A. Abbasfar, D. Divsalar, and Y. Kung, "Accumulate Repeat Accumulate Codes," ISIT 2004, Chicago, USA, June 27 - July 2, 2004.

[4] K. R. Narayanan, J. Li, and C. N. Georghiades, "Product Accumulate Codes: Properties and Performance," ITW2001, Cairns, Australia, September 2-7, 2001.

[5] A. Khandekar, "Graph-based Codes and Iterative Decoding," Ph.D. dissertation, California Institute of Technology, Pasadena, California, USA, June 2002.

[6] T. Richardson, and R. Urbanke, "The Renaissance of Gallager's Low-Density Parity-Check Codes," IEEE Communications Magazine, August 2003, pp. 126-131.

[7] J. Li, K. R. Narayanan, and C. N. Georghiades, "An Efficient Decoding Algorithm for Cycle-free Convolutional Codes and its Applications," IEEE Global Communications Conf., San Antonio, Texas, USA, Nov. 2001, pp. 1063-1067.

[8] J. Li, K. R. Narayanan, and C. N. Georghiades, "Product Accumulate Codes: A Class of Codes With Near-Capacity Performance and Low Decoding Complexity," IEEE Trans. Inform. Theory, vol. 50, no. 1, January 2004, pp. 31-45.

[9] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," IEEE Trans. Inform. Theory, vol. 42, March 1996, pp. 429-445.

[10] H. Jin, "Analysis and Design of Turbo-like Codes," Ph.D. Thesis, California Institute of Technology, Pasadena, California, USA, May 2001.

[11] A. Panagos, "Design and Evaluation of High Performance Error-Correcting Codes," November 2003. http://www.ou.edu/idp/teamlearning/docs/documentation1.pdf