Volume 3, Issue 4, April 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Techniques for Turbo Decoding Using Parallel Processing, Comparative Analysis

Rupinder kaur
M.tech student, SGGSW University
Fatehgarh Sahib, Punjab, India

Sarpreet Singh
Assistant Professor, SGGSW University
Fatehgarh Sahib, Punjab, India

Abstract- Parallel turbo decoding is becoming mandatory in order to achieve high throughput and to reduce latency, both crucial in emerging digital communication applications. This paper explores and analyzes parallelism techniques in convolution turbo decoding with the vertibi and MAP algorithm. A two-level structured classification of parallelism techniques is proposed and discussed: SISO decoder level parallelism and Turbo-decoder level parallelism. The second level of this classification is thoroughly analyzed on the basis of parallelism efficiency criteria, since it offers the best trade-off between achievable parallelism degree and area overhead. MAP, LOG-MAP and 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. This paper has compared the two techniques LOG-MAP and SOVA based parallel processing in Turbo Codes and an algorithm for Vertibi based decoder and encoder has been proposed.

Keywords: Turbo decoder, Forward Error Correction, Viterbi Algorithm, Interleaver

I. Introduction

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. 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 [2] . Turbo codes are nowadays competing with LDPC codes, which provide similar performance. According to Shannon, the ultimate code would be one where a message is sent infinite times, each time shuffled randomly. The receiver has infinite versions of the message albeit corrupted randomly. From these copies, the decoder would be able to decode with near error-free probability the message sent. This is the theory of an ultimate code, the one that can correct all errors for a virtually signal. Turbo code is a step in that direction [3]. 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 a SISO, it firstly computes branch metrics (or γ metrics), which represent the probability of a transition occurring between two trellis states. Then a SISO computes forward and backward recursions. Forward recursion (or α recursion) computes a trellis section (i.e., the probability of all states of the trellis) using the previous trellis section and branch metrics between these two sections, while backward recursion (or β recursion) computes a trellis section using the future trellis section and branch metrics between these two sections. Finally, extrinsic information is computed from the forward recursion, the backward recursion and the extrinsic part of the branch metrics.

Therefore, turbo decoders, because of iterative decoding process and BCJR complexity (bidirectional recursive computing), imply optimal parallelism exploitation in order to achieve high-data rates required in present and future applications. A generalised block diagram of turbo decoder usually uses the following blocks for completing the communication link from encoder to last stage of decoder:

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 [1]. 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. 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 [3]. The decoding most often applied is an iterative form of decoding.

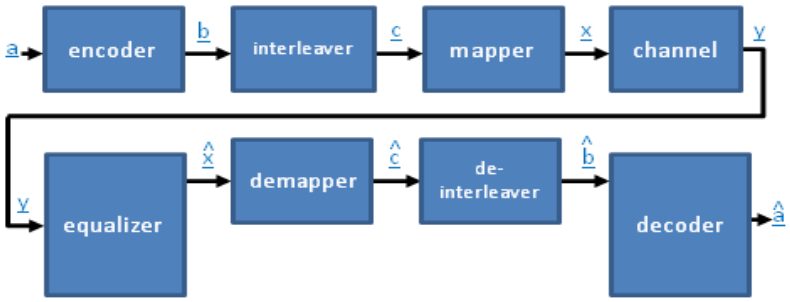

Figure 1: Communication Link encompassing a turbo encoder and decoder

II. Block Diagram Of Pccc Turbo Decoder

The complete turbo system consists of Turbo encoder on transmitter end and a parallel concatenated turbo decoder on receiver end as shown below. A general illustration of such a system is illustrated in Figure 2 and 3 below:

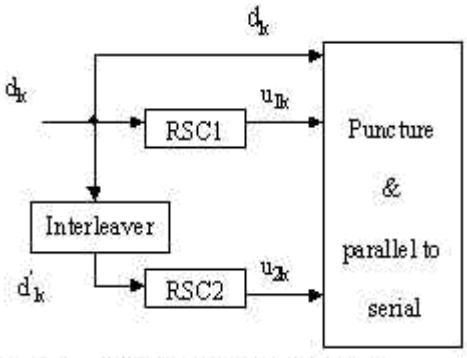

Figure 2: Turbo encoder System employing an Interleaver and puncturer

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, hence improving its error-correction performance.

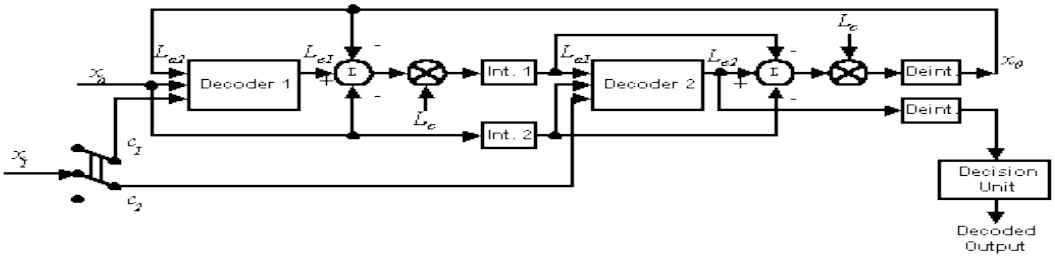

Figure 3: Parallel Concatenated Turbo decoder using iterative decoding via two decoders

The transmitter includes two encoders employed with interleaver for generating the parallel concatenated convolutional codes also known as Recursive Systematic Encoder. Channel employed is often considered as Additive White Gaussian Noise Channel. Receiver includes a front end decoder, deinterleaver, depuncturer and a final decoder. A turbo encoder is the parallel concatenation of recursive systematic convolutional (RSC) codes, separated by an interleaver, as shown in Fig. 2. The data flow dk goes into the first elementary RSC encoder, and after interleaving, it feeds a second elementary RSC encoder [3]. 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 non-systematic convolutional codes is to use their recursive nature and not the fact that they are systematic.[3] A turbo code is far too complex to decode with a single decoder. Instead, each convolutional code in the turbo code is decoded separately with soft information being passed from one decoder to the next. The decoding scheme of a turbo code is shown in Fig. 4. The above decoder consists of two serially interconnected soft-in soft-out (SISO) decoders, which can be SOVA or MAP decoders. x_k and y_k are the channel outputs with x_k corresponding to the systematic encoder output, and y_k is a multiplexed stream of the two punctured encoder outputs[3].

Hence, a demultiplexer is necessary at the receiver. z_k is called the a priori information and is equal to zero for the first iteration.

The decoder soft output, L (dk), also called log-likelihood ratio (LLR), can be separated into three components:

$L(d_k) = L_{sys} + L_{apr} + L_{ext}$

d_k denotes the actual hard decision of the decoder at step k. L ext is called the extrinsic information. It is a function of the redundant information introduced by the encoder and has the same sign as d_k. (L_{sys} + L_{apr}) constitutes the intrinsic information. L_{sys} is the LLR for the channel output and is the received systematic input to the decoder, scaled by the channel reliability. L_{apr} is equal to the extrinsic information produced by the previous decoder [3]. The intrinsic information is subtracted from the soft decoder output (LLR1, LLR2). The resulting output is the extrinsic information, which is passed on to the next decoder. This process is repeated until a desired performance is attained after a number of iterations.

III. Various Techniques For Forward Error Correction And Coding

A no. Of FEC techniques are available for removing redundancy in data transmitted in an AWGN channel. Most commonly used channel coding techniques are: Linear block codes, Convolutional codes and turbo codes.

A. BLOCK CODING TECHNIQUE

Block coding involves encoding a block of information bits into another block of bits with addition of some redundant bits to combat channel errors induced during wireless transmission. To accomplish this, the encoder not only transmits information symbols but also one or more redundant symbols. The decoder uses the redundant symbols to detect and possibly correct errors occurring during transmission as illustrated in the diagram below:

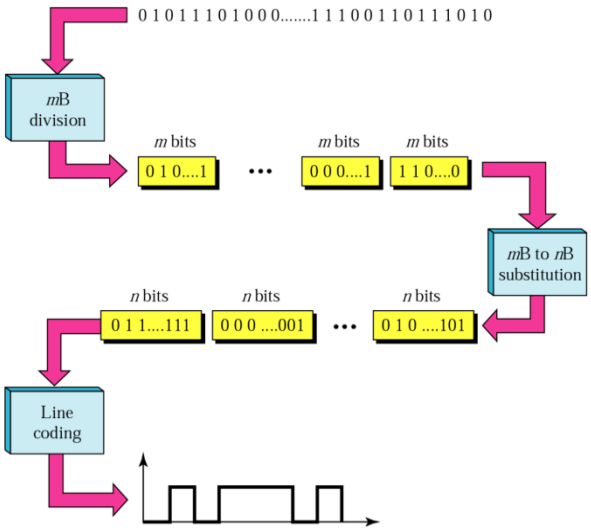

Figure 4: Linear Block Coding

B. CONVOLUTIONAL CODING TECHNIQUE

In telecommunication, a Convolutional code is a type of error-correcting code in which each *m*-bit information symbol (each *m*-bit string) to be encoded is transformed into an *n*-bit symbol, where *m*/*n* is the code *rate* ($n \geq m$) and the transformation is a function of the last *k* information symbols, where *k* is the constraint length of the code.

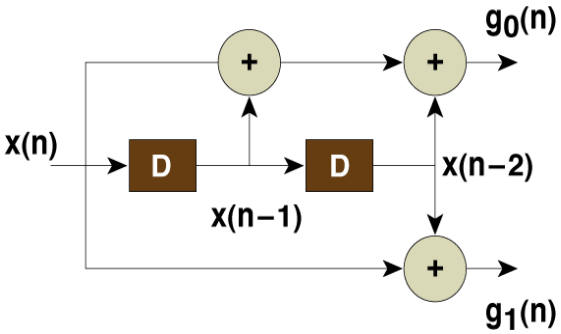

Figure 5: Convolutional Encoder Block Diagram

To convolutionally encode data, start with *k* memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has *n*modulo-2 adders (a modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is: 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0), and *n* generator polynomials — one for each adder (see figure below). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs *n* bits. Now bit shift all register values to the right (m_1 moves to m_0, m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues output until all registers have returned to the zero state.

C. 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.

IV. 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.

V. 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.

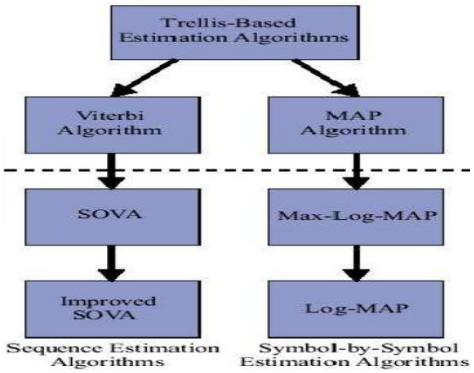

Fig.5. Decoding algorithms for turbo codes [10]

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.

A. 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 $u = (u_1, u_2 \dots u_{N)}$ be the binary random variables representing information bits. In the systematic encoders, one of the outputs $x_s = (x^s_1, x^s_2 \dots x^s_N)$ is identical to the information sequence u. The other is the parity information sequence output $x_p = (x^p_1, x^p_2 \dots x^p_N)$. We assume BPSK modulation and an AWGN channel with noise spectrum density N_o. The noisy versions of the outputs is $y_s = (y^s_1, y^s_2 \dots y^s_N)$ and $y_p = (y^p_1, y^p_2 \dots y^p_N)$, and $y = (y_s, y_p)$ 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)

$$L_R(u_k) = \log \frac{P(u_k = +1 \mid y)}{P(u_k = -1 \mid y)} \quad \text{---------- (1)}$$

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

$$L_R(u_k) = \log \frac{\sum s_k \sum s_{k-1} \gamma_1(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1}).\beta_k(S_k)}{\sum s_k \sum s_{k-1} \gamma_0(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1}).\beta_k(S_k)} \quad (2)$$

Where the forward recursion metric, □ isis the backward recursion metric and □$_i$ is the branch metric. They are defined as

$$\alpha_k(S_k) = \sum_{S_{k-1}} \sum_{i=0} \gamma_i(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1}) \quad \text{---------- (3)}$$

$$\beta_k(S_k) = \sum_{S_{k+1}} \sum_{i=0} \gamma_i(y_{k+1}, S_k, S_{k+1}).\beta_{k+1}(S_{k+1}) \quad \text{---------- (4)}$$

$$\gamma_i((y_k^s, y_k^p), S_{k-1}, S_k) = q(u_k = i \mid S_k, S_{k-1}).p(y_k^s \mid u_k = i).p(y_k^p \mid u_k = i, S_k, S_{k-1}).P_r(S_k \mid S_{k-1})$$

----------(5)

The parameter q (u_k = i/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^s_k | u_k = i) and p (y^p_k| u_k = i, S_k, S_{k-1}) is trivial if the channel is AWGN. The last component P_r (S_k|S_{k-1}) usually has a fixed value for all k. However, this is not the case in the iterative decoding.

B. 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 **IO4** *[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 comect 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 coeficients. 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 *Butfail* 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 **[SI** by following a normalization method similar to ***[5]*** hut 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 m denote a trellis branch into a node and $M_k^{(m)}$ denote the accumulated metric at time k for branch m.

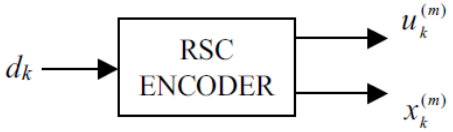

Fig.6. An RSC Encoder

Also, let $u_k^{(m)}$ be the systematic encoder output (in bipolar voltage form) for path m and $x_k^{(m)}$ be the corresponding parity output (Fig. 6). If $y_{k,1}$ and $y_{k,2}$ are the channel outputs corresponding to the systematic and parity outputs of the RSC encoder, then the metric used for the SOVA algorithm becomes

$M_k^{(m)} = M_{k-1}^{(m)} + u_k^{(m)} L_c y_{k,1} + x_k^{(m)} L_c y_{k,2}$

VI. Appilications Of Turbo Decoding In Wireless Communication

A) Mobile radio :

Few environments require more immunity to fading than those found in mobile communications [3]. For applications where delay vs performance is crucial, turbo codes offer a wide trade-off space at decoder complexities equal to or better than conventional concatenated code performance.

B) Digital video :

Turbo codes are used as part of the Digital Video Broadcasting (DVB) standards [3].

C) Long-haul terrestrial wireless :

Microwave towers are spread across and communications between them is subjected to weather induced fading. Since these links utilize high data rates, turbo codes with large interleavers effectively combat fading, while adding only insignificant delay [3].

D) Satellite communications and Deep space communication :

Turbo codes are used for deep-space applications. Low power design is important for many communication satellites that orbit closer to earth. Turbo codes can be used in such applications.

E) Military applications :

The natural applicability of turbo codes to spread-spectrum systems provides increased opportunity for anti-jam and low probability of intercept (LPI) communications. In particular, very steep BER vs SNR curves lead to a sharp demarcation between geographic locations that can receive communication with just sufficient SNR and those where SNR is insufficient [3]. Turbo codes are used in a number of military and defense application.

F) Terrestrial telephony:

Turbo codes are suited for down-to-earth applications such as terrestrial telephony

G) Oceanographic Data-Link :

This data-link is based on the Remote Environmental Data-Link (REDL) design. ViaSat is a company offering satellite networking products.

H) Image processing :

Embedded image codes are very sensitive to channel noise because a single bit error can lead to irreversible loss of synchronization between the encoder and the decoder.

I) WLAN (Wireless LAN) :

Turbo codes can increase the performance of a WLAN over traditional convolutional coding techniques. Using turbo codes, an 802.11a system can be configured for high performance [3].

J) OFDM :

The principles of orthogonal frequency division multiplexing (OFDM) modulation have been around for several decades. The FEC block of an OFDM system can be realizes by either a block-based coding scheme (Reed-Solomon) or turbo codes.

K) xDSL modems :

Turbo codes present a new and very powerful error control technique which allows communication very close to the channel capacity [3]. Turbo codes have outperformed all previously known coding schemes regardless of the targeted channel. Standards based on turbo codes have been defined.

VII. Future Scope And Advancement In Channel Decoding Techniques

Turbo codes discussed so far provides significant improve in the quality of data transmission over a noisy channel. However, there is a trade-off between the performance of Turbo codes and latest upcoming LDPC (low density parity check) codes. LDPC codes are latest advancement in the field of wireless communication and channel coding techniques. In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel,[1][2] and is constructed using a sparse bipartite graph.[3] LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even *arbitrarily* close on the BEC) to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length. LDPC codes are also known as Gallager codes, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960. LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth or return channel–constrained links in the presence of data-corrupting noise. Although implementation of LDPC codes has lagged behind that of other codes, notably turbo codes, the absence of encumbering software patents has made LDPC attractive to some.

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

[2] Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE Christoph Studer, Student Member, IEEE, Christian Benkeser, Member, IEEE, Sandro Belfanti, and Quiting Huang, Fellow, IEEE, January

2011 On the Design of Turbo Codes, D. Divsalar and F. Pollara, Communications Systems and Research Section, November 1995

[3] High Speed Max-Log-MAP Turbo SISO Decoder Implementation Using Branch Metric Normalization J. H. Han1, A. T. Erdogan1,2, T. Arslan1,2 *1University of Edinburgh, School of Engineering and Electronics*, Proceedings of the IEEE Computer Society Annual Symposium on VLSI New Frontiers in VLSI Design, IEEE 2005

[4] Sae-Young Chung, G. David Forney, Thomas J. Richardson, and Rüdiger Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit" IEEE Communications Letters, Vol. 5, No. 2, page(s):58-60, February 2001.

[5] Sadjadpour, H.R. Sloane, N.J.A. Salehi, M. Nebe, G. "Interleaver design for turbo codes", *IEEE Journal on Selected Areas in Communications*, Volume: 19, Issue: 5, page(s): 831-837, May 2001.

[6] T.P. Fowdur, K.M.S. Soyjaudah, "Joint source channel decoding and iterative symbol combining with turbo trellis-coded modulation**",** *Signal Processing*, Volume 89, Issue 4, page(s):570-582, April 2009.

[7] Erl-Huei Lu, Yi-Nan Lin, Wei-Wen Hung ,"Improvement of turbo decoding using cross-entropy", *Computer Communications*, Volume 32, Issue 6, page(s):1034-1038,27 April 2009.

[8] Fan Zhang and Henry D. Pfister, "On the Iterative Decoding of High-Rate LDPC Codes with Applications in Compressed Sensing", arXiv: 0903.2232v2 [cs.IT], 17 June, 2009.

[9] David Haley, Vincent Gaudet, Chris Winstead, Alex Grant, Christian Schlegel, "A dual-function mixed-signal circuit for LDPC encoding/decoding", *Integration, the VLSI Journal*, Volume 42, Issue 3,page(s): 332-339 ,June 2009.

[10] "Performance Analysis of Log-map, SOVA and Modified SOVA Algorithm for Turbo Decoder", Mohammad Salim, R.P. Yadav, S.Ravikanth, Dept. of Electronics & Communication Engineering, MNIT, Jaipur, Rajasthan (India), International Journal of Computer Applications (0975 – 8887) Volume 9– No.11, November 2010

[11] A Lowpower Design Methodology For Turbo Encoder And Decoder, Rajeshwari. M. Banakar, Department Of Electrical Engineering, Indian Institute Of Technology, Delhi, July 2004

