한국정보전자통신기술학회논문지(jkiiect)'18-08, Vol.11 No.4
논문 18-11-04-430 http://dx.doi.org/10.17661/jkiiect.2018.11.4.430

LCPC 부호의 개선된 복호 방식

정호영*

An Improved Decoding Scheme of LCPC Codes

Ho-Young Cheong*

요 약 본 논문에서는 부호 길이가 작은 LCPC 부호에 대한 개선된 복호 방식을 제안하였다. LCPC 부호는 터보 부호나 LDPC 부호에 비해 복잡도가 낮고 요구되는 메모리도 적어 IoT 단말 간 통신에 적합하다. IoT 단말은 에너지가 제한되어 있어서 복잡도가 낮아야 하며 종단 간 지연 시간이 짧아야 하는 경우가 많다. 또한, 전송되는 패킷 길이가 작고 IoT 단말의 신호 처리 능력이 작기 때문에 LCPC 부호 시스템이 가능한 한 간단해야 한다. LCPC 부호는 단일 오류는 모두 정정할 수 있고 2개의 오류 중 일부를 정정할 수 있다. 본 논문에서는 변조기 출력단의 소프트 값을 이용하여 2개의 오류를 모두 정정함으로서 복잡도를 증가시키지 않고서도 비트 오율 성능을 개선하였다. 본 논문에서 제안한 복호 방식을 이용하여 시뮬레이션을 한 결과 기존의 복호 방식에 비해 10^{-4}의 비트 오율에서 약 1.1[dB]의 부호 이득을 얻을 수 있었다.

Abstract In this paper, an improved decoding scheme for low-complexity parity-check(LCPC) code with small code length is proposed. The LCPC code is less complex than the turbo code or low density parity check(LDPC) code and requires less memory, making it suitable for communication between internet-of-things(IoT) devices. The IoT devices are required to have low complexity due to limited energy and have a low end-to-end delay time. In addition, since the packet length to be transmitted is small and the signal processing capability of the IoT terminal is small, the LCPC coding system should be as simple as possible. The LCPC code can correct all single errors and correct some of the two errors. In this paper, the proposed decoding scheme improves the bit error rate(BER) performance without increasing the complexity by correcting both errors using the soft value of the modulator output stage. As a result of the simulation using the proposed decoding scheme, the code gain of about 1.1 [dB] was obtained at the bit error rate of 10^{-5} compared with the existing decoding method.

Key Words : Low-complexity parity-check, Decoding complexity, Soft value, Internet of things, Small code length, Duplicated syndrome

1. 서론

최근의 IoT 환경은 스마트폰, 랩탑과 같은 모바일 기기는 물론 TV, 냉장고 등과 같은 일상적인 생활 기기가 인터넷에 연결되어 인간은 물론 기기들 간의 통신을 통해 서로 연결된 유비쿼터스 사회로 진화하고 있다[1][4].

IoT 기기는 대부분 에너지가 제한된 경우가 많아 데이터를 송수신하는데 많은 에너지를 소비할 수 없고 센서와 같은 단말기의 경우 길이가 짧은 데이터를 끊임없이 전송하는 특성을 갖는다. 에너지 관점에서 효율적인 통신을 하기 위해 일반적으로 사용되는 방법이 오류정정부호를 사용하는 것이다. LCPC 부호는 복잡도가 낮고 짧은 길이를 가지면서도 적정한 신뢰도를 보인다는 점에서 이러한 IoT 환경에 적합한 부호로 제안되었다[2]. Hamming 부호와 같은 패리티 검사 부호는 조합 논리 회로 형태로 복호기를 구성할 수 있을 정도로 복호 알고리즘이 간단하여 요구되는 메모리 용량도 작을 뿐만 아니라 복호 지연도

Funding for this paper was provided by Namseoul University year 2017.
*Department of Information and Communication Engineering, Namseoul University
Received August 06, 2018 Revised August 06, 2018 Accepted August 14, 2018

거의 없다[3]. 패리티 검사 부호의 복호 알고리즘은 터보 부호나 LDPC 부호와 같이 반복 복호 형태가 아니므로 복호 지연이 거의 없고 데이터 처리를 위한 프로세서 성능도 크게 요구되지 않는다. 따라서 에너지 자원이 많지 않은 IoT 기기의 통신에 적합하다고 할 수 있다. 그러나 LCPC 부호는 터보 부호나 LDPC 부호에 비해 낮은 의 비트 오율 성능을 보이는 단점을 갖는데[4][5][6][7], 이는 패리티 검사 부호 대부분이 1 개의 오류를 정정하고 2 개의 오류까지 검출할 수 있는 부호이기 때문이다.

[2]에서 제안한 (n, k) LCPC 부호는 패리티 비트 수 $(n-k)$ 값을 늘려 신드롬(syndrome) 벡터 수를 늘린 후 2 개의 오류 벡터 중 일부를 정정할 수 있도록 하여 비트 오율 성능을 늘리고자 제안하였다. 그러나 이는 늘어난 리던던시(redundancy)를 고려하면 기존의 패리티 검사 부호에 비해 SNR 대비 오율 성능은 크게 개선되었다고 볼 수 없다.

본 논문에서는 변조기 출력단의 소프트 값을 이용하여 2 개의 오류를 모두 정정할 수 있는 복호 알고리즘을 제안하였다. 변조기 출력단의 소프트 값은 신드롬 계산 결과 중복된 2-오류 패턴 중 하나를 구별할 때만 사용되므로 복호기 복잡도는 거의 변화가 없다.

본 논문의 구성은 다음과 같다. 2장에서 LCPC 부호의 특성과 부/복호기 동작원리를 설명하였다. 3 장에서는 기존의 LCPC 복호 알고리즘의 문제점을 보이고 이를 보완할 수 있는 복호 알고리즘을 제안하였다. 4장에서는 BPSK(binary phase shift keying) 변조 방식을 이용하여 AWGN(additive white Gaussian noise) 채널에서 LCPC 부호의 BER 성능을 시뮬레이션을 통해 고찰하였으며, 5장에서 결론을 맺었다.

2. LCPC 부호의 특성

2.1 (n,k) LCPC 부호의 부/복호화

(n,k) LCPC 부호는 선형 블록 부호로 식 (1)과 같은 생성 행렬 $\boldsymbol{G}$를 이용하여 k 비트의 메시지 벡터 $\boldsymbol{m} = (m_1,m_2,...,m_k)$로부터 n 비트의 부호 벡터 $\boldsymbol{c} = (c_1,c_2,...,c_n)$를 얻을 수 있다. 부호 벡터 $\boldsymbol{c}$는 행렬 연산 $\boldsymbol{c} = \boldsymbol{m}\,\boldsymbol{G}$를 통해 얻을 수 있는데 $\boldsymbol{c}$의 처음 k 비트는 메시지 비트가 되며 나머지 $(n-k)$ 비트는 $\boldsymbol{c}$의 뒷부분에 위치하는 체계적 형태를 갖게 된다. 식 (1)은 (9,4) LCPC 부호의 생성 행렬을 나타낸 것이다.

$$\boldsymbol{G} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,1\,0 \\ 0\,1\,0\,0\,1\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,0\,1\,1\,1 \end{bmatrix} \tag{1}$$

부호 벡터 $\boldsymbol{c}$는 변조 과정을 거쳐 채널로 전송되고 수신 단에서 다시 복조 과정을 거쳐 수신 부호 벡터 $\boldsymbol{r} = (r_1,r_2,...,r_n)$이 복호기로 입력된다. 복조기 출력 값은 n 개의 아날로그 값을 갖게 되며 본 논문에서는 이를 소프트 벡터 $\boldsymbol{q} = (q_1,q_2,...,q_n)$라고 부르기로 한다. n 개의 소프트 값은 실수 값이며 경 판정(hard decision)을 통해 이진 값 $\{0,1\}$으로 변환되어 수신 부호 벡터 $\boldsymbol{r}$을 형성한다. 경 판정 과정에서 오류가 발생하게 되며 이때 발생한 오류 벡터를 $\boldsymbol{e} = (e_1,e_2,...,e_n)$으로 표현하기로 한다. 여기에서 e_i의 값은 0 혹은 1의 값을 가지며 0이면 오류가 발생하지 않은 것이고 1이면 오류가 발생한 것을 의미한다. 이제 수신 부호 벡터 $\boldsymbol{r}$은 부호 벡터 $\boldsymbol{c}$와 오류 벡터 $\boldsymbol{e}$를 이용해 식 (2)와 같이 나타낼 수 있다.

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e} \tag{2}$$

(n,k) LCPC 부호의 복호(decoding) 과정은 패리티 검사 행렬 $\boldsymbol{H}$를 통해 이루어지는데 생성 행렬 $\boldsymbol{G}$와 $\boldsymbol{H}$가 항상 식 (3)이 성립하는 관계를 갖도록 $\boldsymbol{H}$를 구성한다. 식 (4)는 식 (3)의 관계가 성립하는 (9,4) LCPC 부호의 패리티 검사 행렬을 나타낸 것이다[3].

$$\boldsymbol{G}\boldsymbol{H}^{\boldsymbol{T}} = \boldsymbol{0} \tag{3}$$

$$\boldsymbol{H} = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,1\,0\,0 \\ 1\,0\,1\,1\,0\,0\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,0\,0\,1 \end{bmatrix} \tag{4}$$

수신 부호 벡터 $\boldsymbol{r}$이 복호기에 입력되면 복호기는 식 (5)의 연산을 통해 $(n-k)$-비트로 구성되는 신드롬 벡터 $\boldsymbol{s} = (s_1,s_2,...,s_{n-k})$를 얻는다.

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^{\boldsymbol{T}} \tag{5}$$

신드롬 벡터 $\boldsymbol{s}$는 $(n-k)$-비트로 구성되므로 신드롬

벡터의 총 수는 $2^{(n-k)}$ 이다. Hamming 부호의 경우 신드롬 벡터의 수와 단일 오류 벡터의 수(**0** 벡터 포함)가 일치하며 일 대 일 대응 관계에 있어 단일 오류는 모두 정정할 수 있다. 하지만 (n,k) LCPC 부호의 경우 신드롬 벡터의 수가 단일 오류 벡터의 수 보다 크기 때문에 단일 오류는 물론 2-오류 패턴의 일부도 정정할 수 있다[2]. 예를 들어, (9,4) LCPC 부호의 경우 신드롬 벡터의 수는 $2^{n-k}=2^5=32$개 이며 **0** 벡터를 포함하여 단일 오류 벡터의 수는 10 개이므로 단일 오류 벡터와 일 대 일 대응이 되는 신드롬 벡터를 제외한 22개의 신드롬 벡터는 2-오류 벡터를 정정하는데 이용할 수 있다. $\boldsymbol{r}=\boldsymbol{c}+\boldsymbol{e}$ 이므로 식 (5)로부터 식(6)를 얻을 수 있다.

$$\begin{aligned}\boldsymbol{s} &= \boldsymbol{r}\boldsymbol{H}^{\boldsymbol{T}} \\ &= (\boldsymbol{c}+\boldsymbol{e})\boldsymbol{H}^{\boldsymbol{T}} \\ &= \boldsymbol{c}\boldsymbol{H}^{\boldsymbol{T}}+\boldsymbol{e}\boldsymbol{H}^{\boldsymbol{T}} \\ &= \boldsymbol{m}\boldsymbol{G}\boldsymbol{H}^{\boldsymbol{T}}+\boldsymbol{e}\boldsymbol{H}^{\boldsymbol{T}} \\ &= \boldsymbol{e}\boldsymbol{H}^{\boldsymbol{T}}\end{aligned} \tag{6}$$

식 (6)에서 오류가 발생하지 않으면 $\boldsymbol{e}=\boldsymbol{0}$이므로 $\boldsymbol{s}=\boldsymbol{0}$인 벡터가 될 것이다. 따라서 $\boldsymbol{s}\neq\boldsymbol{0}$이면 오류가 발생한 경우이며 표 1과 같은 복호 표(look-up table)를 이용하여 오류를 정정하게 된다. 표 1과 표 2는 단일 오류 패턴과 2-오류 패턴에 대한 (9,4) LCPC 부호의 복호 표를 나타낸 것이다[2].

표 1. (9,4) LCPC 부호의 1-오류 패턴과 신드롬 벡터 [2]
Table 1. Error pattern and syndrome vector for single bit error of (9,4) LCPC [2]

오류 패턴 $\boldsymbol{e}$	신드롬 $\boldsymbol{s}$
[000000000]	[00000]
[100000000]	[11110]
[010000000]	[11101]
[001000000]	[11011]
[000100000]	[10111]
[000010000]	[10000]
[000001000]	[01000]
[000000100]	[00100]
[000000010]	[00010]
[000000001]	[00001]

표 2. (9,4) LCPC 부호의 2-오류 패턴과 신드롬 벡터 [2]
Table 2. Error pattern and syndrome vector for double bit errors of (9,4) LCPC [2]

오류 패턴 $\boldsymbol{e}$	신드롬 $\boldsymbol{s}$
[100010000]	[01110]
[100000100]	[11010]
[100000001]	[11111]
[010010000]	[01101]
[010000001]	[11100]
[001010000]	[01011]
[001000010]	[11001]
[001000001]	[11010]
[000110000]	[00111]
[000100100]	[10011]
[000100010]	[10101]
[000100001]	[10110]
[000011000]	[11000]
[000010100]	[10100]
[000010010]	[10010]
[000010001]	[10001]
[000001100]	[01100]
[000001010]	[01010]
[000001001]	[01001]
[000000110]	[00110]
[000000101]	[00101]
[000000011]	[00011]

식 (6)에 의해 계산된 신드롬 벡터 $\boldsymbol{s}$가 **0** 벡터가 아닐 경우 표 1과 표 2의 복호 표를 이용하여 단일 오류 전부 혹은 2-오류 패턴의 일부를 복원할 수 있다. 단일 오류 패턴의 경우에는 대응되는 신드롬 벡터가 유일(unique)하므로 단일 오류가 발생한 경우에는 모두 정정할 수 있다. 그러나 2-오류 패턴의 경우에는 표 2의 복호 표에 의한 오류 정정이 오히려 오류를 더하는 결과를 초래할 수도 있다.

예를 들어, 신드롬 $\boldsymbol{s}$ = [00011]에 해당하는 오류 패턴은 표 2의 복호 표를 참조하면 [000000011]이므로 8번 째 오류와 9번 째 오류를 정정하게 되지만, [110000000]도 $\boldsymbol{s}$ = [00011]에 해당되는 오류 패턴이다. 따라서 1번째 와 2번 째 위치에 오류가 발생한 경우이면 두 개의 오류를 전혀 정정하지 못할 뿐만 아니라 8, 9번 째 위치에 오류를더하는 결과를 갖게 된다. 본 논문에서는 위와 같이 하나의 신드롬 벡터에 두 개 이상의 2-오류 패턴이 존재하는 경우 이를 해결하여 오류 정정 능력을 개선시킬 수 있는 복호 방식을 제안한다.

3. 소프트 값을 이용한 LCPC 부호의 복호

LCPC 복호기에 입력되는 수신 부호 벡터 $\boldsymbol{r}$의 비트 값들은 복조기(demodulator)에 의해 복원된 아날로그 신호 값을 경 판정(hard decision) 과정을 통해 이진 값으로 변환하여 얻는다. 경 판정 과정에서 많은 채널 정보가 사라지게 되며 이는 복호기의 성능 열화로 이어진다.

이를 보완하기 위해 터보 부호나 LDPC 부호의 복호에서는 연 판정(soft decision) 값을 이용하여 복호 성능을 개선하고 있다[8]. 하지만 블록 부호의 패리티 체크 부호의 경우에는 신드롬 계산이나 복호표를 이용한 오류 정정 과정이 경 판정 값을 기초로 이루어지므로 연 판정 값을 복호 과정에 사용할 수 없다. 더욱이 터보 부호나 LDPC 부호에서 연 판정 값을 이용하여 복호 연산을 하는 것은 과도한 연산 능력이 요구되기 때문에 에너지가 제한되어 있고 연산 능력이 부족한 IoT 기기의 통신에 적합하지 않다.

본 절에서는 복조기의 아날로그 출력 값(이후 소프트 값이라고 부른다)에 대해 경 판정을 한 후 이를 버리지 않고 복호기(decoder)에 수신 부호 벡터 $\boldsymbol{r}$과 함께 전해 주어 복호 성능을 개선할 수 있는 복호 알고리즘을 설명한다. 편의 상 n 개의 소프트 값으로 이루어진 벡터를 $\boldsymbol{q} = (q_1, q_2, \cdots, q_n)$으로 표시한다.

그림 1은 $E_b/N_0 = 10[dB]$의 AWGN 채널에서 BPSK 시스템의 복조기 출력 중 오류 비트와 올바른 비트에 해당하는 소프트 값의 전력 분포를 나타낸 것이다. 그림에서 오류 비트에 해당하는 소프트 값의 전력은 거의 0에 근접한 값을 보이며 올바른 비트에 해당하는 소프트 값의 전력은 오류 비트의 소프트 값 전력에 비해 월등히 높은 값을 가진다. 이는 오류 비트에 해당하는 소프트 값의 경우 임계 치(BPSK의 경우 0)를 넘어야 오류가 되기 때문에 전력 값이 작을 확률이 크다. 따라서 소프트 값의 전력은 해당 비트가 오류 비트인지 올바른 비트인지 구분할 수 있는 보조 정보를 담고 있으며, LCPC 부호의 복호에 이용할 수 있다.

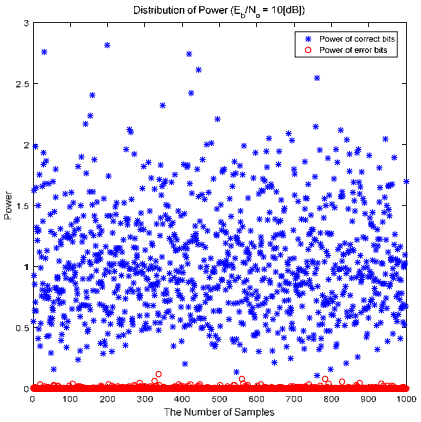

그림 1. 오류 비트와 올바른 비트에 해당하는 소프트 값의 전력 분포 ($E_b/N_0 = 10[dB]$)

Fig. 1. Power distribution of soft values for error bits and correct bits($E_b/N_0 = 10[dB]$)

예를 들어, 신드롬 $\boldsymbol{s} = [00011]$에 해당하는 오류 패턴 $\hat{\boldsymbol{e}}$는 $\boldsymbol{e_1} = [000000011]$과 $\boldsymbol{e_2} = [110000000]$의 두 가지 패턴이 있는데 식 (7)과 같이 소프트 값의 전력을 활용하면 더 정확한 오류 패턴을 추정할 수 있다.

$$\hat{\boldsymbol{e}} = \begin{cases} \boldsymbol{e_1}, & q_8^2 + q_9^2 < q_1^2 + q_2^2 \\ \boldsymbol{e_2}, & otherwise \end{cases} \tag{7}$$

신드롬 벡터 $\boldsymbol{s}$에 해당하는 2-오류 벡터가 3 개 이상 존재하여도 소프트 값의 전력이 가장 작은 오류 패턴을 선정하면 된다.

식 (7)의 계산은 하나의 신드롬 벡터에 2 개 이상의 2-오류 패턴이 존재하는 경우에만 실행되며 계산 량도 극히 적으므로 이로 인해 증가되는 복잡도는 거의 없다고 볼 수 있다. LCPC 부호의 부호 길이(n)가 작으므로 소프트 값을 저장해야 하는 메모리 용량도 n 개 이하이므로 복잡도에 미치는 영향은 거의 없다고 할 수 있다.

4. 시뮬레이션 및 결과 분석

3 장에서 제안한 복호 알고리즘의 BER 성능 개선을 확인하기 위해 다음과 같이 시뮬레이션을 실행하였다. BPSK 변조 방식과 함께 AWGN 채널을 가정하였으며, [2]에서 제안한 LCPC 부호 중 (9,4) LCPC 부호와 (8,3) LCPC 부호를 선정하여 [2]의 복호 방식과 본 논문에서 제안

한 복호 방식의 BER 성능을 비교・분석하였다.

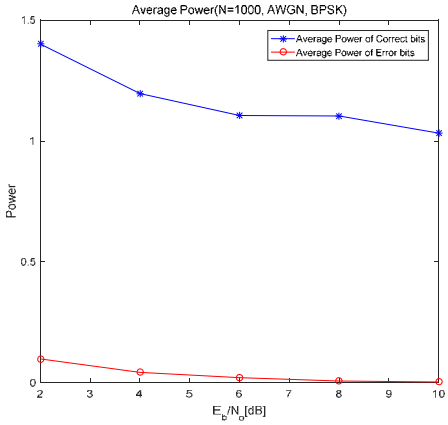

그림 2. SNR에 따른 오류 비트와 올바른 비트에 해당하는 소프트 값의 평균 전력 변화
Fig. 2. Average power of soft values for error bits and correct bits vs. SNR

그림 2는 SNR(E_b/N_0) 변화에 따른 오류 비트와 올바른 비트의 소프트 값의 평균 전력 변화를 나타낸 것이다. 전 영역에 걸쳐 두 평균 전력 값 차이는 줄어들지 않음을 볼 수 있다. SNR이 증가할수록 오류 비트에 해당하는 소프트 값의 전력은 0을 향해 수렴하며 올바른 비트에 해당하는 소프트 값의 전력은 1에 수렴함을 알 수 있다.

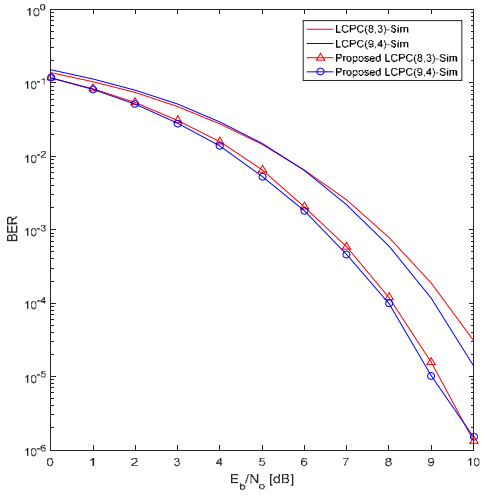

그림 3. LCPC 부호에 대한 제안된 복호 방식의 BER 성능 (AWGN, BPSK)
Fig. 3. BER performance of the proposed decoding scheme for LCPC codes (AWGN, BPSK)

그림 3은 시뮬레이션을 통해 얻은 (9,4) LCPC, (8,3) LCPC 부호에 대한 기존 복호 방식과 본 논문에서 제안한 복호 방식의 BER 성능을 나타낸 것이다. 그림 3에서 (9, 4) LCPC 부호와 (8,3) LCPC 부호에 관계없이 본 논문에서 제안한 복호 방식은 기존 복호 방식에 비해 10^{-4}의 비트 오율에서 약 $1.1[dB] \sim 1.2[dB]$의 성능 개선을 확인할 수 있다. 나머지 LCPC 부호의 경우에도 복호 방식이 동일하므로 동일한 결과를 예측할 수 있으며 이는 복잡도나 메모리의 증가가 거의 없이 얻은 것이므로 향 후 낮은 복잡도와 복호 지연이 없어야 하는 IoT 기기의 오류 정정 부호 방식으로 유용할 것으로 판단된다.

5. 결론

본 논문에서는 복조기의 출력 소프트 값을 활용하여 LCPC 부호의 BER 성능을 개선할 수 있는 복호 방식을 제안하였다. 특히 연산량이나 메모리 용량의 증가 없이 BER 성능 개선을 얻을 수 있으므로 에너지와 연산 자원이 작은 IoT 부호의 오류 정정 부호로 활용될 수 있을 것으로 생각된다.

성능 개선을 확인하기 위해 AWGN, BPSK 변조 방식 환경에서 (9,4) LCPC 부호와 (8,3) LCPC 부호에 대해 시뮬레이션을 실행하였으며, 실행 결과 LCPC 부호의 종류에 관계없이 기존 복호 방식에 비해 제안한 복호 방식은 약 $1.1[dB] \sim 1.2[dB]$의 성능 개선을 확인할 수 있었다. 향 후 IoT 기기 간 통신에 적용하기 위해서는 더 많은 패리티 체크 부호에 대한 실험 및 분석과 성능 개선에 대한 이론적 체계 확립이 필요하다.

REFERENCES

[1] E. Tsimbalo, X. Fafoutis, and R. J. Piechocki, "CRC error correction in IoT applications," IEEE Trans. Ind. Informat., vol. 13, no. 1, pp. 361-369, Feb. 2017
[2] Salah Abdulghani Alabady, and Fadi Al-Turjman,"Low Complexity Parity Check Code for Futuristic Wireless Networks Applications," IEEE Access, Vol. 6, 2018, pp. 18398-18407.
[3] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, second edition, Prentice Hall: Englewood

Cliffs, NJ, 2004.
[4] Sheryl L. Howard, Christian Schlegel, and Kris Iniewski, "Error Control Coding in Low-Power Wireless Sensor Networks: When is ECC Energy-Efficient?,"EURASIP Journal on Wireless Communication and Networking, Volume 2006, pp. 1-14.
[5] Z. Su, Q. Qiu, and H. Zhou, ''Analysis and elimination of short cycles in LDPC convolutional codes,'' in Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC), Chengdu, China, Oct. 2016, pp. 1128-1132.
[6] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, ''Short turbo codes over high order fields,'' IEEE Trans. Commun., vol. 61, no. 6, pp. 2201-2211, Jun. 2013.
[7] C.-Y. Chen, Q. Huang, C.-C. Chao, and S. Lin, ''Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes,'' IEEE Trans. Commun., vol. 58, no. 11, pp. 3140-3147, Nov. 2010.
[8] N. Miladinovic and M. P. C. Fossorier, ''Improved bit-flipping decoding of low-density parity-check codes,'' IEEE Trans. Inf. Theory, vol. 51, no. 4, pp. 1594-1606, Apr. 2005.

저자약력

정 호 영(Ho-Young Cheong) **[종신회원]**

- 1987년 8월 : 연세대학교 대학원 전자공학과 (공학석사)
- 1995년 2월 : 연세대학교 대학원 전자공학과 (공학박사)
- 1995년 4월 ~ 현재 : 남서울대학교 정보통신공학과 교수

<관심분야> 채널부호, 분자통신