단일 오류정정부호의 설계와 검사어 압축†

조성진*, 황윤희**
*부경대학교 수리과학부
**부경대학교 정보보호협동과정
e-mail:sjcho@pknu.ac.kr

Design and Reduction of Check Bits of Single Error Correcting Code †

Sung-Jin Cho*, Yoon-Hee Hwang**
*Division of Mathematical Sciences, Pukyong National Univ.
**Dept. of Information Security, Pukyong National Univ.

요 약

현대사회에서 정보 전달의 중요성이 강조되면서 부호이론에 대한 연구가 빠른 속도로 진척되었다. 따라서 잡음이 있는 통신로를 통하여 정보를 전송하고자 할 때 발생하는 오류를 정정하는 오류정정부호 장치가 필요하게 되었다. 본 논문에서는 특별한 행렬을 이용하여 보내고자 하는 정보를 단일 오류정정 부호로 부호화, 복호화하고, 또 부호화과정에서 검사어를 압축하는 방법을 고안한 것이다.

1. 서론

오늘날 많은 양의 데이터가 다양한 컴퓨터 시스템과 서브시스템 사이에서 디지털 논리 회로와 상호 연결선을 통하여 전송된다. 시스템의 신뢰성은 회로 모듈 사이에서 데이터 전송의 무오류성에 의존한다. 하지만 전자적 잡음이나 장치 결함, 시간 오류 등으로 인하여 시스템에는 언제 발생할지 모르는 잠재적 오류가 항상 존재한다[1][2]. 따라서 시스템의 신뢰성을 향상시키기 위해서 잡음이 있는 통신로를 통하여 데이터를 전송할 때 발생하는 오류를 정정하는 오류정정부호 장치가 필요하게 되었다[3][4]. 본 논문의 구성은 다음과 같다. 2장에서는 특별한 행렬을 구성하여 보내고자 하는 데이터의 부호화와 전송된 부호어를 복호화하는 과정을 제안한다. 3장에서는 생성된 부호어의 검사어를 압축하는 방법을 제안한다. 끝으로 4장에서 결론을 맺는다.

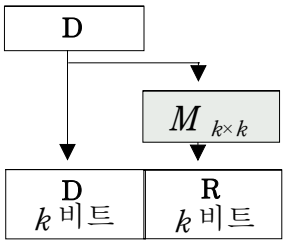
† 본 연구는 한국과학재단 특정기초연구(R01-2003-000-10663 -0)지원으로 수행되었음.

2. 부호화와 복호화

가. 부호화

기본 개념은 k 비트의 데이터(D)가 초기값으로 로드되면 주어진 행렬을 이용하여 검사어(R)를 생성하고 이를 데이터에 연결하여 부호어(C)를 생성한다.

$$R = M \cdot D^t$$
$$C = [D \mid R^t]$$



<그림1> 부호화

정리1> 체 F_2 위에서 선형 (n, k) 부호 C 의 최소거리 즉 최소무게가 d 이기 위한 필요충분조건은 C 의 패리티 검사행렬 H 에서 서로 다른 $d-1$ 개 이하의 열벡터가 독립이다. □

따름정리2> 행렬 M이 다음 두 조건을 만족한다면, M을 이용한 부호화는 부호의 길이가 $2k$, 데이터 길이가 k이고 최소거리가 3인 단일 오류정정부호인 $(2k, k, 3)$ 부호를 생성한다.

① 모든 열벡터는 적어도 두 개의 1을 포함한다.

② 임의의 서로 다른 두 개의 열벡터는 적어도 하나의 위치에서 다르다. □

정리3> k차 정방행렬 M을 다음과 같이 구성하면 $(2k, k, 3)$ 부호를 생성한다.

$$M_{ij} = \begin{cases} 1 & (i = j, \text{ 또는 } i = j-1) \\ 1 & (i = k\text{이고}, j = 1\text{또는}, \\ & \ i = k\text{이고}, j = k-1) \\ 0 & (\text{나머지성분들}) \end{cases}$$

□

예제4> $(16, 8, 3)$ 부호를 생성해 보자. 행렬 M은 정리3에 의해서 다음과 같이 구성한다.

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

데이터 $D = (1, 1, 1, 1, 0, 0, 0, 0)$을 보내고자 할 때 검사어는 다음과 같이 생성된다.

$$R = M \cdot D^t = (0, 0, 0, 1, 0, 0, 0, 1)^t$$

따라서 부호어는 $C = (1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1)$ 이다.

나. 복호화

잡음이 있는 통신로를 통하여 부호어가 수신되면 전송 도중 오류가 발생할 수 있다. 이렇게 수신된 부호어에서 발생한 오류를 정정하기 위해서 먼저 수신된 부호어 ($W = [D' \mid R'^t]$) 의 신드롬을 계산하여야 한다[3][4].

$$S(W) = H \cdot W^t$$

여기서 H는 부호화 과정에 의해 행렬 M과 k차 단위행렬을 연결하여 만든 패리티 검사행렬이다. 즉, $H = [M \mid I_k]$ 이다. 오류벡터 (E) 를 다음과 같이 두면 $S(W)$ 는 다음과 같다.

$$E = [D_e \mid R_e],\ D' = D \oplus D_e, R' = R \oplus R_e$$

$$\begin{aligned} S(W) &= H \cdot [D' \mid R'^t]^t \\ &= [M \mid I_k] \cdot [D' \mid R'^t]^t \\ &= M \cdot D'^t \oplus I_k \cdot R' \\ &= M \cdot D^t \oplus M \cdot D_e{}^t \oplus R \oplus R_e \\ &= H \cdot E^t \end{aligned}$$

정리5> 수신된 부호어 $S(W)$의 값이 0이면 수신된 부호어 W에 오류가 발생하지 않은 것이고, 0이 아니면 W에 오류가 발생한 것이다. □

수신된 부호어 W의 오류벡터 E는 다음과 같다.

$$E^t = H^{-1} \cdot S(W)$$

하지만 패리티 검사행렬 H는 정방행렬이 아니므로 H^{-1}이 존재하지 않는다. 따라서 $k \times 2k$ 패리티 검사행렬 H를 $2k$차 정방행렬 M_{aug}로 변환하기 위해서 k개의 행을 추가로 덧붙인다.

$$M_{aug} = \begin{bmatrix} H \\ \\ \text{추가행} \end{bmatrix}$$

특히, M_{aug}가 역행렬이 존재하도록 구성하되 M_{aug}의 성분 중 1의 수를 최소로 하여 복호화 과정에서 계산량을 줄이도록 한다.

정리6> 체 F_2 위에서 블록행렬 B를 다음과 같이 구성한다.

$$B = \begin{bmatrix} M & I \\ I & O \end{bmatrix}$$

그러면 행렬 B은 역행렬이 존재하고 다음과 같다.

$$B^{-1} = \begin{bmatrix} O & I \\ I & M \end{bmatrix}$$

□

정리6에 의하여 행렬 M에 대한 M_{aug}를 다음과 같이 구성한다.

$$M_{aug} = \begin{bmatrix} M & I \\ I & O \end{bmatrix}$$

이 때 M_{aug}는 정방행렬이고 역행렬이 존재하며, 또한 1의 개수도 최소임을 알 수 있다.

수신된 부호어의 오류벡터 $E = [D_e | R_e{}^t]$ 에 대하여 $S_{aug}(W)$ 를 다음과 같이 정의한다.

$$M_{aug} \cdot E^t := \begin{bmatrix} S(W) \\ S_{aug}(W) \end{bmatrix}$$

위를 이용하여 오류벡터를 구하면 다음과 같다.

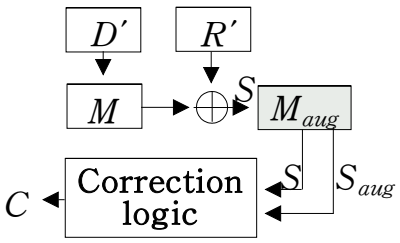
$$E^t = M_{aug}{}^{-1} \begin{bmatrix} S(W) \\ S_{aug}(W) \end{bmatrix}$$

이렇게 구한 오류벡터 E를 이용하여 수신된 부호어 W를 다음과 같이 복호한다.

$$C = W \oplus E$$

특히, M_{aug}를 M, I를 성분으로 하는 블록행렬로 구성함으로써 $M_{aug}{}^{-1}$을 구하기 간단하여 오류벡터를 보다 효율적으로 찾을 수 있어 복호하기 용이하다. 또한, $M_{aug}{}^{-1}$를 구하지 않고도 수신된 부호어 W에 단일 오류가 발생했을 경우, W의 신드롬 $S(W)$를 패리티 검사행렬 H를 이용하여 구하면 H의 i번째 열벡터가 된다. M_{aug}를 다음과 같이 구성하였으므로 $S_{aug}(W)$는 i번째 성분만 1이고 나머지 성분들은 0인 벡터이거나 모든 성분이 0인 벡터가 된다. 이 때, $S(W)$의 1인 성분의 수가 2개 이상이면 이는 데이터 부분에서 오류가 발생한 것이고 M_{aug}에 의해서 오류벡터는 $[S_{aug}(W)^t | 0]$으로 나타낼 수 있다. 또, $S(W)$의 1인 성분의 수가 1개이면 이는 검사어 부분에서 오류가 발생한 것이고 오류벡터는 $[0 | S(W)^t]$로 나타낼 수 있다.

이와 같이 M_{aug}를 위와 같이 기존의 방법과 다르게 구성함으로써 복호화 과정에서 수신된 부호어로부터 구한 $S(W)$와 이에 대응하는 $S_{aug}(W)$를 분석하여 기존의 방법보다 간단한 복호법을 제공한다. <그림 2> 은 제안된 방법을 보여준다.



<그림2> 복호화

<복호 알고리즘>

Step 1.수신된 부호어 $W = [D' | R'^t]$에 대하여 신드롬 $S(W)$를 계산한다.

$$S(W) = M \cdot D'^t \oplus R'$$

Step 2.(신드롬 벡터의 1의 개수) $= 0$ 이면,

$$E = 0$$

(신드롬 벡터의 1의 개수) ≥ 2 이면,

$$E = [S_{aug}(W) | 0]$$

(신드롬 벡터의 1의 개수) $= 1$ 이면,

$$E = [0 | S(W)^t]$$

Step 3. 수신된 부호어 C는 다음과 같이 복호한다.

$$C = W \oplus E$$

예제7> 예4에서 생성한 부호어 $C = (1,1,1,1,0,0,0,0,0,0,0,1,0,0,0,1)$가 전송 도중 세 번째 비트에서 하나의 오류가 발생했다고 하자. 즉 $W = (1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1)$가 수신되었다고 가정하자.

수신된 부호어 W를 복호하기 위하여 신드롬 $S(W)$를 구하면 다음과 같다.

$$S(W) = M \cdot D'^t \oplus R' = (0,1,1,0,0,0,0,0)$$

여기서 $S(W)$은 M의 세 번째 열벡터, M_{aug}의 세 번째 열의 앞 성분들을 나타낸다. 따라서 S_{aug}와 오류벡터 E는 다음과 같다.

$$S_{aug}(W) = (0,0,1,0,0,0,0,0)t$$

$$E = (0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0)^t$$

그러므로 $M_{aug}{}^{-1}$을 구하지 않고도 세 번째 비트에서 오류가 발생했음을 알 수 있다.

따라서 수신된 부호어는 다음과 같이 복호된다.

$$C = W \oplus E$$

$$= (1,1,1,1,0,0,0,0,0,0,0,1,0,0,0,1)$$

3. 압축

앞에서 언급한 $(2k, k, 3)$ 오류정정부호의 설계는 검사어를 생성함에 있어 검사어 비트의 수를 전송하고자 하는 데이터 비트의 수만큼 생성해야 하는 단점이 있다. 여기서는 주어진 데이터 비트에 대하여 검사어 비트수를 효과적으로 줄일 수 있는 방법을 제안한다.

정리8> 부호 C에서 행렬 M에 의해 생성된 i, j 번째 검사어 비트 r_i와 r_j를 $r_i \oplus r_j$로 대체함으로써 검사어 비트를 줄일 수 있다. 이 때 행렬 M의 i, j 번째 행은 서로 같은 위치에서 1을 갖지 않아야 하고, 압축행렬 M'도 행렬 M의 조건들을 만족하여야 한다. □

예제9> 예제4의 $(16, 8, 3)$ 부호에서 전송할 데이터 비트를 $D = (d_0, d_1, \cdots, d_7)$, 생성된 검사어 비트를 $R = (r_0, r_1, \cdots, r_7)$라 두면 검사어 비트 각각은 다음과 같다.

$$
\begin{aligned}
r_0 &= d_0 \oplus d_1 \\
r_1 &= d_1 \oplus d_2 \\
r_2 &= d_2 \oplus d_3 \\
r_3 &= d_3 \oplus d_4 \\
r_4 &= d_4 \oplus d_5 \\
r_5 &= d_5 \oplus d_6 \\
r_6 &= d_6 \oplus d_7 \\
r_7 &= d_0 \oplus d_6 \oplus d_7
\end{aligned}
$$

r_{ij}를 r_i와 r_j를 더한 방정식이라 정의하면 정리8에 의해 위 검사어 비트를 다음과 같이 8비트에서 5비트로 압축할 수 있다.

$$
\begin{aligned}
r_{03} &= d_0 \oplus d_1 \oplus d_3 \oplus d_4 \\
r_{15} &= d_1 \oplus d_2 \oplus d_5 \oplus d_6 \\
r_{26} &= d_2 \oplus d_3 \oplus d_6 \oplus d_7 \\
r_4 &= d_4 \oplus d_5 \\
r_7 &= d_0 \oplus d_6 \oplus d_7
\end{aligned}
$$

따라서 행렬 M에서 유도된 압축행렬 M'는 다음과 같다.

$$
M' = \begin{bmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
$$

이를 이용한 부호화, 복호화 방법은 행렬 M에 의해 생성된 부호에서 제안한 방법과 동일하다.

다음 표은 행렬 M을 압축하여 생성한 검사어 비트수와 기존의 검사어 비트수를 비교한 것이다.

데이터 비트 (k)	검사비트 ($n-k$)		데이터 비트 (k)	검사비트 ($n-k$)	
	Chowdhury	본 연구		Chowdhury	본 연구
9	6	**6**	21	8	**8**
10	6	**6**	22	10	**8**
11	6	**6**	23	10	**8**
12	6	**6**	24	10	**9**
13	6	**6**	25	10	**9**
14	7	**6**	26	11	**9**
15	7	**7**	27	10	**9**
16	8	**7**	28	12	**9**
17	8	**7**	29	12	**9**
18	8	**7**	30	12	**9**
19	8	**7**	31	12	**10**
20	9	**8**	32	13	**10**

4. 결론

본 논문에서는 특별한 행렬을 이용하여 단일 오류정정부호를 효율적으로 부호화, 복호화하는 방법을 제시하였다. 복호화 과정에서 M_{aug}를 M 행렬과 단위행렬을 성분으로 하는 블록행렬로 구성함으로써 $S(W)$와 $S_{aug}(W)$를 분석이용이하여 보다 간단한 복호 방법과 복호 알고리즘을 구성하였다. 또 검사어를 효과적으로 압축하였다.

참고문헌

[1] D.R. Chowdhury, S. Basu, I.S. Gupta, P.P. Chaudhuri, CA Based Byte Error-Correcting Code, IEEE Trans. Computers, Vol. 44, 1995, pp. 371-382.

[2] D.R. Chowdhury, S. Basu, I.S. Gupta, P.P. Chaudhuri, *Design of CAECC - Cellular Automata Based Error Correcting Code*, IEEE Trans. Computers, Vol. 43, 1994, pp. 759-764.

[3] T.R.N. Rao, E. Fujiwara, Error-Control Coding for Computer System, Prentice Hall, New Jersey, 1989.

[4] F.J. Mac Williams and N.J.A. Sloane, The Theory of Error Correcting Codes, North-Holland, 1977.