International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-1 Number-1 Issue-1 September 2011

Performance of Turbo Code for UMTS in AWGN channel

Bhavana Shrivastava[1], Yudhishthir Raut[2], Ravi Shankar Mishra[3]
M.Tech Scholar, NIIST Bhopal, India[1]
Reader, NIIST Bhopal, India[2]
HOD (EC), NIIST Bhopal, India[3]

Abstract

This paper deals with "Turbo codes". The turbo code encoder is built using a parallel concatenation of two RSC codes and associated decoder is SOVA. Simulation carried out for different BER, iterations, constraint lengths and frame sizes (FS) to show performance properties of turbo codes in AWGN channel and using binary shift keying modulation.

Keywords

TURBO CODE, RSC, INTERLEAVER, AWGN and SOVA.

1. Introduction

Over the past decade, there has been great interest in the development of Joint Source Channel Coding (JSCC) techniques. This is motivated by the impressive growth of personal mobile-communications, which aim to support ever more sophisticated services in hostile wireless environments. Channel bandwidth limitations require powerful source compression techniques, but also noisy channels severely impact the compressed data error sensitivity. In this context, JSCC has emerged as a viable approach to the problem. On the other hand, the most recent coding standards for media applications, such as JPEG2000 for still images and H264/AVC for videos, are adopting arithmetic codes for entropy coding. This adoption is due firstly to the higher compression performance given by arithmetic codes when compared to other methods and secondly to the availability of low complex algorithms for its implementation. [1] Turbo codes were introduced in 1993 by Berrou et al. In the last decade turbo codes have attracted considerable attention because of the large coding gains they can achieve in an additive white Gaussian noise (AWGN) channel [2] and is perhaps the most exciting and potentially important development in coding theory in recent years. Turbo codes have been adopted as a channel coding scheme in a number of mobile standards, such as Universal Mobile Telecommunications System (UMTS), cdma2000 and recently in WiMAX and DVB-2. Particularly, for UMTS turbo codes are used for high data rates. [3] This paper considers different data rates iterations, constraint lengths and frame sizes (FS) in AWGN channels, validated by computer simulation .They achieve near Shannon-limit error correction performance with relatively simple component codes and large interleaves. They can be constructed by concatenating at least two component codes in a parallel fashion, separated by an interleave. Simple (2, 1, 4) convolution codes can achieve very good results. In order for a concatenated scheme such as a turbo code to work properly, the decoding algorithm must affect an exchange of soft information between component decoders. The concept behind turbo decoding is to pass soft information from the output of one decoder to the input of the succeeding one, and to iterate this process several times to produce better decisions. Turbo codes are still in the process of standardization but future applications will include mobile communication systems, deep space communications, telemetry and multimedia. Hence, the understanding of turbo encoding and decoding and their performance is of almost important. Performance of turbo code can also be increased by multifold coding [4].

All software implementation of the turbo codes has been done in MATLAB with channel Additive White Gaussian Noise (AWGN) and Binary Phase Shift Keying (BPSK) modulation.

The paper is partitioned as follows. The next section describes the turbo encoder and the decoder. The SOVA decoding algorithms are then outlined. Typical results obtained are then presented, followed by a short conclusion.

2. Turbo Code Encoder

A turbo encoder is the parallel concatenation of recursive systematic convolution (RSC) codes, separated by an interleave, as shown in Fig. 1. The data flow d_k goes into the first elementary RSC encoder, and after interleaving, it feeds a second elementary RSC encoder. The input stream is also systematically transmitted as X_k, and the redundancies produced by encoders 1 and 2 are transmitted as Y_{1k} and Y_{2k}. For turbo codes, the main reason of using RSC encoders as constituent encoders instead of the traditional non-recursive nonsystematic convolutional codes, is to use their recursive nature and not the fact that they are systematic.[6] turbo code with two different structures: The first one is the conventional structure in which three rate 1/2 identical constituent encoders are used to build the

encoder, this is known as the Equal Rate Turba Code (ERTC). The second structure is a modified structure with encoder consists of two different constituent encoders (one of rate = 1/2 and the other of rate = ID), this structure is called Uneqiial Rare Turbo Code (URTC). [5]The global rate of the turbo encoder in Fig. 1 is one-third.

The interleaver is an important design parameter in a turbo code. It takes a particular stream at its input and produces a different sequence as output. Its main purpose at the encoder side is to increase the free distance of the turbo code, [6] hence improving its error-correction performance [7].

There are different types of interleaves, such as the block, pseudo-random, simile, and odd-even interleaves [8]. They differ in the way they shuffle the input symbols. In this paper a random interleaves is used.

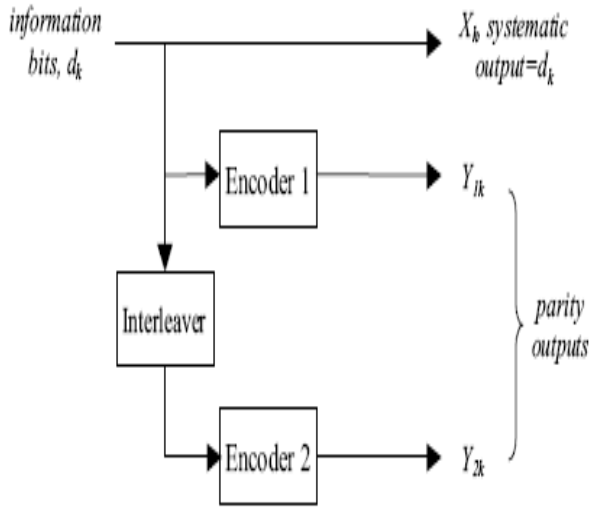

Figure 1: A fundamental turbo encoder

3. Turbo Code Decoder

The turbo code decoder is based on modified Viterbi Algorithm that incorporates reliability values to improve decoding performance. The Viterbi Algorithm (VA) is modified to deliver not only the most likely path sequence in a finite-state Markov chain, but either the a-posteriori probability for each bit or reliability value. With this reliability indicator the modified VA produces soft decisions to be used in decoding of outer codes. In order to design and implement the decoding algorithm, first the concept of reliability for viterbi decoding and the metric that will be used in the modified Viterbi Algorithm for turbo code decoding is described.

3.1 Principle of the General Soft-Output Viterbi Decoder

The Viterbi algorithm produces the ML output sequence for convolutional codes. This algorithm provides optimal sequence estimation for one stage convolutional codes. For concatenated (multistage) convolutional codes, there are two main drawbacks to conventional Viterbi decoders. First, the inner Viterbi decoder produces bursts of bit errors which degrade the performance of the outer Viterbi decoders [9]. Second, the inner Viterbi decoder produces hard decision outputs which prohibit the outer Viterbi decoders from deriving the benefits of soft decisions [9]. Both of these drawbacks can be reduced and the performance of the overall concatenated decoder can be significantly improved if the Viterbi decoders are able to produce reliability (soft-output) values [10]. The reliability values are passed on to subsequent Viterbi decoders as a priori information to improve decoding performance. This modified Viterbi decoder is referred to as the soft-output Viterbi algorithm (SOVA) decoder. Figure 2 shows a concatenated SOVA decoder.

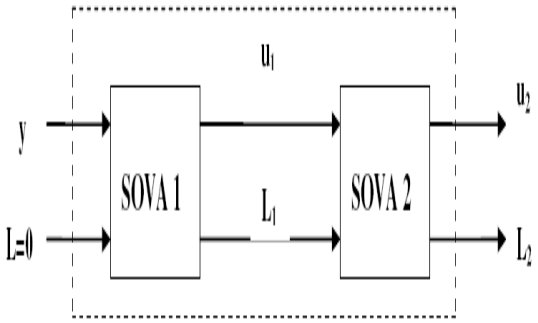

Figure 2: A concatenated SOVA decoder where y represents the received channel values, u represents the hard decision output values, and L represents the associated reliability values.

3.2 Reliability of SOVA decoder

The reliability of the SOVA decoder is calculated from the trellis diagram as shown Fig 3. The solid line indicates the survivor path (assumed here to be part of the final ML path) and the dashed line indicates the competing (concurrent) path at time t for state 1. For the sake of brevity and clarity, survivor and competing paths for other nodes are not shown. The label $S_{1,t}$ represents state 1 and time t. Also, the labels {0,1} shown on each path indicate the estimated binary decision for the paths. The survivor path for this node is assigned an accumulated metric $Vs(S_{1,t})$ and the competing path for this node is assigned an accumulated metric $Vc(S_{1,t})$.

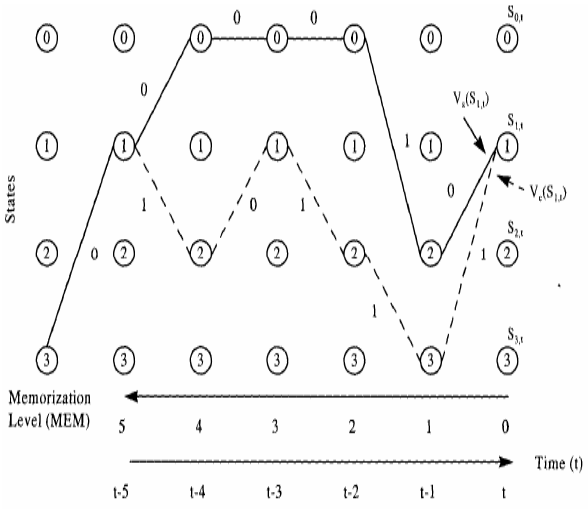

Figure 3: Example of survivor and competing paths for reliability estimation at time t [10].

fundamental information for assigning a reliability value L(t) to node $S_{1,t}$'s survivor path is the absolute difference between the two accumulated metrics, $L(t) = | Vs(S_{1,t}) - Vc(S_{1,t}) |$ [10]. The greater this difference, the more reliable is the survivor path. If the bit on the survivor path at any step is the same as the associated bit on the competing path, the reliability value at that bit position remains unchanged. However, if they differ, for example, at times t-2 and t-4, the reliability values at the corresponding steps needs to be updated. The soft output is then the final updated reliability values of each decoded bit.

1. (a) Initialize time t = 0.
   (b) Initialize $M_0^{(m)}$ only for the zero state in the trellis diagram and all other states to -∞.
2. (a) Set time t = t + 1.
   (b) Compute the metric $M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c y_{t,1} + \sum_{j=2}^{N} x_{t,j}^{(m)} L_c y_{t,j} + u_t^{(m)} L(u_t)$ for each state in the trellis diagram where, m denotes allowable binary trellis branch/transition to a state (m = 1, 2).

$M_t^{(m)}$ is the accumulated metric for time t on branch m.

$u_t^{(m)}$ is the systematic bit (1st bit on N bits) for time t on branch m.

$x_{t,j}^{(m)}$ is the j-th bit on N bits for time t on branch m (2 < j < N).

$y_{t,j}$ is the received value from the channel corresponding to $x_{t,j}^{(m)}$.

$L_c = 4\frac{Eb}{No}$ is the channel reliability value.

$L(u_t)$ is the a-priori reliability value for time t. This value is from the preceding decoder. If there is no preceding decoder, then this value is set to zero.

3. Find $\max_m M_t^{(m)}$ for each state. For simplicity, let $M_t^{(1)}$ denote the survivor path metric and $M_t^{(2)}$ denote the competing path metric.
4. Store $M_t^{(1)}$ and its associated survivor bit and state paths.
5. compute $\Delta_t^0 = \frac{1}{2} | M_t^{(1)} - M_t^{(2)} |$
6. Compare the survivor and computing paths at each state for time t and store the MEMs where the estimated binary decisions of the two paths differ.
7. Update $\Delta_t^{MEM} \approx \min_{k=0,..,MEM}\{\Delta_t^k\}$ for all MEMs from smallest to largest MEM.
8. Go back to Step (2) until the end of the received sequence.
9. Output the estimated bit sequence û and its associated "Soft" or L-value sequence.

$L(\hat{u}) = \hat{u} * \Delta$, where, * operator defines element by element multiplication operation and Δ is the final updated reliability sequence. $L(\hat{u})$ is then processed and passed on as the a-priori sequence $L(u)$ for the succeeding decoder.

3.3 Implementation of SOVA for turbo code [11][12]

The iterative turbo code decoder is composed of two concatenated SOVA component decoders. Figure 4 shows the turbo code decoder structure.

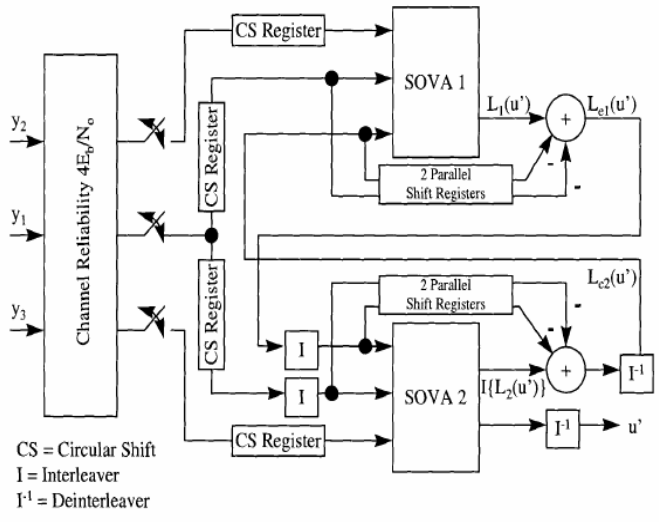

Figure 4: SOVA iterative turbo code decoder.

The turbo code decoder processes the received channel bits on a frame basis. As shown in Figure 5, the received channel bits are de-multiplexed into the systematic stream y_1 and two parity check streams y_2 and y_3 from component encoders 1 and 2 respectively. These bits are weighted by the channel reliability value and loaded on to the CS registers. The registers shown in the figure are used as buffers to store sequences until they are needed. The switches are placed in the open position to prevent the bits from the next frame from being processed until the present frame has been processed.
The SOVA component decoder produces the "soft" or L value $L(u_t')$ for the estimated Bit u_t' (for time t). The "soft" or L- value $L(u_t')$ can be decomposed into three distinct terms as stated in [11].

$$L(u_t') = L(u_t) + L_c y_{t,1} + L_e(u_t')$$

$L_e(u_t')$ is the a-priori value and is produced by the preceding SOVA component encoder.
$L_c y_{t,1}$ is the weighted received systematic channel value.
$L_e(u_t')$ is the extrinsic value produced by the present SOVA component decoder.
The information that is passed between SOVA component decoders is the extrinsic value

$$L_e(u_t') = L(u_t') - L(u_t) - L_c y_{t,1}$$

The a-priori value $L(u_t)$ is subtracted out from the "soft" or L value $L(u_t')$ to prevent passing information back to the decoder from which it was produced. Also, the weighted received systematic channel value $L_c y_{t,1}$ is subtracted out to remove "common" information in the SOVA component decoders. Figure 5 show that the turbo code decoder is a closed loop serial concatenation of SOVA component decoders. In this closed loop decoding scheme, each of the SOVA component decoders estimates the information sequence using a different weighted parity check stream. The turbo code decoder further implements for iterative decoding to provide more dependable reliability/a-priori estimations from the two different weighted parity check streams, hoping to achieve better decoding performance.

The iterative turbo code decoding algorithm for the n-th iteration is as follows:

1. The SOVA1 decoder inputs sequences $4\frac{Eb}{No}y_1$ (systematic), $4\frac{Eb}{No}y_2$ (parity check), and $L_{e2}(u')$ outputs sequence $L_1(u')$ .For the first iteration, sequence $L_{e2}(u') = 0$ because there is no initial a-priori value (no extrinsic values from SOVA2).

2. The extrinsic information from SOVA1 is obtained by

$L_{e1}(u') = L_1(u') - L_{e2}(u') - L_c y_1$, Where $L_c = 4\frac{Eb}{No}$

3. The sequences $4\frac{Eb}{No}y_1$ and $L_{e1}(u')$ are interleaved and denoted as $I\{4\frac{Eb}{No}y_1\}$ and $I\{L_{e1}(u')\}$.

4. The SOVA2 decoder inputs sequences $I\{4\frac{Eb}{No}y_1\}$ (systematic), $I\{4\frac{Eb}{No}y_3\}$ (parity check that was already interleaved by the turbo code encoder), and $I\{L_{e1}(u')\}$ (a-priori information) and outputs sequences $I\{L_2(u')\}$ and $I\{u'\}$.

5. The extrinsic information from SOVA2 is obtained by
$I\{L_{e2}(u')\} = I\{L_2(u')\} - I\{L_{e1}(u')\} - I\{L_c y_1\}$

6. The sequences $I\{L_{e2}(u')\}$ and $I\{u'\}$ are deinterleaved and denoted as $L_{e2}(u')$ and u'. $L_{e2}(u')$ is fed back to SOVA1 as a-priori information for the next iteration and u' is the estimated bits output for the n-th iteration.

4. Results

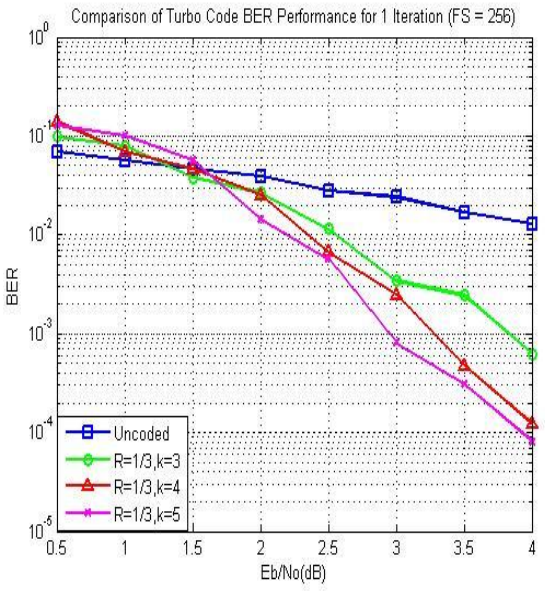

Figure 5: The performance of the rate 1/3 turbo code for different constraint lengths

The performance of the rate 1/3 turbo code in soft decision Viterbi decoding for different constraint lengths is shown in Figure 5. In this figure, it can be seen that as the constraint length increases, the performance of the code also increases, resulting in lower BER. This is the typical characteristic of any convolutional code.

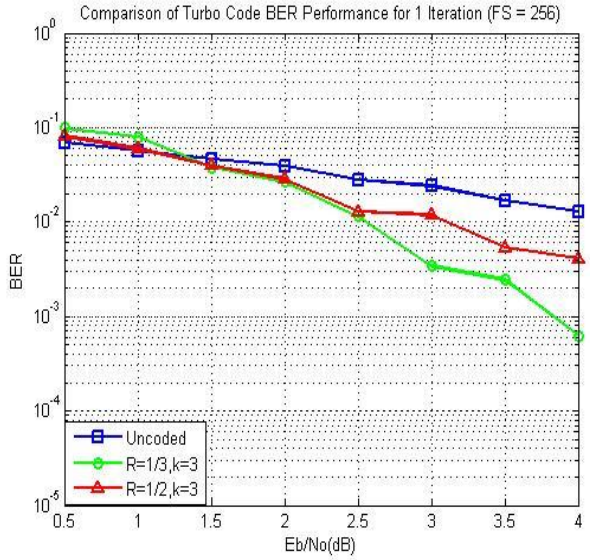

Figure 6: Comparison of Turbo code BER performance for 1 iteration,,FS=256

The simulated performance results of turbo codes with fixed frame sizes but different rates are shown in Figure 6. From these figures, it can be seen that for a fixed constraint length, a decrease in code rate increases the turbo code performance.

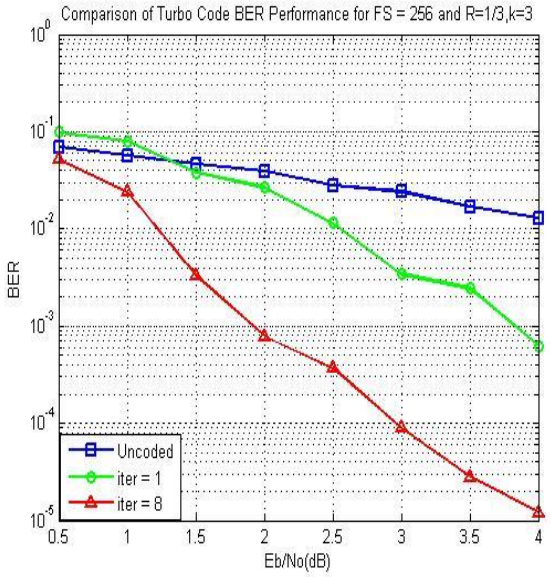

Figure 7: Turbo codes with fixed frame sizes, fixed constraint length and fixed rate

The simulated performance results of turbo codes with fixed frame sizes, fixed constraint length and fixed rate are shown in figure 7. From figure an increase in number of iteration improves the turbo code performance.

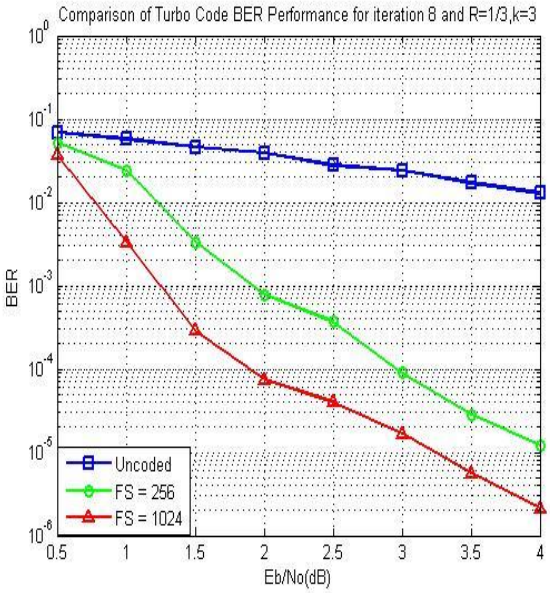

Figure 8: Turbo code with the same constraint length and rates but different frame sizes

The overall iterative (8 iterations) decoding gain for a turbo code with the same constraint length and rates but different frame sizes are shown in Figure 8. As shown in these figures, the overall iterative decoding gain increases as the frame size increases.

5. Conclusion

This paper described the concept of turbo coding, whose basic configuration depends on the parallel concatenation of two component codes (RSC). The three distinct terms of "Soft" or L-value are reviewed, and these values were used for the information exchange between the two SOVA component decoders.

The BER performance for turbo codes is investigated for many different cases. These different cases are summarized under the following three main categories:

1. Turbo code BER performance of 8 decoding iterations for fixed code rates constraint lengths but different frame sizes.
2. Turbo code BER performance of 8 decoding iterations for fixed frame sizes different code rates and constraint lengths.
3. Turbo code BER performance improvement between 1 decoding iteration and 8 decoding iterations for fixed code rates, fixed constraint lengths and fixed frame sizes.

The simulation results showed many interesting properties about turbo codes that are in the same direction with published research work. Some of these important results are listed below:

1. For a fixed turbo code encoder, its performance improves as the frame size increases.
2. For a fixed frame size, the turbo code performance increases under two different conditions. First, for a fixed constraint length, a decrease in code rate improves the performance. Second, for a fixed code rate, an increase in constraint length improves the performance.
3. Considerable decoding gain is observed if more, than one decoding iteration is used.

References

[1] Amin Zribi, Sonia Zaibi, Ramesh Pyndiah, Ammar Bouallègue, "Low-Complexity Joint Source/Channel Turbo Decoding of Arithmetic Codes with Image Transmission Application," dcc, pp.472, Data Compression Conference, 2009.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, 'Near-Shannon limit error-correcting coding and decoding: turbo codes (1)', Proc. IEEE Int. Conf. Commun., May (1993), 1064.

[3] Costas Chaikalis and Nicholas Samaras, Greece. Costas Chaikalis, Nicholas Samaras, "UMTS Dynamic Outer Interleaver Reconfiguration for Indoor Environment Using SOVA Turbo Decoder," icwmc, pp.205-210, 2009 Fifth International Conference on Wireless and Mobile Communications, 2009.

[4] J. Xu C. Tanriover, B. Honary and S. Lin. "Improving turbo code error performance by multifold coding" IEEE Comm., Letters, vol. 6, no. 5, pages 193 195, May 2002.

[5] A .Hmimy , S. C. Gupta . "Performance of Turbo-Codes for W-CDMA Systems in Flat Fading Channels". 452 - 456 vol.1. WCNC. 1999 IEEE.

[6] S. A. Barbulescu and S. S. Pietrobon, 'Terminating the trellis of turbo-codes in the same state', Electronics Let., 31 (1995), 22.

[7] Jung and M. Nabhan, 'Performance evaluation of turbo codes for short frame transmission systems', Electronics Let., 30 (1994), 111.

[8] M. Salehi H. R. Sadjadpour, J. A. Sloane and G. Nebe. "Interleaver design for turbo codes" IEEE J.Select. Areas Commun., vol.19, no.5, pages 831-837, May 2001.

[9] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in Proc., *IEEE* Globecom *Conj* (Dallas,TX, Nov. 1989), pp. 1680-1686.

[10] Claude Berrou, "A Low Complexity Soft-Output Viterbi Decoder Architecture", IEEE Trans. Commun , pp. 737 745, 1996.

[11] J. Hagenauer, "Source controlled channel decoding", IEEE Trans. Commun., vol. 43, pp. 2449-2457, Sep. 1995.

[12] Hagenauer, "Iterative Decoding of Binary Block and Convolutional Codes", IEEE Trans. Commun., vol.42, No. 2, pp. 429-445, MARCH. 1996.