Split LDPC Codes for Hybrid ARQ

Hyeong-Gun Joo*, Song-Nam Hong** *Regular Members,*
Dong-Joon Shin***° *Lifelong Member*

ABSTRACT

In this paper, we propose a new rate-control scheme, called splitting, to construct low-rate codes from high-rate codes by splitting rows of the parity-check matrices of LDPC codes, which can construct rate-compatible LDPC codes having good initial transmission performance. Good low-rate codes can be constructed by making the number of distinct check node degrees as small as possible after splitting. The proposed scheme achieves good cycle property, low decoding complexity, and fast convergence speed, especially compared to the puncturing. Especially, rate-compatible repeat accumulate-type LDPC (RA-Type LDPC) code is constructed using splitting, which covers the code rates from 1/3 to 4/5. Through simulation it is shown that this code outperforms other rate-compatible RA-Type LDPC codes for all rates and can be decoded conveniently and efficiently.

Key Words : RA-Type LDPC Codes, Rate-compatibility, Splitting, HARQ.

I. Introduction

Low density parity check (LDPC) codes provide better performance than turbo codes yet with lower decoding complexity. However, the encoding complexity of LDPC codes is quadratic in the block length, which results in slow encoding. Therefore, efficient encoding algorithm has been studied [1] and various fast-encodable LDPC codes with dual-diagonal parity structure were proposed [2][3], which are called repeat accumulate-type (RA-Type) LDPC codes. The main difference between RA-Type LDPC codes and general LDPCcodes and general irregular LDPC codes is the control of degree-2 parity node connectivity. The dual-diagonal parity structure allows many degree-2 parity nodes while keeping the stability and enables the linear-time encoding.

Rate compatible (RC) codes are a family of nested codes where the codeword bits of the high-rate code are contained in the codeword of low-rate code. Therefore, they can be encoded and decoded using a single encoder/decoder pair. To construct RC LDPC codes, many schemes such as data puncturing (shortening) [4], puncturing [5][6], and extending [7][8] have been proposed. These schemes have several problems such as slow convergence speed, somewhat high decoding complexity, and performance degradation.

In this paper, we propose new rate-control scheme called splitting in order to solve the above problems. Splitting scheme comes from the observation that the quality of the initial transmission is an important factor to achieve high HARQ throughput. Since the mother code for RC-LDPC codes obtained by splitting is the highest-rate code in the desired rate coverage, the mother code can be designed to achieve good initial transmission performance. Also, since a high-degree check node is split into two low-degree check nodes by adding a new common parity node, splitting can generate good low-rate codes by making the check node degree distribution is in a concentrated form after

※ This work was supported by grant No.(R01-2005-000-11064-0) from the Basic Research Program of Korea Science & Engineering Foundation.
* Division of Electronics and Computer Engineering, Hanyang University (yangpablue@ccrl.hanyang..ac.kr),
** Telecommunication R&D Center, Samsung Electronics Co (sn7955.hong@samsung.com),
*** Division of Electronics and Computer Engineering, Hanyang University (djshin@hanyang.ac.kr), (°: 교신저자)
논문번호 : KICS2006-10-406, 접수일자 : 2006년 10월 9일, 최종논문접수일자 : 2007년 10월 4일

splitting. Compared with other rate-control schemes such as shortening, puncturing, and extending, the splitting not only shows faster decoding convergence speed but also has low decoding complexity

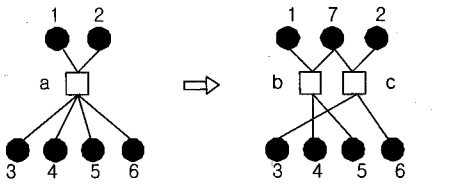
Ⅱ. New Rate-Control Scheme of LDPC Codes: Splitting

To solve the above problems, we propose new rate-control method, called splitting, which splits a degree-i check node into degree-j check node and degree-k check node, $j+k=i+2$, by adding a new degree-2 parity node. The proposed scheme achieves good cycle property, low decoding complexity, and fast convergence speed, especially compared with puncturing. First, we will explain the basic concept of splitting by using the following Theorem 1.

Theorem 1: When a set $A=\{B_1,B_2,...,B_i,C_1,C_2,...,C_j\}$ satisfies the even parity, if each of $B=\{B_1,B_2,...,B_i,B_{i+1}\}$ and $C=\{C_1,C_2,...,C_j,C_{j+1}\}$ also satisfies the even parity, then $B_{i+1}=C_{j+1}$. Note that, in this case, the elements of B and C satisfy the even parity and it implies that A also satisfies the even parity.

Proof: If $B_1\oplus\cdots\oplus B_i\oplus C_1\oplus\cdots\oplus C_j=0$, then we have $B_1\oplus\cdots\oplus B_i=C_1\oplus\cdots\oplus C_j$ where $\oplus$ denotes the modulo-2 addition. Under this condition, if $B_1\oplus\cdots\oplus B_i\oplus B_{i+1}=0$ and $C_1\oplus\cdots\oplus C_j\oplus C_{j+1}=0$, then clearly $B_{i+1}=C_{j+1}$.

Fig.1 shows an example of splitting where upper and lower black circle nodes denote the parity and information nodes, respectively, and square nodes denote the check nodes. Let A_i be the set of variable nodes connected to the check node i. Then, in Fig. 1(a), $A_a=\{v(1),v(2),v(3),v(4),v(5),v(6)\}$ satisfies the even parity where $v(j)$ represents the value of variable node j. In Fig. 1(b), each of $A_b=\{v(1),v(4),v(5),v(7)\}$ and $A_c=\{v(2),v(3),v(6),v(7)\}$ satisfies the even parity and by combining these two sets, it can be



(a) Tanner graph (b) Tanner graph obtained by splitting

Fig. 1 Example of splitting.

a →	1	0	1	0	1	1
b →	1	1	0	1	0	1
c →	0	1	1	1	1	0
d →	1	1	0	0	1	1

(a) Parity-check matrix of mother code.

	Split mother code part						Additional parity part			
a_1 →	1	0	0	0	1	0	1	0	0	0
a_2 →	0	0	1	0	0	1	1	0	0	0
b_1 →	1	0	0	1	0	0	0	1	0	0
b_2 →	0	1	0	0	0	1	0	1	0	0
c_1 →	0	1	0	1	0	0	0	0	1	0
c_2 →	0	0	1	0	1	0	0	0	1	0
d_1 →	1	0	0	0	1	0	0	0	0	1
d_2 →	0	1	0	0	0	1	0	0	0	1

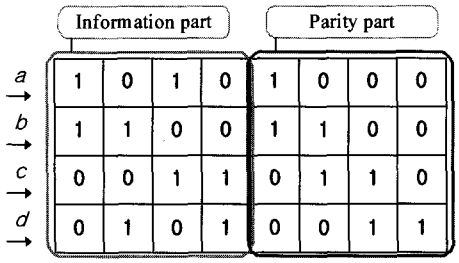
(b) Parity-check matrix after splitting each row.

Fig. 2 Splitting of general LDPC code.

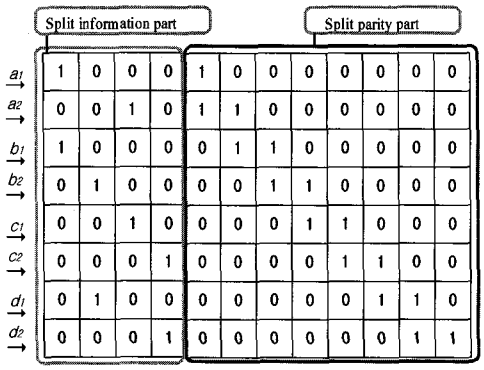
verified that A_a also satisfies the even parity. Therefore, we can split the check node a in Fig. 1(a) into two check nodes b and c in Fig. 1(b) by partitioning A_a into two subsets and adding a new common degree-2 parity bit 7 to them. In other words, a high-degree check node can be split into two lower-degree check nodes by adding a common degree-2 parity bit to generate low-rate code.

For RC-LDPC codes built using splitting, the initial transmission has good performance since good mother code of the highest code rate can be designed. To construct good low-rate codes using splitting, the right degree distribution must be in the concentrated form [9]. Contrary to the extending scheme, splitting does not introduce short cycles and can increase the girth.

Figs 2 and 3 show examples of splitting parity-check



(a) Parity-check matrix of mother code.



(b) Parity-check matrix after splitting each row.

Fig. 3 Splitting of RA-Type LDPC code.

matrices of general and RA-Type LDPC codes. Note that newly added degree-2 parity bits should be connected to the two split rows as explained in Theorem 1.

Fig. 2(b) shows the parity-check matrix after splitting each row of parity-check matrix of mother code in Fig. 2(a). The first row a in Fig. 2(a) is split into two rows a_1 and a_2 sharing a new parity bit in Fig. 2(b) and so on. Fig. 3 is for splitting RA-Type LDPC code and can be similarly explained as Fig. 2.

The parity-check structure obtained by splitting general irregular LDPC codes as in Fig. 2(b) may not provide good performance since newly generated degree-2 parity bits are partitioned into disconnected subsets. In general, connected parity (dual-diagonal) structure can guarantee good performance. Thus, newly generated degree-2 parity bits with disconnected subsets may not provide good performance. However, the parity-check structure in Fig. 3(b) provides good performance since newly added parity bits maintain the pre-existing dual-diagonal parity structure. Therefore, by splitting RA-Type LDPC codes, we can construct good low-rate RA-Type LDPC codes.

Ⅲ. Comparison of Various Rate -Compatible RA-Type LDPC Codes

In this section, we will compare the decoding convergence speed and the decoding complexity when splitting, puncturing, and extending & puncturing are applied to RA-Type LDPC codes. Note that extending & puncturing is the scheme to obtain high-rate codes by puncturing mother code and to obtain low-rate codes by extending mother code. Also, by constructing RC RA-Type LDPC codes using splitting, puncturing, and extending & puncturing, the performances are compared for various code rates.

3.1 Decoding convergence speed of various rate-control schemes

In this subsection, we compare the decoding convergence speed of three rate-control schemes, puncturing, splitting, and extending & puncturing. The decoding convergence speed of punctured LDPC code is slower than that of unpunctured LDPC code since the messages are slowly updated for the punctured nodes which are assigned 0 initial LLR values during the iterative decoding. Thus, when the number of iterations is not sufficient, punctured LDPC code shows worse performance than the unpunctured LDPC code even if both have the same threshold. Therefore, splitting gives the fastest decoding convergence speed and the extending & puncturing is the next since splitting does not make punctured nodes and extending & puncturing makes smaller number of punctured nodes than the puncturing. It can be verified through simulation in Fig. 5.

3.2 Decoding complexity of various rate-control schemes

We consider the decoding complexity per iteration of puncturing, splitting, and extending & puncturing. Punctured LDPC codes need more decoding

operations per iteration than the unpunctured LDPC codes of the same code rate since they are decoded on more complicated Tanner graphs which contain punctured nodes. Suppose that we use the following LLR-BP check node update rule [10].

$$L_{m\to n}(x_n)=\left(\prod_{n'\in N(m)\setminus n} sign(Z_{n'\to m}(x_{n'}))\right) \cdot f\left(\sum_{n'\in N(m)\setminus n} f(|Z_{n'\to m}(x_{n'})|)\right)$$

where $L_{m\to n}$ and $Z_{n\to m}$ represent the messages from the check node m to the variable node n and vice versa, $N(m)$ is the set of neighboring nodes of the node m, $sign(x)$ takes the sign of x, and $f(x)=\log((e^x+1)/(e^x-1))$. Also, the following LLR-BP variable node update rule [10] is used.

$$Z_{n\to m}(x_m)=m_0+\sum_{m'\in N(n)\setminus m} L_{m'\to n}(x_{m'})$$

where m_0 denotes the initial LLR value from the channel at the variable node m. Table 1 shows the computational complexity for variable node and check node message updates (for the detailed explanation, see [10]).

The following two approaches are considered to calculate the number of additions at the variable node message update.

① The addition is done on all incoming LLR messages and the initial LLR value at a variable node and each incoming message is subtracted once to derive the outgoing LLR message through that edge. This approach requires $2d_v$ additions.

② Each outgoing LLR message is computed by using the LLR-BP variable node update rule. This approach requires $d_v(d_v-1)$ additions.

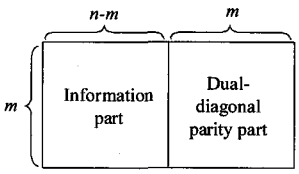
Note that these approaches can be applied to the additions for check node message update and ② is more efficient than ① only when $d_v=2$.

Theorem 2: Suppose that we design two RC RA-Type LDPC codes using splitting and puncturing such that, for the fair comparison, they have $m\times n$ parity-check matrices of the same degree distribution for the lowest code rate R. If the code of rate R is obtained by splitting the mother code of rate $R_m(>R)$, the number of additions per iteration for variable node message updates is less than that of punctured code of rate R_m by $4(n-m)(1/R-1/R_m)$ if ① is used (or $2(n-m)(1/R-1/R_m)$ if ② is used). Also, the number of additions (or special operations) per iteration for check node message updates is also reduced by $3(n-m)(1/R-1/R_m)$(or $4(n-m)(1/R-1/R_m)$).

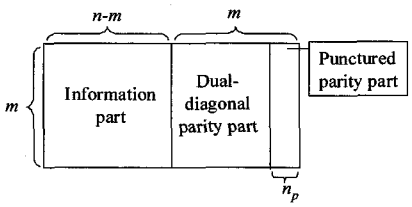
Table. 1 Computational complexity of message update at each of variable and check nodes during each iteration; d_v and d_c denote the degrees of variable and check nodes, respectively.

Node type / Operation type	Number of additions	Number of special operations $f(x)$
Variable node	$2d_v$ (or $d_v(d_v-1)$)	0
Check node	$2d_c-1$	$2d_c$

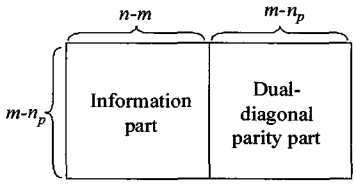
Proof: The RC RA-Type codes for splitting and puncturing have the same degree distribution for the lowest code rate as shown in Fig. 4(a), implying that the numbers of 1's in information parts of parity-check matrices of both codes of rate R are the same. The number n_P of punctured parity bits for the punctured code of rate R_m is $n_P=(n-m)(1/R-1/R_m)$ in Fig. 4(b) and all punctured (or unpunctured) parity nodes have degree 2. The split code of rate R_m does not contain punctured nodes as in Fig. 4(c). Since split and punctured codes of rate R_m have the same number of 1's in information parts of the corresponding parity-check matrices, by using splitting the number of additions per iteration for variable node message updates is reduced by $4n_P=4(n-m)(1/R-1/R_m)$ if ① is used (or $2n_P=2(n-m)(1/R-1/R_m)$ if ② is used) using Table 1. Since the number of total edges of check nodes is the same as that of variable nodes (except channel values), we can see that the total number of edges of check nodes in split code is less than that in punctured code by $2n_P=2(n-m)(1/R-1/R_m)$. Therefore, by using splitting, the number of check node additions (or special operations) per iteration is reduced by



(a) Parity-check matrix of the lowest rate R.



(b) Parity-check matrix of punctured code of rate R_m.



(c) Parity-check matrix of split code of rate R_m.

Fig. 4 Schematic parity-check matrices ($R_m > R$).

□

$3(n-m)(1/R-1/R_m)$ (or $4(n-m)(1/R-1/R_m)$) using Table 1.

3.3 Construction of RC B-LDPC codes for various rate-control schemes

In this subsection, the degree distributions of good mother block-type LDPC (B-LDPC) codes [3], a class of RA-Type LDPC codes, are obtained for splitting, puncturing, and extending & puncturing, and the target range of code rates is from 1/3 to 4/5. We design mother codes for RC B-LDPC codes using splitting and puncturing such that they have the same degree distribution for the lowest-rate codes for the fair comparison. The code parameters for extending & puncturing are determined such that, for the lowest code rate, the maximum variable node degree is limited by 16 which is also the maximum variable node degree for the splitting and puncturing, and the decoding complexity becomes similar to those for splitting and puncturing.

3.3.1 Code parameters for splitting:

The rate-4/5 mother code for splitting has the following degree distributions. The parameters λ and ρ denote the variable node and check node degree distributions, respectively.

$$\lambda(x)=\sum_{i\geq 2}^{d_v^{\max}}\lambda_i x^{i-1}=0.2084x+0.084x^2+0.7076x^4,$$

$$\rho(x)=\sum_{i\geq 2}^{d_c^{\max}}\rho_i x^{i-1}=0.525x^{14}+0.475x^{15}$$

where λ_i and ρ_i denote the fractions of edges incident to the variable and check nodes with degree i, respectively.

3.3.2 Code parameters for puncturing:

The rate-1/3 mother code for puncturing has the following degree distributions [11].

$$\lambda(x)=0.625x+0.291x^2+0.084x^{15},$$
$$\rho(x)=0.8125x^4+0.1875x^5.$$

This code has better asymptotic performance than Turbo code proposed in 3GPP [11]. Also, the optimal puncturing patterns proposed in [5] are used to construct RC B-LDPC codes.

3.3.3 Code parameters for extending & puncturing:

The rate-1/2 mother code for extending & puncturing has the following degree distributions [3].

$$\lambda(x)=0.4583x+0.333x^2+0.2084x^5,,$$
$$\rho(x)=0.666x^5+0.333x^6.$$

To obtain higher-rate codes from the mother code, we use the puncturing scheme in [5] and to obtain lower-rate codes from the mother code, we adopt the extending scheme proposed in [8]. We derive the following degree distributions for the rate-1/3 code by extending.

$$\lambda(x)=0.6112x+0.0555x^2+0.1945x^3$$
$$+0.0833x^5+0.0555x^{11},$$
$$\rho(x)=0.5x^3+0.3333x^5+0.1667x^6.$$

By using the mother B-LDPC codes with degree distributions in (i), (ii), and (iii), we will compare the

Table. 2 Overall decoding complexittes for variable and check node additions and special operations for each code rate.

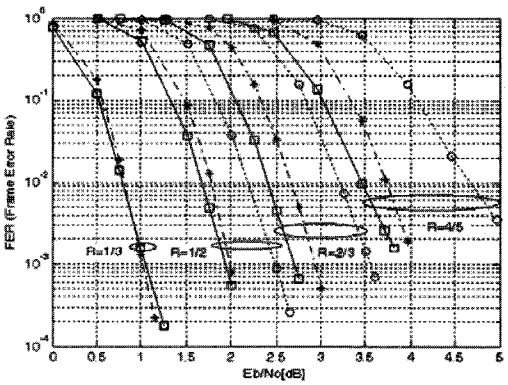
Rate-Control Scheme	Code Rate	Number of Variable (or Check) Node Update Additions	Number of Special Operations $f(x)$
Puncturing	1/3,1/2, 2/3,4/5	438240 (417120)	438240
Splitting	1/3	438240 (417120)	438240
	1/2	353760 (343200)	353760
	2/3	311520 (306240)	311520
	4/5	290400 (287760)	290400
Extending & Puncturing	1/3	457600 (436480)	457600
	1/2, 2/3,4/5	267520 (256960)	267520

decoding complexities of splitting, puncturing, and extending & puncturing using Theorem 2. By using the parameters defined in the subsection III-B, we can get $n_P = (n-m)(1/R-1/R_m) = (n_b-m_b)z(1/R-1/R_m)$ where m_b and n_b are the row and column numbers of model matrix of B-LDPC code and $z \times z$ is the size of a circulant as defined in [3]. Suppose that $z=132$ and the number of iterations is 20. For splitting and puncturing, we use $m_b=16$ and $n_b=24$. Compared with the punctured B-LDPC codes, the overall numbers of additions for variable node message updates using approach ① for split B-LDPC codes are reduced by 84480, 126720, and 147840 for the code rates 1/2, 2/3, and 4/5, respectively. Also, for the check node message updates, the same number of special operations is reduced. The overall numbers of additions for check node message updates are reduced by 73920, 110880, and 129360 for the code rate 1/2, 2/3, and 4/5, respectively. For extending & puncturing, we consider the mother B-LDPC code of rate 1/2 with $m_b=12$ and $n_b=24$. Then, the extended B-LDPC code of rate 1/3 should have $m_b=24$ and $n_b=36$. Suppose that $z=88$ to have the same code length for the puncturing and splitting cases and the number of iterations is 20. For the code rate 1/2, the overall number of additions for variable node message updates using approach ① for B-LDPC code obtained by extending & puncturing is reduced by 170720 and 86240, compared with the puncturing and splitting, respectively, since its parity-check matrix is sparser than those for splitting and puncturing. Also, for the check node message updates, the same number of special operations is reduced. The overall numbers of additions for check node message updates are reduced by 160160 and 86240 compared with the puncturing and splitting, respectively. For other code rates, we can easily compare the complexities of these rate-control schemes. Table 2 shows the overall computational decoding complexity for variable and check node additions and special operations for each code rate.

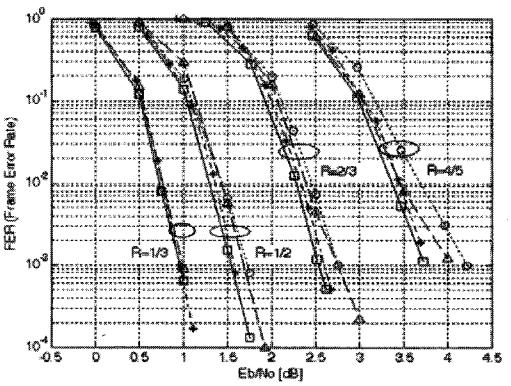
Punctured codes can be encoded and decoded using a single encoder/decoder pair, i.e., the system needs only mother parity-check matrix. For extending & puncturing, a single encoder/decoder pair is needed for punctured codes and different encoder/decoder pair is necessary for each extended code. For splitting, different encoder/decoder pair is needed for each code, i.e., all parity-check matrices for each code rate should be saved. However, this additional complexity may be negligible since the overall decoding complexity of splitting is much lower than that of puncturing.

3.4 Simulation results

For the simulation, BPSK modulation, AWGN channel, 1056 information bits (for splitting and puncturing, we use $m_b=16$, $n_b=24$, and $z=132$, and for extending & puncturing, we consider the mother B-LDPC code of rate 1/2 with $m_b=12$, $n_b=24$, and $z=88$. Then, the extended B-LDPC code of rate 1/3 should have $m_b=24$, $n_b=36$, and $z=88$.), and belief propagation algorithm are used. Since splitting does not introduce punctured nodes in Tanner graph, it gives faster convergence speed than puncturing. When the number of iterations is 20, Fig. 5(a) shows that, at rates between 1/2 and 4/5, RC B-LDPC codes using splitting have about 0.1dB-0.25dB and 0.5dB-1.2dB gains at FER=10-2 over RC B-LDPC codes using extending & puncturing and only puncturing. Fig. 5(b) shows that as the number of iterations increases, the performance gap becomes smaller, and to show



(a) The number of iterations is 20.



(b) The number of iterations is 50.

Fig. 5 Performance comparison of various codes. (Solid line with squares: RC B-LDPC code using splitting; dash-dot line with stars: RC B-LDPC code using extending & puncturing; dotted line with circles: RC B-LDPC code using only puncturing dashed line with triangle: Turbo code.)

the good rate-compatibility and good performance of proposed B-LDPC code, it is compared with Turbo code adopted in 3GPP [12]. Fig. 6 compares the throughputs of type-II HARQ schemes using three RC B-LDPC codes for code rates 1/3, 1/2, 2/3, and 4/5. We assume that noiseless feedback link is available so that the receiver can reliably inform the transmitter of the decoding result and round trip delay is ignored. Throughput η, the measure of the performance of HARQ, is defined in [7]. Fig. 6 shows that the type-II HARQ using RC B-LDPC code obtained by splitting has the best throughput. Therefore, from simulation, we can also verify that RC B-LDPC code obtained by splitting has good performance and fast convergence speed.

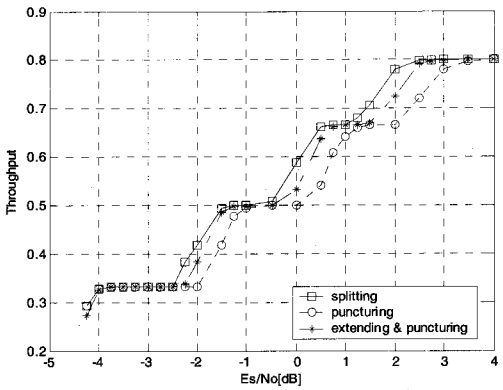


Fig. 6 Throughput comparison of type-II HARQ schemes using various RC B-LDPC codes. (The number of iterations is 20.)

Ⅳ. Conclusions

In this paper, we propose a new rate-control scheme called splitting. It is shown that splitting not only shows faster decoding convergence speed than other rate-control schemes such as puncturing and extending but also has low decoding complexity. By using splitting, we explicitly construct RC B-LDPC code achieving code rates 1/3, 1/2, 2/3, and 4/5, which is compared with RC B-LDPC codes using puncturing only and extending & puncturing. By simulation, we verify that RC B-LDPC code using splitting gives better performance and higher throughput of type-II HARQ than other RC B-LDPC codes, especially when the number of iterations is limited. Therefore, RC RA-Type LDPC codes using splitting can be a good candidate for the next-generation communication system which requires high system throughput.

References

[1] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," IEEE Trans. Inform. Theory, pp. 638-656, Feb. 2001.

[2] H. Jin, A. Khandekar, and R. J. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Int. Symp. on Turbo Codes and Related Topics*, pp. 1-8, Sep. 2000.

[3] IEEE 802.16 Broadband Wireless Access Project, "IEEE C802.16e-05/066r3" Jan. 2005.

[4] U. Dammer, E. Naroska, S. Schmermbeck and U. Schwiegelshohn, "A data puncturing IR-scheme for type-II hybrid ARQ protocols using LDPC codes," *IEEE Global Comm. Conf.(GLOBECOM)*, Nov. 2004 pp. 3012-3016.

[5] S.-N. Hong and D.-S. Park "Rate-Compatible Puncturing for Finite-Length Low-Density Parity-Check codes with Zigzag Parity Structure," accepted in *IEEE Int. Symp. Personal, Indoor and Mobile Radio Comm. (PIMRC'06)*, 2006.

[6] 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*, pp. 728-738, Feb. 2006.

[7] J. Li and K. R. Narayanan, "Rate-compatible low density parity check codes for capacity-approaching ARQ schemes in packet data communications," *Int. Conf. on Comm., Internet, and Info. Tech. (CIIT)*, pp.201-206, Nov. 2002.

[8] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Comm. Lett.*, pp. 159-161, Mar. 2004.

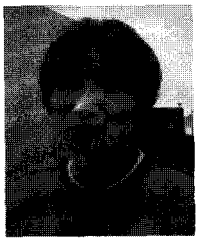
[9] T. J. Richardson, A. Shokrollahi and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, pp. 619-637, Feb. 2001.

[10] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X. Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Comm.*, pp. 1288-1299, Aug. 2005.

[11] "Comparison of structured LDPC Codes and 3GPP Turbo codes," 3GPP TSGR1#43(05) 1360,Nov. 2005

[12] 3rd Generation Partnership Project (3GPP), "High speed downlink packet access," 3GPP TR 25.855, June 2001.

Hyeong-Gun Joo Regular Member



Hyeong-Gun Joo received the B.S. degree in division of electrical and electronics engineering from Gyeongsang National University in 2003 and the M.S. degree in division of electronics and computer engineering from Hanyang University in 2006. He is currently pursuing the Ph.D. degree at the same university. He has been engaged in research and design of iterative decodable code for error correction in communication systems. His research interests include channel code, MIMO, space-time code, and relay networks.

Song-Nam Hong Regular Member

Song-Nam Hong received the B.S. and M.S. degrees in division of electronics and computer engineering from Hanyang University in 2003 and 2005, respectively. Since 2005, he has worked in Telecommunication R&D Center in Samsung Electronics Co., Ltd., Korea. He has been engaged in research and design of iterative decodable code for error correction in communication systems. His area of research interests include channel code, MIMO, space-time code, and relay networks.

Dong-Joon Shin Lifelong Member

Dong-Joon Shin received the B.S. degree in electronics engineering from Seoul National University in 1990, the M.S. degree in electrical engineering from Northwestern University in 1991, and Ph.D. degree in electrical engineering from University of Southern California in 1998. He was a member of technical staff at Hughes network systems from 1999 to 2000. Since 2000, he has been a faculty member of the department of electronics and computer engineering, Hanyang University. His area of research interests includes error correcting codes, LDPC codes, space time codes, OFDM, sequences, and mobile radio communication systems.