Volume 2, Issue 11, November 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Error Detection and Correction: An Introduction

[1] Vikas Gupta
[1]Ph. D Research Scholar
Pacific Academy of Higher Education
And Research University, Udaipur
Rajasthan, India

[2] Dr. Chanderkant Verma
[2] Associate Professor
Department of Computer Applications
Kurushetra University, Kurushetra
Haryana, India

Abstract - In most communication system whether wired or wireless convolutional encoders are used and AWGN introduces errors during transmission. Various error correcting and controlling mechanisms are present. In this paper all mechanisms are studied and best mechanism on the basis of accuracy, complexity and power consumption is selected. There should be trade off between complexity of hardware and power consumption in decoder.

Key Words- FEC, Block Codes, Convolutional Codes, ARQ, HARQ, Viterbi Mechanism.

I. Introduction

Unlike wired digital networks, wireless digital networks are much more prone to bit errors. Packets of bits that are received are more likely to be damaged and considered unusable in a packetized system. Error detection and correction mechanisms are vital and numerous techniques exist for reducing the effect of bit-errors and trying to ensure that the receiver eventually gets an error free version of the packet. The major techniques used are error detection with Automatic Repeat Request (ARQ), Forward Error Correction (FEC) and hybrid forms of ARQ and FEC (H-ARQ).

Forward Error Correction (FEC) is the method of transmitting error correction information along with the message. At the receiver, this error correction information is used to correct any bit-errors that may have occurred during transmission. The improved performance comes at the cost of introducing a considerable amount of redundancy in the transmitted code. There are various FEC codes in use today for the purpose of error correction. Most codes fall into either of two major categories: block codes [11] and convolutional codes [6]. Block codes work with fixed length blocks of code. Convolutional codes deal with data sequentially (i.e. taken a few bits at a time) with the output depending on both the present input as well as previous inputs. In terms of implementation, block codes become very complex as their length increases and are therefore harder to implement. Convolutional codes, in comparison to block codes, are less complex and therefore easier to implement. In packetized digital networks convolutionally coded data would still be transmitted as packets or blocks. However these blocks would be much larger in comparison to those used by block codes. The design of error correcting codes and their corresponding decoders is usually done in isolation. The code is often designed first with the goal of minimizing the gap from Shannon capacity [1] and attaining the target error probability. To reflect the concerns of implementation, the code is usually chosen from a family of codes that can be decoded with low "complexity" [2]. On the implementation side, decoders are carefully designed (see e.g. [3]) for the chosen code with the goal of consuming low power while achieving the required decoding throughput2. This "division of labor" has been extremely successful and forms the paradigm behind many modern long-distance communication system designs.

Shannon-theoretic limits, complemented by modern coding-theoretic constructions [3], have provided codes that are provably good for minimizing transmit power. Can we develop a parallel approach in order to minimize the total system power? With simplistic encoding/decoding models, the issue of fundamental limits on total (transmit +encoding+ decoding) power has been addressed in some recent works [4], [5], [6], [7]. These fundamental limits abstract power consumed in computational nodes [5], [6] and wiring in the encoder/decoder implementation and can provide insights into the choice of the code and its corresponding decoding algorithm. While such theoretical insights can serve to guide the choice of the code family, the simplicity of these theoretical models, which (to an extent) is needed in order to be able to obtain fundamental bounds, also limits their applicability. Even if the models are refined further, the large-deviations techniques used [5], [9] are usually tight only in asymptopia. Thus, at reasonably high error probability (e.g. 10^{-6}) and small distances (e.g. less than five meters), it is unlikely that the bounds themselves can be used to give precise answers on what codes to use. Given the limitations of the fundamental bounds, how do we search for a total-power-efficient code & decoder? After all, for a given block length, there are super-exponentially many possible codes. Further, for each code, there are many possible decoding algorithms. Even when the code and its corresponding decoding algorithm are fixed, there are many possible implementation architectures. Even today, the design and optimized implementation of just a single decoder requires significant effort. It is therefore infeasible to implement and measure the power consumption of every code and decoder in order to determine the best combination.

Ahead in paper various error correction and detection techniques are discussed and best decoding technique on the basis of complexity and power efficiency has been selected for our future work.

II. Forward Error Correction (FEC)

Forward Error Correction is a method used to improve channel capacity by introducing redundant data into the message [8]. This redundant data allows the receiver to detect and correct errors without the need for retransmission of the message. Forward Error Correction proves advantageous in noisy channels when a large number of retransmissions would normally be required before a packet is received without error. It is also used in cases where no backward channel exists from the receiver to the transmitter. A complex algorithm or function is used to encode the message with redundant data. The process of adding redundant data to the message is called channel coding. This encoded message may or may not contain the original information in an unmodified form. Systematic codes are those that have a portion of the output directly resembling the input. Non-systematic codes are those that do not. It was earlier believed that as some degree of noise was present in all communication channels, it would not be possible to have error free communications. This belief was proved wrong by Claude Shannon in 1948. In his paper [8] titled "A Mathematical Theory of Communication", Shannon proved that channel noise limits transmission rate and not the error probability. According to his theory, every communication channel has a capacity C (measured in bits per second), and as long as the transmission rate, R (measured in bits per second), is less than C, it is possible to design an error-free communications system using error control codes. The now famous Shannon-Hartley theorem, describes how this channel capacity can be calculated. However, Shannon did not describe how such codes may be developed. This led to a wide spread effort to develop codes that would produce the very small error probability as predicted by Shannon. There were two major classes of codes that were developed, namely block codes and convolutional codes.

III. Block Code

As described by Proakis [10], linear block codes consist of fixed length vectors called code words. Block codes are described using two integers k and n, and a generator matrix or polynomial. The integer k is the number of data bits in the input to the block encoder. The integer n is the total number of bits in the generated codeword. Also, each n bit codeword is uniquely determined by the k bit input data.

Another parameter used to describe is its weight. This is defined as the number of non zero elements in the code word. In general, each code word has its own weight. If all the M code words have equal weight it is said to be fixed-weight code [11]. Hamming Codes and Cyclic Redundancy Checks are two widely used examples of block codes. They are described below.

A. Hamming Codes

A commonly known linear Block Code is the Hamming code. Hamming codes can detect and correct a single bit-error in a block of data. In these codes, every bit is included in a unique set of parity bits [12]. The presence and location of a single parity bit-error can be determined by analyzing parities of combinations of received bits to produce a table of parities each of which corresponds to a particular bit-error combination. This table of errors is known as the error syndrome. If all parities are correct according to this pattern, it can be concluded that there is not a single bit-error in the message (there may be multiple bit-errors). If there are errors in the parities caused by a single bit-error, the erroneous data bit can be found by adding up the positions of the erroneous parities. While Hamming codes are easy to implement, a problem arises if more than one bit in the received message is erroneous. In some cases, the error may be detected but cannot be corrected. In other cases, the error may go undetected resulting in an incorrect interpretation of transmitted information. Hence, there is a need for more robust error detection and correction schemes that can detect and correct multiple errors in a transmitted message.

B. Cyclic codes and Cyclic Redundancy Checks (CRC)

Cyclic Codes are linear block codes that can be expressed by the following mathematical property.

If C = [c n-1 cn-2 … c1 c0] is a code word of a cyclic code, then [c n-2 cn-3 … c0 cn-1], which is obtained by cyclically shifting all the elements to the left, is also a code word [11]. In other words, every cyclic shift of a codeword results in another codeword. This cyclic structure is very useful in encoding and decoding operations because it is very easy to implement in hardware. A cyclic redundancy check or CRC is a very common form of cyclic code which is used for error detection purposes in communication systems. At the transmitter, a function is used to calculate a value for the CRC check bits based on the data to be transmitted. These check bits are transmitted along with the data to the receiver. The receiver performs the same calculation on the received data and compares it with the CRC check bits that it has received. If they match, it is considered that no bit-errors have occurred during transmission. While it is possible for certain patterns of error to go undetected, a careful selection of the generator function will minimize this possibility. Using different kinds of generator polynomials, it is possible to use CRC's to detect different kinds of errors such as all single bit-errors, all double bit errors, any odd number of errors, or any burst error of length less than a particular value. Due to these properties, the CRC check is a very useful form of error detection. The IEEE 802.11 standard for CRC check polynomial is the CRC-32 [13].

IV. Convolutional Codes

Convolutional codes are codes that are generated sequentially by passing the information sequence through a linear finite-state shift register. A convolutional code is described using three parameters k, n and K. The integer k represents the number of input bits for each shift of the register. The integer n represents the number of output bits generated at each shift of the register. K is an integer known as constraint length, which represents the number of k bit stages present in the encoding shift register [11]. Each possible combination of shift registers together forms a possible state of the encoder. For a code of constraint length K, there exist 2K-1 possible states. Since convolutional codes are processed sequentially, the encoding process can start producing encoded bits as soon as a few bits have been processed and then carry on producing bits for as long as required. Similarly, the decoding process can start as soon as a few bits have been received. In other words, this means is that it is not necessary to wait for the entire data to be received before decoding is started. This makes it ideal in situations where the data to be transmitted is very long and possibly even endless, e.g.: phone conversations.

In packetized digital networks, even convolutional codes are sent as packets of data. However, these packet lengths are usually considerably longer than what would be practical for block codes. Additionally, in block codes, all the blocks or packets would be of the same length. In convolutional codes the packets may have varying lengths. There are alternative ways of describing a convolutional code. It can be expressed as a tree diagram, a trellis diagram or a state diagram. For the purpose of this project, trellis and state diagrams are used. These two diagrams are explained below.

A. State Diagram

The state of the encoder (or decoder) refers to a possible combination of register values in the array of shift registers that the encoder (or decoder) is comprised of. A state diagram shows all possible present states of the encoder as well all the possible state transitions that may occur. In order to create the state diagram, a state transition table may first be made, showing the next state for each possible combination of the present state and input to the decoder. The following tables and figures show how a state diagram is drawn for a convolutional encoder.

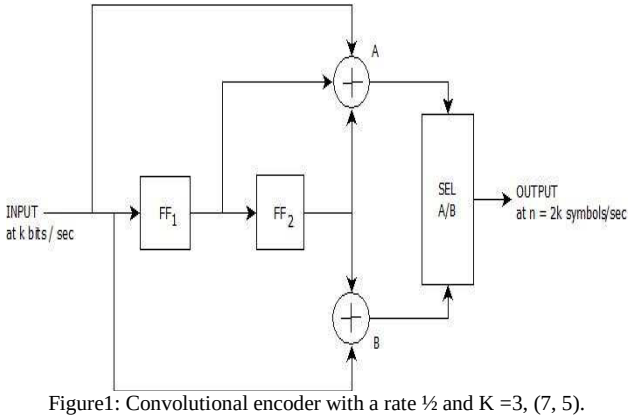

Figure1: Convolutional encoder with a rate ½ and K =3, (7, 5).

Rate ½ is used to denote the fact that for each bit of input the encoder a two bit output. K, the constraint length of the encoder being three, establishes that the input persists for 3 clock cycles. By looking at the transition of shift registers (also known as Flip Flops) FF1 and FF2, the State transition table is created for each combination of Input and Current State. This is shown in Table1. Another table can be created to demonstrate the change in output for each combination of input and previous output.

Current State (FF_1 FF_2)	Next State if	
	Input =0	Input=1
00	00	10
01	00	10
10	01	11
11	01	11

Table 1: State Transition Table

This is called the Output Table and is shown in Table 2.

Current Output	Output Symbols if	
	Input = 0	Input= 1
00	00	11
01	11	00
10	10	01
11	01	10

Table 2: Output Table

Finally, using the information from Table 1 and Table 2, the state diagram is created as shown in Figure 2. The values inside the circles indicate the state of the flip flops. The values on the arrows indicate the output of the encoder.

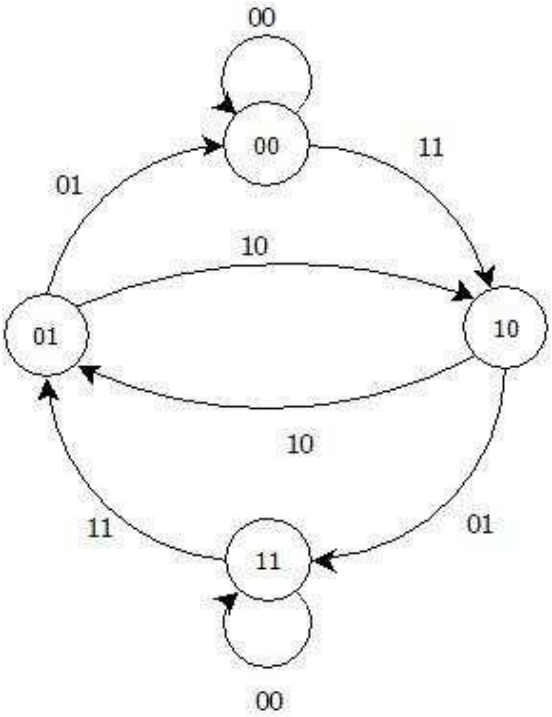

Figure2: State Diagram

B. Trellis Diagram

In a trellis diagram the mappings from current state to next state are done in a slightly different manner as shown in Figure 3. Additionally, the diagram is extended to represent all the time instances until the whole message is decoded. In the following Figure 3, a trellis diagram is drawn for the above mentioned convolutional encoder. The complete trellis diagram will replicate this figure for each time instance that is to be considered.

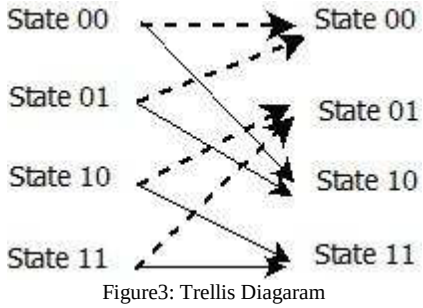

Figure3: Trellis Diagaram

V. Automatic Repeat Request (ARQ)

Automatic Repeat request or ARQ is a method in which the receiver sends back a positive acknowledgement if no errors are detected in the received message. In order to do this, the transmitter sends a Cyclic Redundancy Check or CRC along with the message. The CRC check bits are calculated based on the data to be transmitted. At the receiver, the CRC is calculated again using the received bits. If the calculated CRC bits match those received, the data received is considered accurate and an

acknowledgement is sent back to the transmitter. The sender waits for this acknowledgement. If it does not receive an acknowledgement (ACK) within a predefined time, or if it receives a negative acknowledgement (NAK), it retransmits the message [11].This retransmission is done either until it receives an ACK or until it exceeds a specified number of retransmissions.

This method has a number of drawbacks. Firstly, transmission of a whole message takes much longer as the sender has to keep waiting for acknowledgements from the receiver. Secondly, due to this delay, it is not possible to have practical, real-time, two-way communications. There are a few simple variations to the standard Stop-and-Wait ARQ such as Go-back-N ARQ, selective repeat ARQ. These are described below.

A. 'Stop and Wait' ARQ

In this method, the transmitter sends a packet and waits for a positive acknowledgement. Only once it receives this ACK does it proceed to send the next packet. This method results in a lot of delays as the transmitter has to wait for an acknowledgement. It is also prone to attacks where a malicious user keeps sending NAK messages continuously. As a result the transmitter keeps retransmitting the same packet and the communication channel breaks down.

B. 'Continuous' ARQ

In this method, the transmitter transmits packets continuously until it receives a NAK. A sequence number is assigned to each transmitted packet so that it may be properly referenced by the NAK. There are two ways a NAK is processed.

1) 'Go-back-N' ARQ

In 'Go-back-N' ARQ, the packet that was received in error is retransmitted along with all the packets that followed after it until the NAK was received. N refers to the number of packets that have to be traced back to reach the packet that was received in error. In some cases this value is determined using the sequence number referenced in the NAK. In others, it is calculated using roundtrip delay. The disadvantage of this method is that even though subsequent packages may have been received without error, they have to be discarded and retransmitted again resulting in loss of efficiency. This disadvantage is overcome by using Selective-repeat ARQ.

2) 'Selective-repeat' ARQ

In Selective-repeat ARQ, only the packet that was received in error needs to be retransmitted when an NAK is received. The other packets that have already been sent in the meantime are stored in a buffer and can be used once the packet in error is retransmitted correctly. The transmissions then pick up from where they left off. Continuous ARQ requires a higher memory capacity as compared to Stop and Wait ARQ. However it reduces delay and increases information throughput. The main advantage of ARQ is that as it detects errors (using CRC check bits) but makes no attempt to correct them, it requires much simpler decoding equipment and much less redundancy as compared to Forward Error Correction techniques which are described below. The huge drawback however, is that the ARQ method may require a large number of retransmissions to get the correct packet, especially if the medium is noisy. Hence the delay in getting messages across maybe excessive.

VI. Hybrid Automatic Repeat Request (H-ARQ)

Hybrid Automatic Repeat Request or H-ARQ is another variation of the ARQ method. In this technique, error correction information is also transmitted along with the code. This gives a better performance especially when there are a lot of errors occurring. On the flip side, it introduces a larger amount of redundancy in the information sent and therefore reduces the rate at which the actual information can be transmitted. There are two different kinds of H-ARQ, namely Type I HARQ and Type II HARQ. Type I-HARQ is very similar to ARQ except that in this case both error detection as well as forward error correction (FEC) bits are added to the information before transmission. At the receiver, error correction information is used to correct any errors that occurred during transmission. The error detection information is then used to check whether all errors were corrected. If the transmission channel was poor and many bit-errors occurred, errors may be present even after the error correction process. In this case, when all errors have not been corrected, the packet is discarded and a new packet is requested. In Type II-HARQ, the first transmission is sent with only error detection information. If this transmission is not received error free, the second transmission is sent along with error correction information. If the second transmission is also not error free, information from the first and second packet can be combined to eliminate the error. Transmitting FEC information can double or triple the message length. Error detection information on the other hand requires fewer numbers of additional bits. The advantage of Type II HARQ therefore, is that it increases the efficiency of the code to that of simple ARQ when channel conditions are good and provides the efficiency of Type I HARQ when channel conditions are bad.

VII. Viterbi Mechanism

Viterbi algorithm is the best error correction method used currently in communication systems. It is trade off between complexity of hardware and power consumption. The Viterbi Algorithm (VA) was first proposed as a solution to the decoding of convolutional codes by Andrew J. Viterbi in 1967.

A. Decoding Mechanism

There are two main mechanisms, by which Viterbi decoding may be carried out namely, the Register Exchange mechanism and the Traceback mechanism. Register exchange mechanisms, as explained by Ranpara and Sam Ha [12] store the partially decoded output sequence along the path. The advantage of this approach is that it eliminates the need for traceback and hence

reduces latency. However at each stage, the contents of each register needs to be copied to the next stage. This makes the hardware complex and more energy consuming than the traceback mechanism. Traceback mechanisms use a single bit to indicate whether the survivor branch came from the upper or lower path. This information is used to traceback the surviving path from the final state to the initial state. This path can then be used to obtain the decoded sequence. Traceback mechanisms prove to be less energy consuming and will hence be the approach followed in this project. Decoding may be done using either hard decision inputs or soft decision inputs. Inputs that arrive at the receiver may not be exactly zero or one. Having been affected by noise, they will have values in between and even higher or lower than zero and one. The values may also be complex in nature. In the hard decision Viterbi decoder, each input that arrives at the receiver is converted into a binary value (either 0 or 1). In the soft decision Viterbi decoder, several levels are created and the arriving input is categorized into a level that is closest to its value. If the possible values are split into 8 decision levels, these levels may be represented by 3 bits and this is known as a 3 bit Soft decision.

Figure 4 shows the various stages required to decode data using the Viterbi Algorithm. The decoding mechanism comprises of three major stages namely the Branch Metric Computation Unit, the Path Metric Computation and Add-Compare-Select (ACS) Unit and the Traceback Unit. A schematic representation of the decoder is described below.

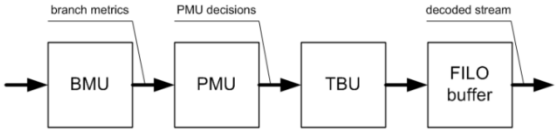

Figure 4: Schematic representation of the Viterbi decoding block

Block 1. Branch Metric Computation (BMC): For each state, the Hamming distance between the received bits and the expected bits is calculated. Hamming distance between two symbols of the same length is calculated as the number of bits that are different between them. These branch metric values are passed to Block 2. If soft decision inputs were to be used, branch metric would be calculated as the squared Euclidean distance between the received symbols [13][14][15]. The squared Euclidean distance is given as (a1-b1)2 + (a2-b2)2 + (a3-b3)2 where a1, a2, a3 and b1, b2, b3 are the three soft decision bits of the received and expected bits respectively.

A *path metric unit* summarizes branch metrics to get metrics for 2^{K-1} paths, where K is the constraint length of the code, one of which can eventually be chosen as *optimal*. Every clock it makes 2^{K-1} decisions, throwing off wittingly nonoptimal paths. The results of these decisions are written to the memory of a traceback unit. The core elements of a PMU are *ACS (Add-Compare-Select)* units. The way in which they are connected between themselves is defined by a specific code's trellis diagram. It is possible to monitor the noise level on the incoming bit stream by monitoring the rate of growth of the "best" path metric. A simpler way to do this is to monitor a single location or "state" and watch it pass "upward" through say four discrete levels within the range of the accumulator. As it passes upward through each of these thresholds, a counter is incremented that reflects the "noise" present on the incoming signal.

Block 3. Traceback Unit: The global winner for the current state is received from Block 2. Its predecessor is selected in the manner described in previous section. In this way, working backwards through the trellis, the path with the minimum accumulated path metric is selected. This path is known as the traceback path. A diagrammatic description will help visualize this process. Figure 5 describes the trellis diagram for a ½ K=3 (7, 5) coder with sample input taken as the received data.

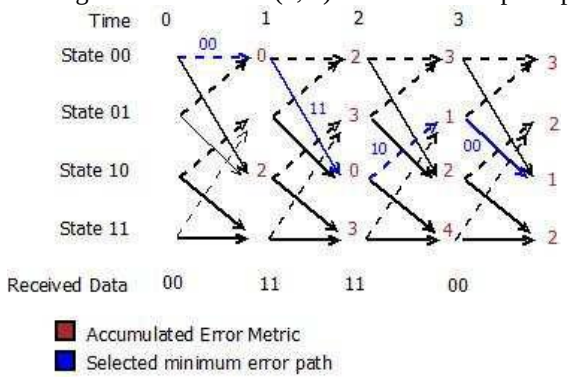

Figure 5: Selected minimum error path for a ½ K = 3 (7, 5) coder

8. Conclusion

All error detection, correction controlling mechanisms has been studied. But it has been found that viterbi is most efficient error correction mechanism in long distance communication. Following points justify my view:

1. Now a days convolutional encoders are used in all communication at the transmitter and the transmitter channel is more prone to Additive White Gaussian Noise (AWGN) which introduces error in data. To correct errors either sequential decoding (Fano coding) or most likelihood mechanism (viterbi decoder) is used. But viterbi decoder corrects error exactly.
2. Viterbi decoder assumes that errors occur infrequently, the probability of error is small and errors are distributed randomly.

References

[1]. Sklar, B., 2001. *Digital Communications – Fundamentals and Applications*. 2nd ed. New Jersey: Prentice Hall
[2]. C. E. Shannon, "A mathematical theory of communication," Bell Sys. Tech. Jour., vol. 27, pp. 379–423, 623–656, Jul./Oct. 1948.
[3]. T. Richardson and R. Urbanke, Modern Coding Theory. Cambridge University Press, 2007.
[4]. Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "An efficient 10 GBASE-T ethernet LDPC decoder design with low error floors," IEEE Journal of Solid-State Circuits, vol. 45, no. 4, pp. 843 –855, Apr. 2010.
[5]. P. Grover, K. Woyach, and A. Sahai, "Towards a communicationtheoretic understanding of system-level power consumption," IEEE J. Select. Areas Commun., vol. 29, no. 8, pp. 1744 – 1755, Sept. 2011.
[6]. P. Grover, A. Goldsmith, and A. Sahai, "Fundamental limits on complexity and power consumption in coded communication," in extended version of paper presented at ISIT'12, Feb. 2012.
[7]. P. Grover and A. Sahai, "Fundamental bounds on the interconnect complexity of decoder implementations," in Proc. of the 45th Annual Conference on Information Sciences and Systems (CISS), March 2011, pp. 1 – 6.
[8]. ——, "Green codes: Energy-efficient short-range communication," in Proceedings of the 2008 IEEE Symposium on Information Theory, Toronto, Canada, Jul. 2008.
[9]. Shannon, C.E., 1948. A Mathematical Theory of Communication. *Bell SystemTechnical Journal*, vol. 27, pp.379-423.
[10]. Proakis, J.G., 2003. *Digital Communications.* 3rd ed. New York: McGraw-Hill, Inc.
[11]. Wikipedia. *General Algorithm - Hamming Codes*. [Online]. Available at: <http://en.wikipedia.org/wiki/Hamming_code#General_algorithm> [Accessed 15 May 2010]
[12]. Ranpara, S.; Dong Sam Ha, 1999. A low-power Viterbi decoder design for wireless communications applications. *IEEE Proceedings of the Twelfth Annual IEEE International Int. ASIC Conference 1999*, Washington, DC, 15-18 Sept. 1999, pp. 377-381
[13]. Wikipedia. *Viterbi Decoder*. [Online]. Available at: http://en.wikipedia.org/wiki/Viterbi_decoder#Euclidean_metric_computation [Accessed 26 August 2010]
[14]. Chip Fleming, 'A tutorial on convolutional coding with Viterbi algorithm',2003.
[15]. K. S. Arunlal1 and Dr. S. A. Hariprasad," AN EFFICIENT VITERBI DECODER" International Journal of Advanced Information Technology (IJAIT) Vol. 2, No.1, February 2012

