GODDARD TECHNICAL STANDARD	GSFC-STD-9100
Goddard Space Flight Center Greenbelt, MD 20771	Approved: 03-19-2008 Expiration Date: 04-10-2022

Low Density Parity Check Code for Rate 7/8

04/10/2017

DO NOT USE PRIOR TO APPROVAL.

Check the Goddard Standards website at http://standards.gsfc.nasa.gov/
to verify that this is the correct version before use.

DOCUMENT HISTORY LOG

Status	Document Revision	Approval Date	Description
Baseline		10-18-2007	Initial Release
		10-27-2015	Extended until 12-31-2016
		02-16-2017	Extended until 12-31-2017
Revision A	2/9/17		Add asynchronous (sliced) transfer frame

FOREWORD

This standard is published by the Goddard Space Flight Center (GSFC) to provide uniform engineering and technical requirements for processes, procedures, practices, and methods that have been endorsed as standard for NASA programs and projects, including requirements for selection, application, and design criteria of an item.

This standard establishes a common GSFC channel coding protocol for bandwidth efficient spacecraft communications.

Original signed by:

George Alcorn

Goddard Technical Standards Coordinator

Original signed by:

Steven Scott

Chief Engineer

Original signed by:

Orlando Figueroa

Director of Applied Engineering and Technology

Original signed by:

Marcus Watkins

Director of Office of Systems Safety and Mission Assurance

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Document Title

1. SCOPE

1.1 Purpose

The purpose of this standard is to establish a common GSFC channel coding protocol for bandwidth efficient spacecraft communications. Currently many Goddard missions use the concatenated Reed-Solomon and convolutional coding technique for space to ground links. While this standard has served NASA well in the past it is bandwidth inefficient. The need for bandwidth efficiency has prompted the Microwave and Communication Systems Branch (Code 567) to search for a new channel code that require less bandwidth without paying a heavy penalty in power requirement and complexity. This document details the result of that search. It gives a technical description of this new channel coding called low density parity check (LDPC) coding (Section 4.1.3). A description of the base LDPC code and its' encoding is given in Section 4.2 and 4.3 and Appendix A. However, the base code needs to be modified to ease implementations for current space and ground systems. This modification is the standard and is described in Section 4.4. In addition, Section 4.5 outlines synchronization issues and the Appendices B and C discuss complexity issues and performance testing respectively. The reader is assumed to have a basic understanding of channel coding theory (linear algebra also) and digital communications. (The reader is encouraged to review [7] for an overview of linear block codes).

1.2 Applicability

This standard is intended for any space to ground communication link that requires data high reliability and power efficiency, and is applicable when cited in contract, program, and other Agency documents as a technical requirement. Mandatory requirements are indicated by the word "shall." Tailoring of this standard for application to a specific program or project shall be approved by the Technical Authority for that program or project.

2. APPLICABLE DOCUMENTS

2.1 General

The documents listed in this section contain provisions that constitute requirements of this standard as cited in the text of section 4. The latest issuances of cited documents shall be used unless otherwise approved by the assigned Technical Authority. The applicable documents are accessible via the NASA Technical Standards System at http://standards.nasa.gov, directly from the Standards Developing Organizations, or from other document distributors.

2.2 Government Documents

(None)

2.3 Non-Government Documents

CCSDS 131.0-B-2 TM Synchronization and Channel Coding. Blue Book

2.4 Order of Precedence

When this standard is applied as a requirement or imposed by contract on a program or project, the technical requirements of this standard take precedence, in the case of conflict, over the technical requirements cited in applicable documents or referenced guidance documents.

3. ACRONYMS AND DEFINITIONS

3.1 Acronyms and Abbreviations

ASM	Attached Synchronization Marker
CCSDS	Consultative Committee for Space Data Systems
FPGA	Field Programmable Gate Array
LDPC	Low Density Parity Check
MSB	Most Significant Bit

3.2 Definitions

Channel Coding (Code): A method of improving the reliability of a noisy communication channel by the addition of redundant information to the transmitted data.

Circulant: A type of square matrix that has for every row (column) defined is a right or left cyclic shift of its preceding row.

Codeword: The data structure of the encoded data that is transmitted over channel.

Cyclic Shift: A left or right shift of an array of elements, i.e. a row or column of a matrix, where the end element is wrapped around to the beginning element.

Decoder: The process responsible for estimating the original transmitted data from the noisy received data.

Encoder: The process responsible for adding redundant information as defined by the Generator matrix.

Generator Matrix: A matrix that is typically associated with the encoding of the data at the transmitter and is specified by the channel code.

Linear block code: A channel code that separates the data stream to be transmitted into blocks of finite number of symbols (bits) based on a matrix specification (Generator and Parity-Check).

Low Density Parity Check Codes: Linear block codes in which the ratio of the total number of 1's to the total number of elements in the parity check matrix is << 0.5.

Matrix: A rectangular array of elements (i.e. numbers) arranged in horizontal rows and vertical columns.

Parity Check Matrix: A matrix that is typically associated with the decoding process at the receiver and is specified by the channel code.

Quasi-cyclic: A type of matrix structure composed of circulant submatrices.

Subcode: A linear block code whose codewords are a subset of a larger base code.

Submatrix: A matrix can be decomposed into smaller matrices, which are called submatrices.

Weight: The number of 1's in column or row of a matrix.

4. REQUIREMENTS

4.1 Preliminaries

4.1.1 Numbering Convention

This document adheres to the following convention with few exceptions: the first bit in a data field (e.g. codeword or column of matrix as it relates to the codeword) to be transmitted is defined to be "Bit 1"; the following bit is defined to be "Bit 2" and so on up to "Bit N", as shown in Figure 1. In the instances where the data field begins with Bit 0, it will end with Bit N-1 but follow the same ascending order. When the field is used to express a binary value, the most significant bit (MSB) is the first transmitted bit of the field.

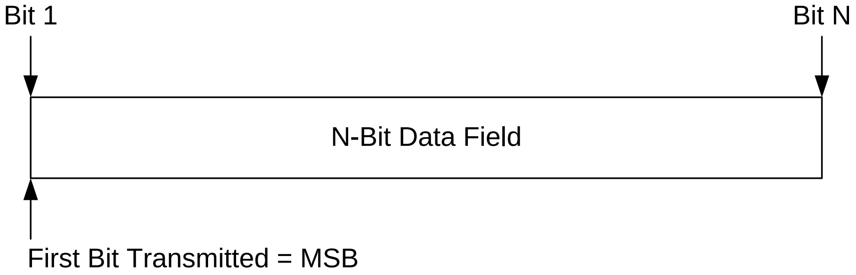

Figure 1: Bit Numbering Convention

4.1.2 Conformance

An implementation conforms to this standard by conforming to Sections 4.2, 4.3, 4.4, 4.5, the normative reference of Section 5.1, and Appendix A. Section 4.2, 4.3, and Appendix A specifies the encoding, while Section 4.4 defines the format of the codeword. Section 4.5 defines the pseudo-randomization of the codeword and codeword synchronization.

4.1.3. Technical Introduction

A linear block code is designated in this specification by (n, k) where n is the length of the codeword (or block) and k is the length of the information sequence. Low Density Parity Check (LDPC) codes are linear block codes in which the ratio of the total number of 1's to the total number of elements in the parity check matrix is << 0.5. The distribution of the 1's determine the structure and performance of the decoder. An LDPC code is defined by its parity check matrix. The k x n generator matrix that is used to encode a linear block code can be derived from the parity check matrix through linear operations. In general, a codeword $\boldsymbol{v}$ is obtained by matrix multiplication of the information sequence or vector $\boldsymbol{u}$ and the generator matrix $\boldsymbol{G}$.

The LDPC code considered in this specification is a member of a class of codes called Quasi-Cyclic codes. The construction of these codes involves juxtaposing smaller circulants (or cyclic submatrices) to form a larger parity check or base matrix.

$$\begin{bmatrix}
0&0&0&0&0&0&0&1&1&0&1&0&0&0&1\\
1&0&0&0&0&0&0&0&1&1&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&1&1&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&1&1&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&1&1&0&1\\
1&0&0&0&1&0&0&0&0&0&0&0&1&1&0\\
0&1&0&0&0&1&0&0&0&0&0&0&0&1&1\\
1&0&1&0&0&0&1&0&0&0&0&0&0&0&1\\
1&1&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&1&0&1&0&0&0&1&0&0&0&0&0&0\\
0&0&1&1&0&1&0&0&0&1&0&0&0&0&0\\
0&0&0&1&1&0&1&0&0&0&1&0&0&0&0\\
0&0&0&0&1&1&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&1&1&0&1&0&0&0&1&0&0\\
0&0&0&0&0&0&1&1&0&1&0&0&0&1&0
\end{bmatrix}$$

Figure 2. Example of a 15 x 15 circulant matrix

An example of a circulant is shown in Figure 2. Notice that every row is one bit right cyclic shift (where the end bit is wrapped around to the beginning bit) of the previous row. The entire circulant is uniquely determined and specified by its first row. For this example the first row has 4 1's or a row weight of 4.

An example of a quasi-cyclic parity check matrix is shown in Figure 3. In this case, a quasi-cyclic 10 x 25 matrix is formed by an array of 2 x 5 circulant submatrices of size 5 x 5. To unambiguously describe this matrix, only the position of the 1's in the first row of every circulant submatrix and the location of each submatrix within the base matrix is needed.

$$\left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc}
1&0&0&1&0&1&1&0&0&0&0&1&0&0&1&0&1&1&0&0&1&0&0&0&1\\
0&1&0&0&1&0&1&1&0&0&1&0&1&0&0&0&0&1&1&0&1&1&0&0&0\\
1&0&1&0&0&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&0&1&1&0&0\\
0&1&0&1&0&0&0&0&1&1&0&0&1&0&1&1&0&0&0&1&0&0&1&1&0\\
0&0&1&0&1&1&0&0&0&1&1&0&0&1&0&1&1&0&0&0&0&0&0&1&1\\
\hline
0&1&0&0&1&0&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&1&0&0&0\\
1&0&1&0&0&1&0&0&0&1&1&0&0&1&0&0&1&0&1&0&0&1&1&0&0\\
0&1&0&1&0&1&1&0&0&0&0&1&0&0&1&0&0&1&0&1&0&0&1&1&0\\
0&0&1&0&1&0&1&1&0&0&1&0&1&0&0&1&0&0&1&0&0&0&0&1&1\\
1&0&0&1&0&0&0&1&1&0&0&1&0&1&0&0&1&0&0&1&1&0&0&0&1
\end{array}\right]$$

Figure 3. Example of a quasi-cyclic matrix

Constructing parity check matrices in this manner produces two positive features:

1. The encoding complexity can be made linear with the code length or parity bits using shift registers, and
2. Encoder and decoder routing complexity in the interconnections of integrated circuits is reduced.

4.2 Base (8176, 7156) LDPC Code

The base LDPC code described in this section is the foundation for the standard shortened code defined in Section 4.4.

The parity check matrix for the base (8176, 7156) LDPC code is formed by using a 2 x 16 array of 511 x 511 square circulants. This creates a parity check matrix of dimension 1022 x 8176. The structure of the parity check base matrix is shown in Figure 4.

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} & A_{1,7} & A_{1,8} & A_{1,9} & A_{1,10} & A_{1,11} & A_{1,12} & A_{1,13} & A_{1,14} & A_{1,15} & A_{1,16} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & A_{2,5} & A_{2,6} & A_{2,7} & A_{2,8} & A_{2,9} & A_{2,10} & A_{2,11} & A_{2,12} & A_{2,13} & A_{2,14} & A_{2,15} & A_{2,16} \end{bmatrix}$$

Figure 4. Base Parity Check Matrix of the (8176, 7156) LDPC code

Each $A_{i,j}$ is a 511 x 511 circulant. The row weight of the each of the 32 circulants is 2, i.e. there are two 1's in each row. The total row weight of each row in the parity check matrix is 2 x 16 or 32. The column weight of each circulant is also 2, i.e. there are two 1's in each column. The total weight of each column in the parity check matrix is 2 x 2 or 4. The position of the 1's in each circulant is defined in table 1. A scatter chart of the parity check matrix is shown in Figure 5 where every 1 bit in the matrix is represented by a point.

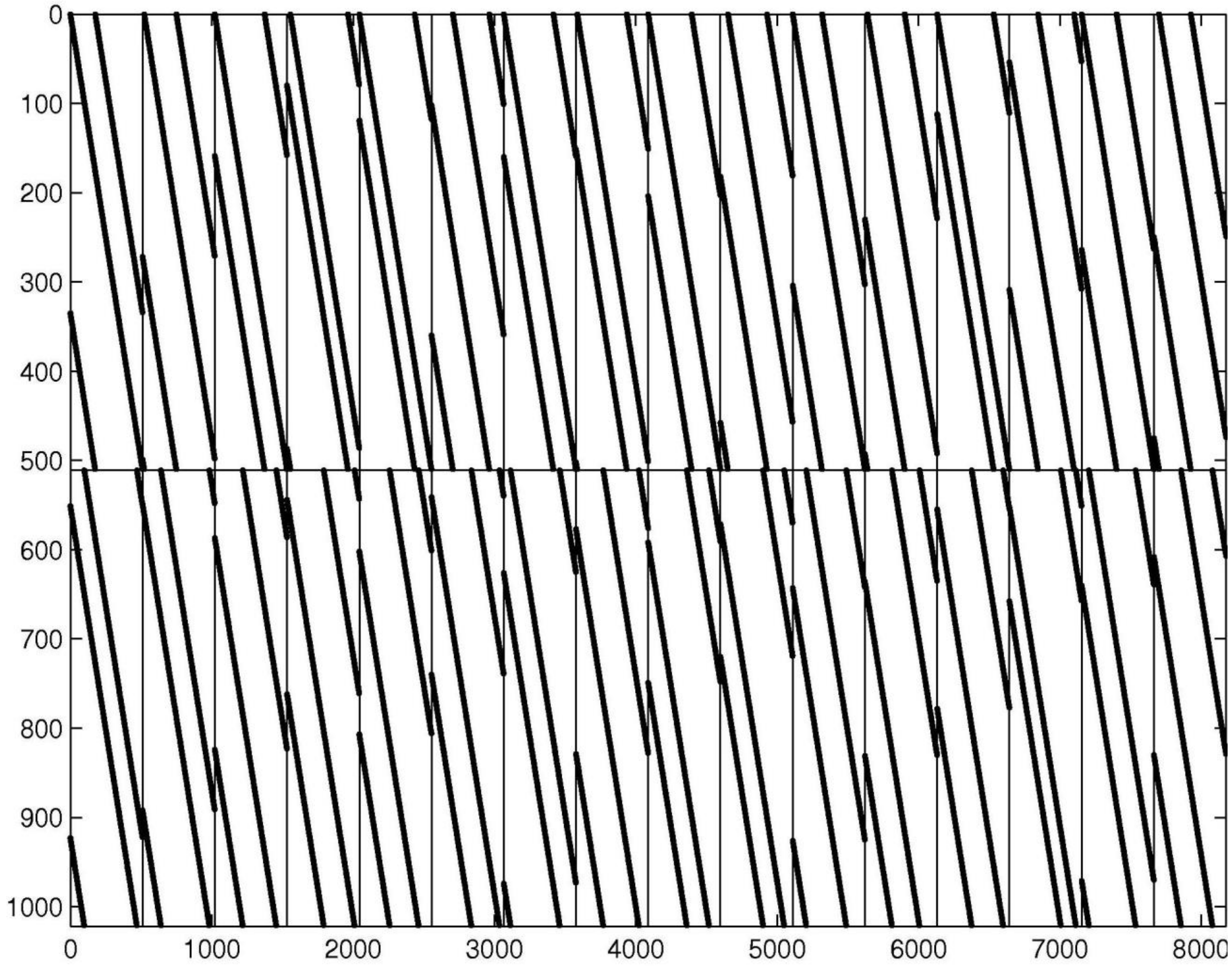

Figure 5. Scatter Chart of Parity Check Matrix

Table 1. Specification of Circulants

Circulate	1’s position in 1st row of circulant	Absolute 1’s position in 1st row of Parity Check Matrix
$A_{1,1}$	0, 176	0, 176
$A_{1,2}$	12, 239	523, 750
$A_{1,3}$	0, 352	1022, 1374
$A_{1,4}$	24, 431	1557, 1964
$A_{1,5}$	0, 392	2044, 2436
$A_{1,6}$	151, 409	2706, 2964
$A_{1,7}$	0, 351	3066, 3417
$A_{1,8}$	9, 359	3586, 3936
$A_{1,9}$	0, 307	4088, 4395
$A_{1,10}$	53, 329	4652, 4928

$A_{1,11}$	0, 207	5110, 5317
$A_{1,12}$	18, 281	5639, 5902
$A_{1,13}$	0, 399	6132, 6531
$A_{1,14}$	202, 457	6845, 7100
$A_{1,15}$	0, 247	7154, 7401
$A_{1,16}$	36, 261	7701, 7926
$A_{2,1}$	99, 471	99, 471
$A_{2,2}$	130, 473	641, 984
$A_{2,3}$	198, 435	1220, 1457
$A_{2,4}$	260, 478	1793, 2011
$A_{2,5}$	215, 420	2259, 2464
$A_{2,6}$	282, 481	2837, 3036
$A_{2,7}$	48, 396	3114, 3462
$A_{2,8}$	193, 445	3770, 4022
$A_{2,9}$	273, 430	4361, 4518
$A_{2,10}$	302, 451	4901, 5050
$A_{2,11}$	96, 379	5206, 5489
$A_{2,12}$	191, 386	5812, 6007
$A_{2,13}$	244, 467	6376, 6599
$A_{2,14}$	364, 470	7007, 7113
$A_{2,15}$	51, 382	7205, 7536
$A_{2,16}$	192, 414	7857, 8079

Note that the numbers in the second column represent the relative column position of the 1's in the first row of each circulant. Since there are only 511 possible positions, these numbers can only range from 0 to 510. The third column represents the absolute position of the 1's in the parity-check matrix. There are exactly 8176 possible; therefore these numbers can only range from 0 to 8175.

4.3 Encoding

The generator matrix for the base (8176, 7156) code (Figure 4) consists of two components:

- The first component is a 7154 x 8176 submatix in systematic-circulant form as shown in Figure 6. It consists of a 7154 x 7154 identity matrix and two columns of 511 x 511 circulants $B_{i,j}$'s, each column consisting of 14 circulants. The I's are the 511 x 511 identity submatrices and the 0's are the all zero 511 x 511 submatrices.
- The second component consists of two independent rows (not shown).

The first component generates a (8176, 7154) LDPC subcode of the (8176, 7156) code and shall be used to implement the standard. The subcode is a subset of codewords from the base code. Each codeword in the subcode consists of 7154 information bits and 1022 parity-check bits. For reasons given in Section 4.4, there are advantages in using only the subcode implementation. The diagram in Figure 7 is an example of how an encoder can be designed using the circulants $B_{i,j}$'s. (Please refer to [8] for additional information on encoding.) Appendix A gives the values of the first row of these circulants and shall be used to generate the subcode. These values are initialized or loaded in the shifted registers at the start of each information sequence. The top circuit is responsible for parity bits P1 to P511 and the bottom circuit for bits P512 to P1022. After initialization, two parity bits are created, P1 and P512, then after 510 shifts, all of the parity bits are generated. Afterwards, a codeword shortening procedure shall be performed in accordance to Section 4.4.

There are many other ways to design the encoder based on the generator matrix in Figure 6. These schemes have complexities that are proportional to the length of the codeword or parity check bits [8].

$$\begin{bmatrix}
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{1,1} & B_{1,2} \\
0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{2,1} & B_{2,2} \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{3,1} & B_{3,2} \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{4,1} & B_{4,2} \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{5,1} & B_{5,2} \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{6,1} & B_{6,2} \\
0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{7,1} & B_{7,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & B_{8,1} & B_{8,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & B_{9,1} & B_{9,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & B_{10,1} & B_{10,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & B_{11,1} & B_{11,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & B_{12,1} & B_{12,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & B_{13,1} & B_{13,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & B_{14,1} & B_{14,2}
\end{bmatrix}$$

Figure 6. Systematic Circulant Generator Matrix of 14 x 16 Circulants

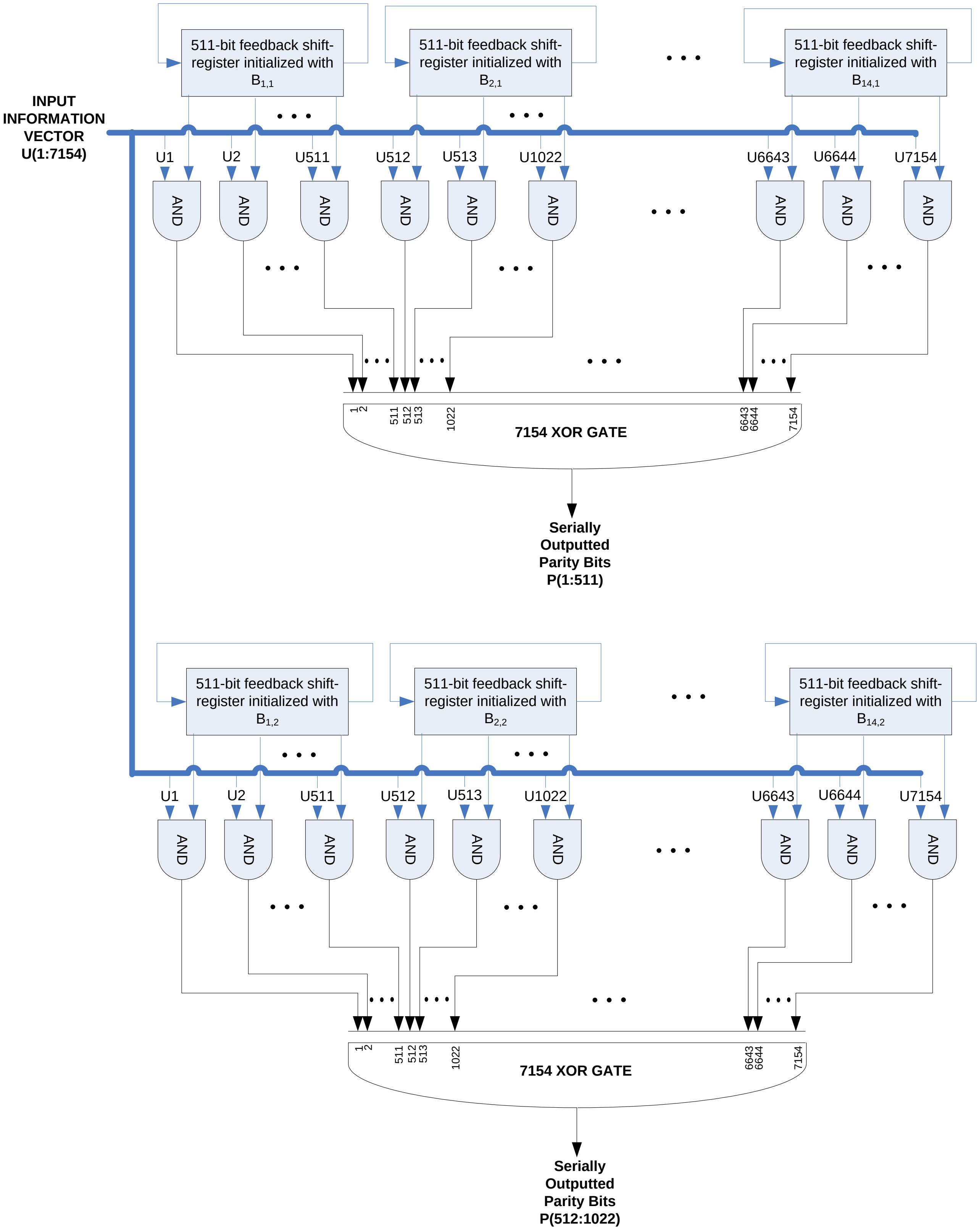
INPUT INFORMATION VECTOR U(1:7154)
511-bit feedback shift-register initialized with $B_{1,1}$
511-bit feedback shift-register initialized with $B_{2,1}$
511-bit feedback shift-register initialized with $B_{14,1}$
U1
U2
U511
U512
U513
U1022
U6643
U6644
U7154
AND
1
2
511
512
513
1022
6643
6644
7154
7154 XOR GATE
Serially Outputted Parity Bits P(1:511)
511-bit feedback shift-register initialized with $B_{1,2}$
511-bit feedback shift-register initialized with $B_{2,2}$
511-bit feedback shift-register initialized with $B_{14,2}$
7154 XOR GATE
Serially Outputted Parity Bits P(512:1022)

Figure 7. Encoder Diagram

4.4 Shortened (8160, 7136) Code Standard

The code described in section 4.2 shall be shortened to the dimensions of (8160, 7136) by the method outlined in this section.

Using the generator matrix given by Figure 6, an encoder can be implemented using circuits described in Section 4.3 and in [8]. This encoder generates a (8176, 7154) LDPC subcode of the (8176, 7156) code. Current spacecraft and ground systems manipulate and process data at 32-bit computer word size. Neither (8176, 7154) or (8176, 7156) is a multiple of 32. It is beneficial to shorten the codeword to the dimensions of (8160, 7136). In other words, by shortening the information sequence to 7136 through the use of 18 bits of virtual fill, the (8176, 7154) subcode encoder can be used. This is accomplished by encoding the virtual fill bits with zeros but not transmitting them; thus the total codeword length becomes 8158. Note that it is not necessary to add two independent rows to the generator matrix to encode the full (8176, 7156) code because these bits would be shortened anyway and so the subcode is sufficient and less complicated for this application. Since the codelength of 8158 is two bits shy of 8160, an exact multiple of 32, two bits of actual transmitted zero fill are appended to end of the codeword to achieve a shortened code dimension of (8160, 7136) bits or (1020, 892) octets or (255, 223) 32-bit words. The shortened codeword is shown in Figure 8.

The received shortened codeword would require the removal of the 2 zero fill bits prior to decoding. The decoder would then reproduce the 18 virtual fill zeros after processing but would, in general, not pass these 18 zeros on to the ground equipment.

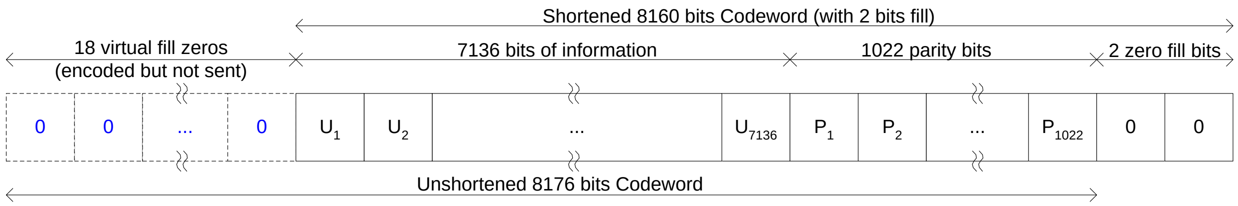

Figure 8. Shortened Codeword Standard

4.5 Randomization and Synchronization

The use of the (8160, 7136) LDPC standard code is systematic and hence does not guarantee sufficient randomization to avoid receiver acquisition problems, acquire or maintain bit (symbol) synchronization, and avoid decoder ambiguities. Therefore, a pseudo-randomizer shall be used after encoding in accordance to CCSDS recommendation 131.0-B-2, TM Synchronization and Channel Coding, Blue Book, August 2011 Section 9 [9].

In addition, codeword synchronization is necessary so that the receiver can identify the beginning of the codeword for proper de-randomizing and decoding. There are two cases covered in this standard, 1. transfer frames aligned with the codeword and 2. a stream of transfer frames each preceded by an ASM specified in section 8.3.3 of [9] and the byte stream sliced to fit the pieces in the codeword message area.

Case 1

The use of an attached sync marker (ASM) as specified in CCSDS recommendation Section 8.2 [9] shall be used. Note that for this case the ASM is not pseudo-randomized.

Case 2

The use of a code sync marker (CSM) using the patterns specified for the ASM in Section 8.3 [9] shall be used.

Note that for this case the ASM in the byte stream is pseudo-randomized but the CSM is not pseudo-randomized.

5. GUIDANCE

5.1 Reference Documents

References:

[1] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *Electro. Lett.*, vol. 32, pp. 1645-1646, Aug. 1996.

[2] R. G. Gallager, “Low density parity check codes,” *IRE Trans. Inform. Theory*, IT-8, pp. 21-28, Jan. 1962.

[3] T. Richardson and R. Urbanke, “Design of capacity-approaching low density parity check codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.

[4] Y. Kou, S. Lin, and M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Trans. Information Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[5] W. Fong, “White Paper for Low Density Parity Check (LDPC) Codes for CCSDS Channel Coding Blue Book,” CCSDS P1B Channel Coding Meeting: Houston, TX, Oct. 2002.

[6] J. Heo, "Analysis of scaling soft information on low density parity check code," *Electro. Lett.*, vol. 39, pp. 219-221, Jan. 2003.

[7] S. Lin and D. Costello, Jr. *Error Control Coding, 2nd Ed.* New Jersey: Pearson Prentice Hall, 2004.

[8] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient Encoding of Quasi-Cyclic Low Density Parity Check Codes," IEEE *Transactions on Communication*, vol. 53, pp. 71-81, Jan. 2006.

[9] CCSDS recommendation 131.0-B-2, TM Synchronization and Channel Coding. Blue Book. August 2011 (or later versions), http://www.ccsds.org.

5.2 Key Word Listing

Channel Coding,

Error Correction Coding,

Bandwidth Efficiency,

Low Density Parity Check Codes

APPENDIX A

GENERATOR MATRIX CIRCULANT TABLE (NORMATIVE)

A.1 Specification

Table 2. Table of Circulants for the Generator Matrix

Circulant	1st row of circulant
$B_{1,1}$	55BF56CC55283DFEEFEA8C8CFF04E1EBD9067710988E25048D67525426939E2068D2DC6FCD2F822BEB6BD96C8A76F4932AAE9BC53AD20A2A9C86BB461E43759C
$B_{1,2}$	6855AE08698A50AA3051768793DC238544AF3FE987391021AAF6383A6503409C3CE971A80B3ECE12363EE809A01D91204F1811123EAB867D3E40E8C652585D28
$B_{2,1}$	62B21CF0AEE0649FA67B7D0EA6551C1CD194CA77501E0FCF8C85867B9CF679C18BCF7939E10F8550661848A4E0A9E9EDB7DAB9EDABA18C168C8E28AACDDEAB1E
$B_{2,2}$	64B71F486AD57125660C4512247B229F0017BA649C6C11148FB00B70808286F1A9790748D296A593FA4FD2C6D7AAF7750F0C71B31AEE5B400C7F5D73AAF00710
$B_{3,1}$	681A8E51420BD8294ECE13E491D618083FFBBA830DB5FAF330209877D801F92B5E07117C57E75F6F0D873B3E520F21EAFD78C1612C6228111A369D5790F5929A
$B_{3,2}$	04DF1DD77F1C20C1FB570D7DD7A1219EAECEA4B2877282651B0FFE713DF338A63263BC0E324A87E2DC1AD64C9F10AAA585ED6905946EE167A73CF04AD2AF9218
$B_{4,1}$	35951FEE6F20C902296C9488003345E6C5526C5519230454C556B8A04FC0DC642D682D94B4594B5197037DF15B5817B26F16D0A3302C09383412822F6D2B234E
$B_{4,2}$	7681CF7F278380E28F1262B22F40BF3405BFB92311A8A34D084C086464777431DBFDDD2E82A2E6742BAD6533B51B2BDEE0377E9F6E63DCA0B0F1DF97E73D5CD8
$B_{5,1}$	188157AE41830744BAE0ADA6295E08B79A44081E111F69BBE7831D07BEEBF76232E065F752D4F218D39B6C5BF20AE5B8FF172A7F1F680E6BF5AAC3C4343736C2
$B_{5,2}$	5D80A6007C175B5C0DD88A442440E2C29C6A136BBCE0D95A58A83B48CA0E7474E9476C92E33D164BFF943A61CE1031DFF441B0B175209B498394F4794644392E
$B_{6,1}$	60CD1F1C282A1612657E8C7C1420332CA245C0756F78744C807966C3E1326438878BD2CCC83388415A612705AB192B3512EEF0D95248F7B73E5B0F412BF76DB4
$B_{6,2}$	434B697B98C9F3E48502C8DBD891D0A0386996146DEBEF11D4B833033E05EDC28F808F25E8F314135E6675B7608B66F7FF3392308242930025DDC4BB65CD7B6E
$B_{7,1}$	766855125CFDC804DAF8DBE3660E8686420230ED4E049DF11D82E357C54FE256EA01F5681D95544C7A1E32B7C30A8E6CF5D0869E754FFDE6AEFA6D7BE8F1B148

$B_{7,2}$	222975D325A487FE560A6D146311578D9C5501D28BC0A1FB48C9BDA173E869133A3AA9506C42AE9F466E85611FC5F8F74E439638D66D2F00C682987A96D8887C
$B_{8,1}$	14B5F98E8D55FC8E9B4EE453C6963E052147A857AC1E08675D99A308E7269FAC5600D7B155DE8CB1BAC786F45B46B523073692DE745FDF10724DDA38FD093B1C
$B_{8,2}$	1B71AFFB8117BCF8B5D002A99FEEA49503C0359B056963FE5271140E626F6F8FCE9F29B37047F9CA89EBCE760405C6277F329065DF21AB3B779AB3E8C8955400
$B_{9,1}$	0008B4E899E5F7E692BDCE69CE3FAD997183CFAEB2785D0C3D9CAE510316D4BD65A2A06CBA7F4E4C4A80839ACA81012343648EEA8DBBA2464A68E115AB3F4034
$B_{9,2}$	5B7FE6808A10EA42FEF0ED9B41920F82023085C106FBBC1F56B567A14257021BC5FDA60CBA05B08FAD6DC3B0410295884C7CCDE0E56347D649DE6DDCEEB0C95E
$B_{10,1}$	5E9B2B33EF82D0E64AA2226D6A0ADCD179D5932EE1CF401B336449D0FF775754CA56650716E61A43F963D59865C7F017F53830514306649822CAA72C152F6EB2
$B_{10,2}$	2CD8140C8A37DE0D0261259F63AA2A420A8F81FECB661DBA5C62DF6C817B4A61D2BC1F068A50DFD0EA8FE1BD387601062E2276A4987A19A70B460C54F215E184
$B_{11,1}$	06F1FF249192F2EAF063488E267EEE994E7760995C4FA6FFA0E4241825A7F5B65C74FB16AC4C891BC008D33AD4FF97523EE5BD14126916E0502FF2F8E4A07FC2
$B_{11,2}$	65287840D00243278F41CE1156D1868F24E02F91D3A1886ACE906CE741662B40B4EFDFB90F76C1ADD884D920AFA8B3427EEB84A759FA02E00635743F50B942F0
$B_{12,1}$	4109DA2A24E41B1F375645229981D4B7E88C36A12DAB64E91C764CC43CCEC188EC8C5855C8FF488BB91003602BEF43DBEC4A621048906A2CDC5DBD4103431DB8
$B_{12,2}$	2185E3BC7076BA51AAD6B199C8C60BCD70E8245B874927136E6D8DD527DF0693DC10A1C8E51B5BE93FF7538FA138B335738F4315361ABF8C73BF40593AE22BE4
$B_{13,1}$	228845775A262505B47288E065B23B4A6D78AFBDDB2356B392C692EF56A35AB4AA27767DE72F058C6484457C95A8CCDD0EF225ABA56B7657B7F0E947DC17F972
$B_{13,2}$	2630C6F79878E50CF5ABD353A6ED80BEACC7169179EA57435E44411BC7D566136DFA983019F3443DE8E4C60940BC4E31DCEAD514D755AF95A622585D69572692
$B_{14,1}$	7273E8342918E097B1C1F5FEF32A150AEF5E11184782B5BD5A1D8071E94578B0AC722D7BF49E8C78D391294371FFBA7B88FABF8CC03A62B940CE60D669DFB7B6
$B_{14,2}$	087EA12042793307045B283D7305E93D8F74725034E77D25D3FF043ADC5F8B5B186DB70A968A816835EFB575952EAE7EA4E76DF0D5F097590E1A2A978025573E

Note that the numbers in the second column represent the hexadecimal representation of the first row of each circulant. Since there are only 511 possible positions, the leftmost bit is padded with a zero to allow a 128 digit hexadecimal number. Table 2 cannot be as efficiently described as Table 1 due to the fact that the generator circulants do not have a low density of 1's.

APPENDIX B

COMPLEXITY (INFORMATIVE)

B.1 FPGA Statistics

The complexity of LDPC codes has been an area of research and discussion. For an FPGA or application specific integrated circuit (ASIC) implementation, the encoder's complexity are dominated by two factors: 1. the total number of required logic gates and 2. The routing complexity. For the code presented in this document, the quasi-cylic property allows for the use of shift registers whose required number of logic gates is proportional to n-k [8] or 8176-7156= 1020 (unshortened). In regards to the routing complexity, there is currently no way to predict this figure and would depend on a number of factors such as the choice of the FPGA or ASIC, routing algorithm and the layout of the device.

The decoder's complexity is larger than the encoder's and even more difficult to predict. The primary complexity factors (the total number of required logic gates and the routing complexity) are a function of the choice of Belief Propagation (BP) decoding algorithms [4, 6, 7] as well as the architectural decisions (i.e. parallel or serial processing, number of bits of finite precision, fixed number of iterations or stopping rule, use of look up tables, etc.) These choices also determine the decoder's bit error rate (BER) performance.

For the development of the base (8176, 7156) code, an FPGA implementation was used to confirm the software simulations. A Xilinx 8000 Virtex-2 FPGA was used for the test. The device contained both the encoder and decoder. The decoder algorithm was a Scaled Min-Sum parallel BP decoder (SMSPD) described in [6]. The encoder algorithm was a shift register based encoder described in [8]. An architectural evaluation was performed prior to implementation to produce a quasi-optimal implementation based on routing, logic requirements and BER performance.

The FPGA had the following statistics: 1. encoder used 2,535 logic slices out of 46,592 available or 5.4% and 4 memory blocks out of 168 available or 2.4%; 2. decoder used 21,803 logic slices out of 46,592 or 46.8% and 137 memory blocks out of 168 or 81.5%. The number of logic slices is an aggregate measure of the number of logic gates required and the routing complexity while the memory blocks figure is the number of dedicated FPGA memory blocks used. It is clear from these statistics that the encoder is of much lower complexity than the decoder using only 5.4% of the logic slices resources while the decoder requires 46%.

Appendix C summarizes the test results.

APPENDIX C

FPGA TEST RESULTS (INFORMATIVE)

C.1 Bit and Block Error Rate Performance

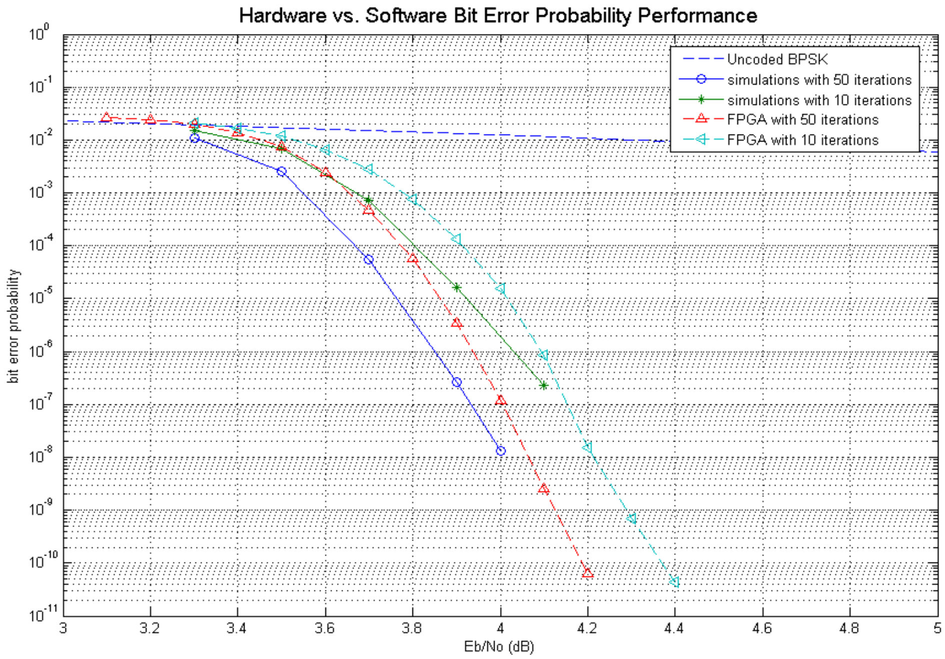

Figure 9. Bit Error Rate Test Results

Figure 9 shows the BER and Figure 10 shows the Block Error Rate (BLER) test results for 50 and 10 maximum iterations from an FPGA implementation of the base (8176, 7156) code. Note that for both cases the difference between simulations and hardware tests was 0.1 dB or less.

The encoder data rate was limited to 2 x system clock while the decoder operated at 14 x system clock / number of iterations. For testing, the system clock was set to 100 MHz, so for 10 iterations, the decoder operated at 140 Mbps. Although, the shortened (8160, 7136) was not tested, it is reasonable to say that the base (8176, 7156) code and the shortened (8160, 7136) standard code will have similar results.

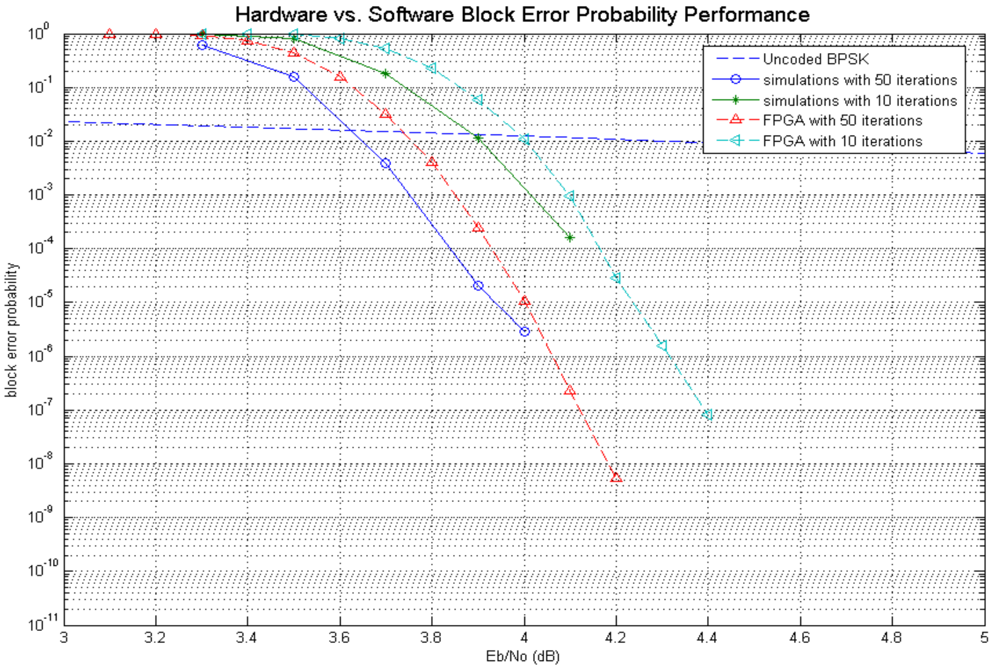

Figure 9. Block Error Rate Test Results