Reed-Solomon 부호의 직접복호법을 이용한 3중 오류정정 복호기 설계

정회원 조 용 석*, 박 상 규**

Design of Triple-Error-Correcting Reed-Solomon Decoder using Direct Decoding Method

Yong-suk Cho*, Sang-kyu Park** *Regular Members*

요 약

본 논문에서는 오류정정 능력이 비교적 작은 경우에 매우 효율적인 직접복호법을 이용하여 기존의 복호기에 비해 하드웨어적으로 매우 간단한 새로운 3중 오류정정 Reed-Solomon 복호기의 설계 방법을 제안한다. 본 논문에서 설계한 3중 오류정정 Reed-Solomon 복호기는 오류위치다항식 및 오류평가다항식의 계산에 GF(2m) 상의 곱셈기가 9개만 사용되어, 기존의 복호기가 약 24개가 소요되는데 비해 매우 간단한 하드웨어로 구현할 수 있는 장점을 가지고 있다. 또한 제어회로도 매우 간단하고, 복호지연도 오증계산에 걸리는 한 블럭만큼만 소요되므로 수신 시퀀스를 저장하는 버퍼 메모리를 절약할 수 있다.

ABSTRACT

In this paper, a new design of a triple-error-correcting (TEC) Reed-Solomon decoder is presented based on direct decoding method which is more efficient for the case of relatively small error correction capability. The proposed decoder requires only 9 GF(2m) multipliers in obtaining the error-locator polynomial and the error-evaluator polynomial, whereas other decoders needs 24 multipliers. Thus, the attractive feature of this decoder is its remarkable simplicity from the point of view of implementation. Futhermore, the proposed TEC Reed-Solomon decoder has very simple control circuit and short decoding delay. Therefore this decoder can be implemented by simple hardware and also save buffer memory which stores received sequence.

I. 서 론

Reed-Solomon 부호는 1960년 I. S. Reed와 G. Solomon에 의해 제안된 부호로, 오류정정능력이 우수하고 랜덤오류(random error) 및 연집오류(burst error)를 동시에 정정할 수 있기 때문에 ATM (Asynchronous Transfer Mode), DTV(Digital Television) 등과 같은 디지털 통신시스템과 CD (Compact Disk), DVD(Digital Video Disk) 등과 같은 데이터 저장시스템에 널리 사용되고 있는 부호이다[1].

Reed-Solomon 부호에 대한 이론적인 연구는 비교적 잘 정리되어 있는 편이지만 더 적은 하드웨어로 더 고속의 복호기를 설계하고자 하는 연구는 여전히 활발한 연구 분야가 되고 있다. 이는 부호의 길이와 정정하고자 하는 오류의 개수에 따라 복호기의 복잡도(complexity) 및 복호지연(decoding delay)이 크게 달라지기 때문이다.

Reed-Solomon 부호의 일반적인 복호 방법은 수신 시퀀스(received sequence)로부터 오증(syndrome)

* 영동대학교 전자공학부 정보통신공학(yscho@youngdong.ac.kr),
** 한양대학교 전자전기공학부(skpark@email.hanyang.ac.kr)
논문번호 : 99086-0305, 접수일자 : 1999년 3월 5일

을 구하고, 그 오증으로부터 오류위치다항식(error-locator polynomial)과 오류평가다항식(error-evaluator polynomial)을 구한 다음, 이들을 이용하여 오류위치(error location)를 찾고 그 위치에 해당하는 오류값(error value)을 구하여 오류를 정정하는 것이다.

이 과정 중에서 가장 핵심이 되는 부분이 오류위치다항식과 오류평가다항식을 구하는 것으로 이의 해법에 따라 Berlekamp-Massey 복호법, Euclid 복호법, PGZ(Peterson-Gorenstein-Zierler) 복호법, 직접복호법 등으로 분류하고 있다[2]. 앞의 2가지 복호법은 오류정정능력이 큰 부호에 효율적인 복호법으로 반복 계산에 의해 복호를 수행하므로 소프트웨어 적으로 복호기를 실현할 경우 적합한 복호법이며, 뒤의 2가지 복호법은 오류정정능력이 비교적 작은 부호에 대하여 하드웨어 적으로 보다 고속의 복호기를 실현하는 경우에 적합한 복호법이다.

최근 제3세대 이동통신망으로 각광받고 있는 IMT-2000(International Mobile Telecommunication)의 무선접속 기술로 제안되고 있는 광대역 CDMA (Wideband Code Division Multiple Access) 시스템에서는 유한체 $GF(2^8)$ 상의 3중 오류정정 (47, 41) Reed-Solomon 부호가 권고되고 있으며 이의 구현에 관한 논문이 발표되고 있다[3].

본 논문에서는 3중 오류정정과 같이, 오류정정 능력이 비교적 작은 경우에 매우 효율적인 직접복호 알고리듬[4]을 이용하여 기존의 복호기에 비해 하드웨어적으로 매우 간단한 새로운 3중 오류정정 Reed-Solomon 복호기의 설계 방법을 제안한다.

3중 오류정정의 경우 기존의 PGZ 알고리듬으로 설계한 복호기[5]와 변형 Euclid 알고리듬으로 설계한 복호기[3],[6]가 오류위치다항식 및 오류평가다항식의 계산에 $GF(2^m)$ 상의 곱셈기가 약 24개가 소요되는데 비해, 본 논문에서 설계한 3중 오류정정 복호기는 9개만 사용되어 매우 간단한 하드웨어로 구현할 수 있는 장점을 가지고 있다. 또한 복호지연도 오증계산에 걸리는 한 블럭만큼의 지연만 소요되므로 매우 짧은 특징을 가지고 있다. 복호지연이 짧다는 것은 하드웨어적으로 보면 수신 시퀀스를 저장하는 버퍼 메모리가 그만큼 절약된다는 것을 의미한다.

먼저 Ⅱ장에서는 Reed-Solomon 부호의 직접복호 알고리듬을 분석한다. Ⅲ장에서는 이를 토대로 기존의 복호기에 비해 하드웨어적으로 매우 간단한 3중 오류정정 Reed-Solomon 복호기의 설계 방법을 제안하고 이를 다른 복호법으로 설계한 기존의 복호기와 하드웨어 복잡도를 비교한다. 끝으로 Ⅳ장에서 결론을 맺는다.

Ⅱ. Reed-Solomon 부호의 직접복호법

Reed-Solomon 부호의 직접복호법은 오증으로부터 오류위치다항식을 구한 다음 오류위치와 이에 해당하는 오류값을 구하는 일반적인 복호법과는 달리, 오증으로부터 직접 오류위치와 오류값을 구하여 오류를 정정하는 방법이다. 2진 BCH 부호에 대한 직접복호법은 Chien[7]이 처음 제안하였고 Forney[8]는 이 방법을 비2진 BCH 부호로 확장하였으며 Horiguchi와 Sato[4]는 개선된 방법을 제안하였다.

유한체 $GF(2^m)$의 원시원(primitive element)을 α라 할 때, t개 이하의 모든 오류를 정정할 수 있는 t중 오류정정 Reed-Solomon 부호의 생성다항식(generator polynomial)은 다음과 같이 정의된다[2].

$$g(x) = \prod_{i=1}^{2t} (x + \alpha^i) \tag{1}$$

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 가정하고 오류위치를 $X_k(=\alpha^{i_k})$, 오류값을 Y_k라 하면 오증 s_j는 다음과 같이 쓸 수 있다.

$$s_j = \sum_{i=1}^{u} Y_i X_i^j = \sum_{i=1}^{u} (Y_i X_i) X_i^{j-1}, \quad j = 1, 2, \cdots\cdots, 2t \tag{2}$$

여기에서 오류위치다항식을 다음과 같이 오류위치의 역수를 근으로 갖는 다항식으로 정의하면

$$\begin{aligned} \sigma(x) &= (1 + X_1 x)(1 + X_2 x) \cdots (1 + X_u x) \\ &= 1 + \sigma_1 x + \sigma_2 x^2 + \cdots\cdots + \sigma_u x^u \end{aligned} \tag{3}$$

오증 s_j와 오류위치다항식의 계수 σ_i간의 관계는 다음과 같은 행렬(matrix)로 표현할 수 있다[9].

$$\begin{bmatrix} s_1 & s_2 & \cdots\cdots & s_u \\ s_2 & s_3 & \cdots\cdots & s_{u+1} \\ s_3 & s_4 & \cdots\cdots & s_{u+2} \\ \vdots & \vdots & \vdots & \vdots \\ s_u & s_{u+1} & \cdots\cdots & s_{2u-1} \end{bmatrix} \cdot \begin{bmatrix} \sigma_u \\ \sigma_{u-1} \\ \sigma_{u-2} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} s_{u+1} \\ s_{u+2} \\ s_{u+3} \\ \vdots \\ s_{2u} \end{bmatrix} \tag{4}$$

식 (4)에서 가장 왼쪽에 있는 $u \times u$ 행렬을 $M_u(s)$라 하면, 이 행렬의 행렬식(determinant)

$|M_u(s)|$는 다음과 같이 전개할 수 있다.

$$|M_u(s)| = \prod_{i=1}^{u} Y_i X_i \prod_{i>j\geq 1}^{u} (X_i + X_j)^2 \tag{5}$$

이 행렬식 $|M_u(s)|$는 오류가 정확하게 u개 발생하였을 때는 0이 아니고 u개 보다 적게 발생하였을 경우에는 0이 된다[9].

여기에서 오증에 관한 다음과 같은 함수를 정의한다.

$$S_j(x) = s_j x^j \quad , \ j=1,2,\cdots\cdots,2t \tag{6}$$

$$A_j(x) = S_j(x) + S_{j+1}(x) \quad , \ j=1,2,\cdots\cdots,2t-1 \tag{7}$$

$$B_j(x) = S_j(x) + S_{j+2}(x) \quad , \ j=1,2,\cdots\cdots,2t-2 \tag{8}$$

식 (6)~(8)에 식 (2)를 대입하여 정리하면 다음과 같이 쓸 수 있다.

$$S_j(x) = \sum_{i=1}^{u} Y_i X_i^j x^j = \sum_{i=1}^{u} Y_i X_i x (X_i x)^{j-1} \tag{9}$$

$$A_j(x) = \sum_{i=1}^{u} [\, Y_i X_i x (1 + X_i x)] (X_i x)^{j-1} \tag{10}$$

$$B_j(x) = \sum_{i=1}^{u} [\, Y_i X_i x (1 + X_i x)^2] (X_i x)^{j-1} \tag{11}$$

여기에서 $A_j(x)$를 가지고 다음과 같이 구성한 $u \times u$ 행렬을 $M_u[A(x)]$라 하자.

$$M_u[A(x)] = \begin{bmatrix} A_1(x) & A_2(x) & \cdots & A_u(x) \\ A_2(x) & A_3(x) & \cdots & A_{u+1}(x) \\ A_3(x) & A_4(x) & \cdots & A_{u+2}(x) \\ \vdots & \vdots & \vdots & \vdots \\ A_u(x) & A_{u+1}(x) & \cdots & A_{2u-1}(x) \end{bmatrix} \tag{12}$$

이 행렬의 행렬식 $|M_u[A(x)]|$를 식 (5)와 같은 방법으로 전개하면 다음과 같이 쓸 수 있다.

$$\begin{aligned} |M_u[A(x)]| &= \prod_{i=1}^{u} Y_i X_i x (1 + X_i x) \cdot \prod_{i>j\geq 1}^{u} (X_i x + X_j x)^2 \\ &= x^{u^2} \cdot |M_u(s)| \cdot \sigma(x) \end{aligned} \tag{13}$$

식 (13)을 살펴보면 행렬식 $|M_u(s)|$는 u개의 오류가 발생하였을 경우 0이 아니므로 $|M_u[A(x)]| = 0$를 만족하는 근은 $\sigma(x) = 0$를 만족하는 근, 즉 오류위치임을 알 수 있다. 따라서 $|M_u[A(x)]|$를 오류위치다항식으로 이용할 수 있다.

$S_j(x)$를 가지고 식 (12)와 같은 방법으로 구성한 행렬을 $M_u[S(x)]$라 하고, 이 행렬의 행렬식 $|M_u[S(x)]|$를 전개하면 다음과 같이 된다.

$$|M_u[S(x)]| = \prod_{i=1}^{u} Y_i X_i x \prod_{i>j\geq 1}^{u} (X_i x + X_j x)^2 \tag{14}$$

여기에 $i=1$, $j=1$, $x = X_1^{-1}$을 대입하면 다음과 같이 쓸 수 있다.

$$\begin{aligned} &|M_u[S(X_1^{-1})]| \\ &= Y_1 X_1 X_1^{-1} \prod_{i=2}^{u} Y_i X_i X_1^{-1} (1 + X_i X_1^{-1})^2 \\ &\quad \cdot \prod_{i>j\geq 2}^{u} (X_i X_1^{-1} + X_j X_1^{-1})^2 \end{aligned} \tag{15}$$

또 $B_j(x)$를 가지고 식 (12)와 같은 방법으로 구성한 행렬식 $|M_{u-1}[B(x)]|$를 전개하면

$$\begin{aligned} &|M_{u-1}[B(x)]| \\ &= \prod_{i=2}^{u} Y_i X_i x (1 + X_i x)^2 \prod_{i>j\geq 2}^{u} (X_i x + X_j x)^2 \end{aligned} \tag{16}$$

가 되며 여기에 $x = X_1^{-1}$을 대입하여 정리하면 다음과 같이 된다.

$$\begin{aligned} &|M_{u-1}[B(X_1^{-1})]| \\ &= \prod_{i=2}^{u} Y_i X_i X_1^{-1} (1 + X_i X_1^{-1})^2 \\ &\quad \cdot \prod_{i>j\geq 2}^{u} (X_i X_1^{-1} + X_j X_1^{-1})^2 \end{aligned} \tag{17}$$

여기에서 식 (15)를 식 (17)로 나누면

$$\frac{|M_u[S(X_1^{-1})]|}{|M_{u-1}[B(X_1^{-1})]|} = Y_1 \tag{18}$$

가 된다. 식 (18)을 일반화하면

$$Y_k = \frac{|M_u[S(X_k^{-1})]|}{|M_{u-1}[B(X_k^{-1})]|} \quad , \ k=1,2,\cdots\cdots,u \tag{19}$$

가 되며 따라서 식 (19)를 이용하면 오류위치 X_k^{-1}에 대응하는 오류값 Y_k를 구할 수 있다.

직접복호법의 복호 절차를 정리하면 다음과 같다.

1) $S_j = s_j$ $(j=1,2,\cdots\cdots,2t)$, $l=0$ 으로 설정한

다. 여기에서 s_j는 오증의 초기값이다.

2) $u=t$로 설정한다.

3) 식 (5)와 같은 행렬식 $|M_u(S)|$를 계산한다.

4) $|M_u(S)|\neq 0$이면 5)로 가고 $|M_u(S)|=0$이면 $u=u-1$로 설정하고 3)으로 되돌아간다.

5) $|M_u(A)|=0$이면 6)으로 가고, 아니면 7)로 간다. 여기에서 $A_j=S_j+S_{j+1}$이다.

6) 오류값 $Y=|M_u(S)|/|M_{u-1}(B)|$를 계산하여 오류를 정정한다. 여기에서 $B_j=S_j+S_{j+2}$이고 $|M_0(B)|=1$이다.

7) $l=n-1$이면 끝낸다. 아니면 $S_j=S_j\alpha^j$, $l=l+1$로 설정하고 2)로 되돌아간다.

이상과 같이 직접복호법은 오증 S_j에 α^j를 곱해 가면서 각 심벌 사이클마다 단계 4)~6)과 같은 계산을 수행하여 오류를 정정한다.

Ⅲ. 3중 오류정정 Reed-Solomon 복호기

한 블럭에서 발생하는 3개 이하의 모든 오류를 정정하는 3중 오류정정 부호는 오류가 1개, 2개, 3개 발생한 경우를 모두 고려하여야 한다. 먼저 오류위치다항식 $|M_1(A)|$, $|M_2(A)|$, $|M_3(A)|$는 다음과 같이 구할 수 있다.

$$|M_1(A)| = A_1 \tag{20}$$

$$|M_2(A)| = \begin{vmatrix} A_1 & A_2 \\ A_2 & A_3 \end{vmatrix} = A_2^2 + A_1A_3 \tag{21}$$

$$\begin{aligned} |M_3(A)| &= \begin{vmatrix} A_1 & A_2 & A_3 \\ A_2 & A_3 & A_4 \\ A_3 & A_4 & A_5 \end{vmatrix} \\ &= A_3^3 + A_1A_4^2 + A_5|M_2(A)| \end{aligned} \tag{22}$$

같은 방법으로 $|M_1(S)|$, $|M_2(S)|$, $|M_3(S)|$, $|M_1(B)|$, $|M_2(B)|$를 구하면 다음과 같이 된다.

$$|M_1(S)| = S_1 \tag{23}$$

$$|M_2(S)| = S_2^2 + S_1S_3 \tag{24}$$

$$|M_3(S)| = S_3^3 + S_1S_4^2 + S_5|M_2(S)| \tag{25}$$

$$|M_1(B)| = B_1 = S_1 + S_3 \tag{26}$$

$$|M_2(B)| = B_2^2 + B_1B_3 \tag{27}$$

또 식 (19)와 같은 오류값 계산식은 실제 발생한 오류의 개수 u가 1, 2, 3일 경우 각각 다르므로 이들을 각각 $Y^{(1)}$, $Y^{(2)}$, $Y^{(3)}$라 하고 계산하면

$$Y^{(1)} = \frac{|M_1(S)|}{|M_0(B)|} = S_1, \quad (|M_0(B)|=1) \tag{28}$$

$$Y^{(2)} = \frac{|M_2(S)|}{|M_1(B)|} = \frac{S_2^2+S_1S_3}{B_1} \tag{29}$$

$$Y^{(3)} = \frac{|M_3(S)|}{|M_2(B)|} = \frac{|M_3(S)|}{B_2^2+B_1B_3} \tag{30}$$

가 된다. 따라서 버퍼 레지스터의 출력과 더해지는 오류값 Y는 다음과 같이 정리할 수 있다.

$$Y = \begin{cases} Y^{(3)}, & \text{if } |M_3(S)|\neq 0 \text{ and } |M_3(A)|=0 \\ Y^{(2)}, & \text{if } |M_3(S)|=0 \text{ and } |M_2(S)|\neq 0 \\ & \quad \text{and } |M_2(A)|=0 \\ Y^{(1)}, & \text{if } |M_3(S)|=0 \text{ and } |M_2(S)|=0 \\ & \quad \text{and } S_1\neq 0 \text{ and } A_1=0 \end{cases} \tag{31}$$

이상의 결과를 이용하면 그림 1~3과 같은 3중 오류정정 Reed-Solomon 복호기를 설계할 수 있다. 그림 1~3에서 2중선은 m비트 버스를 표시한 것이며 점선은 계산된 오증을 병렬 로드(load)하는 것을 나타내고 있다. ⊕는 $GF(2^m)$ 상의 덧셈기로 m개의 2입력 exclusive OR 게이트(이하 XOR2로 표기)로 구현할 수 있으며, [=0]은 입력 비트가 모두 0일 때 0을 출력하는 전영검출기(all zero detector)로 $m-1$개의 2입력 OR 게이트(이하 OR2로 표기)로 구현할 수 있다. 또 [OR]는 비트 별로 OR하는 회로로 m개의 OR2로 구현할 수 있으며, [gate]는 선택(select) 입력이 1일 때에만 입력을 출력으로 통과시키는 회로로 m개의 2입력 AND 게이트(이하 AND2로 표기)로 구현할 수 있다. [MUX]는 m개의 2×1 multiplexer를 나타낸 것이며, [×]은 $GF(2^m)$ 상의 곱셈기를, [( )²]은 $GF(2^m)$ 상의 제곱기를, [INV]는 $GF(2^m)$ 상의 역원기를 표시한 것이다.

위에서 사용한 연산기들 중에서 하드웨어적으로 가장 복잡한 것이 유한체 $GF(2^m)$ 상의 곱셈기와 역원기이다. 원시다항식(primitive polynomial)이 $f(x)=1+x^2+x^3+x^4+x^8$인 유한체 $GF(2^8)$의 경우, 조합회로(combinational logic)를 이용한 병렬 곱셈

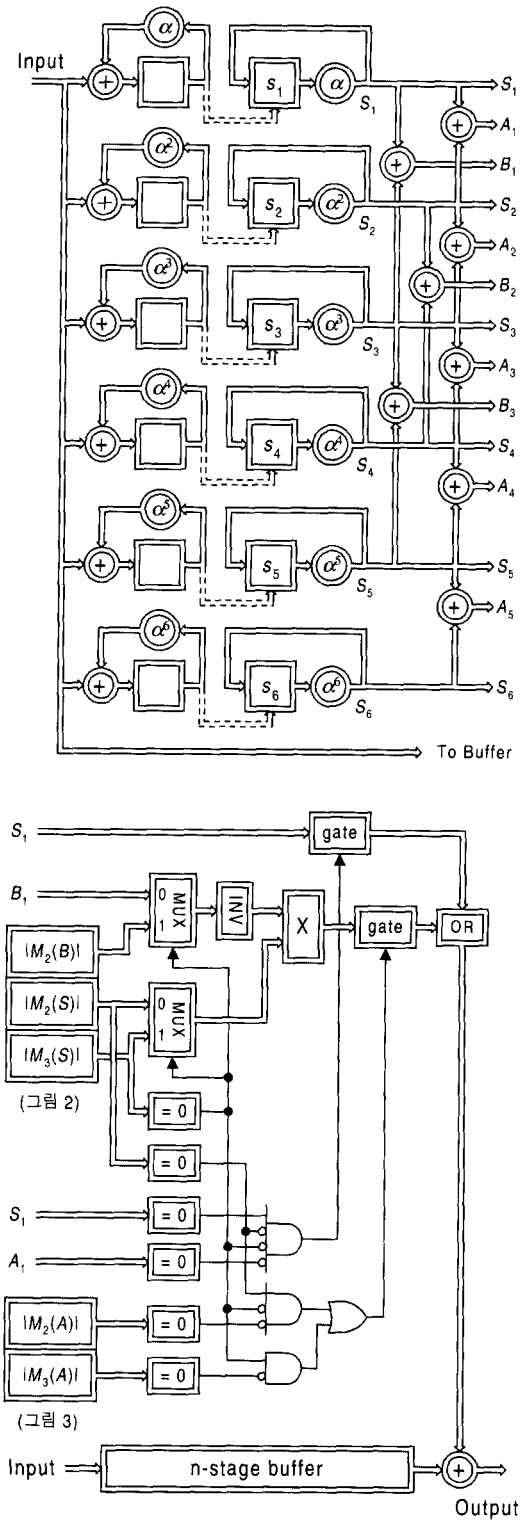

그림 1. 3중 오류정정 Reed-Solomon 복호기의 블럭도

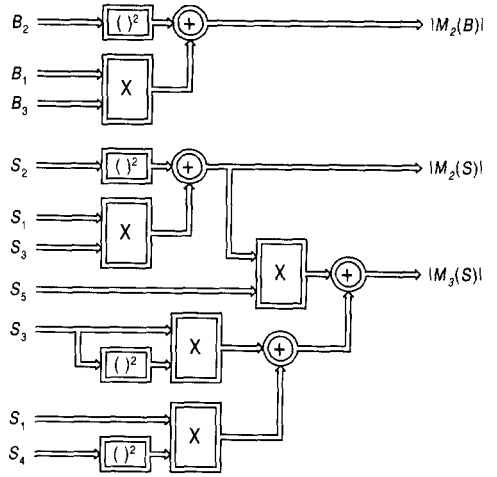

그림 2. 그림 1의 오류값 계산회로

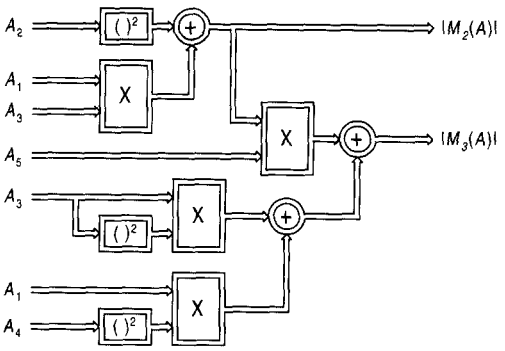

그림 3. 그림 1의 오류위치 계산회로

기는 64개의 AND2와 73개의 XOR2가 소요되며[5], 순서회로(sequential logic)를 이용한 직렬 곱셈기는 24개의 레지스터와 8개의 AND2 그리고 11개의 XOR2가 소요되는 반면 8 클럭 후에 결과를 얻을 수 있다.

유한체 $GF(2^8)$과 같이 유한체의 위수(order) m이 합성수(composite number)인 경우에는 부분체(subfields)를 이용하면 보다 효율적인 곱셈기를 설계할 수 있다[10]. 부분체 $GF(2^4)$를 이용한 $GF(2^8)$상의 병렬 곱셈기는 48개의 AND2와 62개의 XOR2로 구현할 수 있다[11]. 또 직렬 곱셈기도 부분체를 이용하여 구성할 수 있으며 이 경우에는 기존의 직렬 곱셈기에 비하여 하드웨어는 조금 더 많이 소요되지만 더 빠른 시간에 결과를 얻을 수 있다[12].

유한체 $GF(2^m)$에서 0이 아닌 임의의 한 원소의 역원(multiplicative inverse)을 계산하는 가장 간단

한 방법은 ROM(Read Only Memory)을 이용한 표 참조(table lookup) 방법이다. 그러나 이 방법은 유한체의 크기가 작을 때에는 효율적이지만 유한체의 크기가 커지면 ROM의 용량($2^m \times m$ 비트)이 급격하게 증가하는 문제점이 있다. 따라서 알고리듬 적으로 역원을 계산하는 방법인 Fermat의 정리를 이용한 방법이 널리 이용되고 있다.

Fermat의 정리를 이용한 고속 역원 계산 알고리듬을 사용할 경우 역원은 약 $m/2$회의 곱셈으로 구할 수 있다[13]. 따라서 역원기를 병렬로 구현하면 $m/2$개의 병렬 곱셈기가 필요하며, 직렬로 구현하면 1개의 병렬 곱셈기로 구현할 수 있지만 $m/2$ 클럭의 시간이 필요하게 된다. 그러나 부분체를 이용한 방법을 사용하면 병렬 곱셈기와 거의 비슷한 규모의 하드웨어로 병렬 역원기를 구현할 수 있다[14]. 부분체 $GF(2^4)$를 이용한 $GF(2^8)$ 상의 병렬 역원기는 64개의 AND2와 57개의 XOR2, 10개의 OR2 그리고 4개의 NOT 게이트로 구현할 수 있다[11].

이상과 같이 유한체 $GF(2^m)$ 상의 연산기들 중에서 하드웨어적으로 가장 복잡한 것이 역원기이다. 따라서 대부분의 복호기 설계에서는 오류위치다항식과 오류평가다항식 계산 과정에서 역원기의 사용을 피할 수 있는 방법들을 이용하고 있다. 그러므로 Reed-Solomon 복호기의 하드웨어 복잡도에 결정적으로 영향을 미치는 것은 오류위치다항식과 오류평가다항식의 계산에 사용되는 곱셈기의 개수가 된다.

3중 오류정정 Reed-Solomon 복호기의 경우, 오류위치다항식과 오류평가다항식의 계산에 소요되는 $GF(2^m)$ 상의 연산기 개수를 비교하면 표 1과 같다.

표 1. 다른 복호기와의 비교

$GF(2^m)$ 상의 연산기	문헌 [5]의 복호기	문헌 [6]의 복호기	본 논문의 복호기
곱 셈 기	24 개	24 개	9 개
덧 셈 기	15 개	12 개	15 개
제 곱 기	4 개	-	7 개

표 1에서 보듯이 기존의 PGZ 알고리듬으로 설계한 복호기[5]와 변형 Euclid 알고리듬으로 설계한 복호기[6]가 24개의 곱셈기를 사용한 반면에 본 논문에서 설계한 복호기는 그림 2와 그림 3과 같이 9개의 곱셈기만을 사용하므로 하드웨어 복잡도 면에서 기존의 복호기에 비해 훨씬 우수함을 알 수 있다. 또한 여기에서는 오류위치다항식과 오류평가다항식의 계산 부분만을 비교하였지만 제어회로를 포함한 그 외의 부분도 본 논문에서 설계한 복호기가 하드웨어적으로 훨씬 더 간단한 장점을 가지고 있다.

Ⅳ. 결 론

본 논문에서는 Reed-Solomon부호의 복호법 중에서 오류정정능력이 비교적 작을 경우 효율적인 직접복호법을 이용하여 하드웨어적으로 매우 간단한 3중 오류정정 Reed-Solomon 복호기를 설계하였다. 또 유한체 $GF(2^8)$ 상의 단축 3중 오류정정 부호인 (47, 41) Reed-Solomon 부호를 C 언어로 시뮬레이션하여 설계된 복호기의 동작을 검증하였다.

설계된 3중 오류정정 Reed-Solomon 복호기는 오류위치다항식과 오류평가다항식의 계산에 $GF(2^m)$ 상의 곱셈기가 9개만 소요되어 기존의 복호기가 24개가 소요되는데 비해 훨씬 간단한 하드웨어로 구현할 수 있다. 또한 제어회로도 매우 간단하며, 복호지연도 오증계산에 걸리는 부호길이 만큼만 소요되므로 수신 시퀀스를 저장하는 버퍼 메모리를 절약할 수 있는 장점을 가지고 있다.

그러나 제안된 복호기는 3중 오류정정에 한정된 것으로 향후 오류정정능력이 보다 큰 Reed-Solomon 복호기의 저복잡도 및 고속 복호기의 설계에 관한 연구가 필요할 것으로 생각된다.

참 고 문 헌

[1] M. Y. Rhee, Error Correcting Coding Theory, *McGraw-Hill*, New York, 1989.

[2] S. B. Wicker and V. K. Bhargava, Reed-Solomon Codes and Their Applications, *IEEE Press*, 1994.

[3] Y. C. Roh and I. K. Jin, "Implementation of Reed-Solomon CODEC for IMT-2000," *Procee- dings of the 4th International Workshop on Mobile Multimedia Communications*, pp. 147~ 150, Oct. 1997.

[4] T. Horiguchi and Y. Sato, "A Decoding Method for Reed-Solomon Codes over *GF* (2^m)," *Trans. IEICE*, Vol. J66-A, pp. 97~98, Jan. 1983.

[5] A. M. Patel, "On-the-fly Decoder for Multiple Byte Errors," *IBM J. RES. DEVELOP.*, Vol.

30, No. 3, pp. 259~269, May 1986.

[6] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Joseph, and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder," *IEEE Trans. Computers*, Vol. C-34, No. 5, pp. 393~403, May 1985.

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

[8] G. D. Forney, "On Decoding BCH Codes," *IEEE Trans. Information Theory*, Vol. IT-11, pp. 549~ 557, Oct. 1965.

[9] W. W. Peterson and E. J. Weldon, Error-Correcting Codes, *MIT Press*, Cambridge, Mass., 1972.

[10] C. Paar, "A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields," *IEEE Trans. Computers*, Vol. C-45, No. 7 pp. 856~861, July 1996.

[11] 조 용석, "광대역 CDMA 시스템을 위한 효율적인 Reed-Solomon CODEC 설계에 관한 연구," *한양대학교 대학원 박사학위논문*, 1998. 8.

[12] Yong Suk Cho and Sang Kyu Park, "Design of $GF(2^m)$ Multiplier Using Its Subfields," *Electronics Letters*, Vol. 34, No. 7, pp. 650~651, Apr. 1998.

[13] 조 용석, 박 상규, "유한체 $GF(2^m)$ 상의 역원계산 회로 및 나눗셈 회로 설계," *한국통신학회 논문지*, 제23권, 제5호, pp. 1160~1164, 1998. 5.

[14] M. Morii and M. Kasahara, "Efficient Construction of Gate Circuit for Computing Multiplicative Inverses over $GF(2^m)$," *Trans. IEICE*, Vol. E72, No. 1, pp. 37~42, Jan. 1989.

조 용 석(Yong-suk Cho) 정회원

1986년 2월 : 한양대학교 전자통신공학과 졸업
1988년 2월 : 한양대학교 전자통신공학과 석사
1998년 8월 : 한양대학교 전자통신공학과 박사
1989년 4월~1996년 2월 : 한국통신 연구개발원
1996년 3월~현재 : 영동대학교 전자공학부 조교수
<주관심 분야> 부호이론, 디지털통신, IMT-2000

박 상 규(Sang-kyu Park) 정회원

1974년 2월 : 서울대학교 전기공학과 졸업
1980년 5월 : Duke University 통신공학 석사
1987년 5월 : University of Michigan 통신공학박사
1976년 7월~1978년 10월 : 국방과학연구소
1990년 8월~1991년 8월 : University of Southern California 객원교수
1987년 3월~현재 : 한양대학교 공과대학 전자전기공학부 교수
<주관심 분야> 디지털통신, 확산대역통신, IMT-2000