이차원 코드를 위한 개선된 LDPC 코드

김현경° 정철호 한탁돈
연세대학교 컴퓨터과학과
lament°@yonsei.ac.kr, {bright, hantack}@kurene.yonsei.ac.kr

An Improved Low-Density Parity-Check Codes for Two-Dimensional Codes

Hyunkyung Kim° Cheolho Cheong Tack-Don Han
Dept. of Computer Science, Yonsei University

요 약

디지털 신호 및 전송부호의 오류검출에는 예전부터 패리티 체크가 사용되어 왔다. 그러나 패리티 체크 기법은 구현 및 알고리즘이 단순, 간결한 우수성이 있지만 특정 데이터 비트의 경우 오류 검출이 불가능하다는 문제점을 가지고 있다. 이후 패리티 체크 기법은 해밍 코드 및 채널 오류 정정을 위한 LDPC 코드와 같은 다양한 오류검출 및 정정 알고리즘에 적용되어 발전되어 왔으며, 그 중 LDPC 코드의 *bit-flipping* 알고리즘에서는 패리티 기법을 반복적으로 적용하는 방식을 택하고 있다. 본 논문에서는 이러한 채널 오류 정정을 위한 LDPC의 bit-flipping 알고리즘을 이차원 코드에 적용하고, 이 때 bit-flipping 알고리즘이 가지고 있는 문제점을 보완할 수 있는 개선된 LDPC 코드를 제안한다.

1. 서 론

디지털 부호의 기록 및 전송에 있어서는 외부로부터의 영향(잡음, 끊김 등)에 의해 데이터 오류가 발생할 위험이 있다. 오류가 발생한 경우 기록, 전송계에서는 일반적인 통신환경과는 달리 데이터의 재 전송을 요청할 수 없기 때문에 수신계에서 오류의 검출 및 정정을 하지 않으면 안 된다. 이와 같이 수신측에서 수신된 데이터에 대한 오류정정을 가능하게 하는 방법을 FEC (Forward Error Correction) 방법이라 한다[1].

오류 검출 기법 중 가장 간단한 방법은 단일 패리티 체크 코드[2]이다. 단일 패리티 체크 코드는 이진 데이터 부호에 1비트의 체크 비트를 부가하여 데이터 부호에 포함되는 1의 개수를 우수 또는 기수가 되도록 하여 전송하는 방법이다.

이러한 패리티 체크 코드는 이후 해밍 코드 및 LDPC (Low-Density Parity-Check Codes)[1][3][4][5]와 같은 오류 정정 기법에 다양하게 활용되고 있다. 그 중 LDPC 코드를 생성하는 가장 간단한 방법인 *bit-flipping* 알고리즘에서는 단일 패리티 체크 코드를 반복적으로 적용함으로써 손쉽게 오류 검출 및 정정이 가능하다.

그러나 이와 같은 LDPC의 bit-flipping 알고리즘 역시 특정 경우에 무한반복이 됨으로써 오류 검출 및 정정이 불가능한 경우가 발생할 수 있다. 따라서 본 논문에서는 이러한 bit-flipping 알고리즘에서 발생할 수 있는 무한반복의 문제점을 해결하여 손쉽게 오류 검출 및 정정이 가능한 개선된 LDPC 코드를 제안하고자 한다.

2. LDPC(Low-Density Parity-Check) 코드

LDPC 코드는 채널 데이터 전송 시 발생할 수 있는 오류 검출 및 정정 기법으로써 1962년에 Gallager에 의해 처음 제안되었다. LDPC(Low-Density Parity-Check) 라는 말에서 알 수 있듯이 LDPC 코드는 패리티 검사 행렬이 거의 0으로 구성되어 있고 1은 매우 적은 수로만 구성되어 있는 코드이다. (n, j, k) low-density 코드는 전체 코드 블럭의 길이가 n이며, 정보부호의 길이가 j, 체크부호의 길이가 k 비트로 구성된 코드 이다[3].

LDPC 코드에서 중요한 개념으로는 *Tanner graph*[2][3][5]라는 것이 있는데, 이것은 LDPC 코드의 반복적인 디코딩 알고리즘을 결정하는 그래프적인 표현이다. (7,4,3) LDPC 코드에 대한 Tanner graph는 <그림 1>과 같다.

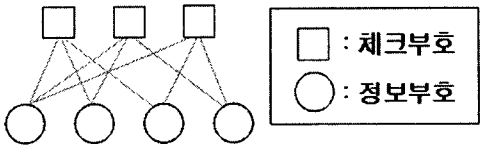

그림 1 . Tanner graph

이와 같은 LDPC 코드를 구성하는 대표적인 알고리즘에는 bit-flipping 알고리즘과 이러한 bit-flipping 알고리즘의 문제점을 개선하기 위한 *soft decision* 기법 중 하나인 *sum-product* 알고리즘[3][4][5]이 있다. 이 중 bit-flipping 알고리즘은 LDPC 코드를 구성하는 가장 간단한 알고리즘으로써 Tanner graph를 이용하여 패리티 체크 연산을 반복적으로 수행하는 방법이다. 또한 bit-flipping 알고리즘은 패리티 체크 코드를 기본으로 하고 있으므로 알고리즘 내의 모든 연산은 모듈러 연산(*modular-2 addition*)으로만 이루어져 있다. 따라서 다른 알고리즘에 비해 계산이 간단하고 알고리즘의 복잡도가 낮다는 우수성을 지니고 있다.

그러나 이와 같은 bit-flipping 알고리즘은 다음과 같

은 문제점을 지니고 있다.

2.1 기존 알고리즘의 문제점

<그림 2>는 bit-flipping 알고리즘을 기반으로 한 LDPC 코드에서의 문제점을 나타내 주는 한 예제이며, 식(1)은 (7,4,3) LDPC 코드에서 체크부호 값을 구하는 계산식이다.

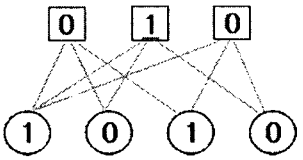

그림 2. 기존 LDPC 코드의 문제 예

$$\begin{aligned} x_5 &= x_1 \oplus x_2 \oplus x_3 \\ x_6 &= x_1 \oplus x_2 \oplus x_4 \\ x_7 &= x_1 \oplus x_3 \oplus x_4 \end{aligned} \qquad (1)$$

bit-flipping 알고리즘은 <그림 2>와 같이 정보비트 노드에 연결되어 있는 체크비트 노드에 각 정보비트 노드의 부호값에 대한 모듈러 연산을 수행하여 체크비트 노드에 전달한다. 그런 후 체크비트 노드에서는 정보비트 노드로부터 전달된 값을 이용하여 체크부호의 값을 계산하고, 이 때 계산된 패리티 결과 값에 오류가 없으면 알고리즘을 종료한다. 만일 그렇지 않을 경우(not satisfied) 체크비트 노드 중 fail이 된 노드에 연결되어 있는 정보부호의 값을 flipping(0→1/1→0 로 변경)한 후 위의 과정을 반복한다.

그런데 이 때, <그림 2>와 같이 특정한 정보비트로 구성되어 있는 코드의 경우 체크비트 노드의 값을 기준으로 했을 때 flipping 작업을 수행할 수 없는 경우가 발생할 수 있다. 그러한 이유는 위의 bit-flipping 알고리즘에서는 체크부호 값과 식(1)을 이용하여 정보부호의 값을 flipping 시키게 되는데, 위와 같이 x_6 하나의 체크부호에서만 오류가 검출 되었을 경우 어떤 정보부호 값을 flipping 시킬지 결정할 수 없기 때문이다. 즉, (7,4,3) LDPC 코드에서는 2개 이상의 체크부호에서 오류가 검출 되어야만 flipping 시킬 정보부호 비트 노드를 결정할 수 있게 되며, 그렇지 않을 경우 flipping을 할 수 없게 되는 것이다.

바로 이러한 이유로 인해 일반적인 LDPC 코드에서는 bit-flipping 알고리즘 보다는 이를 개선한 sum-product 알고리즘을 보다 많이 사용하고 있는 것이다.

3. 개선된 LDPC 코드

앞에서 살펴본 LDPC 코드의 bit-flipping 알고리즘의 문제점을 해결하고자 본 논문에서는 다음과 같은 사실을 전제로 한다.

- 본 논문에서 사용할 LDPC 코드는 (7,4,3) 코드를 전제로 한다.
- 패리티 계산을 위해 사용될 정보부호와 체크부호는 다음과 같이 이루어진다.

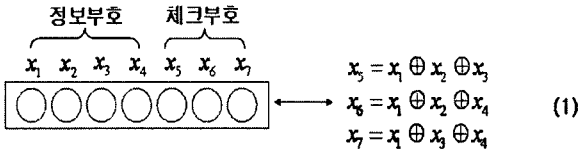

그림 4. 패리티 체크 계산 매트릭스

3.1 bit-flipping 알고리즘의 문제점 해결 방안

LDPC 코드의 bit-flipping 알고리즘에서 <그림 2>와 같이 하나의 체크부호에서만 오류가 발생하였을 경우에는 식(1)과 같은 패리티 체크 연산만으로는 위와 같은 문제를 해결할 수 없다는 것을 보였다. 따라서 이러한 경우에는 다른 방식의 처리가 필요하다.

본 논문에서는 위와 같은 문제가 발생하였을 경우 해당 체크비트 노드에 연결되어 있지 않은 다른 정보비트를 flipping 하는 방법을 제안하고자 한다. 즉, 체크비트 x_6에서 오류가 검출 되었을 경우 기존의 bit-flipping 알고리즘에서는 x_6에 연결되어 있는 정보부호(x_1, x_2, x_3)중 하나를 선택하여 flipping 시켰지만, 본 논문에서 제안하는 방법에 따르면 x_6에 연결되어 있지 않은 x_3을 flipping 시키는 것이다. 그런 이후에 변경된 정보부호 값을 기준으로 하여 새로운 체크부호 값을 얻어내고 그 이후에는 앞에서 말한 과정을 반복해서 수행하면 된다. 만약 둘 이상의 체크부호에서 오류가 검출되었을 경우에는 일반적인 bit-flipping 알고리즘의 처리방식과 동일하게 처리하면 된다.

<그림 5>는 제안된 알고리즘에 의해 <그림 2>의 문제를 해결한 과정을 나타낸 그림이다.

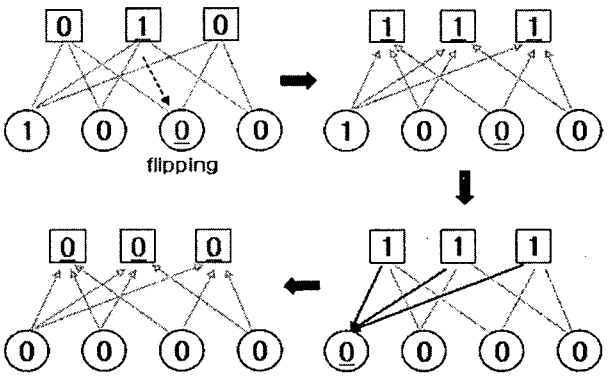

그림 5. 기존 알고리즘의 문제 해결 과정

3.2 개선된 LDPC 코드 알고리즘

위에서 설명한 개선된 LDPC 코드의 전체적인 디코딩 과정을 정리하면 다음과 같다.

Step 1. *Initialization*: 전송된 각각의 정보부호 값이 각

비트 노드에 할당되고, 이 값을 가리키고 있는 체크부호에게 모듈러 연산(modular-2 addition)을 한 결과 값을 보낸다. 이 때 각 정보 비트 노드 및 체크비트 노드에는 2비트씩 할당된다.

Step 2. *Parity update*: 정보 비트 노드로부터 전송된 메시지를 식(1)을 이용하여 체크부호 값이 만족하는지 여부를 계산하기 위해 각각의 체크 비트 노드에 들어갈 값을 계산한다. 만약 패리티 체크 값이 만족되면 알고리즘이 완료되고, 만족하지 않을 경우 식(1)을 이용하여 체크 비트 노드에 연결된 정보를 이용하여 정보 비트 노드에게 계산 결과에 대한 정보를 전송한다. 여기서 하나의 체크 비트 노드만이 값이 만족되지 않을 경우에는 Step 3으로 가고, 그렇지 않을 경우 Step 4로 간다.

Step 3. *Revised Parity update*: 자신의 체크 비트 노드에 걸려 있지 않은 정보 비트 부호에게만 정보를 전송한다.

Step 4. *Bit update*: Step 2 또는 Step 3으로부터 전송된 값이 "not satisfied"이면 해당 정보 비트 노드의 비트 값을 flipping 시키고 그렇지 않으면 알고리즘을 완료 시킨다. flipping 시킨 이후에는 결과 정보를 해당 정보 비트 노드에 연결된 체크 비트 노드에 전달시키고 알고리즘이 완료될 때까지 Step 1부터 반복한다.

3.3 제안된 방법의 확장방안

본 논문에서 제안하고 있는 방법은 (7,4,3) LDPC 코드를 기반으로 하고 있다. 또한 각 정보비트 노드 및 체크비트 노드에는 각 1비트씩의 데이터가 할당된다. 그러나 이와 같은 비트 노드의 구성방식은 이차원 코드와 같이 데이터 량이 증가할 경우 최적화된 성능을 발휘하지 못할 가능성이 있다. 따라서 이와 같은 경우에는 <그림 6>과 같이 각 정보비트 노드 및 체크비트 노드에 $2n$개의 비트를 할당하는 방법을 적용할 수 있다.

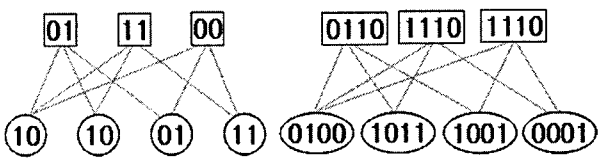

그림 6. 비트 노드 구성의 확장방안

이와 같이 구성을 하면 데이터 량이 늘어나는 경우에도 비트노드의 수의 증가 없이 패리티 연산이 가능하게 된다. 각각의 비트노드에 대한 정보부호의 패리티 연산은 왼쪽과 오른쪽을 각각 1비트씩, 혹은 2비트씩 묶어서 처리하게 되면 기존의 알고리즘을 그대로 적용할 수 있게 된다.

4. 결론 및 향후 과제

본 논문에서는 LDPC 코드를 구성하는 방법 중 bit-flipping 알고리즘이 가지는 기본적인 문제점인 특정 데이터 코드에 대해 오류 검출 및 정정을 하지 못하는 현상을 해결하기 위해 개선된 LDPC 알고리즘을 제안하고 있다. 이를 이용하면 단순한 패리티 연산만으로도 LDPC 코드 구성이 가능하며, Reed-Solomon 코드와 같은 다른 오류 정정 알고리즘에 비해 복잡도를 현저히 줄일 수 있다. 또한 각각의 비트 노드를 $2n$비트씩 구성함으로써 데이터 량이 증가할 경우에도 제안된 알고리즘을 그대로 적용할 수 있을 것으로 예상된다.

그러나 제안하는 방법은 2개 이상의 비트에서 동시에 에러가 발생할 경우 오류 검출 및 정정이 불가능하다는 단점을 가지고 있으며, 향후에는 이러한 문제를 해결할 수 있는 방안에 대한 연구가 필요하다.

감사의 글

본 연구는 BK 사업단 및 특정기초 연구사업의 지원에 의한 것임.

참고문헌

[1] MacWilliams, Florence Jessie, "The Theory of Error-Correcting Codes", *North-Holland*, 1997.

[2] D.Dlepian, "A class of binary signalling alphabets", *Bell Sys. Tech. J.*, vol. 35, pp. 203-234, January, 1956.

[3] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Trans. Inform. Theory*, vol. IT-8, no. 1, pp. 21-28, January 1962.

[4] Sarah J. Johnson, Steven R. Weller, "Low-density parity-check codes: Design and decoding", Technical Report EE02041, University of Newcastle, June 27, 2002.

[5] Amin Shokrollahi, "LDPC Codes: An Introduction", Digital Fountain, Inc., April 2, 2003.

[6] Pishro-Nik, H., Fekri, F., "On decoding of low-density parity-check codes over the binary erasure channel", *Information Theory, IEEE Transactions on*, vol. 50, Issue 3, pp. 439-454, March 2004.