通信情報保護學會論文誌
第 7 卷 第 4 號. 1997. 12

Reed-Solomon 부호의 오류위치 탐지회로 설계

조 용 석*, 박 상 규**

Design of Error Location Searching Circuit for Reed-Solomon Codes

Yong Suk Cho, Sang Kyu Park

요 약

본 논문에서는 Reed-Solomon 부호의 복호에서 오류위치를 찾는 방법을 제안하고 그 회로를 설계한다. 제안된 오류위치 탐지법을 사용하면 Reed-Solomon 복호에서 가장 복잡하고 지연이 많이 걸리는 역원기를 생략할 수 있다. 따라서 기존의 복호기보다 훨씬 간단하고 고속으로 동작하는 Reed-Solomon 복호기를 설계할 수 있다.

Abstract

In this paper, the error location searching method for Reed-Solomon codes and its circuit are proposed. Using the proposed method, all division operations are eliminated from the computation of the error location polynomial. The attractive feature of the method is its remarkable simplicity from the point of view of implementation.

Keyword : Error Correcting Codes, Reed-Solomon codes, BCH codes

I. 서 론

Reed-Solomon 부호는 1960년 I. S. Reed와 G. Solomon[1]에 의해 제안된 부호로, 오류정정 능력이 우수하고 랜덤오류(random error) 및 연집오류(burst error)를 동시에 정정할 수 있기 때문에 DTV(Digital TV), CDPD(Cellular Digital Packet Data) 등과 같은 디지털 통신 시스템과 CD(Compact Disk), DAT(Digital Audio Tape) 등과 같은 데이터 저장시스템에 널리 사용되고 있는 부호이다.

Reed-Solomon 부호의 복호는 일반적으로 수신 시퀀스(received sequence)로부터 오증(syndrome)을 구하고 그 오증으로부터 오류위

* 영동대학교 전자공학부
** 한양대학교 공과대학 전자통신과

치(error location)와 오류값(error value)을 구하여 오류를 정정하는 것이다. 이 과정 중에서 가장 핵심이 되는 부분이 오류위치를 구하는 과정으로 이에 대한 해법에 따라 PGZ(Peterson-Gorenstein-Zierler) 복호법[2],[3], Berlekamp-Massey 복호법[4],[5], Euclid 복호법[6] 등으로 분류된다. 오류정정능력이 비교적 작은 경우에는 PGZ 복호법이 효율적이며 오류정정능력이 클 때는 후자의 두 복호법이 더 효율적이다.

오류위치를 구하는 과정은 일반적으로 오증으로부터 오류위치의 역수를 근(root)으로 갖는 오류위치다항식을 구한 다음 그 근을 구하여 오류위치를 찾는다. Chien[7]은 2진(binary) BCH 부호의 복호에 있어서 부호의 순회성을 이용하여 오류위치를 찾는 간단한 방법을 제안하였다. 본 논문에서는 이 방법을 확장하여 Reed-Solomon 부호의 복호에 적용하고 그 회로를 설계한다.

제안된 방법의 특징은 오류위치를 구하는데 유한체(finite field or Galois field) $GF(2^m)$ 상의 나눗셈을 생략할 수 있다는 점이다. 유한체 $GF(2^m)$ 상의 연산 중에서 나눗셈은 장치화가 가장 복잡하고 어려운 연산이다. 따라서 나눗셈을 사용하지 않으면 복호기의 하드웨어가 간단해지며 그 만큼 복호지연(decoding delay)도 짧아지게 된다.

먼저 Ⅱ.에서는 Reed-Solomon 부호의 일반적인 오류위치 탐지법을 분석하고 이를 토대로 Ⅲ.에서 유한체 $GF(2^m)$ 상의 나눗셈을 생략할 수 있는 새로운 오류위치다항식 계산법을 제안하고 이를 이용한 오류위치 탐지회로를 설계한다. 끝으로 Ⅳ.에서 결론을 맺는다.

Ⅱ. Reed-Solomon 부호의 오류위치 탐지법

부호다항식을 $c(x)$, 오류다항식 $e(x)$라 하면 수신다항식 $r(x)$는 다음과 같이 쓸 수 있다.

$$r(x) = c(x) + e(x) \tag{1}$$

α를 유한체 $GF(2^m)$의 원시원(primitive element)이라 할 때, 생성다항식의 근인 α^j ($j = 1, 2, \cdots\cdots, 2t$)를 식(1)에 대입하면 $c(\alpha^j) = 0$ 이므로 다음과 같이 된다.

$$r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) \tag{2}$$

식 (2)는 오류가 발생하지 않으면 0이고 오류가 발생하면 0이 되지 않는다. 따라서 이 식을 다음과 같이 오증(syndrome)으로 정의한다.

$$S_j \equiv r(\alpha^j) = e(\alpha^j),\ j = 1, 2, \cdots\cdots, 2t \tag{3}$$

오증은 식 (3)과 같이 수신 시퀀스로부터 구할 수 있다.

오증을 구한 다음에는 이 오증으로부터 오류위치다항식을 구한다. 오류다항식을

$$e(x) = e_0 + e_1 x + \cdots\cdots + e_{n-1} x^{n-1} \tag{4}$$

이라 할 때 만약 $u(1 \le u \le t)$ 개의 오류가 $i_1, i_2, \cdots\cdots, i_u$ ($i_1 < i_2 < \cdots\cdots < i_u$) 위치에서 발생하였다고 가정하면, 식 (4)의 오류다항식은 다음과 같이 된다.

$$e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots\cdots + e_{i_u} x^{i_u} \tag{5}$$

여기에서 오류위치 X_k와 오류값 Y_k를 다음과 같이 정의한다.

$$X_k \equiv \alpha^{i_k},\ Y_k \equiv e_{i_k},\ 1 \le k \le u \tag{6}$$

식 (5)와 식 (6)을 식 (3)에 대입하면 오증 S_j는

$$S_j = \sum_{i=1}^{u} Y_i X_i^j,\quad j = 1, 2, \cdots\cdots, 2t \tag{7}$$

가 되며 이를 $j = 1, 2, 3, \ldots\ldots, 2t$에 대하여 풀어쓰면 다음과 같은 $2t$개의 방정식이 된다.

$$
\begin{aligned}
S_1 &= Y_1 X_1 + Y_2 X_2 + \cdots\cdots + Y_u X_u \\
S_2 &= Y_1 X_1^2 + Y_2 X_2^2 + \cdots\cdots + Y_u X_u^2 \\
S_3 &= Y_1 X_1^3 + Y_2 X_2^3 + \cdots\cdots + Y_u X_u^3 \\
&\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad\quad \vdots \quad\quad\quad \vdots \\
S_{2t} &= Y_1 X_1^{2t} + Y_2 X_2^{2t} + \cdots\cdots + Y_u X_u^{2t}
\end{aligned} \tag{8}
$$

Reed-Solomon 부호의 복호는 이 $2t$개의 방정식으로부터 $2u$개의 미지수 $X_1, X_2, \cdots\cdots, X_u$와 $Y_1, Y_2, \cdots\cdots, Y_u$를 구하는 것이다. 그러나 이 방정식은 비선형 방정식이므로 직접 해를 구하는 것은 매우 어렵다. Peterson[2]은 이와 같은 비선형 방정식을 다음과 같은 오류위치다항식을 도입하여 해를 구하는 방법을 처음 제안하였다.

먼저 오류위치의 역수를 근으로 갖는 오류위치다항식을 다음과 같이 정의한다.

$$
\begin{aligned}
\sigma(x) &= (1 + X_1 x)(1 + X_2 x) \cdots\cdots (1 + X_u x) \\
&= 1 + \sigma_1 x + \sigma_2 x^2 + \cdots\cdots + \sigma_u x^u \\
&= \sum_{k=0}^{u} \sigma_k x^k, \quad \sigma_0 = 1
\end{aligned} \tag{9}
$$

식 (9)의 양변에 $Y_i X_i^{j+u}$를 곱하고 $x = X_i^{-1}$을 대입하면

$$
\begin{aligned}
0 &= Y_i X_i^{j+u} (1 + \sigma_1 X_i^{-1} + \cdots\cdots + \sigma_u X_i^{-u}) \\
&= Y_i (X_i^{j+u} + \sigma_1 X_i^{j+u-1} + \cdots\cdots + \sigma_u X_i^{j})
\end{aligned} \tag{10}
$$

가 된다. 여기에서 i를 1부터 u까지 변화시켜서 모두 더하면 다음과 같이 된다.

$$
0 = \sum_{i=1}^{u} Y_i (X_i^{j+u} + \sigma_1 X_i^{j+u-1} + \cdots\cdots + \sigma_u X_i^{j}) \tag{11}
$$

여기에 식 (7)을 대입하면 다음과 같이 쓸 수 있다.

$$
S_{j+u} + \sigma_1 S_{j+u-1} + \cdots\cdots + \sigma_u S_j = 0 \tag{12}
$$

식 (12)를 $j = 1, 2, \cdots\cdots, u$에 대하여 풀어쓰고 행렬로 표현하면 다음과 같이 된다.

$$
\underbrace{\begin{bmatrix} S_u & S_{u-1} & \cdots\cdots & S_1 \\ S_{u+1} & S_u & \cdots\cdots & S_2 \\ S_{u+2} & S_{u+1} & \cdots\cdots & S_3 \\ \vdots & \vdots & \vdots & \vdots \\ S_{2u-1} & S_{2u-2} & \cdots\cdots & S_u \end{bmatrix}}_{\mathrm{A}} \cdot \underbrace{\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_u \end{bmatrix}}_{\sigma} = \underbrace{\begin{bmatrix} S_{u+1} \\ S_{u+2} \\ S_{u+3} \\ \vdots \\ S_{2u} \end{bmatrix}}_{\mathrm{B}} \tag{13}
$$

Peterson[8]은 식 (13)에서 행렬식(determinant) |A|는 오류가 u개 발생하였을 경우에는 0이 아니며(non singular), $u-1$개 이하가 발생하였을 경우에는 0이 된다는 것을 증명하였다. 그러므로 행렬식 |A|를 계산하면 실제 발생한 오류의 개수 u를 찾을 수 있다.

따라서 오증 S_j는 식 (3)으로부터 이미 알고 있는 상수이고, 이 방정식은 선형방정식이므로 식 (8)보다는 비교적 쉽게 해를 구할 수 있다. 오증으로부터 오류위치다항식의 계수 $\{\sigma_i \mid 1 \le i \le u\}$를 구하는 것은 Reed-Solomon 부호의 복호과정 중 가장 어렵고 복잡한 과정으로 이에 대한 해법이 Reed-Solomon 부호의 복호 중에서 가장 핵심이 되는 부분이다.

예를 들어 2개 이하의 모든 오류를 정정할 수 있는 DEC(Double Error Correcting) Reed-Solomon 부호의 오류위치다항식은 식 (13)에서 $u = 2$로 놓으면 식 (14)가 된다.

$$
\begin{bmatrix} S_2 & S_1 \\ S_3 & S_2 \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} \tag{14}
$$

여기에서 행렬식 |A|는

$$
|\mathbf{A}| = \begin{vmatrix} S_2 & S_1 \\ S_3 & S_2 \end{vmatrix} = S_2^2 + S_3 S_1 \tag{15}
$$

가 되며 이 |A|가 0이 아니면 2중 오류가 발생한 것으로, 2중 오류인 경우의 오류위치다항식의 계수는 식 (14)를 σ_1과 σ_2에 대하여 풀면 다음과 같이 된다.

$$\sigma_1^{(2)} = \frac{S_2S_3 + S_1S_4}{S_2^2 + S_1S_3}, \sigma_2^{(2)} = \frac{S_3^2 + S_2S_4}{S_2^2 + S_1S_3} \quad (16)$$

식 (16)에서 혼동을 피하기 위하여 실제 발생한 오류의 개수를 위첨자로 표시하였다. 행렬식 |A|가 0이면 단일 오류가 발생한 것으로, 단일 오류의 경우 오류위치다항식의 계수는

$$\sigma_1^{(1)} = \frac{S_2}{S_1} \quad (17)$$

가 된다. 따라서 식 (16)과 식 (17)을 구현하려면 GF(2^m) 상의 나눗셈기 3개, 곱셈기가 4개, 덧셈기 3개, 제곱기 2개가 필요하다. 여기에서 문제가 되는 것은 나눗셈기로 이것을 구현하는 데에는 많은 회로가 소요되며 지연이 많이 발생하게 된다.

일반적으로 유한체 GF(2^m) 상의 임의의 두 원소 사이의 나눗셈은 한 원소의 역원(inverse element)에 다른 원소를 곱하는 것이다. GF(2^m) 상의 0이 아닌 임의의 한 원소 Z는

$$Z^{2^m-1} = 1 \quad (18)$$

가 되므로 식 (18)의 양변에 Z^{-1}을 곱하여 정리하면 다음과 같이 된다.

$$Z^{-1} = Z^{-1} \cdot Z^{2^m-1} = Z^{2^m-2} = Z^{2^1} \cdot Z^{2^2} \cdot Z^{2^3} \cdot \cdots\cdots \cdot Z^{2^{m-1}} \quad (19)$$

따라서 역원기는 식 (19)를 병렬로 구현하면 $m-1$개의 멱승기(제곱기, 4승기, 8승기, ……)와 $m-2$개의 곱셈기가 필요하게 된다. 직렬로 구현하면 제곱기 1개와 곱셈기 1개가 필요하며 $m-2$ 클럭의 지연이 발생하게 된다. 또 나눗셈기는 역원기에 곱셈기 1개가 추가로 필요하므로 대단히 복잡한 회로가 된다. 그러므로 나눗셈을 생략할 수 있으면 회로 규모가 대폭 축소됨은 물론 지연도 짧아진다.

오류위치 X_i는 정의에 따라 오류위치다항식의 근의 역수이므로, 오류위치를 찾기 위해서는 먼저 GF(2^m) 상의 방정식 $\sigma(x) = 0$의 해를 구하여야 한다. 식 (9)에서 방정식 $\sigma(x) = 0$를 다시 쓰면 다음과 같이 된다.

$$\sum_{k=1}^{\nu} \sigma_k x^k = 1 \quad (20)$$

그러므로 식 (20)의 해를 구하여 역수를 취하면 오류위치를 구할 수 있다. Chien[7]은 이 방정식을 풀지 않고 순회성을 이용하여 오류위치를 구하는 간단한 방법을 제안하였다.

유한체 GF(2^m)은 2^m개의 유한한 원소를 가지고 있으므로, 오류위치다항식 $\sigma(x)$의 근은 유한체 GF(2^m)의 모든 원소를 대입하는 방법으로 찾을 수 있다. 즉 $1, \alpha, \alpha^2, \cdots\cdots, \alpha^{n-1}$ ($n = 2^m - 1$)을 $\sigma(x)$에 대입하여 그 값이 0이 되는 원소가 $\sigma(x)$의 근이 된다.

GF(2^m) 상의 임의의 한 원소 α^i($0 \le i \le n-1, n = 2^m - 1$)가 오류위치다항식 $\sigma(x)$의 근이라고 하면, 오류위치는 정의에 의하여 α^i의 역수인 α^{-i}가 된다. 유한체 GF(2^m)에서는

$$\alpha^n = \alpha^{2^m-1} = 1 \quad (21)$$

이므로, 임의의 한 원소 α^i의 역수 α^{-i}는 다음과 같이 쓸 수 있다.

$$\alpha^{-i} = 1 \cdot \alpha^{-i} = \alpha^n \cdot \alpha^{-i} = \alpha^{n-i}, 0 \le i \le n-1, n = 2^m-1 \quad (22)$$

따라서 α^i가 오류위치다항식 $\sigma(x)$의 근이라고 하면 오류위치는 α^{n-i}가 된다. 그러므로 (n, k) Reed-Solomon 부호에서 α^i가 $\sigma(x)$의 근이라고 하면, 수신심벌 $(r_0, r_1, r_2, \cdots\cdots, r_{n-1})$ 중에서, r_{n-i}심벌이 오류심벌이 된다. 수신심벌이 높은 차수부터 입력되어 차례로 복호 된다고 할 때, r_{n-1}심벌을 복호하려면 α^{n-1}이 오류위치인지를 검사하여 오류위치이면 r_{n-1}심벌에 오류값(error value)을 더하여 출력하고, 아니면

그대로 출력하면 된다.

α^{n-1}이 오류위치인지를 검사하는 것은 α^{n-1}의 역수인 α가 오류위치다항식의 근인지를 검사하는 것이다. 즉 $\sigma(\alpha) = 0$ 인지를 검사하여 0이면 r_{n-1}심벌을 그 위치의 오류값과 더하여 출력하고, 아니면 그대로 출력한다. 같은 방법으로 r_{n-2}심벌은 $\alpha(\sigma^2) = 0$ 인지를, r_{n-3}심벌은 $\sigma(\alpha^3) = 0$ 인지를, ·····, r_0심벌은 $\sigma(\alpha^n) = \sigma(1) = 0$ 인지를 검사하여 0이면 그 위치의 오류값과 더하여 출력하고, 아니면 그대로 출력하는 것이다. 이같은 과정을 식으로 나타내면 다음과 같이 쓸 수 있다.

$$\sigma(\alpha) = 1 + \sigma_1\alpha + \sigma_2\alpha^2 + \cdots\cdots + \sigma_u\alpha^u = 0$$

$$\sigma(\alpha^2) = 1 + \sigma_1\alpha^2 + \sigma_2(\alpha^2)^2 + \cdots\cdots + \sigma_u(\alpha^2)^u = 0$$

$$\vdots$$

$$\sigma(\alpha^n) = 1 + \sigma_1\alpha^n + \sigma_2(\alpha^n)^2 + \cdots\cdots + \sigma_u(\alpha^n)^u = 0 \quad (23)$$

식 (23)은 다음과 같이 식 (20)의 x 대신에 $\alpha_j(j = 1, 2, \cdots\cdots, n)$를 대입한 것이다.

$$\sum_{k=1}^{u} \sigma_k x^k = \sum_{k=1}^{u} \sigma_k(\alpha^j)^k = \sum_{k=1}^{u} \sigma_k(\alpha^k)^j = 1,$$

$$j = 1, 2, \cdots\cdots, n \quad (24)$$

식 (24)를 이용하여 회로를 구성하면 그림 1과 같이 된다.

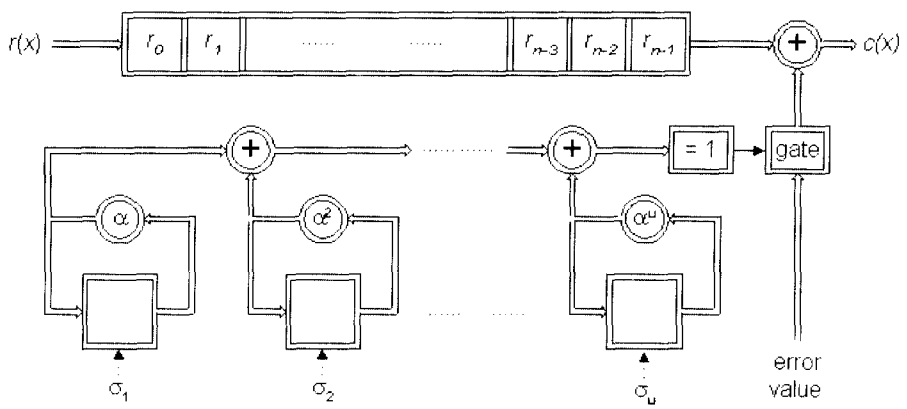


그림 1. Reed-Solomon 부호의 오류위치 탐지회로
Fig. 1. Error location searching circuit for Reed-Solomon codes

그림 1에서 □는 m비트 레지스터이고 점선은 각 레지스터에 초기값으로 $\sigma_1, \sigma_2, \cdots\cdots, \sigma_u$를 로드(load)하는 것을 나타내고 있다. ⓐ는 $GF(2^m)$ 상의 임의의 상수 α^i와의 곱셈기이고 ⊕는 $GF(2^m)$ 상의 덧셈기로 m개의 2입력 Exclusive-OR 게이트로 구현할 수 있다. [=1]는 최하위 비트를 반전시킨 m비트 입력이 모두 0일 때에만 1을 출력하는 회로로 NOT 게이트 1개와 m입력 NOR 게이트로 구현할 수 있다. [gate]는 m개의 AND 게이트로 구현할 수 있다.

Ⅲ. 오류위치 탐지회로 설계

식 (13)에서 행렬식 |A|가 0이 아니면 다음과 같이 쓸 수 있다.

$$\sigma = \mathbf{A}^{-1} \cdot \mathbf{B} \tag{25}$$

여기에서 행렬 A의 여인자(cofactor)를 A_{ik}라 하면, 역행렬(inverse matrix) A^{-1}은

$$A^{-1} = \frac{1}{|A|}\begin{bmatrix} A_{11} & A_{12} & \cdots\cdots & A_{1u} \\ A_{21} & A_{22} & \cdots\cdots & A_{2u} \\ A_{31} & A_{32} & \cdots\cdots & A_{3u} \\ \vdots & \vdots & \vdots & \vdots \\ A_{u1} & A_{u2} & \cdots\cdots & A_{uu} \end{bmatrix}^{T} \tag{26}$$

가 된다. 그러므로 식 (25)는 다음과 같이 쓸 수 있다.

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_u \end{bmatrix} = \frac{1}{|A|}\begin{bmatrix} A_{11} & A_{12} & \cdots\cdots & A_{u1} \\ A_{21} & A_{22} & \cdots\cdots & A_{u2} \\ A_{31} & A_{32} & \cdots\cdots & A_{u3} \\ \vdots & \vdots & \vdots & \vdots \\ A_{u1} & A_{u2} & \cdots\cdots & A_{uu} \end{bmatrix}\begin{bmatrix} S_{u+1} \\ S_{u+2} \\ S_{u+3} \\ \vdots \\ S_{2u} \end{bmatrix} \tag{27}$$

식 (27)을 풀어쓰면 다음과 같이 된다.

$$\sigma_1 = \frac{1}{|A|}(S_{u+1}A_{11} + S_{u+2}A_{21} + \cdots\cdots + S_{2u}A_{u1})$$

$$\sigma_2 = \frac{1}{|A|}(S_{u+1}A_{12} + S_{u+2}A_{22} + \cdots\cdots + S_{2u}A_{u2})$$

$$\vdots$$

$$\sigma_u = \frac{1}{|A|}(S_{u+1}A_{1u} + S_{u+2}A_{2u} + \cdots\cdots + S_{2u}A_{uu}) \tag{28}$$

따라서 식 (28)로부터 다음과 같이 쓸 수 있다.

$$\sigma_k = \frac{1}{|A|}\sum_{i=1}^{u} S_{u+i}A_{ik},\ k = 1, 2, \cdots, u \tag{29}$$

식 (29)를 식 (20)에 대입하면 다음과 같이 된다.

$$\sum_{k=1}^{u}\sigma_k x^k = \sum_{k=1}^{u}\left\{\frac{1}{|A|}\sum_{i=1}^{u} S_{u+i}A_{ik}\right\}x^k = 1 \tag{30}$$

여기에서 |A|는 k와 무관하므로 다음과 같이 쓸 수 있다.

$$\frac{1}{|A|}\sum_{k=1}^{u}\left\{\sum_{i=1}^{u} S_{u+i}A_{ik}\right\}x^k = 1 \tag{31}$$

또 |A|는 0이 아니라고 가정하였으므로 이를 양변에 곱하면

$$\sum_{k=1}^{u}\left\{\sum_{i=1}^{u} S_{u+i}A_{ik}\right\}x^k = |A| \tag{32}$$

가 되며 이를 $k = 1, 2, \cdots, u$에 대하여 풀어쓰면 다음과 같이 된다.

$$\left(\sum_{i=1}^{u} S_{u+i}A_{ik}\right)x + \left(\sum_{i=1}^{u} S_{u+i}A_{i2}\right)x^2 + \cdots\cdots +$$

$$\left(\sum_{i=1}^{u} S_{u+i}A_{iu}\right)x^u = |A| \tag{33}$$

따라서 식 (33)의 해를 구하여 역수를 취하면 오류위치를 구할 수 있다. 그러므로 식 (33)을 오류위치다항식으로 사용할 수 있다. 식 (33)과 식 (9)를 비교하면 오류위치다항식의 계수는 다음과 같이 된다.

$$\sigma_0 = |A|$$

$$\sigma_1 = \sum_{i=1}^{u} S_{u+i}A_{i1}$$
$$= S_{u+1}A_{11} + S_{u+2}A_{21} + \cdots\cdots + S_{2u}A_{u1}$$

$$\sigma_2 = \sum_{i=1}^{u} S_{u+i}A_{i2}$$
$$= S_{u+1}A_{12} + S_{u+2}A_{22} + \cdots\cdots + S_{2u}A_{u2}$$

$$\vdots$$

$$\sigma_u = \sum_{i=1}^{u} S_{u+i} A_{iu}$$
$$= S_{u+1} A_{1u} + S_{u+2} A_{2u} + \cdots\cdots + S_{2u} A_{uu} \quad (34)$$

예를 들어 DEC Reed-Solomon 부호의 오류위치다항식의 계수는 식 (34)에서 $u = 2$로 놓으면 다음과 같이 구할 수 있다.

$$\sigma_0^{(2)} = |\mathbf{A}| = S_2^2 + S_3 S_1$$
$$\sigma_1^{(2)} = S_3 A_{11} + S_4 A_{21} = S_3 S_2 + S_4 S_1$$
$$\sigma_2^{(2)} = S_3 A_{12} + S_4 A_{22} = S_3^2 + S_4 S_2 \quad (35)$$

단일오류가 발생한 경우는 $u = 1$ 이므로 다음과 같이 된다.

$$\sigma_0^{(1)} = |\mathbf{A}| = S_1$$
$$\sigma_1^{(1)} = S_2 A_{11} = S_2 \quad (36)$$

따라서 식 (35)과 식 (36)은 곱셈기 4개, 덧셈기 3개, 제곱기 2개로 구현할 수 있다. 식 (16)과 식 (17)에 비하면 나눗셈기 3개가 필요없어졌으므로 회로가 대폭 간단해졌음을 알 수 있다. 이와 같이 간단해지는 정도는 오류정정능력 t가 커질수록 더욱 커진다.

식 (33)을 이용하여 그림 1과 같은 오류위치 탐지회로를 설계하면 그림 2와 같이 된다. 그림 1과 그림 2를 비교해보면 그림 2에서는 σ_0가 1이 아니므로 GF(2^m) 상의 덧셈기 1개가 추가된다. 그러나 2중 오류정정 Reed-Solomon 부호의 예에서 살펴보았듯이 오류위치다항식을 계산하는 데에는 식 (33)을 이용하면 식 (13)을 이용할 때 보다 훨씬 간단하게 장치화할 수 있다.

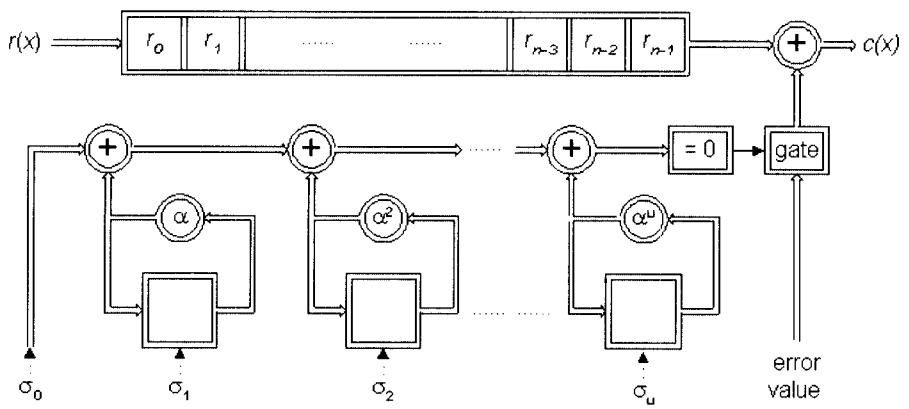


그림 2. 변형된 오류위치 탐지회로
Fig. 2. Modified error location searching circuit

Ⅳ. 결 론

Reed-Solomon 부호의 복호과정 중에서, 오증으로부터 오류위치다항식을 구할 때 종래에는 유한체 GF(2^m) 상의 나눗셈이 필요하였다. 유한체 GF(2^m) 상의 임의의 두 원소 사이의 나눗셈은 일반적으로 한 원소의 역원에 다른 원소를 곱하는 것으로 곱셈기와 역원기를 포함하는 것이다. 따라서 장치화가 대단히 복잡하고 어렵다.

본 논문에서는 Reed-Solomon 부호의 복호에서 오류위치를 찾을 때 나눗셈을 생략할 수 있는 방법을 제안하였다. 이 방법을 이용하면 나눗셈을 사용하지 않으므로 기존의 복호기보다 매우 간단한 하드웨어로 오류위치를 찾을 수 있으며 복호지연도 그 만큼 짧은 고속의 Reed-Solomon 복호기를 실현할 수 있다.

참 고 문 헌

[1] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," J. SIAM, Vol. 8, pp. 300~304, 1960.

[2] W. W. Peterson, "Encoding and Error correction procedure for Bose-Chaudhuri codes," IRE Trans. Inf. Theory, Vol. IT-6, pp. 459~470, September 1960.

[3] D. Gorenstein and N. Zierler, "A Class of Error Correcting Codes in p^m Symbols," J. SIAM, Vol. 9, pp. 207~214, June 1961.

[4] E. R. Berlekamp, Algebraic Coding Theory, New York: McGraw-Hill, 1968.

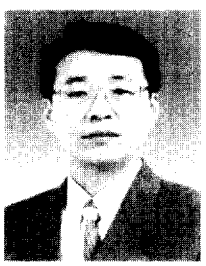
[5] J. L. Massey, "Shift Register Synthesis and BCH Decoding," IEEE Trans. Inform. Theory, Vol. IT-15, pp. 122~127, January 1969.

[6] Y. Sugiyama, Y. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Goppa Codes," Information and Control, Vol. 27, pp. 87~99, 1975.

[7] R. T. Chien, "Cyclic Decoding Procedure for the Bose Chaudhuri Hocquenghem Codes," IEEE Trans. Inform. Theory, Vol. IT-10, pp. 357~363, October 1964

[8] W. W. Peterson and E. J. Weldon, Error-Correcting Codes, 2nd Edition, M.I.T. Press, Cambridge, Mass., pp. 284~285, 1972.

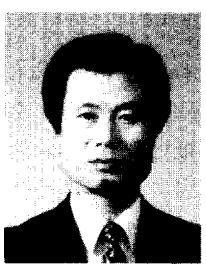
□ 著者紹介



조 용 석

1986년 2월 한양대학교 공과대학 전자통신과 (공학사)
1988년 2월 한양대학교 대학원 전자통신과 (공학석사)
1997년 2월 한양대학교 대학원 전자통신과 박사과정 수료
1989년 4월 ~ 1996년 2월 한국통신 연구개발원
1996년 3월 ~ 현재 영동대학교 전자공학부 전임강사

※ 주관심분야 : 디지털통신, 확산대역통신, 부호이론, PCS, IMT-2000



박 상 규

1974년 2월 서울대학교 전기공학 (공학사)
1980년 5월 Duke University 통신공학 (공학석사)
1987년 5월 University of Michigan 통신공학 (공학박사)
1976년 7월 ~ 1978년 10월 국방과학연구소
1990년 8월 ~ 1991년 8월 University of Southern California 객원교수
1987년 3월 ~ 현재 한양대학교 공과대학 전자통신과 부교수

※ 주관심분야 : 디지털통신, 확산대역통신, 부호이론, PCS, IMT-2000