論 文

위성통신에서의 잡음 면역성 향상을 위한 코드의 개선

正會員 洪 大 植* 正會員 康 昌 彦**

An Improved Channel Codes for the Noise Immunity of Satellite Communication Systems

Dae Sik HONG* *and* Chang Eon KANG**, *Regular Members*

요 약 Reed-Solomon 코드의 디코더를 error-trapping 방법으로 설계했다. (7, 3) Reed Solomon 코드의 인코더및 디코더 구성시 GF(8)의 소자는 3bit 의 2 진수로 표현했다. 하드-웨어 실험은 Apple-Ⅱ(micro-computer)로 제어했으며, 인코딩하는데 걸린 시간은 350μ sec이었고, 디코딩하는데 걸린 시간은 910μ sec이었다. 실험 결과 2 개 이하의 랜덤 에러는 정정되었고, 그 보다 많은 에러는 정정되지 않았다. 또한 4 bit 의 binary burst 에러도 역시 정정되었다. 그리고 (7, 3) Reed-Solomon 코드의 performance를 측정한 결과, 채널 에러가 10^{-3}~10^{-4} 일때 에러 확률이 약 10^{-7}~10^{-9} 정도로 감소되었다.

ABSTRACT The error-trapping decoder is constructed for the (7, 3) Reed-Solomon code. The syndrome register is constructed with the encoder and the substantial test logic circuits. The element of GF(8) is represented by the triple D-flip-floops. The hardware is constructed. And it is controlled by the micro computer(Apple Ⅱ). The time for the encoding and the decoding were 350μ secs and 910μ secs respectively. The experimental results show that the two symbol errors were corrected and 4-bit-binary-burst errors were also corrected.

1. 서 론

에러 정정 코드는 코드 워드간의 최소 거리가 클수록 에러 정정 능력이 커진다[1], [2]. Reed-Solomon 코드는 최소 거리가 가장 큰 코드이다. 즉, Reed-Solomon 코드는 가장 강력한 에러 정정코드이다[1]~[4].

1960년 Reed와 Solomon에 의해 Reed-Solomon 코드가 처음으로 만들어진 후, Berlekamp 등에 의해 디코딩 방법이 연구되었으며[4], 최근에는 위성 통신이나 콤퓨터의 기억 소자간의 통신에 많이 사용되고 있다.

순환 코드의 한 디코딩 방법인 error-trapping 방법은 짧은 코드에 대해서 매우 효율적인 방법이다[1]. 특히, 길이가 짧은 Reed-Solomon 코드의 경우, error-trapping 방법이 Berlekamp 디코더 보다 적용시키기가 더 쉽다.

본 연구에서는 (7, 3) Reed-Solomon 코드의 인코더와 디코더를 설계했다. 인코더 회로는 일반적인 순환 코드의 인코더로 구성했고, 디코더

*, ** 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 : 85-19 (接受 : 1985. 5. 16)

는 error-trapping 방법을 사용해서 구성했다.

실험 회로는 Apple-Ⅱ를 사용해서 clock과 control을 했으며 Apple-Ⅱ의 기억소자를 이용해서 Data를 저장했다.

2. Reed-Solomon 코드

Reed-Solomon 코드는 BCH 코드의 특수한 형태이다.

생성 다항식 $g(X)$는 식-(1)과 같이 된다. $g(X)$의 차수는 $d-1$이다.

$$g(X)=(X-\alpha)(X-\alpha^2)\cdots(X-\alpha^{d-1}) \quad (1)$$

그러므로 이 생성 다항식에 의해 생성된 코드의 길이 n은 $q-1$이고, redundancy 부분은 $d-1$이며 최소 거리는 d이다. (여기서 α는 GF(q)의 한 원소이다.)

이 때 (7, 3) Reed-Solomon 코드의 생성 다항식을 구하면 식-(2)와 같이 나타난다.

$$g(X)=X^4+\alpha^3X^3+X^2+\alpha X+\alpha^3 \quad (2)$$

표 1에서는 BCH 코드와 Reed-Solomon 코드를 비교해서 나타냈다.

표 1 Reed-Solomon 코드의 성질
Characteristic of Reed-Solomon code.

	BCH	Reed-Solomon
$g(X)$의 근	GF(q^m)	GF(q)
$g(X)$의 계수	GF(q)	GF(q)
코드 길이	$n=q^m-1$	$n=q-1$
패리티 길이	$n-k\leq mt$	$n-k=2t$
최소거리(dmin)	$d\geq 2t+1$	$d=2t+1=n-k+1$

3. (7, 3) Reed-Solomon 코드의 설계 및 실험

본 장에서는 (7, 3) Reed-Solomon 코드의 performance를 분석했으며, 코드의 인코더와 디코더를 설계하고 computer simulation에 의해 타당성을 규명했다. 또한 회로를 구성해서 이론과 부합됨을 살펴보았다.

3.1 performance 분석

채널을 BSC라고 가정한다. 코드의 각 symbol은 3bit의 binary로 나타냈으며, 채널에서 발생하는 에러는 랜덤하고, 서로 무관하다고 가정한다.

여기서

P_B : 1bit 에러확률

P_S : 1 symbol 에러확률

P_{NC} : (7, 3) R-S 코드의 정정 안된 확률.

P_{DE} : (7, 3) R-S 코드의 디코딩 에러확률[1]

P_{ND} : 코드를 사용치 않은 경우의 에러확률

이라고 할때, 이들의 값은 다음 식들에서 나타냈다.

$$P_S=\sum_{i=1}^{3}\binom{3}{i}P_B^i(1-P_B)^{3-i} \quad (3)$$

$$P_{ND}=\sum_{i=1}^{3}\binom{3}{i}P_S^i(1-P_S)^{3-i} \quad (4)$$

$$P_{NC}=\sum_{i=3}^{7}\binom{7}{i}P_S^i(1-P_S)^{7-i} \quad (5)$$

$$P_{DE}=\sum_{H=5}^{7}P_{DE}(H) \quad (6)$$

$$P_{DE}(H)=W(H)\cdot\sum_{K=H}^{2}\sum_{k=H}^{H+S}\sum_{r=r_1}^{r_2}\binom{H}{H-S+r}\cdot\binom{S-r}{K-H+S-2r}\cdot\binom{7-H}{S}\cdot(q-2)^{K-H+S-2r}\cdot(q-1)^r\cdot P(K)$$

$$P(K)=\frac{P_S^K(1-P_S)^{7-K}}{(q-1)^K}$$

여기서 $W(H)$는 웨이트가 H인 코드워드의 갯수이다.

위의 각 확률값은 P_B에 대해 나타내면 그림 1과 같이 된다. 이 경우 채널 에러가 $10^{-3}\sim10^{-4}$일 때 코드를 사용하면 에러의 확률(P_{NC})을 $10^{-7}\sim10^{-9}$정도로 줄일 수 있음을 알 수 있다. 또한 실제 디코딩된 확률(P_{DE})은 10^{-20} 가까이 되므로 실제 통신에서 이것은 무시할 수 있다고 본다.

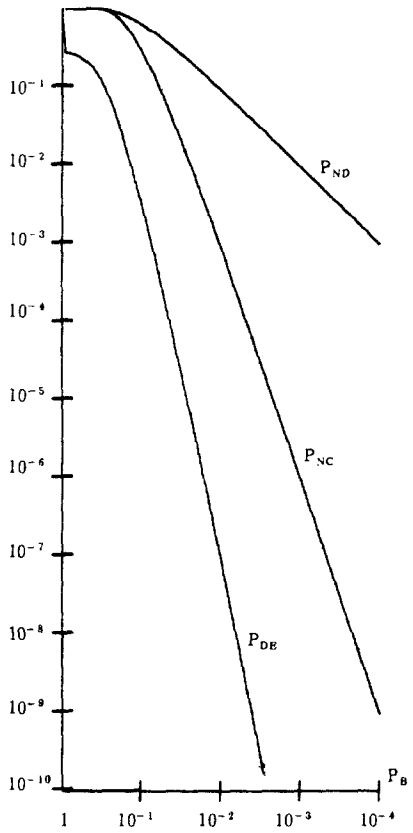

그림 1 Performance of Reed-Solomon코드
Performance of Reed-Solomon Code.

3.2 인 코 더

먼저 4 ($= n - k$)단의 shift register를 사용해서 $X^4 i(X)$를 $g(X)$로 나누는 회로를 구성한다. 여기서 생성다항식 $g(X)$는 앞에서 구한 식 (2)를 사용했다. 이 때 $h(X)$가 $X^4 i(X)$를 $g(X)$로 나눈 나머지라고 하면, $X^4 i(X) + h(X)$가 코드워드가 된다. 즉, 정보의 갯수는 3개이고, redundancy의 갯수는 4개가 되며, 총 7개의 심볼로 한개의 코드워드가 구성된다.

그림 2에서는 인코더 회로와 I/O회로를 나타냈다. 회로의 동작과정은 다음과 같다.

1. 단계 : $G_1 \rightarrow$on, $G_2 \rightarrow$on, $G_3 \rightarrow$off 3개의 정보 심볼을 shift register와 채널로 연속적으로 보낸다. (1단계 과정이 바로 $X^4 i(X)$를 $g(X)$로 나누는 과정이다.

2. 단계 : $G_1 \rightarrow$off, $G_2 \rightarrow$on, $G_3 \rightarrow$off shift register에 남아있는 나머지 4개의 심볼을 채널로 내보낸다.

인코딩 하는 데는 7개의 clock이 필요하다.

3.3 디 코 더

수신된 코드워드 $r(X)$를 미리 X^4만큼 곱한후 (즉, 4번 오른쪽으로 shift 시켜서) 신드롬 레

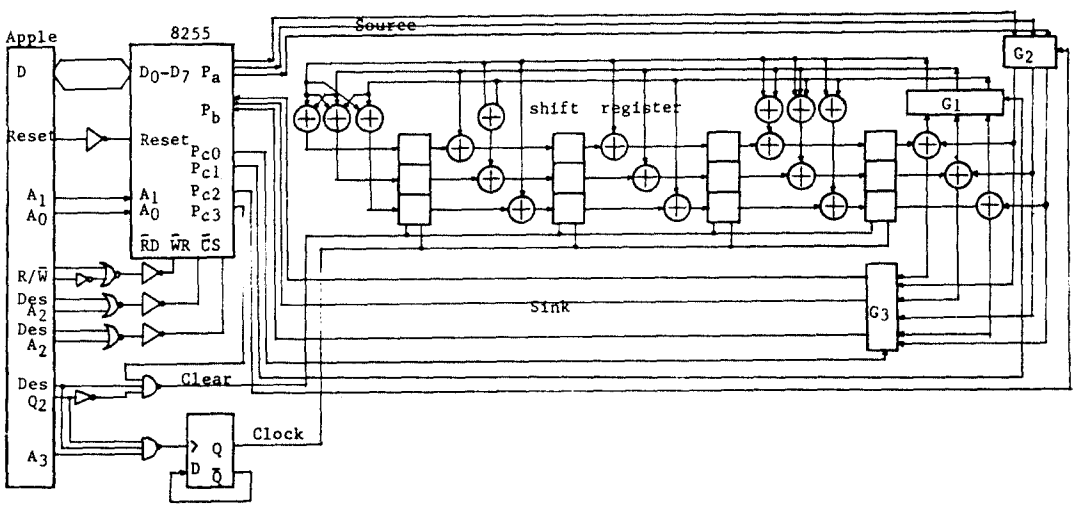

그림 2 인코더
Encoder.

지스터에 입력시킨다. 이 때 신드롬 레지스터에 나타나는 에러의 형태는 수신된 코드워드의 앞 부분 4개의 심볼의 에러형태에 해당한다. 그러므로 신드롬 레지스터의 웨이트가 2이하인 경우(에러가 2개 이하인 경우), 신드롬 레지스터의 제일 오른쪽 값은 버퍼 레지스터의 오른쪽 값에 더해져서 정정과정을 수행하게 된다.

그림 3의 디코더 회로는 다음과 같이 동작한다.

1단계 : G_1→on, G_2→channel
G_3→off, G_4→off

test logic에 의해 신드롬의 웨이트를 측정한다.

1) 3단의 웨이트가 1이하이면 에러가 발견되었다고 간주하며, test logic의 출력은 'YES'로 되면서 버퍼 레지스터의 첫단의 값이 고쳐진다. 또한 shift register로 에러값이 feed-back되어 에러를 reset시켜 준다.

2) 3단의 웨이트가 1보다 큰경우 에러가 발견되지 않았다고 간주되며, test logic의 출력은 'No'가 되며 버퍼 레지스터와 신드롬 레지스터는 한번 더 cyclic shift 한다.

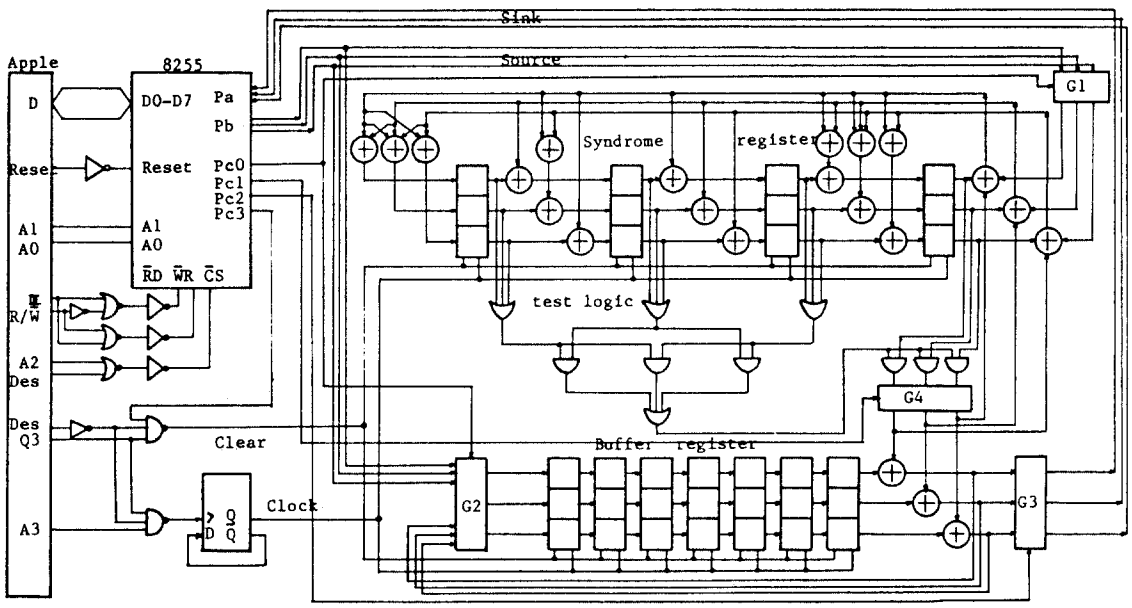

그림 3 디코더
Decoder.

제일로 부터 들어오는 7개의 심볼을 버퍼 레지스터와 신드롬 레지스터에 동시에 입력시킨다. 7개의 심볼이 다 들어간 직후의 신드롬 레지스터의 값이 수신된 코드워드의 신드롬이다.

2단계 : (8 th shift~14-th shift)
G_1→off, G_2→feed back
G_3→off, G_4→on

3단계 : (15-th shift~21 th shift)
G_1→off, G_2→off
G_3→on, G_4→on
2단계와 같은 정정 동작을 계속하게 된다.

4단계 : 21번의 shift가 끝난 뒤 신드롬 레지스터의 값이 '0'이 되지 않으면, 에러는 정정하지 못한 것이다. 즉, 에러의 갯수가 2개보다 많은 경우이다. 이 경우의 코드워드는 버린다.

디코딩에 필요한 clock의 갯수는 21개이다.

4. Computer simulation

2절과 3절에서 설계한 인코더와 디코더 회로를 computer simulation에 의해 검토했다. 그림 4에서는 그 흐름도를 나타냈다.

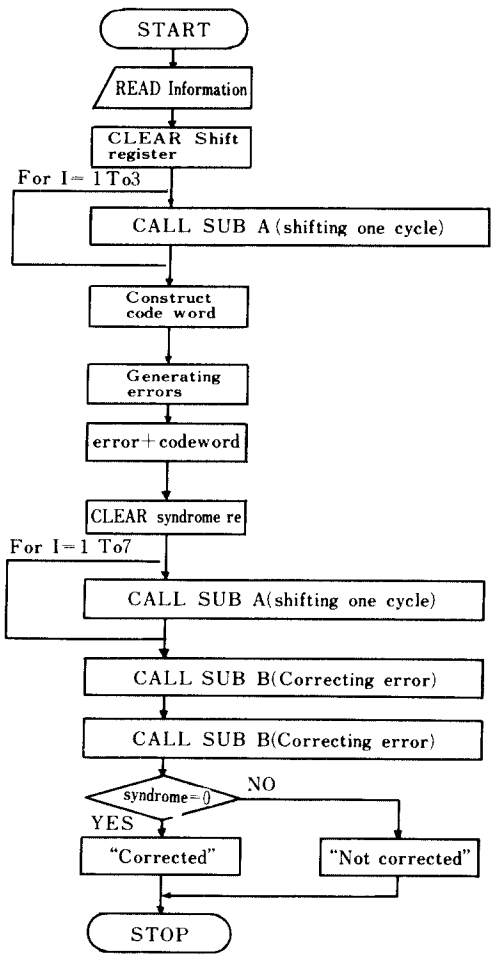

그림 4 흐름도
Flow chart.

5. 실험 결과 및 고찰

본 논문에서는 그림 5와 같이 micro-computer (Apple Ⅱ)를 사용해서 인코더와 디코더를 제어했다.

마이크로 콤퓨터와 인코더(디코더) 간의 데이타 교환은 I/O 회로를 통해서 했으며, 인코더와 디코더의 clock과 clear는 Apple Ⅱ의 clock과 software를 부과해서 구성했다.

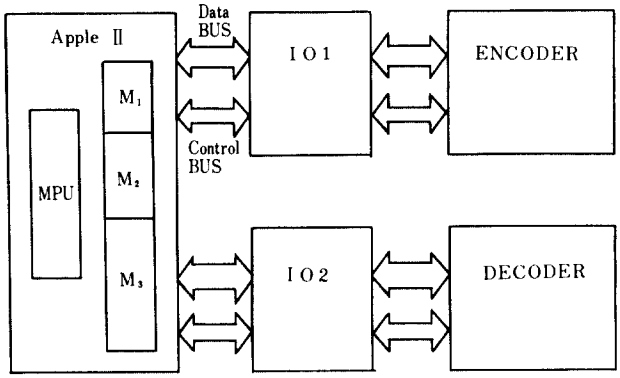

그림 5 실험도
Block diagram of experiments.

인코더와 디코더에 사용한 GF(8)의 소자는 3 bit의 binary로 실현했다. 즉, GF(8)의 한 소자 $\alpha^5(=7)$은 (xxxxx111)로 나타냈다. 이 데이타들은 데이타 버스를 통해 병렬로 전송되며, I/O와 인코더(디코더) 간은 하위 3 bit만 연결하고, I/O와 Apple Ⅱ와는 8 bit 전부 연결해서 전송한 후 하위 3 bit만 처리하도록 했다.

실험시 인코더와 디코더의 timing은 그림 6과 그림 7에서 나타냈다. 이때 clock의 주기가 다른 이유는 software로 clock을 조절하기 때문이다. 즉, macro program의 수행시간이 다르기

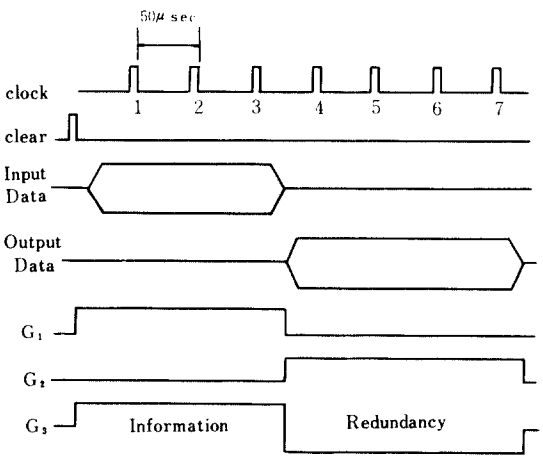

그림 6 인코딩
Timing diagram of the encoder.

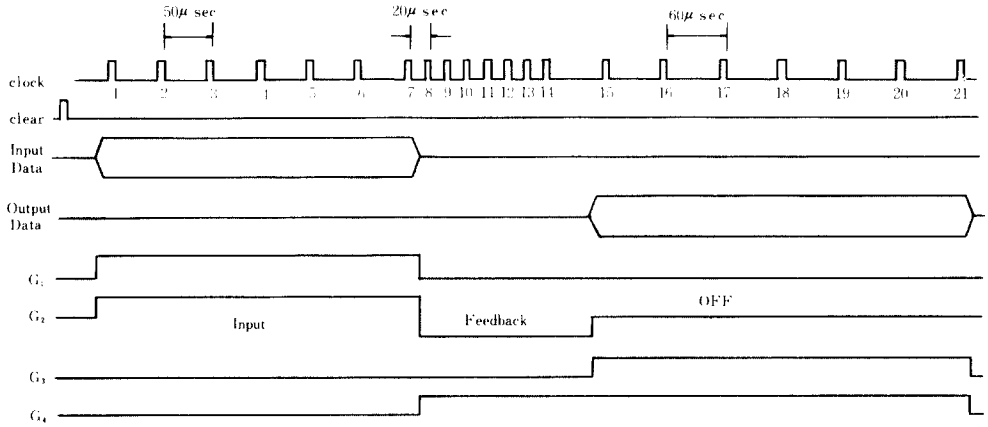

그림 7 디코딩
Timing diagram of the decoder.

때문이다. 실제 통신장비에 hardware 적으로 clock과 clear를 구성하면 이 문제는 없어질 것이다.

실험방법은 먼저 Apple Ⅱ의 기억장소 $ 4000번지에 정보심볼을 저장해 놓는다. 그 값을 I/O 1을 통해서 인코더에 보낸다. 인코더에서는 3번 cyclic shift 시킨 후 shift register에 남아있는 값을 I/O 1을 통해 Apple Ⅱ의 $ 6000번지에 기억시킨다. 그 뒤 software로 $ 4000 번지와 $ 6000번지에 놓여있는 codeword에 랜덤에러를 부과해서 그 값을 I/O 2를 통해 디코더에 내보낸다. 21 clock이 지난 후 그 값을 I/O 2를 통해서 Apple Ⅱ의 $ 9000 번지에 저장한다.

인코딩 하는데는 7 clock이 소요됐으며, 디코딩 하는데는 21 clock이 소요됐다. 1개의 코드워드를 인코딩 하는데는 350μ sec가 걸렸고, 디코딩 하는데는 910μ sec가 걸렸다.

그림 8에서는 인코더, 디코더 및 I/O 회로를 나타냈다.

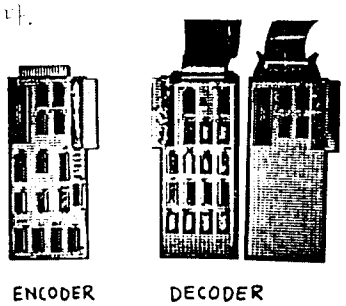

그림 8 실험회로
Circuits.

표 2 실험결과
Resoits.

INFORM	CODE W.	RECEIVED	SINK
001	0013123	0013223	001
005	0054514	0154514	005
010	0104155	0104155	010
017	0176507	0176507	017
027	0271543	0271543	027
043	0433747	0443117	443
061	0616710	0616710	061
105	1052473	1052413	105
122	1223130	1223130	122
141	1413466	1413466	141
144	1447172	1447172	144
157	1576342	1576342	157
175	1753315	1753315	175
205	2053761	2053761	205
221	2217147	2217147	221
237	2373663	2373663	237
252	2525544	2525544	252
272	2720755	2720755	272
300	3001312	3001312	300
311	3116364	3116364	311
333	3333333	3333333	333
337	3374704	3374704	337
357	3571613	3575613	
375	3754163	3754163	375
406	4064220	4064220	406
421	4215763	4215763	421
432	4324553	4324553	432
446	4462602	4162602	446
457	4575674	4575674	457
472	4722171	4722171	472
475	4750623	4750623	475
510	5107463	5107463	510
536	5365003	5365003	536
567	5674457	5674457	567
603	6037541	6437741	603
613	6133414	6133414	613
617	6174023	6174021	617
620	6201435	6201435	620
642	6422040	6422046	642
656	6561522	6561523	656
677	6771610	6377771	677
724	7240165	7240162	724
737	7371355	7371355	737
744	7445756	7445756	744
761	7612053	7612053	761
765	7655464	7655464	765
767	7673622	7673622	767
777	7777777	7777777	777

표 2에서는 실험결과를 나타냈다. 실험시 I INFORM의 값은 총512개(2^9개)의 정보량 중에서 임의로 선택한 값이고 에러는 10^{-2}으로 이 결과는 computer simulation 결과와도 일치하며, 2개 이하의 에러는 정정되었고, 4 bit의 binary burst 에러도 역시 정정되었다. 또한 3개 이상의 에러가 발생한 경우 에러 정정을 하지 못했고, 표의 밑줄 그은 부분처럼 다른 코드워드로 잘못 디코딩되는 경우도 나타났다.

6. 결 론

error-trapping 방법으로 (7, 3) Reed Solomon 코드를 작성한 결과, 아주 간단히 hardware를 실현할 수 있었고, 실험도 간단히 할 수 있었다.

실험결과 2개 이하의 에러와 4 bit의 binary burst 에러를 정정할 수 있었다.

앞으로는 길이가 긴 Reed-Solomon 코드의 좀 더 효율적인 디코딩 방법등이 연구되어야 하겠다.

參 考 文 獻

(1) S. Lin, "An introduction to error-correcting codes," New Jersey : Prentice-Hall, 1970

(2) W. W. Peterson, E. J. Weldon, "Error-correcting codes," Massachusetts : MIT Press, 1972.

(3) E. R. Berlekkamp, "Algebraic coding theory," McGraw-Hill, 1968.

(4) R. W. Blahut, "Theory and practice of error control codes," Massachusetts : Addison-Wesley, 1983.

(5) S. W. Golomb, "Shift register sequences," San Fransisco : Holden Day, 1967.

(6) I. S. Reed, G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Indvs. Appl. Math., vol. 8, no. 2, pp. 300-304, Jun., 1960.

(7) R. C. Bose, D. K. Ray-chudhuri, "On a class of error correcting binary group codes," Information and control 3, pp. 68-79, 1960.

(8) S. S. Yuu, Y. C. Liu, "On decoding of maximum-distance separable linear codes," IEEE Trans. on Information Theory, vol. IT-17, no. 4, pp. 487-491, July 1971.

(9) J. L. Massey, "Shift-register synthesis and BCH decoding," IEEE Trans. on Information Theory, vol. IT-15, no. 1, pp. 122-128, Jan., 1969.

(10) R. C. singleton, "Maximum distance Q-Nary codes," IEEE Trans. on Information Theory, pp. 116-118, April 1964.

(11) Z. M. Huntoon, A. M. Michelson, "On the computation of the probability of post-decoding error events for block codes," IEEE trans. on Information Theory, pp. 399-403, May 1977.

(12) V. K. Wei, "An error-trapping decoder for nonbinary cyclic codes," IEEE Trans. on Information Theory, vol. IT-30, no. 3, pp. 538-541, May 1984.

(13) L. Rudolph, M. E. Mitchell, "Implementation of decoders for cyclic codes," IEEE Trans. on Information Theory, vol. IT-5, pp. 114-123, Sept., 1959.

洪 大 植 (Dae Sik HONG) 正會員

1961年 1月 4日生

1979年 3月~1982年 2月 : 延世大學校工科大學電子工學科卒業.

1983年 3月~1985年 2月 : 延世大學校大學院電子工學科(工學碩士)

1985年 3月~現在 : 延世大學校電子工學科 研究助教

康 昌 彦 P. 122 와 同 一.