Modeling and Simulation of a Turbo Encoder and Decoder for Wireless Communication Systems

Sayantan Choudhury

Abstract

This report presents information on modeling and implementation of turbo codes. Turbo coding is a very powerful error correction technique that has made a tremendous impact on channel coding in the last few years. It outperforms all previously known coding schemes by achieving near Shannon limit error correction using simple component codes and large interleavers. The use of turbo codes enhances the data transmission efficiency in digital communications systems. Turbo codes can also be used to provide a robust error correction solution to combat channel fading.This report gives a brief overview of the encoding and decoding mechanism of the turbo coding scheme, describes the *maximum a posteriori* (MAP) algorithm in detail, models the compuation using synchronous dataflow and discusses a few implementation issues of the MAP algorithm.

I. Introduction

Today's world thrives on information exchange. Hence the need of the day is that the information be protected well enough to be transmitted over a noisy environment. This is achieved by adding redundant bits to the information bit streams. If the purpose of adding redundancy is just to detect errors and inform the sender to retransmit the information, it is known as automatic repeat request (ARQ). Forward error correction (FEC) is another way of adding redundancy to the information bit stream so that errors can be detected and corrected, which prevents the need for retransmission. The price paid for adding such redundancy is a faster transmission rate in order to send the same amount of information bits per unit time, which implies a larger bandwidth requirement. The advantage however, is that the Signal to Noise Ratio (SNR) can be reduced significantly (also referred to as Coding Gain). In wireless systems, one of the most important performance criterion is low transmission power as that can extend battery life and cause lesser co-channel interference.

By increasing the codeword length or the encoder memory and using "good" codes, one can theoretically approach the limiting channel capacity [1]. However finding such a code and implementing an encoder and decoder in real time has been an active area of research for a very long time. Turbo codes [2], [3] are powerful for error correction, which enable reliable communication with bit rates close to Shannon limit for a given bit error probability (BER) [4]. Turbo codes are in fact a parallel concatenation of two recursive systematic convolutional codes. The fundamental difference between convolution codes and turbo codes is that the performance improves in the former case by increasing the constraint length while for turbo codes, the constraint length has a pretty small value. Moreover, it achieves a significant coding gain at lower coding rates. An important factor for achieving this improvement is due to the "soft-input/ soft-output" decoding algorithm to produce soft decisions.

The primary motivation for doing this project was to gain a deeper understanding of turbo codes, learn ways to model it using an appropriate model of computation and to do an optimized implementation of a turbo encoder and decoder. Turbo decoders require a significant number of iterations, which leads to higher latency. Thus efficient implementation of turbo codes in order to meet real time constraints is an active area of

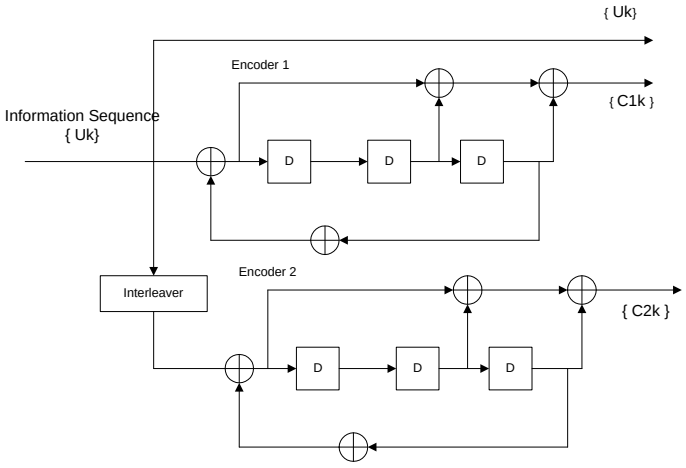


Fig. 1. Turbo Encoder based on CDMA 2000 standard

research. The turbo encoder is described in Section II. Section III gives an overview of the turbo decoder followed by a detailed derivation of the *maximum a posteriori* (MAP) algorithm in Section IV . Modeling and implementation is described in Section V followed by sliding window design in VI.

II. Turbo Encoder

The general structure of a turbo encoder consists of two rate half Recursive Systematic Convolutional (RSC) encoders Encoder 1 and Encoder 2 as shown in Fig 1. The N bit data block is first encoded by Encoder 1. The same data block is also interleaved and encoded by Encoder 2. The main purpose of the interleaver is to randomize bursty error patterns so that it can be correctly decoded. It also helps to increase the minimum distance of the turbo code.

III. Turbo Decoder

An iterative decoding is proposed in [2], [3] which is basically a modification of the Bahl decoding algorithm [5]. The modification is necessary due to the recursive nature of the encoders. The difference in this algorithm from the Viterbi algorithm [6] is that while the former produces hard decisions, this one produces soft decisions. Thus, instead of outputting only 0 or 1, the output range is continuous and is a measure of the log-likelihood ratio of every bit estimate. The iterative feedback scheme is shown in Fig. 2.

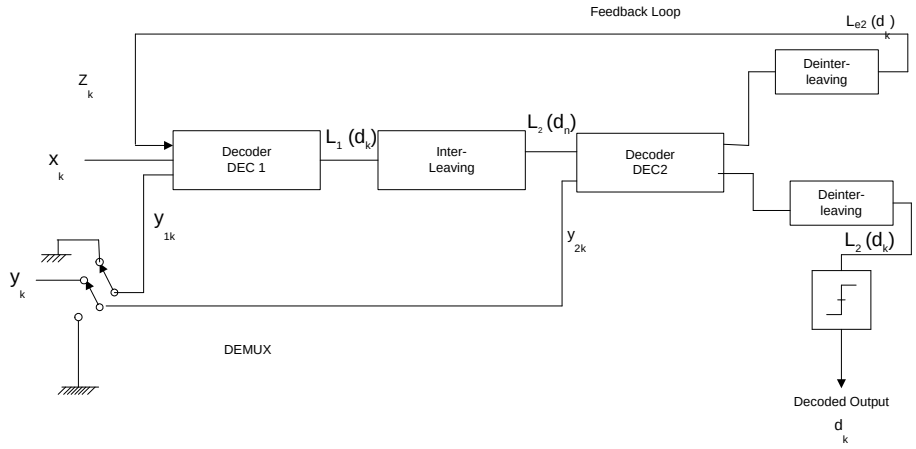


Fig. 2. Turbo Decoder

IV. Map Algorithm

There are two main types of soft decision decoding algorithms which are most commonly used. The first one is a modified Viterbi algorithm called the Soft Output Viterbi Algorithm (SOVA) [7]. The second type of algorithm is the *maximum a posteriori* (MAP) algorithm. The complexity of the MAP algorithm is quite higher than the Viterbi algorithm hence the SOVA is preferred in real time applications due to its lower latency. However, the performance of MAP is about 0.5 dB better than SOVA at lower SNR and high BERs [8]. This is very important for turbo codes since the output BERs from the first stage of iterative decoding is quite high and any improvement at this stage leads to significant overall performance improvements.

For my project, I implemented the log-MAP algorithm which is a simplification of the original MAP algortithm [9]–[11]. The important steps of the MAP algorithm as presented in [12] are as follows:

1. Computation of branch metric δ : The branch metric at time k and state m, is denoted as $\delta_k^{i,m}$. The state at time instant k and the decoded output are represented as S_k and d_k respectively. Thus

$$P(d_k = i, S_k = m, R_k) \stackrel{\Delta}{=} \delta_k^{i,m} \tag{1}$$

and it equates to

$$\delta_k^{i,m} = A_k \pi_k^i \exp\left[\frac{1}{\sigma^2}(x_k u_k^i + y_k v_k^{i,m})\right] \tag{2}$$

where A_k is a constant, π_k^i is defined as $P(d_k = i)$, the *a priori* probability of d_k, σ is the noise variance of the AWGN channel, x_k and y_k are the noisy received data bit and the corresponding noisy parity bit respectively, u_k is the transmitted data bit and v_k is the parity bit. $R_k = (x_k, y_k)$ is the received symbol at time k. Note that v_k depends on the state m since the code has memory.

2. Computation of forward state metric α : The forward state metric at time k and state m, is denoted as α_k^m. Thus for $i = 1, 0$

$$P(R_1^{k-1}|d_k = i, S_k = m, R_k^N) = P(R_1^{k-1}|S_k = m) \stackrel{\Delta}{=} \alpha_k^{i,m} \tag{3}$$

where $R_1^N = (R_1, \ldots, R_k, \ldots, R_N)$ is the demodulator output of the received bit sequence which has been corrupted by channel noise. x_k and y_k are defined as

$$x_k = (2u_k - 1) + p_k \tag{4}$$

$$y_k = (2v_k - 1) + q_k \tag{5}$$

with p_k and q_k being two independent normally distributed random variables with variance σ^2. R_1^N can be written as

$$R_1^N = \{R_1^{k-1}, R_k, R_{k+1}^N\} \tag{6}$$

α_k^m can be recursively calculated using

$$\alpha_k^m = \sum_{j=0}^{1} \alpha_{k-1}^{b(j,m)} \, \delta_{k-1}^{j,b(j,m)} \tag{7}$$

where $b(j, m)$ is the state going backwards in time from state m, via the previous branch corresponding to input j. Equation [7] indicates that the new forward state metric at time k and state m is obtained by summing two weighted state metrics from time $k - 1$.

3. Computation of the backward state metric β : The backward state metric β_k^m, at time k and state m, is described by

$$P(R_{k+1}^N|d_k = i, S_k = m, R_k) = P(R_{k+1}^N|S_{k+1} = f(i, m)) \stackrel{\Delta}{=} \beta_{k+1}^{f(i,m)} \tag{8}$$

where $f(i, m)$ is the next state, given an input i and state m, and $\beta_{k+1}^{f(i,m)}$ is the backward state metric at time $k + 1$ and state $f(i, m)$. β_k^m is obtained using

$$\beta_k^m = \sum_{j=0}^{1} \delta_k^{j,m} \, \beta_{k+1}^{f(j,m)} \tag{9}$$

Equation [9] indicates that the new backward state metric at time k and state m is obtained by summing two weighted state metrics from time $k+1$.

4. Computation of the extrinsic likelihood The ratio of *a posteriori probabilities* (APPs), known as the likelihood ratio $\Lambda(\hat{d}_k)$, or its logarithm $L(\hat{d}_k)$ called the log-likelihood ratio (LLR) is given by :

$$\Lambda(\hat{d}_k) = \frac{\sum\limits_{m} \lambda_k^{1,m}}{\sum\limits_{m} \lambda_k^{0,m}} \tag{10}$$

and

$$L(\hat{d}_k) = \log\left[\frac{\sum\limits_{m} \lambda_k^{1,m}}{\sum\limits_{m} \lambda_k^{0,m}}\right] \tag{11}$$

$\lambda_k^{1,m}$ is the joint probability that data $d_k = i$ and state $S_k = m$, conditioned on the received binary sequence R_1^N, observed from time $k = 1$ through some time N. It can be mathematically expressed as

$$\lambda_k^{1,m} = P(d_k = i, S_k = m | R_1^N) \tag{12}$$

Equation [11] can be expressed as

$$\Lambda(\hat{d}_k) = \frac{\sum\limits_{m} \alpha_k^m \, \delta_k^{1,m} \, \beta_{k+1}^{f(1,m)}}{\sum\limits_{m} \alpha_k^m \, \delta_k^{0,m} \, \beta_{k+1}^{f(0,m)}} \tag{13}$$

and

$$Ł(\hat{d}_k) = \log\left[\frac{\sum\limits_{m} \alpha_k^m \, \delta_k^{1,m} \, \beta_{k+1}^{f(1,m)}}{\sum\limits_{m} \alpha_k^m \, \delta_k^{0,m} \, \beta_{k+1}^{f(0,m)}}\right] \tag{14}$$

Substituting Equation [2] into Equation [13], we get

$$\Lambda(\hat{d}_k) = \pi_k \, \exp(\frac{2x_k}{\sigma^2}) \left[\frac{\sum\limits_{m} \alpha_k^m \, \exp(\frac{y_k v_k^{1,m}}{\sigma^2}) \, \beta_{k+1}^{f(1,m)}}{\sum\limits_{m} \alpha_k^m \, \exp(\frac{y_k v_k^{0,m}}{\sigma^2}) \, \beta_{k+1}^{f(0,m)}}\right] \tag{15}$$

$$= \pi_k \, \exp(\frac{2x_k}{\sigma^2}) \, \pi_k^e \tag{16}$$

and

$$L(\hat{d}_k) = L(d_k) + L_c(x_k) + L_e(\hat{d}_k) \tag{17}$$

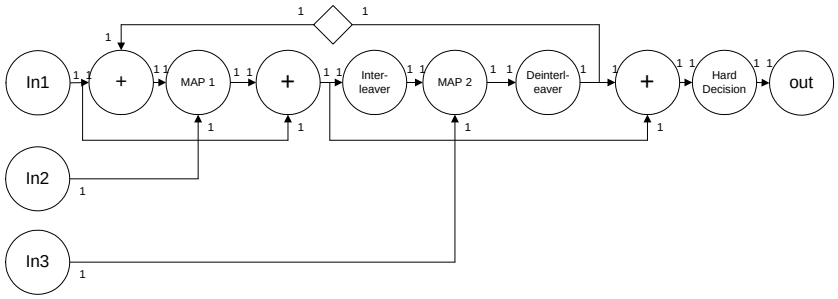


Fig. 3. Turbo Decoder SDF Model

where $\pi_k = \pi_k^1/\pi_k^0$ is the input *a priori probability* ratio, and π_k^e is the output extrinsic likelihood, each at time k. Thus π_k^e is a correction factor which changes the prior knowledge of a data bit. The correction terms are passed from one decoder to the next and the entire process is iterated a number of times in order to minimize the probability of error. The extrinsic likelihood π_k^e, resulting from a particular iteration replaces the *a priori* likelihood ratio π_{k+1} for the next iteration.

V. Modeling and implementation

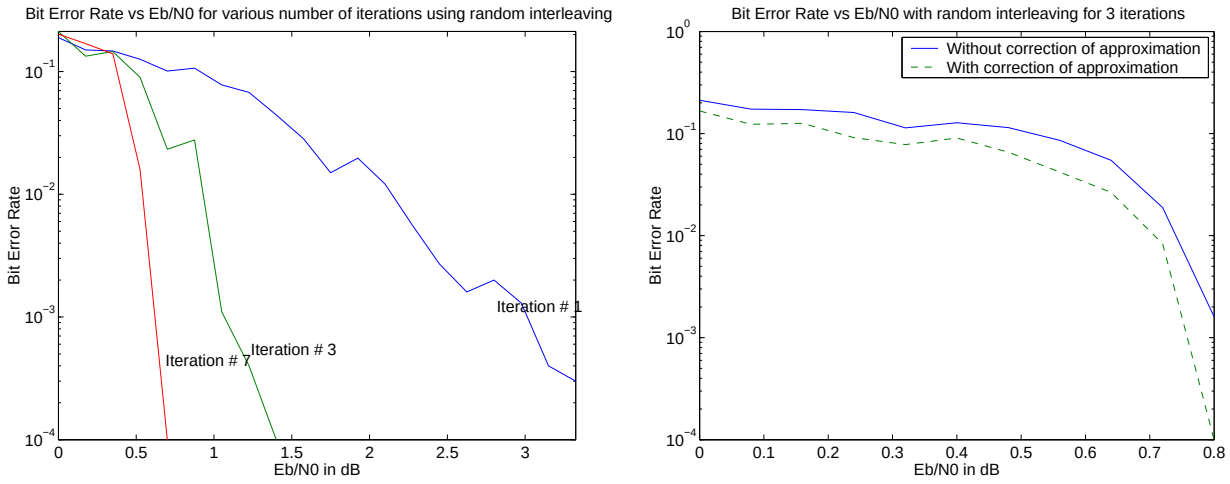
Due to the dataflow intensive nature of the system, a synchronous dataflow graph (SDF) [13] is well-suited for modeling the system. The individual components of the encoder/decoder blocks can be modeled as SDF actors, each of which can be implemented in hardware or software. The SDF model of the encoder is identical to that shown in Fig. 1. It is a homogeneous SDF with 1 token being equal to 1 bit. The detailed SDF model of the turbo decoder is shown in Fig. 3. It is also a homogeneous SDF graph.

The implementation of the MAP algorithm is based on that given in [14]. The direct implementation of MAP is computationally intensive and hence not cost effective for real time applications. In order to minimize the decoding complexity, the logarithms of the state metrics are taken. This converts the multiplication operation to additions (Log-MAP algorithm). The problem with the Log-MAP algorithm is that now we have logarithms of sum of exponentials. This can be simplified using the Jacobian logarithm [8], [15],

$$\log(e^{L1} + e^{L2}) = \max(L_1, L_2) + \log(1 + e^{|L_1 - L_2|}) \tag{18}$$

Most implementations compute the maximum term and ignore the correction factor (Max-Log-MAP algorithm). The performance of the turbo coding scheme is shown in Fig 4(a).

As suggested in [15], I used a small lookup table for the correction factors. The performance improvement obtained by using the correction factor is shown in Fig 4(b).



(a) Performance of turbo codes in a AWGN channel using random interleavers

(b) Performance improvement using a lookup table for correction factor

Fig. 4. Performance curves of the turbo coding scheme

VI. Sliding Window Implementation

In order to reduce the working memory, sliding window technique [10] is used. The important issue with sliding window technique is that trellis termination in every window is wasteful and is normally not done. Instead the β values for initialization are all taken equal to $1/M$ where M is the number of states. Usually the decoding window consists of valid bits and some excess inaccurately decoded bits of length equal to 4-5 times the constraint length. Since, the decoding over the last few bits is not accurate, the decoding is perfomed again in the next window as shown in Fig. 5. Thus this scheme increases the latency by performing the decoding twice over certain portions of the buffer while reducing the memory requirement.

VII. Conclusion and Future Work

In this project, I modeled and implemented a turbo encoder and decoder scheme using the MAP algorithm. A couple of enhancements to the conventional log-MAP decoding

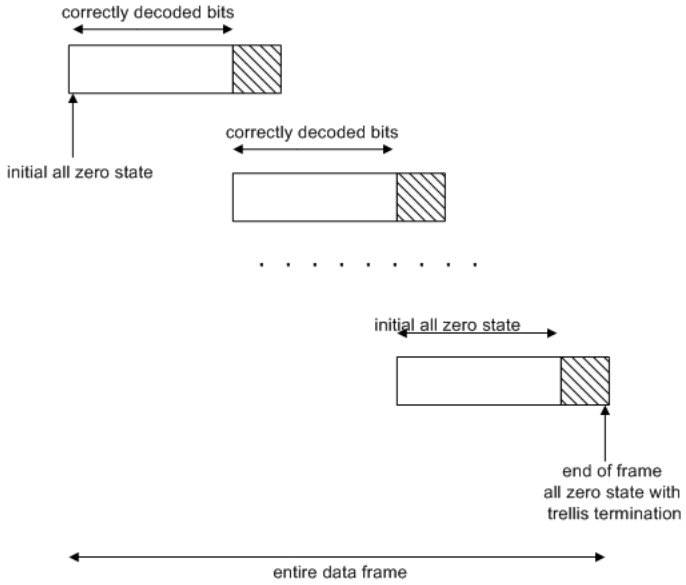


Fig. 5. Sliding Window Implementation of the MAP algorithm

was made by using a correction factor for the MAX approximation and using a sliding window implementation to reduce the memory.

The advantage of turbo codes over existing coding schemes is that it attains a very low BER at low signal-to-noise ratios. This makes it suitable for wireless applications where low transmission power is desired. However, the performance of turbo codes on Rayleigh and Ricean fading channels remain an active subject of research. The portability of this code to Advanced Design System (ADS) would be very beneficial for code re-usability and also the synthesis capability of ADS would result in faster product development.

References

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, August 1991.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding And Decoding: Turbo-Codes," in *Proceedings of IEEE International Communications Conference*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[3] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding And Decoding: Turbo-Codes," *IEEE Trans. on Communications*, vol. 44, pp. 1261–1271, Oct. 1996.

[4] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.

[5] L. Bahl, J. Jelinek, J. Raviv, and F. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. on Information Theory*, vol. IT-20, pp. 284–287, Feb 1974.

[6] G. D. Forney, "The Viterbi Algorithm," *Proc. of the IEEE*, vol. 61, pp. 268–278, Mar 1973.

[7] B. Vucetic, J. Yuan, and J. Yuan, *Turbo Codes - Principles and Applications* . Kluwer Academic Publishers, July 2000.

[8] S. S. Pietrobon, "Implementation and Performance of A Turbo/MAP Decoder," *Int. Journal of Satellite Communications*, vol. 16, pp. 23–46, 1998.

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

[10] S. A. Barbulescu, *Iterative Decoding of Turbo Codes and Other Concatenated Codes*. PhD thesis, University of South Australia, Feb 1996.

[11] S. Pietrobon and S. A. Barbulescu, "A Simplification of the Modified Bahl Decoding Algorithm for Systematic Convolutional Codes," in *Int. Symp. on Information Theory and its Applications*, (Sydney, Australia), pp. 1073–1077, Nov 1994.

[12] B. Sklar, *Digital Communications: Fundamentals and Applications*, ch. 8, pp. 475–509. Prentice-Hall, 2nd ed., Jan 2001.

[13] E. Lee and D. Messerschmitt, "Synchronous Data Flow," *Proc. of the IEEE*, vol. 75, pp. 1235–1245, Sept 1987.

[14] J. Nikolic-Popovic, "Implementing a MAP Decoder for CDMA 2000 Turbo Codes on a TMS320C62x DSP Device," *Texas Instruments Application Report, SPRA629*, May 2000.

[15] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo Decoding," *European Transactions on Telecommunications*, vol. 8, pp. 119–125, Mar-Apr 1997.