通信情報保護學會論文誌
第 10 券 第 1 號, 2000. 3

ATM 적응계층에 적용 가능한 (128,124) Reed Solomon 부호의 직접복호법 및 VHDL 시뮬레이션*

김 창 규**

Direct Decoding Algorithm of (128,124) Reed-Solomon Codes for ATM adaptation layer and Its VHDL Simulation

Chang-kyu Kim**

요 약

AAL-1에서는 (128,124) Reed-Solomon부호를 사용한 인터리버 및 디인터리버에 의해 ATM 셀에서 발생하는 오류를 정정하고 있다. Reed-Solomon부호의 복호법 중 직접복호법은 오류위치다항식의 계산없이 오류위치와 오류치를 알 수 있으며 유한체 $GF(2^m)$의 표현에서 정규기저를 사용하면 곱셈과 나눗셈을 단순하게 비트 이동만으로 처리할 수 있다. 직접복호법과 정규기저를 사용하여 ATM 적응계층에 적용 가능한 (128,124) Reed-Solomon부호의 복호기를 설계하고 VHDL로 시뮬레이션 하였으며 이 복호기는 동일한 복호회로에 의해 둘 또는 하나의 심벌에 발생한 오류를 정정할 수 있다.

ABSTRACT

The (128,124) Reed-Solomon codes with an interleaved and deinterleaved scheme are used for the error correction of the AAL-1. Direct decoding method offers advantages of calculating error positions and error values without error-locator polynomial. Representation of finite field $GF(2^m)$ using normal basis reduces the complexity of multiplication or inverse over $GF(2^m)$. In this paper, an efficient decoding circuit of (128,124) Reed-Solomon codes which is applicable to the ATM cell is proposed and simulated by using VHDL. This designed Reed-Solomon decoder can correct double or single symbol errors with the same decoding circuit.

keyword : *AAL-1, direct decoding method, normal basis, Massey-Omura multiplier*

Ⅰ. 서 론

최근 디지털 통신시스템의 고속화로 인하여 데이터의 전송과 저장에 있어서 보다 효율적이고 신뢰성 있는 데이터 송수신을 요구하고 있다. 특히 초고속 정보통신망의 핵심 기술인 ATM 기술은 국내외에서 관련 시스템의 구현 작업이 한창 진행 중에 있다. AAL-1(ATM adaptation layer-1)에서의 오류정정은 (128,124) Reed-Solomon 부호를 사용한 인터리버 및 디인터리버에 의해 전체 47개의 ATM 셀에서 발생하는 4개의 셀 손실(또는 삭제)을 갖는 오류형태, 47개의 셀에서 2개의 셀 손실과 각 열에서 하나 이하의 심벌 오류를 갖는 오류형태, 그리고 128개의 심벌 중 2개 이하의 심벌에 발생한 오류를

* 이 논문은 동의대학교 자체 학술연구조성비 지원에 의해 수행된 연구임
** 동의대학교 전자통신공학과(E-mail : ckkim@hyomin.dongeui.ac.kr)

정정할 수 있다.[1,4]

Reed-Solomon부호(이하 RS부호)의 복호에서는 유한체 $GF(2^m)$의 연산이 필수적이며 복호 알고리즘에 따라 복호기의 복잡도는 달라진다.[4,5] 본 논문에서는 RS부호의 복호에서 오류위치다항식의 계산 없이 오류위치와 오류치를 알 수 있는 직접복호법을 소개하고, 유한체 $GF(2^m)$의 원소를 정규기저를 사용하여 표현함으로서 복호과정에서 만나게 되는 곱셈과 나눗셈을 보다 효율적으로 처리할 수 있도록 하여 복호기에서 유한체 연산의 복잡도를 줄일 수 있는 방법을 제시한다. 그리고 이들을 ATM 적응계층에 적용 가능한 (128,124) RS부호에 적용하여 복호기를 VHDL로 시뮬레이션함으로서 복호기의 동작에 대한 타당성을 검증한다.

II. 직접복호법

RS부호의 복호에서는 일반적으로 오증(syndrome)으로부터 오류위치다항식(error-locator polynomial)의 계수를 구하는 과정이 필요하다. Chien[2]은 2원 BCH부호에 대해 오류위치다항식을 구하지 않고 바로 오류를 정정할 수 있는 직접복호법을 제안하고 이를 설계하였다. 그러나 RS부호는 비2원 부호로서 오류위치뿐 아니라 오류치(error value)도 계산되어야 오류를 정정할 수 있다. 오류위치다항식을 구하지 않고 오류위치와 오류치를 계산할 수 있는 RS부호의 복호법으로 직접복호법(direct decoding method)[3]이 있다.

t 개의 오류를 정정할 수 있는 RS부호에서 오류위치 번호를 Z_j, 오류치를 Y_j라 할 때 수신계열에 대한 오증과 오류위치다항식은

$$S_i = \sum_{j=1}^{t} Y_j Z_j^i \qquad i = 1,2,\cdots,2t \tag{2.1}$$

$$\sigma(x) = \prod_{j=1}^{t} (1 + Z_j x) \tag{2.2}$$

로 각각 표현된다. 오증요소로 구성되는 $t \times t$ 정방행렬(square matrix)을

$$M_t = \begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \tag{2.3}$$

라 하고 Vandermonde 행렬[7,8,9] W를

$$W = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ Z_1 & Z_2 & \cdots & Z_t \\ \vdots & \vdots & \vdots & \vdots \\ Z_1^{t-1} & Z_2^{t-1} & \cdots & Z_t^{t-1} \end{bmatrix} \tag{2.4}$$

Λ를

$$\Lambda = \begin{bmatrix} Y_1 Z_1 & & & \\ & Y_2 Z_2 & & \\ & & \ddots & \\ & & & Y_t Z_t \end{bmatrix} \tag{2.5}$$

와 같은 대각행렬(diagonal matrix)로 놓으면, 행렬 W, Λ 그리고 W의 전치행렬(transpose matrix) W^T로 구성되는 행렬 $W\Lambda W^T$는

$$W\Lambda W^T = \begin{bmatrix} Y_1Z_1 + Y_2Z_2 + \cdots + Y_tZ_t & Y_1Z_1^2 + Y_2Z_2^2 + \cdots + Y_tZ_t^2 & \cdots \\ Y_1Z_1^2 + Y_2Z_2^2 + \cdots + Y_tZ_t^2 & Y_1Z_1^3 + Y_2Z_2^3 + \cdots + Y_tZ_t^3 & \cdots \\ \vdots & \vdots & \\ Y_1Z_1^t + Y_2Z_2^t + \cdots + Y_tZ_t^t & Y_1Z_1^{t+1} + Y_2Z_2^{t+1} + \cdots + Y_tZ_t^{t+1} & \cdots \\ Y_1Z_1^{t+1} + Y_2Z_2^{t+1} + \cdots + Y_tZ_t^{t+1} & Y_1Z_1^{t+2} + Y_2Z_2^{t+2} + \cdots + Y_tZ_t^{t+2} & \cdots \\ \vdots & \vdots & \\ Y_1Z_1^{2t-1} + Y_2Z_2^{2t-1} + \cdots + Y_tZ_t^{2t-1} & Y_1Z_1^{2t} + Y_2Z_2^{2t} + \cdots + Y_tZ_t^{2t} & \cdots \end{bmatrix} \tag{2.6}$$

이므로 오증요소로 식을 표현하면

$$W\Lambda W^T = \begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \tag{2.7}$$

이다. 이것은 앞에서 가정한 (2.3)식의 행렬 M_t와

동일한 행렬이다. 그러므로,

$$det[M_t] = det[W \Lambda W^T] = det[W]\, det[\Lambda] det[W^T] \quad (2.8)$$

와 같이 된다. 시스템에서 실제 발생한 오류의 수를 ν 라 하면, $t > \nu$ 이면 $det[M_t] = 0$ 이고, $t = \nu$ 이면 $det[M_t] \neq 0$ 이다. 즉, 오증요소로 이루어진 $t \times t$ 행렬

$$M_t(S) = \begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \quad (2.9)$$

의 행렬식 $det[M_t(S)]$에서 실제 발생한 오류의 수 ν와 t가 일치하면 $det[M_t(S)] \neq 0$ 이고, $t > \nu$ 이면 $det[M_t(S)] = 0$ 임을 알 수 있다. 위 식의 행렬식 $det[M_t(S)]$를 계산하면 아래와 같다.

$$det[M_t(S)] = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ Z_1 & Z_2 & \cdots & Z_t \\ \vdots & \vdots & \vdots & \vdots \\ Z_1^{t-1} & Z_2^{t-1} & \cdots & Z_t^{t-1} \end{vmatrix} \quad (2.10)$$
$$\begin{vmatrix} Y_1Z_1 & & & \\ & Y_2Z_2 & & \\ & & \ddots & \\ & & & Y_tZ_t \end{vmatrix} \begin{vmatrix} 1 & Z_1 & \cdots & Z_1^{t-1} \\ 1 & Z_2 & \cdots & Z_2^{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & Z_t & \cdots & Z_t^{t-1} \end{vmatrix}$$

여기서, Vadermonde 행렬식은

$$det\begin{bmatrix} 1 & 1 & \cdots & 1 \\ Z_1 & Z_2 & \cdots & Z_t \\ \vdots & \vdots & \vdots & \vdots \\ Z_1^{t-1} & Z_2^{t-1} & \cdots & Z_t^{t-1} \end{bmatrix} = det\begin{bmatrix} 1 & Z_1 & \cdots & Z_1^{t-1} \\ 1 & Z_2 & \cdots & Z_2^{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & Z_t & \cdots & Z_t^{t-1} \end{bmatrix} = \prod_{j>i\geq 1}^{t} (Z_j + Z_i) \quad (2.11)$$

이다. 따라서,

$$det[M_t(S)] = \prod_{j=1}^{t} Y_jZ_j \prod_{j>i\geq 1}^{t} (Z_j + Z_i)^2 \quad (2.12)$$

이고 오증요소를 이용한 다항식

$$S_i(x) = S_i x^i,\ i = 1, 2, 3, \cdots, 2t \quad (2.13)$$

$$A_i(x) = S_i(x) + S_{i+1}(x),\ i = 1, 2, 3, \cdots, 2t-1 \quad (2.14)$$

$$B_i(x) = S_i(x) + S_{i+2}(x),\ i = 1, 2, 3, \cdots, 2t-2 \quad (2.15)$$

들은 오증요소에 대한 표현 식(2.1)을 식(2.13), 식(2.14), 식(2.15)에 각각 대입하여

$$S_i(x) = \sum_{j=1}^{t} Y_jZ_j^i x^i = \sum_{j=1}^{t} Y_j(Z_jx)^i \quad i = 1, 2, 3, \cdots, 2t \quad (2.16)$$

$$A_i(x) = \sum_{j=1}^{t} Y_j(Z_jx)^i + \sum_{j=1}^{t} Y_j(Z_jx)^{i+1} = \sum_{j=1}^{t} Y_j(1 + Z_jx)(Z_jx)^i \quad i = 1, 2, 3, \cdots, 2t-1 \quad (2.17)$$

$$B_i(x) = \sum_{j=1}^{t} Y_j(Z_jx)^i + \sum_{j=1}^{t} Y_j(Z_jx)^{i+2} = \sum_{j=1}^{t} Y_j(1 + Z_jx)^2(Z_jx)^i \quad i = 1, 2, 3, \cdots, 2t-2 \quad (2.18)$$

로 표현할 수 있고, $S(x)$에 대한 $det[M_t(S(x))]$는

$$det[M_t(S(x))] = \prod_{j=1}^{t} Y_jZ_jx \prod_{j>i\geq 1}^{t} (Z_jx + Z_ix)^2 \quad (2.19)$$

이다. 또한

$$\begin{aligned} &det[M_t(A(x))] \\ &= \prod_{j=1}^{t} Y_j(1 + Z_jx)Z_jx \prod_{j>i\geq 1}^{t} (Z_jx + Z_ix)^2 \\ &= \prod_{j=1}^{t} Y_jZ_jx \prod_{j=1}^{t}(1 + Z_jx) \prod_{j>i\geq 1}^{t} (Z_jx + Z_ix)^2 \\ &= \prod_{j=1}^{t} Y_jZ_jx \prod_{j>i\geq 1}^{t} (Z_j + Z_i)^2 x^{(3t''' - t)/2} \prod_{j=1}^{t}(1 + Z_jx) \\ &= det[M_t(S(x))]\, x^{(3t''' - t)/2} \prod_{j=1}^{t}(1 + Z_jx) \end{aligned} \quad (2.20)$$

가 된다. 식(2.20)에서 $\prod_{j=1}^{t}(1+Z_j x)$는 앞에서 정의한 오류위치다항식과 동일하다. 따라서 $det[M_t(S(x))]\neq 0$ 이라면, 오류위치번호의 역수 Z_j^{-1}은 $det[M_t(A(x))]=0$ 의 근이 된다. 즉, $det[M_t(A(x))]=0$ 일 때 x의 값이 오류위치번호의 역수이다.

오류위치번호의 역수 $x=Z_1^{-1}$에 대해서 $det[M_t(A(Z_1^{-1}))]=0$ 이며 식(2.19)에 x를 대입하면

$$\begin{aligned} &det[M_t\ (S(Z_1^{-1}))] \\ &= \prod_{j=1}^{t} Y_j Z_j Z_1^{-1} \prod_{j>i\geq 1}^{t}(Z_j Z_1^{-1}+Z_i Z_1^{-1})^2 \\ &= Y_1 \prod_{j=2}^{t} Y_j Z_j Z_1^{-1}(1+Z_j Z_1^{-1})^2 \\ &\quad \prod_{j>i\geq 2}^{t}(Z_j Z_1^{-1}+Z_j Z_1^{-1})^2 \end{aligned} \tag{2.21}$$

가 된다. 그런데 식(2.18)에서

$$B_1(Z_1^{-1})=\sum_{j=2}^{t} Y_j(1+Z_j Z_1^{-1})^2(Z_j Z_1^{-1})^2 \tag{2.22}$$

이므로

$$\begin{aligned} &det[M_{t-1}(B(Z_1^{-1}))] \\ &= \prod_{j=2}^{t} Y_j Z_j Z_1^{-1}(1+Z_j Z_1^{-1})^2 \prod_{j>i\geq 2}^{t}(Z_j Z_1^{-1}+Z_i Z_1^{-1})^2 \end{aligned} \tag{2.23}$$

이다. 따라서 식(2.21)과 식(2.23)으로부터 Z_1위치에서의 오류치 Y_1은

$$Y_1=\frac{det[M_t(S(Z_1^{-1}))]}{det[M_{t-1}(B(Z_1^{-1}))]} \tag{2.24}$$

로 계산된다. 같은 방법으로 다른 오류위치에 대해서도 구해진다. 따라서 오류위치에 대해서 다음과 같은 결과를 얻을 수 있다. 임의의 오류위치에 대해 x_i가 오류위치번호의 역수이면

$$det[M_t(A(x_i))]=0 \tag{2.25}$$

이고 그때의 오류위치는

$$Y_i=\frac{det[M_t(S(x_i))]}{det[M_{t-1}(B(x_i))]} \tag{2.26}$$

이다.

이상에서 알아본 바와 같이 직접복호에서는 오증으로부터 곧바로 오류위치와 오류치를 계산할 수 있다. 복호를 위해서는 $det[M_t(S(x))]$, $det[M_t(A(x))]$, $det[M_{t-1}(B(x))]$를 계산해야 하는데 이는 식(2.25), (2.26)에 x_i대신 유한체의 원소 $\alpha, \alpha^2, \cdots, \alpha^{2^m-1}$를 차례로 대입하면서 (2.25)식을 만족하는 x_i를 (2.26)식에 대입하여 오류치를 구하는 것이다.

Ⅲ. 정규기저

유한체(finite field) $GF(2^m)$는 RS 부호의 해석에서 가장 중요한 부분을 차지하고 있다. 일반적으로 유한체 $GF(2^m)$의 원소들은 다항식 표현, 벡터 표현 또는 지수표현으로 나타낸다. $GF(2^m)$의 선형독립인 m개의 원소를 사용하여 임의의 원소를 다항식으로 표현할 수 있을 때 이 m개의 원소를 $GF(2^m)$의 기저(basis)라 한다. 유한체 $GF(2^m)$의 원시원소를 α라 하면 임의의 원소 β는 $GF(2)$ 상의 다항식

$$\beta = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{m-1}\alpha^{m-1} \tag{3.1}$$

로 표현할 수 있으며 $M=\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$를 유한체의 표준기저(standard basis)라 한다.

유한체 상에서의 연산에서 두 원소의 덧셈은 비트별 XOR에 의해 계산할 수 있으므로 용이하지만 곱셈 또는 나눗셈은 구현이 복잡하다. 그러나 정규기저(normal basis)를 사용하면 연속적인 순회치환(cyclic shift)에 의해 구현의 복잡함을 해결할 수 있다. $GF(2^m)$의 원소 중 m개의 원소 $N=\{\delta, \delta^2, \delta^4, \cdots, \delta^{2^{m-1}}\}$은 $GF(2^m)$의 기저가 될 수 있으며 임의의 원소 β는 다음과 같이 $GF(2)$상의 다항식으로 유일하게 표현된다.

$$\beta = b_0\delta + b_1\delta^2 + b_2\delta^4 + \cdots + b_{m-1}\delta^{2^{m-1}} \tag{3.2}$$

이 m개의 원소를 유한체 $GF(2^m)$의 정규기저라 한다. (3.2)식과 같이 표현된 $GF(2)$상의 다항식을 자승하면,

$$\beta^2 = b_{m-1}\delta + b_0\delta^2 + b_1\delta^4 + \cdots + b_{m-2}\delta^{2^{m-1}} \tag{3.3}$$

가 되며 β를 m차원 벡터 $\beta=(b_0, b_1, b_2, \cdots, b_{m-1})$로 표현하면 그것의 자승은 $\beta^2=(b_{m-1}, b_0, b_1, \cdots, b_{m-2})$

로 된다. 즉, 정규기저 표현에서 β^2을 얻기 위해서는 β를 한번 순회치환(cyclic shift)하면 된다. 따라서 정규기저를 사용하면 $GF(2^m)$의 연산에서 임의의 원소의 자승은 간단한 논리회로에 의해 구현된다.

$\lambda=(a_0, a_1, \cdots, a_{m-1})$와 $\mu=(b_0, b_1, \cdots, b_{m-1})$가 정규기저로 표현된 $GF(2^m)$의 두 원소라 하면 두 원소의 곱 $\nu=(c_0, c_1, \cdots, c_{m-1})$의 마지막요소 c_{m-1}은 λ와 μ의 요소들의 2진함수

$$c_{m-1}=f(a_0, a_1, \cdots, a_{m-1}; b_0, b_1, \cdots, b_{m-1}) \quad (3.4)$$

의 결과이다. 그리고 정규기저 표현에서 임의의 원소의 자승은 한번의 순회치환으로 이루어지므로 ν^2의 마지막 요소 c_{m-2}는 λ^2와 μ^2를 수행한 후에 나타나는 요소들을 사용하여 (2.4)식과 같은 논리함수에 의해 구해진다. 즉,

$$c_{m-2}=f(a_{m-1}, a_0, \cdots, a_{m-2}; b_{m-1}, b_0, \cdots, b_{m-2}) \quad (3.5)$$

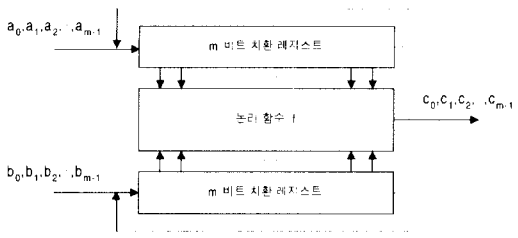
가 된다. 따라서 λ와 μ를 순회치환하여 그것의 자승한 값을 만들면서 요소들을 함수 f의 입력으로 삼으면 두 원소의 곱을 그림 3.1과 같은 방법으로 구할 수 있으며 이를 Massey-Omura 승산기[1]라 한다.

유한체 $GF(2^m)$의 임의의 원소에 어떤 원소를 나눈다는 것은 그 원소의 역원을 곱하는 것과 같다. $GF(2^m)$의 원소 β에 대해 이것의 역원은 $\beta^{-1}=\beta^{2^m-2}$이다. 그리고

$$2^m-2=2+2^2+2^3+\cdots+2^{m-1} \quad (3.6)$$

이므로 β의 역원 β^{-1}은 다음 식과 같이 표현할 수 있다.

$$\beta^{-1}=(\beta^2)(\beta^{2^2})(\beta^{2^3})\cdots(\beta^{2^{m-1}}) \quad (3.7)$$



〔그림 3.1〕 $GF(2^m)$상의 Massey-Omura 승산기

따라서 β가 정규기저로 표현되면 순회치환에 의해 자승한 값을 얻을수 있으므로 어떤 원소의 역원은 $m-1$번의 연속적인 순회치환과 앞에서 언급한 Massey-Omura 승산기를 이용하여 구할 수 있다. 두 치환 레지스터에 역원을 구하고자하는 유한체의 원소 β를 한번 순회치환한 β^2과 1을 각각 초기값으로 하여 승산하면 β^2이 구해진다. 승산한 결과를 귀환시켜 1을 β^2로 대치시키고 β^2을 한번 순회치환하여 β^{2^2}이 되도록 한 후 승산하면 $(\beta^2)(\beta^{2^2})$이 구해진다. 계속해서 같은 방법으로 전체 $m-1$번의 연속적인 순회치환에 의해 정규기저로 표현된 유한체의 역원을 계산할 수 있다.

이상에서 보듯이 유한체의 원소를 정규기저로 표현하면 표준기저로 표현할 때보다 하드웨어 구현시 훨씬 간단하게 곱셈과 나눗셈이 수행된다. 그러나 정규기저의 발견이 쉽지 않으며 정규기저 중에서도 최소의 곱셈항을 갖는 논리함수에 의해 승산이 수행될 수 있는 최적의 정규기저[6]를 발견하여야 한다. 최적의 정규기저의 경우 곱셈항의 수는 표준기저를 사용한 것에 비하면 절반에 해당한다. 본 논문에서는 (128,124) RS부호의 복호기 해석을 위해 원시다항식(primitive polynomial)이 $p(x)=1+x+x^2+x^7+x^8$일 때 $GF(2^8)$의 모든 정규기저를 구하고 이중에서 연산의 복잡도를 최소화할 수 있는 정규기저를 찾아서 RS 부호의 복호과정에 필수적인 유한체의 연산에 적용하였다. 표 3.1에서와 같이 $GF(2^8)$의 정규기저 중 곱셈항의 수를 최소화하여 연산의 복잡도를 줄일 수 있는 경우는 원시원소가 $\delta=\alpha^{19}$일 때이다. 예를 들어 정규기저로 표현된 임의의 두 원소를

$$\lambda=a_0\delta+a_1\delta^2+a_2\delta^4+a_3\delta^8+a_4\delta^{16}+a_5\delta^{32}+a_6\delta^{64}+a_7\delta^{128} \quad (3.8)$$

$$\mu=b_0\delta+b_1\delta^2+b_2\delta^4+b_3\delta^8+b_4\delta^{16}+b_5\delta^{32}+b_6\delta^{64}+b_7\delta^{128} \quad (3.9)$$

라 하면, 두 원소의 곱인 $\nu=\lambda\mu$의 마지막요소 c_7을 계산할 수 있는 2진함수는

$$\begin{aligned} c_7=&a_0b_5+a_0b_6+a_1b_3+a_1b_5+a_2b_1+a_2b_5\\ &+a_2b_6+a_2b_7+a_3b_1+a_3b_4+a_4b_2+a_4b_3\\ &+a_5b_0+a_5b_1+a_5b_2+a_5b_6+a_6b_0+a_6b_2\\ &+a_6b_5+a_6b_6+a_7b_2 \end{aligned} \quad (3.10)$$

이며, 곱셈항의 수는 21개로 최소이다.

〔표 3.1〕 $p(x)=1+x+x^2+x^7+x^8$ 일 때의 정규기저와 곱셈항 수

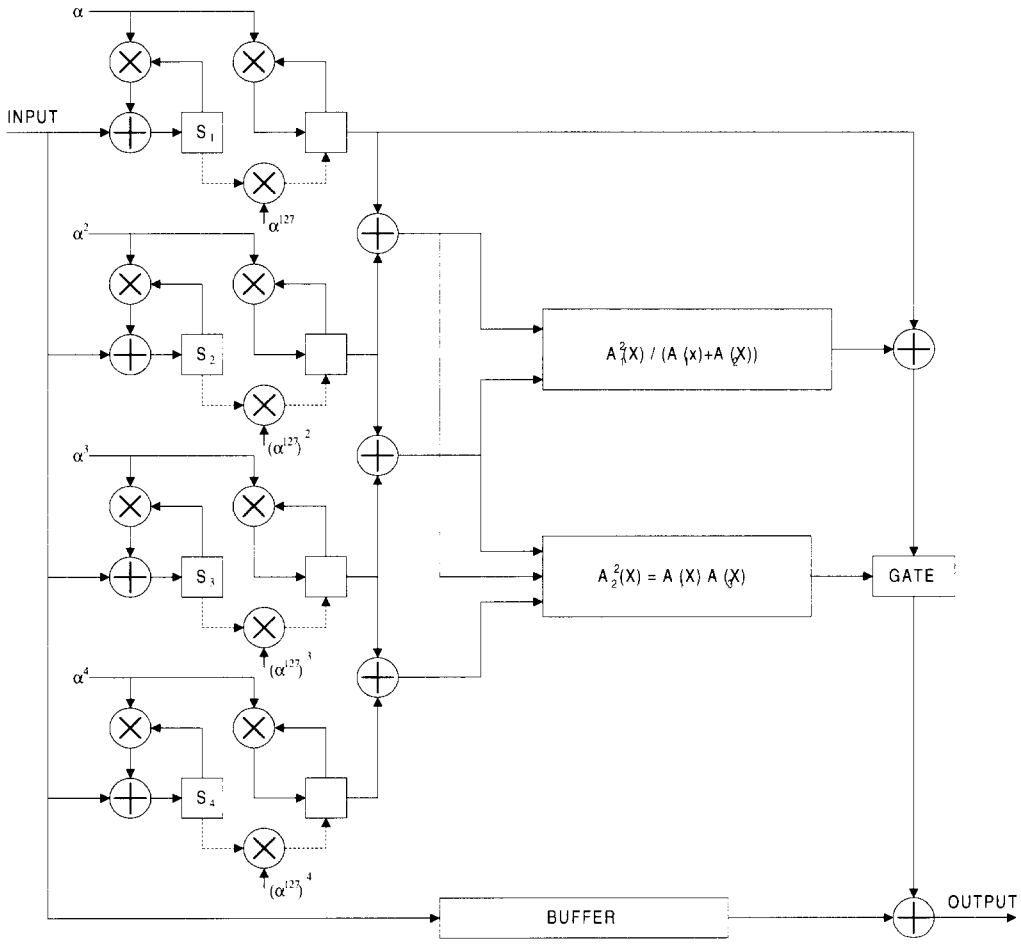
곱셈항의 수	원시원소의 멱
21	19
25	111
27	15, 27, 91, 127
29	1, 7, 87, 95
31	3, 5, 61
33	21, 55
35	37

Ⅳ. (128,124) RS 부호의 복호기 및 시뮬레이션

지금까지 전개하였던 정규기저와 직접복호법에 대한 이론들을 ATM 적응계층에 적용 가능한 (128,124) RS 부호에 적용하여 복호기를 설계한다. 두 개 이하의 심벌에 발생한 오류를 정정할 수 있는 (128,124) RS 부호는 (255,251) RS부호를 127심벌 단축시킨 부호로서 유한체 $GF(2^8)$ 상에서 해석되며 생성다항식은

$$g(x)=\prod_{i=1}^{4}(x+\alpha^i)$$
$$=x^4+\alpha^{43}x^3+\alpha^{52}x^2+\alpha^{48}x+\alpha^{10} \qquad (4.1)$$

이다. 직접복호법에 의한 (128,124) RS부호의 복호기 블록도는 그림 4.1과 같다. 이 복호기는 두 개 이하의 심벌에 발생한 오류를 정정할 수 있으므로 오증은 4개의 유한체 $GF(2^8)$ 원소로 구해진다. 한편, 이 부호는 127심벌을 단축시킨 것이므로 구해진 오증요소에 α^{127}, $(\alpha^{127})^2$, $(\alpha^{127})^3$, $(\alpha^{127})^4$를 각각 곱한



〔그림 4.1〕 직접복호법에 의한 (128,124) RS부호의 복호기 블록도

값을 초기값으로 하여 오류위치와 오류치를 구한다. 오류위치는 (2.20)식에 의해서 구할 수 있다.

$$det[M_2(A(x))]=\begin{vmatrix} A_1(x) & A_2(x) \\ A_2(x) & A_3(x) \end{vmatrix} \quad (4.2)$$

이 식을 전개하여 $det[M_2(A(x))]=0$을 만족하면 이것이 오류위치번호의 역수이다.

$$A_1(x)A_3(x)+A_2^2(x)=0 \quad (4.3)$$

$$A_2(x)^2=A_1(x)A_3(x) \quad (4.4)$$

즉, 식(4.4)의 근이 오류위치번호의 역수이다. 식(4.4)에 유한체 $GF(2^8)$의 원소 $\alpha, \alpha^2, \cdots, \alpha^{2^m-1}$를 차례로 대입하면서 근을 구하면 근의 역수가 바로 오류위치이다. 복호기의 동작에서는 (4.4)식을 만족하는 경우 수신계열에 오류치를 XOR하여 오류를 정정할 수 있도록 구현하였다. 오류치는 식(2.26)에 의하여

$$Y=\frac{det[M_2(S(x))]}{det[M_1(B(x))]} \quad (4.5)$$

와 같이 표현된다. 그러므로

$$det[M_2(s(x))] = \begin{vmatrix} S_1(x) & S_2(x) \\ S_2(x) & S_3(x) \end{vmatrix} = S_1(x)S_3(x)+S_2^2(x) \quad (4.6)$$

$$det[M_1(B(x))]=B_1(x)=S_1(x)+S_3(x) \quad (4.7)$$

이므로, 식(4.5)는

$$Y=\frac{S_1(x)S_3(x)+S_2^2(x)}{S_1(x)+S_3(x)} \quad (4.8)$$

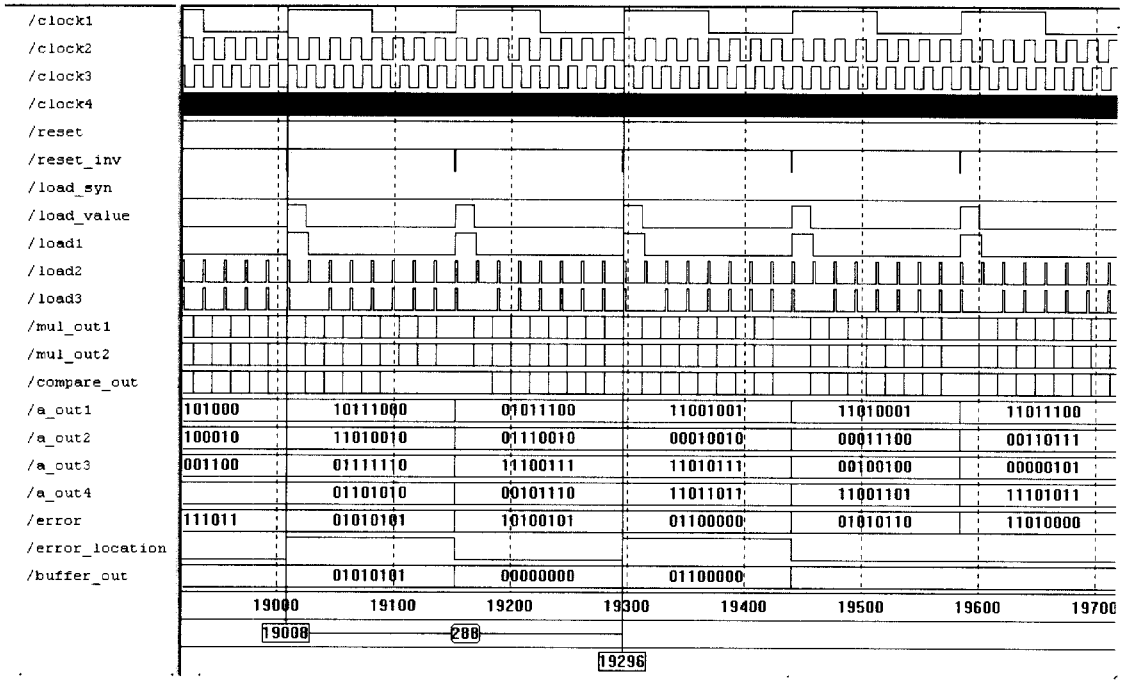
과 같고, 식(4.8)을 변형하면

$$Y=\frac{S_1^2(x)+S_2^2(x)}{S_1(x)+S_3(x)}+S_1(x) \quad (4.9)$$

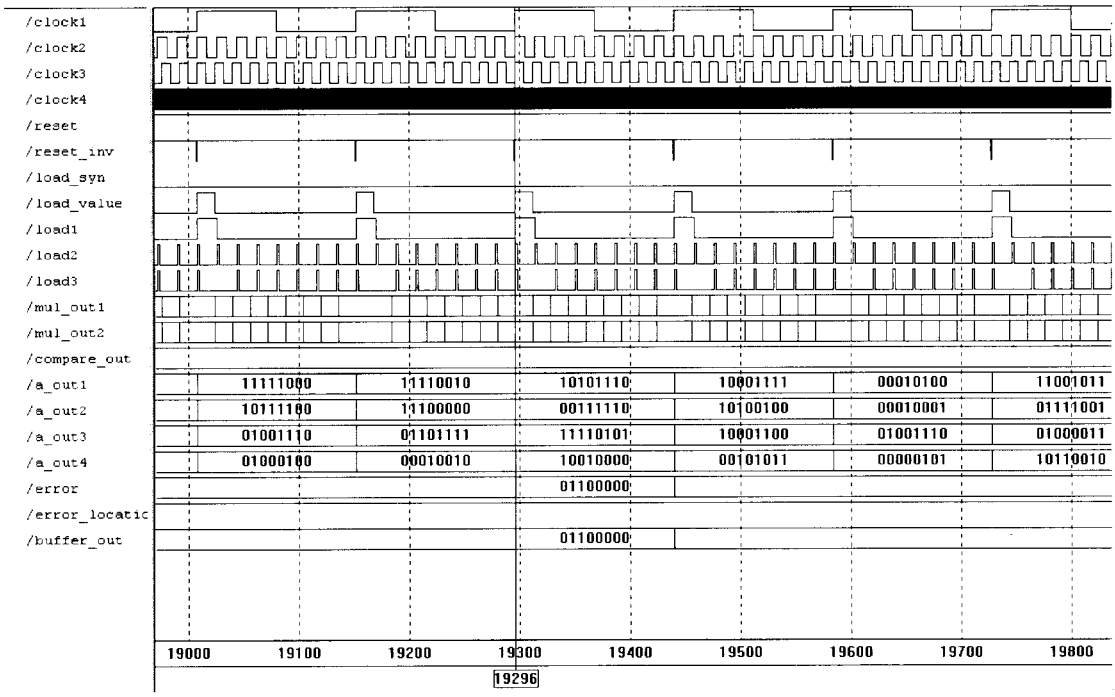
와 같이 되며 이 식은 다시

$$Y=\frac{A_1^2(x)}{A_1(x)+A_2(x)}+S_1(x) \quad (4.10)$$

로 쓸 수 있다. (4.10) 식에 위에서 구한 오류위치



〔그림 4.2〕 2중 오류일 경우의 복호기 시뮬레이션



〔그림 4.3〕 단일 오류일 경우의 복호기 시뮬레이션

번호의 역수를 대입하여 식을 전개하면 오류치가 구해진다.

이상에서 기술한 복호과정을 MyVHDL로 시뮬레이션하였다. 시뮬레이션은 오증을 계산하는 과정, 오류위치를 계산하는 과정, 오류치를 계산하는 과정으로 나누어 진행되었으며, 본 논문에서는 모든 심벌이 '0'인 부호어를 전송하고 x^{126}위치에 α^{85}(01010101), x^{124}위치에 α^{22}(01100000)와 같은 형태의 두 개의 심벌 오류가 발생한 경우를 가정하여 입력이 완료된 후 오류가 발생한 위치에서 구해진 오류치가 XOR되어 완전히 복호된 출력이 나타남을 시뮬레이션하였다. 시뮬레이션을 위해 심벌 클럭 clock1(본 논문에서는 144ns 클럭을 사용하였으며 하나의 심벌 클럭 동안 곱셈 및 나눗셈이 수행된다.), 곱셈을 위한 clock2, 그리고 나눗셈을 위한 clock3과 clock4가 필요하며, 그림 4.2에서 보는 바와 같이 복호기에서 계산되는 오류치 error는 항상 변화하지만 오류가 발생한 위치에서만 error_location이 '1'이 되고 이때 error가 buffer_out의 수신계열과 XOR되어 오류가 정정됨을 확인할 수 있다. 복호기의 시뮬레이션에서 주목할 점은 최적의 정규기저로 표현된 유한체상의 곱셈 또는 나눗셈을 한번의 심벌 클럭내에 계산되도록 하여 모든 수신계열이 buffer에 축적되면서 구해지는 오증으로부터 오류치가 계산될 때까지 3번의 심벌 클럭이 필요하므로 복호하기까지는 3회의 심벌 클럭 지연이 생긴다. 본 논문에서는 144ns(심벌 클럭)에 오증계산이 시작되어 127개의 심벌 클럭이 지난 후(18576ns) 오증계산이 완료되고 132번째 심벌 클럭이 시작되는 시점(19008ns)부터 복호가 수행되도록 하였다.

특히 직접복호법을 사용하면 복호회로의 수정 없이 단일 오류도 정정된다. 그림4.3에서 보듯이 x^{124} 위치에 α^{22}(01100000)와 같은 오류가 발생한 경우, error_location은 항상 '1'이지만 error는 오류가 발생한 위치에서만 값을 가지며 나머지 위치에서는 항상 '0'이므로 동일한 복호회로를 사용하여 2중 오류뿐 아니라 단일 오류도 정정 가능하다.

V. 결 론

RS 부호의 복호에서는 일반적으로 오류위치다항식을 구하고 그것의 근으로부터 오류위치를 구한 후 오류치를 구하는 알고리즘으로 복호하고 있다. 본 논문에서는 오증이 계산되면 곧바로 오류위치와 오류치를 구할 수 있는 직접복호법을 이용하여 ATM 적응계층에 적용 가능한 (128,124) RS부호의 복호

기를 설계하고 이를 VHDL로 시뮬레이션하였다. 부호의 복호에서 중요한 부분을 차지하는 유한체 연산의 복잡도를 줄이기 위해 곱셈항의 수를 최소화할 수 있는 최적 정규기저를 이용하여 유한체 $GF(2^8)$를 형성하였다. 이는 곱셈항을 표준기저의 절반으로 줄일 수 있으며 치환 레지스터를 이용하여 간단히 연산이 해결된다. 그리고 동일한 복호회로를 사용하여 2중 오류 및 단일 오류를 정정할 수 있는 장점이 있으며 구해진 오증요소에 유한체의 원소를 곱하는 동작과 유한체 상의 곱셈 및 나눗셈이 필요하므로 오증이 구해진 후 복호하기 까지는 3번의 심벌 클럭이 필요하다.

참 고 문 헌

[1] C. C Wang, T. K. Truong, H. M. Shao, J. K. Omura and I. S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$", IEEE Trans. Computers, Vol. C-34, pp. 709-716, 1985.

[2] Chien, R. T., "Cycle Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes", IEEE Trans. Inf. Theory, IT-10, pp. 357-363, 1964.

[3] Horiguchi, T., and Y. Sato, "A Decoding Method for Reed-Solomon Codes over $GF(2^m)$", Trans. IECE(Jpn.), pp.97-98, 1983.

[4] 염흥열, "AAL-1에 적용 가능한 (128,124) RS 부호의 복호 알고리즘과 FPGA 실현", 통신정보보호학회논문집, Vol. 7, No. 1, pp. 33-44, 1997.

[5] 강경식, 박진수, "5중 오류정정 (31,21) RS부호의 효율적인 복호 알고리즘과 VHDL 시뮬레이션", 통신정보보호학회논문집, Vol. 8, No. 2, pp. 93-106, 1998.

[6] 김창규, "유한체 GF(2m)의 고속 승산을 위한 최적 정규기저", 동의논집 제28집 자연과학편(II), pp. 373-380, 1998.

[7] A. J. Menezes, Application of Finite Fields, Kluwer Academic Pub., Norwell, Mass. 1993.

[8] M. Y. Rhee, Error-Correction Coding Theory, New York, McGraw-Hill, 1989.

[9] S. B. Wicker, Error Control Systems for Digital Communication and Storage, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

〈著者紹介〉

김 창 규 (Chang-kyu Kim) 정회원

1981년 2월 : 한양대학교 전자통신공학과 졸업
1984년 8월 : 한양대학교 전자통신공학과 석사
1989년 8월 : 한양대학교 전자통신공학과 박사
1988년 3월 ~ 현재 : 동의대학교 전자통신공학과 교수
1999년 ~ 현재 : 동의대학교 정보통신연구소 소장
〈관심분야〉 부호이론, 암호이론