Min-Sum 반복 복호 알고리즘을 사용한 Tree-LDPC의 성능과 수렴 분석

준회원 노 광 석*, 정회원 허 준*, 정 규 혁**

Performance and Convergence Analysis of Tree-LDPC codes on the Min-Sum Iterative Decoding Algorithm

Kwang-seok Noh* *Associate Member*, Jun Heo*, Kyuhyuk Chung** *Regular Members*

요 약

본 논문에서는 Tree-LDPC 코드의 성능을 scaling 인자를 이용한 min-sum 알고리즘을 사용하여 나타내고, 그 때의 water fall 영역에서의 점근 성능은 density evolution 기법을 사용하여 나타낸다. Density evolution 기법을 통하여 얻어진 최적의 scaling 인자를 사용하게 되면 min-sum 알고리즘을 사용하는 Tree-LDPC 코드는 sum-product 알고리즘을 사용했을 때와 비슷한 성능을 나타낼 정도로 상당한 성능 이득을 갖게 되는 반면 sum-product 알고리즘을 사용했을 때보다 복호 복잡도가 훨씬 줄어들게 된다. 작은 인터리버 크기를 갖는 Tree-LDPC 복호기를 FPGA(Field Programmable Gate Array)로 구현하였다.

Key Words : Tree-LDPC codes, density evolution, scaling factor, iterative decoding

ABSTRACT

In this paper, the performance of Tree-LDPC code[1] is presented based on the min-sum algorithm with scaling and the asymptotic performance in the water fall region is shown by density evolution. We presents that the Tree-LDPC code show a significant performance gain by scaling with the optimal scaling factor which is obtained by density evolution methods. We also show that the performance of min-sum with scaling is as good as the performance of sum-product while the decoding complexity of min-sum algorithm is much lower than that of sum-product algorithm. The Tree-LDPC decoder is implemented on a FPGA chip with a small interleaver size.

Ⅰ. 서 론

최근 이동통신은 고속 데이터 전송과 이동성이 중요시 되고 있으며 이는 국제 표준에서도 고려되고 있다. 이러한 국제 표준 중 가장 선도적인 위치를 차지하고 있는 하나를 든다면 단연 국제 휴대 인터넷 표준 IEEE 802.16e 라고 볼 수 있다. 현재 활발하게 국제 표준화가 진행 중인 IEEE 802.16e 에서 고려되고 있는 물리 계층의 기술은 OFDMA와 LDPC[2] 코드이다. 빠른 이동성을 보장하는데 필연적으로 따라오는 심각한 페이딩 채널의 변화를 손쉽게 보정하기 위해 OFDMA가 채택되었으며 낮은 신호대 잡음비 영역에서 고속 데이터 전송을 가능케 하기 위해 LDPC 코드가 채택되었다.

다른 채널 코딩 기술에 비하여 LDPC 코드는 고속 데이터 전송에 필요한 높은 코드 레이트에서의

※ 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임. (R08-2003-000-10165-0)
* 건국대학교 전자공학과 ({kwnoh, junheo}@konkuk.ac.kr), 차세대 혁신 기술 연구원(NITRI)
** 단국대학교 정보컴퓨터학부 (khchung@dankook.ac.kr)
논문번호 : KICS2005-10-398, 접수일자 : 2005년 10월 4일

탁월한 성능을 나타내며 아울러 BP(Belief Propagation) 알고리즘을 사용한 복호 알고리즘 하드웨어 구현에 있어서 전체적으로 패러렐 프로세싱이 가능하다는 장점이 있다. 또한 LDPC 코드 자체적으로 에러 디텍션 기능을 가지고 있어 추가적인 오버 헤드인 CRC 코드를 필요하지 않는 것이 또 다른 장점이다.

이러한 많은 장점들을 가지고 있는 LDPC 코드 기술은 인코딩에 있어서 많은 계산 량을 가지는 것이 하나의 단점으로 여겨져 왔다. 반면 간단한 repetition 코드와 accumulator를 하나의 인터리버로 연결시킨 RA 계통의 코드[3]는 LDPC 코드와 같은 좋은 성능을 보이진 못하지만 인코딩이 간단하다는 장점이 있다.

본 논문에서는 LDPC 코드의 복잡한 인코딩의 단점을 극복하기 위하여 간단한 인코딩이 가능한 RA 코드 계통의 기술들과 LDPC 코드의 기술을 접목시킨 Tree-LDPC 코드 기술에 대하여 살펴보고 하드웨어 구현에 적합한 min-sum 알고리즘을 사용했을 때의 성능 저하를 보정해 줄 수 있는 최적의 scaling 인자를 구하고자 한다. Ⅱ장에서는 Tree-LDPC 코드 구조에 대해 살펴보고, Ⅲ장에서는 Density Evolution을 이용하여 Tree-LDPC 코드의 분석 및 최적의 scaling 인자를 찾는 기법에 대해 고찰한다, Ⅳ장에서는 Tree-LDPC 복호기에 대한 하드웨어 구현을 살펴보고,, Ⅴ장에서 결론을 맺도록 한다.

Ⅱ. Tree LDPC 코드 구조

Tree LDPC 코드는 tree 코드 구조를 가지는 parity check 행렬로 표현되게 된다. Tree 구조, 다시 말해 tree 코드는 recursive 정보 비트 d_r, non-recursive 정보 비트 d_{nr}, puncturing 패리티 비트 p_p, 그리고 non-puncturing 패리티 비트 p_{np}로 구성된다. 이들 사이의 관계는 다음과 같은 식으로 표현될 수 있다.

$$p_p[k] = \sum_{i=0}^{J_r-1} d_r[(k-1)J_r + i] + p_p[k-1],\ k=1,2,3,\cdots$$

$$p_{np}[k] = \sum_{i=0}^{J_{nr}-1} d_{nr}[(k-1)J_{nr} + i] + p_p[k],\ k=1,2,3,\cdots$$

여기에서 J_r은 recursive SPC(Single Parity Check)에 속하는 recursive 비트의 수를 나타내며 J_{nr}은 non-recursive SPC(Single Parity Check)에 속하는 non-recursive 비트의 수를 나타내고 $p_p[0]$ 0이다.

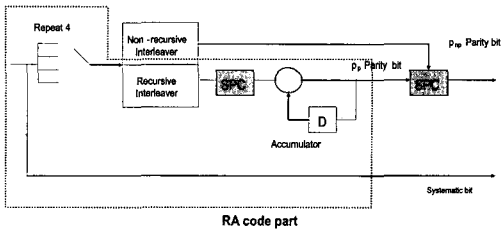

그림 1. Tree-LDPC 코드의 블록 다이어그램
(Block diagram of tree LDPC codes

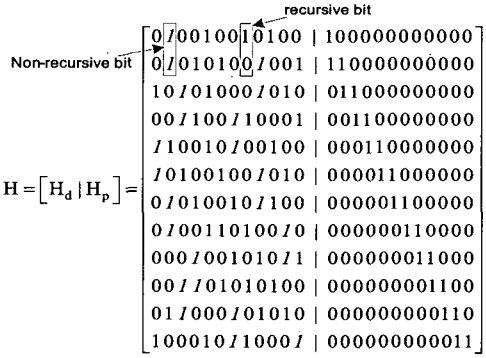

그림 2. Tree-LDPC 코드의 parity check 행렬 H의 예
(An example for parity check matrix H of a Tree-LDPC code)

Tree-LDPC 코드는 Turbo-like codes와 LDPC 코드의 특징을 가지고 있다. 각 특징을 바탕으로 Tree-LDPC 코드는 다음 2가지 형태로 구성될 수 있다. 첫 번째는 SCC(Serial Concatenated Codes)의 형태이고, 다른 하나는 Parity Check 행렬을 사용하는 선형블록부호의 형태이다. 첫 번째로, SCC의 경우에 Puncturing 패리티 비트는 recursive 비트의 SPC와 accumulator로부터 얻어진다. 한편 non- puncturing 패리티 비트는 puncturing 패리티 비트의 SPC와 non-recursive 정보 비트로부터 얻어지게 된다. 그림 1은 SCC 형태의 Tree- LDPC 코드를 non-recursive 정보 비트 부분과 RA 코드 부분으로 나누어 블록 다이어그램으로 도시해 놓았다. 두 번째로 Tree-LDPC 코드를 parity check 행렬을 이용하여 선형블록부호의 형태로 나타낸다. 좋은 성능을 보장하기 위해선 하나의 정보 비트가 적어도 두 개의 recursive 코드에 연결돼야만 한다는 사실은 잘 알려져 있다. 이러한 법칙을 따라 정보 비트가 적어도 두 번은 recursive check sum에 속하도록 parity check 행렬이 설계되어졌다($J_r \geq 2$). 그림 2는 Tree-LDPC 코드에 사용된 parity check 행렬 H의 예이다. H_d는 repetition과 interleaver에 해당하

며, H_p는 accumulator에 해당한다. 이럴 때 구조화된(constrained) LDPC 코드에서 parity check 행렬 H의 각각의 행은 하나의 SPC로 표현되어지게 된다. 이러한 SPC들은 accumulator를 통하여 연결되어진다. 이러한 역할을 하는 부분이 parity check 행렬 H의 듀얼 다이어고날 구조를 가지는 H_p부분이다. 그러므로 구조화된(constrained) LDPC 코드는 RA 코드와 마찬가지로 선형 시간으로 인코딩하는 것이 가능하게 된다. 본 예시에서 각각의 행은 2개의 non-re-cursive 비트를 가지게 되는데, 그림 2에서는 이탤릭체로 표현되었다. 이러한 non- recursive 비트는 다음과 같은 수식으로 표현되어지는 것이 가능하다.

$$p_{np}[k]=p_{np}[k-1]+\sum_{i=0}^{J_{nr}-1}d_{nr}[(k-1)J_{nr}+i]+\sum_{i=0}^{J_r-1}d_r[(k-1)J_r+i]+\sum_{i=0}^{J_{nr}-1}d_{nr}[(k-2)J_{nr}+i]$$

Ⅲ. Density Evolution과 성능

Density evolution[4] 기술은 최근에 LDPC 코드의 수렴 성능의 분석을 위해 널리 사용되어지고 있다. Density evolution 기술을 통해 얻어지는 threshold 값은 에러 확률이 제로로 수렴하는 최소한의 SNR(dB) 값을 나타낸다. 본 논문에서는 가우시안 근사화(GA)를 이용하여 Tree-LDPC 코드의 threshold 값을 유도하였다.

가우시안 근사화를 이용한 density evolution은 message의 평균을 추적하여 계산할 수 있다. 변수 노드 및 punctured parity 노드에서는 노드의 입력 message의 합이 출력 message가 되고, check 노드에서는 sum-product 알고리즘을 사용할 때는 tanh(·)을 사용하고, min-sum 알고리즘을 사용할 때는 입력값들의 절대값의 최소치와 입력값들의 부호로 출력이 결정된다.

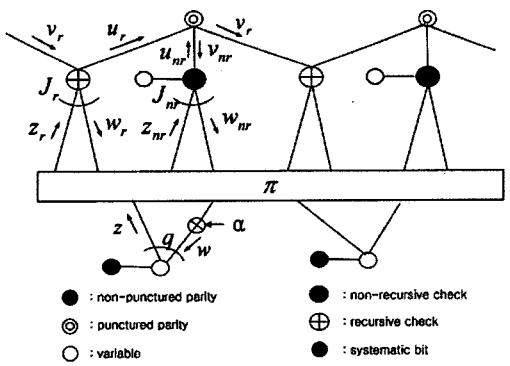

그림 3. Scaling factor를 사용한 Tree-LDPC 코드의 bipartite 그래프
(The bipartite graph of the irregular Tree- LDPC codes with scaling factor)

3.1 Density Evolution과 Threshold

본 장에서는 min-sum 알고리즘에서 scaling factor를 갖는 Tree-LDPC 코드의 density evolution을 논문[5]를 바탕으로 나타낸다. Tree-LDPC 코드의 변수 노드와 check 노드에서의 min-sum message 전달 알고리즘은 다음과 같이 나타낸다.

$$z=\lambda_c+\sum_{i=1}^{q-1}w_i \tag{1}$$

$$u_r=sign(v_r,z_{r_1},\cdots,z_{r_{J_r-1}})\min[|v_r|,|z_{r_1}|,\cdots,|z_{r_{J_r-1}}|]$$

$$u_{nr}=sign(\lambda_c,z_{nr_1},\cdots,z_{nr_{J_{nr}-1}})\min[|\lambda_c|,|z_{nr_1}|,\cdots,|z_{nr_{J_{nr}-1}}|]$$

z는 변수 노드의 Log Likelihood Ratio(LLR) 출력이고, λ_c는 채널로부터 받은 message를 나타낸다. u_r, u_{nr}은 각각 recursive check 노드, non-recursive check 노드의 LLR 출력이다.

가우시안 근사화를 이용하여 min-sum density evolution을 구하려면 반복 복호를 하는 동안에 LLR값들의 평균 $\overline{z}$, $\overline{u}_r$, $\overline{u}_{nr}$과 분산 σ_z^2, σ_{ur}^2, σ_{unr}^2을 추적하면 된다. 교환되는 message에 independent and identically distribution 가정을 적용하면 $\overline{z}$와 σ_z^2는 다음과 같이 구할 수 있다. 여기서 주의할 것은 sum-product의 경우와는 다르게 min-sum에서는 message 의 확률 밀도 함수(PDF)에서 symmetric condition 이 성립하지 않으므로 평균과 분산을 모두 구해야 하는 것이다[6].

$$\overline{z}=\overline{\lambda}_c+(q-1)\overline{w} \tag{2}$$

$\sigma_z^2=\sigma_{\lambda_c}^2+(q-1)\sigma_w^2$ $\overline{u}_r$, σ_r^2의 계산은 좀 더 복잡하여, J_r=3 일 때 check 노드의 출력 message의 확률 밀도 함수(PDF)는 [6]에 의해 다음과 같이 나타낸다.

$$f_{u_r}(x) = f_{v1}(x)(1-F_{v2}(x)) + f_{v2}(x)(1-F_{v1}(x)) + f_{v1}(-x)F_{v2}(-x) + f_{v2}(-x)F_{v1}(-x),\ x>0$$
$$f_{u_r}(x) = f_{v1}(x)(1-F_{v2}(-x)) + f_{v2}(x)(1-F_{v1}(-x)) + f_{v1}(-x)F_{v2}(x) + f_{v2}(-x)F_{v1}(x),\ x<0 \quad (3)$$

f(x)와 F(x)는 각각 PDF와 누적 밀도 함수(CDF)를 나타낸다. $J_r > 3$인 경우 $f_{u_r}(x)$는 추가적인 PDF $f_{v_i}(x)$를 이용하여 ($i \le J_r - 1$) 다음과 같은 recursive update를 통해 구할 수 있다.

$$f_{u_r}(x) = G(\cdots G(G(f_{v_1}, f_{v_2}), f_{v_3}), \cdots, f_{v_{J_r-1}}) \quad (4)$$

함수 G(·)는 식 (4)의 출력 message의 PDF를 구하는 함수의 단축 표기이다. 추적해가는 평균과 분산의 첫 번째 반복에서 사용되는 초기값은 다음과 같이 채널로부터 수신된 message λ_c로 나타낸다. λ_c는 symmetric 채널에서 받은 값이기 때문에 symmetric 조건이 유지된다. 따라서 분산 $\sigma^2_{\lambda_c}$는 $2\overline{\lambda}_c$이다.

$$\overline{z}^{(0)} = \overline{\lambda}_c$$
$$\overline{u}_{nr}^{(0)} = sign(\overline{\lambda}_c) sign(\prod_{1}^{J_{nr}} \overline{z}_{nr}^{(0)}) \min[|\overline{\lambda}_c|, |\overline{z}_{nr}^{(0)}|]$$
$$\sigma_z^{2\,(0)} = \sigma_{\lambda_c}^2 = 2\overline{\lambda}_c,\ \sigma_{u_{nr}}^{2\,(0)} = \sigma_{u_{nr}}^2 \quad (5)$$

(l)번째 반복에서의 평균과 분산은 (l-1)번째 반복에서의 평균과 분산을 이용하여 구한다. 변수 노드에서 recursive check 노드와 non-recursive check 노드로 가는 message의 평균과 분산은 다음과 같다.

$$\overline{z}^{(l)} = \overline{\lambda}_c + (q-1)\times \overline{w}^{(l-1)}$$
$$\sigma_z^{2\,(l)} = \sigma_{\lambda_c}^2 + (q-1)\times\sigma_w^{2\,(l-1)}$$
$$\overline{z}_r^{(l)} = \overline{z}_{nr}^{(l)} = \overline{z}^{(l)}$$
$$\sigma_{z_r}^{2\,(l)} = \sigma_{z_{nr}}^{2\,(l)} = \sigma_z^{2\,(l)} \quad (6)$$

마찬가지로, punctured parity 노드에서 recursive check 노드와 non-recursive check 노드로 가는 message의 평균과 분산은 다음과 같다.

$$\overline{v}_r = \overline{u}_r + \overline{u}_{nr}$$
$$\sigma_{v_r}^{2\,(l)} = \sigma_{u_r}^{2\,(l)} + \sigma_{u_{nr}}^{2\,(l)}$$
$$\overline{v}_{nr} = 2\times\overline{u}_r$$
$$\sigma_{v_{nr}}^{2\,(l)} = 2\times\sigma_{v_{nr}}^{2\,(l)} \quad (7)$$

Message의 평균과 분산을 구하는 과정을 요약해 보면, 우선 z의 평균과 분산과 가우시안 근사화를 이용해서 z의 PDF, CDF를 구한다. 이 것과 식 (3)을 이용하여 u_{nr}의 PDF, CDF를 구하고 이를 이용하여 u_{nr}의 평균과 분산을 구한다. 그 다음, u_{nr}의 평균과 분산을 이용하여 v_r의 평균과 분산을 구하고, 이 값을 이용하여 v_r의 PDF, CDF를 구한다. 마찬가지 방법으로 u_r과 v_{nr}, w_r, w_{nr}의 평균과 분산, PDF, CDF를 구하고 w'는 다음과 같이 구한다.

$$w' = \frac{J_r}{J_r + J_{nr}} \times w_r + \frac{J_{nr}}{J_r + J_{nr}} \times w_{nr} \quad (8)$$

scaling 인자 α로 인하여 w는 다음과 같이 바뀐다.

$$\overline{w} = \alpha\,\overline{w'}$$
$$\sigma_w^{2\,(l)} = \alpha^2\sigma_{w'}^{2\,(l)} \quad (9)$$

위와 같은 방법으로 (l+1)번째 반복을 수행한다.

표 1은 각 scaling 인자에 따라 density evolution에 의해 구해진 threshold들을 나타낸다. recursive 비트와 non-recursive 비트의 비율이 2 대 2, 3대 1인 경우 각각 scaling 인자가 0.7, 0.6일 때 가장 낮은 threshold를 갖으며 이때의 scaling 인자가 최적의 값임을 알 수 있다.

표 1. 다양한 scaling 인자에 따른 Tree-LDPC 코드의 threshold (Thresholds of Tree-LDPC codes for different scaling factors.)

Scaling Factor α	Min-sum algorithm with scaling factor (E_b/No dB)		Sum-product algorithm without scaling factor (E_b/No dB)	
α	J_r=2,J_{nr}=2	J_r=3,J_{nr}=1	J_r=2,J_{nr}=2	J_r=3,J_{nr}=1
1	1.61	1.47	0.65	0.77
0.9	1.33	1.19		
0.8	1.02	0.94		
0.7	0.82	0.81		
0.6	5.52	0.80		
0.5		1.27		

Ⅳ. 시뮬레이션 결과 및 성능 분석

그림 4에서는 Tree-LDPC 코드의 성능을 컴퓨터 시뮬레이션을 통해서 구해보았다. 본 시뮬레이션 실험에 사용된 Tree-LDPC 코드는 정보 블록의 크기는 2048비트이고, repetition 수는 4이고 recursive비트와 non-recursive 비트의 비율이 2 대 2인 inner code를 갖는다. AWGN 채널을 사용하였으며반복복호의 수는 20번으로 고정하였고, 전체 코드 rate는 1/2이다. 비교를 위해 동일한 조건에서 Tree-LDPC 코드에 scaling 인자를 사용하지 않은 sum-product 알고리즘 성능도 나타내었다.

Density evolution에서 얻은 최적의 scaling 인자(α=0.7)가 다른 scaling 인자를 사용한 것에 비해 성능이 우수한 것을 시뮬레이션을 통해 확인 할 수 있다.

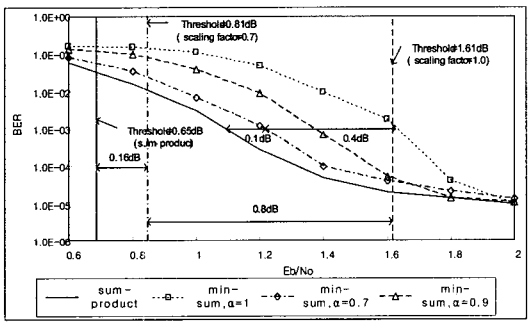

그림 4. AWGN에서 다양한 scaling 인자값에 대한 2 대 2 Tree-LDPC 코드의 성능
(Performance of Tree-LDPC 2-to-2 code over AWGN with various scaling factors)

최적의 scaling 인자를 갖고 min-sum 알고리즘을 사용하는 Tree-LDPC 코드와 scaling 인자를 갖지 않은 min-sum 알고리즘을 사용하는 Tree-LDPC 코드에 대한 성능 향상은 Bit Error Rate(BER)가 10^{-3}인 영역에서 약 0.4 dB이며 최적의 scaling인자를 갖고 min-sum 알고리즘을 사용하는 Tree-LDPC 코드와 sum-product 알고리즘을 사용하는 Tree-LDPC 코드는 0.1 dB 차이로 성능이 거의 근접한 것을 알 수 있다. 이러한 성능 향상은 density evolution을 통해 얻어진 threshold 결과와도 일치하는 것을 볼 수 있다.

Ⅴ. 하드웨어 구현

Tree-LDPC 코드의 복호기를 FPGA를 이용하여

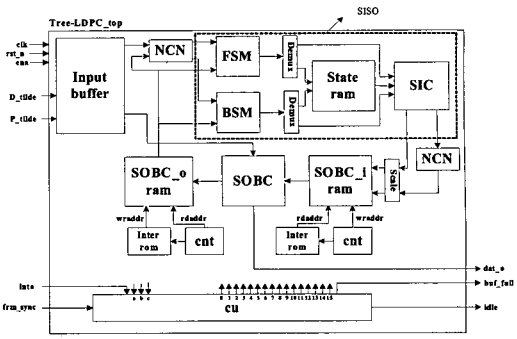

그림 5. Tree-LDPC 코드의 블록도(block diagram of Tree-LDPC code)

구현한 블록도를 그림 5에 나타내었다. Tree 코드 부분에 대한 복호 알고리즘으로는 Forward-backward 알고리즘(FBA)를 사용하였으며, repetition 코드 부분에 대한 복호 알고리즘은 SOBC(Soft Out Broad Caster)를 사용하였다. FSM 블록과 BSM 블록은 각각 Forward State Metric, Back-ward State Metric을 계산하는 블록이고, SIC는 Soft Information을 계산한다. NCN은 Tree 코드를 복호하는 블록이고, SOBC는 Repetition 코드를 복호하는 블록을 각각 나타낸다. 정보 블록의 크기는 1024비트이며 Xilinx사의 xc2v3000 FPGA 칩을 사용하였다.

Ⅵ. 결 론

본 논문에서는 Tree-LDPC 코드가 최적의 scaling 인자를 갖는 min-sum 알고리즘을 사용하는 경우에 커다란 성능 이득을 갖는다는 것을 보였다. 또한 최적의 scaling 인자를 결정하는 방법으로 density evolution이 어떻게 사용되는지 살펴보았다. density evolution을 통하여 얻은 결과는 컴퓨터 시뮬레이션을 통해 확인하였다.

참 고 문 헌

[1] Jun Heo and Kyuhyuk Chung, "Tree-LDPC codes for IEEE 802.16 broadband wireless internet," ICCE, Las Vegas January 2005.

[2] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21-28, January 1962.

[3] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for turbo-like codes," in Proc. 36th Annu. Allerton Conference Communi-

cation, Control, Computing, pp. 201-210, September 1998.

[3] L. Ping and W. Y. Keying, "Concatenated tree codes: A low complexity, high-performance approach," IEEE Trans. Inform. Theory, vol. 47, pp. 791-799, February 2001.

[4] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under . message-passing decoding," IEEE Trans. Inform.

[5] Jun Heo, "Performance and Convergence Analysis of Improved MIN-SUM Iterative Decoding Algorithm," IEICE Trans. on Comm., vol.E87-B, no 10., pp. 2847-2858, Oct. 2004.

[6] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithm based on density evolution," in Proc. Globecom Conf., pp.1021-1025, 2001.

노 광 석 (Kwang-seok Noh) 준회원

2004년 건국대학교 전자공학과 학사

2004년~현재 건국대학교 전자공학과 석사과정

<관심분야> 통신공학, 채널코딩

허 준 (Jun Heo) 정회원

1989년 서울대학교 전자공학과 학사

1991년 서울대학교 대학원 전자공학과 석사

1991년~1996년 LG전자 영상미디어 연구소, 선임연구원

1996년~1997년 미국 Stanford University, information System Lab., 객원연구원

2000년~2001년 미국 Trellisware Technologies Inc., Member of technical staff

2002년 미국 University Southern California 박사

2002년~2003년 Hynix반도체(주) System IC Comp. 책임연구원

2003년~현재 건국대학교 전자공학부, 조교수

<관심분야> 통신공학, 부호이론, 채널코딩

정 규 혁 (Kyuhyuk Chung) 정회원

1997년 성균관대학교 전자공학과 학사

1998년 미국 University Southern California 대학원 석사

2003년 미국 University Southern California 대학원 박사

2004년~현재 LG 전자 이동통신

2005년 9월~현재 단국대학교 정보컴퓨터학부, 전임강사

<관심분야> 통신공학, 부호이론, 채널코딩