LDPC FEC Optilux Simulations Report

Joe Kurian Eappen , EE13B080
Mahesh R. , EE13B087

August 13, 2016

1 Aim

To measure performances of rate 0.5 error correcting codes for LDPC and Convolutional codes in the QPSK and 16QAM modulation schemes over an optical channel simulated by means of *Optilux*.

1.1 Encoder Decoder Used

The proposed FEC scheme is a concatenation of an outer hard decision BCH code and an inner LDPC code. The inner IRA (Irregular Repeat Accumulate) LDPC codes are those proposed by the second generation satellite digital video broadcasting standard with rate $\frac{1}{2}$, which leads to an FEC overhead of 100%.

2 Introduction

The development of Error control codes have increased the resistance of Digital communication systems to noise drastically. Error control codes can be thought of as a processing done on the message bits to introduce some redundancy or extra information before transmitting which is used to check for errors at the decoding end of the communication system. *e.g.* for a rate 0.5 code, we introduce k extra bits for every k message bits during transmission so that the information rate is $\frac{k}{k+k} = 0.5$ (100% Overhead).

Since improving these Error control codes implies we can transmit data reliably at lower SNRs in an digital communication system, a lot of work has been done exploring various coding methods. Here we examine the performance of Convolutional Codes and LDPC codes (a type of Linear Block Code) used in transmission for the QPSK and 16QAM Modulation schemes over an optical channel.

2.1 Convolutional & LDPC codes

Convolutional codes are a popular variety of Error-Correction codes that were almost always preferred over other coding schemes before the outbreak of LDPC codes. Turbo codes, which are a class of Convolutional codes, have shown to perform closely to the Shannon's limit. Hence, these codes have a close competition with the state-of-the-art FEC LDPC codes. As the name suggests, Convolutional codes are obtained by convolving the incoming-message bits with a binary sequence known as the Generator polynomial to obtain the output sequence or codeword. The Viterbi-algorithm is the most famous algorithm that enabled efficient decoding of Convolutional codes before Turbo codes were found. Note that Turbo codes are also Convolutional codes that differ from the Viterbi-Convolutional codes only in the way they are decoded. Turbo codes have shown good performance with higher Constraint-lengths whereas the Viterbi-decoding requires the Constraint length to be short for otherwise the algorithmic complexity is very high. For Viterbi-decoding of convolutional codes, the algorithmic complexity increases exponentially with Constraint length. Nevertheless, Viterbi algorithm is still widely used for it is the most-efficient algorithm that gives the Optimal solution. i.e, the best estimate of the codeword (Maximum-Likelihood).

LDPC codes are a subclass of Linear block codes that have come to prominence during the 1990s with the advancement of computational capability and compete with Turbo codes as modern codes which nearly reach the theoretical maximum noise threshold (the Shannon limit). They differ from normal Linear Block codes in their iterative decoding method made possible by the sparsity (low number of ones) of their parity check matrix (H). For further information, refer [1] and [2].

3 Encoding & Decoding Algorithms

3.1 Convolutional coding

3.1.1 Encoding

In Convolutional coding, as the name suggests, the message or input bits are convolved with the so called Generator polynomials to obtain the output sequence known as the codeword. The Constraint length of a Convolutional code can be defined as the number of input bits required to produce the current output bit. Typically, this number is less than or equal to 7. The current output bit is produced by XOR-ing the current input bit with a few of the previous input bits. For this purpose, a length-7 constrained CC requires a 7-bit register to store the last 6-input bits with the current incoming bit.

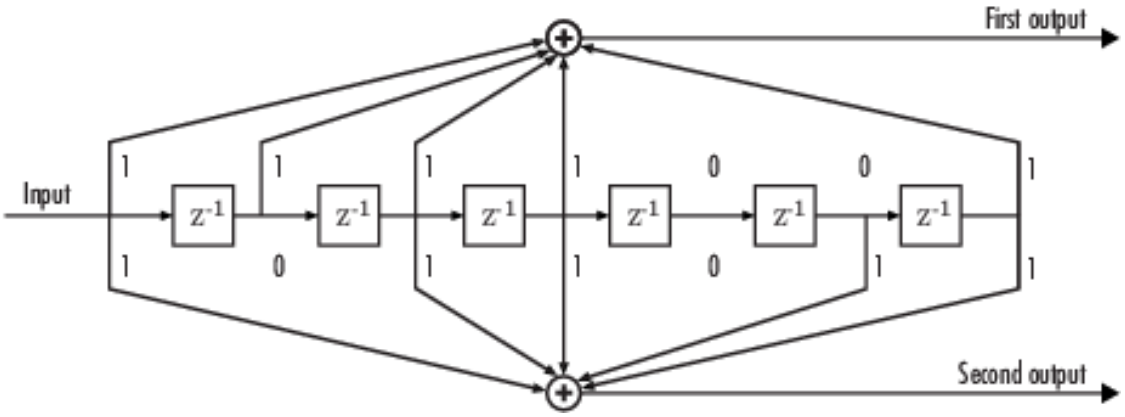

Figure 1: Rate-1/2 Convolutional encoder used in our case, Constraint length= 7

The matlab function that defines this encoder is poly2trellis(7,[171,133]). The first parameter in the function is the constraint length that is equal to 7 which is the no. of input bits required to specify the current output bit or it is the no. of delay elements(D-flipflops) in the encoder +1.

The numbers within the square brackets represent the generator polynomials(in octal form) for each of the two output ports. The generator polynomial for the upper output is given by $1+D+D^2+D^3+D^6$ which is simply represented as $(1111001)_2$ which is $(171)_8$ in octal form.

Similarly, polynomial for the lower output is $1 + D^2 + D^3 + D^5 + D^6$ which is $(1011011)_2$ and is $(133)_8$ in the Octal representation.

Let the incoming message bit be u[k], u[k-1] is the previous message bit and u[k-2] is last to last bit and so on.

Output at the upper port(gp1=171) : $op1[k] = XOR(u[k], u[k-1], u[k-2], u[k-3], u[k-6])$

Output at the lower port(gp2=133) : $op2[k] = XOR(u[k], u[k-2], u[k-3], u[k-5], u[k-6])$

Net output bits produced for the current input bit : $(op1[k], op2[k])$

3.1.2 Viterbi decoding

The Viterbi decoding of Convolutional codes uses a dynamic programming approach to find efficiently, the Minimum-Distance path or the path which minimizes the error between one of the pre-defined set of codewords and the received noisy codeword in what is known as the Trellis diagram. This is also known as Maximum-likelihood(ML) decoding. In practice, the complexity($\sim 2^k$) of ML-decoding increases

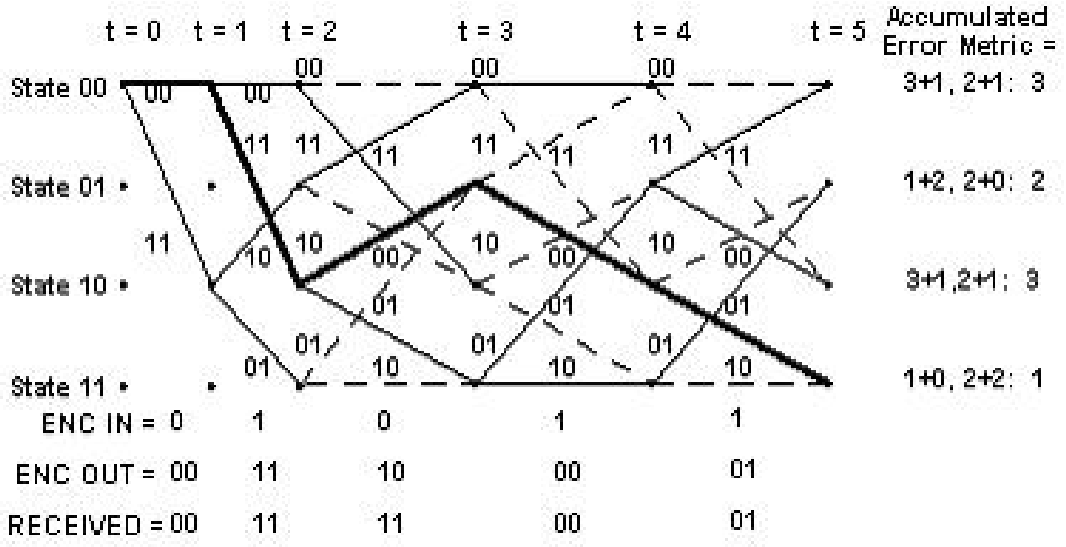

Figure 2: Trellis Diagram Example (Constraint length =3)

exponentially with the codeword length(k), if not for the Viterbi algorithm. The Viterbi algorithm does it in linear time with the codeword length, k.

The path which gives the least accumulated error till that node starting from the source node is remembered at every node. Let, the Trellis points shown in the diagram be represented in a matrix with index i for the four rows(#states in each level) and j for the columns.

$$\text{Min\_path}(i,j) = \text{Minimum}[\text{Min\_path}(k,j-1) + \text{Dist}((k,j-1),(i,j))]$$

k runs from 1 to 4 i.e, nodes in the previous level or column. Solving this equation recursively using dynamic programming is an efficient way of finding the most-likely codeword.

3.2 LDPC coding

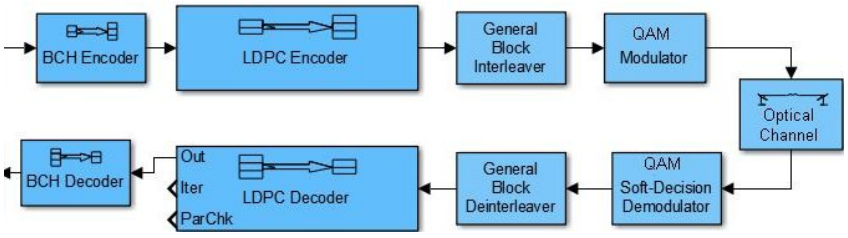

Figure 3: LDPC scheme used

The LDPC scheme used is a concatenation of a BCH code and an LDPC code used by the DVB-S2 standard. The BCH code is used to correct the errors that might slip through the LDPC code. ($\mathbf{G}$)

The encoder output is similar to one generated by a Linear Block Code with Generator matrix as codeword ($\mathbf{c}$) is related to message ($\mathbf{m}$) as $\mathbf{c} = \mathbf{mG}$ where $\mathbf{m}$ is $1 \times k$ and $\mathbf{G}$ is $k \times n$, k being the message length and n being the codeword length.

The encoder can be thought of as converting the $\mathbf{H}$ matrix (the $n-k \times n$ parity check matrix) into a $\mathbf{G}$ matrix ($k \times n$) and forming the codeword like a normal Linear Block code. Since the H matrix is very large and sparse this involves comparitively high memory overheads and computational power. Several Efficient encoding techniques have been developed [3] but their outputs are equivalent to this.

The decoder is described in the flowchart with the decoding process being carried out for each individual received codeword. The received codeword is in the form of a vector of length n of LLR values for each bit found by the QAM demodulator. The Message Passing decoding algorithm (MPA)

can be represented well by means of a Tanner graph which is an LDPC code representation with $n-k$ check nodes for the parity check matrix constraints, n variable nodes for the individual codeword bits and edges joining the variable nodes and the check nodes to show a link between a code bit and a parity check constraint (*i.e.* a one in the **H** matrix). The MPA algorithm uses a form of the Turbo principle to send information between the nodes after which the LLR value is calculated for each codeword bit. A hard decision is made on this LLR value and the estimated codeword is checked to comply with the parity check matrix. This process is iterated either until a suitable codeword is found or the number of iterations reaches a specified maximum (50 in our simulations).

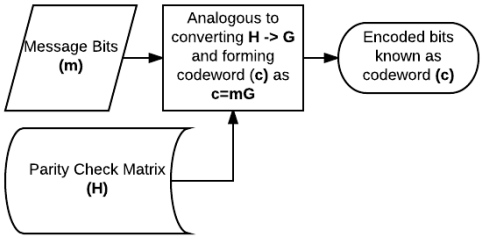

Figure 4: LDPC Encoding Algorithm

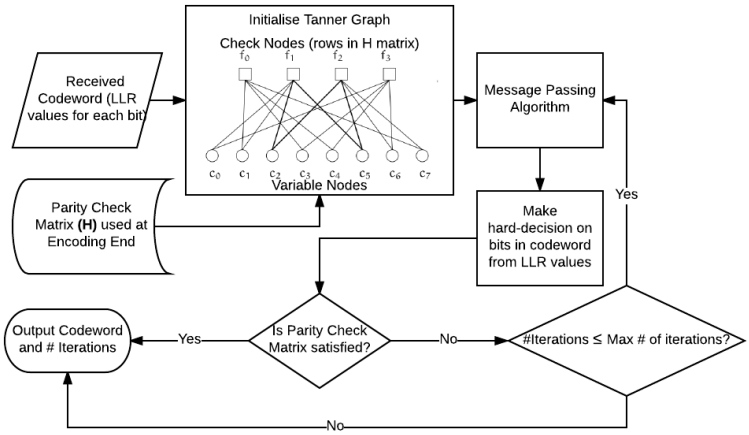

Figure 5: LDPC Decoding Algorithm

4 Simulation & Results (Matlab)

We used the DVB-S2 standard for the inner LDPC code and an existing demo for the encoder/decoder system present in Matlab using the communication systems toolbox was modified to allow different rates and constellations. In our simulations we used the `comm.LDPCEncoder` and `comm.LDPCDecoder` objects from the `comm` Matlab toolbox.

The inbuilt Matlab functions were used for convolutional coding and Viterbi decoding with the following parameters:

Code Rate = 1/2

Constraint length = 7

p1 = [1 1 1 1 0 0 1];

p2 = [1 0 1 1 0 1 1];

A few codewords of size 64800 bits each were generated offline in Matlab for the LDPC and Convolutional codes which were used for the simulation in Optilux for the QPSK and 16QAM modulation schemes.

For the simulation, optical impairments were modelled namely CD (80km), frequency offset and phase noise and were compensated for. Conversion between OSNR and E_s/N_o were done by

$$\text{OSNR} = p \times \frac{R_s}{2B_{ref}} \times \frac{E_s}{N_o}$$

where $p = 1$(Single Pol.),$R_s = 25$Gbd and $B_{ref} = 12.5$GHz.

4.1 QPSK

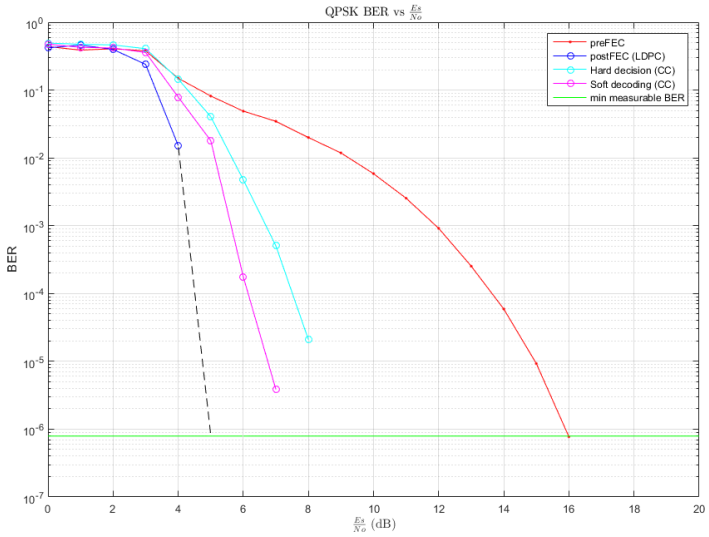

Figure 6: BER vs E_s/N_o, rate 0.5 codes (QPSK)

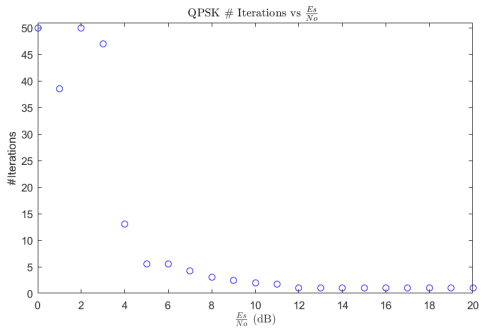

Figure 7: # Iterations vs E_s/N_o

Net Coding Gain of rate 0.5 LDPC Codes

BER	NCG (Soft CC)	NCG (PreFEC)
10^{-3}	∼1.35dB	∼7.75dB
10^{-5}	∼2.0dB	∼10.2dB

4.2 16QAM

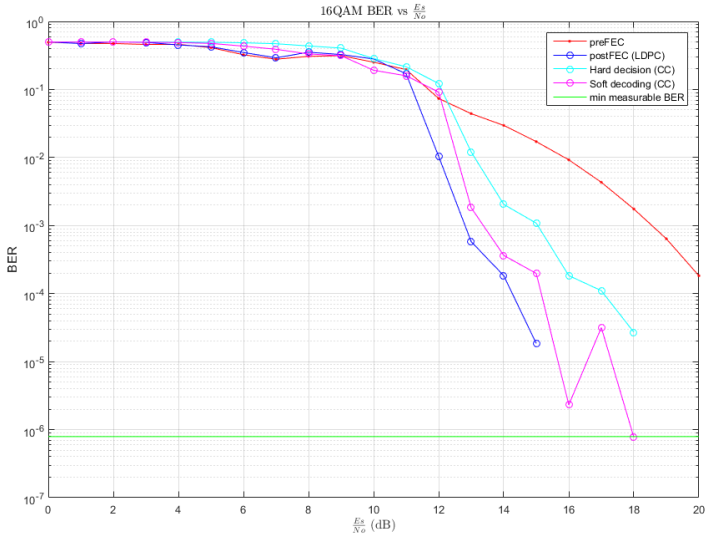

Figure 8: BER vs E_s/N_o, rate 0.5 codes (16QAM)

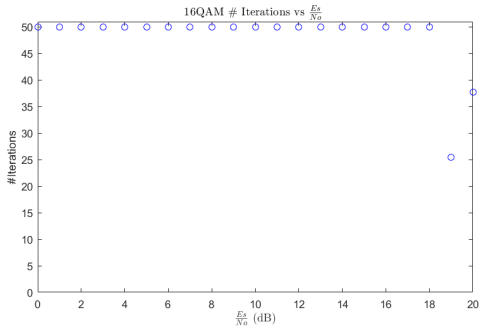

Figure 9: # Iterations vs E_s/N_o

Net Coding Gain of rate 0.5 LDPC Codes

BER	NCG (Soft CC)	NCG (PreFEC)
10^{-3}	0.6dB	5.8dB
10^{-4}	0.9dB	∼6dB
10^{-5}	∼0.5dB	–

5 Conclusions

We see how Forward Error Correction (FEC) Codes help get a much better BER at lower SNR values than simply sending the message bits via QAM modulation with hard decision decoding.

6 Future Progress

We may also obtain results for various rates by puncturing the LDPC code. This can be done for rates just above 0.5 by modifying the parity check matrix accordingly.

We could further experimentally verify the results using an appropriate hardware setup.

7 Acknowledgements

Mr. Lakshmi Narayanan and Ms. Smaranika for their assistance in the Optilux simulation.
Mr. Varughese M. for his guidance.

References

[1] An Introduction to LDPC Codes, William E.Ryan, University of Arizona

[2] Channel Codes- Classical and Modern by William E.Ryan and Shu Lin

[3] Efficient Encoding of Low-Density Parity-Check Codes by Thomas J. Richardson and Rüdiger L. Urbanke