DOI: 10.21917/ijct.2012.0069

STACK DECODING OF LINEAR BLOCK CODES FOR DISCRETE MEMORYLESS CHANNEL USING TREE DIAGRAM

H. Prashantha Kumar[1], U. Sripati[2], K. Rajesh Shetty[3] and B. Shankarananda[4]

[1,2,3]*Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, India*

E-mail: [1]hprashanthakumar@gmail.com, [2]sripati_acharya@yahoo.co.in and [3]krshetty_nitte@yahoo.co.in

[4]*Vivekananda Institute of Technology, India*

E-mail: bsnanda2000@gmail.com

Abstract

The boundaries between block and convolutional codes have become diffused after recent advances in the understanding of the trellis structure of block codes and the tail-biting structure of some convolutional codes. Therefore, decoding algorithms traditionally proposed for decoding convolutional codes have been applied for decoding certain classes of block codes. This paper presents the decoding of block codes using tree structure. Many good block codes are presently known. Several of them have been used in applications ranging from deep space communication to error control in storage systems. But the primary difficulty with applying Viterbi or BCJR algorithms to decode of block codes is that, even though they are optimum decoding methods, the promised bit error rates are not achieved in practice at data rates close to capacity. This is because the decoding effort is fixed and grows with block length, and thus only short block length codes can be used. Therefore, an important practical question is whether a suboptimal realizable soft decision decoding method can be found for block codes. A noteworthy result which provides a partial answer to this question is described in the following sections. This result of near optimum decoding will be used as motivation for the investigation of different soft decision decoding methods for linear block codes which can lead to the development of efficient decoding algorithms. The code tree can be treated as an expanded version of the trellis, where every path is totally distinct from every other path. We have derived the tree structure for (8, 4) and (16, 11) extended Hamming codes and have succeeded in implementing the soft decision stack algorithm to decode them. For the discrete memoryless channel, gains in excess of 1.5dB at a bit error rate of 10^{-5} with respect to conventional hard decision decoding are demonstrated for these codes.

Keywords:

Extended Hamming Codes, Tree Diagram, Soft Decision Decoding, Discrete Memoryless Channel, Fano Metric

1. INRODUCTION

Error control coding (ECC) is commonly used to achieve reliable transmission of information. Channel codes enable a decoder to recover from errors produced by noise in a communication channel. Codes ensure higher noise tolerance at the receiver by adding redundancy into the user data to achieve better separation of data sequences. ECC algorithms have constituted a significant enabler in the telecommunications revolution, the internet, digital recording and space exploration. The past decade has seen tremendous growth in availability and deployment of wireless services. This has been made possible by development of powerful signal processing algorithms to ensure efficient spectral usage/error free communication and development of hardware platforms on which these algorithms could be run. Thus developments in algorithm design and microelectronics have gone hand in hand to create the infrastructure for the information revolution that has transformed the way in which human beings live and work [1]. Error correcting codes can be divided into two classes according to the manner in which redundancy is added to the messages: block and convolutional. Block codes implement a one-to-one mapping of a set of k information symbols on to a set of n codeword symbols. We call this code as an (n, k) linear block code. The n-k symbols in a codeword are a function of the information symbols, and provide redundancy that can be used for error correction and/or detection purposes. The minimum distance d_{min} of a block code **C** is the smallest Hamming distance between any two codewords in the code.

Both types of coding schemes have found practical applications. Historically convolutional codes have been preferred, apparently because of the availability of the soft-decision Viterbi decoding algorithm and the belief over many years that block codes could not be efficiently decoded with soft-decisions. The main problem is the fundamentally algebraic structure of block codes. Although this structure allows elegant algebraic decoding techniques to be applied when hard decisions are made, the reliance on finite field arithmetic for decoding makes it difficult to exploit soft decisions. The break through which enabled the possibility of using soft decision decoding (SDD) for decoding block codes was provided by [2] who showed that any linear block codes can be represented by a trellis, and that the Viterbi algorithm can therefore be used for soft decision decoding. For example, a (5, 4) parity check code is represented by the parity check matrix,

$$\mathbf{H} = [1\ 1\ 1\ 1\ 1]$$

Its syndrome trellis is shown in Fig.1.

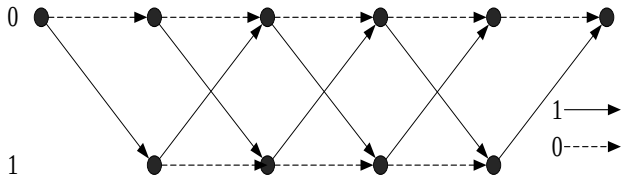

Fig.1. Trellis representation of a (5, 4) parity check code

It is interesting to note that there is no need to label the branches with the coded bits. A transition between two states with the same level corresponds to coded bit 0. Linear block codes have trellises with a time-varying number of states. This trellis is simple and has a regular structure. The minimum number of states can be quite large, for example, 2^{64} for the (128, 64) extended BCH code [3]. Although a certain permutation of the code achieves 2^{43} states, which is still exceedingly large for practical implementations of the Viterbi or

Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [4]. In spite of exponential increase in computational complexity, the soft decision decoding using trellis diagram performs 2 to 3dB better than hard decision decoding (HDD) over additive white Gaussian noise (AWGN) channel. This much amount of coding gain is very significant. 3dB of coding gain can reduce the required bandwidth by 50% or increase data throughput by a factor of 2 or increase range by 40% or reduce antenna size by 30% or reduce transmitter power by a factor of 2. Therefore collectively we can say that coding gain increases the system performance or reduces cost or both [5]. Since SDD increases the error correcting capability of the code by correcting more number of soft errors and henceforth increases the coding gain compare to HDD. The potential of SDD over HDD is illustrated in Fig.2 for the (5, 4) single parity check code. From the graph we conclude that at bit error rate (BER) of 10^{-5}, SDD using Viterbi algorithm performs 2.3dB better than HDD.

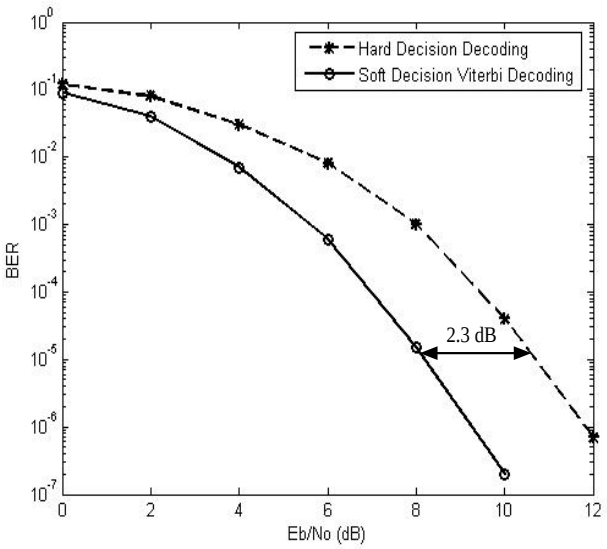

Fig.2. Performance for the (5, 4) block code with HDD and SDD over AWGN channel using Viterbi algorithm

2. TREE REPRESENTATION OF SYSTEMATIC LINEAR BLOCK CODES

It is well known that the fixed amount of computation required by the Viterbi algorithm is not always needed, particularly when the noise is light or signal to noise ratio is high [6]. For example, assume that an (n, k) linear block code is transmitted without error over a channel. The Viterbi algorithm will still perform on the order of $2^{\min\{k,\ n-k\}}$ computations per decoded information block, all of which is wasted effort in this case. In other words, it is often desirable to have a decoding procedure whose effort is adaptable to the noise level. Sequential decoding using tree diagram is such a type of algorithm. Sequential decoding describes any algorithm for decoding channel codes which successively explores the code tree by moving to new nodes from an already explored node. The purpose of tree searching algorithms is to search through the nodes of the code tree in efficient way, that is, without having to examine too many nodes, in an attempt to find the maximum likelihood path. Each node examined represents a path through part of the tree. Whether a particular path is likely to be part of the maximum likelihood path depends on the metric value associated with that path. The metric is a measure of the "closeness" of a path to the received sequence [7]. Every linear block code can be represented graphically by means of a tree. Fig.3 represents general tree representation for an (n, k) systematic linear block code.

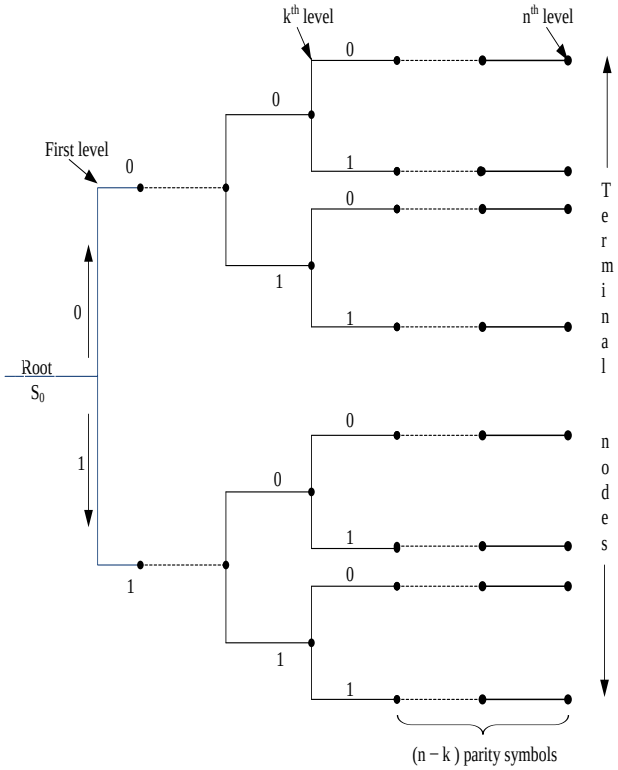

Fig.3. Tree representation of a binary (n, k) block code

This tree has the following structures:

1) Tree consists of $n + 1$ levels.
2) For $0 \le i \le k$, there are 2^i nodes at the i^{th} level of the tree. There is only one node s_0 at the zeroth level of the tree called the initial node (or root) of the tree, and there are 2^k nodes at the n^{th} level of the tree, which are called the terminal node of the tree.
3) For $0 \le i \le k$, there are two branches leaving every node s_i at level -i and connecting to two different nodes at level –(i+1). One branch is labeled with an information symbol 0, and the other branch is labeled with an information symbol 1. For $k \le i \le n$, there is only one branch leaving every node s_i at level -i and connecting to one node at level – (i+1). This branch is labeled with a parity check symbol, either 0or 1.
4) The label sequence of path connecting the initial node s_0 to a node s_k at the k^{th} level corresponds to an information sequence **m** of k bits. The label sequence of the path connecting the initial node s_0 through a node s_k at the k^{th} level to a terminal node s_n at the n^{th} level is a codeword **C**. The label sequence of the tail connecting

node s_k to node s_n corresponds to the *n-k* parity check symbols of the codeword.

The generator matrix **G** of an (8, 4) extended Hamming code with minimum Hamming distance d_{min}=4 in systematic form is given below.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

The above generator matrix generates all possible valid codewords in systematic form. Fig.4 shows the tree representation of a binary extended Hamming code generated by **G.**

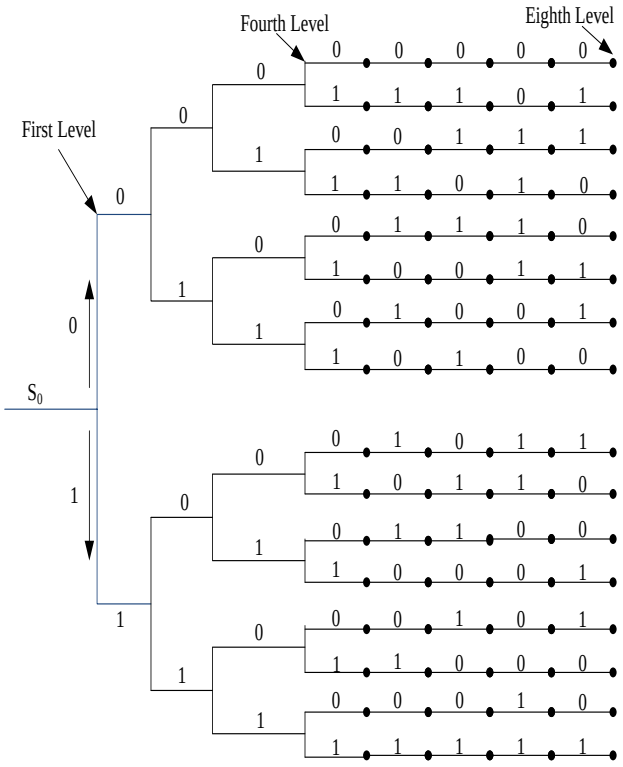

Fig.4. Tree representation of an (8, 4) Hamming code

3. DECODING WITH THE STACK ALGORITHM

The tree representation of a linear block code can be used to facilitate stack decoding. In the Zigangirov - Jelinek (ZJ) or stack algorithm, an ordered list or stack of previously examined paths of different lengths is kept in storage. Each stack entry contains a path along with its metric, the path with the largest metric is placed on top, and the others are listed in order of decreasing metric. Each decoding step consists of extending the top path in the stack by computing the Fano metrics of its succeeding branches and then adding these to the metric of the top path to form new paths, called the successors of the top path. The top path is then deleted from the stack, its successors are inserted and the stack is rearranged in order of decreasing metric values. When the top path in the stack has highest Fano metric and also it is the end of tree, the algorithm terminates.

Stack algorithm commonly uses a probabilistic branch metric, namely, the Fano metric, which can be written for a continuous (or Gaussian) channel as [3]

$$M(\eta \mid v_l) = -\log_2[1+\exp(-4\frac{(2v_l-1)r_l\sqrt{Es}}{N_0})] \qquad (1)$$

where, $M(r_1|v_1)$ is the branch metric for the l^{th} branch, E_s is the energy per transmitted bit and N_0 is the one-sided noise power density. For a discrete memoryless channel with a uniformly distributed source and a crossover probability *p*, the above Fano metric reduces to

$$M(\eta|v_l) = \log_2 p(\eta|v_l) - \log_2 p(\eta) - R \qquad (2)$$

Here, *R* is the rate of the code in use, $p(r_1|v_1)$ is the channel transition probability of the received symbol r_1 given the transmitted symbol v_1, $p(r_1)$ is a channel output symbol probability [8]. Fano's original selection of this metric was based on a heuristic argument, and on occasion other researchers/designers have used other metrics.

We assume a binary phase shift keying (BPSK) modulation, where the bits $c_i \in\{0, 1\}$ are mapped to the transmission bits $x_i\in\{+1, -1\}$ corresponding to the relation

$$x_i = (-1)^{c_i}; i \in [1, n] \qquad (3)$$

After transmission over the AWGN channel, we obtain the probability distribution depicted in Fig.5.

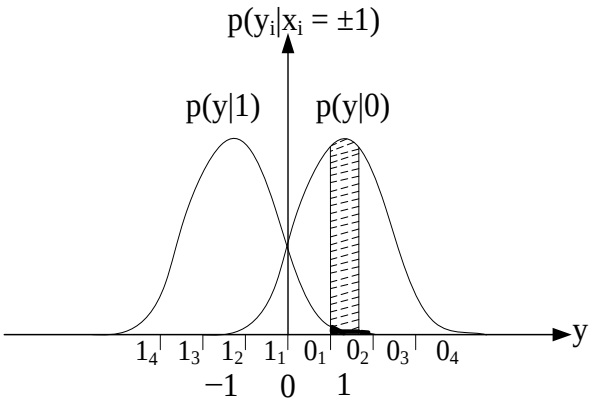

Fig.5. PDF for received symbol **y**

We assume that the *y*-axis in previous figure is divided in to intervals of width Δy. In practical systems, this value is often quantized. In our decoder analysis, the received signal is quantized to 3 bits, resulting in 2^3 different quantization levels, using uniformly spaced quantization thresholds [9]. The block interprets 0_4 as the most confident decision that the codeword bit is a 0 and interprets 1_4 as the most confident decision that the codeword bit is a 1. The values in between these represent less confident decisions. Thus a binary input, continuous valued output has changed to 8-ary DMC. Below figure shows 8-level soft quantized DMC.

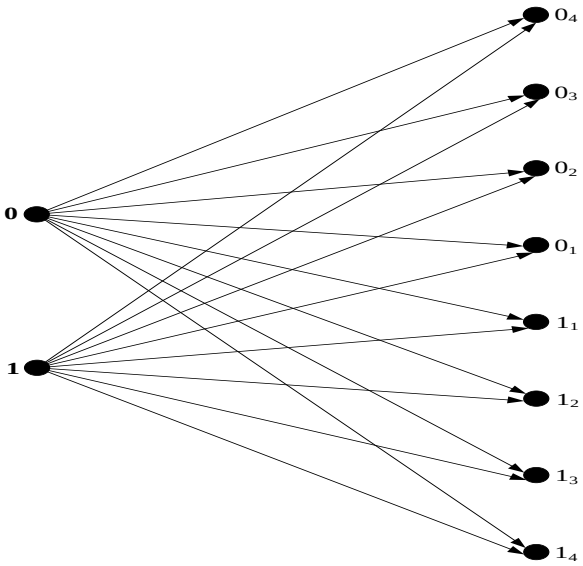

Fig.6. Binary input 8-ary output DMC

We now proceed to a description of the stack decoding algorithm by means of a set of rules for updating the stack entries and for backward or forward translations through the tree.

Step 1: Load the stack with the origin node in the tree, whose metric is taken to be zero.

Step 2: Compute the metrics of the successors of the top path in the stack.

Step 3: Delete the top path from the stack.

Step 4: Insert the new paths in the stack and rearrange the stack in order of decreasing metric values.

Step 5: If the top path in the stack ends at terminal node in the tree with highest metric value, stop. Otherwise, return to Step 2.

When the algorithm terminates, the top path with highest metric in the stack is taken as the decoded path. We present an example to illustrate these ideas.

Example 1:

A binary (8, 4) extended Hamming code associated with the tree in Fig.4 is used to encode the information sequence x=(0000), resulting in the codeword v = (00000000). This codeword is transmitted over the binary input, 8-ary output DMC with transition probabilities $p(r|v)$ given by the entries [10] shown in Table.1.

Table.1. Transition probabilities $p(r|v)$ for binary input, 8-ary output DMC

v \ r	0_4	0_3	0_2	0_1	1_1	1_2	1_3	1_4
0	0.434	0.197	0.167	0.111	0.058	0.023	0.008	0.002
1	0.002	0.008	0.023	0.058	0.111	0.167	0.197	0.434

The sequence r = $0_4 0_4 0_4 1_1 1_1 1_1 0_4 0_4$ is received. Error bits (4^{th}, 5^{th}, and 6^{th} position) are shown in dark. Using Eq.(2), the Fano metric is computed as follows.

Table.1(a). Fano metrics for binary input, 8-ary output DMC

v \ r	0_4	0_3	0_2	0_1	1_1	1_2	1_3	1_4
0	0.49	0.44	0.31	-0.11	-1.04	-2.55	-4.18	-7.27
1	-7.27	-4.18	-2.55	-1.04	-0.11	0.31	0.44	0.49

The metrics are scaled by 9/0.49 to obtain integer metrics as shown below.

Table.1(b). Scaled Fano metrics

v \ r	0_4	0_3	0_2	0_1	1_1	1_2	1_3	1_4
0	9	8	6	-2	-18	-46	-75	-131
1	-131	-75	-46	-18	-2	6	8	9

Use this Fano metric and apply stack algorithm to obtain the transmitted codeword. The results are shown in Table.2.

Table.2. Decoding steps for the stack algorithm

Step 1 0(9) 1(-131)	**Step 2** 00(18) 01(-122) 1(-131)	**Step 3** 000(27) 001(-113) 01(-122) 1(-131)
Step 4 0001(25) 0000(9)	**Step 5** 00011(23) 0000(9) 001(-113) 01(-122) 1(-131)	**Step 6** 000111(21) 0000(9) 001(-113) 01(-122) 1(-131) contd.,
Step 7 0001110(30) 0000(9) 001(-113) 01(-122) 1(-131)	**Step 8** 0000(9) 00011101(-101) 001(-113) 01(-122) 1(-131)	**Step 9** 00000(-9) 00011101(-101) 001(-113) 01(-122) 1(-131)
Step 10 000000(-27) 00011101(-101) 001(-113) 01(-122) 1(-131)	**Step 11** 0000000(-18) 00011101(-101) 001(-113) 01(-122) 1(-131)	**Step 12** 00000000(-9)→**Successful Decoding** 00011101(-101) 001(-113) 01(-122) 1(-131)

Here we can note that stack algorithm corrected three bit soft errors. We have checked the algorithm exhaustively for many error patterns of this kind and found that stack decoder corrected all of them. Hence, by employing this approach, we were able to correct many error patterns of weight exceeding the error correcting capability (hard decision) of the code.

Simulations have been performed by employing eight level soft quantization and known channel state information (CSI). Simulation results (Fig.7 and Fig.8) quantify the bit error rate (BER) for HDD and stack decoding of (8, 4) and (16, 11) Hamming codes.

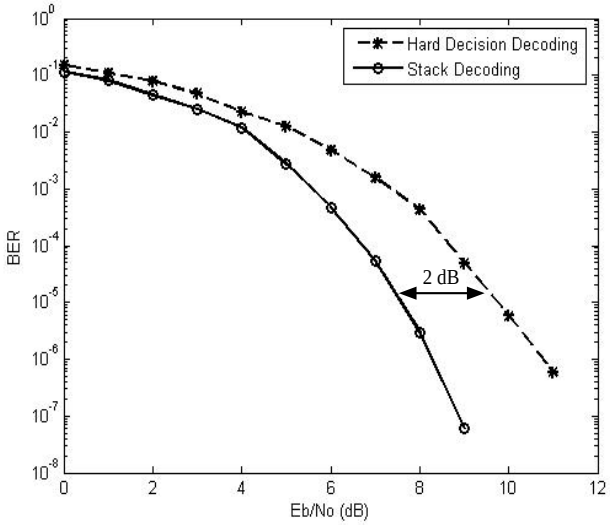

Fig.7. Performance for the (8, 4) extended Hamming code over AWGN with HDD and Stack decoding

For an (8, 4) Hamming code, a BER of 10^{-5} using HDD requires an SNR of 9.8dB while for the soft decision stack algorithm, the same BER is achieved with 7.8dB. Hence, soft decision stack decoding performs 2dB better than HDD.

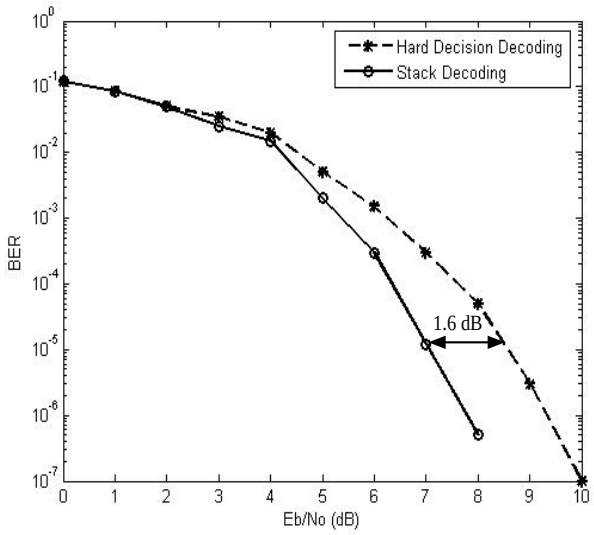

Fig.8. Performance for the (16, 11) extended Hamming code over AWGN with HDD and Stack decoding

In a similar manner, it is observed that the (16, 11) Hamming code with HDD achieves a BER of 10^{-5} at an SNR of 8.6dB while the soft decision stack algorithm achieves the same BER at an SNR of 7dB. Thus, soft decision stack decoding performs 1.6dB better than HDD.

4. CONCLUSION

A simple, efficient and near optimal decoding scheme for linear block codes using tree representation has been proposed in this paper. It is interesting to notice that the technique proposed in this paper can be used to decode any linear block code. Sequential decoding schemes have some drawbacks (such as variable decoding effort) that are well known in the context of decoding of convolutional codes. Since block codes have a finite tree, the average number of computations and the decoding effort are always bounded. Very noisy received sequences typically require a large number of computations with a stack decoder, sometimes more than the fixed number of computations required by the Viterbi algorithm; however, since very noisy received sequences do not occur very often, the average number of computations performed by a stack decoder is normally much less than fixed number performed by the Viterbi algorithm. It is well known that the (8, 4) single bit error correcting extended Hamming code with hard decision decoding corrects only a single bit error over the span of the codeword, while soft decision stack decoding corrects many three bit patterns of soft errors. This in turn results in a coding gain for transmission over the AWGN channel when compared to HDD.

One of the main challenges in adoption and deployment of wireless networked sensing applications is ensuring reliable sensor data collection and aggregation, while satisfying the low cost, low energy operating constraints typical of such applications. A wireless sensor network is inherently vulnerable to different sources of unreliability due to transient failures in circuits and communication channels. The sources of unreliability can be classified into two categories: (1) faults that change behavior permanently, and (2) failures that lead to transient deviations from normal behavior, termed as soft failures [11]. Hence it is necessary to provide a proper error control scheme to reduce the bit error rate in such applications.

Asymmetric codes with low encoding complexity (encoding is usually performed at sensor nodes which are simple in construction and have power constraint), possessing modest error correcting capability (because of low data rates and proximity between transmitter and receiver) with moderately high decoding complexity (decoding is usually done at the base station which has greater resources than the sensor nodes) are employed. The codes along with the decoding schemes proposed in this paper are well suited to meet this requirement.

REFERENCES

[1] Daniel J. Costello and G. David Forney, "Channel coding: The road to channel capacity", *Proceedings of IEEE*, Vol. 95, No. 6, pp. 1150-1177, 2007.

[2] J.K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis", *IEEE Transactions on Information Theory*, Vol. 24, No. 1, pp. 76-80, 1978.

[3] Vladislav Sorokine and Frank R. Kschischang, "A Sequential Decoder for Linear Block Codes with a Variable Bias-Term Metric", *IEEE Transactions on Information Theory*, Vol. 44, No. 1, pp. 410-416, 1998.

[4] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Transactions on Information Theory*, Vol. IT-20, pp. 284-287, 1974.

[5] B. Thomson, "*Improving bandwidth utilization with turbo product codes*", available for download at - www.ewh.ieee.org/r6/scv/comsoc/0009.pdf

[6] Richard E. Blahut, "*Algebraic Codes for Data Transmission*", First Edition, Cambridge University Press, 2003.

[7] Stephen B. Wicker, "*Error Control Systems for Digital Communication and Storage*", First Edition, Prentice Hall, 1995.

[8] Shu Lin and Daniel J. Costello, "*Error Control Coding*", Second Edition, Prentice Hall, 2004.

[9] Wu-Hsiang J. Chen. *et al*., "Quantization issues for soft decision decoding of linear block codes", *IEEE Transactions on Communications*, Vol. 47, No. 6, pp. 789-795, 1999.

[10] R. Johannesson and K. Zigangirov, "*Fundamentals of Convolutional Coding*", First Edition, Wiley-IEEE Press, 2001.

[11] Ivan Stojmenovic, "*Handbook of Sensor Networks: Algorithms and Architectures*", First Edition, Wiley-Interscience, 2005.