# Structured Puncturing for Rate-Compatible B-LDPC Codes with Dual-Diagonal Parity Structure

Hyo Yol Park, *Student Member, IEEE,* Kwang Soon Kim, *Senior Member, IEEE,* Dong Ho Kim, *Member, IEEE*, and Keum Chan Whang

***Abstract*—In this paper, we propose a generalized formula for generating puncturing patterns for block-type low-density parity-check (B-LDPC) codes with dual-diagonal parity structure. The proposed formula distributes punctured bits uniformly in the zigzag edge connections, as well as maximizes the minimum recovery speed and the reliability of each punctured node. Also, the proposed puncturing can be applied to any B-LDPC code with dual-diagonal parity structure and can provide efficient bit-wise puncturing patterns even when the number of puncturing bits is not equal to an integer multiple of the block size. Simulation results show that the proposed punctured B-LDPC codes are better than existing punctured B-LDPC codes and even dedicated B-LDPC codes used in commercial standards.**

***Index Terms*—Block-type LDPC code, rate-compatible code, hybrid ARQ.**

## I. INTRODUCTION

LOW-density parity-check (LDPC) codes have emerged as very powerful channel codes due to their better performance and lower decoding complexity compared to turbo codes. However, the large and sparse parity-check matrix of an LDPC code results in serious drawbacks such as high encoding complexity and large memory requirement. To overcome these problems, quasi-cyclic LDPC codes [1] with dual-diagonal structure, called block-type LDPC (B-LDPC) code, have been proposed [2][3]. These allow us an efficient method of encoding in linear time with complexity proportional to the codeword length [4]. In addition, such B-LDPC codes can support various codeword lengths while reducing the memory requirements and providing performance comparable to solid randomly constructed LDPC codes. Due to such advantages, IEEE 802.16e adopted them as an optional forward error correction (FEC) code scheme [3]. Even so, rate incompatibility remains a major drawback compared to turbo codes. Thus, several parity-check matrices are required to support various code rates, and a hybrid automatic repeat and request (H-ARQ) using incremental redundancy (IR) cannot be supported in IEEE 802.16e. Recently, several puncturing techniques for randomly constructed LDPC codes have been proposed, such as those in [5]-[8]. Although they show relatively good error performance, the puncturing patterns must be obtained through a non-deterministic and non-real time algorithm. Thus, a large amount of memory is required to prepare several bit-wise puncturing patterns at both a transmitter and a receiver, especially when the number of possible code rates and codeword lengths is not small. In [9], some simple block-wise puncturing patterns for B-LDPC codes with dual-diagonal parity structure were proposed based on maximizing the number of surviving check nodes of each punctured bit. However, only a few block-wise puncturing patterns for given B-LDPC codes were offered without any generalization. In addition, those are useless when the number of puncturings is not equal to an integer multiple of the block size.

In this paper, we propose a deterministic and structured bit-wise puncturing method for B-LPDC codes with dual-diagonal parity structure. The proposed puncturing can be applied to any B-LDPC codes with dual-diagonal parity structure, thus allowing any amount of bit-wise puncturing with comparable performance to dedicated B-LDPC codes. In addition, owing to the block-type dual-diagonal parity structure, structured puncturing patterns can be obtained by using simple equations. Thus, we do not need to buffer several complex puncturing patterns at either a transmitter or a receiver, but rather generate the puncturing patterns in real time using the proposed formula. Here, the number of $k$-step recoverable ($k$-SR) nodes [6]-[9] at the lowest value of $k$ is maximized using the proposed block grouping, which selects the location of an unpunctured bit as close to a punctured bit as possible. Here, a $k$-SR node denotes a punctured variable node whose log likelihood ratio (LLR) value begins to be updated from its surviving check nodes at the $k$th iteration. In addition, the punctured bits are distributed as uniformly as possible through all connections (paths) composed of only parity variable nodes in the Tanner graph associated with the parity check matrix by using both the proposed uniform sequence and the proposed permutation. Since the punctured bits are distributed uniformly, the performance degradation due to the puncturing is also distributed evenly. Thus, the proposed puncturing is expected to enhance the overall decoding performance, especially when the amount of puncturing becomes large. This will be justified from the simulation results.

## II. B-LDPC CODES WITH DUAL-DIAGONAL PARITY STRUCTURE

An LDPC code is defined by an $m$ by $n$ parity check matrix, where $n$ is the length of the codeword and $m$ is the number of parity check bits in the codeword. An $m$ by $n$ parity check

Manuscript received April 17, 2007; revised August 14, 2007; accepted October 2, 2007. The associate editor coordinating the review of this paper and approving it for publication was X. Wang.

H. Y. Park, K. S. Kim (corresponding author), and K. C. Whang are with the Department of Electrical and Electronic Engineering, Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea (e-mail: ks.kim@yonsei.ac.kr).

D. H. Kim is with the Department of Media Engineering, Seoul National University of Technology, 172 Gongreung 2-dong, Nowon-gu, Seoul 139-743, Korea.

Digital Object Identifier 10.1109/T-WC.2008.070409

matrix $\mathbf{H}$ of a B-LDPC code can be defined using the sub-matrices:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^{a_{1,1}} & \mathbf{P}^{a_{1,2}} & \dots & \mathbf{P}^{a_{1,n_b}} \\ \mathbf{P}^{a_{2,1}} & \mathbf{P}^{a_{2,2}} & \dots & \mathbf{P}^{a_{2,n_b}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{a_{m_b,1}} & \mathbf{P}^{a_{m_b,1}} & \dots & \mathbf{P}^{a_{m_b,n_b}} \end{bmatrix}, \quad (1)$$

where $\mathbf{P}^{a_{i,j}}$ denotes a $z \times z$ sub-matrix (denoted as a block in the sequel), which is either an $a_{i,j}$ times circularly-shifted version of a $z \times z$ identity matrix or a zero matrix. Here, $\mathbf{H}$ can be partitioned as $\mathbf{H} = [(\mathbf{H}_s)_{zm_b \times zk_b} | (\mathbf{H}_p)_{zm_b \times zm_b}]$, where $\mathbf{H}_s$ ($\mathbf{H}_p$) corresponds to the systematic (parity) bits, $n = zn_b$, $m = zm_b$, and $k_b = n_b - m_b$. In addition, the dual-diagonal parity structure, $\mathbf{H}_p$, is further partitioned as

$$\mathbf{H}_p = [\mathbf{H}_o | \mathbf{H}_d] = \left[\begin{array}{c|cccccc} \mathbf{P}^{b_1} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^{b_2} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{P}^{b_3} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{P}^{p} & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{P}^{b_{m_b-1}} & \mathbf{I} \\ \mathbf{P}^{q} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{P}^{b_{m_b}} \end{array}\right], \quad (2)$$

where the matrix $\mathbf{H}_o$ has three non-zero blocks and $\mathbf{H}_d$ is the $zm_b \times z(m_b - 1)$ matrix with dual-diagonal blocks. Here, the block at the $i$th row and the $j$th column of $\mathbf{H}_d$ is equal to a $z \times z$ identity matrix when $i = j$, is equal to $\mathbf{P}^{b_i}$ when $i = j + 1$, and is a zero matrix elsewhere. Note that the row block location of $\mathbf{P}^p$ is conventionally chosen at about half of $m$ [3], even though it can be selected arbitrarily [2]. This parity structure allows us a linear time efficient encoding, especially when the following equation is satisfied [2].

$$q = \left(\sum_{i=1}^{m_b} b_i\right) \bmod z \quad \text{and} \quad p = \left(-\sum_{i=l+1}^{m_b} b_i\right) \bmod z. \quad (3)$$

Here, $\mathbf{P}^p$ is assumed to be located at the $l$th row block of $\mathbf{H}_o$. Note that the LDPC codes used in IEEE 802.16e [3] are special cases of the B-LDPC code with $b_2 = b_3 = \dots = b_{m_b} = 0$. In this case, $b_1 = q$ and $p = 0$ from (3). Here, $q$ is selected to be prime with the block size $z$. Then, all the parity blocks themselves make $(z+2)$ block cycles, as defined in [2]: one $2l$-block cycle, one $2(m_b - l + 1)$-block cycle, and $z$ $2m_b$-block cycles. Note that each kind of the block cycles makes a cycle with a length of $2zl$, a cycle with a length of $2z(m_b - l + 1)$ and $z$ parallel cycles with a length of $2m_b$ by proposition 3 in [2], which will be denoted as connections 1, 2, and 3 in this paper, respectively.

## III. The Proposed Puncturing Method

Because all parity bits (all candidate bits for puncturing) are connected through the zigzag edge connections, grouping $k$-SR nodes and controlling the reliability of each $k$-SR node can be conveniently achieved in the Tanner graph associated with $\mathbf{H}_p$ only. Also, owing to the dual-diagonal parity block structure common in any B-LDPC code, we can obtain a general puncturing formula.

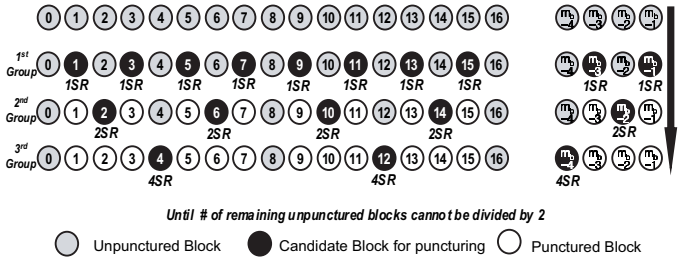


Fig. 1. The proposed block grouping.

Before beginning the puncturing process, we first group the candidate puncturing blocks to maximize the number of $k$-SR nodes at the lowest value of $k$. Here, only the blocks of degree-2 nodes are selected to allocate the surviving check nodes evenly to each punctured node. Initially, the $(2n+1)$th blocks $(n = 0, 1, ..., m_b/2 - 1)$ are selected as the first candidate blocks. By doing so, $m_b z/2$ 1-SR nodes can be obtained (maximized) and the number of surviving check nodes of each 1-SR node becomes 2 (that is, each 1-SR node can receive the LLR messages from all its neighboring unpunctured bits). When all bits in the first selected blocks are punctured, $2(2n+1)$th blocks $(n = 0, 1, ..., m_b/4 - 1)$ are then selected as the second candidate blocks. By doing so, $m_b z/4$ 2-SR nodes can be obtained (maximized) and the number of surviving check nodes of each 2-SR node becomes 2 (that is, each 2-SR node can receive the LLR messages from all its neighboring 1-SR nodes). In this way, at the $t$th step, $2^{t-1}$-SR nodes $(t = 1, 2, ...)$ in the $2^{t-1}(2n+1)$th blocks $(n = 0, 1, ..., m_b/2^t - 1)$ are selected as the $t$th candidate blocks to maximize the number of $2^{t-1}$-SR nodes until the number of remaining unpunctured blocks becomes an odd number. If there remain an odd number of unpunctured blocks, all those with degree-2 nodes are selected as the last candidate blocks. In Fig. 1, the proposed block grouping is shown.

Using the proposed grouping, the proposed puncturing algorithm can be summarized by the following procedure. At first, the group index, $t$, and the current number of punctured bits, $i$, are initialized to 1 and 0, respectively; the puncturing process continues until $i$ is equal to $N_p$. If $m_b/2^t$ is an integer and there are remaining candidate blocks in the current group $(i < (2^t - 1)m_b z/2^t)$, the location of the $i$th puncturing bit is determined by $L_1(i)$. When all the candidate blocks in the current group are punctured, the group index, $t$, is increased by 1. If $m_b/2^t$ is not an integer (i.e., there are an odd number of unpunctured blocks), the location of the $i$th puncturing bit is determined by $L_2(i)$. After puncturing one bit, $i$ is increased by 1 and the process continues. In Fig. 2, the proposed puncturing pattern generation algorithm is shown.

Determining the $i$th puncturing location, either $L_1(i)$ or $L_2(i)$, can be divided into the two steps: the block selection and the bit selection in the chosen block. Here, we consider three strategies. All strategies choose only one bit at a given time from a selected block. The first method is to choose the leftmost unpunctured candidate block and the leftmost unpunctured bit in the selected block at each time (denoted as the 'grouping only' method in the sequel). Note that this is a simple and straightforward way but cannot guarantee uniform

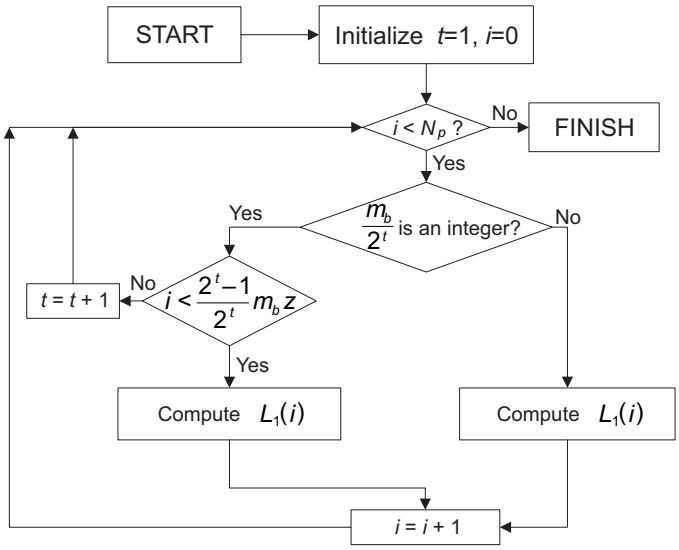


Fig. 2. The flow chart of the proposed bit-wise puncturing pattern generation.

distribution of punctured bits. In this case, the location of the punctured bits is determined as

$$L_1^{PG}(i) = 2^{t-1}(2\text{mod}(\lfloor i/z \rfloor, m_b/2^t) + 1)z + \text{mod}(i,z), \quad (4)$$
$$L_2^{PG}(i) = 2^{t-1}(2\text{mod}(I^t(i), J^t) + 2)z + \text{mod}(i,z), \quad (5)$$

where $\text{mod}(x,y) \triangleq x \bmod y$, $I^t(i) = \lfloor i/z - (m_b - J^t - 1) \rfloor$, $J^t = m_b/2^{t-1} - 1$, and $\lfloor c \rfloor$ denotes the greatest integer not exceeding $c$. Here, the superscript *PG* denotes that only the proposed block grouping is used. In (4) and (5), the first term determines the block location and the second term determines the bit location within the block.

Although the 'grouping only' method can maximize the minimum recovery speed due to the proposed grouping, there exist bit nodes with much lower reliability than others because a uniform distribution in the connections is not achieved; this degrades the overall decoding performance. In order to distribute the punctured bits more uniformly, we propose a uniform selection sequence[1] among the $n$ samples, $\mathbf{u_n}$, which is defined as

$$\mathbf{u_1} = \{0\}, \quad (6)$$
$$\mathbf{u_{2k}} = \{u_k(0), u_k(0)+k, u_k(1), u_k(1)+k, ..., u_k(k-1), u_k(k-1)+k\}, \quad (7)$$
$$\mathbf{u_{2k+1}} = \{k, u_k(0), u_k(0)+k+1, u_k(1), u_k(1)+k+1, ..., u_k(k-1), u_k(k-1)+k+1\}, \quad (8)$$

where $k$ is a natural number and $u_n(i)$, $i = 0, 1, ..., n-1$, denotes the $i$th element in the sequence $\mathbf{u_n}$. Note that the sequence $\mathbf{u_n}$ can be obtained in a recursive manner so that it can be easily computed in a real-time processor. Also, note that any partial sequence of $\mathbf{u_n}$ with the form $\{u_n(jn/2^k + 0), u_n(jn/2^k + 1), ..., u_n((j+1)n/2^k - 1)\}$, $k = 1, 2, ..., \log_2 n$ and $j = 0, 1, ..., 2^k - 1$, is also a uniform selection sequence. Therefore, the punctured bits can be distributed quite uniformly within a graph using the uniform sequence as the block selection sequence. Let us define $K_1^t(i)$ and $K_2^t(i)$ as

$$K_1^t(i) = 2^{t-1}(2u_{m_b/2^t}(\text{mod}(\lfloor i/z \rfloor, m_b/2^t)) + 1), \quad (9)$$
$$K_2^t(i) = 2^{t-1}(2u_{J^t}(\text{mod}(I^t(i), J^t)) + 2). \quad (10)$$

Then, (4) and (5) are modified using the proposed block selection sequence as follows.

$$L_1^{PB}(i) = K_1^t(i)z + \text{mod}(i,z), \quad (11)$$
$$L_2^{PB}(i) = K_2^t(i)z + \text{mod}(i,z). \quad (12)$$

Here, the superscript *PB* denotes that the proposed block grouping and the proposed block selection sequence are used. Note that the locations of the punctured bits are roughly uniform within the connections due to the proposed block selection sequence. However, because every non-zero block, $P^{a(i,j)}$ for $a(i,j) \neq 0$, permutes its connections, a careful bit selection process is required to choose puncturing bits uniformly within the overall permuted connections when the number of puncturings is not an integer multiple of $z$.

To distribute the punctured bits most uniformly in the overall permuted connections, the proposed bit selection process using both the uniform sequence and the permutation can be performed as follows: the bits in a block are punctured in the order determined by $u'_z(i)$ and $u''_z(i)$, which is defined as

$$u'_z(i) = \text{mod}(\text{mod}(p + \sum_{k=1}^{l} b_k, z)u_z(i), z), \quad (13)$$
$$u''_z(i) = \text{mod}(\text{mod}(p - q + \sum_{k=l+1}^{m_b} b_k, z)u_z(i), z), \quad (14)$$

for $i = 0, 1, ..., z$. Then, the location of the $i$th punctured bit is determined by

$$L_1^{Prop}(i) = K_1^t(i)z + c_z(mod(i,z)), \quad (15)$$
$$L_2^{Prop}(i) = K_2^t(i)z + d_z(mod(i,z)), \quad (16)$$
$$c_z(mod(i,z)) = \begin{cases} u'_z(mod(i,z)), & \text{if } K_1^t(i) \leq l, \\ u''_z(mod(i,z)), & \text{otherwise,} \end{cases} \quad (17)$$
$$d_z(mod(i,z)) = \begin{cases} u'_z(mod(i,z)), & \text{if } K_2^t(i) \leq l, \\ u''_z(mod(i,z)), & \text{otherwise.} \end{cases} \quad (18)$$

Note that in (15) and (16), we can distribute the punctured bits uniformly in the overall permuted zigzag connections using the proposed block grouping, block selection, and bit selection processes. Also, note that when $b_i = 0$ for $2 \leq i \leq m_b$ as in [3], (13) and (14) can be simplified as

$$u'_z(i) = mod(b_1 u_z(i), z), \quad (19)$$
$$u''_z(i) = mod((z-q)u_z(i), z), \quad (20)$$

for $i = 0, 1, ..., z$. Simple examples of the 'grouping only' method and the proposed method are shown in Fig. 3 when $z = 4$, $b_1 = q = 1$, $p = b_i = 0$ for $2 \leq i \leq 8$ and the total number of bits for puncturing, $N_p$, is 6. Here, blocks 1, 3, 5, and 7 are selected as the first candidate blocks. Then, the six bit nodes in the selected blocks are punctured in the order determined by each strategy. As shown in Fig. 3, the 'grouping only' method first selects the leftmost unpunctured block among all first candidate blocks and the leftmost unpunctured bit of the block. After puncturing six bits, the distribution of

[1]In the special case where $n$ is a power of 2, $u_n$ is the same to the bit reverse order commonly used in fast Fourier transform (FFT).

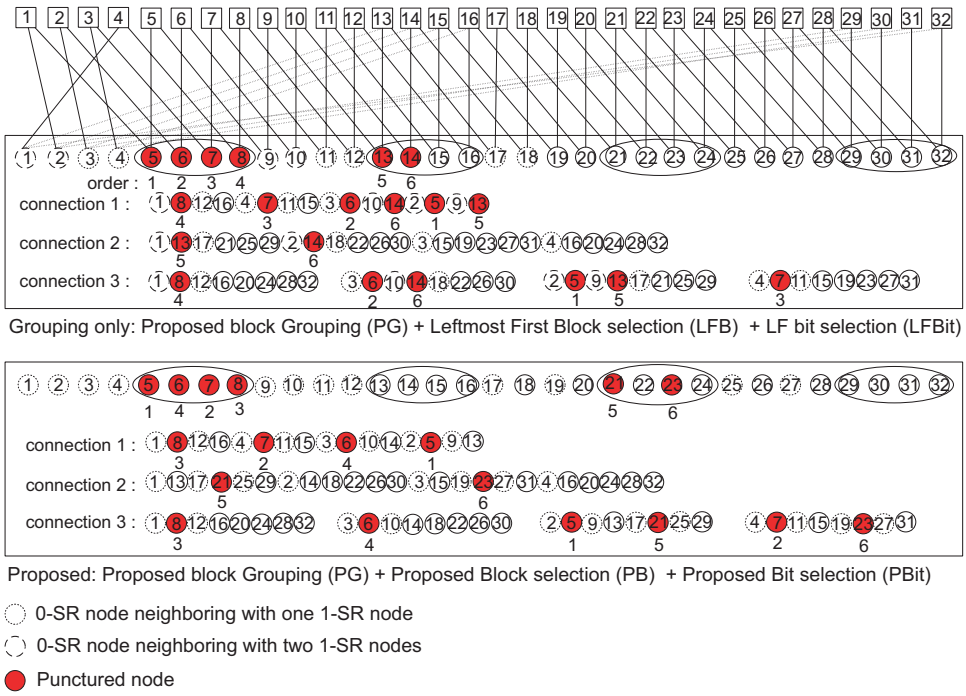


Fig. 3. Comparison of the two bit-wise puncturing methods (the 'grouping only' method and the proposed method).

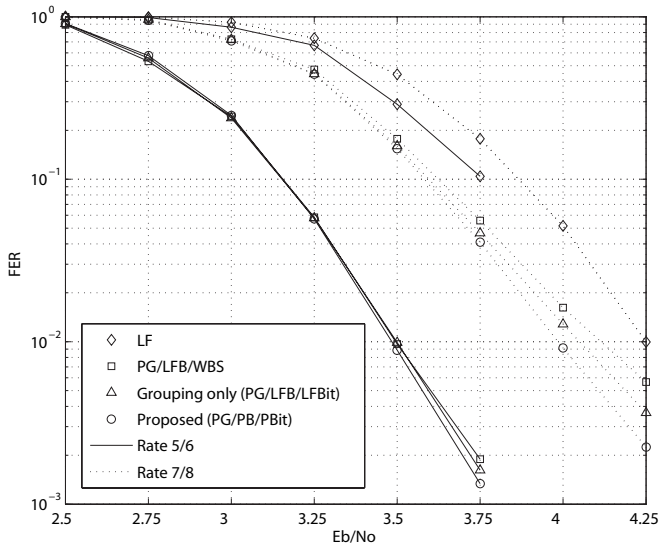


Fig. 4. Performance comparison of the bit-wise puncturing methods.

the punctured bits is not uniform in connections 1, 2, and 3, as shown in Fig. 3. However, the proposed method chooses a block using the proposed block selection and bit selection process. Note that the first four bits are determined using the sequence $\mathbf{u}'_{\mathbf{z}}$ and last two bits are determined using the sequence $\mathbf{u}''_{\mathbf{z}}$. From Fig. 3, it is seen that the six punctured bits are distributed uniformly in all three connections.

## IV. Simulation Results

In Fig. 4, the performances of the four puncturing bit selection methods are compared. Here, the B-LDPC code with a rate of 1/2, $n$=2304, and $z$=96 in [3] is used as the mother code and the B-LDPC code with a rate of 5/6 (7/8) is made by puncturing 922 (987) parity bits from the mother code. Note that the number of punctured bits is not divided by $z$ (=96) and the method in [9] cannot be applied. Here 'LF' denotes the method of selecting the leftmost bit first for puncturing without using the proposed block grouping. Thus, the performance is severely degraded because there are many $k$-SR nodes with relatively large values of $k$. 'PG/LFB/WBS' denotes the process of using the proposed grouping and the leftmost block selection as in 'grouping only' method. The bit selection method intentionally chooses a bit right after the permutation (the worst bit selection). Looking at the performance difference between the 'LF' and 'PG/LFB/WBS' cases, it can be seen that the proposed block grouping significantly improves the performance of a punctured B-LDPC code. Although some similar block grouping patterns were suggested in [9], the proposed block grouping is a generalized method applicable to any B-LDPC code with dual-diagonal parity structure. Among the four bit-wise puncturing methods, the proposed puncturing shows the best FER performance. Moreover, the performance difference between the proposed method and 'PG/LFB/WBS' case shows that the proposed block selection and bit selection process contributes a non-negligible performance gain. Finally, it is seen that the 'grouping only' method shows a performance somewhere between that of the proposed method and that of 'PG/LFB/WBS'. Thus, we can also use it without a severe performance degradation when the permutation information of the mother code is not given.

In Fig. 5, the proposed punctured B-LDPC codes with rates of 2/3 ($n$=1728) and 3/4 ($n$=1536) from the same mother code are compared with the dedicated B-LDPC codes in [3] and the turbo codes in [10] with the same rates. From the results, it is clear that the proposed punctured B-LDPC codes are comparable to the dedicated B-LDPC codes and much better than the turbo codes from a moderate to high SNR region.

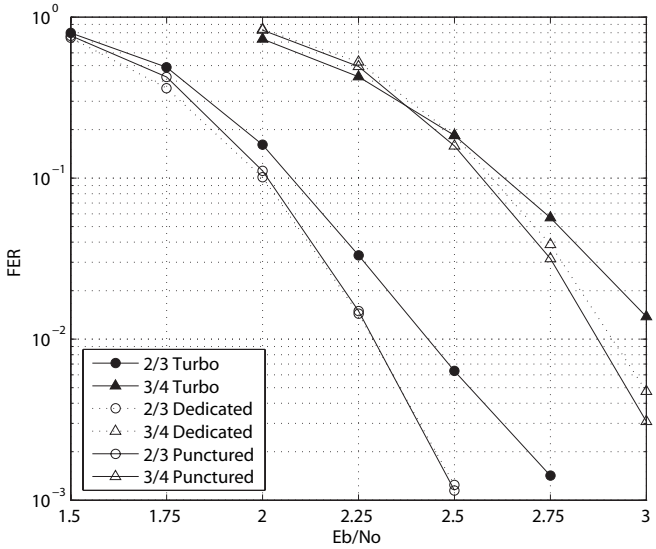


Fig. 5. Performance of the proposed punctured LDPC codes from 802.16e LDPC code with rate of 1/2.

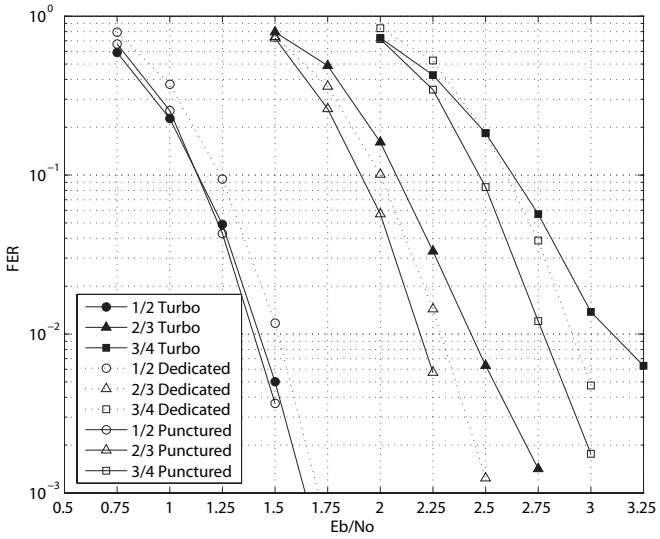


Fig. 6. Performance of the proposed punctured LDPC codes from a mother LDPC code with a rate of 1/3.

In Fig. 6, the proposed punctured B-LDPC codes with rates of 1/2 ($n$=2304), 2/3 ($n$=1728), and 3/4 ($n$=1536) from another dual-diagonal parity structured mother B-LDPC code constructed from the design criterion in [2] with a rate of 1/3, $n$=3456 ($m_b$=64, $n_b$=96, $z$=36), are compared with the dedicated B-LDPC codes in [3] and the turbo codes in [10] with the same rates. From the results, the proposed punctured B-LDPC codes show better performance than both the dedicated B-LDPC codes and the turbo codes. Thus, by using the proposed puncturing, we can obtain rate-compatible B-LDPC codes with a better performance than those in commercial standards from a good single mother B-LDPC code.

## V. Conclusion

In this paper, a deterministic and structured method was proposed to generate puncturing patterns of B-LDPC codes with dual-diagonal parity structure. This method does not require memory for buffering puncturing patterns at either a transmitter or a receiver, and can provide an easy calculation of the puncturing pattern by using the zigzag structure of the parity check matrix. From the simulation results, it was shown that the proposed B-LDPC codes perform better than the dedicated B-LDPC codes or turbo codes used in commercial standards. Thus, we can substitute the mandatory turbo code or the optional B-LDPC codes of several rates in [3] with the proposed puncturing formula and a single good mother B-LDPC code to obtain both complexity reduction and performance improvement. In addition, any kind of H-ARQ technique can be applied by using the proposed punctured B-LDPC codes.

## VI. Acknowledgement

This work was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2006-(C1090-0603-0011)).

## References

[1] R. M. Tanner, D. Sridhara, A. Sridhara, T. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 12, pp. 2966-2984, Dec. 2004.

[2] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic LDPC codes for fast encoding," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 2894–2901, Aug. 2005.

[3] IEEE P802.16e, IEEE Standard for Local and Metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access System, Feb. 2006.

[4] T. J. Richardson, and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638-656. Feb. 2001.

[5] J. Ha, J. Kim, and S. W. McLaughlin, "Optimal puncturing distribution for rate compatible low density parity check code," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2824-2836, Nov. 2004.

[6] J. Ha, J. Kim, and S. W. McLaughlin, "Puncturing for finite length low-density parity-check codes," in *Proc. Inter. Symp. Inform. Theory (ISIT)*, pp. 151, Chicago, USA, June 2004.

[7] J. Ha, J. Kim, D. Klinc, and S. W. Mclaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 728-738, Feb. 2006.

[8] H. Y. Park, J. W. Kang, K. S. Kim, and K. C. Whang, "Efficient puncturing method for rate compatible low-density parity-check codes," *IEEE Trans. Wireless Commun.*, vol. 6, no. 7, pp. 3914-3919. Aug. 2007.

[9] E. Choi, S. Suh, and J. Kim, "Rate-compatible puncturing for low-density parity-check codes with dual-diagonal parity structure," in *Proc. IEEE Symp. Person., Indoor Mobile Radio Commun. (PIMRC)*, pp. 2642-2646, Berlin, Germany, Sept. 2005.

[10] 3GPP TR 25.848: Physical layer aspects of UTRA High Speed Downlink Packet Access, v4.0.0, Mar. 2001.