論 文

순환 곱 코드의 간단한 두 단계 다수결 논리 디코더

正會員 鄭 然 湖 正會員 康 昌 彦

A Simplified Two-Step Majority-Logic Decoder for Cyclic Product Codes

Yon Ho JEONG *and Chang Eon KANG **, *Regular Members*

要 約 本 論文에서는(7, 4) 循環 코드와 (3, 1) 循環 코드의 곱의 디코더가, 같은 코드를 使用하는 보통의 다른 두 段階 多數決 論理 디코더에 비해서, 적은 수의 多數決 게이트들을 使用하도록 設計되었고, 多數決 게이트로서 ROM(read only memory)을 使用한 結果로 디코더는 간단한 構造로 製作되었다. 한 개의 受信語(혹은 21 bits)을 完全히 訂正시키는데 42개의 클럭 펄스가 經過하였다. 그래서 이 디코딩은 두 개의 디코더들과 二次元 語의 配列을 함께 使用한 從來의 디코딩에 비해서 디코딩 時間이 약 0.7배가 되었다.

ABSTRACT In this paper, A decoder for the product of the(7,4)cyclic code and the(3,1)cyclic code was designed with less majority gates than other ordinary two-step majority-logic decoder using the same codes, then it was constructed in simple structure as a result of the use of a ROM as a majority gate. It took 42 clock pulses to correct a received word(or 21 bits) entirely. And so the decoding time in this decoding was multiplied by a factor of about 0.7 relative to the decoding time in the previous decoding in which two decoders and two-dimensional word arrays were used together.

1. 序 論

1965년에 Burton과 Weldon[1]이 순환 곱 코드의 이론을 체계화시켰고, 1968년에 Abramson[2]은 순환 곱 코드의 디코딩을 두 개의 디코더의 종속 접속(cascade)으로 수행하였으며, 1970년에 Lin과 Weldon은[3] 순환 곱 코드의 다수결 논리 디코딩의 부분적인 이론을 마련하였다. 이밖에 순환 곱 코드의 다수결 논리 디코딩에 관한 여러 논문들이[4, 5, 6] 이 발표되었다. 순환 곱 코드는 다른 코드에 비해서 많은 대수 구조(algebraic structure)를 지니고, 짧은 성분 코드(행 코드와 열 코드)에 의해 표현되는 장점을 지니고 있다. 순환 곱 코드는 Mattson-Solomon 다항식에 의해서 명확하게 설명되고 있다[3]. 성분코드들 중에서 하나는 최소 거리 d_1을 가지는 한 단계 다수결 논리 순환 코드이고 다른 하나는 최소 거리 d_2를 가지는 L단계 다수결 논리 순환코드이면, 곱 코드는 최소 거리 d_1d_2를 가지는 L단계 다수결 논리 순환 코드가 된다[3, 4]. 다수결 논리 디코더는 크게 세 부분으로 나누어 신드롬 레지스터(syndrome register), 다수결 게이트, 바

*, ** 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 : 85-15 (接受 1985. 4. 4)

버 레지스터(buffer register) 및 정정용 게이트(exclusive-OR)이다. 다수결 논리 디코더는 한 단계형과 L단계형이 있으며 이 분류[7, 9]는 다수결 게이트의 수와 배열에 의해서 결정된다. 다수결 게이트의 수와 배열은 디코더의 복잡성에 크게 영향을 주고 있다. 본 논문에서는 (7, 4) 순환 코드와(3, 1) 순환 코드의 곱의 디코더를, 한 개의 두 단계 다수결 논리형이면서 다수결 게이트의 수의 절약형으로 설계하였다. 종래의 두 단계 다수결 논리 디코더 경우, 디코더의 다수결 게이트는 제 1 레벨에서 J개(J는 게이트의 입력 수), 제 2 레벨에서 한 개가 필요하다. 이와 같은 방법을 따르면, (7, 4) 순환 코드와(3, 1) 순환 코드의 곱의 디코더에서는 J값이 8이므로 필요한 다수결 게이트의 수는 전체적으로 9개이다. 이에 비해서 본 논문의 디코더는 전체적으로 3개의 다수결 게이트들을 사용하도록 설계되어 디코더가 간단해지고, 제작 비용도 감소되었다.

2. 순환 곱 코드

순환 곱 코드에서 코드 길이, 정보 길이, 최소거리는 성분코드(행코드와 열코드)의 대응하는 값들의 곱이 된다[3, 7, 9]. 즉 행코드 C_1의 코드 길이, 정보길이, 최소거리를 각각 n_1, k_1, d_1으로 표시하고, 열코드 C_2의 코드 길이, 정보길이, 최소거리를 각각 n_2, k_2, d_2로 표시하면 C_1과 C_2의 곱의 코드길이, 정보길이, 최소거리는 각각 n_1n_2, k_1k_2, d_1d_2가 된다. 이 후, 사용하는 기호들은 행코드 C_1에 관한 내용에는 첨자 1, 열코드 C_2에 관한 내용에는 첨자 2를 붙이고 순환 곱코드는 첨자없이 사용하여 기호를 구분하기로 한다. (7, 4) 순환코드와 (3, 1) 순환코드의 곱은(21, 4) 순환 곱 코드이며, 성분코드의 최소거리는 각각 3이므로 순환 곱 코드의 최소거리는 9이다. (7, 4) 행 코드의 생성 다항식 $g_1(x)$, 검사 다항식 $h_1(x)$, 또 (3, 1) 열 코드의 생성 다항식 $g_2(x)$, 검사 다항식 $h_2(x)$는 각각 식(1)~(4)이다.

$$g_1(x) = x^3 + x + 1 \tag{1}$$

$$h_1(x) = (x+1)(x^3+x^2+1) \tag{2}$$

$$g_2(x) = x^2 + x + 1 \tag{3}$$

$$h_2(x) = x + 1 \tag{4}$$

곱 코드가 순환적(cyclic)이려면 첫 째, 두 성분 코드는 모두 순환적이고, 둘째 두 성분 코드들의 코드길이는 서로 소(relative prime)의 관계 즉,

$$an_1 + bn_2 = 1 \ (a,\ b\text{는 상수}) \tag{5}$$

이 성립하여야 하며[3, 7, 9], 또 순환적인 성질을 유지하려면 코드 어(code word)를 채널로 방출할 때, 코드 어의 2차원 배열을 1차원 배열로 변환시키는 순서에 관한 일정한 규칙[1,2,7,8,9]이 있다. 표 1에서 코드 어(혹은 수신 어, 에러원소)의 2차원 배열을 보여주고 있고, 표 2에서 변환순서를 보여주고 있다. 표 1의 숫자들은 두 자리로서 첫 자리는 행의 번호이고 둘째 자리는 열의 번호를 의미한다. 표 2의 숫자들은 코드 다항식의 차수에 해당되며 일렬로 정돈한 것은 1차원 배열이 된다. 여기서 높은 숫자의 비트를 먼저 방출하도록 정했다. 코드 어의 2차원 배열에서 행 코드는 우측이 정보 비트, 좌측이 검사 비트이고, 열 코드는 하측이 정보 비트, 상측이 검사 비트이다. 이리하여 순환 곱 코드의 정보 비트들은 2차원 배열에서 23, 24, 25, 26이고, 1차원 배열에서 17, 11, 5, 20이다.

표 1 코드 어(혹은 수신어, 에러원소)의 2차원배열
A two-dimensional array of a code word(or received word, error element).

00	01	02	03	04	05	06
10	11	12	13	14	15	16
20	21	22	23	24	25	26

표 2 변환 순서
The order of conversion.

0	15	9	3	18	12	6
7	1	16	10	4	19	13
14	8	2	17	11	5	20

보통 코드 어는 채널의 교란(disturbance) 에 의한 에러 벡터가 더해져 변질되고 있다. 수신

어(received word)의 배열은 1차원이며, 다시 2차원 배열로 변환시키면 표 1을 따른다.

이번에는 순환 곱 코드의 생성 다항식, 생성 행렬, 검사행렬을 차례로 구하면 다음과 같다. 식 (5)에서 상수 a와 b는 $a=1$, $b=-2$ 이므로 생성 다항식 $g(x)$는[1, 3]

$$g(x)=LCM\{GCD[g_1(x^{bn_2}),\ x^{n_1n_2}-1],\ GCD[g_2(x^{an_1}),\ x^{n_1n_2}-1]\}$$
$$=(X^{14}+X^7+1)\ (X^3+X+1)=X^{17}+X^{15}+X^{14}+X^{10}+X^8+X^7+X^3+X+1 \quad (6)$$

이다. 한편, 생성 행렬 G와 검사 행렬 H를 구하면 각각 식 (7), (8)이 된다.

$$G=\begin{bmatrix} 1&1&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0\\ 0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0\\ 1&1&1&0&0&1&0&1&1&1&0&0&1&0&1&1&1&0&0&1&0\\ 1&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&1 \end{bmatrix} \quad (7)$$

$$H=\begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&1&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1&1 \end{bmatrix} \quad (8)$$

3. 순환 곱 코드의 디코더

먼저 에러 정정에 필요한 신드롬은 검사행렬을 사용하여 구하고, 다수결 게이트의 입력 연결에 필요한 검사 합(check sums)을 구하기로 한다[3, 4, 7, 9]. 에러의 벡터는

$$\bar{e}=(e_0,\ e_1,\ e_2,\ \cdots\cdots,\ e_{20}) \quad (9)$$

일때 $\bar{e}$에 대응한 신드롬은

$$\bar{S}=(S_0,\ S_1,\ S_2,\ \cdots\cdots,\ S_{16})=\bar{e}H^T \quad (10)$$

이다. 식(10)을 전개할 때 신드롬의 원소들은 에러의 1차원 배열의 원소이며, 에러의 2차원 배열의 원소로 변환하면 식(11)이 된다.

$$S_0=e_{00}+e_{23}+e_{15}+e_{26}$$
$$S_1=e_{11}+e_{23}+e_{04}+e_{15}$$
$$S_2=e_{22}+e_{04}+e_{15}+e_{26}$$
$$S_3=e_{03}+e_{23}$$
$$S_4=e_{14}+e_{04}$$
$$S_5=e_{25}+e_{15}$$
$$S_6=e_{06}+e_{26}$$
$$S_7=e_{10}+e_{23}+e_{15}+e_{26}$$

$$
\begin{aligned}
S_8 &= e_{21}+e_{23}+e_{04}+e_{15} \\
S_9 &= e_{02}+e_{04}+e_{15}+e_{26} \\
S_{10} &= e_{13}+e_{23} \\
S_{11} &= e_{24}+e_{04} \\
S_{12} &= e_{05}+e_{15} \\
S_{13} &= e_{16}+e_{26} \\
S_{14} &= e_{20}+e_{23}+e_{15}+e_{26} \\
S_{15} &= e_{01}+e_{23}+e_{04}+e_{15} \\
S_{16} &= e_{12}+e_{04}+e_{15}+e_{26}
\end{aligned} \tag{11}
$$

검사 합은 한 개 혹은 몇 개의 에러 원소에 대해서 orthogonal이 되도록 만든 식이며 또한 신드롬의 선형 조합으로 표현되는데, 다수결 게이트의 입력 연결에 사용된다. 디코더는 다수결 게이트의 수와 배열에 의해서 두 가지의 유형으로 분류할 수 있다. 그림 1의 첫 번째 유형의 디코더에서 다수결 게이트는 제 1 레벨의 8개와 제 2 레벨의 1개가 필요하며 전체적으로 9개가 필요하다. 첫 번째 유형의 디코더 설계에 필요한 검사 합은 식(12)의 에러 원소의 합에 대해서 orthogonal이 되도록 만든다.

$$
\begin{aligned}
E_1 &= e_{25}+e_{26} \\
E_2 &= e_{24}+e_{26} \\
E_3 &= e_{23}+e_{26} \\
E_4 &= e_{15}+e_{26} \\
E_5 &= e_{14}+e_{26} \\
E_6 &= e_{13}+e_{26} \\
E_7 &= e_{05}+e_{26} \\
E_8 &= e_{04}+e_{26}
\end{aligned} \tag{12}
$$

검사합 $A_{1K}, \cdots, A_{8K}(K=0, 1, \cdots, 8)$은 각각 식(13)~(20)이 된다.

$$
\begin{aligned}
A_{11} &= S_0+S_5 = e_{00}+e_{23}+(e_{25}+e_{26}) \\
A_{12} &= S_5+S_6 = e_{06}+e_{15}+(e_{25}+e_{26}) \\
A_{13} &= S_5+S_{16} = e_{12}+e_{04}+(e_{25}+e_{26}) \\
A_{14} &= S_2+S_4+S_5 = e_{22}+e_{14}+(e_{25}+e_{26}) \\
A_{15} &= S_5+S_9+S_{11} = e_{02}+e_{24}+(e_{25}+e_{26}) \\
A_{16} &= S_3+S_5+S_7 = e_{03}+e_{10}+(e_{25}+e_{26}) \\
A_{17} &= S_5+S_{10}+S_{14} = e_{13}+e_{20}+(e_{25}+e_{26}) \\
A_{18} &= S_5+S_{12}+S_{13} = e_{05}+e_{16}+(e_{25}+e_{26})
\end{aligned} \tag{13}
$$

$$
\begin{aligned}
A_{21} &= S_2+S_{11} = e_{22}+e_{15}+(e_{24}+e_{26}) \\
A_{22} &= S_6+S_{11} = e_{06}+e_{04}+(e_{24}+e_{26}) \\
A_{23} &= S_0+S_1+S_{11} = e_{00}+e_{11}+(e_{24}+e_{26}) \\
A_{24} &= S_7+S_8+S_{11} = e_{10}+e_{21}+(e_{24}+e_{26}) \\
A_{25} &= S_{11}+S_{14}+S_{15} = e_{20}+e_{01}+(e_{24}+e_{26}) \\
A_{26} &= S_5+S_9+S_{11} = e_{25}+e_{02}+(e_{24}+e_{26}) \\
A_{27} &= S_4+S_{11}+S_{13} = e_{14}+e_{16}+(e_{24}+e_{26}) \\
A_{28} &= S_{11}+S_{12}+S_{16} = e_{05}+e_{12}+(e_{24}+e_{26})
\end{aligned} \tag{14}
$$

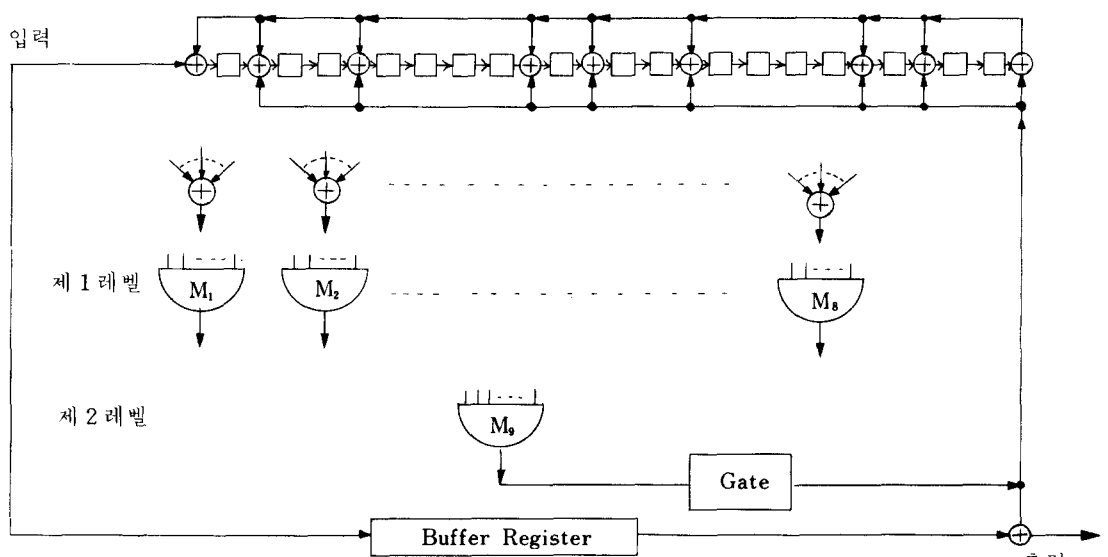


그림 1 (7, 4)순환코드와 (3, 1)순환 코드의 곱의 다수결 논리 디코더 : 제 1 유형
A majority-logic decoder for the product of the(7, 4)cyclic code and the(3,1)cyclic code; Type Ⅰ.

$$
\begin{aligned}
A_{31} &= S_0 = e_{00}+e_{15}+(e_{23}+e_{26})\\
A_{32} &= S_1+S_2 = e_{11}+e_{22}+(e_{23}+e_{26})\\
A_{33} &= S_8+S_9 = e_{21}+e_{02}+(e_{23}+e_{26})\\
A_{34} &= S_{15}+S_{16} = e_{01}+e_{12}+(e_{23}+e_{26})\\
A_{35} &= S_3+S_6 = e_{03}+e_{06}+(e_{23}+e_{26})\\
A_{36} &= S_5+S_7 = e_{25}+e_{10}+(e_{23}+e_{26})\\
A_{37} &= S_{10}+S_{13} = e_{13}+e_{16}+(e_{23}+e_{26})\\
A_{38} &= S_{12}+S_{14} = e_{05}+e_{20}+(e_{23}+e_{26})
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
A_{41} &= S_0 = e_{00}+e_{23}+(e_{15}+e_{26})\\
A_{42} &= S_2 = e_{22}+e_{04}+(e_{15}+e_{26})\\
A_{43} &= S_5+S_6 = e_{25}+e_{06}+(e_{15}+e_{26})\\
A_{44} &= S_{12}+S_{13} = e_{05}+e_{16}+(e_{15}+e_{26})\\
A_{45} &= S_3+S_7 = e_{03}+e_{10}+(e_{15}+e_{26})\\
A_{46} &= S_{10}+S_{14} = e_{13}+e_{20}+(e_{15}+e_{26})\\
A_{47} &= S_4+S_9 = e_{14}+e_{02}+(e_{15}+e_{26})\\
A_{48} &= S_{11}+S_{16} = e_{24}+e_{12}+(e_{15}+e_{26})
\end{aligned}
\tag{16}
$$

$$
\begin{aligned}
A_{51} &= S_2+S_4 = e_{22}+e_{15}+(e_{14}+e_{26})\\
A_{52} &= S_4+S_6 = e_{06}+e_{04}+(e_{14}+e_{26})\\
A_{53} &= S_0+S_1+S_4 = e_{00}+e_{11}+(e_{14}+e_{26})\\
A_{54} &= S_4+S_7+S_8 = e_{10}+e_{21}+(e_{14}+e_{26})\\
A_{55} &= S_4+S_{14}+S_{15} = e_{20}+e_{01}+(e_{14}+e_{26})\\
A_{56} &= S_4+S_{11}+S_{13} = e_{24}+e_{16}+(e_{14}+e_{26})\\
A_{57} &= S_4+S_5+S_9 = e_{25}+e_{02}+(e_{14}+e_{26})\\
A_{58} &= S_4+S_{12}+S_{16} = e_{05}+e_{12}+(e_{14}+e_{26})
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
A_{61} &= S_0+S_{10} = e_{00}+e_{15}+(e_{13}+e_{26})\\
A_{62} &= S_6+S_{10} = e_{06}+e_{23}+(e_{13}+e_{26})\\
A_{63} &= S_3+S_{10}+S_{13} = e_{03}+e_{16}+(e_{13}+e_{26})\\
A_{64} &= S_1+S_2+S_{10} = e_{11}+e_{22}+(e_{13}+e_{26})\\
A_{65} &= S_8+S_9+S_{10} = e_{21}+e_{02}+(e_{13}+e_{26})\\
A_{66} &= S_{10}+S_{15}+S_{16} = e_{01}+e_{12}+(e_{13}+e_{26})\\
A_{67} &= S_5+S_7+S_{10} = e_{25}+e_{10}+(e_{13}+e_{26})\\
A_{68} &= S_{10}+S_{12}+S_{14} = e_{05}+e_{20}+(e_{13}+e_{26})
\end{aligned}
\tag{18}
$$

$$
\begin{aligned}
A_{71} &= S_0+S_{12} = e_{00}+e_{23}+(e_{05}+e_{26})\\
A_{72} &= S_6+S_{12} = e_{06}+e_{15}+(e_{05}+e_{26})\\
A_{73} &= S_{12}+S_{16} = e_{12}+e_{04}+(e_{05}+e_{26})\\
A_{74} &= S_2+S_4+S_{12} = e_{22}+e_{14}+(e_{05}+e_{26})\\
A_{75} &= S_9+S_{11}+S_{12} = e_{02}+e_{24}+(e_{05}+e_{26})\\
A_{76} &= S_3+S_7+S_{12} = e_{03}+e_{10}+(e_{05}+e_{26})\\
A_{77} &= S_{10}+S_{12}+S_{14} = e_{13}+e_{20}+(e_{05}+e_{26})\\
A_{78} &= S_5+S_{12}+S_{13} = e_{25}+e_{16}+(e_{05}+e_{26})
\end{aligned}
\tag{19}
$$

$$
\begin{aligned}
A_{81} &= S_2 = e_{22}+e_{15}+(e_{04}+e_{26})\\
A_{82} &= S_0+S_1 = e_{00}+e_{11}+(e_{04}+e_{26})\\
A_{83} &= S_7+S_8 = e_{10}+e_{21}+(e_{04}+e_{26})\\
A_{84} &= S_{14}+S_{15} = e_{20}+e_{01}+(e_{04}+e_{26})\\
A_{85} &= S_4+S_6 = e_{14}+e_{06}+(e_{04}+e_{26})\\
A_{86} &= S_{11}+S_{13} = e_{24}+e_{16}+(e_{04}+e_{26})\\
A_{87} &= S_5+S_9 = e_{25}+e_{02}+(e_{04}+e_{26})\\
A_{88} &= S_{12}+S_{16} = e_{05}+e_{12}+(e_{04}+e_{26})
\end{aligned}
\tag{20}
$$

제 1 레벨의 다수결 게이트의 출력들은 E_1, E_2, …, E_8의 각각에 대한 평가치이며 이들 출력들이 제 2 레벨 다수결 게이트의 입력으로 연결되어 있다. 제 2 레벨의 다수결 게이트의 출력은 처음 수신된 비트의 에러 원소 e_{26}에 대한 평가치가 되어 에러의 유무를 판정할 수 있다.

한편 두 번째 유형의 디코더는 다수결 게이트의 절약형으로서 다수결 게이트는 제 1 레벨의 2개와 제 2 레벨의 한 개가 필요하며 전체적으로 3개가 필요하다. 따라서 첫 번째 유형의 디코더에 비해서 다수결 게이트의 수는 $\frac{1}{3}$에 해당된다. 두 번째 유형의 디코더 설계에 필요한 검사합 A'_{1K}, A'_{2K} $(K=0, 1, \cdots, 8)$은 각각 $e_{15}+e_{04}$, $e_{15}+e_{23}$ 에 대해서 **orthogonal** 이 되도록 만든 식이며 식(21), (22)가 된다.

$$
\begin{aligned}
A'_{11} &= S_1+S_{10} = e_{13}+e_{11}+(e_{15}+e_{04})\\
A'_{12} &= S_{13}+S_{16} = e_{16}+e_{12}+(e_{15}+e_{04})\\
A'_{13} &= S_3+S_{15} = e_{03}+e_{01}+(e_{15}+e_{04})\\
A'_{14} &= S_6+S_9 = e_{06}+e_{02}+(e_{15}+e_{04})\\
A'_{15} &= S_8 = e_{23}+e_{21}+(e_{15}+e_{04})\\
A'_{16} &= S_2 = e_{26}+e_{22}+(e_{15}+e_{04})\\
A'_{17} &= S_4+S_{12} = e_{05}+e_{14}+(e_{15}+e_{04})\\
A'_{18} &= S_5+S_{11} = e_{25}+e_{24}+(e_{15}+e_{04})
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
A'_{21} &= S_7+S_{13} = e_{16}+e_{10}+(e_{15}+e_{23})\\
A'_{22} &= S_1+S_4 = e_{14}+e_{11}+(e_{15}+e_{23})\\
A'_{23} &= S_0+S_6 = e_{06}+e_{00}+(e_{15}+e_{23})\\
A'_{24} &= S_{15} = e_{04}+e_{01}+(e_{15}+e_{23})\\
A'_{25} &= S_{14} = e_{26}+e_{20}+(e_{15}+e_{23})\\
A'_{26} &= S_8+S_{11} = e_{24}+e_{21}+(e_{15}+e_{23})\\
A'_{27} &= S_5+S_{10} = e_{25}+e_{13}+(e_{15}+e_{23})\\
A'_{28} &= S_3+S_{12} = e_{05}+e_{03}+(e_{15}+e_{23})
\end{aligned}
\tag{22}
$$

제 1 레벨의 다수결 게이트의 출력은 각각 $e_{15}+e_{04}$, $e_{15}+e_{23}$에 대한 평가치이다. 제 1 레벨의 다수결 게이트의 출력들은 신드롬과 결합하여 제 2 레벨의 다수결 게이트의 입력으로 연결시킨다. 먼저 $P_1=e_{15}+e_{04}$, $P_2=e_{15}+e_{23}$로 놓으면 제 2 레벨의 다수결 게이트의 입력에 관계되는 식은 식(23)이 된다.

$$\begin{aligned}
F_1 &= P_1+S_6+S_{12}=e_{06}+e_{05}+e_{04}+e_{26}\\
F_2 &= P_1+S_2=e_{22}+e_{26}\\
F_3 &= P_1+S_9=e_{02}+e_{26}\\
F_4 &= P_1+S_{16}=e_{12}+e_{26}\\
F_5 &= P_2+S_0=e_{00}+e_{26}\\
F_6 &= P_2+S_7=e_{10}+e_{26}\\
F_7 &= P_2+S_{14}=e_{20}+e_{26}\\
F_8 &= P_2+S_5+S_{13}=e_{25}+e_{23}+e_{16}+e_{26}
\end{aligned} \tag{23}$$

두 번째 유형의 디코더는 그림 2에서 보여주고 있다.

본 논문에서는 두 번째 유형의 디코더를 선택하여 제작하였는데 디코더 구성에 사용된 IC 는 표 3 과 같다.

표 3 디코더 구성에 사용된 IC들
The IC's used for the decoder.

디코더의 주요 부분	IC 번호
Mod-2게이트(e-OR)	74 LS 86
신드롬 레지스터	74 LS 273
버퍼 레지스터	74 LS 125
다수결 게이트(ROM)	2716
부정(인버터)	74 LS 04

4. 實驗 및 結果 考察

그림 3의 실험도와 같이 마이크로 컴퓨터(Apple Ⅱ)를 사용하여 디코더의 에러 정정 실험을 하였다. CPU, 기억장치(memory), 입출력 장치(I/O), 클럭 펄스는 마이크로 컴퓨터에 내장된 것을 사용하고, 보조기억장치로서 디스

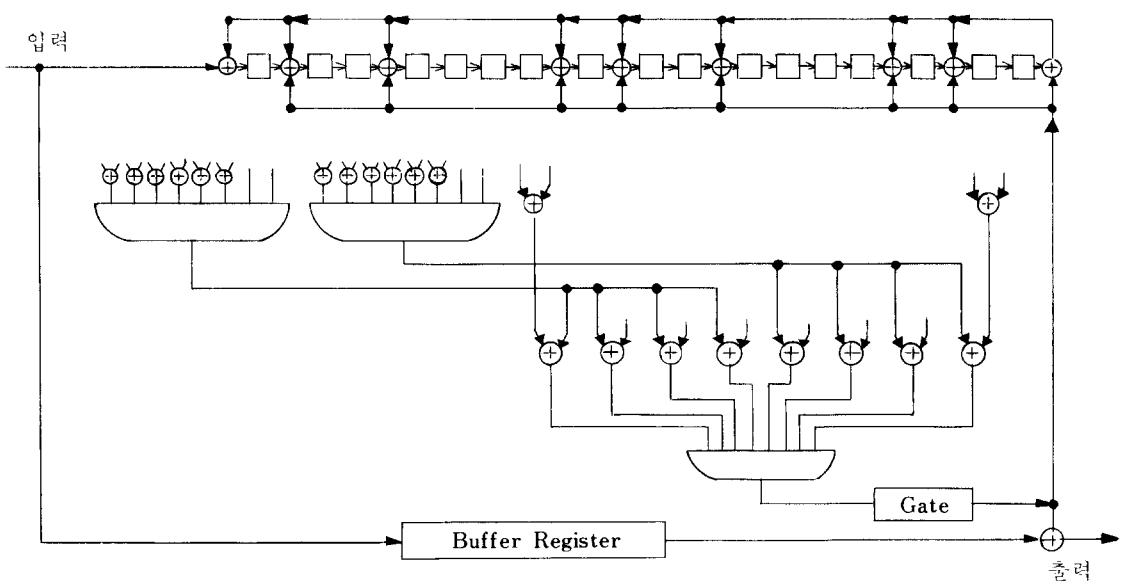


그림 2 (7, 1) 순환 코드와 (3, 1) 순환 코드의 곱의 다수결 논리 디코더 : 제 2 유형.
A majority-logic decoder for the product of the (7,4) cyclic code and the (3,1) cyclic code; Type Ⅱ.

크를 사용하였다. 디코더의 제어 프로그램은 기억장소(memory address) $ 800번지($는 16진수를 의미함)~$1000번지에 저장시키고, 수신 어는 $ 3000번지~$ 3999번지, 정정된 어는 $ 4000~$ 4999번지를 사용하였다. 또 외부의 디스크에 에러 패턴을 저장시켰다. 코드 어를 마이크로 컴퓨터의 키 보드를 사용하여 입력시키고 이와함께 에러 패턴을 입력시켜서 코드 어와 결합하여 수신 어를 만든다. 수신어는 $ 3000~$ 3999 번지에 저장시키며 I/O를 통해서 디코더로 보내고, 디코더에 의해 정정된 어는 I/O 를 통해서 $ 4000~$ 4999번지에 저장시켰다. 클럭 펄스 디코더의 동작을 조절하는 역할을 하며, 1 MHz에서 사용되었다.

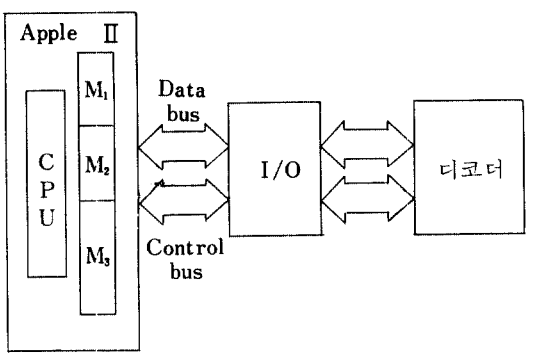


그림 3 실험도.
A block diagram of the experimental apparatus.

디코더의 동작을 살펴보면 다음과 같다. 처음 21개의 클럭 펄스가 경과하는 동안 다수결 게이트의 출력에 연결된 게이트는 차단상태이며 처음 수신된 비트의 신드롬이 신드롬 레지스터에서 만들어지고 동시에 한 개의 수신 어(21비트)가 바퍼 레지스터에 저장된다. 이 후, 두 번째 21개의 클럭 펄스가 경과하는 동안 디코더의 입력이 차단되고 다수결 게이트의 출력에 연결된 게이트는 열려져 있어서, 수신 어의 에러 정정이 수행된다. 한 개의 수신 어를 정정시키는데 필요한 클럭 펄스는 42개이며, 클럭 펄스의 주파수 1 MHz 에서 디코딩 시간은 42 μsec 이다. 한편 두 개의 디코더들과 2 차원어의 배열을 함께 사용한 종래의 디코딩[2]에서는 한 개의 수신어를 정정시키는데 62개의 클럭 펄스가 경과하며, 동일한 클럭 펄스의 주파수 1 MHz에서 디코딩 시간은 62 μsec 이다. 따라서 본 논문의 디코딩은 동일한 코드를 사용한 종래의 디코딩에 비해서 약 0.7 배의 디코딩 시간이 걸린다. 또 디코더는 4 비트의 산발 에러와 동시에 7 비트의 연집 에러의 정정 능력을 가지고 있음이 실험 결과 확인되었다.

5. 結 論

본 논문에서는 (7, 4) 순환 코드와 (3, 1) 순환 코드의 곱의 디코더를 다수결 게이트의 절약형으로 설계하였고, 다수결 게이트로서 기억소자 ROM을 사용하여 디코더를 간단한 구조로 제작하였다. 한 개의 수신어를 정정시키는데 42개의 클럭 펄스가 경과하여, 동일한 코드를 사용한 종래의 디코딩에 비해서 디코딩 시간은 약 0.7배가 되었다. 이 디코더는 4 비트의 산발 에러와 7 비트의 연집 에러의 정정 능력을 가지고 있음이 실험을 통해서 확인되었다.

參 考 文 獻

(1) H.O. Burton, E.J. Weldon Jr., "Cyclic product codes," IEEE Trans, on Inform Theory vol. IT-11, pp. 433-439, 1965.

(2) N. Abramson, "Cascade decoding of cyclic product codes," IEEE Trans on Comm. Tech. vol. COM-16, pp. 398-402, 1968.

(3) S. Lin, E.J. Weldon Jr., "Further results on cyclic product codes," IEEE Trans on Inform. Theory vol. IT-16, pp. 452-459, 1970.

(4) N.Q. Duc, "On the Lin-weldon majority-logic decoding algorithm for product codes," IEEE Trans. on Inform. Theory vol. IT-19, pp. 581-583, 1973.

(5) N.Q. Duc, L.V. Sattebol, "Further results on majority-logic decoding of product codes," IEEE Trans. on Inform. Theory vol. IT-18, pp. 308-310, 1972.

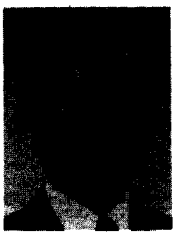
(6) Y.L. Lee, M.C. Cheng, "Cyclic mapping of product codes," IEEE Trans. on Inform. Theory vol. IT-21, pp. 233-235, 1975.

(7) S. Lin, "An Introduction to error-correcting codes," Prentice-Hall, 1970.

(8) W. W. Peterson, E.J. Weldon Jr., "Error-correcting codes," 2nd Edition, The M.I.T. Press, 1972.

(9) S. Lin, "Error control coding," Prentice-Hall, 1983.

鄭 然 湖 (Yon Ho JEONG) 正會員
1953年 2 月 3 日生
1971年3月～1975年2月：延世大學校工科大學電子工學科 (工學士)
1978年 8 月～1980年10月：延世大學校大學院電子工學科 (工學碩士)
1981年 3 月～現在：延世大學校 大學院 電子工學科 博士課程



康 昌 彦 (Chang Eon KANG) 正會員
1938年 8 月26日生
1960年：延世大學校電氣工學科 (工學士)
1965年：延世大學校大學院電氣工學科 (工學碩士)
1969年：美國미시간주립대학교大學院電氣工學科 (工學碩士)
1973年：美國미시간주립대학교大學院電氣工學科 (工學博士)
1967年～1973年：美國미시간주립대학교工業研究所先任研空員
1973年～1981年：美國노던일리노이대학교電氣工學科助教授, 副教授
1982年～現在：延世大學校電子工學科 教授, 本學會研究調査委員會 委員長