論文 86-23-3-3

효율적인 에러 정정을 위한 콘케티네이티드 코딩 시스템

(Concatenated Coding System for an Effective Error Correction)

康 法 周*, 康 昌 彦*

(Beob Joo Kang and Chang Eon Kang)

要 約

본 논문은 내부 코드로써 이진 코드를 사용하고 외부 코드로써 비이진 코드를 사용하여, concatenated 코딩 시스템을 구성하였다. 코드 길이가 길어짐에 따라 지수 급수적으로 회로가 복잡해짐으로, 효율적인 통신 시스템을 구성하기 위하여 이진 코드와 비이진 코드를 cascade로 연결하는 코딩 방식 방식을 이용하였고 실제로 (7, 3) burst 에러정정 콘드와 (7, 3) Reed-Solomon 코드를 사용하여 회로를 설계하고 실현하였다.

인코더는 parallel to serial 회로를 이용하였고, 디코더는 serial to parallel 회로를 이용하여 설계하였다.

(49, 9) concatenated 코딩 시스템을 설계한 후, 채널 에러에 따른 에러 확률을 구하고 다른 코드들의 에러 확률과 비교하였으며 마이크로 컴퓨터를 이용하여 실험을 하였다.

Abstract

A concatenated coding system using a binary code as the inner code and a nonbinary code as the outer code has been constructed for the purpose of error correction. The complexity of a conventional coding system grows exponentially as the code length of a block code becomes longer. To reduce the complexity for long codes, an effective communication system has been proposed by cascading two codes - binary and nonbinary codes. Using a parallel-to-serial circuit and a serial-to-parallel circuit, the concatenated coding system has been designed and constructed by employing a (7, 3) burst error correcting code as the inner code and a (7, 3) Reed-Solomon code as the outer code. This system has been simulated and tested using a micro-computer. For the (49, 9) concatenated coding system, the error probability of the channel has been evaluated and compared to different coding systems.

I. 序 論

디지틀 신호를 채널에 전송할 때, 랜덤 에러와 burst 에러가 복합적인 형태로 나타날 때와 에러가 많이 발생하는 경우에는 에러 정정 능력이 큰 코드를 요구한다. 즉, 최소 거리가 큰 코드를 요구하게 되는데 최소 거리가 커짐에 따라 일반적으로 코드 길이가 커지며 설계상의 복잡성이 지수 급수적으로 증가한다. 이러한 이유로 짧은 코드들을 이용하여 긴 코드를 구성하게 되었고, 설계상의 복잡성을 줄일 수 있는 방법들이 고안되었다. 그 중 하나로 1954년에 Elias에 의하여 곱셈 코드가 소개되었다.[3,4] 이 곱셈 코드는 두개의 이진 코드를 이용하여 구성한 코드이다.

*正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)

接受日字 : 1985年 12月 21日

다른 방법으로 1966년에 Forney는 이진 코드와 비이진 코드를 이용하여 concatenated 코드를 구성하는 방법을 소개하였다.[2]

본 논문에서는 블럭 코드와 Reed-Solomon코드를 이용하여 두단계의 concatenation을 실현하기 위하여(7,3) burst 에러 정정 코드와(7,3) Reed-Solomon코드를 이용하여(49, 9) concatenated 코드를 설계하였다. 여기에서 내부 코드 및 외부 코드의 인코더는 일반적인 순환 코드의 인코딩 방식을 적용하였고, 에러 trapping방식을 이용하여 디코더를 구성하여 두개의 디코더들을 cascade로 연결함으로써 concatenated 코드를 구성하였다. 이러한 인코더와 디코더에 대한 컴퓨터 시뮬레이션을 통하여 에러 정정 능력과 채널 에러에 따른 에러 performance를 구하고, 다른 코드들의 error performance와 비교하였다.

그리고 하드웨어 실현시, parallel to serial 회로와 serial to parallel 회로를 이용하여 구성하는 방법을 보였으며, 실험을 통하여 동작 여부를 살펴 보았다.

Ⅱ. Concatenated 코드의 구성

Concatenated 코딩은 코드 길이가 짧은 코드들로부터 코드 길이가 긴 코드를 실현하는 방법이다.

그림 1과 같이 외부 코드는 (n_2, k_2) Reed-Solomon 코드를 사용하고, 내부 코드로는 (n_1, k_1) 블럭 코드(또는 컨볼루셔널 코드)를 사용한다.

GF(2)에 대하여 최소 거리가 d_1인 (n_1, k_1) 코드를 내부 코드라 하고, GF(2^{k_1})에 대하여 최소 거리가 d_2인 (n_2, k_2) 코드를 외부 코드라 할 때, 그림2에서 II_i를 외부 코드의 i번째 정보 심벌이라 하고 III_i를 내부 코드의 i번째 검사 심벌이라 한다. 이때, 내부 코드는 II_i와 III_i에 의하여 (II_i, III_i)와 같이 구성된다.

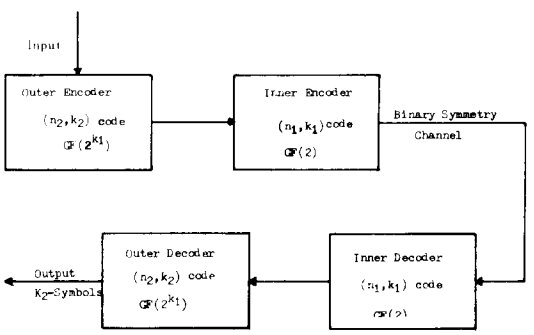

그림 1. Concatenated 코드를 이용한 통신 system
Fig. 1. Communication system using concatenated codes.

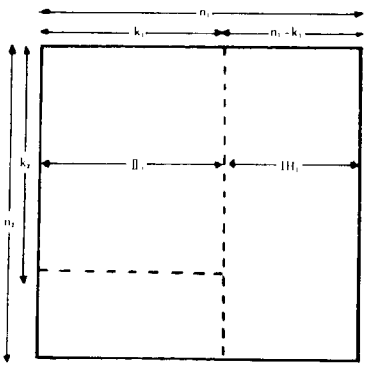

그림 2. Concatenated 코드의 2차원 배열
Fig. 2. Array of concatenated code.

벡터 II_i와 III_i는 다음과 같이 표현될 수 있다.

$$\mathrm{II}_i=(I_{i1}, I_{i2}, \cdots, I_{ik1}) \quad (1)$$

$$\mathrm{III}_i=(h_{i1}, h_{i2}, \cdots, h_{n1-k1}),\ i=1, 2, \cdots, n_2 \quad (2)$$

여기서 I_{ij}(j=1, 2, ⋯, k_1)과 h_{ij}(j=1, 2, ⋯, n_1-k_1)은 GF(2)의 소자들이다.

이렇게 하여 구성된 concatenated 코드의 매개 변수는 식(3)과 같다.

$$N=n_1n_2$$
$$K=k_1k_2$$
$$D_{min}\approx d_{1min}d_{2min}$$
$$R=k_1k_2/n_1n_2=r_1r_2$$
$$t=\left[\frac{D_{min}-1}{2}\right] \quad (3)$$

Ⅲ. 컴퓨터 시뮬레이션 및 실험

앞에서 구성한 (n_1n_2, k_1k_2) concatenated 코드의 인코딩 및 디코딩 동작을 수행해 보기 위하여, 내부 코드로 (7, 3) burst 에러 정정 코드를 이용하고 외부 코드로 (7, 3) Reed-Solomon 코드를 이용하여 (49, 9) 코드를 구성하고 인코더와 디코더의 동작을 컴퓨터에 의해 수행해 보고 마이크로 컴퓨터와 인터페이스하여 실험하였다.

1. 컴퓨터 시뮬레이션

(49, 9) concatenated 코드의 인코더와 디코더의 회로 구성은 다음과 같다.

우선, 내부 인코더와 디코더의 구성을 고찰해 보기로 한다.

(7, 3) burst 에러 정정 코드는 (7, 3) Hamming 코드와 같은 생성 다항식을 가진다. 여기서 (7,3) Hamming 코드는 최소 거리가 4이므로 한개의 에러를 정정하며, 두개의 에러가 발생한 경우 에러들을 발견할

수 있는 능력을 가지나 본 논문에서는 에러 정정 능력을 높이기 위하여 burst 에러 길이가 2인 burst 에러를 정정할 수 있는 코드로 디코더를 변형하였다. 따라서 (7, 3) burst 에러 정정 코드의 생성 다항식은 다음의 식(4)와 같다.

$$g(x)=(1+X)\,p(x) \qquad (4)$$

여기에서 p(x)는 m차의 원시 다항식이다. 따라서 m이 3인 (7, 3) burst 에러 정정 코드의 생성 다항식은 식(5)와 같다.

$$g(x)=(1+x)(1+x+x^3)$$
$$=x^4+x^3+x^2+1 \qquad (5)$$

다음으로 외부 인코더와 디코더의 구성은 다음과 같다.

t-에러 정정 Reed-Solomon 코드의 생성 다항식은 식(6)과 같다.

$$g(x)=(x+\alpha)(x+\alpha^2)\cdots(x+\alpha^{2t}) \qquad (6)$$

여기서 (7, 3) Reed-Solomon 코드는 2개의 에러를 정정하므로, (7, 3) Reed-Solomon 코드의 생성 다항식은 식(7)과 같다.

$$g(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)$$
$$=\alpha^3+\alpha x+x^2+\alpha^3x^3+x^4 \qquad (7)$$

GF(2)와 GF(8)의 관계는 표 1과 같고 식(7)의 α, α^3에 대한 값은 표 2와 같다.

그리고 내부 디코더와 외부 디코더는 에러 trapping 방법을 이용하여 구성하였다.

표 1. GF(8)의 소자
Table 1. Element of GF(8).

Eield element	Blnary notation
0	0 0 0
1	0 0 1
α	0 1 0
α^2	1 0 0
$\alpha^3=\alpha+1$	0 1 1
$\alpha^4=\alpha^2+\alpha$	1 1 0
$\alpha^5=\alpha^2+\alpha+1$	1 1 1
$\alpha^6=\alpha^2+1$	1 0 1

(M : Multiplier)

표 2. α^3, α의 곱셈
Table 2. Multiplication of α^3, α.

Octal	Input		Output		Output	
	GF(8)	$a_2a_1a_0$	M-α	$b_2b_1b_0$	M-α^3	$b_2b_1b_0$
0	0	0 0 0	0	0 0 0	0	0 0 0
1	1	0 0 1	α	0 1 0	α^3	0 1 1
2	α	0 1 0	α^2	1 0 0	α^4	1 1 0
4	α^2	1 0 0	α^3	0 1 1	α^5	1 1 1
3	α^3	0 1 1	α^4	1 1 0	α^6	1 0 1
6	α^4	1 1 0	α^5	1 1 1	1	0 0 1
7	α^5	1 1 1	α^6	1 0 1	α	0 1 0
5	α^6	1 0 1	1	0 0 1	α^2	1 0 0

(M : Multiplier)

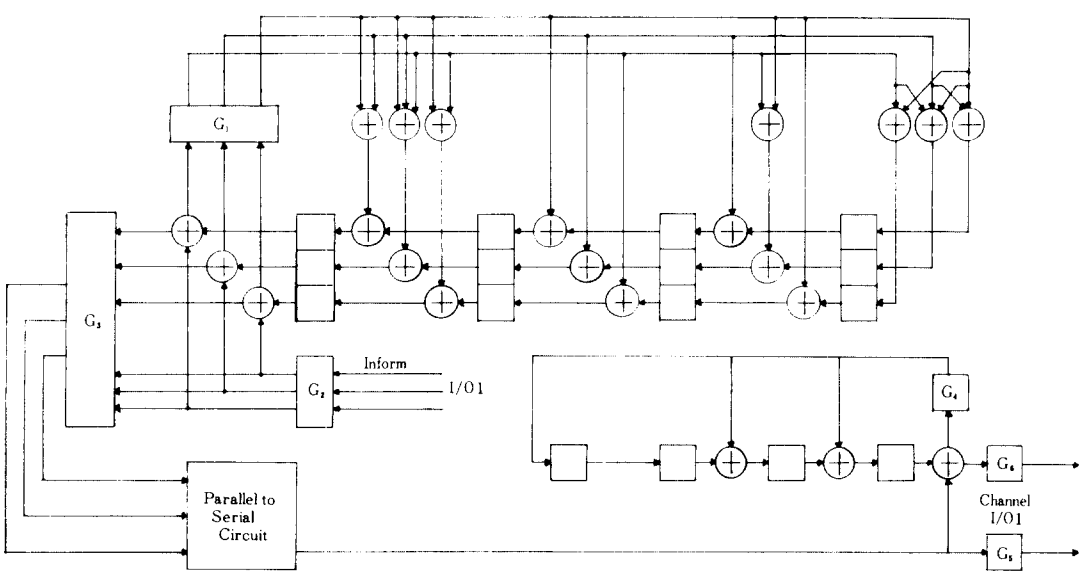

그림 3. (49, 9) Concatenated 코드의 인코더
Fig. 3. (49, 9) Concatenated code encoder.

Parallel to serial 회로와 serial to parallel 회로를 이용하여 구성한 (49, 9) concatenated 코드의 인코더 및 디코더 회로는 그림 3 과 그림 4 와 같다. 여기에서 두 코드간의 클럭 주기를 달리함으로써 두 코드의 concatenation을 실현하였다. 즉, (n_1, k_1) 내부 코드와 (n_2, k_2) 외부 코드를 concatenation할 때, 외부 인코더와 내부 인코더간의 클럭 주기의 비를 $1/n_1$으로 하고 외부 디코더와 내부 디코더 간의 클럭 주기의 비를 $1/(n_1+k_1)$으로 하여 두 코드를 cascade로 연결하였다.

이렇게 함으로써 전체적인 디코딩 시간에서, n_2개의 클럭을 줄일 수 있었고 외부 디코더에 신드럼이 형성된 후, 내부 디코더와 동일한 클럭 주기로 외부 디코더를 동작시킴으로써 빠른 디코딩 시간을 꾀하였다.

따라서 (49, 9) concatenated 코드의 인코더 및 디

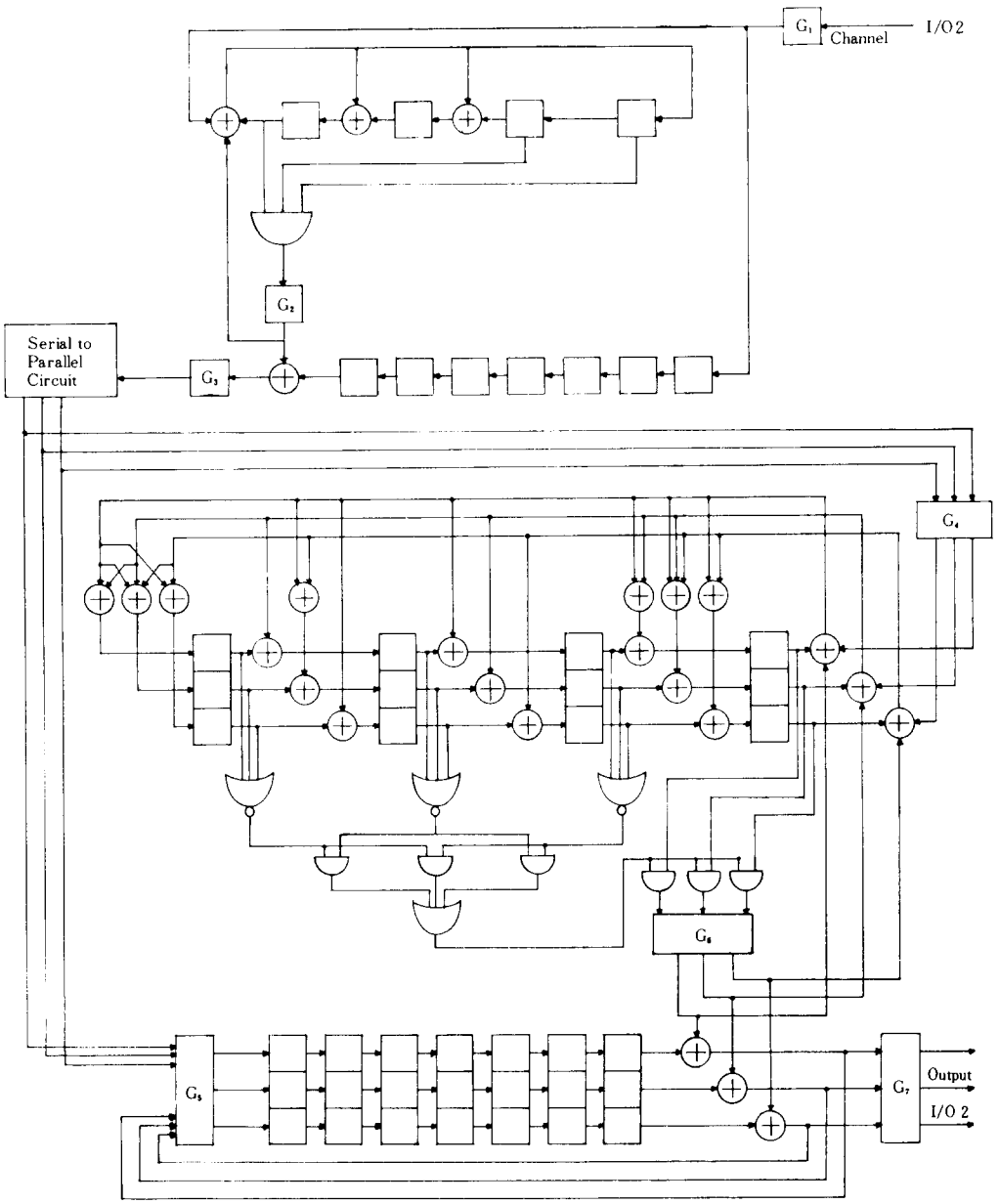

그림 4. (49, 9) Concatenated 코드의 디코더
Fig. 4. (49, 9) Concatenated code decoder.

코더에서, 외부 인코더와 내부 인코더 간의 클럭 주기의 비는 1/7이고 외부 디코더와 내부 디코더 사이의 클럭 주기의 비는 1/10이다.

그림 5 에서는 프로그램 순서도를 나타내었고, 채 에러의 확률 Pe를 0.15로 하였을 경우의 결과는 표 3 과 같다.

표 3 에서 보듯이 (49, 9) concatenated 코드의 디코더는 9 개 이하의 랜덤 에러를 정정할 수 있었으며, 길이가 12이하의 모든 burst 에러를 정정할 수 있었다. 또한 9 개 이상의 에러도 정정할 수 있는 경우가 있었는데, 이러한 경우는 에러 패턴에 따라 정정될 수 있음을 프로그램 수행 결과로서 확인할 수 있었다.

2. 실험

본 실험에서는 data의 입·출력을 원활히 수행하고 회로의 gate들을 제어하기 위해, 그리고 인코더와 디코더의 클럭을 해결하기 위하여 마이크로 컴퓨터에 인터페이스하여 실험하였다.

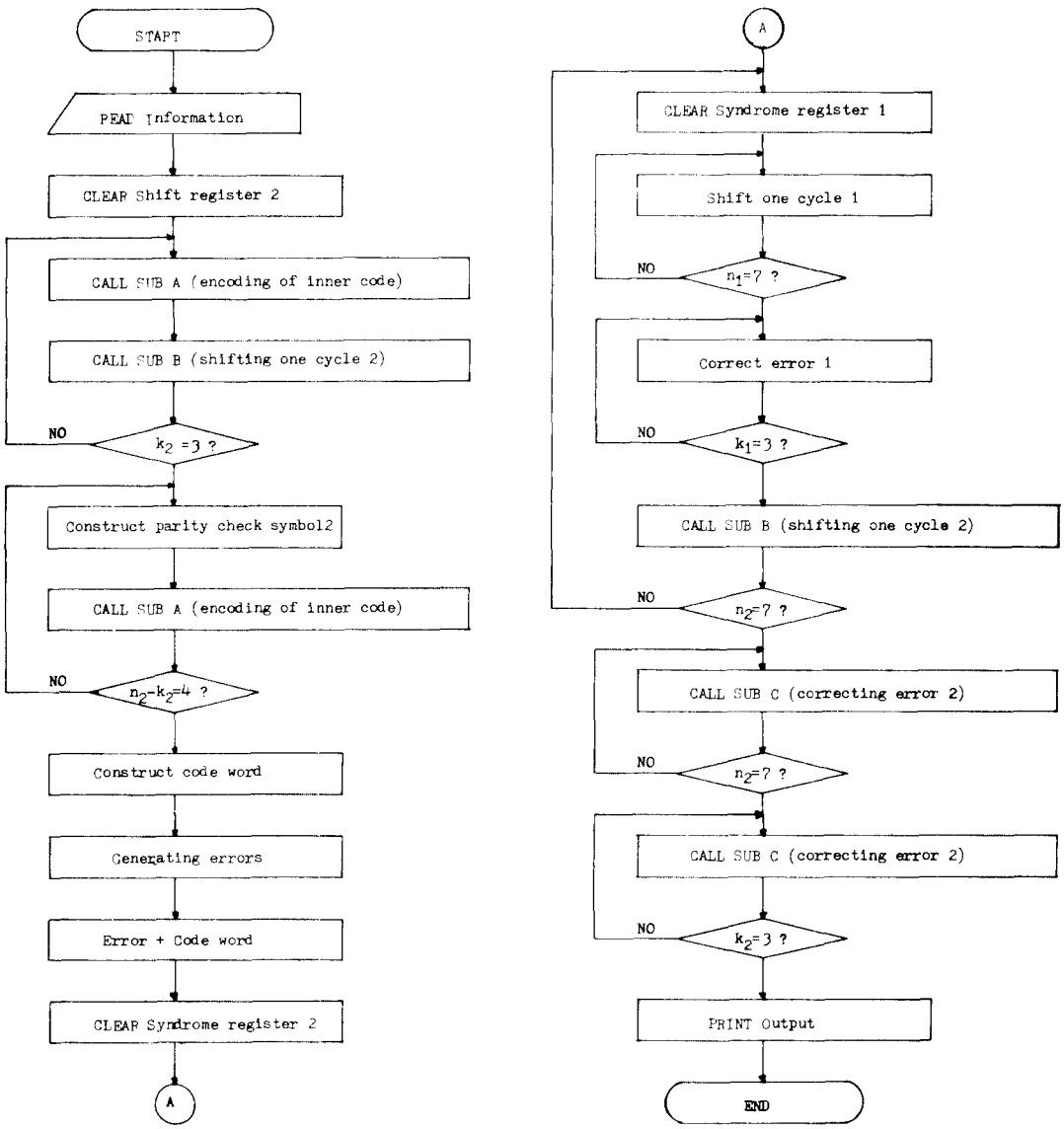

그림 5. 프로그램 순서도
Fig. 5. Flow chart.

표 3. 컴퓨터 시뮬레이션 결과
Table 3. Computer simulation result.

INFORD	INFORD	INFORD	INFORD
000	000	000	000
000	000	000	000
001	011	101	111
CODE WORD	CODE WORD	CODE WORD	CODE WORD
0000000	0000000	0000000	0000000
0000000	0000000	0000000	0000000
0011101	0111010	1010011	1110000
0011101	0111010	1010011	1110000
0100111	1101001	0011101	1010011
0100111	1101001	0011101	1010011
0111010	1010011	1001110	0100011
ERROR PATTERN	ERROR PATTERN	ERROR PATTERN	ERROR PATTERN
0010000	0000100	1100000	0000000
0001000	1000001	0000000	0000001
0000000	0000010	0100000	0110000
0000000	0001000	0100100	0000000
0010010	1000000	0000000	0000000
0000100	0010000	0000001	0010100
0111010	0001000	0110000	0010000
RECEIVED WORD	RECEIVED WORD	RECEIVED WORD	RECEIVED WORD
0010000	0000100	1100000	0000100
0001000	1000001	0000000	0000101
0011101	0111000	1110011	1000010
0111010	1011011	1101010	0100101
0001111	1111010	1010011	1110100
0100011	1111001	0011100	1000111
0000000	1011011	1111110	0110111
OUTPUT	OUTPUT	OUTPUT	OUTPUT
000	000	000	000
000	000	000	001
001	011	101	100
INFORM	INFORM	INFORM	INFORM
000	000	000	000
000	000	000	001
010	100	110	000
CODE WORD	CODE WORD	CODE WORD	CODE WORD
0000000	0000000	0000000	0000000
0000000	0000000	0000000	0011101
0100111	1001110	1101001	0000000
1101001	1110100	0011101	1001110
0100111	1001110	1101001	0011101
1001110	0111010	1110100	1010111
1101001	1110100	0011101	1010011
ERROR PATTERN	ERROR PATTERN	ERROR PATTERN	ERROR PATTERN
1100000	1000010	0000000	0100000
0000110	0000000	0000011	0000101
0000000	0000101	0010100	0000100
0000011	0010000	0101011	0010100
0110000	0001000	0010000	0000000
0000100	0000000	0000100	1000000
0110100	1000000	0000110	0000000
RECEIVED WORD	RECEIVED WORD	RECEIVED WORD	RECEIVED WORD
1100000	1000010	0000000	0100000
0000110	0000000	0000011	0011000
0101111	1001011	1111101	0000100
1101010	1100100	0110110	1011010
0010111	1000110	1111001	0011101
1001010	0111010	1110000	0010011
1011101	0110100	0011011	1010011
OUTPUT	OUTPUT	OUTPUT	OUTPUT
000	000	000	000
000	000	000	001
010	100	110	000

마이크로 컴퓨터를 이용한 실험 구성도는 그림 6 과 같다.

실험 과정은 다음과 같다.

(1) 마이크로 컴퓨터의 memory X에 정보 심벌을 저장해 놓는다.

(2) 정보 심벌들을 I/O1을 통하여 인코더로 전송한 후, 인코딩된 값을 다시 I/O2를 통해서 마이크로 컴퓨터에 받아들여 memory Y에 저장한다.

(3) 프로그램에 의해 에러를 생성한 후 memory Y에 들어 있는 코드 단어와 더하여 I/O2를 통해서 디코더에 보낸다. 디코딩된 값은 다시 I/O2를 통해서 마이크로 컴퓨터에 받아 들여 memory Z에 저장한다.

인코딩하는데 49 클럭이 소요되었고 디코딩하는데 80 클럭이 소요되었다.

위의 실험 과정에 의한 결과는 표 4 와 같다.

Ⅳ. 에러 performance

채널을 BSC (binary symmetric channel) 이라 하고, 채널에서 발행하는 에러가 랜덤이고 서로 독립적으로 발생한다고 가정하였을 경우에 블럭 코드의 에러 확률은 식 (8) 과 같다.

$$P_b = \sum_{i=t+1}^{n} \binom{n}{i} P_e^i (1-P_e)^{n-i} \qquad (8)$$

단일 burst 에러 정정 코드의 에러 확률 P_b는

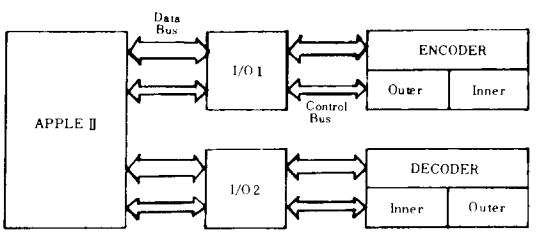

그림 6. 실험 구성도
Fig. 6. Block diagram of experiment.

표 4. 실험 결과
Table 4. Experimental result.

INFORM	CODE WORD	RECEIVED WORD	OUTPUT
0 0 7	0 0 7 2 7 5 2	0 0 7 2 6 5 0	0 0 7
0 1 6	0 1 6 5 7 2 4	0 1 6 5 7 2 4	0 1 6
0 2 5	0 2 5 7 7 0 5	0 2 4 7 7 7 5	0 2 5
0 3 4	0 3 4 0 7 7 3	0 3 4 0 7 7 3	0 3 4
0 4 3	0 4 3 3 7 4 7	0 6 3 7 7 4 4	0 4 3
0 5 2	0 5 2 4 7 3 1	4 5 2 4 7 3 1	0 5 2
0 6 1	0 6 1 6 7 1 0	0 6 1 6 5 1 3	0 6 1
0 7 0	0 7 0 1 7 6 6	0 7 3 1 7 4 6	0 7 0
0 7 7	0 7 7 3 0 3 7	0 7 7 3 0 3 7	0 7 7
1 0 6	1 0 6 7 7 1 6	4 0 7 4 0 1 6	4 0 7
1 1 5	1 1 5 6 5 2 6	1 1 5 2 5 7 6	1 1 5
1 2 4	1 2 4 2 7 4 1	1 2 4 2 6 5 1	1 2 4
1 3 3	1 3 3 4 1 4 6	1 3 3 4 1 4 6	1 3 3
1 4 2	1 4 2 6 7 0 3	1 4 2 6 7 0 3	1 4 2
1 5 1	1 5 1 7 5 3 3	1 3 1 7 5 0 3	1 5 1
1 6 0	1 6 0 3 7 5 4	1 6 0 3 6 5 4	1 6 0
1 6 7	1 6 7 1 0 0 6	1 6 7 1 0 0 7	1 6 7
1 7 6	1 7 6 6 0 7 0	1 7 6 3 0 7 0	1 7 6
2 0 5	2 0 5 3 7 6 1	7 0 3 3 1 6 1	7 5 3
2 1 4	2 1 4 4 7 1 7	2 1 2 4 7 3 7	2 1 4
2 2 3	2 2 3 1 3 0 1	2 2 3 4 3 0 1	2 2 3
2 3 2	2 3 2 6 3 7 7	2 3 2 6 3 5 7	2 3 2
2 4 1	2 4 1 2 7 7 4	7 4 7 2 7 7 5	7 4 7
2 5 0	2 5 0 5 7 0 2	2 4 0 6 7 5 2	2 5 0
2 5 7	2 5 7 7 0 5 0	2 5 7 7 4 6 0	2 5 7
2 6 6	2 6 6 3 2 3 7	2 6 6 3 2 3 7	2 6 6
2 7 5	2 7 5 2 0 0 7	2 7 5 2 0 0 4	2 7 5
3 0 4	3 0 4 6 7 2 5	3 0 4 7 7 2 5	3 0 4
3 1 3	3 1 3 0 1 2 2	2 1 3 0 1 0 2	3 1 3
3 2 2	3 2 2 4 3 4 5	5 2 2 4 3 3 5	3 2 2
3 3 1	3 3 1 5 1 7 5	3 3 1 5 1 7 5	3 3 1
3 4 0	3 4 0 7 7 3 0	3 4 2 7 4 1 0	2 4 2
3 4 7	3 4 7 5 0 6 2	3 4 7 5 0 6 2	3 4 7
3 5 6	3 5 6 2 0 1 4	0 7 6 2 0 1 4	3 5 6
3 6 5	3 6 5 0 0 3 5	3 6 5 0 0 3 2	3 6 5
3 7 4	3 7 4 7 0 4 3	3 7 4 7 3 6 3	3 7 4
4 0 3	4 0 3 0 7 3 4	4 0 3 0 7 5 4	4 0 3
4 1 2	4 1 2 7 7 4 2	4 1 2 7 3 4 3	4 1 2
4 2 1	4 2 1 5 7 6 3	4 2 1 2 7 6 7	4 2 1
4 3 0	4 3 0 2 7 1 5	4 3 0 2 7 1 5	4 3 0
4 3 7	4 3 7 0 0 4 7	4 3 7 0 0 4 7	4 3 7
4 4 6	4 4 6 2 6 0 2	4 4 6 2 6 0 2	4 4 6
4 5 5	4 5 5 3 4 3 2	4 5 5 3 4 3 2	4 5 5
4 6 4	4 6 4 7 6 5 5	4 6 4 7 6 5 5	4 6 4
4 7 3	4 7 3 1 0 5 2	6 7 6 1 0 5 2	4 7 3
5 0 2	5 0 2 5 7 7 0	5 0 2 5 7 5 0	5 0 2

$$P_{ub}=\sum_{i=2}^{n}\binom{n}{i}P_e^i(1-P_e)^{n-i}-\sum_{j=2}^{l}N_jP_e^j(1-P_e)^{n-j} \quad (9)$$

으로 나타나며, 여기에서 l은 정정할 수 있는 burst 에러 길이이고, N_j는 j=2, 3, …, l일 때 burst 에러의 수이다.

그리고 concatenated 코드의 에러 확률 P_B는 다음과 같이 구해진다. 내부 디코더에서의 심벌 에러 확률은

$$P_o=\sum_{i=t_1+1}^{n_1}\binom{n_1}{i}P_e^i(1-P_e)^{n_1-i} \quad (10)$$

과 같이 되고, 외부 디코더에서 심벌 에러를 정정하지 못할 확률은 식(11)과 같다.

$$P_b=\sum_{i=t_2+1}^{n_2}\binom{n_2}{i}P_o^i(1-P_o)^{n_2-i} \quad (11)$$

여기서 P_e는 채널 에러 확률, t는 블럭 코드의 에러 정정 능력, n_1은 내부 코드 길이, n_2는 외부 코드 길이, t_1은 내부 코드의 정정 능력, t_2는 외부 코드의 정정 능력을 나타낸다.

1차 코드와 2차 코드를 (7, 3) 순환 코드로써 이용한 (49, 9) 곱셈 코드와 (49, 9) concatenated 코드의 에러 확률은 그림 8과 같이 나타나고, 내부 코드를 (6, 3) 코드를 사용하고 (7, 3) Reed-Solomon 코드와 concateation한 (42, 9) concatenated 코드와 비슷한 코드 길이와 부호화율을 갖는 (63, 16) BCH 코드의 에러 확률은 그림 8와 같다.

그림 7과 그림 8에서 동일한 코드 길이와 정보량에서 concatenated 코드가 곱셈 코드보다 에러 확률이 적음을 알 수 있고, 이진 코드로서 에러 정정 능력이

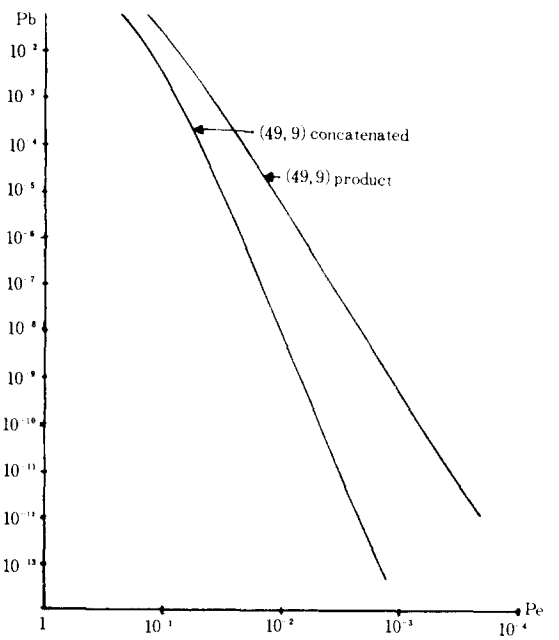

그림 7. (49, 9) Concatenated 코드와 (49, 9) 곱셈 코드의 에러 확률
Fig. 7. Error probability of (49, 9) concatenated code and (49, 9) product code.

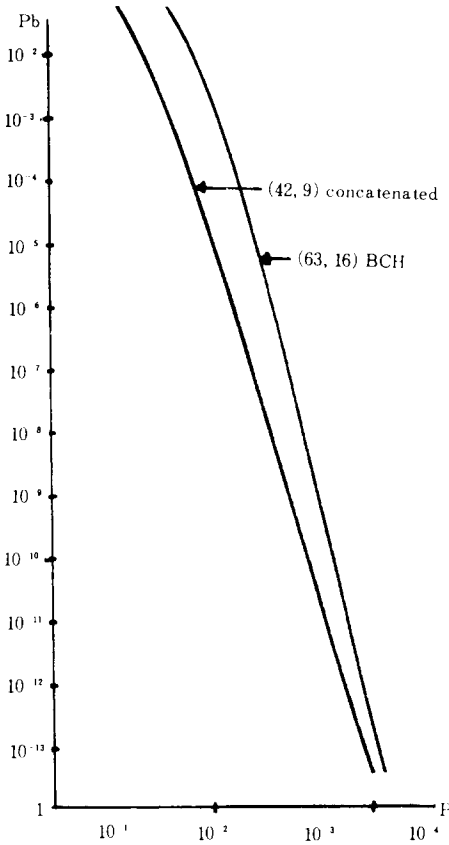

그림 8. (42, 9) Concatenated 코드와 (63, 16) BCH 코드의 에러 확률

Fig. 8. Error probability of (42, 9) concatenated code and (63, 16) BCH code.

가장 좋은 BCH 코드와 concatenated 코드의 에러 확률을 비교할 때 concatenated 코드의 에러 확률이 적음을 알 수 있다. 이것은 긴 코드를 설계할 경우, 두 코드를 cascade로 연결한 코드중에서 concatenated코드의 에러 확률이 이진 코드에서 가장 적음을 알 수 있다.

V. 結 論

긴 코드를 설계할 경우 단일 코드로 설계하면 회로 설계가 대단히 복잡해지므로, 짧은 코드를 이용하여 긴 코드를 구성하는 방법 중에서 이진 코드와 비이진 코드를 concatenation시키는 방법을 실현하였다.

(7, 3) burst 에러 정정 코드와 (7, 3) Reed-Solomon 코드를 이용하여 (49, 9) concatenated 코드를 구성하였고, parallel to serial 회로와 serial to parallel 회로를 이용하여 (49, 9) concatenated 코드의 인코더 및 디코더를 제작하였으며 클럭 주기를 달리함으로써 디코딩 시간을 줄였다.

컴퓨터 시뮬레이션 결과, 9개 이하의 랜덤 에러들을 정정할 수 있었으며 길이가 12이하인 burst 에러도 정정할 수 있었다.

(49, 9) concatenated 코딩 시스템에 대한 에러 performance를 구하였고, 동일한 코드 길이와 부호화율을 갖는 순환 곱셈 코드와 비슷한 코드 길이와 부호화율을 갖는 BCH 코드의 에러 확률들과 비교한 결과, concatenated 코딩 시스템의 에러 확률이 적음을 알 수 있었다.

이러한 concatenated 코딩 시스템은 랜덤 및 burst 에러가 복합적으로 발생하는 채널이나, 에러가 많이 발생하는 채널에 효율적이며, 긴 코드를 구성하는 경우에 코딩 시스템의 복잡성을 줄일 수 있으므로 효율적이다.

參 考 文 獻

[1] Lin, S., *An Introduction to Error Correcting Codes,* Prentice-Hall, 1970.

[2] G.D. Forney, Jr., *Concatenated Codes,* MIT Press, 1966.

[3] Burton, H.O., and E.J. Weldon, Jr., "Cyclic Product Codes," *IEEE Trans. on Information Theory,* IT-11, pp. 433-440, 1965.

[4] Elias, P., "Error Free Coding," *IRE Trans. on Information Theory,* PGIT-4, pp. 29-37, 1954.

[5] Lin-Nan, L., "Concatenated Coding Systems Employing a Unit-Memory Convolutional Code and a Byte-Oriented Decoding Algorithm," *IEEE Trans. on Communication Technology,* COM-25, pp. 1064-1074, 1977.

[6] Hirasawa, S., M. Kasahara, Y. Sugiyama, and T. Namekawa, "Modified Product Codes," *IEEE Trans. on Information Theory,* IT-30, pp. 299-306, 1984.

[7] Blahut, R.E., *Theory and Practice of Error Control Codes,* Addison-Wesley, 1983.

[8] Reed, I.S., G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc. Indust. Appl. Math.* vol. 8, pp. 300-304, 1960. *