PARALLEL PROCESSING BASED TURBO DECODER DESIGN USING VERTIBI ALGORITHM

1Rupinder kaur, 2Sarpreet Singh

1M.tech student, SGGSW University
Fatehgarh Sahib, Punjab, India

2Assistant Professor, SGGSW University
Fatehgarh Sahib, Punjab, India

ABSTRACT

The turbo encoder is employed to increase the free distance of the turbo code, hence improving its error-correction performance. A Design of Turbo encoder and decoder. In data transmission, turbo coding helps achieve near Shannon limit performance. SOVA Algorithms have been studied and their design considerations have been presented. Turbo coding is an advanced error correction technique widely used in the communications industry. Turbo encoders and decoders are key elements in today's communication systems to achieve the best possible data reception with the fewest possible errors. The basis of turbo coding is to introduce redundancy in the data to be transmitted through a channel. The redundant data helps to recover original data from the received data. The design problem of generating the turbo code and decoding the code iteratively using SOVA detectors has been considered. This paper has proposed a design of a TURBO Decoder using SOVA Algorithm.

KEYWORDS: Turbo decoder, Forward Error Correction, Viterbi Algorithm

1. INTRODUCTION

Turbo code is a step in that direction. But it turns out that for an acceptable performance we do not really need to send the information infinite number of times, just two or three times provides pretty decent results for our earthly channels. In Turbo codes, particularly the parallel structure, Recursive systematic convolutional (RSC) codes working in parallel are used to create the “random” versions of the message. The parallel structure uses two or more RSC codes, each with a different interleaver. The purpose of the interleaver is to offer each encoder an uncorrelated or a “random” version of the information, resulting in parity bits from each RSC that are independent. How “independent” these parity bits are, is essentially a function of the type and length/depth of the interleaver. The design of interleaver in itself is a science. In a typical Viterbi code, the messages are decoded in blocks of only about 200 bits or so, where as in Turbo coding the blocks are on the order of 16K bits long. The reason for this length is to effectively randomize the sequence going to the second encoder. The longer the block length, the better is its correlation with the message from the first encoder, i.e. the correlation is low. On the receiving side, there are same number of decoders as on the encoder side, each working on the same information and an independent set of parity bit. This type of structure is called Parallel Concatenated Convolutional Code or PCCC. The convolutional codes used in turbo codes usually have small constraint length. Where a longer constraint length is an advantage in stand-alone convolutional codes, it does not lead to better performance in TC and increases computation complexity and delay. The codes in PCCC must be RSC. The RSC property allows the use of systematic bit as a standard to which the independent parity bits from the different coders are used to assess its reliability. The decoding most often applied is an iterative form of decoding. Turbo decoding [1] is increasingly proposed in emerging and future digital communication systems, for example, fiber-optic communication, wireless communication, and storage applications. Practical turbo decoder designs, as the one used for IEEE 802.16e, Long-term Evolution (LTE) or IEEE 802.11 standards, require high data throughput (several hundred of Mbps) and low...
latency (ten ms or so). To cope with these requirements, turbo decoder implementations have to be massively parallel. Therefore, the parallelism involved in implementations of this iterative process has to be carefully analyzed through real industrial constraints such as area and throughput. The prevalence of turbo codes in communication systems has also nurtured the usage of decoding techniques that iteratively exchange messages based on the probability of decoded bits, also known as “soft” information. This means that the decision on the outcome of a received bit is predicated on the existence of a spectrum of values indicating the likelihood of a “0” or a “1” value. To compare, traditional methods of decoding such as those that employ the Viterbi algorithm make “hard” decisions—it is either a “0” or a “1” and nothing in between. Figure 1 below illustrates the role of message passing in the decoding process.

Figure 1: Message Passing from transmitter to receiver

2. LITERATURE REVIEW

[1] shows design and implementation aspects of parallel turbo-decoders that reach the 326.4 Mb/s LTE peak data-rate using multiple soft-input soft-output decoders that operate in parallel. [2] shows the design of new turbo codes that can achieve near-Shannon-limit performance. The design criterion for random interleavers is based on maximizing the effective free distance of the turbo code, i.e., the minimum output weight of codewords due to weight-2 input sequences. An upper bound on the effective free distance of a turbo code has been derived. A review on multiple turbo codes (parallel concatenation of q convolutional codes), which increase the so-called ‘interleaving gain” as q and the interleaver size increase, and a suitable decoder structure derived from an approximation to the maximum a posteriori probability decision rule has been shown. A new rate 1/3, 2/3, 3/4, and 4/5 constituent codes have been developed to be used in the turbo encoder structure.[3] shows development of an application specific design methodology for low power solutions. The methodology starts from high level models which can be used for software solution and proceeds towards high performance hardware solutions. The effect on performance due to variation in parameters like frame length, number of iterations, type of encoding scheme and type of the interleaver in the presence of additive white Gaussian noise has been studied with the floating point C model. In order to obtain the effect of quantization and word length variation, a fixed point model of the application has also been developed.

3. RESEARCH PROBLEM

Our problem is to make a design to generate the turbo code and decode the code iteratively using parallel concatenated turbo decoding using viterbi algorithm.

4. RESEARCH METHODOLOGY
Using SOVA Detectors for turbo decoding and compute the performance of the system receiver by computing the Bit Error Rate. Iterative Decoding scheme would be implemented for the same.

5. TURBO CODING TECHNIQUE

In information theory, turbo codes are a class of high-performance forward error correction (FEC) codes, which were the first practical codes to closely approach the channel capacity, a theoretical maximum for the code rate at which reliable communication is still possible given a specific noise level. Turbo codes are finding use in (deep space) satellite communications and other applications where designers seek to achieve reliable information transfer over bandwidth- or latency-constrained communication links in the presence of data-corrupting noise. Turbo codes are nowadays competing with LDPC codes, which provide similar performance.

6. PUNCTURING IN TURBO CODES

While for deep space applications low-rate codes are appropriate, in other situations such as satellite communications and magnetic read applications, a rate of ½ or higher is preferred [10]. The role of the turbo code puncturer is identical to that of its Convolutional code counterpart to periodically delete selected bits to reduce coding overhead. For the case of iterative decoding, it is preferable to delete only parity bits.

7. TURBO DECODING ALGORITHMS

Figure 5 below shows the various decoding algorithms available for decoding of turbo codes. All the algorithms are based upon the trellis-based estimation. The trellis based estimation algorithms are classified into two types. They are sequence estimation algorithms and symbol-by-symbol estimation algorithms. The Viterbi algorithm, SOVA (soft output Viterbi algorithm) and improved SOVA are classified as sequence estimation algorithms [10]. Whereas the MAP algorithm, Max-Log-Map and the Log-Map algorithm are classified as symbol-by-symbol estimation algorithms. In general the symbol-by-symbol estimation algorithms are more complex than the sequence estimation algorithms but their BER performance is much better than the sequence estimation algorithms.

![Decoding Algorithms for Turbo Codes](image)

The MAP, SOVA, LOG-MAP, MAX-LOG-MAP, improved SOVA, all these algorithms produce soft-outputs. The Viterbi algorithm is a hard-decision output decoding algorithm. SOVA is soft-output producing Viterbi algorithm.

7.1 MAP Algorithm (HARD DECISION DECODING)

Maximum A Posteriori algorithm, now commonly named BCJR algorithm, presents an optimal decoding method for linear codes which minimizes the symbol error probability. This is in contrast to the traditionally used Viterbi algorithm, which is based on the principle of maximum length sequence estimation (MLSE) [10]. In essence the Viterbi algorithm minimizes the probability of sequence (or word) error, which does not translate to minimizing the probability of individual bit (symbol) errors.

Let \( u = (u_1, u_2 \ldots u_N) \) be the binary random variables representing information bits. In the systematic encoders, one of the outputs \( x_s = (x_1^s, x_2^s \ldots x_N^s) \) is identical to the information sequence \( u \). The other is the parity information sequence output \( x_p = (x_1^p, x_2^p \ldots x_N^p) \). We assume BPSK modulation and an AWGN channel with noise spectrum density \( N_0 \). The noisy versions of the outputs are \( y_s = (y_1^s, y_2^s \ldots y_N^s) \) and \( y_p = (y_1^p, y_2^p \ldots y_N^p) \). The noise is used for simplicity. In the MAP decoder, the decoder decides whether \( u_k = +1 \) or \( u_k = -1 \) depending on the sign of the following log-likelihood.
ratio (LLR)
\[ L_R (u_k) = \log \frac{P(u_k = +1 \mid y)}{P(u_k = -1 \mid y)} \]

The ‘a priori’ probability of information bits generated by the other MAP decoder must be considered in iterative decoders. MAP decoding, as indicated in the equations above, is a multiplication-intensive operation as it stands. The algorithm is likely to be considered too complex for implementation in many communication systems. However, the algorithm can be rearranged in the log-domain to reduce the computational. Therefore, the log-domain computations of the BCJR algorithm can be separated into three main categories for radix-2 trellises:
1. Branch metric computation
2. Forward/Backward metric Computation
3. Combination of forward and backward state metrics

Let \( S_k \) denote the state of the encoder at time \( k \). It can take values from 0 to \( 2M-1 \) where \( M \) is the number of memory elements in the encoder. LLR can be rewritten as
\[ L_R (u_k) = \log \sum_{s_k} \sum_{s_{k-1}} s_k \gamma_i (y_k, S_{k-1}, S_k) \alpha_{k-1} (S_{k-1}) \beta_k (S_k) \]

where the forward recursion metric, \( \mathbf{\alpha} \) is is the backward recursion metric and \( \mathbf{\beta} \) is is the branch metric. They are defined as
\[ \alpha_k (S_k) = \sum_{s_{k-1}} \gamma_i (y_k, S_{k-1}, S_k) \alpha_{k-1} (S_{k-1}) \]
\[ \beta_k (S_k) = \sum_{s_{k+1}} \gamma_i (y_{k+1}, S_k, S_{k+1}) \beta_{k+1} (S_{k+1}) \]

\[ \gamma_i (y_k^i, y_{k-1}^i, S_{k-1}, S_k) = q(u_k = i \mid S_k, S_{k-1}) p(y_k^i \mid u_k = i) p(y_{k-1}^i \mid u_k = i, S_k, S_{k-1}) P_i (S_k \mid S_{k-1}) \]

The parameter \( q (u_k = i \mid S_k, S_{k-1}) \) is either one or zero depending on whether \( u_k = i \) is possible for the transition from state \( S_{k-1} \) to \( S_k \) or not. Calculating \( p(y_k^i \mid u_k = i) \) and \( p(y_{k-1}^i \mid u_k = i, S_k, S_{k-1}) \) is trivial if the channel is AWGN. The last component \( P_i (S_k \mid S_{k-1}) \) usually has a fixed value for all \( k \). However, this is not the case in the iterative decoding.

7.2 SOVA Algorithm (SOFT DECISION DECODING)

However, in practice the MAP turbo decoder is too complex to be implemented due to the large number of multiplications and the need of non-linear functions. For that reason, two simplified versions of it were proposed in the past, namely Log-MAP and Max-Log-MAP [3]. The latter algorithm is sub-optimum in terms of bit error rate (BER) performance but easier to be implemented, as it requires only additions and the max operator.

Another sub-optimum algorithm that is suitable for turbo decoding is the soft output Viterbi algorithm (SOVA). It is a modified Viterbi algorithm (VA) that produces, in addition to the most likely path sequence, a reliability value of each estimated bit [4]. It was found that the iterative SOVA is 0.7 dB worse than the MAP algorithm at BER of 10^-4 [3]. This is largely because the SOVA considers only two path sequences to update its soft output, namely the survivor and the concurrent path sequences. A first attempt to improve the SOVA was reported in [5] with two proposed modifications, so as to connect its soft output and to follow a Gaussian distribution. In the first modification, the extrinsic information is normalized by multiplying with a correcting factor \( c \) that depends on the variance of the decoder output, while in the second one (that is less effective) the correlation in the decoder input is eliminating by inserting two more correcting coefficients. Another attempt to improve the SOVA was described in [6] where the reliability of the soft output is limited to a small range of values. In [6] was also described the SOVA updating soft output rule by Bufail and it was later shown that this is equivalent to the Max-Log-MAP algorithm [7]. Finally, the Max-Log-MAP turbo decoder was improved in [8] by following a normalization method similar to [5] but keeping constant the correcting factor \( c \). It is known that the performance of a SOVA (soft output Viterbi algorithm) turbo decoder can be improved, as the extrinsic information that is produced at its output is over optimistic. A new parameter associated with the branch metrics calculation in the standard Viterbi algorithm is introduced that affects the turbo code performance. Different parameter values show a simulation improvement in the AWGN channel as well as in an uncorrelated Rayleigh fading channel [10]. There are different efficient approaches proposed to improve the performance of soft-output Viterbi algorithm (SOVA)-based turbo decoders. In the first approach, an easily obtainable variable and a simple mapping function are used to compute a target scaling factor to normalize the extrinsic information output from turbo decoders. The scaling factor can be a variable scaling factor or a fixed scaling factor. In Variable scaling factor method, a scaling factor \( c \) of should be employed to normalize the soft output of SOVA.
decoders. In practice, to compute the mean and variance of the soft output from SOVA decoders, multiplication and addition operations must be performed at each symbol-processing cycle within each iterative decoding. Also, to compute the final scaling factor, a division operation must be performed before the next iteration begins. All of these imply that a practical SOVA-based turbo decoder with the normalization process embedded may work either with a larger clock cycle period or with a considerable extra latency when pipeline techniques are employed [10]. The SOVA is a modified Viterbi algorithm which produces soft outputs associated with the decoded bit sequence. These modifications include a modified metric computation and a reliability value update along the maximum likelihood (ML) path. Let $d_k$ denote a trellis branch into a node and $M_k^{(m)}$ denote the accumulated metric at time $k$ for branch $m$.

Let $m$ denote a trellis branch into a node and $M_k^{(m)}$ denote the accumulated metric at time $k$ for branch $m$.

\[
\begin{align*}
M_k^{(m)} &= M_{k-1}^{(m)} + u_k^{(m)} L_c y_{k,1} + x_k^{(m)} L_c y_{k,2}
\end{align*}
\]

8. RESULTS

I. FOR PUNCTURED CODES

- Frame size = 400
- Code generator:
  
  \[
  \begin{bmatrix}
  1 & 1 & 1 & 1 & 1 \\
  1 & 0 & 0 & 0 & 1 
  \end{bmatrix}
  \]
- Unpunctured, code rate = 1/3
- Iteration number = 5
- Terminate frame errors = 15
- $E_b / N_0$ (dB) = 2.00

Probability of Bit error rate vs SNR for punctured codes using SOVA DECODER

Bit Error Rate (from iteration 1 to iteration 5):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>2.2727e-002</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>1.6835e-003</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>0.0000e+000</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>0.0000e+000</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>0.0000e+000</td>
</tr>
</tbody>
</table>

Frame Error Rate (from iteration 1 to iteration 5):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>6.6667e-001</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>3.3333e-001</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>0.0000e+000</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>0.0000e+000</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>0.0000e+000</td>
</tr>
</tbody>
</table>

II. FOR UNPUNCTURED CODES
9. CONCLUSIONS FROM THE RESULT

It has been seen that probability of bit error has reduced significantly up to three times less in SOVA Decoder as compared to LOG-MAP Decoder. However, time for transmission and BER calculation in SOVA has increased significantly.

- Maximum probability of bit error in SOVA decoder (punctured codes): 0.03,
- It has also been seen that bit error rate and frame error rate reduces significantly with respect to SNR of the channel for each iteration at every frame transmission.
- Also, the number of frames transmitted without error has been significantly increased in SOVA decoder for particular frame error termination. As studied, for 15 termination frames, number of frames decoded is increased to 129 frames in SOVA decoder.

REFERENCES

[1] A New Method of Improving SOVA Turbo Decoding for AWGN, Rayleigh and Rician Fading Channels, Stylianos Papaharalabos, Peter Sweeney, Barry G. Evans Centre for Communication Systems Research (CCSR) University of Surrey, Guildford, Surrey, GU2 7XH, UK, April 27, 2010, IEEE Xplore


