

LOW DENSITY PARITY CHECK ERROR CORRECTING LIMITATION TESTING

Scott Wolfson
Greysen Blumkin
Alvia Sandberg
U.S. Army Redstone Test Center
Redstone Arsenal, AL 35898-8052
scott.c.wolfson.civ@mail.mil

ABSTRACT

The continual advances in military system technologies compel the Test & Evaluation community to constantly mature and adapt available test methodologies. This process requires a clear understanding of methodology capabilities and limitations. The primary objective of this paper is to provide details pertaining to the error correcting capabilities of the six Low Density Parity Check variants identified in the IRIG 106 telemetry standard. This analysis will be performed with representative telemetry equipment in a controlled environment where the number of induced transmitted bit errors will be increased until error correction failures are observed. This study will also investigate the effects, if any, of consecutive versus sparsely induced bit errors on correction capabilities.

INTRODUCTION

In communications, Low Density Parity Check (LDPC) is a Forward Error Correction (FEC) technique used to detect and correct bit errors in received data. First developed by Robert Gallager in 1963 and rediscovered in 1996, LDPC has been implemented in numerous communication protocols which include satellite communications, Wi-Fi 802.11 and 10G Ethernet [1]. The primary advantages of using FEC are increases in data integrity for simplex communication channels and retransmit reductions in duplex communication channels while the primary disadvantage is erroneous data if bit errors exceed the limitations of the selected FEC schema [2]. While FEC is a broad term and includes many error correction techniques, LDPC is classified as a block code where redundant error correction bits, or parity bits, are appended to a fixed size data block prior to transmission. When applied to a simplex telemetry link, six schemas have been identified and approved for use in the IRIG 106 standard [3]. While implementing these LDPC variants in a telemetry encoder and researching the algorithm, we discovered a lack of empirical data detailing the error correction capabilities and limitations of the algorithm. Instead, while performing a literature review on this topic we identified numerous journal publications detailing noisy channel performance and Monte Carlo simulations [4,5,6].

LDPC TELEMETRY IMPLEMENTATION

Details pertaining to the implementation of an LDPC encoder can be found in Appendix 2-D of the IRIG 106 standard. To summarize, the encoder generates unique parities based on the block of information data provided and these parity bits are appended to the information data prior to randomization and transmission. The six LDPC schemes currently approved for telemetry use are defined by the size of the information block and the parity length. These six variants are presented in Table 1.

Table 1 Data Block and Parity Sizes for Telemetry LDPC Variants

Information Block Size (bits)	Parity Length (bits)		
	Rate 1/2	Rate 2/3	Rate 4/5
1024	1024	512	256
4096	4096	2048	1024

To generate the unique parity, circulant matrices are used. These circulant matrices are provided in the IRIG 106 standard. The IRIG 106 standard also details the method for generating the circulant matrices for each of the six variants based on the provided sparse parity check matrices and is beyond the scope of this paper.

Figure 1 depicts the parity generation process where the variables K , M , m and r are set based on the LDPC variant selected. As illustrated, the unique parity is generated by performing a recursive bitwise XOR if the input information data bit is a logic 1 and accumulating the results. This recursive XOR and accumulation process continues for the entire 1024 or 4096 bit information data block resulting in the unique parity.

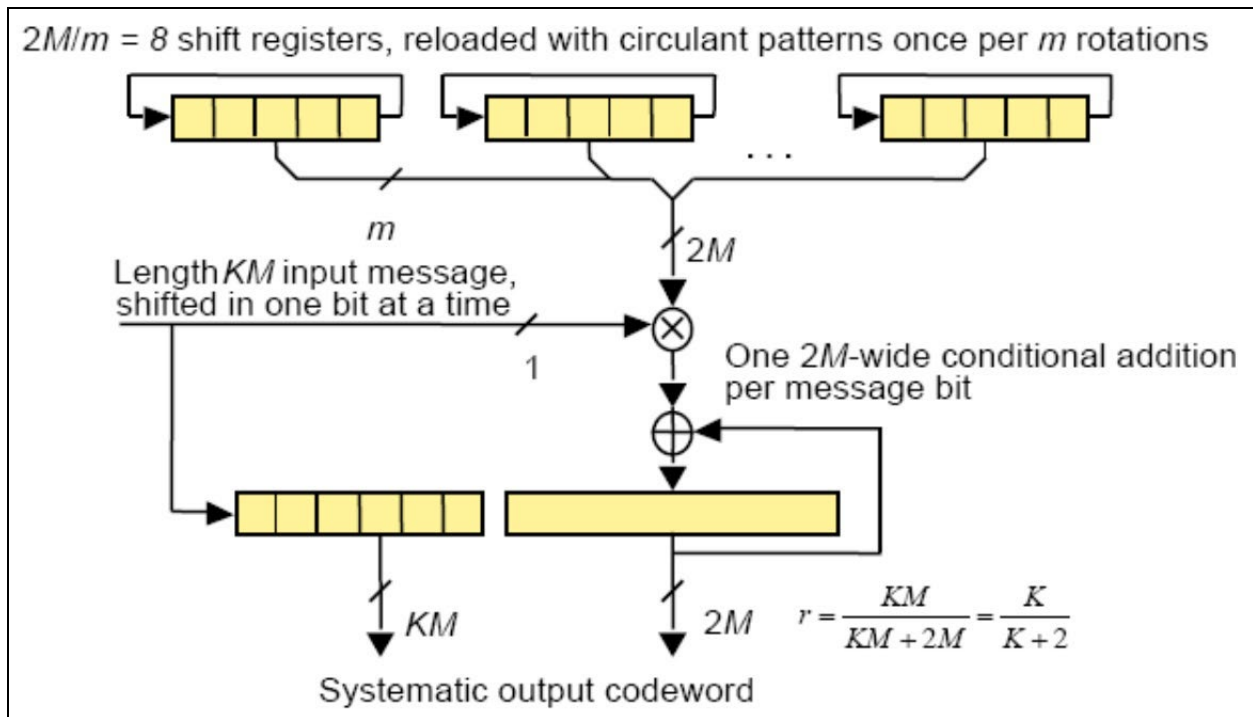


Figure 1 Quasi-Cyclic Encoder Using Feedback Shift Register [3]

Similar to standard telemetry encoder designs, the data block structures of LDPC encoder schemes contain a sync word and are randomized prior to transmission; however, notable differences do exist. The first difference is the prepended sync word. For LDPC the sync word, or Attached Synchronization Marker (ASM), is limited to one of two options and is based on the information data block size. The fact that the ASM (as defined in the LDPC standard) is not randomized is the second noteworthy difference. Figure 2, Table 2 and Table 3 have been included to illustrate the randomization and transmitted ASM/codeblock structure of LDPC.

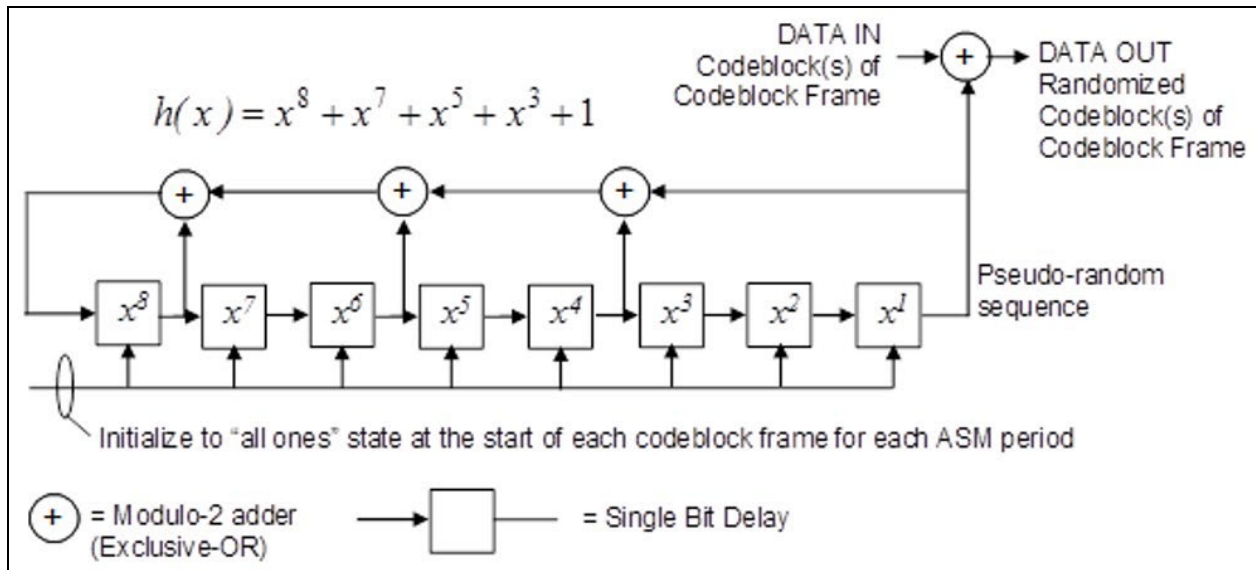


Figure 2 Codeblock Randomizer [3]

Table 2 Transmitted Data Structure [3]

Block Size (bits)	
1024	64-bit ASM + Randomized Data Block and Parity bits
4096	256-bit ASM + Randomized Data Block and Parity bits

Table 3 Attached Synchronization Markers (ASM) [3]

Block Size (bits)	
1024	0xFCB88938D8D76A4F
4096	0xFCB88938D8D76A4FFCB88938D8D76A4F034776C7272895B0FCB88938D8D76A4F

LDPC ALGORITHM TEST SCENARIOS

Using the information provided in the IRIG 106 telemetry standards, an LDPC encoder was designed using firmware and instantiated on a Field Programmable Gate Array (FPGA). The firmware was designed with LDPC testability features allowing the user to select which LDPC variant as well as providing dynamic insertion of bit errors and bit error locations after the parity is calculated and prior to data randomization and transmission.

The test setup used to observe and determine the error correcting abilities of each variant is illustrated in Figure 3. Two decoms were used, one decom was used to display the data information block after LDPC decoding and error correction, and the other decom was used to display the fully assembled blocks consisting of the ASM, information data block, and parity bits. This provided the ability to verify bit errors were properly inserted into the data block (via Decom 2), and properly corrected by the LDPC decoder in the TM Receiver (via Decom 1). Derived math parameters were also used to provide a real-time evaluation of the received data pattern and detect when bit errors were not being properly corrected by the LDPC decoder algorithm.

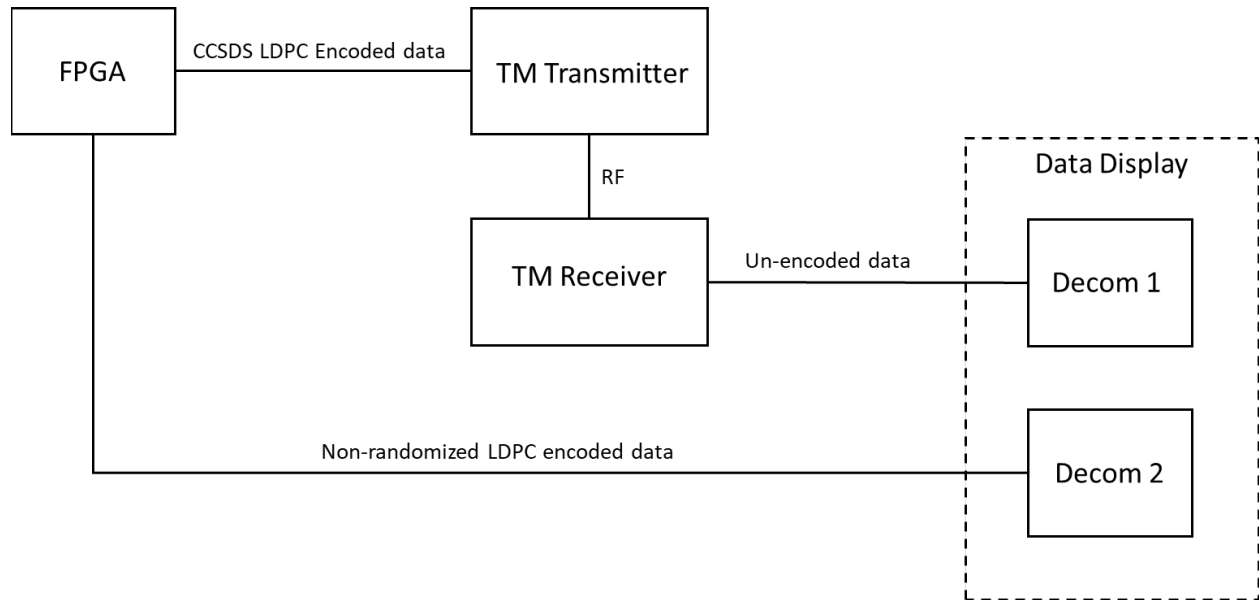


Figure 3 Test Setup

To perform the testing, a fixed information data block, summarized in Figure 4, was used to generate the parity bits for each variant. Prior to error insertion, the two decoms were used to verify no errors existed in either data path. A single bit error was then introduced at the beginning of the transmitted information data block. Again, the two decoms were used to verify the error was properly inserted and corrected by the LDPC decoder functionality of the receiver. The number of consecutive inserted bit errors were then increased until anomalies were detected at the receiver thus indicating the error correction capabilities of the LDPC decoder had been reached. This test was repeated for all six LDPC variants. Subsequent testing was performed with consecutive and non-consecutive bit errors inserted at multiple locations within the information data block.

Byte Number	1	2	3	4	5	6	$\frac{K}{8} - 3$	$\frac{K}{8} - 2$	$\frac{K}{8} - 1$	$\frac{K}{8}$
Value	0x0000	0x0001	0x0002	$\frac{K}{16} - 4$	$\frac{K}{16} - 3$	0xFE6B	0x2840

Figure 4 Information Data Block, K = Information Block Size

Additional testing was performed to determine whether the performance of the LDPC variants are affected by the content of the data. To do this, the information data block was changed from a zero-dominant pattern to a one-dominant pattern. The resulting information data block pattern started at a value of 0xFFFF in the first two bytes and was decremented until the minimum value allowed by each block size was reached. A subset of the previous consecutive and non-consecutive bit error tests were repeated.

LDPC TEST RESULTS

The first test conducted using the previously described setup, FPGA firmware and zero-dominant data block was designed to test how many consecutive bit errors within the data block were necessary to cause the LDPC variant to fail to correct the bit errors at the receiver. This test scenario was conducted on all 6 IRIG 106 LDPC variants. There were three slightly different consecutive methodologies applied in order to test the location dependency of the consecutive bit error throughout the data block. First, consecutive bit errors were added to the beginning of the data block until errors were detected by the decom from the LDPC decoded waveform. The second methodology applied consecutive bit errors to the middle of the data block until errors were detected. The third methodology was to apply consecutive bit errors to the end of the data block until errors were detected. Note that for this third variation, there was no carry-over of applied data block bit errors to any subsequent data blocks and the final applied consecutive bit error coincided with the final data block bit. The resulting number of consecutive bit errors that were corrected by each variant using these test methodologies are shown below in Table 4. For example, for the case of Block Size = 1024, Rate 4/5, this LDPC variant corrected 23 consecutive bit errors at the beginning of the data block. However, on the 24th applied consecutive bit error, the decom displayed errors in the resulting decoded data block. This was the original and main testing goal of this paper and thus, most of the focus and effort went into determining the failure points of data block bit errors for each variant. All subsequent tests were slight deviations from this main goal and consecutive bit error methodology. It's a notable observation that the results shown in Table 4 were repeatable and there was statistically no variation in the resulting numbers.

Table 4 Consecutive Bit Errors Successfully Corrected

	Block Size = 1024 bits			Block Size = 4096 bits		
	Rate 4/5	Rate 2/3	Rate 1/2	Rate 4/5	Rate 2/3	Rate 1/2
Beginning of Block	23	52	162	87	227	688
Middle of Block	22	71	113	81	293	469
End of Block	25	55	117	111	226	472

The next test conducted involved the same setup and procedure as the consecutive bit error testing with the main difference of separating the applied bit errors by a gap of correct bits at increasing values between applied bit errors. The main purpose of this test was to determine if non-consecutive bit errors were easier for the LDPC variants to correct versus consecutive. Table 5 shows the results of using this methodology for the case of Block Size = 1024, Rate 4/5. The notable observation was that there was no significant variance in the overall number of bit errors that could be corrected using this non-consecutive bit error methodology by the LDPC

coding when compared to the similar consecutive bit error methodology. This observation was confirmed throughout the other variants but not recorded in this paper. While it may be possible to observe other subtle anomalies and characteristics with non-consecutive bit errors, it was not determined to be a worthwhile investigative study for this paper.

Table 5 Non-Consecutive Bit Errors Successfully Corrected

Block Size = 1024 bits	Error every nth bit				
Rate 4/5	n=2	n=3	n=4	n=5	n=6
Beginning of Block	22	26	23	15	26

The next test conducted involved repeating many of the consecutive bit error scenarios for a ones-dominant data block versus the original zero-dominant data block. It was not surprising to observe that the ones-dominant data block did not produce any significant difference to the results shown in Table 4. In fact, the results were statistically identical. The most noticeable observation from this was that the data content did not appear to change the results when it comes to the data values being heavily favored to either ones or zeros.

While we had access to the test setup and the FPGA code, we took the opportunity perform a few special bonus tests just for pure scientific curiosity. The first bonus test was to add consecutive bit errors to the parity after the correct parity had been generated using a Block Size = 1024, Rate 4/5. The receiver was able to correctly decode up to 12 consecutive bit errors at the beginning of the parity section without re-interpreting any of the information data block bits as a result. As a additional observation, adding an additional 5 consecutive bit errors to the information data block was also able to be correctly decoded by the receiver. The second bonus test or observation was conducted by adding a single bit error to the ASM which the receiver was able to correct and decode. Additional destructive testing of the parity and/or ASM section of the LDPC message could be completed and may be interesting but was not a focus of this paper.

One notable observation from this testing was that when any particular variant of LDPC would reach or surpass the point that all the bit errors could be corrected, the resulting data blocks did not always “fail” to the same value. In other words, bad or corrupted data was random not predictable or repeating values. This is very likely due to a vendor specific implementation of the LDPC decoder and correlation technique. The processes used to correlate and correct the bits combined with our testing method of providing very “healthy looking” bit errors was a big reason for this observation. Much more could be reported on these details; however, this would change the focus of this testing and is not the reason for this paper.

Based on the testing conducted through the processes described in this paper, an approximate error tolerance can be calculated for each IRIG 106 LDPC variant. These numbers are likely quite subjective given the limited testing and limited resources. However, given all of those assumptions and limitations, the observed approximate error tolerances are shown in Table 6. Note that there are many pros and cons to each LDPC variant so this is only one piece of information to consider when determining which variant may be most appropriate for any particular scenario.

Table 6 Approximate Percentage Error Tolerance Observed

Block Size = 1024 bits			Block Size = 4096 bits		
Rate 4/5	Rate 2/3	Rate 1/2	Rate 4/5	Rate 2/3	Rate 1/2
2.28%	5.79%	12.76%	1.90%	5.07%	11.07%

CONCLUSIONS

As expected, the LDPC algorithms approved for use in the IRIG 106 telemetry standard demonstrated error correction capabilities. The material presented in this paper provides supplemental empirical data that compares and contrasts the error correction algorithm limitations of each of the six LDPC variants and is intended to aid telemetry encoder designers when selecting the appropriate LDPC variant based on test requirements. To summarize, we discovered the location of consecutive and non-consecutive bit errors in the information data resulted in some variances in error correction abilities when it comes to beginning, middle and end of the data block. Contrarily, having consecutive versus non-consecutive bit errors and data content (i.e. majority 1 or majority 0) had little to no impact on error correction abilities. Additionally, we discovered the LDPC algorithms tolerated bit errors occurring in the parity bits and did not erroneously change/correct bits in the information data. Based on the findings presented and a review of FEC trends, additional research is recommended to determine if emerging error correction schemes should be considered for use in telemetry.

REFERENCES

- [1] Low-Density-Parity-Check Code, Wikipedia, 2021. (Available on-line at https://en.wikipedia.org/wiki/Low-density_parity-check_code).
- [2] Forward Error Correction (FEC), Tutorialspoint, 2021. (Available on-line at <https://www.tutorialspoint.com/forward-error-correction-fec>).
- [3] Range Commanders Council Telemetry Group, Range Commanders Council, White Sands Missile Range, New Mexico, *IRIG Standard 106-20: Telemetry Standards*, 2020. (Available on-line at <https://www.wsmr.army.mil/RCCsite/Pages/Publications.aspx>).
- [4] Consultive Committee for Space Data Systems (CCSDS), "Low density parity check codes for use in near-Earth and deep space applications (131.1-O-2 Orange Book)," Sep. 2007.
- [5] P. Santini, M. Battaglioni, M. Baldi and F. Chiaraluce, "Analysis of the error correction capability of LDPC and MDPC codes under parallel bit-flipping decoding and application cryptography," in *Proc. IEEE International Conference on Communications*, vol. 68, no. 8, pp. 4648–4660, Aug. 2020.
- [6] S. K. Chilappagari, D. V. Nguyen, B. Vasic and M. W. Marcellin, "On the guaranteed error correction capability of LDPC codes," in *Proc. 2008 IEEE International Symposium on Information Theory*, 2008, pp. 434-438, doi: 10.1109/ISIT.2008.4595023.