Indian Streams Research Journal
Volume 2, Issue. 7, Aug 2012
ISSN:-2230-7850

Available online at www.isrj.net

ORIGINAL ARTICLE

The Turbo Code and an Efficient Decoder Implementation using MAP Algorithm for Software Defined Radios

Rupesh Singh and Nidhi Singh

(Associate Professor)
Research scholar CMJ University
SriGanganagar Engineering College,SriGanganagar.

Abstract:

This paper provides a description of the turbo code used by the UMTS third-generation cellular standard, as standardized by the Third-Generation Partnership Project (3GPP), and proposes an efficient decoder suitable for insertion into software-defined radio or for use in computer simulations. Because the decoder is implemented in software, rather than hardware, singleprecision floating-point arithmetic is assumed and a variable number of decoder iterations is not only possible but desirable. The well-known log-MAP decoding algorithm with detailed analysis is proposed and the simulation results are shown.

INTRODUCTION:

Due to their near Shannon-capacity [1] performance, turbo codes have received a considerable amount of attention since their introduction [2]. They are particularly attractive for cellular communication systems and have been included in the specifications for both the WCDMA (UMTS) and CDMA2000 third-generation cellular standards. At this time, the reasons for the superior performance of turbo codes [3,4] and the associated decoding algorithm [5,6] are, for the most part, understood.

The purpose of this paper is to explain the phenomenal performance of turbo codes and to derive the decoding algorithm. Also the purpose is to clearly explain an efficient decoding algorithm suitable for immediate implementation in a software radio receiver. In order to provide a concrete example, the discussion is limited to the turbo code used by the Universal Mobile Telecommunications System (UMTS) specification, as standardized by the Third-Generation Partnership Project (3GPP) [7]. The decoding algorithm is based on the log-MAP algorithm [8], although many parts of the algorithm have been simplified without any loss in performance.

Some critical implementation issues are discussed, in particular the decoding iterations. Simple, but effective, solutions for MAP Algorithm are proposed and illustrated through computer simulations. In the description of the algorithm, we have assumed that the reader has a working knowledge of the Viterbi algorithm [9]. Information on the Viterbi algorithm can be found in a tutorial paper by Forney [10].

2) The UMTS Turbo Encoder and Decoder:

As shown in Fig. 1, the UMTS turbo encoder is composed of two constraint length 4 recursive systematic convolutional (RSC) encoders concatenated in parallel [12]. The feedforward generator is 15 and the feedback generator is 13, both in octal. The number of data bits at the input of the turbo encoder is K. Data is encoded by the first (i.e., upper) encoder in its natural order and by the second (i.e., lower) encoder after being interleaved. At first, the two switches are in the up position. The interleaver is a matrix with 5, 10, or 20 rows and between 8 and 256 columns (inclusive), depending on the size of the input word. Data is read into the interleaver in a rowwise fashion (with the first data bit placed in the upper-left position of the

Please cite this Article as : Rupesh Singh and Nidhi Singh , The Turbo Code and an Efficient Decoder Implementation... : Indian Streams Research Journal (Aug. ; 2012)

matrix). Intrarow permutations are performed on each row of the matrix in accordance with a rather complicated algorithm, which is fully described in the specification [11].

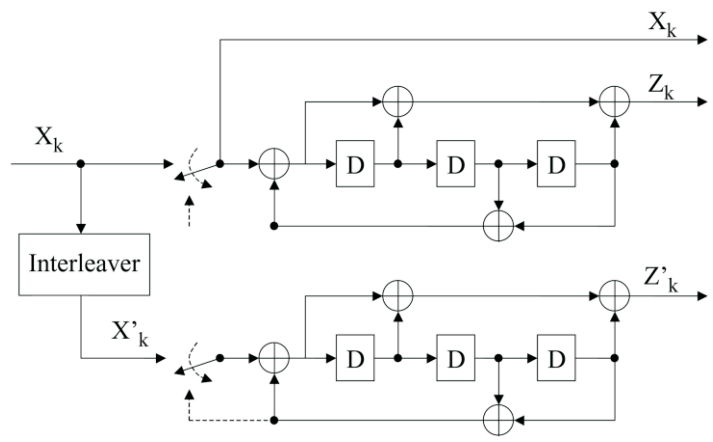


Fig.1

The data bits are transmitted together with the parity bits generated by the two encoders (the systematic output of the lower encoder is not used and thus not shown in the diagram). Thus, the overall code rate of the encoder is rate 1/3, not including the tail bits. Zk is the parity output from the upper (uninterleaved) encoder, and Z'k is the parity output from the lower (interleaved) encoder. After the K data bits have been encoded, the trellises of both encoders are forced back to the all-zeros state by the proper selection of tail bits. Unlike conventional convolutional codes, which can always be terminated with a tail of zeros, the tail bits of an RSC will de- pend on the state of the encoder. Because the states of the two RSC encoders will usually be different after the data has been encoded, the tails for each encoder must be separately calculated and transmitted. The tail bits are generated for each encoder by throwing the two switches into the down position, thus causing the inputs to the two encoders to be indicated by the dotted lines.

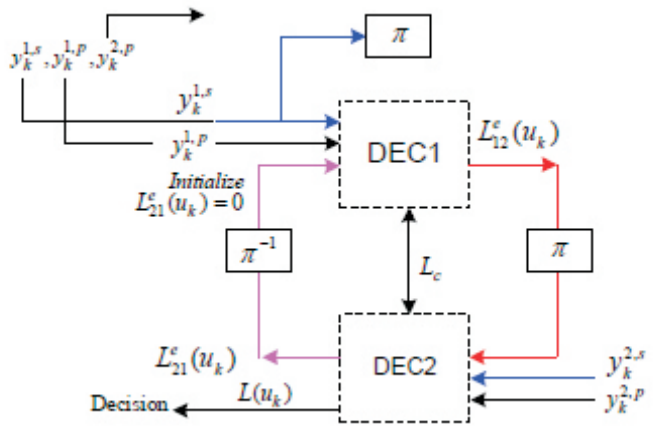
Decoder: For each time tick k, decoding is done by calculating the L-values of a +1 bit. If it is positive, the decision is in favor of a +1. Calculation of the L-values or L(uk) is quite a complex process. The main equation used to calculate the L-value is this.

$$L(u_k) = \left[L^e(u_k) + Lc \cdot y_k^{1,s}\right] + \log \frac{\sum_{u^+} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\beta}_k(s) \cdot \gamma_k^e(s',s)}{\sum_{u^-} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\beta}_k(s) \cdot \gamma_k^e(s',s)}$$

From this computation if L(uk) is positive, then uk is equal to +1. The first term, is the a-priori value from Decoder 2. This is the L-value of the a-priori probability of the bit in question. At first, the decoder has no idea what it is. A good guess is to assume it is 0.5. The second term, is computed by multiplying the systematic information with Lc. This value is the channel L-value and gives an indication of channel SNR. The third big term with all kinds of squiggly items is the a-posteriori probability.

This number is calculated for each trellis segment. Remember that we can only have one result for a trellis section, a +1 or -1. The L(uk) calculation tells us what that number is, The big equation can be written simply as sum of three pieces of information. L-apriori – This is our initial guess about a +1 bit in first iteration L-Channel - This is related to channel SNR and systematic bit and is equal to Le – the computed information in each iteration is called the a-posteriori L-value. The L-channel value does not change from iteration to iteration since is it given by . Neither the Lc nor the systematic bit changes from iteration to iteration, So lets called it K. The only two items that change are the a-priori and a-posteriori L-values. A-priori value goes in, it is used to compute the new a-posteriori probability and then it can be used to compute L(uk) or is passed to the next decoder. Although in the example we compute L(uk) each time, during actual decoding this is not done. Only a-posteriori metric is computed and decoders and keep doing this either a fixed number of times or until it converges. L-posteriori is also called Extrinsic information.

A branch metric is the correlation of the received signal with its trellis values. In a nutshell, if the received values are the same sign as the coded bits, then the correlation will be high. For each decoder, there are full transition branch metrics which incorporate, the systematic and the parity bits and partial branch metrics which incorporate only the parity bits.



Fif.2 Iteration cycle of MAP Turbo Decoding

1. Computing full branch metrics:
The full branch metric is given by the equation

$$\gamma(s',s) \propto \exp\left[\frac{1}{2}\cdot L^e(c_k^1)\cdot c_k^1 + Lc\cdot\frac{1}{2}\cdot y_k^{1,s}\cdot c_k^1\right]\exp\left[\sum_{i=2}^{q}\left(Lc\cdot\frac{1}{2}\cdot y_k^{i,p}\cdot c_k^i\right)\right]$$

2. Compute partial branch metrics:
These values are based only on parity bits. We will use these to compute the total extrinsic L-values. The equation for calculating the partial metrics is given by:

$$\gamma^*(s',s) \triangleq \exp\left[\sum_{i=2}^{q}\left(Lc\cdot\frac{1}{2}\cdot y_k^{i,p}\cdot c_k^i\right)\right]$$

3. Calculation of forward metrics Encoding always starts in state 1. So we assume that signal can only be in state 1 at time k =0. The initial value of the forward metric is set at 1.0. The other three states are given value of 0.0. Meaning, these are not possible states. Thereafter the forward metrics are computed recursively for each state (not branches). The equation is:

$$\tilde{\alpha}_k(s) = \frac{\sum_{s'}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)}{\sum_{s}\sum_{s'}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)}$$

4. Computing backward state metrics:

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s}\tilde{\beta}_k(s)\gamma_k(s',s)}{\sum_{s}\sum_{s'}\tilde{\alpha}_{k-2}(s')\gamma_{k-1}(s',s)}$$

We assume that the signal will end in state 1. (Tail bits are added to force it end in state 1.) The ending state is always s1, so it gets a value of 1.0 just as the forward state metric. The rest of the three states at k = 7 are given a backward state value of zero.

5. Computing Extrinsic L-values: Now we will compute the Extrinsic L-value. Which are given by the product of the all three metrics.

$$\sigma_k(s) = \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s',s) \cdot \tilde{\beta}_k(s)$$

To compute the L-value we need the ratio of these numbers for the +1 and -1 branches. For that we add the top four branch metrics and divide by the sum of the bottom four branch metrics to satisfy the following equation:

$$\log \frac{\sum_{u^+} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\beta}_k(s) \cdot \gamma_k^e(s',s)}{\sum_{u^-} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\beta}_k(s) \cdot \gamma_k^e(s',s)}$$

Take the natural log of the ratio. This is the extrinsic value output for this iteration. These values after interleaving go to Decoder 2 and become it's a-priori probabilities. Normally the decoder at this point would stop because it has done its job of computing the extrinsic value. But we will compute the L-value of the bit to show you how the bit decisions can be made at any iteration.

3) SIMULATION RESULTS

The BER performance of the Simplified-Log-MAP algorithm is compared to that of the MAP, Log-MAP, and Max- Log-MAP algorithms. The simulation results are for a Turbo code with 1/2 code rate, N = 1024, m = 3, and the feedback and feedforward generator polynomials equal to I 5oct and I 7oct respectively. Three iterations were used for the simulations. As we can see from the results, the Log-MAP decoding algorithm has similar performance to the MAP algorithm. The performance loss for the MAX-Log-MAP compared to the MAP algorithm is from 0.2 dB. The SNR requirement for a given BER is higher at larger constellation sizes, therefore, the approximation of the logarithm has more significant effect on the BER performance of the MAX-Log-MAP algorithm.

Operation	MAP	Max-Log-MAP	Simplified-Log-MAP	Log-MAP
Maximization	2M - 1	5M - 2	3M - 2	4M - 4
Addition	4M	10M - 2	12M - 2	14M - 4
Multiplication	10M	0	0	0
Table Look-up	0	0	2M	4M - 2
Total Operation	14M	10M - 2	12M - 2	14M - 4
Total Ops. for T1 in Mops.	345.86	240.86	290.27	333.5

Table:1 Complexity Comparison obetween different MAP Algorithms

The Simplified Log-MAP has a negligible performance degradation compared to the MAP algorithm for QPSK constellation, while the performance loss is approximately 0.1 dB. It can be concluded from the above results that Simplified Log-MAP algorithm together with the new hardware implementation are an appropriate choices for implementing Turbo decoders in practice without any significant loss in performance.

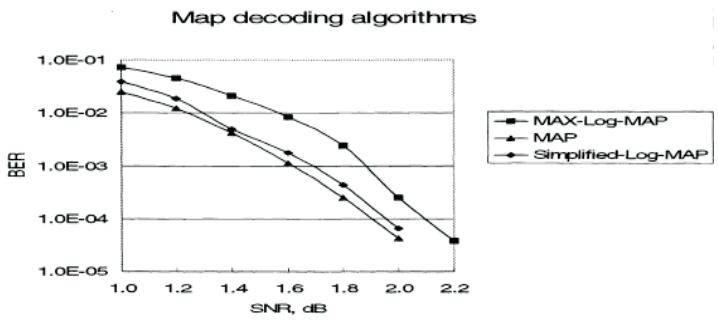


Fig. BER performance of MAP-based decoding algorithms

REFERENCES:

[1] SHANNON, C. E.: 'A mathematical theory of communication', Bell Syst. Tech. J., July and October 1948, 27, pp.379–423 and 623–656.
[2] C. Berrou, A. Glavieux, and P. Thitimasjshima, Near Shannon limit error-correcting coding and decoding: Turbo-codes(1), Proc.IEEE Int. Conf. on Commun. (Geneva, Switzerland), pp. 1064–1070, May 1993.
[3] S. Benedetto and G. Montorsi, Unveiling turbo codes: Some results on parallel concatenated coding schemes, IEEE Trans. Inform.Theory, Vol. 42, pp. 409–428, Mar. 1996.
[4] L. C. Perez, J. Seghers, and D. J. Costello, A distance spectrum interpretation of turbo codes, IEEE Trans. Inform. Theory, Vol. 42, pp. 1698–1708, Nov. 1996.
[5] D. Divsalar, S. Dolinar, and F. Pollara, Iterative turbo decoder analysis based on density evolution, IEEE J. Select. Areas Commun., Vol. 19, pp. 891–907, May 2001.
[6] S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, IEEE Trans. Commun., Vol. 49, pp. 1727– 1737, Oct. 2001.
[7] European Telecommunications Standards Institute, Universal mobile telecommunications system (UMTS): Multiplexing and channel coding (FDD), 3GPP TS 125.212 version 3.4.0, pp. 14–20, Sept. 23, 2000.
[8] P. Robertson, P. Hoeher, and E. Villebrun, Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding, European Trans. on Telecommun., Vol. 8, pp. 119–125, Mar./Apr. 1997.
[9] A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding Algorithm, IEEE Trans. Inform. Theory, Vol. 13, pp. 260–269, Apr. 1967.
[10] G. D. Forney, The Viterbi algorithm, Proc. IEEE, Vol. 61, pp.268–278, Mar. 1973.
[11] European Telecommunications Standards Institute, Universal mobile telecommunications system (UMTS): Multiplexing and channel coding (FDD), 3GPP TS 125.212 version 3.4.0, pp. 14–20, Sept. 23, 2000.