DVB-S2 Low Density Parity Check Codes with Near Shannon Limit Performance

Mustafa Eroz, Feng-Wen Sun, Lin-Nan Lee

Hughes Network Systems
11717 Exploration Lane
Germantown, MD 20876, USA

Abstract:

Low density parity check (LDPC) codes are chosen for the second generation digital video broadcasting (DVB) standard. In this paper, we review LDPC codes in general, present belief propagation decoding algorithm in simple terms, describe the standardized LDPC codes and show their performance.

Index Terms: Low density parity check codes, iterative decoding, digital video broadcasting.

1 Introduction:

In 2002, Digital Video Broadcasting (DVB) standards body started a search for a more efficient transmission technology for second generation satellite applications (DVB-S2). The first generation, DVB-S, was introduced as a standard in 1994 and is now used widely for television and data broadcasting services throughout the world. It uses QPSK modulation and concatenated convolutional and Reed-Solomon codes. But since then more powerful coding techniques have been feasible due to advances in VLSI technology. As a result, DVB-S2 Project called for new coding proposals that use higher order modulation with a goal of 30% throughput increase for the same bandwidth and power. After closely examining several candidates in terms of performance and estimated ASIC size, the committee chose a solution based on low density parity check (LDPC) codes which actually delivered more than 35% throughput increase with respect to DVB-S. In this paper, we describe the specifics of the standardized LDPC codes in detail.

In the following section, we first review LDPC codes in general and a belief propagation decoding algorithm that works in the logarithm domain. In Section 3, we describe the restrictions that we impose on random codes to simplify implementation. Section 4 presents standardized LDPC codes for DVB-S2. Their performance compared to the DVB-S standard and theoretical Shannon limit is given in Section 5.

2 Review of LDPC Codes:

LDPC codes were discovered by Gallager in 1962 [1], but they were not given much attention for decades as the technology at the time was not mature for efficient implementation. Motivated by the success of iterative decoding of turbo codes, MacKay and Neal reintroduced LDPC codes in 1995 [2],[3], and generated great interest and activity on the subject. Unlike turbo codes, LDPC codes have an easily parallelizable decoding algorithm which consists of simple operations such as addition, comparison

and table look-up. Moreover the degree of parallelism is "adjustable" which makes it easy to trade-off throughput and complexity.

LDPC codes are linear block codes with sparse parity check matrices $H_{(N-K)xN}$, where each block of *K* information bits are encoded to a codeword of size *N*.
As an example, an LDPC code of codeword size *N*=8 and rate 1/2 can be specified by the following parity check matrix.

$$H = \begin{array}{c} \begin{matrix} n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{matrix}$$

The same code can be equivalently represented by the bipartite (Tanner) graph in Figure 1 which connects each check equation (check node) to its participating bits (bit nodes). Parity check equations imply that for a valid codeword, modulo-2 sum of adjacent bits of every check node has to be zero.

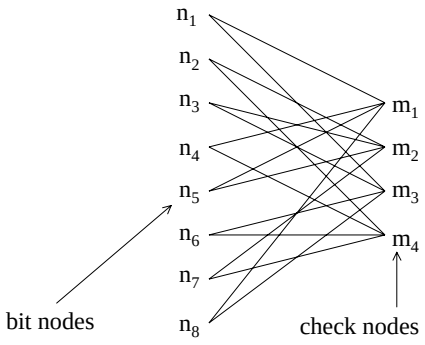


Figure 1. Bipartite (Tanner) graph of an LDPC code

The purpose of the decoder is to determine the transmitted values of the bits. Bit nodes and check nodes communicate with each other to accomplish that. The decoding starts by assigning the received channel value of every bit to all the outgoing edges from the corresponding bit node to its adjacent check nodes. Upon receiving that, the check nodes make use of the parity check equations to update the bit node information and sends it back. Each bit node then performs a soft majority vote among the information reaching from its adjacent check nodes. At this point, if the hard decisions on the bits satisfy all of the parity check equations, it means a valid codeword has been found and the process stops. Otherwise bit nodes go on sending the result of their soft majority votes to the check nodes.

In the following sections, we briefly describe the above decoding algorithm more quantitatively. The number of edges adjacent to a node is called the *degree* of that node.

2.1 Initialization:

Let *x* denote the transmitted binary phase shift keying (BPSK) symbol corresponding to a codeword bit, and let *y* denote the noisy received symbol,

$$y = x + z \quad \text{where } z \text{ is a Gaussian random variable with zero mean.}$$

Let us assume that $x = +1$ when the transmitted bit 0 and $x = -1$ when the transmitted bit is 1.

Let $u = \log \dfrac{p(x = +1 \mid y)}{p(x = -1 \mid y)}$ denote the a-priori log-likelihood ratio for the transmitted bit.

The sign of u signals the hard decision on the transmitted bit, whereas the magnitude of u gives an indication on the reliability of the decision, the bigger is the magnitude, the higher is the reliability.

Decoding starts by assigning the a-priori log-likelihood to all of the outgoing edges of every bit node:

$$v_{n \to k_i} = u_n, \qquad n = 0,1,\ldots, N-1, \quad i = 1,2,\ldots, \deg(bit\ node \quad n)$$

Here $v_{n \to k_i}$ denotes the message that goes from bit node *n* to its adjacent check node k_i, u_n denotes the a-priori log-likelihood for the bit *n* and *N* is the codeword size. The initialization process is also shown in Figure 2.

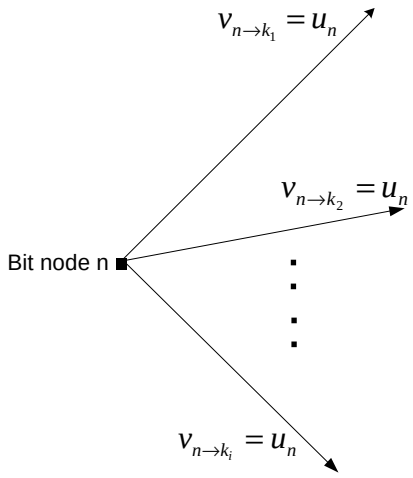


Figure 2. Initialization of outgoing messages from bit nodes

2.2 Check node update:

Let us denote the incoming messages to the check node k from its d_c adjacent bit nodes by $v_{n_1 \to k}, v_{n_2 \to k}, \ldots, v_{n_{dc} \to k}$ (see Figure 3). Our aim is to compute the outgoing messages from the check node k back to d_c adjacent bit nodes. Let us denote these messages by $w_{k \to n_1}, w_{k \to n_2}, \ldots, w_{k \to n_{dc}}$.

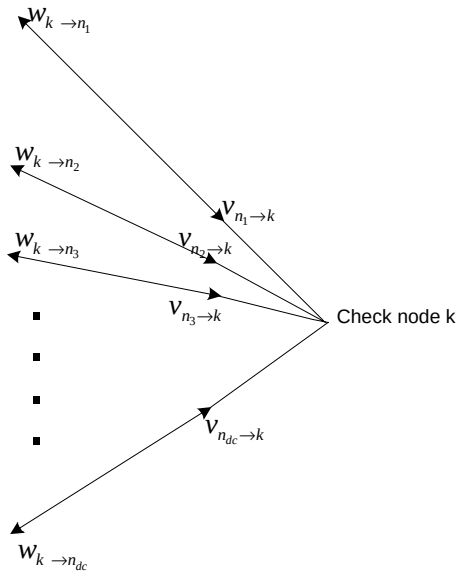


Figure 3. Message update at check nodes

Each outgoing message from check node k to its adjacent bit nodes is computed as

$$w_{k\to n_i} = g(v_{n_1\to k}, v_{n_2\to k}, ..., v_{n_{i-1}\to k}, v_{n_{i+1}\to k}, ..., v_{n_{dc}\to k})$$

where $g(a,b) = sign(a)\times sign(b)\times\{\min(|a|,|b|)\} + LUT_g(a,b)$ and

$$LUT_g(a,b) = \log(1+e^{-|a+b|}) - \log(1+e^{-|a-b|})$$

In practice, $LUT_g(.)$ function is implemented using a small look up table. Also it can be shown that the g function with multiple inputs can be recursively computed, i.e.

$$g(v_{n_1\to k}, v_{n_2\to k}, ..., v_{n_{i-1}\to k}, v_{n_{i+1}\to k}, ..., v_{n_{dc}\to k}) = g(g(v_{n_1\to k}, v_{n_2\to k}, ..., v_{n_{i-1}\to k}, v_{n_{i+1}\to k}, ..., v_{n_{dc-1}\to k}), v_{n_{dc}\to k})$$

It is easy to intuitively understand check node computations, if we ignore the small correction factor, i.e. $LUT_g = 0$. In that case, we have

$$sign(w_{k\to n_i}) = sign(v_{n_1\to k})\times sign(v_{n_2\to k})\times....\times sign(v_{n_{i-1}\to k})\times sign(v_{n_{i+1}\to k})\times....\times sign(v_{n_{dc}\to k})$$

and $|w_{k\to n_i}| = \min(|v_{n_1\to k}|, |v_{n_2\to k}|, ..., |v_{n_{i-1}\to k}|, |v_{n_{i+1}\to k}|, ..., |v_{n_{dc}\to k}|)$

The first equality is merely a re-statement of the fact that hard decision on a certain bit is modulo-2 sum of all the other bits that participate in the same parity check equation, whereas the second equality states that this hard decision can only be as reliable as the least reliable bit in the modulo-2 sum.

2.3 Bit Node Update:

Let us denote the incoming messages to the bit node n from its d_v adjacent check nodes by $w_{k_1\to n}, w_{k_2\to n}, ..., w_{k_{dv}\to n}$ (see Figure 4). Our aim is to compute the outgoing messages

from the bit node n back to d_v adjacent check nodes. Let us denote these messages by $v_{n\to k_1}, v_{n\to k_2}, \ldots, v_{n\to k_{dv}}$.

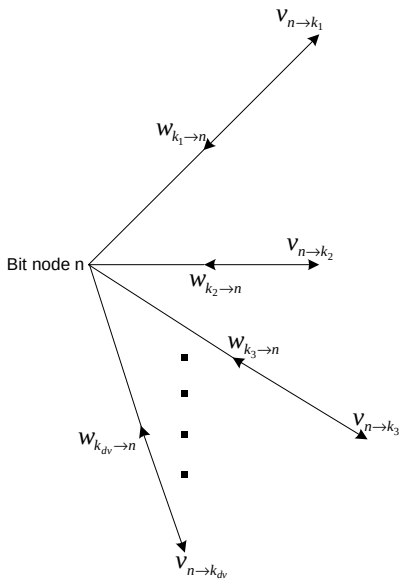


Figure 4. Message update at bit nodes

They are computed as follows: $v_{n\to k_i} = u_n + \sum_{j\neq i} w_{k_j\to n}$. Intuitively, this is a soft majority vote on the value of the bit *n*, using all relevant information except $w_{k_i\to n}$

2.4 Hard Decision Making:

After the bit node updates, hard decision can be made for each bit *n* by looking at the sign of $v_{n\to k_i} + w_{k_i\to n}$ for any k_i. If the hard decisions satisfy all the parity check equations, it means a valid codeword has been found, therefore the process stops. Otherwise another check node/bit node update is performed. If no convergence is achieved after a pre-determined number of iterations, the current output is given out and a decoding failure can be declared.

3 Structure of Parity Check Matrices of Standardized LDPC Codes:

LDPC codes can be specified through their parity check matrices. On the other hand, the standardized LDPC codes are tens of thousands of bits long. Therefore for DVB-S2 codes, certain structure is imposed on parity check matrices *H*, to facilitate the description of the codes and for easy encoding. The next subsections are devoted to describe this structure.

3.1 Lower Triangular Parity Check Matrices for Easy Encoding:

Even though parity check matrices of LDPC codes are sparse, in general generator matrices are needed for encoding. Of course, for any linear code once the parity check matrix is known, a generator matrix can be derived using for instance Gaussian elimination method. But the resulting generator matrix would no longer be sparse leading to storage and encoding complexity problems. Even though there are methods that partially solves the problem [4], restricting a sub-matrix of the parity check matrix to be lower triangular eliminates the need to derive a generator matrix and leads to linear encoding complexity.

More specifically, we restrict the parity check matrix to be of the form, $H_{(N-K)xN} = [A_{(N-K)xK}\ B_{(N-K)x(N-K)}]$ where B is staircase lower triangular as in Figure 5.

$$B = \begin{bmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ & 1 & 1 & & 0 & \\ & & 1 & 1 & & \\ & & & 1 & \ddots & \\ & 0 & & & \ddots & 1 \\ & & & & & 1 \quad 1 \end{bmatrix}$$

Figure 5. Submatrix of Parity Check Matrix

Then any information block $\boldsymbol{i} = (i_0, i_1, ..., i_{k-1})$ is encoded to a codeword $\boldsymbol{c} = (i_0, i_1, ..., i_{k-1}, p_0, p_1, ... p_{n-k-1})$ using $H\mathbf{c}^T = \mathbf{0}$, and recursively solving for parity bits.

$$a_{00}i_0 + a_{01}i_1 + ... + a_{0,k-1}i_{k-1} + p_0 = 0 \Rightarrow Solve\ p_0$$
$$a_{10}i_0 + a_{11}i_1 + ... + a_{1,k-1}i_{k-1} + p_0 + p_1 = 0 \Rightarrow Solve\ p_1$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{N-K-1,0}i_0 + a_{N-K-1,1}i_1 + ... + a_{N-K-1,k-1}i_{k-1} + p_{N-K-2} + p_{N-K-1} = 0 \Rightarrow Solve\ p_{N-K-1}$$

Note that the matrix A is sparse; as a result encoding has linear complexity with respect to the block length.

Our simulations show that the above lower-triangular restriction on the parity check matrix lead to negligible (within 0.1 dB) performance loss with respect to a general parity check matrix for the cases of practical interest to us.

3.2 Periodicity on Parity Check Matrices for Reduced Storage:

The following restriction on the *A* submatrix of parity check matrix design reduces the storage requirement of the matrix description by a factor of *M*.
For a group of M bit nodes, if the check nodes connected to the first bit node of degree, say d_v, are numbered as $c_1, c_2, ..., c_{d_v}$ then the check nodes connected to the i^{th} bit node

$(i \le M)$ are numbered as,

$\{c_1 + (i-1)q\} \bmod (N-K), \{c_2 + (i-1)q\} \bmod (N-K), ..., \{c_{d_v} + (i-1)q\} \bmod (N-K)$

where $N-K$ = total number of check nodes and $q = \frac{N-K}{M}$.

For the following groups of M bit nodes, the check nodes connected to the first bit node of the group are in general randomly chosen so that the resulting LDPC code is cycle-4 free and occurrence of cycle-6 is minimized to the extent a solution can be found within a reasonable search time. Too many short cycles (such as cycle-4 and cycle-6), where one can find cycles in the bipartite graph containing 4 or 6 nodes, are detrimental to code performance since they lead to non-extrinsic information being fed back after a small number of iterations.

From the above description, it is clear that adjacent check nodes of only one bit node need to be specified in a group of *M*, simplifying the code description by a factor of *M*. In DVB-S2, we choose *M* = 360.

4 Description of Standardized LDPC Codes:

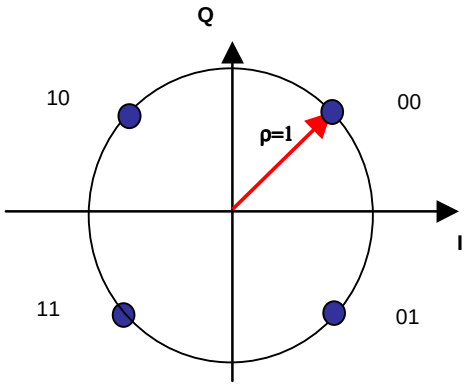
In DVB-S2, a wide range of bandwidth efficiency from 0.5 bits/symbol up to 4.5 bits/symbol is covered by defining ten different code rates 1/4, 1/3, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 and 9/10 with four different modulation schemes QPSK, 8-PSK, 16-APSK and 32-APSK. Rate 1/4, 1/3, 1/2 and 2/3 codes are also used in the low priority branch of hierarchical 8-PSK of backward compatible mode, as described in the DVB-S2 standard document. For each code rate, a parity check matrix is specified by listing adjacent check nodes for the first bit node in a group of *M* = 360. The coded block length is *N* = 64800 bits for all rates. To improve the performance, irregular LDPC codes are used where degrees of bit nodes are varying. The reason for this improvement is due to the fact that bit nodes with high degrees collect more information from their adjacent check nodes and they get corrected first after a small number of iterations. They then help other bit nodes get corrected through iterative decoding, similar to "wave effect". When all bit nodes have the same degrees, as in regular codes, this wave effect is not present and all the bit nodes can simultaneously get stuck during the decoding process. The list of bit node degrees and the total number of nodes with those degrees are shown in Table 1 for all the code rates.

Table 1. Number of Bit Nodes of Various Degrees

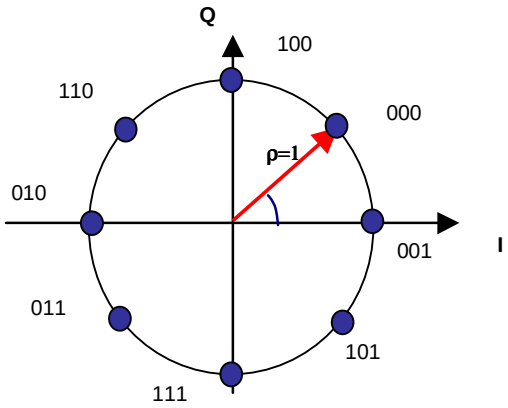
Code Rate	13	12	11	8	4	3	2	1
1/4		5400				10800	48599	1
1/3		7200				14400	43199	1
1/2				12960		19440	32399	1
3/5		12960				25920	25919	1
2/3	4320					38880	21599	1
3/4		5400				43200	16199	1
4/5			6480			45360	12959	1
5/6	5400					48600	10799	1
8/9					7200	50400	7199	1
9/10					6480	51840	6479	1

Constellation labelings are shown in Figure 6. 16-APSK and 32-APSK (as opposed to 16-QAM and 32-QAM) are chosen due to their "non-linearity friendly" characteristics.

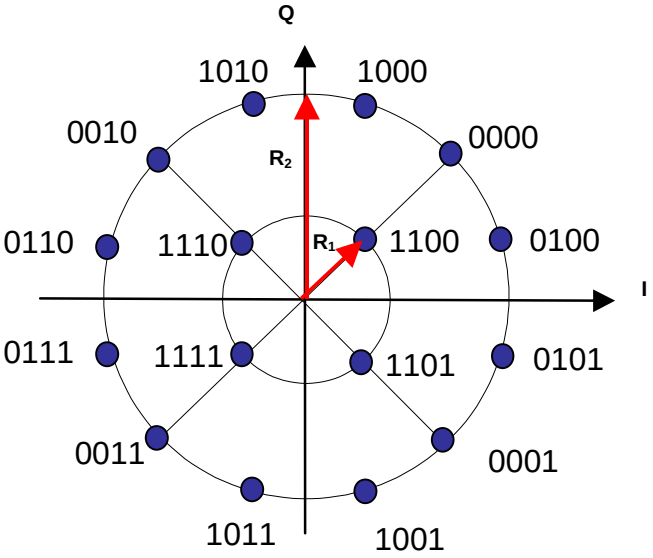
Moreover their performance on linear channels are almost as good as their QAM competitors.



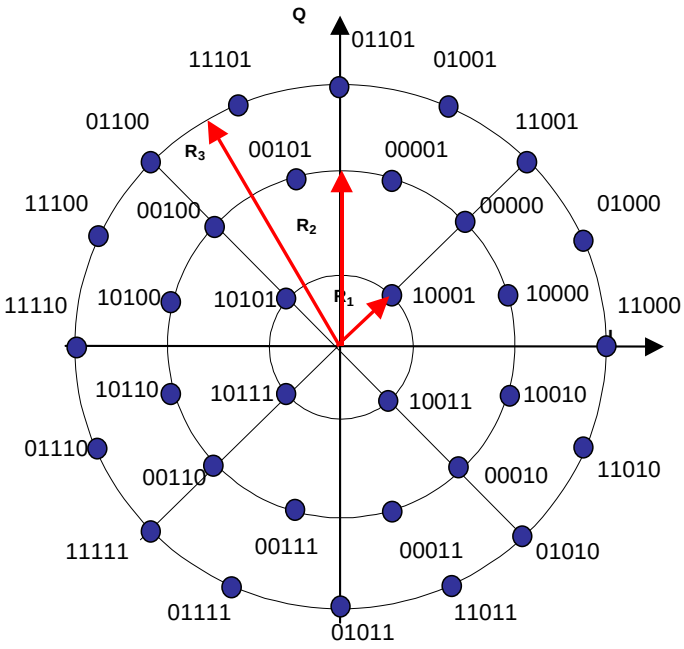
(a) QPSK



(b) 8-PSK



(c) 16-APSK



(d) 32-APSK

Figure 6. Constellation Labelings

5 Performance Results:

Even though the above design restricts the parity check matrix to be structured, the performance is still very good due to the careful choice of check node/bit node connections. Performance of various code rates with different constellations on AWGN channel is depicted in Figure 7. Maximum number of decoder iterations is 50. If a valid codeword is not found by then, the decoder outputs its current bit estimates at the end of 50 iterations. Each LDPC frame is divided to form multiple MPEG packets, 188 bytes each. Since the error rate requirements of DVB-S2 are rather stringent (10^{-7} packet error rate), an outer BCH code with the same block length as LDPC frame and an error correction capability of up to 12 bits is employed.

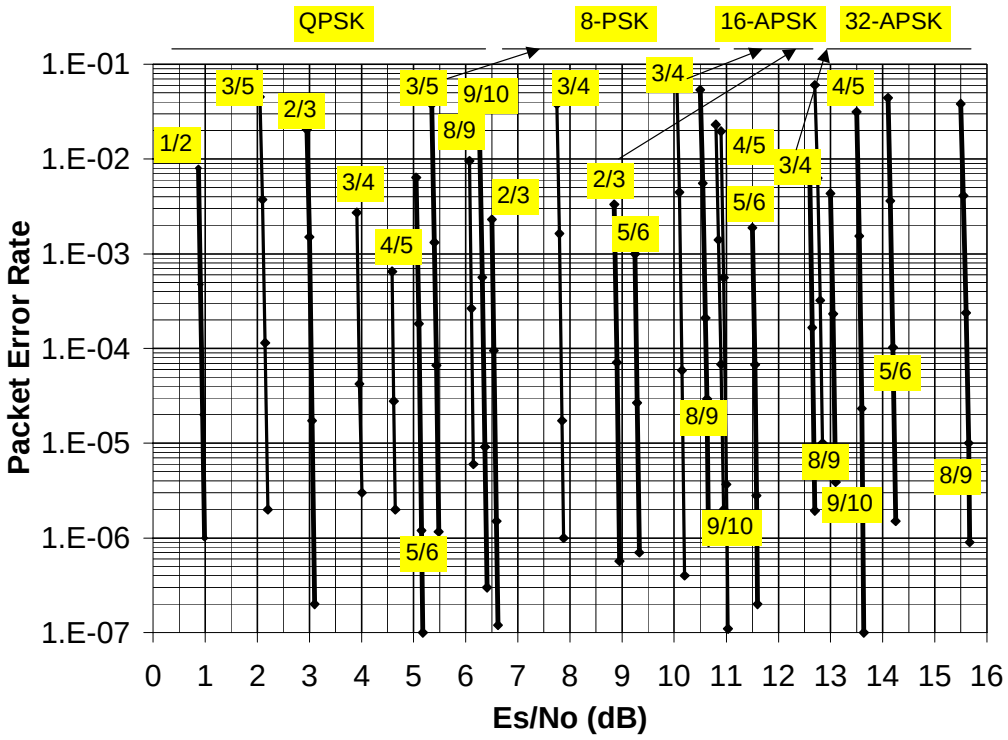


Figure 7. Performance of LDPC+BCH Codes over AWGN Channel, *N*=64800 bits

C/N requirements of DVB-S2 concatenated LDPC and BCH codes at 10^{-7} MPEG PER for various code rates and modulation schemes are shown in Figure 8. For comparison, the performance of DVB-S code and Shannon limits of constellations are also plotted. It is

important to note that the DVB-S2 design follows closely the theoretical limit for the entire range of operation. Also compared to DVB-S concatenated convolutional and Reed-Solomon codes, a capacity improvement of more than 35% is achieved at the same C/N.

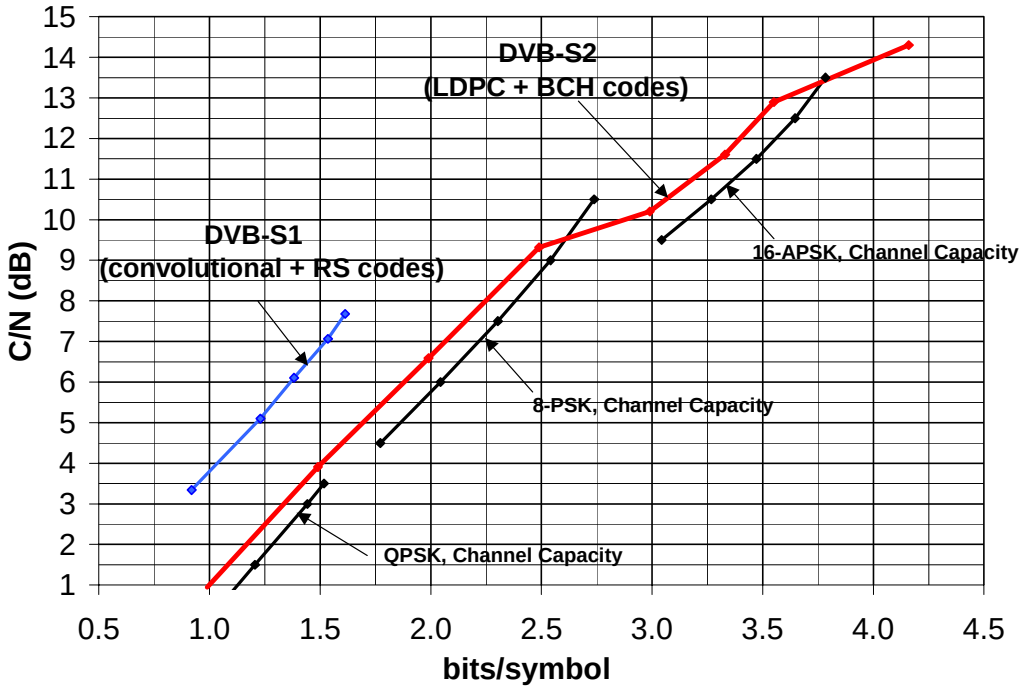


Figure 13. Comparison of DVB-S2 (LDPC+BCH) Codes to DVB-S1 and Channel Capacity

6 Conclusion:

LDPC codes of DVB-S2 approach Shannon limit to within 0.6-0.8 dB for a wide range of throughput and yet they are easy to implement. It may be hard to justify their replacements for decades to come.

References:

1. R. G. Gallager, "Low density parity check codes", IRE *Trans. Info. Theory*, 1962, IT-8, pp. 21-28
2. D. J. MacKay and R. M. Neal, "Good codes based on very sparse matrices", 5th IMA Conf. 1995, pp.100-111
3. D. J. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes",
   Electronics Lett. Mar. 1997, vol. 33, no.6, pp. 457-458

4. T. Richardson and R. Urbanke, “Efficient encoding of low-density parity check codes”, *IEEE Trans. Info. Theory*, vol. 47, pp.638-656, Feb. 2001
5. T. Richardson, A. Shokrollahi and R. Urbanke, “Design of capacity approaching irregular low density parity check codes”, IEEE Trans. Inform. Theory, Feb. 2001, vol. 47, pp. 619-637