通信情報保護學會論文誌
第 7 卷 第 1 號. 1997. 3

AAL-1 에 적용가능한 (128, 124) RS 부호의 복호 알고리즘과 FPGA 실현

염 흥 열*

Decoding Algorithm of (128,124) RS Code for AAL-1 and Its FPGA Implementation

Heung-Youl Youm

요 약

BISDN(Broadband Integrated Service Digital Network)의 AAL-1(ATM Adaptation Layer-1)에서는 오류정정능력이 2인 (128,124) RS(Reed Slolmon) 부호를 이용하여 ATM 셀에서 발생하는 오류를 정정하고 있다. 본 논문에서는 기존의 RS 복호 알고리즘을 분석한 후, 이를 바탕으로 AAL-1 기본 오류정정 모드에 적용 가능한 복잡도가 낮고 고속 동작이 가능한 복호 알고리즘을 제시하고, 부호기와 복호기를 VHDL로 부호화하고 설계한 후, 관련 회로를 시뮬레이션한다. 또한 시뮬레이션된 회로를 XACT을 이용하여 XC 4025 FPGA에 실현하여 제안되 복호 알고리즘의 타당성을 확인한다.

Abstract

The (128,124) RS(Reed-Solomon) codes with 2 error-correcting-capability is used for the error correction scheme of the ATM cell, which is recommended in I.363 Rec. for Broadband ISDN. Typical decoding algorithm for RS codes are Berlekamp-Massey iterative algorithm, Chien's search algorithm, Forney's decoding algorithm, and Perterson-Gorenstein-Zierler algorithm. We propose the efficient decoding algorithm for (128,124) RS codes, and also present basic formula for multiplication circuit in GF(28). We design the encoder and decoder pairs of (128,124) RS codes, and simulate the designed encoder and decoder by using the VHDL. We implement the encoder and decoder with the XC-4025 FPGA. And technical data for the implemented decoder are presented.

* 순천향대학교 전기전자공학부

1. 서 론

AAL-1(ATM Adaptation Layer-1)은 CS (Convergence Sublayer)와 SAR(Segmentation and Reassembly) 부계층으로 구성된다.[1] AAL-1에서의 오류정정 방식은 지연이 매우 큰 오류정정능력이 2인 (128, 124) RS 부호를 이용하는 방식과 지연이 작은 오류정정능력이 3인 (94, 88) RS 부호를 이용하는 방식이 있다. AAL-1에서의 (128, 124) RS 부호는 전체 47 개의 ATM 셀에서 발생하는 4개의 셀 손실(삭제)을 갖는 오류 형태, 47개의 셀에서 2개의 셀 손실과 각 열에서 하나 이하의 심벌 오류를 갖는 오류 형태, 그리고 128개의 심벌중 2심벌 이하의 오류 형태를 정정할 수 있는 오류정정능력이 우수한 부호이다. AAL-1 에서의 오류정정은 (128, 124) RS 부호의 부호기, 인터리버 및 디인터리버, 그리고 복호기로 실현된다. AAL-1에서의 오류정정 모드는 셀에서 발생하는 랜덤 오류만을 정정하는 기본 모드와 셀 손실까지를 정정할 수 있는 확장 오류정정 모드가 있다. AAL-1계층상의 오류정정회로는 MPEG 2 최대 속도를 고려하여 16Mbps의 데이타 속도로 동작되어야 한다. 즉, 부호기, 복호기, 그리고 인터리버/디인터리버는 적어도 16Mbps 데이타 속도로 동작 가능해야 한다. (128, 124) RS 부호는 (255, 251) RS 부호를 127심벌 단축한 단축(Shortened) RS 부호이다. 그리고 생성다항식의 근이 α^{120}에서 α^{123}이다. (255, 251) RS부호의 복호기는 단축 특성과 근의 특성을 고려하여 설계되어야 한다. 본 논문에서 제안된 복호 알고리즘은 상기의 특성이 고려되었다.

본 논문에서는 기존의 RS 복호 알고리즘을 분석한 후,[5, 6, 7] 이를 바탕으로 AAL-1기본 오류정정 모드에 적용 가능한 오류정정능력이 2인 RS 부호의 복잡도가 낮고 고속 동작이 가능한 복호 알고리즘을 제시하고, 부호기와 복호기를 VHDL로 부호화하고 설계한 후, 관련 회로를 시뮬레이션한다. 또한 시뮬레이션된 회로를 XACT 을 이용하여 XC-4025 FPGA에 배치 및 배선하였으며, Exampler의 Time Explorer와 V시스템을 이용하여 게이트-레벨 시뮬레이션을 수행하여, 제안된 복호 알고리즘의 타당성을 확인한다. 또한 이를 XC-4025로 실현한 결과 모든 동작이 정상적으로 동작됨을 확인한다.

2. RS 부호의 복호 알고리즘 및 곱셈기

AAL-1에서 이용되는 RS 부호는 오류정정능력이 2이고 최소 거리가 5인 $GF(2^8)$상의 (128, 124) RS 부호를 이용한다. $GF(2^8)$을 생성하기 위한 원시다항식(Primitive Polynomial)은 식 (2.1)과 같으며, (128, 124) RS부호의 생성 다항식(Generator Polynomial)은 식 (2.2)와 같이 표준화되었다.[2, 3, 4]

$$p(x) = x^8+x^7+x^2+x+1 \qquad (2.1)$$

$$g(x) = \prod_{i=0}^{3}(x-\alpha^{i+k}),\ k=120$$

$$= (x+\alpha^{120})(x+\alpha^{121})(x+\alpha^{122})(x+\alpha^{123}) \qquad (2.2)$$

$$= (x^4+\alpha^{162}x^3+\alpha^{35}x^2+\alpha^{150}x+\alpha^{231}$$

RS 부호의 오증요소 s_i ($1 \le i \le l+2t-1$, l:정수) 를 식 (2.3)과 같이 정의한다.

$$s_{i-1} = r(\alpha^i) = e(\alpha^i),\ 1 \le i \le l+2t-1 \qquad (2.3)$$

오증요소 생성회로는 식 (2.4)에 바탕을 두고 실현될 수 있다.

$$s_{i-1} = (\cdots(((r_{n-1}\alpha^i+r_{n-2})\alpha^i+r_{n-3})\alpha^i+r_{n-4})\cdots+r_1)\alpha^i+r_0 \quad (2.4)$$

오류위치 x_{i1}을 X_1이라 표현하자. v개의 오류가 Xi, $1 \le i \le v$위치에서 발생하였다면 오류 형태는 식 (2.5)와 같다.

$$e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots + e_{i_v} x^{i_v} \qquad (2.5)$$

오류 위치 다항식(Error Location Polynomial)의 근을 구하기 위하여 다항식의 근이 오류 위치의 역수 X_1^{-1}, X_2^{-1}, ... , X_v^{-1} 인 오류 위치 다항식을 식 (2.6)과 같이 정의한다.

$$\begin{aligned}\sigma(x) &= (1+X_1x)(1+X_2x)\cdots(1+X_vx) \\ &= 1+\sigma_1x+\sigma_2x^2+\cdots+\sigma_vx^v \\ &= \prod_{i=1}^{v}(1+X_ix) \end{aligned} \qquad (2.6)$$

오증 요소 s_i와 오류 위치 다항식 $\sigma(x)$의 계수와의 관계는 식 (2.7)과 같은 행렬로 표현된다.

$$\begin{bmatrix} S_v \\ \cdots \\ S_{2v-2} \\ S_{2v-1} \end{bmatrix} = \begin{bmatrix} S_0 & S_1 & \cdots & S_{v-2} & S_{v-1} \\ S_1 & S_2 & \cdots & S_{v-1} & S_v \\ & \cdots & & & \\ S_{v-1} & S_v & \cdots & S_{2v-3} & S_{2v-2} \end{bmatrix} \cdot \begin{bmatrix} \sigma_v \\ \cdots \\ \sigma_{v-1} \\ \sigma_1 \end{bmatrix} \qquad (2.7)$$

식 (2.7)을 이용하여 $\sigma(x)$의 계수를 구하면 v개의 오류에 대한 오류 위치 다항식의 계수를 구할 수 있다. 오류 위치 다항식 $\sigma(x)$의 근이 오류 위치 번호의 역수이므로 $GF(2^m)$의 모든 2^m-1개의 원소를 오류 위치 다항식 $\sigma(x)$의 x에 대입하고 $\sigma(\alpha^t)=0$, $0 \le t \le 2^m-1$이 되면, α^t는 $\sigma(x)$의 근이 된다. 그러나 실제 발생한 오류 위치는 α^{n-t}이다. 일반적으로 i번째 오증 요소는 식 (2.8)과 같다.

$$S_i = \sum_{k=1}^{v} Y_k X_k^i \qquad (2.8)$$

오증 요소들로 실현된 오증다항식은 식 (2.8)과 같이 정의된다.

$$s(x) = \sum_{i=0}^{\infty} S_i x^i \qquad (2.9)$$

식 (2.8)을 식 (2.9)에 대입하면 식 (2.10)을 얻을 수 있다.

$$\begin{aligned} s(x) &= \sum_{i=0}^{\infty}\{\sum_{k=1}^{v} Y_k(X^k)^i\}x^i \\ &= \sum_{k=1}^{v}\{\sum_{i=0}^{\infty} Y_k(X_kx)^i\} \\ &= \sum_{k=1}^{v}\frac{Y_k}{(1+X_kx)} \end{aligned} \qquad (2.10)$$

오류치 평가 다항식(Error Evaluator Polynomial), $\Omega(x)$를 식 (2.11)과 같이 정의한다.

$$\begin{aligned} \Omega(x) &= s(x)\sigma \bmod x^{2t} \\ &= \sum_{k=1}^{v}\frac{Y_k}{(1+X_kx)}\prod_{l=1}^{v}(1+X_lx) \\ &= \sum_{k=1}^{v} Y_k \prod_{l=1,l\neq k}^{v}(1+X_lx) \end{aligned} \qquad (2.11)$$

식 (2.11)에서 알 수 있듯이 오류 평가 다항식은 v-1 차 임을 알 수 있다. 따라서, 오류 평가다항식 $\Omega(x)$의 계수는 식 (2.12)를 이용한다.

$$\begin{aligned} \Omega_0 &= s_0 \\ \Omega_1 &= s_0\sigma_1+s_1 \\ \Omega_2 &= s_0\sigma_2+s_1\sigma_1+s_2 \\ &\cdots \\ \Omega_{v-1} &= s_0\sigma_{v-1}+s_1\sigma_{v-2}+\cdots s_{v-2}\sigma_1+s_{v-1} \end{aligned} \qquad (2.12)$$

식 (2.11)로부터 X_k의 위치에 대응되는 오류치는 식 (2.13)을 이용하여 구할 수 있다.

$$Y_k = \frac{\Omega(X_k^{-1})}{\prod_{i=1, i\neq k}^{v}(1+X_iX_k^{-1})} \qquad (2.13)$$

따라서, 식 (2.13)를 이용하면 오류 위치에 대응되는 오류치를 구할 수 있다. 한편, GF(2^8)상에서의 곱셈기는 복호기의 복잡도에 심각한 영향을 미친다. 복호기 설계시 이용된 GF(2^8)상의 곱셈기를 설계하기 위한 바탕식을 제시한다. GF(2^8)상의 두 원소를 a, b라 할 때, 두 원소를 곱한 결과를 식으로 표현하면 식 (2.14)와 같다. 식 (2.14)는 곱셈기 설계시 이용될 수 있다.

$$\begin{aligned}
a \cdot b = {} & (a_0+a_1\alpha+a_2\alpha^2+a_3\alpha^3+a_4\alpha^4+a_5\alpha^5+a_6\alpha^6+a_7\alpha^7)\cdot \\
& (b_0+b_1\alpha+b_2\alpha^2+b_3\alpha^3+b_4\alpha^4+b_5\alpha^5+b_6\alpha^6+b_7\alpha^7) \\
= {} & (a_0\cdot b_0+a_1\cdot b_7+a_2\cdot b_6+a_3\cdot b_5+a_4\cdot b_4+a_5\cdot b_3 \\
& +a_6\cdot b_2+a_7\cdot b_1+a_2\cdot b_7+a_3\cdot b_6+a_4\cdot b_5+a_5\cdot b_4 \\
& +a_6\cdot b_3+a_7\cdot b_2+a_3\cdot b_7+a_4\cdot b_6+a_5\cdot b_5+a_6\cdot b_4 \\
& +a_7\cdot b_3+a_4\cdot b_7+a_5\cdot b_6+a_6\cdot b_5+a_7\cdot b_4+a_5\cdot b_7 \\
& +a_6\cdot b_6+a_7\cdot b_5+a_6\cdot b_7+a_7\cdot b_6) \\
& +(a_0\cdot b_1+a_1\cdot b_0+a_1\cdot b_7+a_2\cdot b_6+a_3\cdot b_5+a_4\cdot b_4+a_5\cdot b_3 \\
& +a_6\cdot b_2+a_7\cdot b_1+a_7\cdot b_7)\alpha \\
& +(a_0\cdot b_2+a_1\cdot b_1+a_2\cdot b_0+a_1\cdot b_7+a_2\cdot b_6+a_3\cdot b_5 \\
& +a_4\cdot b_4+a_5\cdot b_3+a_6\cdot b_2+a_7\cdot b_1+a_3\cdot b_7+a_4\cdot b_6 \\
& +a_5\cdot b_5+a_6\cdot b_4+a_7\cdot b_3+a_4\cdot b_7+a_5\cdot b_6+a_6\cdot b_5 \\
& +a_7\cdot b_4+a_5\cdot b_7+a_6\cdot b_6+a_7\cdot b_5+a_6\cdot b_7+a_7\cdot b_6)\alpha^2 \\
& +(a_0\cdot b_3+a_1\cdot b_2+a_2\cdot b_1+a_3\cdot b_0+a_2\cdot b_7+a_3\cdot b_6 \\
& +a_4\cdot b_5+a_5\cdot b_4+a_6\cdot b_3+a_7\cdot b_2+a_4\cdot b_7+a_5\cdot b_6 \\
& +a_6\cdot b_5+a_7\cdot b_4+a_5\cdot b_7+a_6\cdot b_6+a_7\cdot b_5+a_6\cdot b_7+a_7\cdot b_6 \\
& +a_7\cdot b_7)\alpha^3 \\
& +(a_0\cdot b_4+a_1\cdot b_3+a_2\cdot b_2+a_3\cdot b_1+a_4\cdot b_0+a_3\cdot b_7 \\
& +a_4\cdot b_6+a_5\cdot b_5+a_6\cdot b_4+a_7\cdot b_3+a_5\cdot b_7+a_6\cdot b_6 \\
& +a_6\cdot b_7+a_7\cdot b_5+a_7\cdot b_6+a_7\cdot b_7)\alpha^4 \\
& +(a_0\cdot b_5+a_1\cdot b_4+a_2\cdot b_3+a_3\cdot b_2+a_4\cdot b_1+a_5\cdot b_0 \\
& +a_4\cdot b_7+a_5\cdot b_6+a_6\cdot b_5+a_7\cdot b_4+a_6\cdot b_7+a_7\cdot b_6)\alpha^5 \\
& +(a_0\cdot b_6+a_1\cdot b_5+a_2\cdot b_4+a_3\cdot b_3+a_4\cdot b_2+a_5\cdot b_1 \\
& +a_6\cdot b_0+a_5\cdot b_7+a_6\cdot b_6+a_7\cdot b_5+a_7\cdot b_7)\alpha^6 \\
& +(a_0\cdot b_7+a_1\cdot b_6+a_2\cdot b_5+a_3\cdot b_4+a_4\cdot b_3+a_5\cdot b_2 \\
& +a_6\cdot b_1+a_7\cdot b_0+a_1\cdot b_7+a_2\cdot b_6+a_3\cdot b_5+a_4\cdot b_4 \\
& +a_5\cdot b_3+a_6\cdot b_2+a_7\cdot b_1+a_2\cdot b_7+a_3\cdot b_6+a_4\cdot b_5 \\
& +a_5\cdot b_4+a_6\cdot b_3+a_7\cdot b_2+a_3\cdot b_7+a_4\cdot b_6+a_5\cdot b_5 \\
& +a_6\cdot b_4+a_7\cdot b_3+a_4\cdot b_7+a_5\cdot b_6+a_6\cdot b_5+a_7\cdot b_4 \\
& +a_5\cdot b_7+a_6\cdot b_6+a_7\cdot b_5)\alpha^7 \qquad (2.14)
\end{aligned}$$

3. (128,124) RS 부호의 효율적인 복호 알고리즘

(128, 124) RS 부호의 부호어의 길이는 128 옥텟이다. 47개의 128옥텟의 부호어는 채널 상에서 발생하는 연집 오류를 랜덤 오류로 변환하기 위한 용도인 인터리버를 거쳐서 47개의 셀로 사상된다. 이를 위한 FEC 블럭의 구조는 그림 3.1과 같다.[1] 그림 3.1에서 알 수 있듯이 47개의 연속적인 FEC(Forward Error Correction) 프레임들은 하나의 FEC블럭을 재구성된다. 각 프레임은 128옥텟으로 구성되므로 하나의 FEC블럭은 6,016옥텟으로 구성되어 있다. 하나의 FEC블럭은 CS-PDU(Convergence Sublayer Protocol Data Unit)이다. CS-PDU동기를 위하여 SAR-PDU의 CS 표시 비트(Indicator Bit)를 이용한다. AAL-1의 (128, 124) RS 부호는 그림 3.2와 같은 깊이가 47인 옥텟 인터리버(Octet Interleaver)를 이용한다. 따라서 옥텟 인터리버는 128개의 열과 47개의 행으로 구성된다. 인터리버는 셀 손실 또는 셀에서 버스트성으로 발생하는 오류를 랜덤화하는 역할을 수행한다. 따라서 인터리버는 (128, 124) RS 부호의 복호기가 각 열에서 2개 이하의 옥텟 오류, 2개의 셀 손실 및 하나의 옥텟 오류, 그리고 4개의 셀 손실을 교정할 수 있도록 한다.

I.363의 기본 모드에 적용 가능한 (128, 124) RS 부호의 복호 알고리즘을 제안한다.[7,8] 오류가 2개 발생했다고 가정하면 오류위치, 오

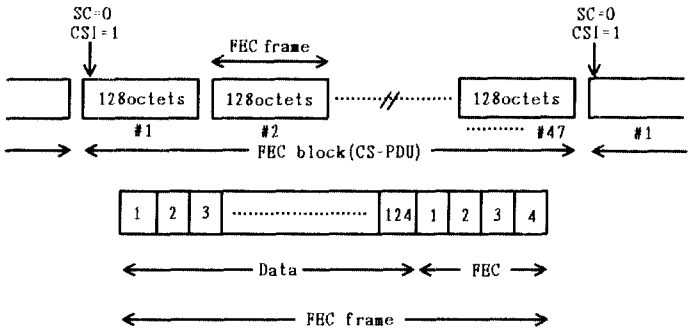

그림 3.1 (128,124) RS 부호의 FEC 블럭의 구조

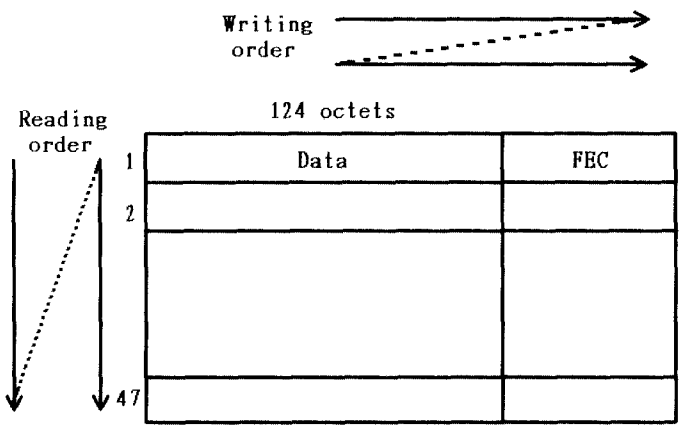

그림 3.2 (128,124) RS 부호의 인터리버

류치, 그리고 오증요소와의 관계는 식 (3.1)과 같다.

$$\begin{aligned} s_0 &= e_i(\alpha^{120\cdot i}) + e_j(\alpha^{120\cdot j}) \\ s_1 &= e_i(\alpha^{121\cdot i}) + e_j(\alpha^{121\cdot j}) \\ s_2 &= e_i(\alpha^{122\cdot i}) + e_j(\alpha^{122\cdot j}) \\ s_3 &= e_i(\alpha^{123\cdot i}) + e_j(\alpha^{123\cdot j}) \end{aligned} \qquad (3.1)$$

식 (3.1)을 변경하여 식 (3.2)와 같은 오류 정정능력이 2인 RS 부호의 오류 위치 다항식을 구할 수 있다.

$$(s_1\alpha^i+s_2) + \frac{s_0\alpha^i+s_1}{s_1\alpha^{2i}+s_2}(s_1\alpha^{2i}+s_3) = 0 \qquad (3.2)$$

복호 과정은 식 (3.2)에서의 i에 n-1에서 0까지의 수를 대입함으로서 개시된다. 만약 i가 오류 위치이면 식 (3.2)의 관계식을 만족한다. 따라서 식 (3.2)의 관계식이 만족되면 해당 위치는 오류 위치라 가정한다. 해당 위치에 대응되는 오류치는 식 (3.3)과 같이 구할 수 있다.

$$\begin{aligned} e_i &= \frac{1}{(\alpha^{120\cdot i})}\left[s_0 + e_j(\alpha^{120\cdot j})\right] \\ &= \frac{1}{(\alpha^{120\cdot i})}\left[s_0 + (s_0\alpha^i+s_1)\cdot\frac{s_0\alpha^i+s_1}{s_0\alpha^{2i}+s_2}\right] \end{aligned} \qquad (3.3)$$

따라서 식 (3.2)를 이용하여 오류 위치를 알아내고 식 (3.3) 을 이용하여 오류치를 계산

할 수 있다.

오류가 1개 발생한 경우의 오류 위치 다항식과 오류치는 각각 식 (3.4)와 식 (3.5)와 같다.

$$s_0\alpha^i+s_1=0 \qquad (3.4)$$

$$e_i = \frac{1}{(\alpha^{120})^i} \cdot s_0 \qquad (3.5)$$

복호 과정은 먼저 식 (3.4)에서 i에 n-1에서 0까지의 수를 대입함으로서 시작된다. 만약 i가 오류 위치이면 식 (3.4)의 관계식을 만족한다. 따라서 식 (3.4)를 이용하여 오류 위치를 알아내고 식 (3.5)를 이용하여 해당 오류 위치에 대응되는 오류치를 계산할 수 있다. 오증요소를 이용하여 오류의 갯수를 판별하는 알고리즘은 식 (3.6)을 이용한다.

$$\begin{aligned} A &= s_0s_2+s_1^2 \\ B &= s_1s_2+s_0s_3 \\ C &= s_1s_3+s_2^2 \end{aligned} \qquad (3.6)$$

오류가 2인 경우의 판별식은 식 (3.7)과 같다.

$$A\neq 0, B\neq 0, C\neq 0 \qquad (3.7)$$

오류가 한개 발생한 경우의 판별식은 식 (3.8) 과 같다.

$$A=0, B=0, C=0, s_0\neq 0, s_3\neq 0 \qquad (3.8)$$

오류가 없는 경우의 판별식은 식 (3.9)와 같다.

$$s_0=0, s_1=0\ s_2=0, s_3=0 \qquad (3.9)$$

식 (3.4)와 식 (3.5)을 이용하여 구현된 (128, 124) 복호기의 단일 오류 발생 시의 오류 정정 회로는 그림 3.3과 같다. 그리고 식 (3.2)와 식 (3.3)을 이용하여 실현된 이중 오류 발생 시의 오류 정정 회로는 그림 3.4와 같다. 그림 3.3에서 알 수 있듯이 오류정정능력이 1인 RS 부호의 복호기는 오증요소 계산부, 오증 요소 저장부, 127비트 단축 효과를 고려한 단축 효과 계산부, 오류 위치 계산 회로, 그리고 오류치 계산 회로로 구성될 수 있다. 여기에 오류의 갯수를 예측하는 오류 갯수 추정부와 복호기의 각 단에서 요구되는 타이밍

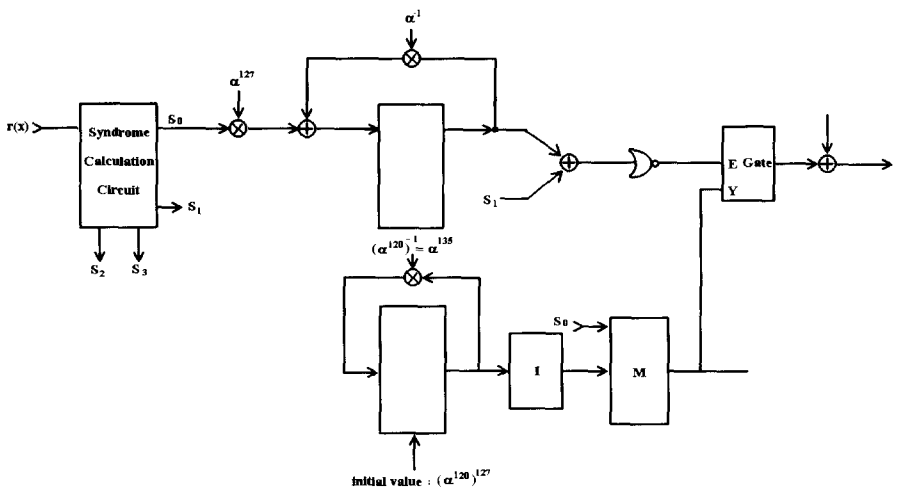

그림 3.3 (128, 124) RS 부호의 단일 오류 정정 회로

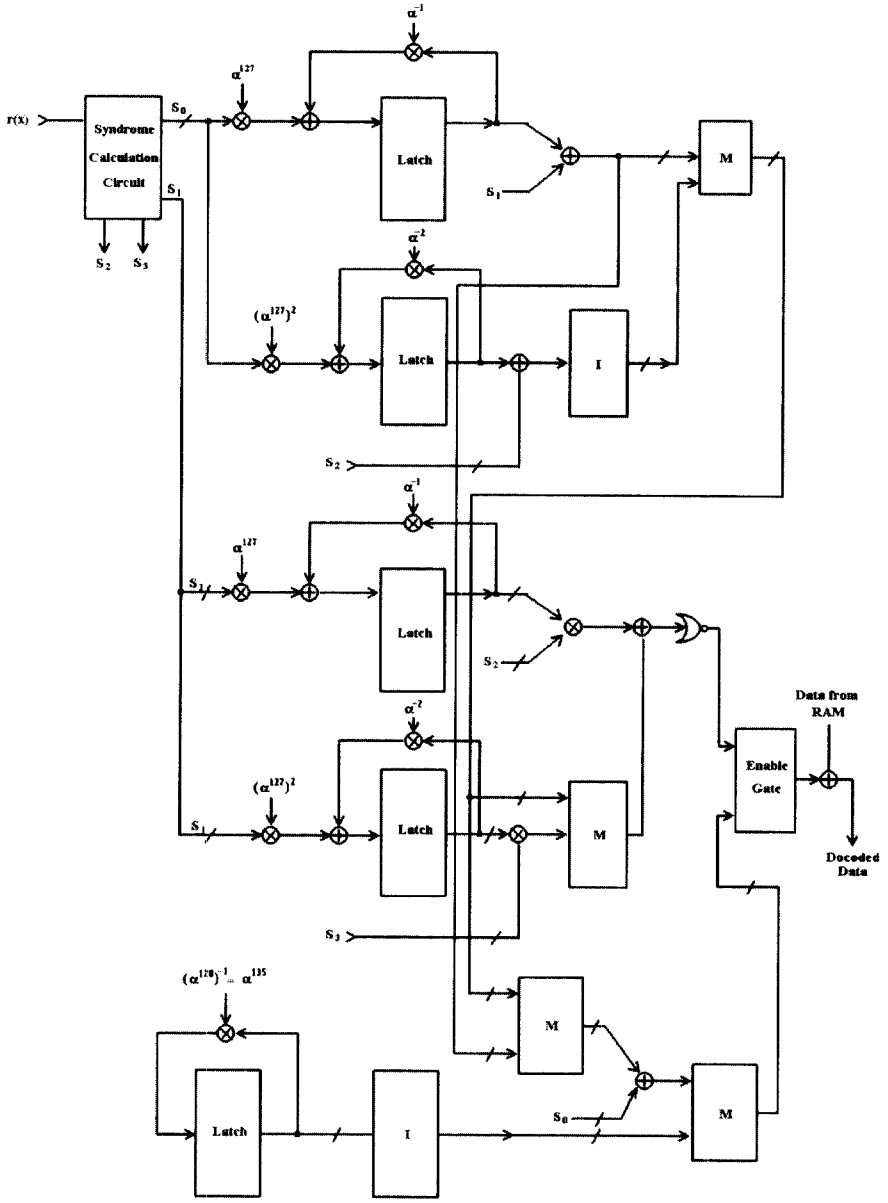

그림 3.4 (128, 124) RS 부호의 이중 오류정정회로

을 생성하는 제어 회로가 추가된다.

그림 3.4에서 알 수 있듯이 오류정정능력이 2인 RS 부호의 복호기는 오증요소 계산부, 오증 요소 저장부, 127비트 단축 효과를 고려한 단축 효과 계산부, 오류 위치 계산 회로, 그리고 오류치 계산 회로로 구성될 수 있다. 여기에 오류의 갯수를 예측하는 오류 갯수 추정부와 복호기의 각 단에서 요구되는 타이밍을 생

성하는 제어 회로가 요구된다. 그림 3.4와 같은 구조를 갖는 오류정정능력이 2인 (128, 124) RS 부호의 복호기는 132심벌 후에 오류를 정정하면서 부호 계열을 출력한다. 132심벌 지연중 128심벌 지연은 오증요소를 계산하기 위한 심벌 지연이고 나머지 4심벌은 복호기에서 발생하는 지연이다. 따라서 Chien회로를 사용한 경우의 지연보다 1프레임(128심벌)만큼 작다.

4. (128, 124) RS 부호의 VHDL 시뮬레이션 및 FPGA 실현

4.1 (128, 124) RS 부호의 부호기

시뮬레이션시에 기본 클럭은 100nsec 클럭을 이용하였다. 부호기 입력인 정보 계열이 12,800ns - 13,300ns 까지 "FF" 이고 13,300ns - 25,600ns 까지 "00" 일 경우, 부호기의 패리티 값은 "86", "24", "0C", "A6"가 되며, 이 패리티 심벌 들은 정보 계열 뒤에 부가되어 부호 계열을 형성한다. 부호기의 시뮬레이션 결과는 원래의 C 언어 시뮬레이션 결과와 정확히 일치함을 확인할 수 있었다.

4.2 (128,124) RS 부호의 복호기 시뮬레이션

복호기는 오증을 계산하기 위하여 반드시 요구되는 프레임 지연(128바이트 클럭)에 실제 오류를 정정하기 위해 요구되는 4바이트 클럭의 지연을 더하여 전체 132 바이트 클럭 후에 복호기 출력에서 오류가 정정됨을 확인하였다. (128, 124) RS 부호의 복호기 시뮬레이션시 사용된 내부 신호들은 다음과 같다.

- rx : 수신 계열
- clk : 바이트 클럭(Byte Clock)
- reset : Frame 클럭(128 바이트 클럭마다 발생)
- dcode_t : 복호기 출력 데이타(오류정정이 완료된 계열)
- syn0 - syn3 : 오증 요소들
- dc1 : 오류의 개수가 1인 경우의 추정된 오류값
- dc2 : 오류의 개수가 2인 경우의 추정된 오류값
- sylo : 프레임 단위로 오증 요소를 latch 하는 클럭
- syre : 오증 회로를 reset 하는 신호
- latlo : 변형된 오증을 latch 하는 클럭
- latre : 오류 위치 및 오류치를 찾기 위한 회로의 latch 신호
- rxdelay : 지연된 입력 rx
- ena_t1 : 오류가 한개인 경우의 판별 신호
- ena_t2 : 오류가 두개인 경우의 판별 신호
- cordata : 오류 개수 판별 알고리즘에 따른 추정된 오류 값

부호 계열의 첫 번째와 두 번째 심벌에서 오류가 발생했을 경우, 즉 그림 4.1.a 와 같은 타이밍에서 오류가 발생한 경우를 가정하고 오류 정정을 수행하였다. 여기서 복호기의 첫 번째 단에서는 클럭의 하강 에지에서 데이타를 래취함으로서 입력 데이타를 안정되게 래취하도록 구현되었다.

- 첫 번째 심벌 : "FF" => "F8"
- 두 번째 심벌 : "FF" => "F8"

이 경우 복호기 출력은 그림 4.1.b에서 알 수 있듯이 다음과 같은 타이밍에서 교정됨을 확인하였다.

- 첫 번째 심벌 : "F8" => "FF"
- 두 번째 심벌 : "F8" => "FF"

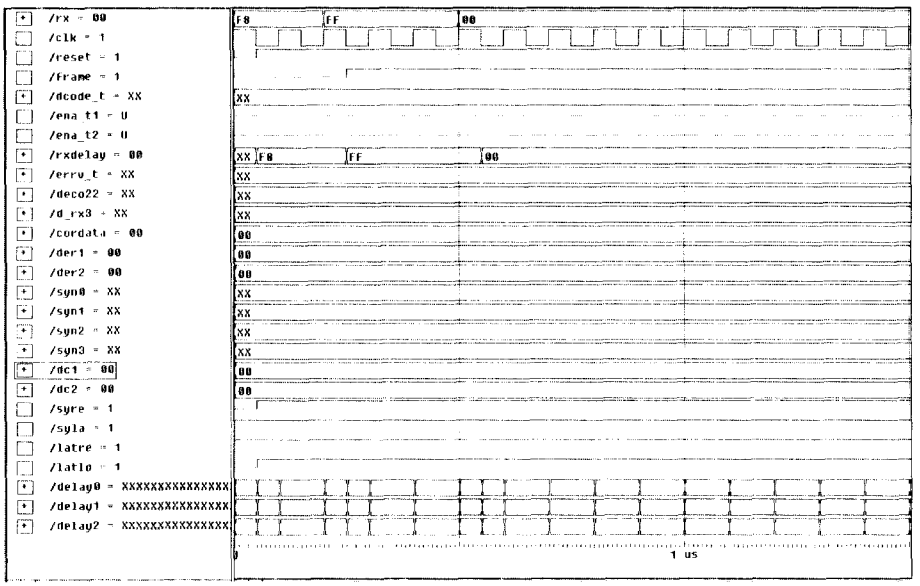

그림 4.1.a 오류가 2인 때의 복호기 입력 타이밍

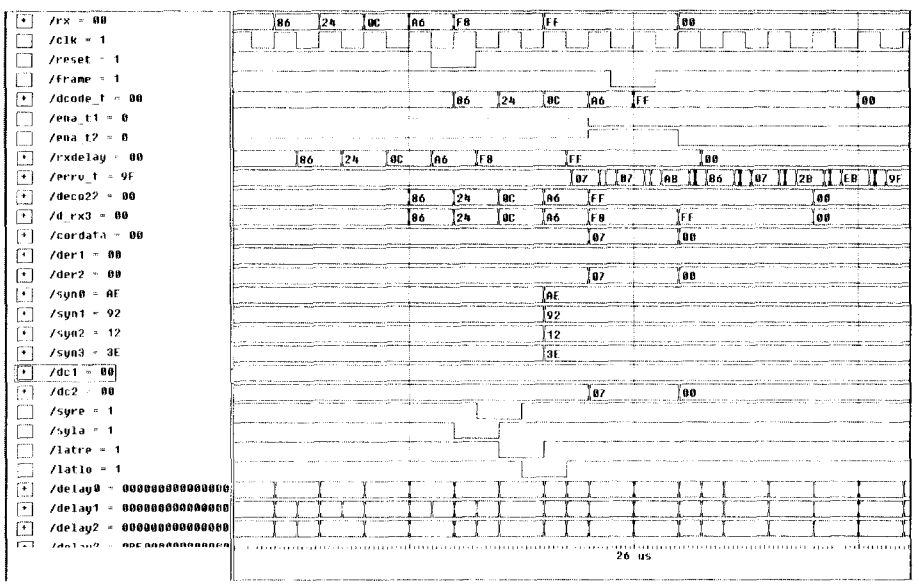

그림 4.1.b 오류가 2인 때의 복호기 출력 타이밍

4.3 실현 환경

논리 합성 및 타이밍 레벨 시뮬레이션을 위하여 Examplar 사의 Galileo 를 이용하였으며, Xilinx 사의 XACT 툴을 이용하여 배치 및 배선을 하였다. 이를 종합하면 표 4.1과 같다. FPGA 를 위한 VHDL 부호화는 PC 에서 수행하였고, 논리 합성은 워크스테이션에서 Galileo Logic

표 4.1 FPGA 실현 환경

품 명	용 도
Workstation(SPARC-10)	FPGA구현을 위한 워크스테이션
PC-486	VHDL 부호화
Examplar의 Galileo · Logic Synthesis · Timing Analysis · V System(Vital Library)	 회로 합성 타이밍 분석 게이트 레벨 시뮬레이션
PC version V system	VHDL 코딩
Xilinx 사의 XACT	XC4025에 배치 및 배선

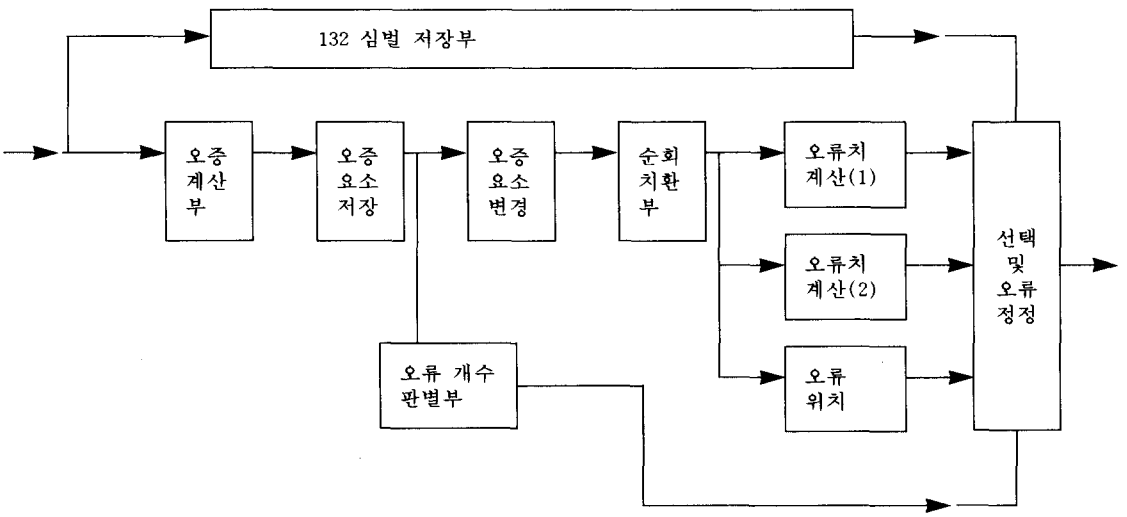

그림 4.2 (128,124) RS 복호기 블럭도

Explorer 을 이용하여 수행하였으며, Galileo Logic Explorer 의 출력 파일인 filename.xnf 파일을 Xilinx 배선 및 배치 툴인 XACT를 이용하여 배치 및 배선을 수행하였다. Xilinx배선 및 배치 툴은 타이밍 annotated XNF파일을 생성한다. Galileo Time Explorer 는 filename.xnf파일을 읽어서 관련 타이밍 정보를 포함하는 SDF(Standard Delay Format)파일과 VHDL netlist 정보를 출력한다. VHDL 시뮬레이터인 V-System 은 VHDL netlist 파일과 SDF 파일을 입력하고, Xilinx vital 라이브러리를 이용하여 배치 및 배선된 설계의 정확성을 검사한다.

4.4 (128,124) RS 부호의 복호기 FPGA 실현

(128 ,124) RS 복호기는 제어 신호 생성부,

표 4.2 (128, 124) RS 복호기의 입출력 신호

입 력 신 호			출 력 신 호		
신호명	비트수	핀번호	신호명	비트수	핀번호
수신 Byte Clock	1	B2	출력 프레임 클럭	1	G1
입력 데이터	8	V10~V17	출력 데이터	8	V2~V8
수신 프레임 클럭	1	B16			

표 4.3 (128,124) RS 부호의 복호기의 복잡도

	최 대 개 수	사 용 개 수
CLB수	1024	1024
I/O핀 수	192	21
F.G 함수	2048	1295
H 함수	1024	342
CLB 플립플럽	2048	1328

오증 생성부, 오증 요소 저장부, 오증 요소 변경부, 변경된 오증 요소 저장 및 순회부, 오류가 2인 경우의 오류위치 판별부, 오류가 2인 경우의 오류치 계산부, 오류가 1인 경우의 오류위치 판별부, 오류 개수 판별부, 132심벌 저장부, 오류치 선택 및 오류정정부로 구성된다. 이를 그림으로 표현하면 그림 4.2과 같다. (128, 124) RS 복호기는 하나의 XC4025로 실현 가능하였다. (128, 124) RS 부호의 복호기는 8개의 입력 데이타 신호, 8비트의 출력 데이타 신호, 기본 바이트 클럭, 그리고 수신 프레임의 경계를 나타내는 프레임 클럭과 복호기 출력 데이타의 프레임 경계를 나타내는 프레임 클럭으로 구성된다. (128, 124) RS 부호의 복호기 입출력 신호는 표 4.2와 같다.

(128, 124) RS 복호기는 하나의 XC4025로 실현될 수 있으며, 실제 배치 및 배선이 완료된 (128, 124) RS 복호의 실제 복잡도는 표 4.3과 같다.

5. 결 론

본 논문에서는 기존의 RS 부호의 부호화 및 복호 알고리즘을 분석한 후, 이를 바탕으로 AAL-1에 적용 가능한 오류정정능력이 2인 RS 부호에 대한 복호 알고리즘을 제시하고 복호기의 구조를 블럭도로 제시하였다. 또한 (128, 124) RS 부호의 부호기와 복호기를 하드웨어 합성 언어인 VHDL로 부호화하고, 부호기와 복호기를 VHDL로 시뮬레이션한 결과를 제시하였다. 시뮬레이션 결과 제시된 (128, 124) RS 부호의 부호화 및 복호 알고리즘의 타당성을 확인하였다. 또한 본 논문에서는 Examplar사의 Galileo 논리 합성기(Logic Explorer) 를 이용하여 VHDL 회로를 실제 회로로 합성한 후, 자이링스사의 XACT 배치 및 배선 툴을 이용하여 XC-4025 FPGA에 배치 및 라우팅하였으며, Galileo 타임 Explorer와 V 시스템을 이용하여 게이트-레벨 시뮬레이션을 수행하였다. 실현된 복호기는 1295개의 F 함수 및 G 함수, 342개의 H 함수 생성기,

1024개의 CLB, 1328개의 CLB 플립 프럽, 21개의 입출력핀으로 실현되였다. 또한 최대 임계 경로(Critical Path)가 175 nsec임을 고려하면 약 40Mbps 데이타 속도를 처리할 수 있음을 확인하였다. 따라서 따라서 복호기가 16Mbps 이상의 속도로 동작될 수 있음을 확인하였다. 오류정정 능력이 2인 RS 부호의 복호기는 XC-4025 FPGA 칩으로 실현될 수 있음을 확인하였다. 추후에는 셀 손실로 인한 삭제 복호 및 오류 정정이 가능한 (128, 124) RS 복호 알고리즘도 제안하여 관련 기능이 FPGA로 실현될 예정이다. 끝으로 본 연구 수행중 도와주신 김영백군, 박동규 교수님께 깊이 감사드립니다. 본 연구는 한국통신연구개발본부 HAN/BSDN 위탁연구사업의 연구결과의 일부입니다.

참 고 문 헌

[1] ITU Rec. I.363.1, B-ISDN ATM Adaptation Layer (AAL) Specification, Types 1 and 2, ITU-T SG13, July 1995.

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

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

[4] Lin, S. and Costello, D. J., Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englwood Cliffs, N.J, 1983.

[5] Forney, G.D., "On Decoding BCH Codes," IEEE Trans. on Inf. Theory, IT-11, pp.577-580, 1965.

[6] F. Polkinhorn, "Decoding of Double Error Correcting Bose-Chaudhuri Code," IEEE Trans. on Inform. Theory, IT-12, pp.480-481, Oct. 1966.

[7] 염 흥열, 박 동규외, "AAL-1 에서의 FEC 구현," 순천향대학교, 연구보고서, 연구개발원 한국통신연구 개발원, 1996.4.

[8] T.T.Tjhung, Choon Tiong Ng, and Choon Sum Ng, "On the Use of Reed-Solomon Codes in Improving Digital FM Transmission Performance in Rayleigh Fading," IEEE Tr. on Vehicular Technology, Vol.44, No.1, February 1995.

□ 著者紹介

염 흥 열(종신회원)

1981년 漢陽大學校 電子工學科 卒業(學士)
1983년 漢陽大學校 大學院 電子工學科 卒業(工學碩士)
1990년 漢陽大學校 大學院 電子工學科 卒業(工學博士)
1982년 12월 ~ 1990년 9월 韓國電子通信研究所 先任研究員
1990년 3월 ~ 현재 順天鄕大學校 工科大學 電子工學科 副教授
1997년 3월 ~ 현재 한국통신정보보호학회 총무이사

※ 관심분야 : 암호이론, 부호이론, 이동통신 분야