Minimum Sum Algorithm Decoder for LDPC Nonregular Parity Check Matrix in BPSK System

Yi Hua Chen, **Jue Hsuan Hsiao**, **Zong Yi Saio**, **and Hua Ting Syu**
Oriental Institute of Technology
Institute of Information and Communication Engineering, New Taipei, Taiwan

Abstract— Referring to the approximate lower triangular low-density parity-check code check matrix of the IEEE P802.11n™/D1.04 (Part 11: Wireless Local Area Network Medium Access Control and Physical Layer specifications), this study established a decoder based on LabVIEW program language on a single program architecture that can adjust the transmission end to generate diverse codeword patterns, including three subblock sizes (27, 54, and 81 bit) and four code rates (1/2, 2/3, 3/4, and 5/6). Combined with the minimum sum algorithm (MSA), the decoder completed decoding tasks by changing the check node and variable node structures on the basis of the selected subblock size and code rate. In addition to providing an introduction on the decoding mechanism of the MSA and completing decoding program optimization and analysis of bit error rate(BER) performance curves, this study applied the LabVIEW program to simulate the BER of the ratio of energy per bit to the spectral noise density (Eb/No) at each point from 0 to 10 dB, when subblock sizes (27, 54, and 81 bit) were combined with code rates (1/2, 2/3, and 5/6) operating in an additive white Gaussian noise channel environment. The error rate performance curve diagrams of two studies were referenced (regular weight (3, 6) and 802.11n irregular subblock size 27 bit combined with code rate 5/6) and compared with the simulation outcome yielded in this study. The result showed that the subblock size did not affect the error rate, but the code rates substantially affected the error rates. When the code rate was set to 1/2, the error correction performance of the irregular check matrix was considerably higher than that of the regular check matrix.

1. INTRODUCTION

Shannon proposed Shannon's theory in 1948 [1], stating that by transmitting information bit through channel coding and maintaining the data rate (R_b) within the range of the channel capacity (C), the bit error rate (BER) resulting from data passing through the channel can be effectively reduced. When the encoded codeword lengthens, the data BER can approach infinitesimal. This limit is called Shannon limit [2]. Several studies have identified a considerable amount of error correcting codes [3]. In particular, the low-density parity-check (LDPC) code [4] proposed by Gallager has been the most widely applied in recent years. Despite the complex coding and decoding calculation, the LDPC code transmits at a data rate closest to the Shannon limit channel capacity. This study applied the approximate lower triangular LDPC code check matrix combined with the system architecture illustrated in Figure 1. After the information bit is encoded, binary phase-shift keying (BPSK) modulation is used. The bit then passes through the additive white Gaussian noise (AWGN) channel to the receiver end where BPSK demodulation is performed. Finally, the minimum sum algorithm (MSA) decoder [5] was used to decode the information bit.

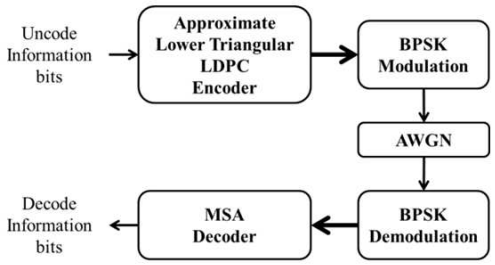

Figure 1: Diagram of the encoding and decoding system architecture.

The MSA decoding method used in this study was generated according to a simplified sum product algorithm (SPA) [6]. Unlike SPA, MSA requires no complex computation during decoding, thus increasing the speed of decoding calculation.

In this study, the MSA decoding method was applied to the approximate lower triangular LDPC encoding specifications specified in IEEE 802.11n for a wireless local area network (WLAN) [7]. The appendix in the specification records 12 code check matrices, including combinations of three subblocks (27, 54, and 81 bit) and four code rates (1/2, 2/3, 3/4, and 5/6). The MSA decoder proposed in this study integrated all 12 combinations of code check matrices into the program. Users can input different subblocks and code rates and the system automatically generates codewords for the program to decode. The decoding performance and gain of combinations of subblocks (27, 54, and 81 bit) and code rates (1/2, 2/3, and 5/6) were compared in this study.

2. APPROXIMATE LOWER TRIANGULAR LDPC CODE

2.1. Approximate Lower Triangular LDPC Code Check Matrix Architecture

The LDPC code is a type of linear block code, which is generally encoded by calculating the information bit vector and generate matrix (G) to obtain the redundancy bit. Subsequently, an encoding codeword can be determined by adding information bit and redundancy bit. During decoding, check matrices and codeword calculation were used to correct errors.

Compared with general linear block codes, LDPC codes require only check matrices to complete encoding and decoding. In this study, the encoding method programmed on the basis of the approximate lower triangular LDPC specifications in 802.11n for WLAN in [8] was adopted. This encoding method can adjust three subblocks (27, 54, and 81 bit) and four code rates (1/2, 2/3, 3/4, and 5/6) to generate 12 coding codewords that meet 802.11n specifications. Different subblocks and code rates have different code check matrices. Table 1 lists the matrix sizes.

Table 1: Size of 12 check matrices that meet the 802.11n approximate lower triangular LDPC specifications.

Z \ Code rate	1/2	2/3	3/4	5/6
27	324×648	216×648	162×648	108×648
54	648×1296	432×1296	324×1296	216×1296
81	972×1944	648×1944	486×1944	324×1944

2.2. LabVIEW Implementation of IEEE 802.11n Approximate Lower Triangular LDPC Check Matrix

Richardson and Urbanke proposed approximate lower triangular LDPC coding in 2001 [9]. This encoding method divides the designed check matrix H into six submatrices (A, B, C, D, T, and E). Through formula calculation, redundancy bits (p) p_1 and p_2 can be determined. Adding the information bit vector (m) to the resulting redundancy bits obtains codeword $X = [m\ p_1\ p_2]$. The formulae of the redundancy bits p_1 and p_2 are expressed in (1) and (2).

$$P_1 = ET^{-1}\left(Am^T\right) + Cm^T \tag{1}$$

$$P_2 = T^{-1}\left(Am^T + BP_1{}^T\right) \tag{2}$$

In [8], a detailed explanation indicated that in addition to encoding formulae, implementation flows are required to satisfy the approximate lower triangular LDPC encoding specifications in the IEEE 802.11n for WLAN (Figure 2). The flow was programmed on a LabVIEW platform, and a complete program architecture and packaged SubVI are shown in Figures 3 and 4.

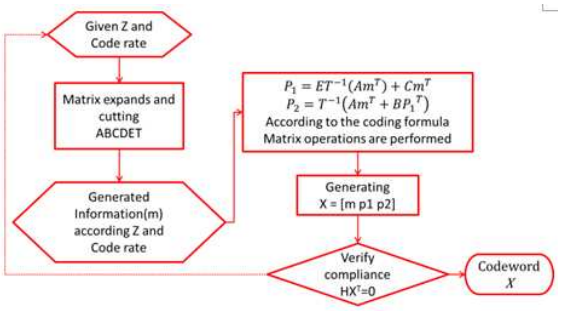

Figure 2: Approximate lower triangular LDPC encoding flowchart.

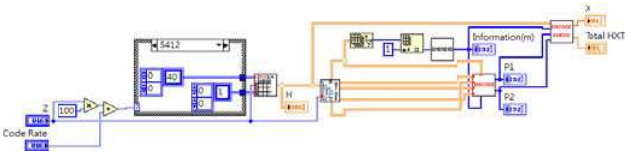

Figure 3: Approximate lower triangular LDPC encoding program architecture.

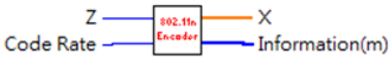

Figure 4: Complete encoding program SubVI.

3. MSA DECODING

During the decoding process, the concept of a Tanner graph [10] must be used to transform the columns and rows in check matrices into variable nodes $B_{(x_i)}$ and check nodes $C_{(x_i)}$ in the Tanner graph and observe 0 and 1 values in check matrices. Check nodes and variable nodes are linked. Through the linked paths, the soft information of signal log likelihood ratio (LLR) is transmitted. However, as shown in Table 1, the size of check matrices is counted in three digits according to the 802.11n approximate lower triangular LDPC specifications. To delineate the decoding method, the following descriptions refer to the weight (3, 6) regular parity-check matrix used in [11] (Figure 5). On the basis of this example check matrix, the generation method of the Tanner graph and the process and steps of transmitting signal LLR soft information are explained.

To clarify the corresponding situations between check and variable nodes, the researchers organized data in Figure 5 and marked $B_{(i)}$ and $C_{(i)}$ on the columns and rows and denoted the 0 and 1 values in check matrices in color blocks (Figure 6). A gray block represents 1 and a white block represents 0. Using check nodes as the main nodes and variable nodes as subsidiary nodes creates six Tanner graphs (Figure 7). By contrast, using variable nodes as the main nodes and check nodes as secondary nodes creates 12 Tanner graphs (Figure 8).

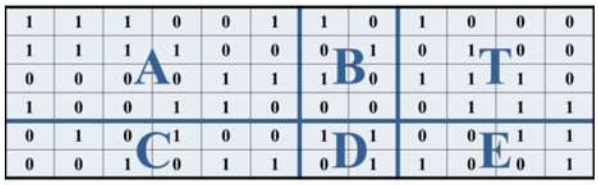

Figure 5: Weight (3, 6) regular parity check matrix.

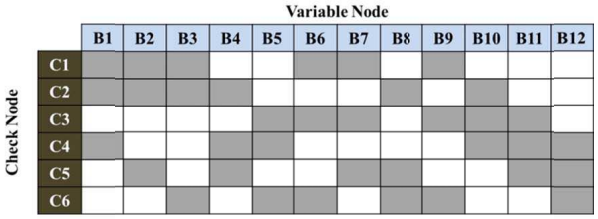

Figure 6: Node corresponding diagram of weight (3, 6) regular parity-check matrix.

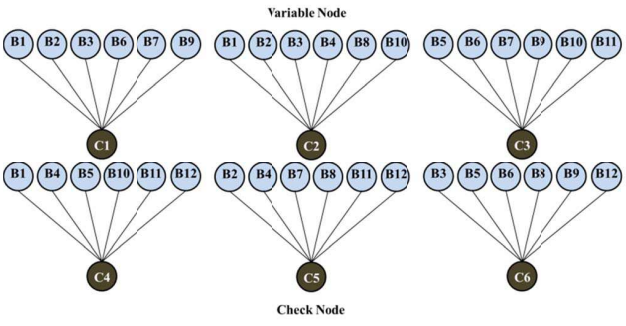

Figure 7: Tanner graph applying check nodes as main nodes and variable nodes as secondary nodes in the weight (3, 6) regular parity-check matrix.

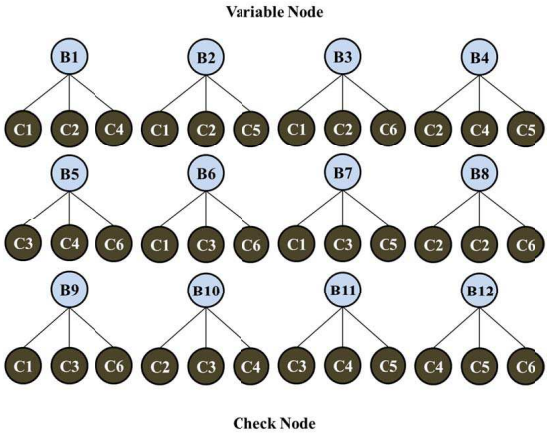

Figure 8: Twelve Tanner graphs applying variable nodes as main nodes and check nodes as secondary nodes in the weight (3, 6) regular paritycheck matrix.

4. USING LABVIEW TO IMPLEMENT MSA DECODING

LabVIEW software was used to implement the MSA decoding flow (Figure 9). To ensure that the MSA flow conforms to the IEEE 802.11n LDPC specifications for WLAN, judgment is required to be incorporated into the four steps to simplify the program. The aforementioned four steps are separately packaged into four SubVIs. The program architecture and SubVI output–input relationship are introduced as follows.

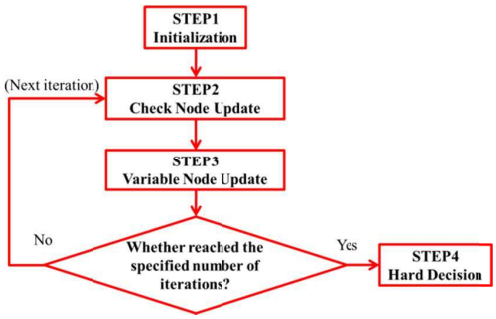

Figure 9: Decoding steps flowchart.

4.1. Transmitting Initialized LLR Soft Information to the Check Matrix

To accommodate the 12 irregular check matrices composed of three subblocks (27, 54, and 81 bit) and four code rates (1/2, 2/3, 3/4, and 5/6), and the three-digit matrix size in the specifications, Tanner graphs were not plotted during the design process. Instead, the graphs were automatically generated by the system during Steps 2 and 3. Thus, some changes were made in Step 1. After calculating the initialized LLR soft information by using (3), the information was not transmitted to variable nodes, but directly substituted into the check matrix. As shown in Figure 10, at the left side in the program structure of Step 1 SubVI, Z (subblock), code rates, σ, and encoded codewords were input to pass through the AWGN channel for information in. At the right output side, the check matrix H, corresponding to Z (subblock) and code rates, initialized LLR soft information, and the outcome of substituting initialized LLR soft information into the check matrix (STEP1) were output. Figure 11 shows the pinout of packaged SubVI.

$$L_{(x_i)} = 2y_i/\sigma^2 \tag{3}$$

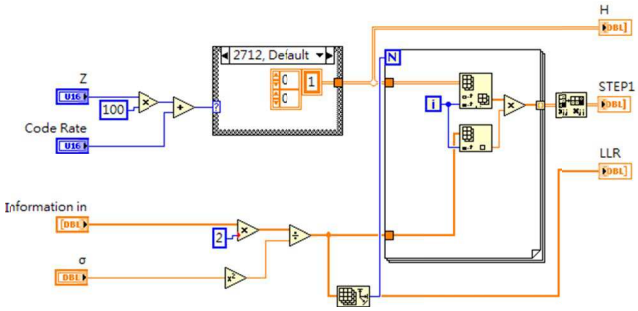

Figure 10: Program architecture of transmitting initialized LLR soft information to the check matrix.

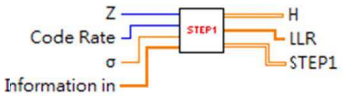

Figure 11: STEP1 SubVI pinout.

4.2. LLR Tanner Graph Generation and Variable Node LLR Soft Information Recalculation

Because the IEEE 802.11n approximate lower triangular LDPC specifications for WLAN has 12 check matrices, the Tanner graphs of each check matrix differ from each other. To simplify the program and save memory, the contacts between each node in the program were not recorded. Instead, the system automatically generated such records when necessary. In Step 2, the Tanner graph in Figure 7 is required and thus generated using the following method: the initialized LLR soft information in Step 1 is substituted into the check matrix. Subsequently, the matrix is decomposed by column to extract nonzero data, thereby obtaining the tanner graph of variable nodes

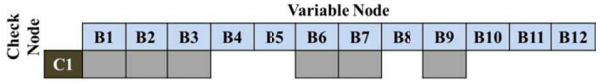

Figure 12: Tanner graph automatic generation method based on an example of weight (3, 6) regular parity-check matrix.

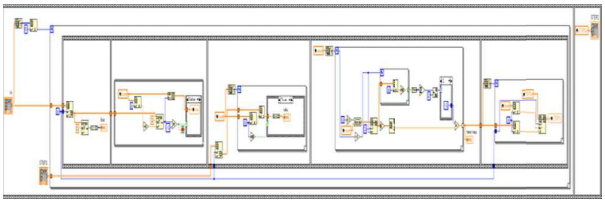

Figure 13: Program architecture of generating Tanner graphs and recalculating variable node LLR soft information.

Figure 14: STEP2 SubVI pinout.

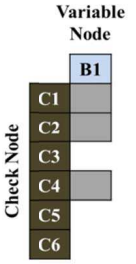

Figure 15: Tanner graph automatic generation method based on an example of weight (3, 6) regular parity-check matrix.

corresponding to single check nodes. Figure 12 shows the weight (3, 6) regular parity-check matrix with automatically generated Tanner graphs.

After the Tanner graph is generated, (4) is used to calculate new LLR soft information and feedback to the check matrix for output. Figure 13 shows the program structure of SubVI in Step 2. Figure 14 shows the pinout of packaged SubVI. At the left side of the figure, the check matrix (H) and output result of Step 1 (STEP1) are the input. At the right side, the result of new LLR soft information feedback to the check matrix is the output (STEP2).

$$\Lambda_{m\to n}(x_i) = \text{sign}\left(\prod_{n'\in N(m)\backslash n} \lambda_{n'\to m}(x_i)\right) \times \min_{n'\in N(m)\backslash n}\left(|\lambda_{n'\to m}(x_i)|\right) \tag{4}$$

4.3. LLR Tanner Graph Generation and Check Node LLR Soft Information Recalculation

This step requires the Tanner graph shown in Figure 8. Similar to the previous step, the system automatically generates the Tanner graph and decomposes the output in the previous step by rows to extract nonzero data. Subsequently, Tanner graphs of check nodes corresponding to single variable nodes can be obtained. Figure 15 shows the Tanner graph automatic generation method based on an example of weight (3, 6) regular parity-check matrix.

After the Tanner graph is generated, (5) is used to calculate new LLR soft information, which is then fedback to the check matrix input. Figure 16 shows the program structure of SubVI in Step 2. Figure 17 shows the pinout of packaged SubVI. At the left side, the check matrix (H), output results of Step 2 (STEP2), and initialized LLR soft information required for formula calculation (LLR) are the input. At the right side, the result of new LLR soft information feedback to the check matrix is the output (STEP3).

$$\lambda_{n\to m}(x_i) = L(x_i) + \sum\nolimits_{m'\in M(n)\backslash m} \mathbf{\Lambda}_{m'\to n}(x_i) \tag{5}$$

4.4. LLR Soft Information Sum

When the iteration reaches the specified number, (6) is used to sum the LLR soft information on the check nodes output in STEP3 and combine it with the initialized LLR soft information.

Figure 18 shows the program structure of SubVI in Step 4. Figure 19 shows the pinout of the packaged SubVI. At the left side, the output results of Step 3 (STEP3) and the initialized LLR

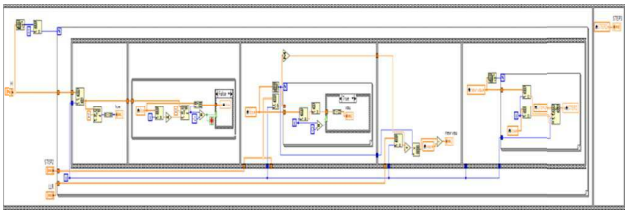

Figure 16: Program architecture of generating Tanner graphs and recalculating check node LLR soft information.

Figure 17: STEP3 SubVI pinout.

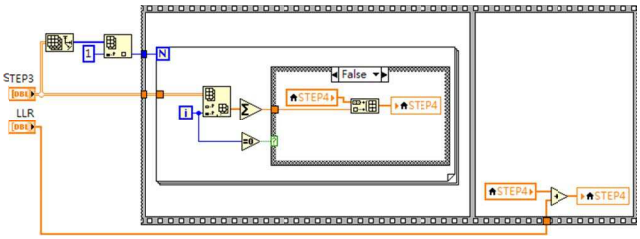

Figure 18: Program architecture of summed LLR soft information on check nodes combined with initialized LLR soft information.

Figure 19: STEP4 SubVI pinout.

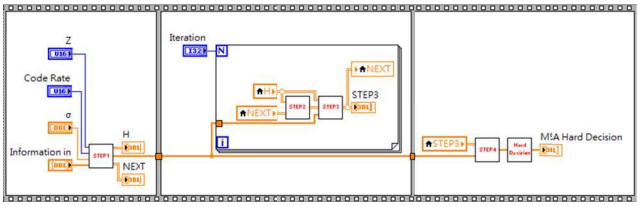

Figure 20: Complete MSA decoding program architecture.

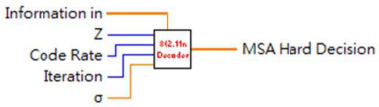

Figure 21: MSA decoding SubVI pinout.

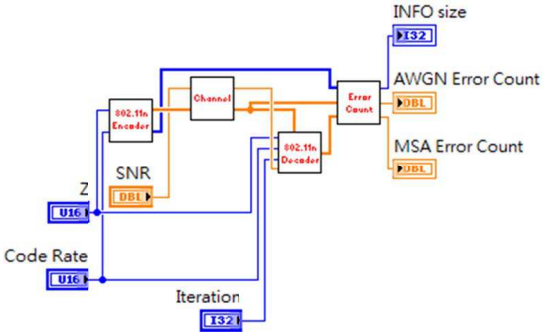

Figure 22: Complete encoding and decoding program architecture meeting IEEE 802.11n LDPC specifications.

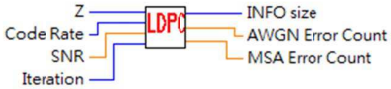

Figure 23: Complete encoding and decoding SubVI pinout satisfying IEEE 802.11n LDPC specifications

soft information (LLR) are the input. At the right side, the summed LLR soft information is the output (STEP4).

$$(x_i) = L(x_i) + \sum\nolimits_{m \in M(n) \backslash m} \mathbf{\Lambda}_{m \to n}(x_i) \tag{6}$$

Finally, the SubVIs in each step are connected and combined with the loop judgment of iteration numbers and hard decisions to complete the MSA decoding program that meets the IEEE 802.11n approximate lower triangular LDPC specifications for WLAN (Figure 20). Subsequently, the completed program is packaged as SubVI (Figure 21).

4.5. Complete Approximate Lower Triangular LDPC Decoding Structure

Incorporated with the decoder proposed by [8], the proposed MSA decoding method completes 12 check matrix decoders that meet IEEE 802.11n LDPC specifications for WLAN. An AWGN channel is added in the middle of the encoder and decoders, as shown in the complete encoding and decoding program architecture of Figure 22.

Figure 23 presents the packaged SubVI pinout. At the left side, subblock (Z), code rates, signal-to-noise ratio (SNR), and iteration are the input. At the right side, the information bit number (INFO size), number of error bits before decoding (AWGN Error Count), and number of error bits after MSA decoding (MSA Error Count) are the output.

5. ERROR RATE PERFORMANCE ANALYSIS OF IEEE 802.11N APPROXIMATE LOWER TRIANGULAR LDPC SPECIFICATIONS

The LabVIEW program was used to simulate the AWGN channel environment, with Eb/No from 0 to 10 dB. For every 1 dB interval, 81 Mbit random number data were generated. The corresponding relationship of Eb/No and σ is listed in Table 2. Encoding was completed by combining subblock sizes (27, 54, and 81 bit) and code rates (1/2, 2/3, and 5/6) and using BPSK modulation. The MSA single iteration was used to perform a decoding calculation.

Table 2: Comparison table of Eb/No and σ.

Eb/No, dB	0	1	2	3	4	5	6	7	8	9	10
σ	0.707	0.630	0.562	0.501	0.446	0.398	0.354	0.316	0.282	0.251	0.224

The result was mutually validated according to [12] the error rate curves of an irregular check matrix with $Z = 27$ and $Z = 81$ bit in the same code rate of 5/6. The validation results showed that the error rate curves were nearly identical (Figure 24). Thus, the value of Z did not influence the error rate. When the code rate is identical, the error rates yield nearly identical results. However, in this study, only AWGN channels were simulated. The Z value affected the length of codewords. If increasingly complex channel environments can be provided, such as the Rayleigh fading or burst error channels, discourse with improved integrity can be attained.

In a comparison of three error rate curves calculated at the receiver end after decoding and the error rate curve simply modulated through BPSK without adding error correcting codes (Figure 25), the results revealed that when the BER was 10^{-5}, Eb/No required 9.6 dB in the AWGN environment. If approximate lower triangular LDPC codes were added to the curves, the Eb/No calculated using MSA decoding, when $Z = 27$ bit, code rate = 5/6, was reduced to 6.35 dB. When $Z = 54$ bit and code rate = 2/3, Eb/No was 6.0 dB. When $Z = 81$ bit, code rate = 1/2, Eb/No was 5.7 dB. The coding gain increased by a range from 3.25 to 3.9 dB, compared with that of the AWGN channels. When the Eb/No washigher than 7 dB, the BER approximated 0.

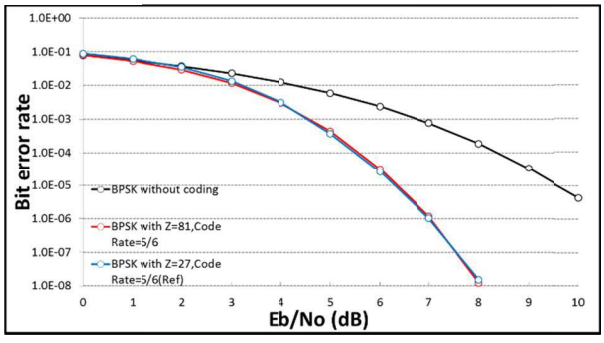

Figure 24: Error rate curve comparison of subblock size = 81 bit combined with code rate = 5/6 with the combination of subblock size = 27 bit and code rate = 5/6 in the literature.

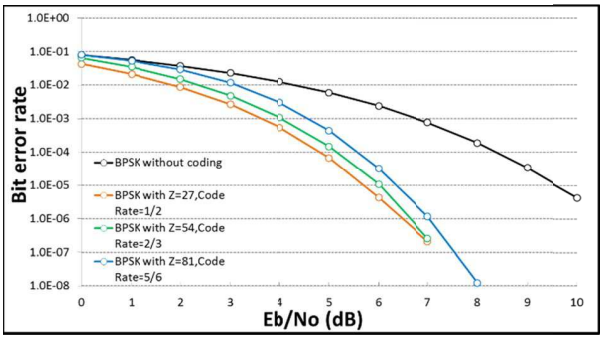

Figure 25: A comparison of error rate curve without coding and LDPC coded and decoded error rate curves generated using LabView simulation of BPSK modulation through the AWGN channel at varying subblock sizes (27, 54, 81 bit) and code rates (1/2, 2/3, and 5/6).

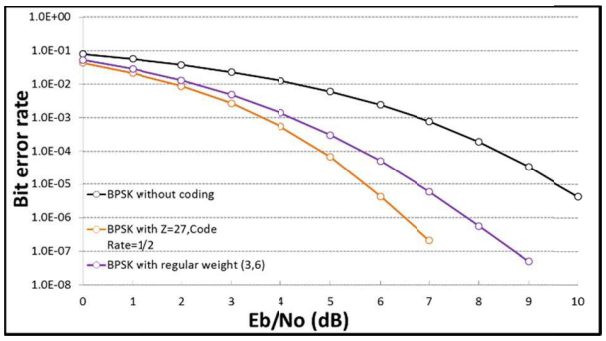

Figure 26: Comparison of error rate curves with subblock size = 27 bit, combined with code rate = 1/2 and with regular weight (3, 6) fromthe literature.

Finally, the results of this study were compared with the BER performance of regular weight (3, 6) from [12]. The code rate was identically 1/2. When the BER was 10^{-6}, the Eb/No that the regular weight (3, 6) required was 7.75 dB, and Eb/No = 6.5 dB when the irregular check matrix was $Z = 27$ bit. The coding gain was 1.25 dB.

6. CONCLUSION

This study applied the irregular check matrix in IEEE 802.11n specifications and adopted the LabVIEW to program and complete diverse encoders. In addition, MSA decoding methods were used to generate various decoders. The simulation adopted BPSK modulation to pass through the AWGN channel and compared the BER curves with those of previous studies. The results indicated that the subblock size in the AWGN channel environment did not affect error rates. However, different code rates yielded 3.25 to 3.9 dB coding gains when BER was 10^{-5}. The comparison results of the error correcting performances of the irregular code check matrix and weight (3, 6) regular code check matrix in 802.11n specifications showed that when the code rate was identically 1/2 and the BER was 10^{-6}, a 1.25 dB coding gain difference existed, thus verifying that the irregular check matrix outperformed the regular check matrix.

Future studies should use the sum-of product algorithm or forward-backward algorithm to complete decoding [6, 13]. Moreover, Rayleigh fading or Ricean fading can be added in the channels to further determine the variations of error rates in multiple situations.

REFERENCES

1. Shannon, C. E., "A mathematical theory of communication," *Bell Syst. Tech. J.*, 379–423 (Part 1); 623–656 (Part 2), Jul. 1948.
2. Berrou, C., A. Glavieux, and P. Thitimajshima,, "Near Shannon limit error-correcting coding and decoding: Trubo-codes," *IEEE International Conference on Communications, ICC'93*, Vol. 2, 1064–1070, Geneva, May 1993.
3. Sklar, B., *Digital Communications Fundamentals and Applications* 2nd Edition, Prentice Hall, 2001.
4. Gallager, R. G., "Low-density parity-check codes," *IRE Trans. Inform. Theroy*, 21–28, Jan. 1962.
5. Chen, J. and P. C. Marc, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans Commun.*, Vol. 50, No. 3, 406–414, Mar. 2002.
6. Kschischang, F. R., B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, Vol. 47, 498–519, Feb. 2001.
7. IEEE P802.11n^{TM}/D1.04, Draft Amendment to STANDARD for Information Technology-Telecommunications and Information Exchange, 2006.
8. Chen, Y. H., J. H. Hsiao, and Z. Y. Siao, "Wi-Fi LDPC encoder with approximate lower triangular diverse implementation and verification," *Multi-Conference on Systems, Signals & Devices (SSD)*, 1–6, 2014.
9. Richardson, T. J. and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IRE Trans. Inform. Theroy*, Vol. 47, No. 2, 638–656, 2001.
10. Michael Tanner, R., "A recursive approach to low complexity codes," *IRE Trans. Inform. Theroy*, Vol. 27, No. 5, 533–547, Sep. 1981.

11. Chen, Y.-H., C.-L. Chu, and J.-S. He, "FPGA implementation and verification of LDPC minimum sum algorithm decoder with weight (3, 6) regular parity check matrix," *ICEMI' 2013*, 682–686, Aug. 2013.
12. He, J.-S., "Implementation of LDPC encoder and decoder on SDR wireless communication system," Thesis-of-Master-Degree, O.I.T Institute of Information and Communication Engineer, Jul. 2013.
13. Bahl, L. R., J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, Vol. 20, 284–287, Mar. 1974.