論 文

코드 레이트가 높은 시스템에 있어서의 비이진코드의 디코더 설계

正會員 鄭 一 石* 正會員 康 昌 彦**

Decoder Design of a Nonbinary Code in the System with a High Code Rate

Il Suk JEONG,* *and* Chang Eon KANG**, *Regular Members*

요 약 본 논문은 코드 레이트가 R, 에러 정정 능력이 t일때, R>1/t를 만족하는 비이진 코드의 디코더 설계에 관한 연구이다. 에러 트래핑 디코딩 방식으로 설계하기위해 카버링 단항식 개념을 도입하였으며, 실제 이를 이용하여 (15, 11) Reed-Solomon 코드의 디코더를 구현하였다. 이 디코더 시스템은 Galois Field 곱셈 및 나눗셈 회로를 필요로 하지 않으므로 간단히 구성할 수 있었으며, 마이크로 컴퓨터를 이용하여 실험하였다. 본 연구의 결과로서, 이 디코더는 하나의 코드 워드를 디코딩하는데 60클럭이 소요되었으며 2개의 심볼 에러와 8개의 이진 버스트 에러를 정정할 수 있으며, 성능은 채널 에러 확률이 5×10^{-4}~5×10^{-5}정도일 때 가장 효율적임을 알 수 있었다.

ABSTRACT In this paper the decoder of nonbinary code satisfying R>1/t has been designed and constructed, where R is the code rate and t is the error correcting capability. In order to design the error trapping decoder, the concept of covering monomial is used and them the decoder system using the (15, 11)Reed-Solomon code is implemented. Without Galois Field multiplication and division circuits, the decoder system is simply constructed. In the decoding process, it takes 60 clocks to decode one code word. Two symbol errors and eight binary burst errors are simultaneously corrected. This coding system is shown to be efficient when the channel error probability is approximately from 5×10^{-4} to 5×10^{-5}.

1. 서 론

순환 코드의 디코딩 방법중의 하나인 에러 트래핑(Error-Trapping)방법은 짧은 코드에(코드의 길이 : ~32심볼) 대해서 매우 효율적인 방법이다[1,2,3] 특히 길이가 짧은 Reed-Solomon 코드의 경우, 에러 트래핑 방법이 Berlekamp 디코더 보다 적용시키기가 더욱 쉽다[4].

에러 트래핑 방법은 Meggitt 디코더의 변형으로 Prange에 의해 처음으로 제안되었고[2], Rudolph와 Mitchell에 의해 1964년에 발표되었다[5]. 그 뒤 1970년에는 Lin[1], 그리고 1971년에 Yau와 Liu에 의해 체계화 되었다[6].

에러 트래핑 디코더에서 수신 코드 워드는 모든 에러가 $(n-k)$ 패리티 검사 부분에 한정될 때 에러 패턴은 신드롬과 같은 형태를 갖게 되고, 에

*, ** 延世大學校工科大學電子工學科
Dept. of Electronic Engineering, Yonsei University, Seoul, 120 Korea.
論文番號 : 86-06 (接受 1986. 1. 29)

러 정정 과정은 단순히 코드 워드의 패리티 검사 부분에서 신드롬의 값을 빼주면 된다.

그러나 코드 레이트(code rate) $R<1/t$를 만족하지 않는 코드까지 확장하기 위해 에러 트래핑 디코더를 변형해야 하며, 1964년에 Macwilliams, 1971년에 Yau와 Liu, 1981년에 Chan 과 Games에 의해 여러가지 방안이 제시되었다[6][7][8]. 특히 Kasami는 카버링(Covering) 다항식을 이용하여 이진(binary) 코드에 대한 변형된 에러 트래핑 디코더를 제시했다[9].

이런 연관성으로 최근에는 비이진(nonbinary) 순환 코드를 설계하기 위해 카버링 다항식 대신, 축소된 개념인 카버링 단항식이 도입되었다[10]. 이것은 에러의 대부분이 $(n-k)$ 패리티 검사 부분에 있고, 몇개의 에러는 정보 구간내에 있는 에러를 정정할 수 있다.

본 논문에서는 카버링 단항식 개념을 도입하여 2개의 심볼 에러를 정정할 수 있는 (15, 11) Reed-Solomon(RS) 코드의 디코더를 에러 트래핑 방법으로 설계했다.

2. 이론 해석

(1) 에러 트래핑 방법

코드 레이트 $R<1/t$를 만족하는 t에러 정정 순환 코드를 생각할 때, $V(X)$를 전송 코드 워드라 하고 $r(X)$를 수신코드 워드라고 한다.

이때 통신 채널에 의해 야기된 에러의 형태는 $e(X)=r(X)+v(X)$로 나타난다. 이 경우 수신된 코드 워드 $r(X)$의 신드롬(syndrome) $s(X)$는 $e(X)$를 생성 다항식 $g(X)$로 나눈 나머지이다. 즉 식(1)과 같이 나타난다.

$$e(X)=q(X)\,g(X)+s(X) \qquad (1)$$

에러 $e(X)$가 $r(X)$의 $n-k$패리티 검사 부분에 한정될 때, $e(X)$의 차수는 $n-k$보다 작아져 식(1)에서 $q(X)=0$이 되고 $e(X)=s(X)$가 된다. 즉 $r(X)$의 에러와 같은 형태가 되어 단순히 $r(X)$의 패리티 검사 부분에서 신드롬의 값을 빼주면 된다.

만약 에러가 형태가 $r(X)$의 $n-k$패리티 검사 부분에 한정 되지 않을 경우에 $r(X)$를 순환 이동시켜 에러의 형태가 $n-k$패리티 검사 부분에 한정되게 한 후, 같은 방법으로 에러 정정이 수행된다.

그림 (1)에는 위의 방법을 토대로 해서 $n-k$신드롬 레지스터를 사용하여 에러 트래핑 디코더를 구성한 것이다.

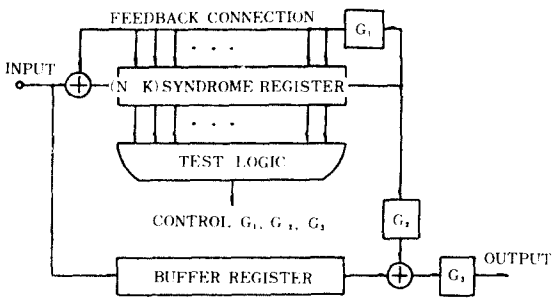

그림 1 일반적인 에러 트래핑 디코더
General error-trapping decoder.

(2) Reed-Solomon 코드

Reed-Solomon(RS) 코드는 순환 심볼 에러 정정 코드로서, 채널을 통하여 코드 워드의 전송시 발생하는 이진 버스트(binary burst) 에러를 정정할 수 있다.

RS 코드는 Galois Field 심볼의 블락 구조로 되어 있으며 각 심볼은 Galois Field $GF(2^m)$의 소자(element)이다.

RS 코드의 매개 변수는 다음과 같다[13].

- 심볼당 비트(bit)수; m
- 코드 워드의 길이; $n=2^m-1$
- 정정할 수 있는 최대 에러수; t
- 최소 거리(minimum distance); $d=2t+1$
- 패리티 검사 심볼의 수; $2t$
- 정보 심볼의 수; $k=n-2t$

(15, 11)-2-에러 정정 RS 코드에서는 $m=4$, $n=15$, $t=2$, $d=5$, $2t=4$, $k=11$이 된다.

RS 코드의 생성 다항식 $g(X)$는 식(2)로 주어진다.

$$g(X)=\prod_{j=l}^{2t-l+1}(X-\alpha^j)=\sum_{i=0}^{2t} g_i X^i \qquad (2)$$

본 논문에서는 회로를 간단히 하기 위해 $l=2$로 선택했다.

(3)카버링 단항식을 이용한 에러 정정 방법[9],[10]

에러 다항식 $e(X)$를 정보 부분과 패리티 검사 부분으로 나누어 나타내면 식(3)과 같다.

$$e(X)=e_I(X)+e_P(X) \qquad (3)$$

여기서 $e_I(X)=e_{n-k}X^{n-k}+e_{n-k+1}X^{n-k+1}+\cdots+e_{n-1}X^{n-1}$, $e_P(X)=e_0+e_1X+\cdots+e_{n-k-1}X^{n-k-1}$이다.

만약 $e_I(X)=\varepsilon X^{n-k+a}$ ($\varepsilon\neq 0$, $0\leq a<k$)의 단항식 형태로 두면 $e_I(X)$는 $n-k$차 이상이므로 $g(X)$로 나누면 식(4)와 같다.

$$e_I(X)=q(X)\ g(X)+\varepsilon\rho_a(X) \qquad (4)$$

여기서 $\rho_a(X)=\rho_{a0}+\rho_{a1}X+\cdots+\rho_{a\,n-k-1}X^{n-k-1}$이다. 따라서 신드롬 $S(X)$는 식(5)와 같이 된다.

$$S(X)=\varepsilon\rho_a(X)+e_P(X) \qquad (5)$$

식(4), (5)로부터 정보 구간내의 에러 패턴 $e_I(X)$를 알면 검사 구간내의 에러 패턴 $e_P(X)$를 구할 수 있다. 따라서 에러 다항식 $e(X)$는 다음과 같이 된다.

$$e(X)=S(X)+\varepsilon\rho_a(X)+\varepsilon X^{n-k+a} \qquad (6)$$

여기서 $S(X)+\varepsilon\rho_a(X)$는 단항식 εX^{n-k+a}와 $e_I(X)$가 일치할 때의 패리티 검사 구간 내에 존재하는 에러 형태이다.

임의의 i ($0\leq i\leq n-k-1$)에 대해 $U_{a,i}=\rho_{a,i}^{-1}S_i=\varepsilon+\rho_{a,i}^{-1}e_i$라 둘 때, $U_a(X)$를 다음과 같이 정의한다.

$$U_a(X)=U_{a0}+U_{a1}X+\cdots+U_{a\,n-k-1}X^{n-k-1} \qquad (7)$$

만약 $t-1$이하의 에러가 패리티 검사 부분에 있다면 $e_P(X)$는 $t-1$이하의 영이 아닌 상수를 가지고 $U_a(X)$는 ε와 동일한 $n-k-t+1$개 이상의 상수를 가진다. 식(4)에 의해 $e(X)$의 정보 부분 혹은 $e(X)$의 순환 이동의 정보 부분이 단지 X^{n-k+a}에 영이 아닌 상수를 가진다면 단항식 X^{n-k+a} ($0\leq a<k$)는 에러 $e(X)$를 카버링(covering)한다.

설계코자 하는 디코더는 $S=\{0\}\cup\{X^{n-k+a},\ 0\leq a<k\}$인 카버링 단항식 세트(set)가 필요하며 이것은 t이하의 영이 아닌 상수를 가진 모든 에러 $e(X)$를 카버링한다.

t에러 정정 (n, k)순환 코드에 카버링 세트는 코드 레이트 $R<2/t$일 때만 존재하며, $t=2$에 대해 카버링 세트의 최소 크기는 $b=\lceil n/2(n-k)\rceil$이며, 카버링 세트는 $\{0, X^{2(n-k)-1}, X^{3(n-k)-1}, \cdots, X^{b(n-k)-1}\}$이다. 여기서 $\lceil z\rceil$는 실수 z보다 크거나 같은 최소 정수이다[10].

zero 단항식은 $e(X)$의 정보 부분 혹은 순환 이동의 정보 부분이 zero인 에러 $e(X)$를 카버링한다.

(4)에러 트래핑 디코딩 알고리즘

카버링 세트 $S=\{0\}\cup\{X^{n-k+a},\ 0\leq a<k\}$이 주어졌을 때 디코딩 과정은 아래의 단계와 같다. 여기서 신드롬 $S(X)$의 Hamming 웨이트 $W[S(X)]$는 $S(X)$의 영이 아닌 상수이다.

[제 1 단계] 신드롬 $S(X)$와 모든 a에 대한 $U_a(X)$를 계산한다. 만약 $W[S(X)]\leq t$ 이면, $e(X)=S(X)$이고 에러를 정정한다.

[제 2 단계] $W[S(X)]\geq t$일때 어떤 a에 대해서 $U_a(X)$의 $n-k-t+1$개 이상의 상수가 ε와 동일하다면 $e(X)=\varepsilon X^{n-k+a}+S(X)+\varepsilon\rho_a(X)$가 되고 에러를 정정한다.

[제 3 단계] 위의 [제 1 단계] [제 2 단계]가 성립하지 않는다면 $1<i<n$에 대해 $S^{(i)}(X)=XS^{(i-1)}(X)\bmod g(X)$를 계산한 후 위의 과정

을 되풀이 한다.

만약 n번 순환 이동 후 에러가 정정되지 않는다면 정정 불가능한 에러 패턴이 검출된 것이다.

3. (15, 11) Reed-Solomon 코드의 디코더

(1) 인코더 설계

$GF(2^4)$의 원시 소자(primitive element)를 α라 두면 원시 기약(primitive irreducible) 다항식이 $f(X)=X^4+X+1$이므로 $\alpha^4=\alpha+1$이 성립한다. 이를 바탕으로 $GF(2^4)$의 Field 소자를 표현하면 표(1)과 같이 나타난다.

정보 다항식을 $i(X)=C_{2t}X^{2t}+C_{2t+1}X^{2t+1}+\cdots+C_{n-1}X^{n-1}$, 패리티 검사 다항식을 $P(X)=C_0+C_1X+\cdots+C_{2t-1}X^{2t-1}$이라 두자. 그러면 인코딩된 RS코드 다항식은 다음과 같이 표현된다.

$$V(X)=i(X)+p(X) \tag{8}$$

표 1 $GF(16)$의 소자
The element of GF(16)

GF element	Notation	Binary Notation
0	0	0000
1	1	0001
α^1	2	0010
α^2	4	0100
α^3	8	1000
α^4	3	0011
α^5	6	0110
α^6	12	1100
α^7	11	1011
α^8	5	0101
α^9	10	1010
α^{10}	7	0111
α^{11}	14	1110
α^{12}	15	1111
α^{13}	13	1101
α^{14}	9	1001

$V(X)$는 또한 $g(X)$의 곱으로 이루어지고, 결국 인코딩 과정은 $i(X)$와 $g(X)$로 부터 $p(X)$를 얻는 과정이며 다음과 같이 나눗셈 알고리즘에 의하여 얻는다.

$$i(X)=q(X)\ g(X)+r(X) \tag{9}$$

$p(X)=-r(X)$라 두면 $q(X)g(X)=i(X)-r(X)=i(X)+p(X)=V(X)$가 되고 결국 $V(X)$는 식(8)에서 $p(X)=-r(X)$로 두어 구해진다.

(15, 11) RS 코드의 생성 다항식은 식(2)로부터 다음과 같이 구해진다.

$$g(X)=(X-\alpha^2)\ (X-\alpha^3)\ (X-\alpha^4)\ (X-\alpha^5)$$
$$=X^4+\alpha^{14}X^3+\alpha^8X^2+\alpha^6X+\alpha^{14} \tag{10}$$

그림 2에 (15, 11) RS 코드의 인코더 구조를

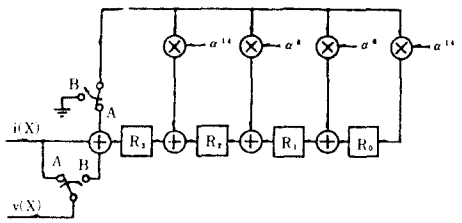

그림 2 (15, 11) RS코드의 인코더
Encoder oof the (15, 11) RS code.

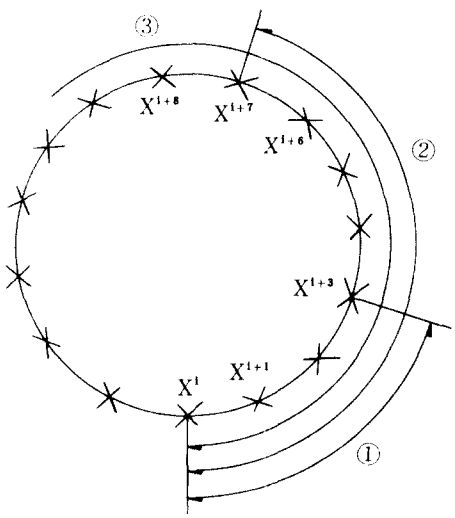

그림 3 (15, 11) RS 코드의 covering 범위
covering range of the (15, 11) RS code.

나타냈으며, 연결된 사각의 선은 4 선이고 R_i ($0 \leqq i \leqq 3$)는 4 bit 레지스터를 나타낸다. 초기에 모든 레지스터는 0으로 저장되어 있고 스위치는 A의 위치에 있다.

정보 심볼 C_{15}, C_{14}, …, C_4가 인코더의 다항셈 회로에 입력되는 동시에 채널로 나간다. 또의 상수가 생성되어 레지스터에 순차적으로 저장되고 나머지 상수가 연속적으로 계산된다.

C_4가 회로에 입력되는 순간 스위치는 위치 B로 연결되고 계속해서 C_3, C_2, C_1, C_0가 인코더로부터 전송된다. 스위치가 zero로 연결되어 있으므로 레지스터의 내용은 변하지 않는다.

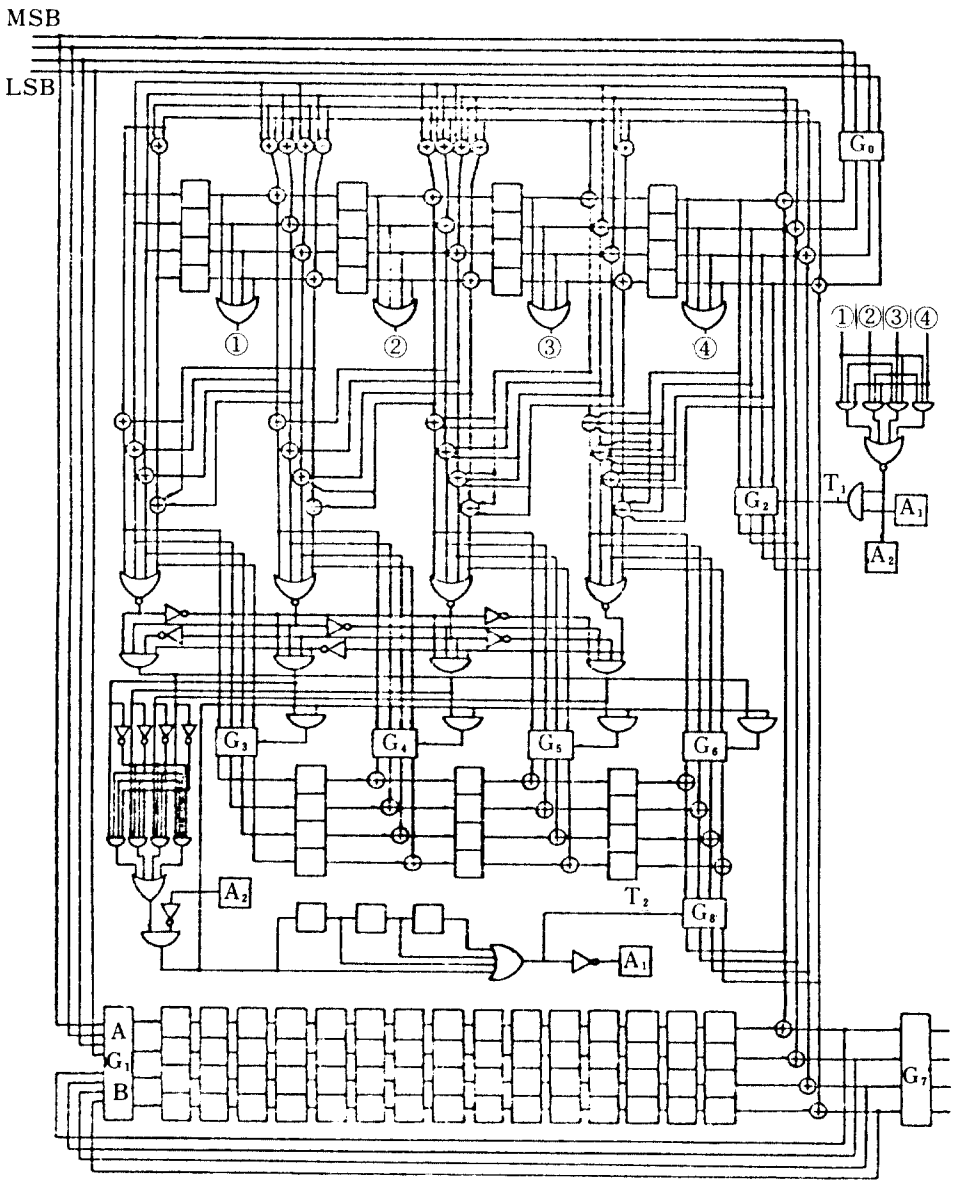

그림 4 (15, 11) RS 코드의 디코더
Decoder of the (15, 11) RS code.

(2) 디코더 설계

2-에러 정정 (15, 11) RS 코드의 카버링 세트는 $\{0, X^7\}$으로 주어진다. 어떤 2개의 심볼 에러 패턴 $e_iX^i+e_jX^j$ ($i<j$)를 생각하면 다음과 같이 카버링되고 그림 3에 도시하였다.

① $j-i\leq 3$: zero단항식에 의해 카버링.

② $4\leq j-i\leq 7$: X^7에 의해 카버링.

③ $j-i\geq 8$: 순환 이동에 의해 카버링.

X^7을 생성 다항식 $g(X)$로 나눈 나머지는 다음과 같다.

$$\rho(X)=\alpha^{10}X^3+\alpha^{13}X^2+\alpha^{13}X+\alpha^{11} \tag{11}$$

여기서 상수 $\hat{U}_i$를 $\hat{U}_i=S_i+\rho_i\rho_{i+1}^{-1}S_{i+1}$ ($0\leq i\leq 3$)으로 정의하면, $n-k-t+1=3$이므로 3개의 U_i가 0일때 두개의 연속적인 $\hat{U}_i$가 0이어서 디코딩 과정을 단순화 시킨다. $\hat{U}_i$를 $0\leq i\leq 3$에 대하여 아래에 구했다.

$$\hat{U}_0=S_0+\alpha^{13}S_1$$
$$\hat{U}_1=S_1+S_2$$
$$\hat{U}_2=S_2+\alpha^3S_3$$
$$\hat{U}_3=S_3+\alpha^9S_3^{(t-1)}$$

위의 이론을 토대로 디코더를 구성하면 그림 4와 같으며, Galois Field의 곱셈및 덧셈은 그림 5와 같이 된다.

그림 4의 디코딩 과정은 다음과 같다. 초기에 모든 게이트들은 "OFF"되어 있다.

〔제 1 단계〕 채널로 부터 들어오는 15개의 심볼은 버퍼 레지스터와 신드롬 레지스터로 동시에 들어오며 이 수행 과정 동안 G_0, G_1의 A단자를 "ON"시킨다.

〔제 2 단계〕 16번째 순환 이동부터 45번째 순환 이동까지 G_0이 "OFF"되고, G_1의 B 단자를 "ON"시킨다.

1) 만약 적어도 두개의 S_i가 0이면 $T_1=1$이 되고 G_2를 "ON"시켜 에러를 정정한다.

2) 만약 $T_2=1$이면 (연속적인 두개의 U_i가 zero) G_3, G_4, G_5, G_6를 "ON"시켜 4번의 순환 이동을 수행하여 에러를 정정한다.

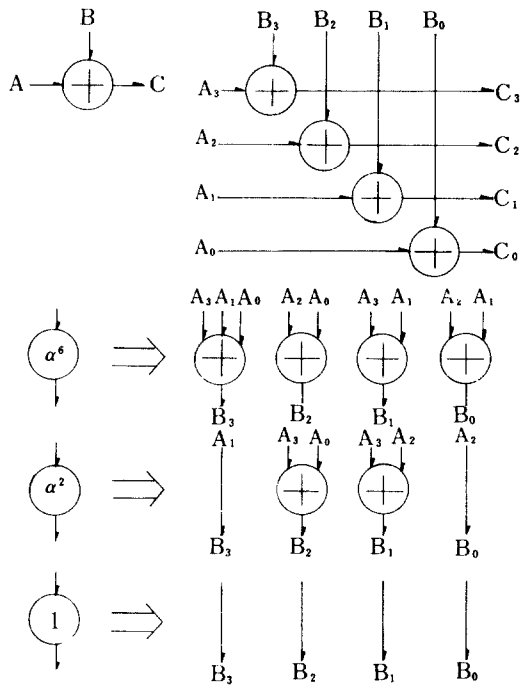

그림 5 Galois Field의 곱셈과 덧셈 예
The example of the Galois Field multiplication and addition.

〔제 3 단계〕 45번의 순환 이동이 끝난 뒤, 신드롬 레지스터의 값이 0이 되지 않으면 에러를 정정하지 못한 것으로 수신 코드 워드를 버리게 된다. 위의 과정이 끝나면 15번 순환 이동 동안 G_7이 "ON"되고 나머지 게이트는 "OFF" 되어 정정된 코드 워드가 빠져 나간다.

하나의 수신 코드 워드를 디코딩하는데는 60번의 순환이동이 필요하다. 그렇지만 〔제 1 단계〕와 〔제 3 단계〕는 동시에 수행 가능하므로 하나의 코드워드를 디코딩하는 데는 45번의 순환 이동이 필요하다.

4. 성능 분석

통신 채널이 BSC(Binary Symmetric Channel)이고 채널에서 발생하는 에러는 랜덤하고 서로 무관하다고 가정할 때, (15, 11) RS 코드의 성능을 알아 보았다.

〈Notation〉

P_B : 1 bit의 에러가 발생할 확률

P_S : 1 심볼의 에러가 발생할 확률

P_{ND} : 코드를 사용하지 않고 4 개의 심볼을 채널로 통해 그대로 전송했을 때 발생하는 에러의 확률.

P_{NC} : (15, 11) *RS* 코드의 정정 안될 에러 확률

P_{ICD} : (15, 11) *RS* 코드의 디코딩 에러 확률

P_B를 주었을 때, 식(12)에서 식(15)까지 위의 각 확률을 구하면?

$$P_S = \sum_{i=1}^{4} \binom{4}{i} P_B^i (1-P_B)^{4-i} \tag{12}$$

$$P_{ND} = \sum_{i=1}^{11} \binom{11}{i} P_S^i (1-P_S)^{11-i} \tag{13}$$

$$P_{NC} = \sum_{i=3}^{15} \binom{15}{i} P_S^i (1-P_S)^{15-i} \tag{14}$$

$$P_{ICD} = \sum_{h=5}^{15} P_{ICD}(h) \tag{15}$$

표 2 (15, 11) RS 코드의 웨이트 분포
Weight distribution of the (15, 11) RS code.

Weight	Number
0	1
1	0
2	0
3	0
4	0
5	45045
6	825825
7	1.689188E+07
8	2.511476E+08
9	2.934183E+09
10	2.642313E+10
11	1.801594E+11
12	9.007961E+11
13	3.118141E+12
14	6.68173E+12
15	6.681731E+12

표 3 (15, 11) RS 코드의 performance
performance of the (15, 11) RS code.

BIT ERROR	UNCODED WORD	(7,3)R-S	DECODING ERROR
.1	.9903018	.9322337	.3313106
.05	.8953261	.5456675	.1813583
.04	.8340665	.3986287	.1297584
.01	.3573885	1.950317E-02	5.887159E-03
.008	.2977147	1.081162E-02	3.245294E-03
.005	.1979239	2.975739E-03	8.856322E-04
.003	.1238329	6.965373E-04	2.061113E-04
.001	4.306713E-02	2.796533E-05	8.227207E-06
.0008	3.460127E-02	1.443447E-05	4.244041E-06
.0005	2.176515E-02	3.567066E-06	1.047873E-06
.0003	1.311521E-02	7.767477E-07	2.280465E-07
.0001	4.390553E-03	2.900231E-08	8.509839E-09
.00008	3.513953E-03	1.486122E-08	4.360314E-09
.00005	2.197638E-03	3.632638E-09	1.06573E-09
.00003	1.319149E-03	7.852855E-10	2.303707E-10
.00001	4.399054E-04	2.910821E-11	8.538658E-12
.000008	3.519395E-04	1.490461E-11	4.372122E-12
.000005	2.199763E-04	3.639261E-12	1.067532E-12
.000003	1.319915E-04	7.861448E-13	2.306045E-13
.000001	4.399905E-05	2.911882E-14	8.541545E-15

$$P_{ICD}(h)=W(h)\sum_{S=0}^{2}\sum_{k=h-s}^{h+s}\sum_{r=r_1}^{r_2}\binom{h}{h-s+r}\binom{s-r}{k-h+s-2r}\cdot\binom{15-h}{r}\cdot 14^{k-h+2r}\cdot 15^{r}\cdot P(k)$$

$$P(k)=\frac{P_S^k(1-P_S)^{15-k}}{15^k}$$

$$W(h)=\binom{n}{h}\sum_{i=0}^{h-d}(-1)^i\binom{l}{i}((n+1)^{l-d+1-i}-1)$$

(여기서 $d=n-k+1$, $d\leqq h\leqq n$)

$$r_1=\max\{0,\ k-h\}$$

$$r_2=[(k-h+s)/2]$$

위 식에서 $W(h)$는 웨이트가 h인 코드 워드의 갯수를 표시하며 표(2)에 (15, 11) RS 코드의 웨이트 분포를 나타내었다. 여기서 최소 거리 d는 5임을 알 수 있다.

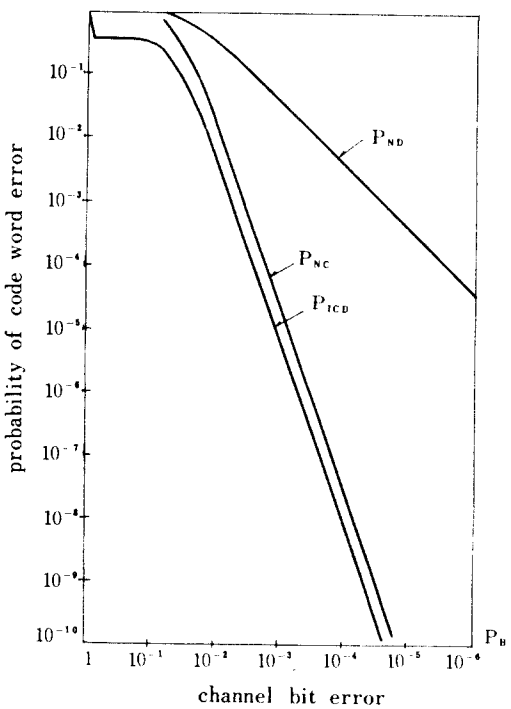

그림 6 (15, 11) RS 코드의 performance 그래프
performance graph of the (15, 11) RS code.

위 식들에 의해서 계산된 확률을 표(3)에 나타내었고, 그림(6)에서는 그래프로 나타내었다. 그래프로부터 이 코드는 채널 에러가 $5\times10^{-4}\sim5\times10^{-5}$ 일때 효율적인 것임을 알 수 있다.

5. 실험 및 결과 고찰

디코더를 구성한 후, 그림(7)과 같이 마이크로 컴퓨터(Apple-Ⅱ)와 연결하여 동작시켰다.

레지스터의 클리어과 클리어는 Apple-Ⅱ의 클럭(1 MHz)을 사용하여 만들었고 각 게이트들은 소프트웨어로 제어했다. I/O 는 8255를 사용하여 구성했으며 입력 데이터는 컴퓨터 시뮬레이션에 의해 구한 코드 워드에 에러를 발생시켜 한심볼씩(4 bit) 병렬로 전송하였다.

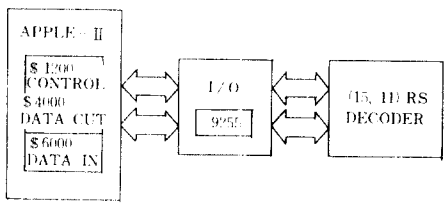

그림 7 실험 구성도
Experimental modeling.

디코딩의 타이밍 그래프를 그림(8), 실험 결과를 그림(9)에 실었다.

1개의 코드 워드를 디코딩하는 데는 1.5msec가 소요되었으며 실험 결과 2개 이하의 에러가 발생한 경우 에러가 모두 정정되었다. 또한 8개의 연속적인 이진 버스트 에러도 정정되었다.

6. 결 론

카버링 난항식 개념을 도입하여 2개의 심볼 에러를 정정할 수 있는 (15, 11) Reed-Solomon 코드의 디코더를 에러 트래핑 방법으로 설계하였다.

에러 트래핑 방법을 사용함으로써, Berlekamp 디코더에 필요한 Galois Field의 곱셈과 나눗셈

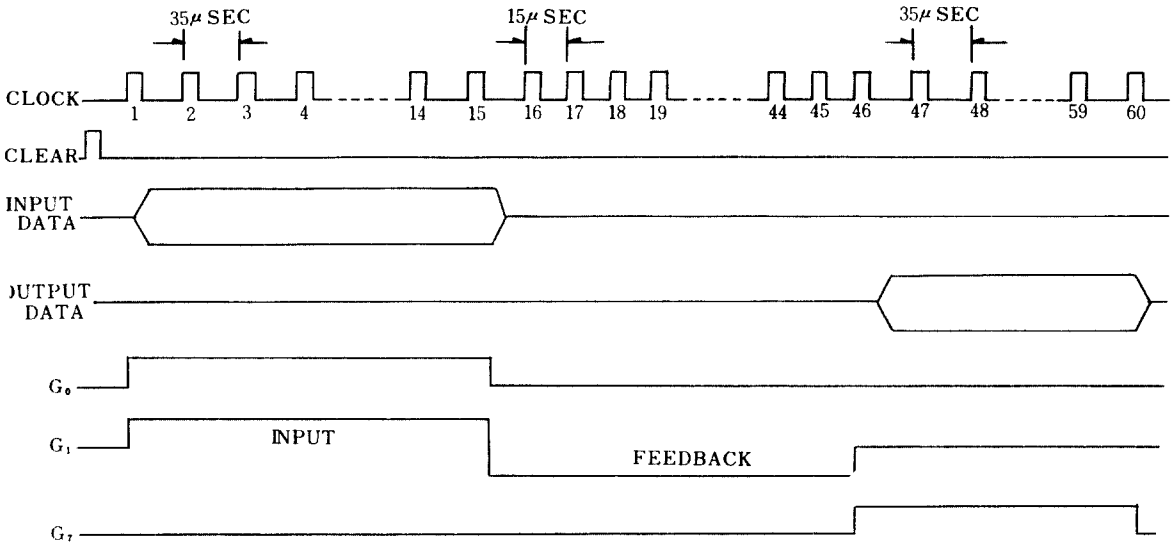

그림 8 디코더 회로의 타이밍 그래프
Timing graph of the decoder circuit.

RECEIVED C.	CORRECTED C.	INFORMATION
123456789ABA5FC	123456789ABA5FC	123456789AB
42F99C460B17F77	42699C460B17F77	42699C460B1
0C500A689B37D91	0C500A689B37191	0C500A689B3
A1745AB893D99A8	01545AB893D99A8	01545AB893D
ABF5491394FE046	ABF55A1394FE046	ABF55A1394F
5FADDA6777C3DC0	5FADFA6477C3DC0	5FADFA6477C
5DB8BA6133C5453	5DBBBA6433C5453	5DBBBA6433C
1E10B270D4F603A	1E10B240D4F003A	1E10B240D4F
5EB0E5EAEE55979	6EB0E5EAEE57979	6EB0E5EAEE5
ED1A53DD1440D21	ED1AD3DD1440D71	ED1AD3DD144
ED1AD3DD14F0D7E	ED1AD3DD1440D71	ED1AD3DD144
A391001900C72C7	A3910010000072C7	A3910010000
0F9884B1A46C5B0	00988AB1A46C5B0	00988AB1A46
1F9800E5FEFE6D9	10980015FEFE6D9	10980015FEF
779800557DB4277	77980055FDB4278	77980055FDB
83AA8FD4DEF2C3E	83AA70D4DEF2C3E	83AA70D4DEF
03AA0024DEF2C3E	03AA0023DEF2C2E	03AA0023DEF

그림 9 실험 결과
Experimental results.

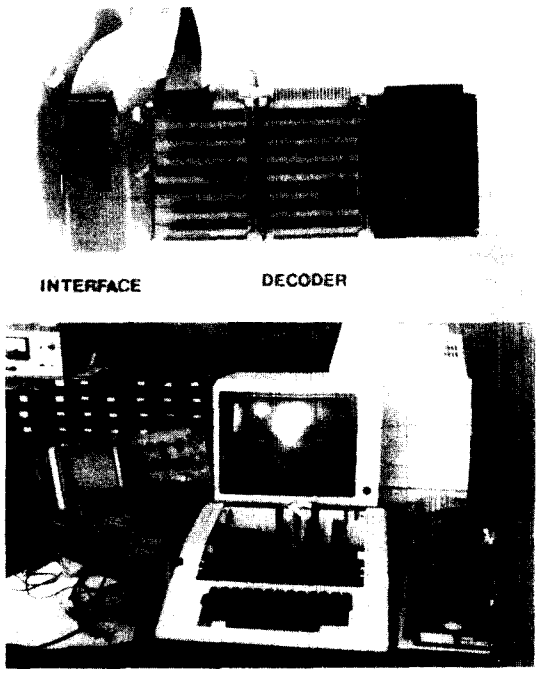

그림10 실험시스템
Experimental system.

회로를 따로 구성할 필요없이 조합 논리 회로로 간단히 디코더를 구성할 수 있었다.

실험은 구성된 회로를 마이크로 컴퓨터와 인터페이스시켜 소프트웨어로 제어하였으며, 실험결과 2개 이하의 심볼에러와 8비트의 이진 버스트 에러가 정정되었다. 디코딩에 필요한 실행은 60개로서 1.5msec가 소요되었으며 이 코드의 성능은 채널 에러가 $5\times10^{-4}\sim5\times10^{-5}$일 때 효율적인 것으로 나타났다.

Reed-Solomon 코드는 통신 위성 및 군사용 통신 시스템에 사용되고 있으며 최근에는 컴팩트 디스크(compact disc) 등 오디오 시스템에도 사용되고 있다.

많은 에러($t>3$)를 정정할 수 있는 코드의 디코더를 설계하기 위해 카버링 세트를 구하는 문제와 긴 코드의 디코더를 에러 트래핑 방법으로 간단히 구성할 수 있는 방법등이 앞으로 더 연구되어야 할 분야라고 생각한다.

參 考 文 獻

(1) S. Lin, An Introduction to Error-Correcting Codes, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.
(2) W. W. Peterson and E. J. Weldon, Error-Correcting Codes, 2nd ed., The MIT press, 1972.
(3) 홍대식, 강창언, "위성 통신에서의 심음 면역성 향상을 위한 코드의 재선," 한국 통신학회지, vol. 10, no. 3, pp. 147~153, 6월, 1985.
(4) R. E. Blahut, Theory and Practice of Error Control Codes, Addison-Wesley, 1983.
(5) L. Rudolph and M. E. Mitchell, "Implementation of decoders for cyclic codes," IEEE Trans. on Inform. Theory, vol. IT-5, pp. 114~123, Sep, 1959.
(6) S. S. Yau and Liu, "On decoding of maximun-distance Seper able linear codes," IEEE Traans. on Inform. Theory, vol. IT-17, no. 4, pp. 487~491, July, 1971.
(7) F. J. MacWilliams, "Performance decoding of systematic codes," Bell Syst. Tech. J., vol. 43, pp. 485~505, Sep. 1983.
(8) A. H. Chan and R. A. Games, "(n, k, t)-covering systems and Error-trapping decoding," IEEE Trans. on Inform. Theory, vol. IT-27, no. 5, pp. 643-646, Sep, 1982.
(9) T. Kasami, "A decoding procedure for multiple-Error correcting cyclic codes," IEEE Trans. on Inform. Theory, vol. IT-10, No. 4, pp. 134-138, April, 1964.
(10) S. Lin, Error Control Coding, Prentice-Hall, 1983.
(11) E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.
(12) 정연호, 강창언, "심음 채널에 에러 정정 코딩을 적용한 고효율 통신 시스템," 연세대학교 산업 기술연구소 논문집, 제15집, 제1권(B), pp. 63-67, 6월, 1983.
(13) E. R. Berlekamp, ""Bit-serial reed-solomon encoders," IEEE Trans. on Inform. Theory, vol. IT-28, no. 6, pp. 869-874, Nov, 1982.
(14) Z. M. Hunton and A. M. Michelson, "On the computation of the probability of post-decoding error events for block codes," IEEE Trans on Inform. Theory, vol. 23, no. 3, pp. 399-403, May, 1977.
(15) D. L. Chon, A. H. Levesque, J. H. Meyn, A. W. Pierce, "Performance of selected block and convolutional codes on a fading HF channel," IEEE Trans. on Inform. Theory, vol. IT-14, no. 5, pp. 627-639, Sep, 1968.
(16) 김치령, 강창언, "Cascade 방식을 이용한 순환 곱셈 코드의 시스템 설계," 전자공학회지, vol. 22, no. 5, pp. 24-28, 9월, 1985.
(17) 김영곤, 강창언, "다수 논리 결정자를 이용한 Reed-Muller 코드의 시스템 설계," 한국통신학회지, vol. 10, no. 5, pp. 1-9, 10월, 1985.
(18) 정연호, 강창언, "순환 곱 코드의 간단한 두단계 다수결논리 디코더," 한국통신학회지, vol. 10, no. 3, pp. 115-122, 6월, 1985.

鄭 一 石(Il Sek JEONG) 正會員
1961年 5 月20日生
1984年 2 月 : 慶北大學校 工科大學電子工學科卒業(工學士)
1984年 3 月～1986年 2 月 : 延世大學校大學院電子工學科 卒業(工學碩士)
1986年 1 月～現在 : 三星電子(株) 綜合研究所勤務(研究員)

康 昌 彦 : P. 24 참조