AN EFFICIENT DECODER SCHEME FOR DOUBLE BINARY CIRCULAR TURBO CODES

C. Zhan[1], T. Arslan[1], A. T. Erdogan[1], S. MacDougall[2]

[1] School of Electronics and Engineering, University of Edinburgh, Edinburgh, UK

[2] Freescale Semiconductor U.K. Limited East Kilbride, UK

ABSTRACT

Recently, double binary circular Turbo code has received tremendous attention. Due to its better error-correcting capability than classical Turbo code, it has commenced practical applications in current communication standards, such as DVB-RSC and IEEE 802.16 (Wimax). However current decoding schemes will incur a huge computation complexity. In this paper, authors present a novel decoding scheme for double binary circular Turbo codes, which will not only reduce the computation complexity, but also give at least 0.5dB performance gain compared with current decoding schemes.

1. INTRODUCTION

Turbo code is first presented to the coding community in 1993, which has received a big success in communication world. In recent years, the researchers have shown that the non-binary circular Turbo code can offer many advantages [1] in comparison with the classical single binary Turbo code, and it has replaced classical Turbo code as one of the channel codes in new communication standards, such as DVB-RSC and Wimax.

However, due to its non-binary and circular properties, the decoder design is much more complex than classical Turbo decoder [2]. Authors in [3] [4] [5] mentioned the modified MAP algorithm or BCJR algorithm to decode the double binary Turbo code which must compute three Log-likelihood ratio (LLR) values. In addition, a pre-decoder (prologue) was introduced to estimate the initial forward and backward trellis state. Apparently, these proposed decoding schemes will need at least 3 times more complexity compared with classical Turbo decode, and make them impractical for the real system implementations.

In this paper, the authors address a novel decoding scheme for double binary circular Turbo code without LLR computation and pre-decoder procedure, which can decrease the computation complexity and increase the decoding performance by 0.5dB. Different MAP approximations based on this decoding scheme are also compared in this paper.

The paper is organized as follows: section 2 describes the encoder architecture of double binary circular Turbo code. Decoder procedure will be presented in section 3. The performance simulation and comparison are addressed in section 4. Finally, conclusion is accessed in section 5.

2. ENCODING SCHEME

Double binary Turbo encoder is a parallel concatenation of two double binary recursive systematic convolutional *(RSC)* codes, which is shown in Figure 1. Data couples (A, B), rather than single bit streams, are fed to the encoder twice, in a normal order and in an interleaved order. For each data couple, the encoded codeword involves 2 systematic bits which are the copy of input pair and 4 parity bits (Y1, W1, Y2 and W2) for the normal order and the interleaved order, respectively.

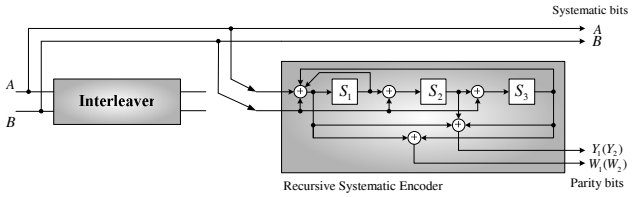

Figure 1 Double binary circular Turbo encoder

2.1 Circular Turbo code

Circular coding, called "tailing-biting" technique [6], can ensure that the ending trellis state is equal to the initial trellis state, which is called circular state S_C. Thus the trellis structure can be seemed as a circle. Unlike the classical Turbo code which usees redundant tail bits to drive the trellis to the zero-state, circular Turbo code does not need tail bits. So there is no rate loss and spectral efficiency of the transmission is not reduced.

Since the value of circular state S_C depends on the contents of the sequence to be encoded, a pre-encoding operation is required to determine the circular state for normal order and interleaved order, respectively. To perform a complete encoding operation of the data sequence, the data sequence has to be encoded four times instead of twice in the classical Turbo encoder. But this is not a real problem, as the encoding operation can be performed at a much higher frequency compared with the data rate.

3. DECODER PROCEDURE

3.1 Decoder structure

The architecture of the decoder is shown in Figure 2. The Turbo decoder calls for two component MAP decoders and operates in an iterative manner, where the log domain extrinsic probabilities output $\ln P_{out}^{ex}(u_k \mid y)$ from one MAP decoder, will be passed to another as the priori probabilities of received codeword. In our proposed decoder scheme, two pairs of forward metric and backward metrics will also be fed back to next iterative operation. The details of the proposed decoder scheme will be described in the following sections.

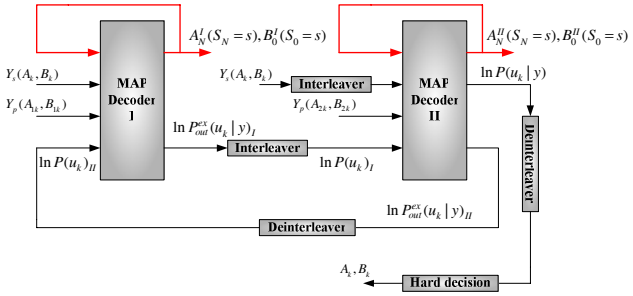

Figure 2 Proposed Turbo decoder architecture

3.2 MAP decoder algorithm

Due to the double binary property, we cannot simply judge original message on one LLR value of a posteriori probabilities as that of the classical Turbo decoder. Authors in [3] [4] and [5] mentioned a modified MAP algorithm or BCJR algorithm which must calculate three LLRs values $L_1 = \ln\left(\frac{P(u_k = (01) \mid y}{P(u_k = (00) \mid y}\right)$, $L_2 = \ln\left(\frac{P(u_k = (10) \mid y}{P(u_k = (00) \mid y}\right)$ and $L_3 = \ln\left(\frac{P(u_k = (11) \mid y}{P(u_k = (00) \mid y}\right)$ to decode double binary Turbo code, and consequently the computational complexity is increased. But if carefully consider the principle of MAP algorithm, we can find that there is no need to compute the LLR values in double binary Turbo decoder.

As the name implies, the MAP algorithm is used to find the maximum value of posterior possibility, equal to find the maximum value of $P(u_k \mid y)$. For the double binary Turbo decoder, we can compute four probabilities $P(u_k = (0,0) \mid y)$, $P(u_k = (0,1) \mid y)$, $P(u_k = (1,0) \mid y)$ and $P(u_k = (1,1) \mid y)$ directly, then select the maximum one as the decoded data.

Since we do not consider LLR, new equations will be deduced to calculate posteriori possibility directly. To simplify the implementation, the operations are transferred into the logarithmic domain. Log domain posteriori possibility of each data pair can be defined as:

$$\ln P(u_k \mid y) = \ln(\sum \exp(B_k(s) + \Gamma_k(s', s) + A_{k-1}(s'))) \quad (1)$$

$$\text{where} \begin{cases} A_k(s) = \ln(\sum_{all\ s'} \exp(\Gamma_k(s', s) + A_{k-1}(s')) \\ B_{k-1}(s') = \ln(\sum_{all\ s} \exp(\Gamma_k(s', s) + B_k(s)) \\ \Gamma_i(s', s) = \left(\sum_{l=1}^{m+n} x_{kl} \cdot y_{kl}\right) + \ln P(u_k) \end{cases}$$

m is the size of systematic part and n is the size of parity part. X_{kl} and Y_{kl} are the codeword and received noisy signal, respectively.

After every MAP decoder operation, 4 log domain extrinsic information:

$$\ln P_{out}^{ex}(u_k \mid y) = \ln P(u_k \mid y) - \sum_{l=1}^{m} x_{kl} \cdot y_{kl} - \ln P(u_k) \quad (2)$$

will be sent to the other MAP decoder to replace the $\ln P(u_k)$ in equation (1) rather than sending three LLRs.

3.3 MAX* function approximation

However the Log-MAP algorithm described in section 3.2 is not easy to be implemented. There are some techniques to simplify the Log-MAP algorithm by invoking an approximation for the sake of reducing the associated implementation complexity of MAX^* function, where $MAX^*(x, y) = \ln(e^x + e^y)$

3.3.1 Constant-log-MAP algorithm

In Constant-log-MAP algorithm, the MAX^* can be computed by:

$$MAX^*(x, y) = \max(x, y) + \begin{cases} 0 & if \mid y - x \mid > T \\ C & if \mid y - x \mid \le T \end{cases} \quad (3)$$

Where it is shown in [7] that the best parameters are C = 0.5 and T=1.5.

3.3.2 Linear-log-MAP algorithm

The linear-log-MAP algorithm, first introduced in [8], uses the following linear approximation:

$$MAX^*(x, y) = \max(x, y) + \begin{cases} 0 & if \mid y - x \mid > T \\ a(\mid y - x \mid -T) & if \mid y - x \mid \le T \end{cases} \quad (4)$$

In [7] a solution is found by miniming the total squared error between the exact correction function and its linear approximation where a=-0.24904 and T = 2.5068. In our performance simulation, we exploited the parameters provided by Freescale [9].

3.3.3 *Max-log-MAP / Enhanced Max-log-MAP algorithm*

In the Max-log approximation, MAX^* is computed by:

$$MAX^* = \max(x, y) \quad (5)$$

Due to the application of the Max-Log approximation in metric calculations, the extrinsic information is less reliable than the original MAP algorithm. Therefore, Max-Log-MAP performance can be improved by multiplying the extrinsic information with a coefficient smaller than 1.0, typically around 0.75, to revise the errors in decoding procedure, which is called Enhanced Max-log-MAP algorithm in [10]. The performance comparisons of these approximations will be provided in section 5.

3.4 Circular decoder

In the classical Turbo decoder, since we have known that the trellis starts at zero state and ends at zero state, the initial conditions for forward metric and backward metric of the MAP decoding procedure can be defined as:

$$\begin{aligned} &A_0(S_0 = 0) = 0 \\ &B_N(S_N = 0) = 0 \\ &A_0(S_0 = s) = -\infty \\ &B_N(S_N = s) = -\infty \quad for\ all\ \ s \neq 0 \end{aligned} \quad (6)$$

But for circular Turbo code, we only know the final trellis state is equal to the initial trellis state, but do not know exactly which state is the circular state (S_C) used by the encoder. This raises a problem to initialize the forward and backward metrics. Some researchers suggested to include a pre-decoder (prologue) estimate the initial state [3] [4]. Obviously, this scheme will increase the latency and computational complexity.

Since iterative decoder is adopted in the real system, we propose a new algorithm which does not need the use of pre-decoder. In the proposed algorithm, at the beginning of the decoding process, all states will be assumed to be equiprobable. Apparently, some side errors may be produced by the decoder in the first iteration. However, during the decoder processing, the initialization errors can be removed by the MAP algorithm and the final state probabilities of the forward and backward metrics will be more creditable than the initial status. The most important feature of the proposed decoder scheme is final forward and backward metrics will be feedback to the next iteration as the initial condition for the forward and backward metrics,

$$\begin{aligned} &A_0(S_0 = s) = A'_N(S_N = s) \\ &B_N(S_N = s) = B'_0(S_0 = s) \end{aligned} \quad (7)$$

which means the initial values of the forward and backward metrics are the same with the final metrics of the last iteration.

Although the performance of the proposed algorithm in the first iteration will be worse than the approach proposed in [3] and [4], the decoder will become more and more creditable, as the number of iteration increased,

4 SIMULATION RESULTS

Simulations were performed to illustrate the performance of the proposed decoding scheme and 4 different MAX* function approximations. Two test cases were used, 144 bits per frame and 1920 bits per frame. For both cases, 8 decoder iterations and code rate 1/2 are performed. AWGN channel with BPSK modulation is assumed.

4.1 Feedback verse prologue

In order to illustrate the performance difference between feedback scheme and prologue scheme, 144 bits per frame with Enhanced Max-Log-Map decoding algorithm is used as a test case. From Figure 3, we can easily see that the decoding scheme with forward and backward metric feedback can get 0.5dB performance gain compared with using prologue to initialize forward and backward metrics.

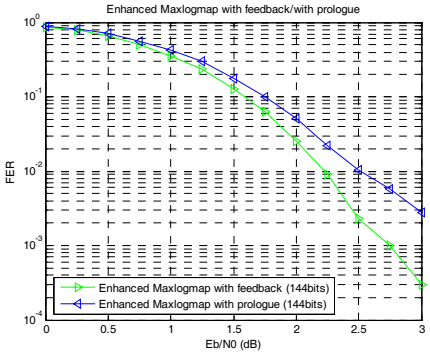

Figure 3 FER of decoder scheme with feedback and prologue

4.2 FER performance

The frame error rate (FER) simulation of 4 different MAX* function approximations and Log-Map algorithm for 144 bits per frame and 1920 bits per frame are shown in Figure 4 and Figure 5, respectively. All of the implementations exploit forward and backward metrics feedback and avoid LLR computations.

From the figures 4 and 5, we can see that FER performance of Enhanced Max-Log-Map and linear-Log-Map algorithms are very close to that of Log-Map. However, in terms of the computational complexity, presented in section 3.3, Enhanced Max-Log-Map will take less hardware resources than linear Log-Map.

4.3 Average number of iterations

Figures 6 and 7 present average number of iterations required for each approximation for frame size 144bits and frame size 1920bits, respectively. With the increasing of

Eb/N0, Max-log-Map and Enhanced Max-log-Map require less decoder iteration than the other algorithms.

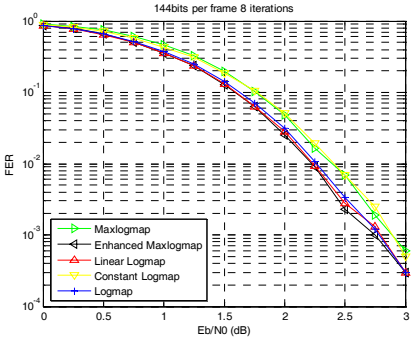

Figure 4 144bits frame sample after 8 iterations Turbo decoding

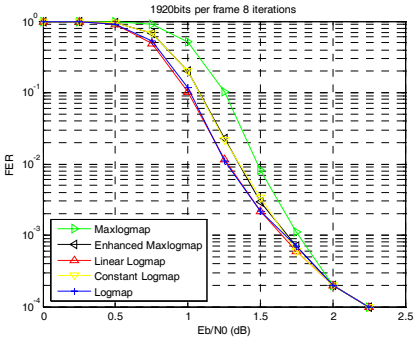

Figure 5 1920bits frame sample after 8 iterations Turbo decoding

From these simulations, it can be concluded that Enhanced Max-Log-Map algorithm can achieve the best trade off between performance and computational complexity, which is recommended in hardware implementation by the authors.

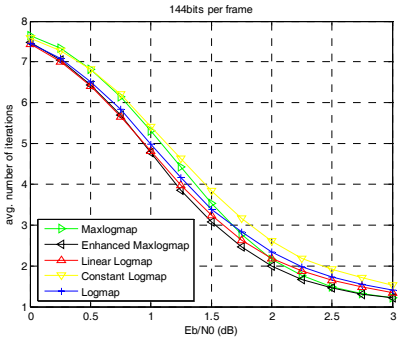

Figure 6 Average iteration required for 144bits frame

CONCLUSION

This paper has discussed a new decoder scheme for double binary circular Turbo code, which requires less computational complexity and achieves better decoding performance than the current decoder schemes for double binary circular Turbo code. Performance and computational complexity of Log-Map algorithm and different MAX* function approximations based on the novel decoding scheme were compared in this paper. It was shown that Enhanced Max-Log-Map algorithm with forward and backward metric feedback can achieve the best trade-off between performance and computation complexity.

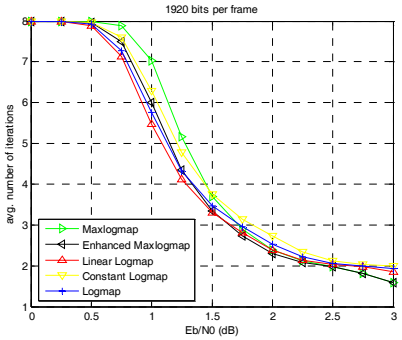

Figure 7 Average iteration required for 1920bits frame

REFERENCES

[1] C. Berrou, M. Jezequel, C. Douillard and S.Kerouedan, "The advantages of non-binary Turbo codes", *Proc. IEEE information Theory Workshop*, pp. 61-63, Sept. 2001

[2] Woodard, J.P. and Hanzo, L., "Comparative study of Turbo decoding techniques: an overview", IEEE Transactions on Vehicular Technology, Volume 49, Issue 6, Nov. 2000 Page(s):2208 - 2233

[3] D. Giancristofaro and A. Bartolazzi, "Novel DVB_RSC standard Turbo code: details and performances of a decoding algorithm", *ESA conference*, 2001

[4] C. Douillard, M. Jezequel and C. Berrou, "The Turbo code Standard for DVB-RCS", *2nd International Symposium on Turbo Codes & Related Topics*, Brest, France, Sept. 2000, pp. 535 – 538.

[5] Yingzi Gao, Soleymani, M.R.,"Triple-binary circular recursive systematic convolutional Turbo codes", *the 5th International Symposium on Wireless Personal Multimedia Communications*, Volume 3, 27-30 Oct. 2002 Page(s):951 - 955 vol.3

[6] Weiss, C., Bettstetter, C. and Riedel, S., "Code construction and decoding of parallel concatenated tail-biting codes", IEEE Transactions on Information Theory, Volume 47, Issue 1, Jan. 2001 Page(s):366 – 386

[7] M.C. Valenti and J. Sun, "The UMTS Turbo Code and an Efficient Decoder Implementation Suitable for Software-Defined Radios", International Journal of Wireless Information Networks, 2001

[8] J.F. Cheng and T. Ottosson, "Linearly approximated log-MAP algorithms for Turbo coding", Proc. of IEEE VTC, May 2000

[9] Freescale Semiconductor, Inc. www.freescale.com

[10] J. Vogt and A. Finger, "Improving the MAX-Log-MAP Turbo decoder", Electronics letters, Vol.36 No.23, pth November 2000