(204,188) Reed-Solomon 복호기 설계

정회원 김 진 규*, 강 성 태*, 유 영 갑*, 조 경 록*

Design of a (204,188) Reed-Solomon Decoder

Jinkyu Kim*, Sungtae Kang*, Younggap You*, Kyoungrok Cho* ***Regular Members***

요 약

본 논문에서는 회로크기와 계산시간에서 효율적인 Reed-Solomon(RS) 복호기의 새로운 구조를 제안한다. 제안한 구조는 다음과 같이 두 가지 특징을 가진다. 첫째, 두 개의 유클리드 셀을 순환구조로 하였으며, 이는 유클리드 블록을 완전 파이프라인으로 설계하는 경우에 비해 회로의 크기가 1/8정도로 감소되었다. 둘째, 2개의 순환구조 유클리드 블록은 기준주파수의 2배로 동작할 수 있어 연산시간이 감소되었다. 본 논문에서는 C언어와 Matlab을 이용하여 각각의 알고리즘을 검증하고, VHDL로 설계하여 FPGA로 동작을 검증한다.

ABSTRACT

In this paper, we propose a novel RS decoder design yielding smaller circuit size and shorter coding latency. The proposed architecture of RS decoder has the following two features. First, circuit size is reduced by using Euclid algorithm with mutual operation between cells. Second, coding latency is reduced by using higher frequency than syndrome and error value calculation block. We performed simulation with C language and MATLAB in order to verify the decoding algorithm and implemented using FPGA chips in VHDL.

I. 서 론

현재 디지털 통신 시스템에서는 연집오류에 강한 정정능력을 보이는 채널코딩방식으로 RS부호를 많이 이용하고 있다. RS부호는 우수한 정정 능력으로 Compact Disc Player(CD-P)와 Digital Audio Tape(DAT) Recorder, Digital VCR, ATM 등 여러 분야에서 널리 이용되고 있다[1][2]. RS부호의 부호기는 나눗셈기 하나를 이용하여 쉽게 회로화가 가능하지만 복호기는 상대적으로 복잡한 알고리즘과 연산회로를 요구한다. 따라서 지금까지 복호를 쉽게 하기 위한 알고리즘이 많이 연구되어왔다. 그 중 지금까지 잘 알려져 있는 방법이 세 가지가 있다. 첫째, 신드롬값이 오류위치와 오류값에 의해 결정되는 것에 착안하여 직접 행렬계산을 이용한 Peterson-Gorenstein-Zierler 복호 알고리즘이 있다[7]. 그러나 이 알고리즘의 경우에 계산해야 하는 신드롬값이 많아지는 경우 회로가 너무 커진다. 둘째, Berlekamp의 반복 알고리즘에 Massey의 Feedback Shift Register(FSR)를 이용한 Berlekamp-Massey 알고리즘이 있다[8]. 이 알고리즘 또한 회로가 복잡하고, 많은 계산량 때문에 복호시간이 길어진다. 셋째, 유클리드 알고리즘을 덧셈과 곱셈만으로 GCD 계산이 가능하게 한 수정된 유클리드 알고리즘이 있다[3]. 수정된 유클리드 알고리즘은 회로구현이 비교적 용이하다는 장점등으로 널리 사용되고 있다. RS복호기의 유클리드 알고리즘 계산 블록을 시스톨릭 어레이 방식으로 설계한 경우 계산시간은 $(n-k)\cdot(n-k+1)$개 이상의 클럭이 필요하다[3]. 부호간의 최소거리가 커질수록 계산되는 신드롬값이 많아지므로 회로의 크기는 최소거리의 제곱값에 비례하여 커지게 된다. 또한 계산시간이 그 만큼 길어지므로 수신된 코드를 저장하기 위한 FIFO 메모리

* 충북대학교 정보통신공학과 (jkkim@hbt.chungbuk.ac.kr)
논문번호:99461-1116, 접수일자 : 1999년 11월 16일

의 크기도 그만큼 늘어나게 된다. 셀을 이용한 MUX방식으로 설계하는 경우에는 셀의 앞단과 뒷단에 DMUX와 MUX가 추가되고, 셀이 많을수록 제어가 복잡해지므로 설계가 용이하지 않은 편이다[4].

본 논문에서는 수정된 유클리드 알고리즘에 새로운 구조를 제안하여 하드웨어의 크기가 커지는 단점을 해결하고자 한다. 이를 위하여 유클리드 셀을 두 개만으로 구성하여 회로의 크기를 줄이고 유클리드 블록을 2배의 주파수로 동작하도록 설계하여 계산시간을 줄이고자 한다. 본 논문의 Ⅱ장에서는 복호 알고리즘, Ⅲ장에서는 제안된 복호기의 회로구조와 회로의 구현, Ⅳ장에서는 실험결과 및 고찰 그리고 Ⅴ장으로 결론을 맺는다.

Ⅱ. RS부호의 복호 알고리즘

RS부호의 부호화 과정과 함께 신드롬 계산 과정을 설명하고자 한다. RS 부호는 (n, k)로 표현되며, 여기서 n은 부호어의 길이, k는 정보어의 길이이다. 정정 능력 t는 $(n-k)/2$로 표현되며, 부호간의 최소거리 d_m는 $n-k+1$의 값을 가진다. 먼저 부호화 과정을 설명하면 RS 부호다항식은 x^{16}를 곱한 정보다항식과 검사다항식의 합으로 이루어진다. 검사다항식은 x^{16}를 곱한 정보다항식을 생성다항식으로 나눈 나머지 다항식이다.

부호화 과정을 수식으로 자세히 살펴보면 다음과 같다. RS 부호기에서 코드값을 생성할 때 사용하는 생성다항식 $g(x)$는 식 (1)과 같다.

$$g(x) = (x+\alpha^1)(x+\alpha^2)\cdots(x+\alpha^{16}) = \sum_{i=0}^{16} g_i x^i \quad (1)$$

생성다항식은 α^1부터 α^{16}까지 16개의 근을 가지며 이 근들은 복호과정에서 신드롬을 계산하는데 이용된다. 입력되는 정보다항식을 식 (2)와 같이 표현할 때 생성되는 부호 다항식은 식 (3)과 같다.

$$d(x) = d_0 + d_1 x + \ldots + d_{203} x^{203} \quad (2)$$

$$c(x) = x^{16} d(x) + p(x) = \sum_{i=0}^{203} c_i x^i \quad (3)$$

식 (3)에서 검사다항식은 식(4)와 같다.

$$p(x) = x^{16} d(x) \bmod g(x) \quad (4)$$

신드롬 값은 수신된 부호 다항식에 생성다항식의 근을 대입하고 수신 받은 부호에 대하여 오류의 판별여부와 정정을 위한 단서를 제공하기 위해 계산한다. 오류가 없는 경우에는 신드롬값이 '0'이 오류인 경우에는 오류위치값과 오류값이 곱해진 형태로 계산된다.

신드롬 계산과정을 수식으로 살펴보면 다음과 같다. 식 (3)을 이용하여 얻은 부호 다항식은 채널을 통한 전송과정에서 오류가 첨가된다. 원래의 부호어와 오류가 더해져서 수신된 부호어의 관계는 식 (5)와 같다.

$$r(x) = \sum_{i=0}^{203} r_i x^i = \sum_{i=0}^{203} c_i x^i + \sum_{i=0}^{203} e_i x^i \quad (5)$$

신드롬 계산은 식 (6)과 같다. α^1부터 α^{16}까지 각각 16개의 근을 대입한 신드롬의 값은 총 16개가 계산된다.

$$S_j = \sum_{i=0}^{203} r_i (\alpha^j)^i \qquad (1 \le j \le 16) \quad (6)$$

오류위치 다항식과 오류평가 다항식을 계산하는 과정을 설명하고자 한다. 오류위치 다항식은 오류위치값의 역수를 근으로 갖는다. 오류평가 다항식은 오류값을 계산하기 위해 필요한 다항식이다.

오류위치 및 오류평가 다항식을 구하는 수정된 유클리드 알고리즘에 대한 수식 유도과정을 설명하고자 한다. 먼저 오류위치 다항식과 신드롬 다항식을 정의하고 두 다항식을 서로 곱하여 유클리드 키 방정식을 도출한다. 그 후에 유클리드 키 방정식을 항상 만족하는 수정된 유클리드 알고리즘에 대해 설명하고자 한다.

유클리드 키 방정식을 도출하기 위해 오류위치의 역수를 근으로 갖는 오류위치 다항식 $\lambda(x)$를 식 (7)과 같이 정의한다. $2t$ 개의 신드롬 값을 계수로 갖는 신드롬 다항식 $S(x)$를 식 (8)과 같이 정의한다.

$$\lambda(x) = \prod_{i=1}^{v} (1 + X_i x) = \sum_{i=0}^{v} \sigma_i x^i \quad (7)$$

$$S(x) = \sum_{i=1}^{2t} \sum_{j=1}^{v} Y_j X_j^i x^{i-1} \quad (8)$$

식 (7)과 식 (8)를 곱하여 식 (9)와 같이 전개한다.

$$S(x)\lambda(x) = \sum_{i=1}^{2t} \sum_{j=1}^{v} Y_j X_j^i x^{i-1} \cdot \prod_{k=1}^{v} (1 + X_k x) = \sum_{j=1}^{v} X_j Y_j \prod_{k=1, k\neq j}^{v} (1 + X_k x)(1 + X_j^{2t} x^{2t}) \quad (9)$$

식 (9)에서 x^{2t}를 곱한 부분과 곱하지 않은 부분으로 나누어 식 (10), 식 (11)로 정의하자.

$$R(x)=\sum_{j=1}^{v} X_j Y_j \prod_{k=1,k\neq j}^{v}(1+X_k x) \tag{10}$$

$$\gamma(x)=\sum_{j=1}^{v} X_j Y_j \prod_{k=1,k\neq j}^{v}(1+X_k x)(X_j^{2t}) \tag{11}$$

새롭게 정의한 $R(x)$, $\gamma(x)$를 이용하여 식 (9)를 다시 전개하면 식 (12)과 같다.

$$S(x)\lambda(x)=R(x)+x^{2t}\gamma(x) \tag{12}$$

식 (12)을 유클리드 키 방정식이라고 한다[3].

수정된 유클리드 알고리즘은 반복할 때마다 항상 유클리드 키 방정식을 만족한다. 알고리즘은 식 (13)와 같은 초기값을 갖고, 식 (14)과 식 (15)를 최대 $2t$ 번까지 수행하게 된다.

$$R_0(x)=x^{2t} \qquad Q_0(x)=\sum_{i=1}^{2t} S_i x^{i-1}$$

$$\lambda_0(x)=0 \qquad \mu_0(x)=1 \tag{13}$$

$$\begin{aligned} R_i(x)&=[\sigma_{i-1}b_{i-1}R_{i-1}(x)+\bar{\sigma}_{i-1}a_{i-1}Q_{i-1}(x)]\\ &\quad -x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}Q_{i-1}(x)+\bar{\sigma}_{i-1}b_{i-1}R_{i-1}(x)]\\ \lambda_i(x)&=[\sigma_{i-1}b_{i-1}\lambda_{i-1}(x)+\bar{\sigma}_{i-1}a_{i-1}\mu_{i-1}(x)]\\ &\quad -x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}\mu_{i-1}(x)+\bar{\sigma}_{i-1}b_{i-1}\lambda_{i-1}(x)]\\ \gamma_i(x)&=[\sigma_{i-1}b_{i-1}\gamma_{i-1}(x)+\bar{\sigma}_{i-1}a_{i-1}\eta_{i-1}(x)]\\ &\quad -x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}\eta_{i-1}(x)+\bar{\sigma}_{i-1}b_{i-1}\gamma_{i-1}(x)] \end{aligned} \tag{14}$$

$$\begin{aligned} Q_i(x)&=\sigma_{i-1}Q_{i-1}(x)+\bar{\sigma}_{i-1}R_{i-1}(x)\\ \mu_i(x)&=\sigma_{i-1}\mu_{i-1}(x)+\bar{\sigma}_{i-1}\lambda_{i-1}(x)\\ \eta_i(x)&=\sigma_{i-1}\eta_{i-1}(x)+\bar{\sigma}_{i-1}\gamma_{i-1}(x) \end{aligned} \tag{15}$$

여기에서 a_{i-1}와 b_{i-1}는 각각 $R_{i-1}(x)$와 $Q_{i-1}(x)$의 최고차항의 계수이다.

$$l_{i-1}=\deg(R_{i-1}(x))-\deg(Q_{i-1}(x))$$

$$\begin{aligned} &\text{if } l_{i-1}\geq 0 \quad \textit{then} \quad \sigma_{i-1}=1\\ &\textit{else} \quad \sigma_{i-1}=0 \end{aligned}$$

알고리즘은 $\lambda(x)$가 $R(x)$보다 차수가 크면 멈춘다. 멈춘 후에는 $\lambda(x)$가 오류위치 다항식 $\sigma(x)$가 되며, $R(x)$는 오류평가 다항식 $\omega(x)$가 된다.

오류위치 다항식 $\sigma(x)$와 오류평가 다항식 $\omega(x)$의 계수값을 이용하여 오류위치와 오류값을 계산하는 과정을 설명하고자 한다. 오류위치는 오류위치 다항식에 근을 대입하여 구할 수 있고 오류값은 오류평가 다항식을 미분한 오류위치 다항식으로 나누어 구할 수 있다.

유클리드 알고리즘에서 구한 오류위치 다항식 $\sigma(x)$는 식 (16)과 같다. 오류평가 다항식 $\omega(x)$는 식 (17)과 같다.

$$\sigma(x)=\prod_{i=1}^{v}(1+X_i x) \tag{16}$$

$$\omega(x)=\sum_{j=1}^{v} X_j Y_j \prod_{i=1,i\neq j}^{v}(1+X_i x) \tag{17}$$

여기서 X_j, Y_j 는 각각 오류위치값과 오류값이다.

식 (16)을 미분해 보면, 식 (18)이 된다.

$$\sigma'(x)=\sum_{j=1}^{v} X_j \prod_{i=1,i\neq j}^{v}(1+X_i x) \tag{18}$$

식 (17)을 분자식에 식 (18)을 분모식으로 놓고 x 대신에 X_j^{-1}를 대입하면 식 (19)와 같이 된다.

$$Y_j=\frac{\sum_{j=1}^{v} X_j Y_j \prod_{i=1,i\neq j}^{v}(1+X_i X_j^{-1})}{\sum_{j=1}^{v} X_j \prod_{i=1,i\neq j}^{v}(1+X_i X_j^{-1})}=\frac{\omega(X_j^{-1})}{\sigma'(X_j^{-1})} \tag{19}$$

따라서 오류위치값은 식 (16)에서 x대신에 α^{52}부터 α^{255}를 대입하여 결과가 '0'이 나오는 경우에 그 때의 근의 역수가 된다. 오류값은 식 (19)에 오류위치값의 역수를 대입함으로써 구할 수 있다.

Ⅲ. 제안한 RS 복호기의 구조

본 논문에서는 $GF(256)$상에서 8개까지 오류를 정정할 수 있는 (204, 188) RS 복호기를 설계하였다. 한 심볼은 8bits로 구성된다. 부호간의 최소 거리는 17이다. 그림 1은 설계된 RS 복호기의 전체구조이다. 복호과정은 신드롬(Syndrome) 계산, 오류위치 다항식(Error locator polynomial)과 오류평가 다항식(Error evaluator polynomial) 계산, 오류 값과 오류 위치의 계산 그리고 오류정정으로 구성된다. 데이터 처리는 3단의 파이프라인형태로 구성하였다. 복호기로 수신되는 데이터는 FIFO에서 408 심볼 클럭동안 지연된 후에 정정되어진다.

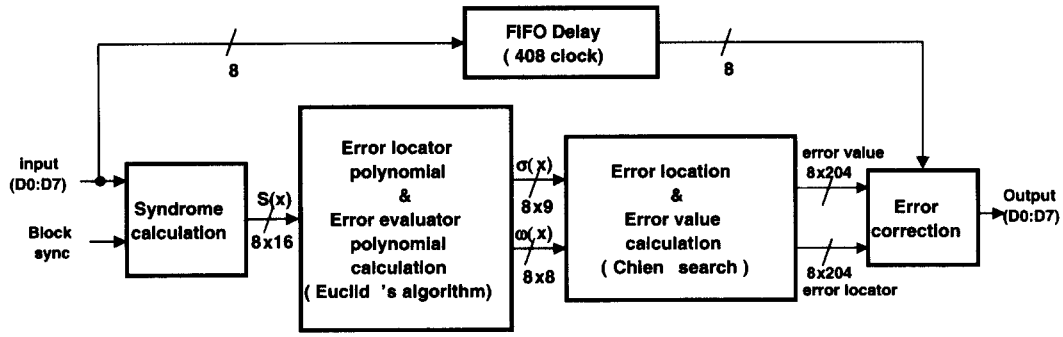


그림 1. RS 복호기의 전체 구조

3.1 신드롬 계산

신드롬에 대한 회로 구현은 그림 2와 같다. 204개의 수신 심볼은 심볼 클럭에 동기되어 r_{203}부터 r_0까지 높은 차수부터 입력된다. 입력된 심볼은 R_0부터 R_{15}까지의 레지스터에 저장된다. 저장된 후에는 심볼 클럭마다 α^1부터 α^{16}까지의 상수곱셈기로 곱해진다. 곱해진 결과값은 다음에 입력되는 심볼과 더해져서 다시 저장된다. 이러한 반복과정은 r_{203}부터 r_0까지 입력될 동안 계속된다.

반복계산이 끝난 후에 R_0부터 R_{15}까지 저장되어 있는 값들이 16개의 신드롬 값들이 된다. 이 신드롬 값들은 2배의 심볼 클럭 주파수에 동기되어 레지스터 B_0부터 B_{15}까지 전송된다. 레지스터 B_0부터 B_{15}까지 저장된 신드롬값은 다항식 형태로 유클리드 알고리즘 계산 블록으로 전송된다.

3.2 오류위치 및 오류평가 다항식

본 논문에서 제안한 유클리드 알고리즘 블록은 그림 3과 같다. 두 개의 셀이 묶여서 서로 제어하는 루프구조로 되어있고 제어신호는 $R(x)$, $Q(x)$ 다항식에 의해서만 발생한다. 출력된 제어신호들은 $R(x)$, $Q(x)$, $\lambda(x)$, $\mu(x)$ 다항식을 계산하는 셀을 제어하게 된다. 앞단의 MUX는 sync 신호에 의해

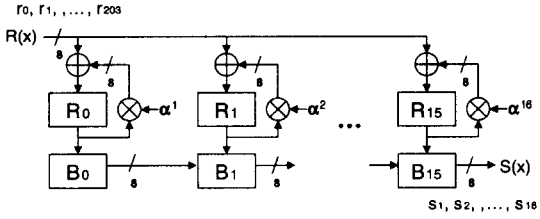


그림 2. 신드롬 계산 구조

두 가지 동작을 수행한다. sync 신호가 '1' 인 경우에는 신드롬 다항식 $S(x)$를 받아들인다. sync 신호가 '0'인 경우에는 두 개의 유클리드 셀이 루프를 형성하여 반복계산을 한다.

유클리드 기본셀의 구조를 그림 4에 나타냈다. σ_i 값에 따라 두 가지 동작을 수행하며, 먼저 σ_i 의 값이 '1'인 경우에는 MUX1과 MUX2를 제어하여 y_i의 값을 register array A로 보낸다. x_i는 register array B를 통과한 후에 b_i값과 곱해진다. y_i는 $R(x)$와 $Q(x)$의 차수 차인 l_i로 제어되어 register array C를 통과한 후에 a_i와 곱해진다. 따라서 x_{i+1}은 $b_i x_i + a_i y_i x^{l_i}$로 연산된다.

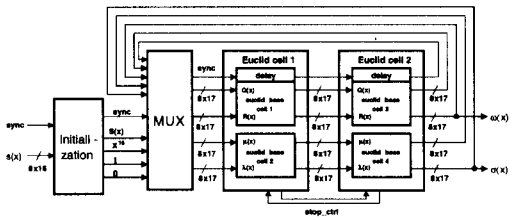


그림 3. 유클리드 알고리즘 계산 블록

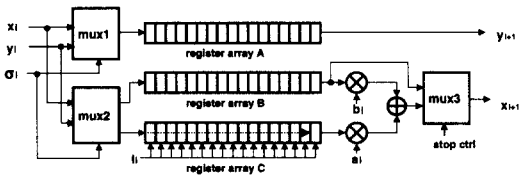


그림 4. 유클리드 기본셀 구조

σ_i 의 값이 '0'인 경우에는 MUX1과 MUX2를 제어하여 x_i의 값을 register array A로 보내준다. y_i는 register array B를 통과한 후에 a_i값과 곱해

진다. x_i는 $R(x)$와 $Q(x)$의 차수 차인 l_i로 제어되어 register array C를 통과한 후에 b_i와 곱해진다. 따라서 x_{i+1}은 $a_i y_i + b_i x_i x^{l_i}$로 연산된다. 유클리드 알고리즘의 중단조건이 만족되면 stop ctrl 신호와 σ_i신호를 제어하여 x_i, y_i가 그대로 출력된다.

3.3 오류위치 및 오류값

그림 5에 오류값과 오류위치를 계산하는 회로를 보였다. 전체구조는 Chien search 회로[6], inverse table ROM, 심볼곱셈기 그리고 정정회로로 나뉘어진다. 먼저 오류위치 다항식 $\sigma(x)$의 계수를 입력받은 Chien search 회로는 $\sigma(\alpha^{52})$, $\sigma(\alpha^{53})$, $\cdots$, $\sigma(\alpha^{255})$의 순서로 계산을 한다. 오류값의 역수가 근인 경우에는 '0'의 출력을 정정블록으로 보낸다. 오류평가 다항식 $\omega(x)$의 계수를 입력받는 Chien search 회로도 $\omega(\alpha^{52})$, $\omega(\alpha^{53})$, $\cdots$, $\omega(\alpha^{255})$의 순서로 계산된다.

그림 6에 Chien search 회로를 보였다. 오류위치와 오류평가 다항식의 계수를 입력받아 x대신에 α^{52}, α^{53}, $\cdots$, α^{255} 를 대입하여 다항식의 값을 계산한다. 두 배의 심볼클럭 주파수에 동기하여 $\sigma(x)$와 $\omega(x)$의 계수는 높은 차수부터 B0부터 B8까지의 레지스터에 입력된다. Chien search 회로는 sync에 따라 두가지 동작을 수행한다. sync신호가 '1' 인 경우에는 B_0부터 B_8까지 저장되어 있는 값을 동시에 R_0부터 R_8까지의 레지스터로 전송한다. sync 신호가 '0' 인 경우에는 R_0부터 R_8까지 저장되어 있는 값이 상수곱셈기를 통해 반복연산을 수행한다. 유클리드 블록에서 입력되는 각 다항식의 계수는 R_0부터 R_8까지 입력되기전에 상수곱셈기 (α^0, α^{51}, α^{102}, α^{153}, α^{204}, α^0, α^{51}, α^{102}, α^{153})로 곱해진다. 이렇게 곱해줌으로써 Chien search 회로는 α^{51}부터 대입한 효과를 가진다.

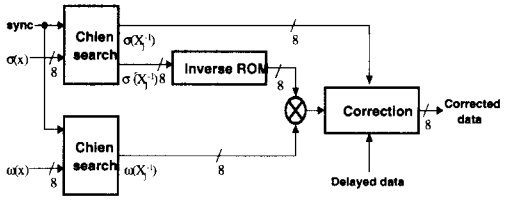


그림 5. 오류값과 오류위치를 가지고 정정하는 구조

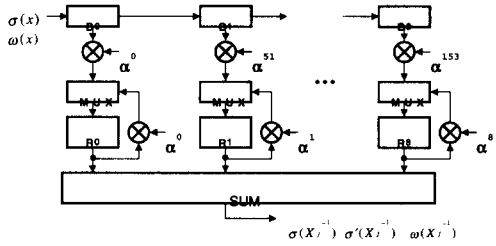


그림 6. Chien search 구조

Inverse table ROM은 α^i 값이 입력되면 α^{255-i} 값을 출력으로 내보낸다. 주소값은 8 bits이고, 데이터도 8 bits이다.

심볼곱셈기는 $GF(256)$상에서 두 입력에 대하여 곱셈을 연산한다. 조합회로로 구성하였고 입·출력되는 심볼의 버스폭을 8 bits로 함으로써 병렬형 방식으로 설계하였다.

오류 정정은 검출된 오류위치에서 오류값과 메모리에 저장된 부호어를 더함으로써 이루어진다.

Ⅳ. 실험 결과 및 고찰

회로 구현과정은 네 가지의 단계를 거치게 되며, VHDL을 이용한 회로합성, 시뮬레이션후 FPGA를 제작하고 실험하였다. $GF(256)$에 대한 역수값을 얻기 위해서 256 byte 크기의 table ROM을 사용하였다. 수신 데이터를 408클럭동안 지연시키기 위하여, 408 byte RAM을 파이프라인 구조로 구성하여 사용하였다.

실험 방법은 다음과 같다. 두 대의 PC를 이용하여 한 쪽 PC에서 영상데이터를 부호화하여 BPSK로 변조한 다음 가우시안 채널에 적용하여 FPGA로 보내고 다른 PC에서는 FPGA에서 정정된 영상데이터를 수신하여 화면에 출력되게 하였다. 그림 7에 $E_b/N_0 = 6.5dB$ 하에서 가우시안 채널에 의해 오류가 더해진 영상과 복원된 영상을 보였다.

(a) (b)

그림 7. $E_b/N_0 = 6.5dB$ 에서 테스트 결과:
(a) 오류가 발생된 영상; (b) 오류를 정정한 영상

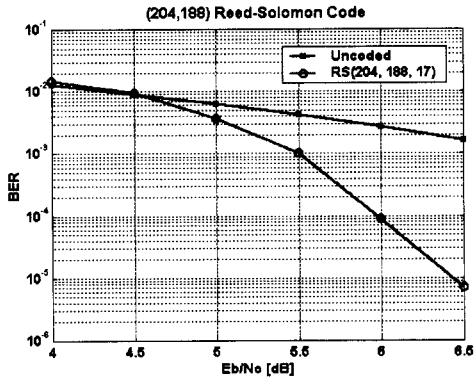


그림 8. 가우시안 채널하에서 비트오류률 (BPSK로 변조한 경우)

그림 8에는 각각의 E_b/N_0하에서 부호화하지 않은 신호와 (204, 188) RS 부호로 부호화하여 수신된 신호에 대한 BER(bit error rate)를 보였다. 그림 8에 나타난 것처럼 $E_b/N_0=6.0dB$ 이하에서는 BER 값이 10^{-4}이하로 작아짐을 알 수 있다.

표 1은 유클리드 알고리즘을 계산하는 데 필요한 셀의 개수를 기존 논문과 비교한 것이다. 유클리드 셀의 크기와 계산시간은 $(n-k)$값에 비례하여 증가한다. $GF(256)$상에서 $(n-k)$값이 동일한 RS부호인 경우에 유클리드셀의 계산량이 동일함으로 한 셀의 회로 크기는 일정하다. 따라서 회로의 크기를 유클리드 셀에 기초하여 비교하였다. 본 논문에서 제안한 방법을 이용하면, 두 개의 유클리드 셀만으로 계산이 가능하다. 표 1에서 보여진 바와 같이 시스톨릭 어레이 경우에 셀을 파이프라인 형태로 배열하므로 전체 소요되는 유클리드 셀의 개수는 $(n-k)$값을 갖는다[3]. 셀을 이용한 MUX 방식인 경우 유클리드 셀의 앞단과 뒷단에 DMUX와 MUX를 추가시킨 형태로 셀의 개수는 $(n-k)^2/n$ 보다 큰 가장 작은 정수가 된다[4]. 본 논문에서 제안한

표 1. 유클리드 알고리즘을 계산하기 위해 사용되는 셀 개수의 비교

RS Code	Full systolic array[3]	Multi-plexing on Recursive Cells[4]	Proposed
(15, 9)	6	3	2
(31, 15)	16	9	2
(204, 188)	16	2	2
(255, 223)	32	5	2

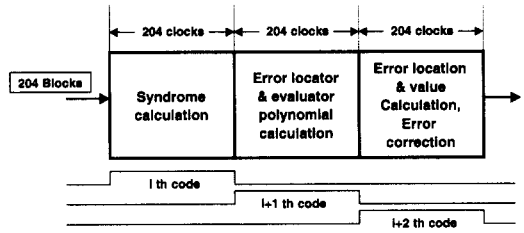


그림 9. 제안된 RS 복호기의 동작도

방식인 경우 코드규격에 유연하게 단지 2개의 유클리드 셀만으로도 계산이 가능하다. 따라서 RS(255, 223)인 경우 제안된 유클리드 알고리즘 블록을 적용하면 시스톨릭 어레이 방식에 비하여 1/16, 셀을 이용한 MUX 방식에 비하여 2/5의 크기를 갖는다. (204, 188)을 시스톨릭 어레이 방식으로 설계할 경우 $(n-k)=16$ 이므로 16개의 유클리드 셀이 필요하다. 따라서 본 논문에서 제안한 방법으로 설계할 경우 파이프라인으로 설계한 경우와 같은 데이터 처리율을 가지면서 유클리드 블록은 1/8의 크기로 감소한다. 또한 유클리드 블록은 다른 블록과는 독립적으로 계산된다. 따라서 유클리드 알고리즘 블록의 클럭주파수를 높임으로써 $(n-k)$ 값이 커지더라도 한 부호어 안에 계산할 수 있다.

그림 9에는 제안한 RS 복호기의 동작도를 보였다. 전체 동작은 3단의 파이프라인으로 구성된다. 먼저 i번째 수신받은 부호어는 204 심볼클럭동안 신드롬을 계산한다. 다음 i+1 번째 204 심볼클럭동안 오류위치 다항식과 오류평가 다항식의 계수를 구하게 된다. 마지막으로 i+2 번째 204 심볼클럭동안은 오류위치와 오류값을 계산하면서 수신된 코드의 오류를 정정하게 된다. 따라서 소요되는 FIFO 지연값은 408 심볼 클럭이다.

V. 결 론

본 논문에서는 8개의 심볼오류까지 정정이 가능한 (204, 188) RS복호기를 두 개의 유클리드 셀만을 이용하여 설계하였다. VHDL 코딩기법을 이용하여 제안한 구조를 회로로 합성하고 FPGA로 구현하여 동작을 검증하였다. 본 논문에서 제안한 구조는 다음과 같은 이점을 갖는다. 첫째, 두 개의 유클리드 셀을 서로 제어하도록 설계하여 반복적으로 연산이 수행되게 하였다. 따라서 유클리드 알고리즘 계산 블럭을 파이프라인으로 설계할 때보다 회로가 1/8 크기로 감소하는 효과를 가진다. 둘째, 신드롬

계산, 오류위치와 오류값 계산 블록보다 2배 이상의 클럭주파수로 동작이 가능하도록 설계하였다. 따라서 유클리드 알고리즘 계산시간을 $2t\cdot(2t+1)$ 클럭 시간에서 $t\cdot(2t+1)$ 클럭시간으로 줄일 수 있었다. 본 논문에서 설계한 RS복호기 구조는 (204, 188) 뿐만 아니라 다른 부호크기를 갖는 RS부호에 대한 복호기 설계에도 용이하게 적용될 것이다. 또한 통신시스템 one-chip화 추세에 맞춰 라이브러리로 활용될 수 있을 것이다.

참 고 문 헌

[1] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, prentice-Hall, 1995.

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

[3] H. M. Shao, et al., "A VLSI design of a pipeline Reed-Solomon decoder," *IEEE Trans. comput.*, vol. C-34, no. 5, pp. 393-402, May 1985.

[4] H. M. Shao and I. S. Reed, "On the VLSI design of a pipeline Reed-Solomon decoder using systolic arrays," *IEEE Trans. Comput.*, vol. C-37, no. 10, pp. 1273-1279, Oct. 1988

[5] T. Iwaki, et al., "Architecture for a high speed Reed-Solomon decoder," *IEEE Trans. Comsumer Electronics*, vol. 40, no. 1, pp. 75-82, Feb. 1994.

[6] R. T. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghm codes," *IEEE Trans. Inform. Theory*, vol. IT-10, pp.357-363, April 1964.

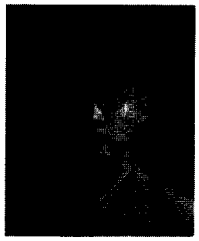
[7] D. Gorenstein and N. Zierler, "A Class of Error Correction Codes in p^m Symbols," *Journal of the Society of Industrial and Applied Mathematics*, vol. 9, pp. 207-214, June 1961.

[8] J. L. Massey, "Shift Register Synthesis and BCH Decoding," *IEEE Transactions on Information Thoery,* vol. IT-15, No.1, 122-127, January 1969.

[9] G. D. Forney, "On Decoding BCH Codes," *IEEE Transactions on Information Theory*, vol. IT-11, pp.549-557, October 1965.

[10] S. Kwon, and H. Shin, "An area efficient VLSI architecture of a Reed-Solomon decoder/encoder for digital VCR's," *IEEE Trans. Consumer Electron.*, pp. 1019-1027, Nov. 1997.

김 진 규(Jinkyu Kim) 준회원

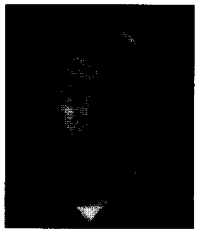


1998년 : 충북대학교 정보통신공학과 학사.

1998년 3월~현재 : 충북대학교 정보통신 공학과 석사과정.

<주관심 분야> 디지털회로설계, 채널코딩 및 통신시스템.

강 성 태(Sungtae Kang) 비회원



1998년 : 충북대학교 정보통신공학과 학사.

1998년 3월~현재 : 충북대학교 정보통신 공학과 석사과정.

<주관심 분야> 집적회로 설계, 채널 코딩.

유 영 갑(Young-gap You) 정회원

1975년 : 서강대학교 전자공학과 공학사.

1981년 : 미시간대학교(미국) 전기전산공학과 공학석사.

1986년 : 미시간대학교(미국) 전기전산공학과 공학박사.

1988년 3월~현재 : 충북대학교 전기 전자 공학부 교수.

<주관심 분야> Computer architecture, Memory testing, 고속 시스템 설계, HDTV, ATM등

조 경 록(Kyoung-rok Cho) 정회원

1977년 : 경북대학교 전자공학과 공학사.
1989년 : 일본 동경대학교 전자공학과 공학석사.
1992년 : 일본 동경대학교 전자공학과 공학박사.
1979년~1986년 : (주)금성사 TV연구소 선임연구원.
1999년~2000년 : 오레곤주립대학 객원교수.
1992년~현재 : 충북대학교 정보통신공학과 부교수.
<주관심 분야> VLSI 시스템설계, 통신 시스템용 LSI 개발, 고속 마이크로프로세서 설계.