Log-Likelihood Ratio to Improve Hard Decision Viterbi Algorithm

Dr. Mahmood Farhan Mosleh
Electrical and Electronic Technical College /Baghdad
E-Mail: malmhmde@yahoo.com.
Dr. Fa'aza Abbas Abid
Institute of Technology/ Baghdad

Received on: 9/4/2012 & Accepted on: 31/1/2013

ABSTRACT

Hard decision of Viterbi decoder suffer from the need of high Signal to Noise Ratio (SNR) to achieve reasonable Bit Error Rate (BER). For the purpose of improving the efficiency of its performance, it must increase the constraint length of the code, in this case highlights the problem of complexity in the structure. Several methods are used to solve this problem. In this paper the Log-Likelihood Ratio (LLR) with 3 bit soft decision and unquantized scheme has been implemented with simple transceiver using Convolutional codes. Results are achieved using the last version of Matlab (R2011b) illustrates that such scheme 2.7 dB over conventional system. In addition it has been examine such system with increasing the speed of data rate to double, the results of simulation confirm that it need 7 dB to achieve 10^{-6} BER.

Keywords: Viterbi decoder, LLR, Unquantized, and soft decision.

نسبة الامكانية اللوغارتمية لتحسين القرار الجازم لخوارزمية فيتربي

الخلاصة

كاشف فيتربي نوع القرار الجازم يعاني من ارتفاع نسبة الاشارة الى الضوضاء للحصول على قيمه معقولة من معدل الخطاء. ولغرض تحسين كفائة اداء هذا الكاشف، فانه يتم زيادة طول الكلمة المشفرة ممايؤدي الى بروز مشكلة التعقيد الكبير في التركيب. عدد من الطرق استخدمت لحل هذه المشكلة. في هذا البحث تم استخدام نسبة الامكانية اللوغارتمية (LLR) ب ٣ قطع نوع القرار الناعم وكذلك النوع الغير مقرب وتم تطبيقها على نظام ارسال واستقبال بسيط باستخدام المشفر النوع الالتفافي. النتائج التي تم الحصول عليها باستخدام برنامج الماتلاب النسخة (R2011b) اوضحت بان هذا الرتكيب تقدم بمقدار 2.7 dB على النظام العادي. بالاضافة الى ذلك تم تقييم اداء هذا النظام مع زيادة سرعة البيانات الى الضعف، وايدت نتائج المحاكاة بانه يحتاج الى 7 dB للحصول على BER 10^{-6}.

https://doi.org/10.30684/etj.2013.82189
2412-0758/University of Technology-Iraq, Baghdad, Iraq

INTRODUCTION

A convolutional code can be represented as a trellis and an encoded information sequence as a unique path through that trellis [1]. Using the received signal, the Viterbi decoder selects the trellis path that most likely represents the transmitted information. This selection is based on comparing metrics associated with each path. If desired signal amplitude and interference variance changes during the encoded frame, these quantities must be estimated at the Viterbi decoder input and incorporated into the metric calculation [2].

It is a well known fact that the complexity of the most common algorithm for decoding convolutional codes, the Viterbi algorithm, grows exponentially with the memory *m* of the encoder. Therefore, the application of the Viterbi algorithm is restricted to memory sizes of up to $m = 10$. If, however, a certain application requires an extremely low error probability, we have to use an encoder with so great an *m,* that the Viterbi algorithm would be hopelessly complex [3].

To overcome this problem, other decoder structures, having optimal soft decisions rather than hard decisions at the demodulator allow the Viterbi decoder to give superior bit error rate performance at the receiver. The idea of soft decisions is to provide the next stage of decoding with some reliability information for the input bits in addition to their probable values. Most commonly, the soft or hard decoding is done using the phase differential of successive symbols [1]. However, it is possible to improve the estimation of the soft bits if the amplitude information contained in successive symbols is also used. The use of soft bits at the input to the Viterbi decoder of a convolutional encoder is well established in the literature. A general theory of soft decision decoding is given in [4] where the author provides a detailed study of the generation and the use of reliability information in convolutional decoders as a modified MAP soft decoding algorithm.

At the receiver, it is highly likely that the soft-decision decoder should be implemented with fixed point arithmetic. Namely, the received signal should be demapped and quantized before sending to the soft decoder [5]. The questions are "what is the optimal demapping arithmetic and how many quantization bits are enough?" Tosato [6] pointed out that the optimal BPSK demapping arithmetic is simple by multiplying the received symbols with the Channel State Information (CSI) coefficients, from the LLR point of view, while we also show that this arithmetic is optimal from the viewpoint of channel capacity. Dasgupta [7] discussed the quantization effect to turbo decoder under Additive White Gaussian Noise (AWGN) channel and pointed out that 3 or 4 bits quantized decoder offers indistinguishable performance from unquantized one. Ohhashi [8] and Jeong [9] derived that 6-bit uniform quantizing in Rayleigh fading channel approaches optimal bit error rate (BER) with proper choice of the clipping level for LDPC and Turbo decoders, respectively.

This paper deals with LLR with soft decision and unquantized mode instead of hard decision of Viterbi algorithm to improve the performance of transceiver with Convolutional Codes, try to increase the speed of data rate and evaluate the performance of the conventional and proposed scheme.

The paper is organized as follows. Section 2 presents the convolutional encoder and decoder using Viterbi algorithm. Section 3 describes a brief overview of LLR Algebra. In

section 4 was to provide a brief description of the system used. Experimental results are presented in Section 5 and finally the conclusion is presented in Section 6.

CONVOLUTIONAL CODES

Encoder

A convolutional code is the set of all codewords generated by a convolutional encoder which is typically a linear sequential circuit, with or without feedback. The important parameters of an encoder are the code rate and the memory. The memory of a convolutional encoder is the number of delay elements in the encoder circuit. A binary convolutional encoder of rate *k/n (k < n)* generates an *n*-bit word for every *k*-bit word at the input [10].

Denote a binary convolutional code by a three-tuple (*n, k, m*), which corresponds to an encoder for which *n* output bits are generated whenever *k* input bits are received, and for which the current *n* outputs are linear combinations of the present *k* input bits and the previous $m \times k$ input bits. Because *m* designates the number of previous *k*-bit input blocks that must be memorized in the encoder, *m* is called the memory order of the convolutional code. Figure (1) illustrates the encoder for the binary *(2, 1, 2)* convolutional code with generators $g_1 = 7$ (octal) and $g_2 = 5$ (octal), where g_i is the generator polynomial characterizing the *i*th output. [11].

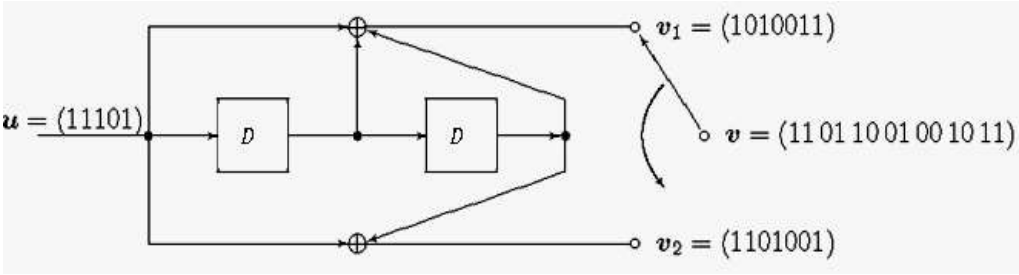

Figure (1) Convolutional Encoder [11].

The information that is input to a rate *k/n* convolutional encoder is a sequence *u* of *k*-tuples [10]:

$$u_t = \left(u_t^1, \ldots\ldots\ldots, u_t^k\right) \qquad \ldots. (1)$$

The coded output of the encoder is a sequence *v* of *n*-tuples:

$$v_t = (v_t^1, \ldots\ldots\ldots, v_t^n) \qquad \ldots. (2)$$

It is also common in coding theory parlance to use the delay element *D* as an operator, thereby representing the input and output sequences as:

$$u(D) = (u_0 + Du_1 + D^2u_2 + \ldots.) \qquad \ldots (3)$$

And

$$v(D) = (v_0 + Dv_1 + D^2 v_2 + \ldots.) \quad \ldots (4)$$

The input and output sequences of the convolutional encoder can also be represented in composite form:

$$u = \left(u_0^1,\ u_0^2, \ldots.., u_0^k;\ u_1^1, u_1^2, \ldots, u_1^k;\ u_2^1, u_2^2, \ldots.., u_2^k, \ldots.\right) \quad \ldots (5)$$

$$v = (v_0^1,\ v_0^2, \ldots.., v_0^n;\ v_1^1, v_1^2, \ldots, v_1^n;\ v_2^1, v_2^2, \ldots.., v_2^n, \ldots.) \quad \ldots (6)$$

A codeword in a convolutional code is an output sequence of the encoder, generated by the multiplication of an input sequence with the generator matrix. It is given by:

$$v(D) = u(D)G(D) \quad \ldots. (7)$$

It is easy to verify that the generator matrix is a matrix. The polynomial generator matrix *G(D)* of the encoder a $k \times n$ *is* given by [10]:

$$G(D) = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \ldots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \ldots & g_2^{(n)}(D) \\ \vdots & \vdots & & \vdots \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \ldots & g_k^{(n)}(D) \end{bmatrix} \quad \ldots (8)$$

Viterbi decoder

Viterbi algorithm is a well-known maximum-likelihood algorithm for decoding of convolutional codes and is an optimal (in a maximum-likelihood sense) algorithm for decoding of a convolutional code using trellis. Its main drawback is that the decoding complexity grows exponentially with the code length. So, it can be utilized only for relatively short codes [12].

A Viterbi algorithm consists of the following three major parts as shown in Figure (2):

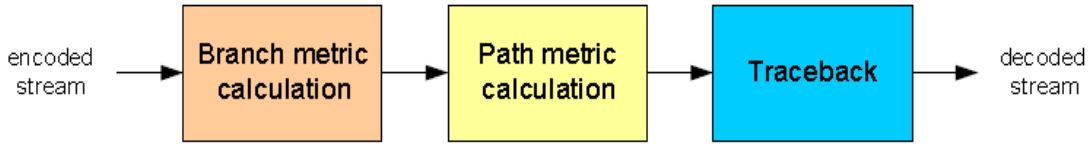

Figure (2) Viterbi decoder data flow [12].

a- **Branch metric calculation**

Methods of branch metric calculation are different for hard decision and soft decision decoders.

For a *hard decision* decoder, a branch metric is a Hamming distance between the received pair of bits and the "ideal" pair. Therefore, a branch metric can take values of 0,

1 and 2. Thus for every input pair we have 4 branch metrics (one for each pair of "ideal" values).

For a *soft decision* decoder, a branch metric is measured using the Euclidean distance. Let x be the first received bit in the pair, y – the second, x_0 and y_0 – the "ideal" values. Then branch metric is.

$$M_b = (x - x_0)^2 + (y - y_0)^2 \qquad \ldots (9)$$

Furthermore, when we calculate 4 branch metric for a soft decision decoder, we don't actually need to know absolute metric values – only the difference between them makes sense. So, nothing will change if we subtract one value from the all four branch metrics:

$$M_b = (x^2 - 2xx_0 + x_0^2) + (y^2 - 2yy_0 + y_0^2) \qquad \ldots. (10)$$

$$M_b^* = M_b - x^2 - y^2 = (x_0^2 - 2xx_0) + (y_0^2 - 2yy_0) \qquad \ldots (11)$$

Note that the second formula, M_b^* , can be calculated without hardware multiplication: x_0^2 and y_0^2 can be pre-calculated, and multiplication of x by x_0 and y by y_0 can be done very easily in hardware given that x_0 and y_0 are constants. It should be also noted that M_b^* is a signed variable and should be calculated in 2's complement format [11].

b-**Path Metric Calculation**

Path metrics are calculated using a procedure called ACS (Add-Compare-Select). This procedure is repeated for every encoder state:

Add – for a given state, we know two states on the previous step which can move to this state, and the output bit pairs that correspond to these transitions. To calculate new path metrics, we add the previous path metrics with the corresponding branch metrics.

Compare, select – we now have two paths, ending in a given state. One of them (with greater metric) is dropped. As there are 2^{K+1} encoder states, we have 2^{K+1} survivor paths at any given time.

It is important that the difference between two survivor path metrics cannot exceed δ log (K−1), where δ is a difference between maximum and minimum possible branch metrics. The problem with path metrics is that they tend to grow constantly and will eventually overflow. But, since the absolute values of path metric don't actually matter, and the difference between them is limited, a data type with a certain number of bits will be sufficient. There are two ways of dealing with this problem: First, since the absolute values of path metric don't actually matter, we can at any time subtract an identical value from the metric of every path. It is usually done when *all* path metrics exceed a chosen threshold (in this case the threshold value is subtracted from every path metric). This method is simple, but not very efficient when implemented in hardware, and the second approach allows overflow, but uses a sufficient number of bits to be able to detect whether the overflow took place or not. The *compare* procedure must be modified in this case.

The whole range of the data type's capacity is divided into 4 equal parts as shown in Figure (3). If one path metric is in the 3-rd quarter, and the other – in the 0-th, then the overflow took place and the path in the 3-rd quarter should be selected. In other cases an ordinary compare procedure is applied. This works, because a difference between path metrics can't exceed a threshold value, and the range of path variable is selected such that it is at least two times greater than the threshold [12].

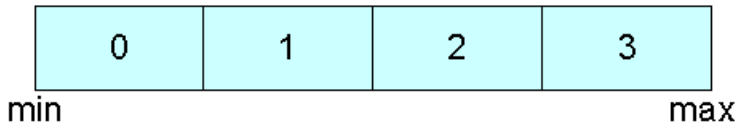

Figure (3) A modulo-normalization approach.

c-Traceback

It has been proven that all survivor paths merge after decoding a sufficiently large block of data *D* on Figure (4), i.e. they differ only in their endings and have the common beginning as illustrate in Figure (5).

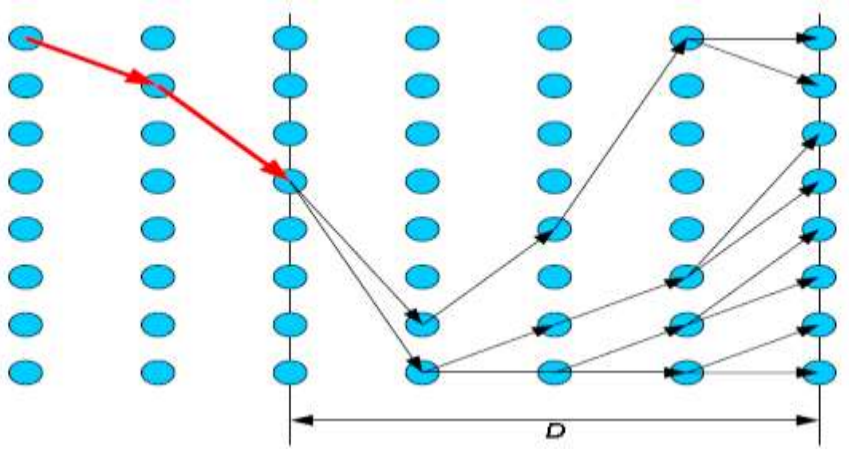

Figure (4) Survivor paths graph example.

If we decode a continuous stream of data, we want our decoder to have finite latency. It is obvious that when some part of path at the beginning of the graph belongs to every survivor path, the decoded bits corresponding to this part can be sent to the output. Given the above statement, we can perform the decoding as follows:

i- Find the survivor paths for $N+D$ input pairs of bits.
ii- *Trace back* from the end of any survivor paths to the beginning.
iii- Send N bits to the output.
iv- Find the survivor paths for another N pairs of input bits.
v- Go to step ii.

In these procedure D is an important parameter called *decoding depth*. A decoding depth should be considerably large for quality decoding, no less than $5K$. Increasing D

decreases the probability of a decoding error, but also increases latency. As for *N*, it specifies how many bits we are sending to the output after each traceback. For example, if *N*=1, the latency is minimal, but the decoder needs to trace the whole tree every step. It is computationally ineffective. In hardware implementations *N* usually equals *D* [11].

LOG-LIKELIHOOD ALGEBRA

The log-likelihood algebra was developed by Hagenauer [13]. Let x be a binary random variable and let $\Pr\{x = x'\}$ denote the probability that the random variable x takes on the value x'. The log-likelihood ratio of x is defined as:

$$L(x) = ln\frac{\Pr\{x=0\}}{\Pr\{x=1\}}. \quad \ldots (12)$$

From the L-value we can calculate the probabilities:

$$\Pr\{x = 0\} = \frac{1}{1+e^{-L(x)}} \quad and \ \Pr\{x = 1\}\frac{1}{1+e^{L(x)}} \quad \ldots (13)$$

If the binary random variable x is conditioned on another random variable y, or on two statistically independent random variables y_1 and y_2, we obtain the conditional L-values:

$$L(x \mid y) = L(y \mid x) + L(x) \quad \ldots (14)$$

$$or \quad L(x \mid y_1, y_2) = L(y_1 \mid x) + L(y_2 \mid x) + L(x) \ldots (15)$$

For the Gaussian channel with binary phase shift keying and signal-to-noise ratio E_S/N_0 we have the a-posteriori L-value:

$$L(b_i \mid r_i) = L(r_i \mid b_i) + L(b_i) = 4\frac{E_S}{N_0}r_i + L(b_i) \ldots (16)$$

where b_i is the transmitted bit and r_i is the received symbol [14].

SYSTEM MODEL

Model that was used in this research consists of two parts, the first being conventional system for the purpose of experimentation and comparison, and the second part for the purpose of testing the proposed system:

Conventional system

Components of such system are shown in Figure (5). The data source block produces the information source for this simulation. The block generates a frame of 9×10^4 random bits to simulate. The Convolutional Encoder block encodes the data from the data source generator. The encoder's constraint length is a scalar since the encoder has one input. The value of the constraint length is the number of bits stored in the shift register, including the current input. In this paper there are six memory registers, and the current input is one bit. Thus the constraint length of the code is 7. The code generator is a 1-by-2 matrix of octal numbers because the encoder has one input and two outputs.

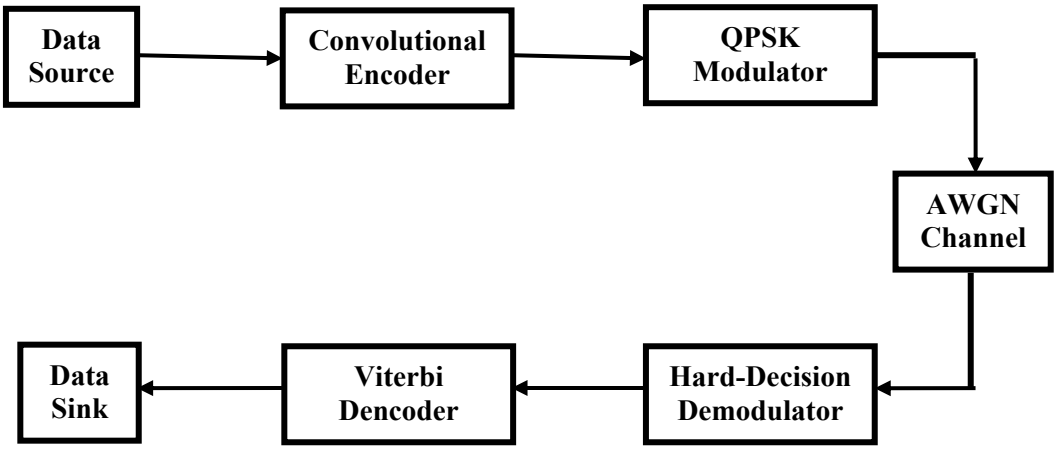

Figure (5) The block diagram of conventional system.

The first element in the matrix indicates which input values contribute to the first output, and the second element in the matrix indicates which input values contribute to the second output. The code generator is [171 133], and the code rate is ½. While the message data entering the Convolutional Encoder block is a scalar bit stream, the encoded data leaving the block is a stream of binary vectors of length 2. The modulator object is to 8 Quaternary PSK (8QPSK) modulate the signal. Set up the PSK modulator to accept bits and employ Gray encoding.

The AWGN Channel block simulates transmission over a noisy channel. The parameters for the block are set as follows: The Mode parameter for this block is set to Signal to noise ratio (Es/No) mode. The Es/No parameter is set to 2 dB. This value typically is changed from one simulation run to the next. The Symbol period is set to 0.5 seconds because the code has rate 1/2. The demodulator object will be used later to compute bitwise hard decision values. Then use of Viterbi decoder is to decode the demodulated symbol stream. This system uses hard decision ('hard') and the continuous decoding option ('cont') which causes a delay in the decoded stream of 32 symbols (traceback length = 32).

Proposed system

This scheme illustrates the improvement in BER performance when using log-likelihood ratio (LLR) instead of hard decision demodulation in a convolutionally coded signal. Such model shown in Fig. 6 simulates a convolutionally coded communication system having one transmitter, an AWGN channel (with same parameters of 4-1 subsection) and two receivers. The modulated signal passes through an additive white Gaussian noise channel. The first receiver has the demodulator configured to compute log-likelihood ratios (LLRs) that are then quantized using a 3-bit quantizer. The modulator object is to Quaternary ASK (QAM) modulate the signal instead of QPSK for previous subsection in order to get good comparison. A Viterbi decoder that is set up for

soft decision decoding processes these quantized values. The LLR values computed by the demodulator are multiplied by -1 to map them to the right quantizer index for use with Viterbi Decoder.

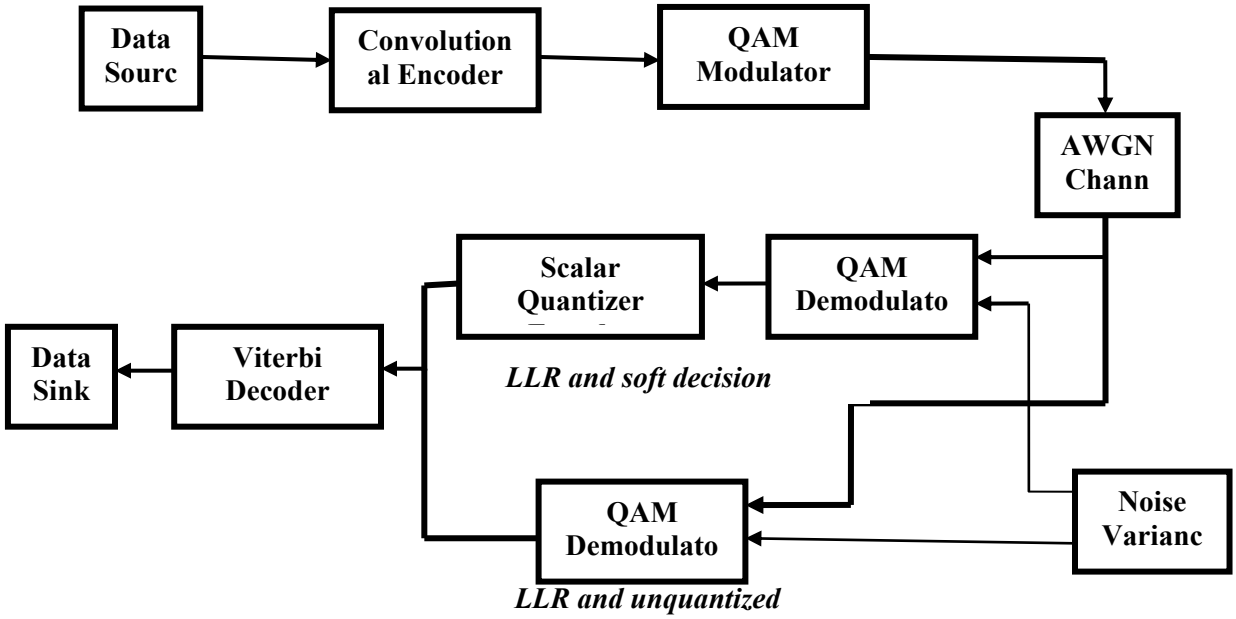

Figure (6) The proposed scheme.

To compute the LLR, the demodulator must be given the variance of noise as seen at its input. The second receiver includes a demodulator that computes LLRs which are processed by a Viterbi decoder that is set up in unquantized mode.

SIMULATIONS AND RESULTS

All steps of the simulation were performed using MATLAB software updated version R2011b. This was done for comparison of two phases, as follows:

i- First start by setting parameters of 4-1 subsection needed for the simulation. It has been used BITERR to compare the original messages to the demodulated messages. BITERR is used to calculate the bit error rate. The error rates are calculated for both the channel and for the decoded bit stream. It demonstrates how to create a convolutional code simulation driver in MATLAB®. Figure (7) shows the conventional system performance. This system needs Eb/No of 6 dB to achieve 10^{-5} BER, although the code used a good specifications of parameters referred to earlier, and the scheme is relatively simpler, but the results are not encouraging.

Ii-The next step simulates the proposed system over (0-6) dB range of Eb/No values by executing the scheme of Fig 6. It plots BER results as they are generated for LLR soft decision of 3 bit quantizer and unquantized model to compare the two situations. A comparison of simulation results with hard decision results is also shown.

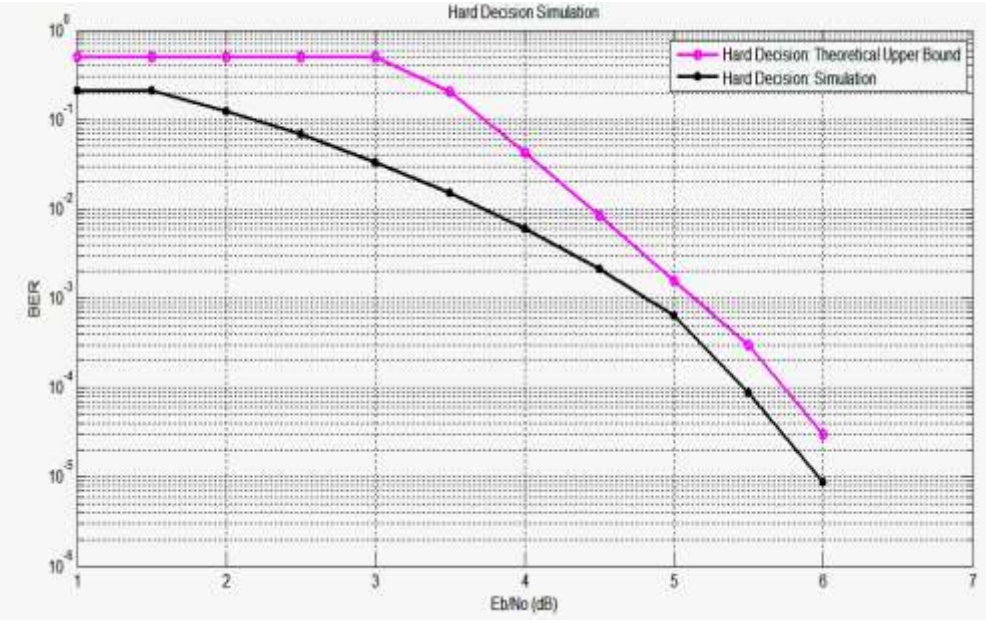

Figure (7) Hard Decision.

Figure (8) illustrates the performance of proposed system. The LLR with soft decision and unquantized scheme needs only 4 dB and 3.25 dB respectively, to achieve 10^{-6} BER. It describes the lead over previous system by about 2 dB for soft decision of LLR and 2.7 dB for unquantized scheme at 10^{-6} BER.

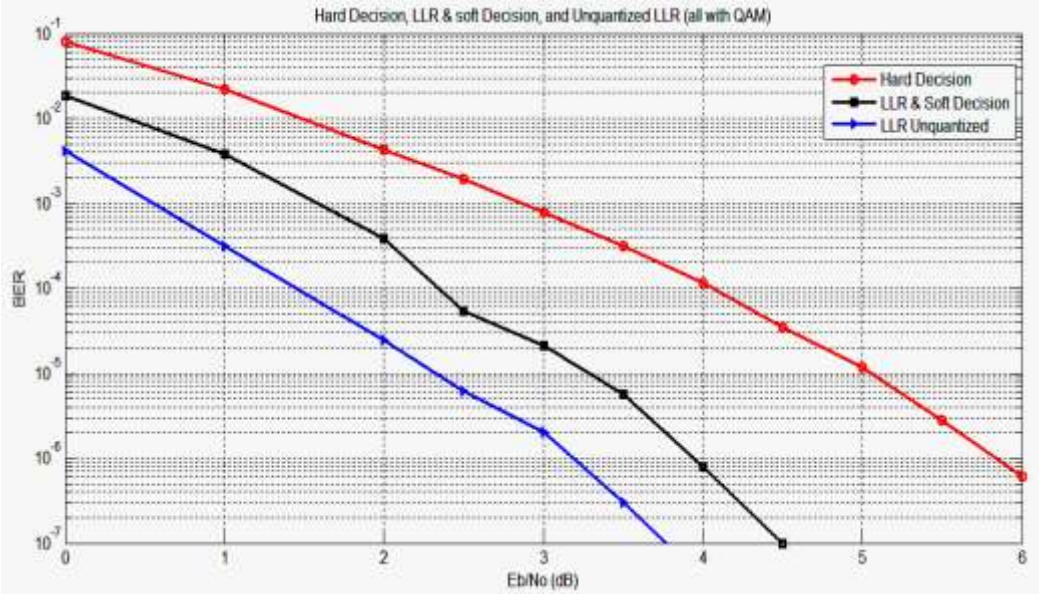

Figure (8) Performance of proposed scheme for QAM modulator.

Iii-For the purpose of evaluating the performance of this system with high speed of data rate, it has been tested over higher level of constellation to show the possibility of this system. Figure (9) shows the performance of such system while keeping all parameters constant except for data speed have increased to double, it was found that the performance of hard decision cannot be adopted because of the high SNR (10 dB at 10^{-6} BER), when it could be to accept his performance by using the proposed structure, and should be noted here that the proposed system increases the complexity of the system but offset by clear improvement in performance. As is evident from Figure (9), the new scheme has achieved a profit of about 2.3 dB for soft decision and up to 3 dB in for the unquantized mod.

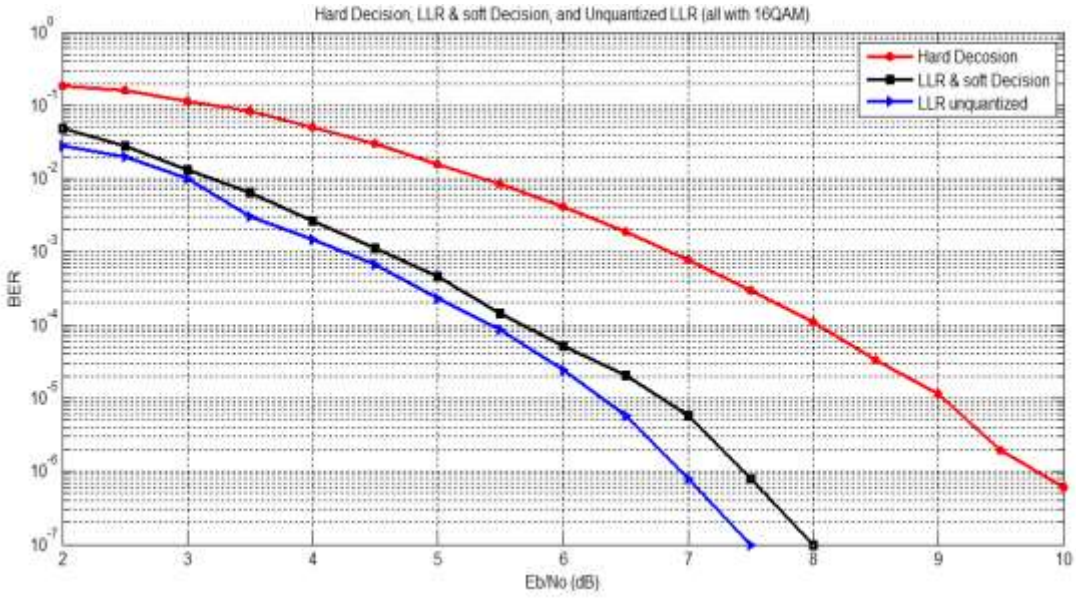

Figure (9) Performance of proposed scheme for 16QAM modulator.

CONCLUSIONS

In this paper it has been evaluate the performance of hard decision viterbi decoder system. Results show that this decoder was not normal level for modern communications. It appeared that is need 6dB to achieve 10^{-5} BER with AWGN channel in the simplest case (QPSK) modulator so that when the amount of data increases will its condition worsens. It cannot increase the level of constellation more than QPSK. But using LLR, it is clearly improved the performance of such decoder. The simulation results of new decoder with soft decision and unquantized mode illustrates that it can reduces the Eb/No to 4 dB and 3.25 dB respectively to achieve 10^{-6} BER which mean that gained more than 2.5 dB. On the other hand when increasing the speed of data rate to double (16QAM), such system needs at least to 7 dB for previous BER, while up to 10 dB for hard decision making such system is very suitable for modern communications.

REFERENCES

[1].Proakis,J. G. "Digital Communications", Mc-Graw Hill, 4 edition, 2001.

[2].Rahnema and Y. Antia, M. "Optimum soft decision decoding with channel state information in the presence of fading," IEEE Communications Magazine, vol. 35, no. 7,pp. 110{111, July 1997.

[3].LIN, s., and COSTELLO, D.J., JII.: "Error control coding – Fundamentals and applications" 1st edn., Prentice-Hall, Englewood Cliffs, NJ, 1983

[4].Wang, X. A. and Wicker, S. B., "A soft-output decoding algorithm for concatenated systems," IEEE Transactions on Information Theory, vol. 42(2), pp. 543-553, 1996.

[5].Qiuliang Xie, Kewu Peng, Jian Song, Changyong Pan and Zhixing Yang, "Soft-Qiuliang Xie, Kewu Peng, Jian Song, Changyong Pan and Zhixing Yang, "Soft-Quantized Demodulation for BPSK over Discrete-Time Memoryless Fadi ng Channel", in IEEE International Conference on Communica-tions, 2008, PP. 855-859.

[6].Tosato and P. Bisaglia, F. "Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2," in IEEE International Conference on COFDM with application to HIPERLAN/2," in IEEE International Conference on Communica-tions,(ICC'02), 2002, pp. 664–668.

[7].Dasgupta and C. N. Georghiades, U. "Turbo decoding of quantized data," *IEEE* Trans. On Communications, vol. 50, pp. 56–64, Jan. 2002.

[8]. Ohhashi and T. Ohtsuki,A. "Performance of low-density parity-check (LDPC) code with UMP BP-based algorithm and quantizer on Rayleigh fading channels," in Vehicular Technology Conference, (VTC'03), pp. 1881–1885, Apr. 2003.

[9]. Jeong and D. Hsia, "Optimal quantization for soft-decision turbo decoder," in Vehicular Technology Conference,(VTC'99), pp. 1620–1624, Sept. 1999.

[10]. Sudharshan Srinivasan, and Steven S. Pietrobon, "Decoding of High Rate Convolutional Codes Using the Dual Trellis", *IEEE Transactions on Information Theory,* Vol. 56, No. 1, Jan. 2010, PP.273-294.

[11]. Lin, S. D.J. Costello, "*Error Control Coding"*, Pearson Prentice Hall, USA, 2004.

[12]. Rajashri Khanai, Dr. G. H. Kulkarni, "Performance Analysis of Conventional Crypto-coding", *International Journal of Latest Trends in Computing, Volume 2, Issue 1, March 2011*

[13]. Hagenauer, J. Offer, E. and Papke, L. Iterative decoding of binary block and convolutional codes. IEEE Transactions on Information Theory, 42, 429–445, 1996.

[14]. Andr´e Neubauer, "*Coding Theory Algorithms, Architectures, and Applications*", John Wiley & Sons Ltd, England, 2007.