논문 2008-45TC-2-13

LDPC 부호를 위한 행 분할 알고리즘

(Row-splitting Algorithm for Low Density Parity Check Codes)

정 만 호*, 이 종 훈**, 김 수 영**, 송 상 섭**

(Manho Jung, Jonghoon Lee, Sooyoung Kim, and Sangseob Song)

요 약

실용적은 통신 시스템은 고정된 한 부호화율이 아닌 다양한 부호화율에서 동작을 할 필요가 있다. 본 논문에서는 서로 다른 다양한 부호화율을 위한 LDPC(Low-Density Parity-Check) 부호를 분석했다. 한 특정한 모부호에서 행을 분할하여 LDPC 부호가 서로 다른 부호화율에서도 동작하도록 하는데 이 기술의 장점은 부호화율이 변하더라도 LDPC 부호의 블록 길이가 항상 일정하게 유지됨으로써 천공(Puncturing) 이나 쇼트닝(Shortening) 기법처럼 블록 길이가 줄어드는 단점을 보완한다는 것이다. 행 분할(Row-splitting) 기법이라 명명한 이 기술은 말 그대로 H 행렬의 행을 분할하여 부호화율을 낮추기 때문에 비슷한 행 합산(Row-combining) 기법의 단점 또한 보완할 수 있다.

Abstract

Practical communication systems need to operate at various different rates. This paper describes and analyzes low-density parity check codes for various different rates. From a specific mother code, it allows LDPC codes for different rate. The advantage of this technique is that each different rate LDPC codes have a same block length as mother code though the rate changes so it can make up for the weak points of puncturing and shortening which reduce their block length as the rate changes. Row-splitting method is to split the row, so that the rate changes from a higher rate to lower rate and cause of its own property, it can overcome the defect of row-combining method.

Keywords : Row-splitting, LDPC codes, Block length

Ⅰ. 서 론

실제 통신 시스템은 서로 다른 여러 부호화율에서 동작할 수 있어야 한다. 하나의 하드웨어가 전송된 신호를 거의 모든 부호화율에서 복호를 할 수 있어야 하는데 이러한 방법들로는 천공(Puncturing)과 쇼트닝(Shortening) 이라는 방법이 있다.[1] 천공의 경우 부호화율을 증가시킬 수 있는데 이는 부호의 블록 길이를 줄여주고 이는 성능의 저하로 이어진다. LDPC 부호는 부호화율이 높아질수록 성능이 하락하기 때문에 특히 부호화율이 높은 부호의 경우 성능의 하락은 매우 중요한 사항이 된다.[2] 쇼트닝의 경우도 마찬가지이다. 부호화율을 낮추면서 블록 길이가 줄어들기 때문에 원래의 블록 길이를 가지고 있는 부호화 비교해 볼때 성능의 하락을 볼 수 있다.

위에서 언급한 바와 같이, 천공과 쇼트닝은 부호화율을 변화시킬 수 있기 때문에 단일 하드웨어에 사용이 가능하지만 블록 길이가 줄어듦에 의한 성능의 하락은 피할 수 없다. 그래서 만약, 블록 길이를 유지할 수 있다면 이러한 성능의 하락은 어느 정도 해결을 할 수 있을 것이라고 기대되어지는데, 행 합산(Row-combining) 기법[3]과 여기서 제시하는 행 분할(Row-splitting) 기법은 모 부호(mother code)와 동일한 블록 길이를 유지하기 때문이다.

이 논문에서는 행 합산 기법에 대해 간단히 짚어보고 행 분할 기법의 방법과 여러 가지 분할 방법을 소개하며 원래 부호와 성능을 서로 비교해 보도록 한다.

* 학생회원, ** 정회원, 전북대학교 전자정보공학부
(Dept. of Electronics and Information, Chonbuk National university)
접수일자: 2007년11월12일, 수정완료일: 2008년2월14일

Ⅱ. 본 론

1. 행 합산(Row-combining) 기법

논문 [3]에서는 모 LDPC 부호로써 부호화율이 1/2이고 블록 길이가 12 인 행렬을 예로 들고 있으며 그 행렬은 식 (1)과 같다.

$$H_{1/2}=\begin{bmatrix}111000000000\\100110000000\\000011100000\\000000111000\\000000001110\\001000000011\end{bmatrix} \quad (1)$$

위 행렬을 부호화율이 3/4인 행렬로 바꿔보자. 식 (2)에서 보여지듯이 부호화율이 1/2인 부호에서 행 합산 기법을 이용하서 3/4로 바꾼 것이다.

$$H_{3/4}=\begin{bmatrix}111000111000\\100110001110\\001011100011\end{bmatrix} \quad (2)$$

이 행렬의 첫 번째 행은 식 (1) 행렬의 첫 번째와 네 번째 행이 합산된 것이고 두 번째 행은 식 (1)의 두 번째와 다섯 번째 행이 합산된 것이고 세 번째 행은 식 (1)의 세 번째와 여섯 번째 행이 합산이 된 것이다. 여기서 우리가 주목할 점은 부호화율이 1/2인 H 행렬은 합산될 행끼리 비교해 봤을 때 같은 열에 '1'이 위치해 있지 않다는 점이다. 그림 1에서 보여 지듯이 행 합산을 할 때 행 합산을 하게 될 행끼리 비교 시 같은 열에 '1'이 위치하게 되면 이는 패리티의 정보를 바꾸기 때문에 원래 모 부호와는 전혀 다른 H 행렬이 생성이 되게 된다.

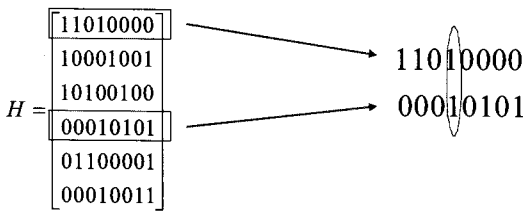


그림 1. 행 합산 기법이 잘못 시행된 예
Fig. 1. Wrong case when combining the row.

2. 행 분할(Row-splitting) 기법

행 분할 기법은 앞서 소개했던 행 합산 기법과 비슷한 방법이나 행을 분할하는 기법이기 때문에 행 합산 기법과는 반대되는 개념이다. 모 부호로부터 행을 분할하기 때문에 부호화율을 낮아지며 식 (3)에서 보여지는 부호화율이 1/2이고 블록 길이가 12인 모 부호를 예로 들어 좀 더 자세히 알아보자.

$$H_{6\times 12}=\begin{bmatrix}110100001101\\111110010000\\101011000110\\000111101010\\011000110011\\000001111101\end{bmatrix} \quad (3)$$

이 행렬을 이용해서 부호화율이 1/3인 부호로 바꾸고자 한다. 행 분할된 행렬은 식 (4)에 나와 있다.

$$H_{8\times 12}=\begin{bmatrix}110100000000\\111000000000\\101011000110\\000111101010\\011000110011\\000001111101\\000000001101\\000110010000\end{bmatrix} \quad (4)$$

식 (4)의 행렬의 일곱 번째와 여덟 번째의 행은 모 부호의 첫 번째와 두 번째의 행이 각각 분할되어 생성이 된 경우이다. 이후에 더 자세히 설명이 되겠지만 위에 든 예에서는 중심 분할 방법을 이용한 것으로 모 부호의 경우 각 행에 6개씩의 '1'이 존재하는데 이중 처음 3개의 '1'은 분할하고자 하는 행에 그대로 남아 있고 나머지 3개의 '1'은 7번째 새로이 생성된 행으로 이동이 된 경우이다.

행 분할 기법을 시행할 때, 모든 행을 다 분할할 필요는 없다. 위 예에서는 식 (3)의 첫 번째와 두 번째 행을 분할했지만 대신에 다섯 번째나 여섯 번째 행을 선택하여 분할을 해도 된다. 예상했듯이 이는 행을 분할하는 기법이기 때문에 1에서 소개했던 행 합산 기법과는 달리 분할할 행을 선택함에 있어서 조건이 없다는 것을 알 수 있다. 행을 분할하는 방법에는 여러 가지가 있다. 그중 본 논문에서는 3가지의 방법을 제안하고자 한다. 그 방법으로는 중심 분할 방법, 교대 분할 방법, 그리고 마지막으로 임의 분할 방법을 제시한다.

가. 중심 분할 방법(Center split)

중심 분할 방법은 분할하고자 하는 행을 절반으로 나누어 절반은 원래의 행에 그리고 나머지 절반은 새로 생성된 행으로 이동하는 방법이다. 여기서 절반으로 나눈다는 것은 각 행의 '1'의 개수를 기준으로 한다. 즉, 첫 번째부터 $w_r/2$ 번째 까지의 '1'은 원래의 행에 남겨두고 $w_r/2+1$ 번째부터 w_r 까지의 '1'은 새로 생성된 행으로 이동을 시켜준다. 여기서 w_r은 행에 대하여 '1'의

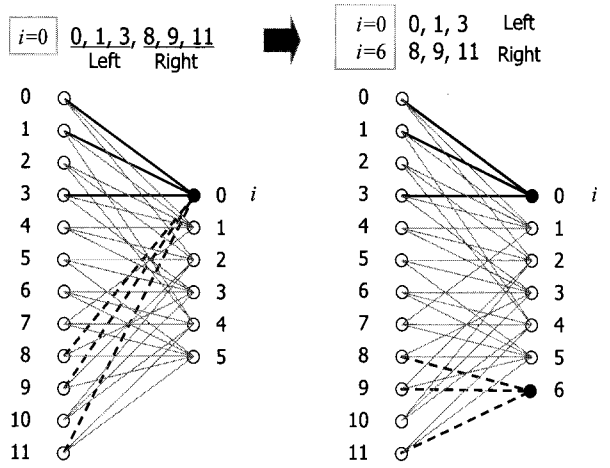


그림 2. 태너 그래프를 이용한 중심 분할 방법 과정
Fig. 2. Center split.

개수를 나타내 주는 수치이다.

다음에 보여 지는 그림 2 는 중심 분할 방법에 대해 설명하고 있다. 식 (3)을 태너 그래프(Tanner's graph)로 나타내었으며 i 는 행의 번호를 0부터 나타내었으며 그 옆에 보여 지는 숫자들은 Mackay의 LDPC 행렬 표현 방법에 의한 것이다. 즉, 0, 1, 3, 8, 9, 11 은 첫 번째(i=0)행에 총 6개의 '1' 이 있다는 의미이며 각각 '1'의 위치는 첫 번째, 두 번째, 네 번째, 아홉 번째, 열 번째, 열두 번째 열에 위치해 있다는 의미이다. 그림 2에서 보이듯이 첫 번째, 두 번째, 네 번째 열에 위치해 있는 '1' 들은 첫 행에 그대로 남아있게 되며 아홉 번째, 열 번째, 열두 번째 열에 위치해 있는 '1'들은 새로이 생성되는 7번째 행으로 이동이 되게 된다.

나. 교대 분할 방법(Alternate split)

교대 분할 방법은 분할하고자 하는 행의 '1'들의 위치가 짝수인가 홀수인가를 기준으로 나누는 방법이다. 즉, 해당 행에서 '1'들만을 봤을 때 그 순서가 짝수인가 또는 홀수 인가에 따라 해당 행에 남을 '1'과 새로이 생성된 행으로 이동할 '1'이 결정이 된다. '1'의 위치가 홀수인 경우 분할하고자 하는 행에 그대로 남겨두고 '1'의 위치가 짝수인 경우 새로이 생성된 행으로 이동을 해주는 방법이다. 물론 짝수인 '1'들을 남기고 홀수인 '1'들을 새로이 생성된 행으로 옮겨도 된다.

다음에 보여지는 그림 3은 교대 분할 방법을 태너 그래프와 Mackay 의 LDPC 행렬 표현 방법으로 나타낸 것이다. '홀수'로 구분되는 '1'들은 두 번째, 아홉 번째, 열두 번째 열에 위치한 '1'들로서(1,8,11) 해당 행에서 새로이 생성된 행으로 이동되어진 경우이다.

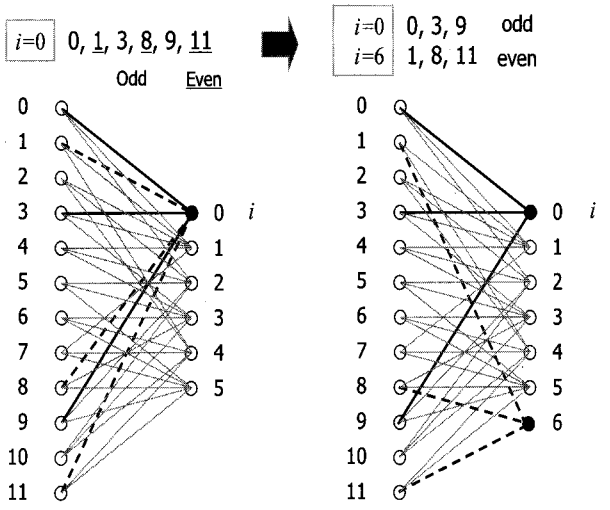


그림 3. 태너 그래프를 이용한 교대 분할 방법 과정
Fig. 3. Alternate split.

다. 임의 분할 방법(Random split)

임의 분할 방법은 분할하고자 하는 행의 '1'들을 무작위로 선택하여 선택된 '1'들은 분할하고자 하는 행에 남기고 선택되지 않은 '1'들은 새로이 생성된 행으로 옮기는 방법이다. 다음에 보여 지는 그림 4는 역시 마찬가지로 임의 분할 방법을 태너 그래프와 Mackay의 LDPC 행렬 표현 방법으로 나타낸 것이다. 여기서는 두 번째, 열 번째, 열두 번째 '1'이 임의로 선택이 된 경우이다.

행 분할 방법의 장점은 분할할 행을 선택함에 있어서 조건이 없다는 것이다. 그림 1에서 보여 졌듯이 행 합산의 경우 합산이 될 행끼리는 같인 열에 '1'이 위치해 있으면 안 된다는 조건이 있다. 만약 합산될 행끼리 같

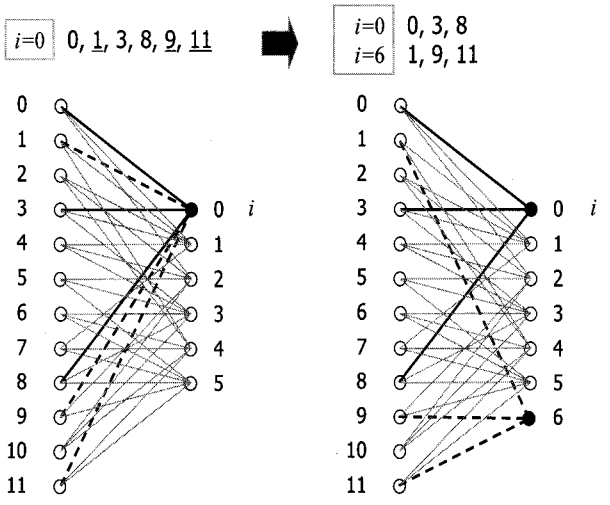


그림 4. 태너 그래프를 이용한 임의 분할 과정
Fig. 4. Random split.

은 열에 '1'이 위치하게 되면 합산의 과정에서 XOR 연산을 거치게 되는데 '1'끼리 XOR 연산이 이루어지게 되면서 패리티의 정보가 모 부호와는 다른 정보를 띄게 된다. 또한 알고리즘을 구현하면서 '1'의 위치를 비교할 다른 알고리즘이 필요하다는 단점도 있다.

Ⅲ. 실 험

그림 5는 AWGN 환경에서 보여지는 행 합산 기법으로 생성된 부호와 그에 따라 변한 부호화율을 원래의 부호화율로 지니고 있는 부호(Stand Alone;SA 라 명명)를 서로 비교하여 보여준 성능 분석 그래프이다.[3] 부호의 블록 길이는 1944로 했으며 최대 복호 반복 횟수는 50이다.

그림 6은 앞서 본문에서 소개했던 3가지의 행 분할 방법으로 생성된 부호의 성능을 서로 비교한 그래프이다. 모 부호는 블록길이가 1908이고 부호화율이 0.9인 부호로 설정했으며 각각 3가지의 방법으로 행을 분할하여 0.44의 부호화율을 생성, 서로의 성능을 비교한 그래프이다. 그래프에서 확인한 바와 같이 교대 분할 방법이나 임의 분할 방법보다 중심 분할 방법이 더 성능이 잘 나오는 것으로 볼 수 있다.

그림 6의 결과를 바탕으로 그림 7에서 SA 부호와 행 분할 기법(중심 분할)을 이용하여 생성된 부호를 서로 비교해 보았다. 채널 환경은 AWGN 이었으며 블록 길이는 1908, 최대 복호 반복 횟수는 50번으로 제한하였다.

성능의 비교 결과 행 분할 기법의 성능은 그 본래의 모부호 보다 낮음을 알 수 있다. 그에 대한 원인으로는 LDPC 부호의 경우 regular와 irregular 로 나누어지게 되는데 일반적으로 regular 의 성능이 더 좋게 나온다. regular LDPC 부호를 행 분할 할 경우 특정 행은 분할이 되지 않는 경우도 있어 irregular 로 변하게 되어 성능의 저하가 오게 된다.

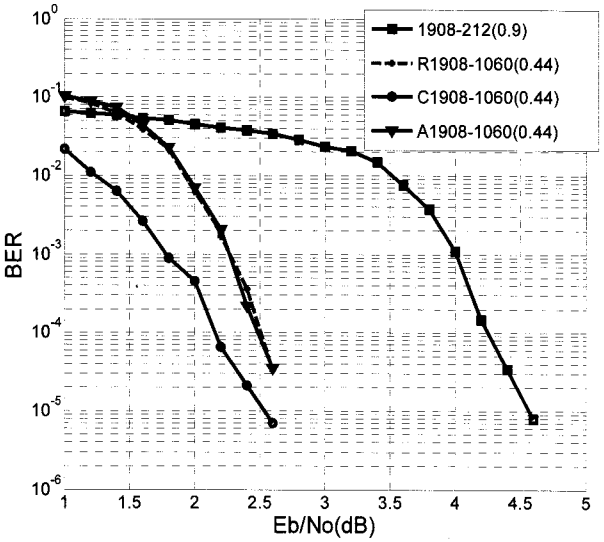


그림 6. 3가지 분할 방법들의 성능 비교
(R:임의분할, C:중심분할, A:교대분할)
Fig. 6. Performance comparison of 3 different ways of row-splitting.

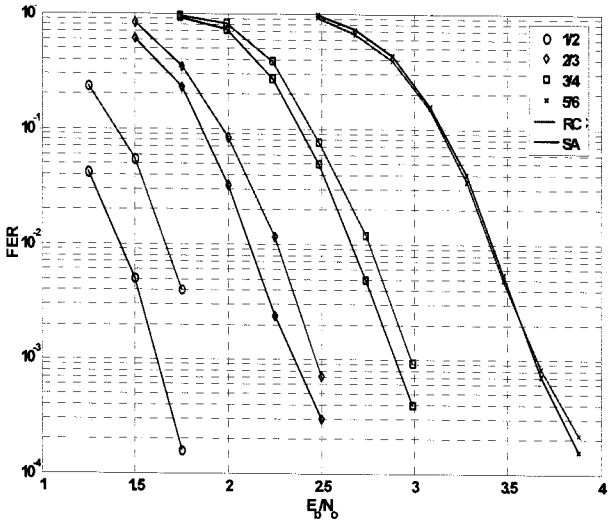


그림 5. 행 합산 기법과 SA 부호의 성능 비교
(블록길이 1944, 최대 복호 반복 횟수 50번)
Fig. 5. Performance of row-combining and stand alone codes.
(1994 block length, 50 iterations)

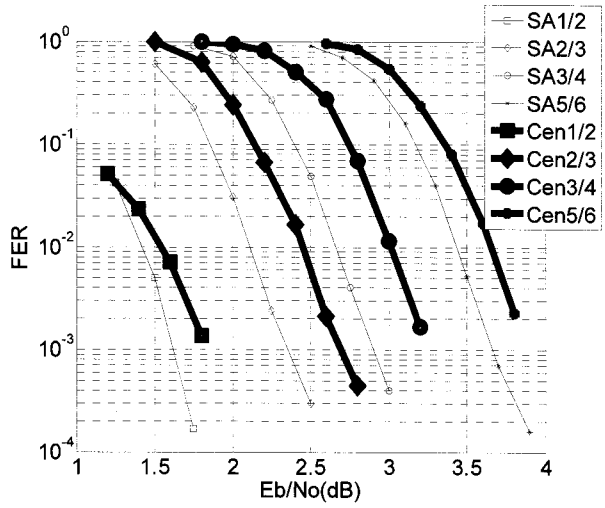


그림 7. 행 분할 기법과 SA 부호의 성능 비교
(블록길이 1908, 최대 복호 반복 횟수 50번)
Fig. 7. Performance of row-splitting and stand alone codes.
(block length 1908, 50 iterations)

Ⅳ. 결 론

논문 [2]와 [4]에서 알 수 있듯이 블록 부호는 블록 길이가 무한대로 갈수록 더 좋은 성능을 보여주게 된다. 부호화율을 변화시키는 부호가 실제 통신 시스템에는 천공과 쇼트닝의 형태로 적용이 되어 왔으나 이들은 이미 언급한대로 블록 길이가 줄어들기 때문에 모 부호보다 낮은 성능을 보여주고 있다.

본 논문에서 제안한 행 분할 기법은 블록 길이를 유지하면서 부호화율을 변화시킬 수 있다는 점으로 원래 그 부호화유을 가지고 있는 부호와 성능의 비슷할 것으로 예상했으나 행 분할을 거치면서 irregular 로 바뀌게 되어 성능의 저하가 있게 되었다. 그러나 여전히 블록 길이를 유지한다는 점은 블록 부호로서 큰 장점으로 자리 매김할 수 있으며 분할 방법에 있어서 좀 더 효율적인 알고리즘을 개발한다면 성능의 향상을 기대 할 수 있겠다.

참 고 문 헌

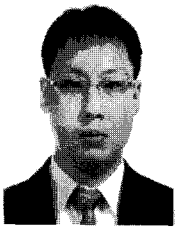
[1] Tao Tian and Christopher R. Jones, "Construction of Rate-compatible LDPC Codes Utilizing Information Shortening and Parity Puncturing", *EURASIP Journal on Wireless Communications and Networking 2005:5*, pages 779-795

[2] D.J.C. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low-Density Parity-Check Codes", *IEE Electronics Letters*, vol. 32, no. 18, pages 1645-1655, 29th Aug. 1996.

[3] A.I. Vila Casado, Stefano Valle, Wen-Yen Weng, and Richard D. Wesel, "Constant-Blocklength Multiple-Rate LDPC Codes for Analog-Decoding Implementations", *Proceedings Analog Decoding Workshop*, June 2006.

[4] Sergio Verdu, "Fifty Years of Shannon Theory", *IEEE Transactions on Information Theory*, Vol. 44, No. 6, October 1998.

저 자 소 개



정 만 호(학생회원)
2006년 전북대학교 전자공학과 학사졸업.
2008년 전북대학교 전자공학과 석사졸업예정
<주관심분야 : 채널 코딩, LDPC>

이 종 훈(정회원)
1996년 전북대학교 전자공학과 학사졸업.
1998년 전북대학교 전자공학과 석사졸업.
2006년 전북대학교 전자공학과 박사수료
<주관심분야 : 디지털 통신, OFDM>

김 수 영(정회원)
1992년 Surrey대학교 전자공학과 석사졸업.
1995년 Surrey대학교 전자공학과 박사졸업.
2008년 현재 전북대학교 전자공학과교수
<주관심분야 : 위성통신, 오류정정부호>

송 상 섭(정회원)
1980년 카이스트대학교 전자공학과 석사졸업.
1990년 Manitoba대학교 전자공학과 박사졸업.
2008년 현재 전북대학교 전자공학과교수
<주관심분야 : 채널 코딩, 오류정정부호, 무선랜>