논문 2008-45SD-3-4

# Shannon Capacity에 근접하는 고효율의 6-ary Runlength-Limited Code

## ( Very Efficient 6-ary Runlength-Limited Code Approaching Shannon Capacity for Optical Storage Channels )

지 윤 규*

( Yoon Kyoo Jhee )

요 약

고효율의 $d=3$인 6-ary runlength-limited code를 연구하였다. Rate가 6/7일 경우 98.87%의 효율을 얻을 수 있었고 rate가 13/15일 경우 Shannon capacity에 근접하는 무려 99.95%의 효율을 얻을 수 있었다. 또한 6-ary RLL code를 효과적으로 검출하기 위한 partial response mode에 관하여 고찰하였다.

Abstract

Very efficient 6-ary runlength-limited codes for a six-level optical recording channel are presented when $d=3$. The 6-ary(3, 15) code of rate 6/7 is given achieving coding efficiency of 98.87%. The efficiency of rate 13/15, (3, 20) code is 99.95%, which approaches the Shannon capacity. To increase the accuracy of reading 6-ary signal, partial response modes are also investigated.

**Keywords :** code efficiency, Shannon capacity, optical recording, runlength-limited code

## I. 서 론

광 기록 장치는 디스크의 표면에 "pits" 또는 "lands"로 나타내는 이진 상태를 주로 사용하여 왔다. 이 방법은 미디어의 포화상태를 이용한 것이다. 이러한 이진 기록 대신에 multilevel(ML) 기록 방법을 사용하면 coding density를 높여 정보의 전달 속도를 현저히 증가시킬 수 있다. 예를 들자면 eight recording level을 사용하면 3 bit/stored-symbol이 되어 포화상태를 이용한 1 bit보다 정보전달 능력이 향상된다. 이 장점을 살려서 6-ary (2, 4) code가 Optex Communications Corp.에서 개발되었다[1].

* 정회원, 이화여자대학교 전자정보통신학전공
(Dept. of Information Electronics Engineering, Ewha Womans University)
접수일자: 2007년9월17일, 수정완료일: 2008년3월15일

$(d, k)$ runlength-limited(RLL) encoder로부터 생성된 연속된 M-ary 신호는 0이 아닌 symbol 사이에 $d$개 이상 그리고 $k$개 이하의 0들이 있어야 한다. $d$개의 제한 조건은 intersymbol interference(ISI)를 방지하기 위한 것이고 $k$개의 제한 조건은 timing signal을 추출해내기 위하여 필요한 것이다. Timing recovery 회로의 신호처리 기술이 발전함에 따라 $k$값의 제한이 많이 완화되었다.

RLL code를 구성하기 위하여 일반적으로 state splitting/merging 방법을 많이 사용한다. 이는 $m/n = R \le C$를 만족하는 양의 정수 $m$과 $n$가 존재하면 state splitting /merging 방법으로 sliding block decoder를 갖는 finite state encoder를 항상 구성할 수 있다는 정리에 기인한다[2]. $R$값을 constraint capacity $C$의 값에 근접하게 설계하면 code efficiency $e = R/C$를 높일 수 있다.

그러나 state splitting/merging 방법에 사용되는 approximate eigenvector는 encoder size의 upper bound를 느슨하게 설정한다. 이 미흡함을 보완하기 위하여 specific encoder structure에 적용되는 부등식을 사용할 수 있다. 이를 이용하여 binary channel의 경우 $d=1$ 과 2일 때 설계하였고[3] 3-ary signal의 경우에는 $d=2$ 일 때 구성하였다[4]. 이 방법은 $k$의 제한이 유동적인 경우에 매우 효과적이다.

본 논문에서는 위의 방법을 확장하여 $d=3$ 일 경우 매우 효율적인 6-ary RLL code를 구성하였다. 이는 state splitting/merging 방법을 이용한 6-ary(3, 8)의 결과[1]와 비교할 때 code efficiency를 97%에서 code rate가 6/7일 경우 98.87%로 높일 수 있었고 code rate가 13/15일 경우는 무려 99.95%까지 높여 Shannon capacity에 근접하는 6-ary RLL code를 설계할 수 있었다. 또한 6-ary RLL code를 효과적으로 검출하기 위한 partial response mode에 관하여 고찰하였다. II WKD에서는 $d=3$ 일 경우 6-ary RLL code를 설계하는 방법을 설명하고 IIIWKD에서 결론을 맺는다.

## II. 효율을 증가시키는 6-ary $d=3$ RLL code의 설계

ML (M-ary) recording channel은 disk에 "marks"의 폭(width)과 강도(intensity)로 나타낸다. 강도는 M개의 level로 나누어 사용하고 폭은 최소 폭($T_{\min}$)과 최대 폭($T_{\max}$) 사이의 discrete한 값으로 취한다. 여기서 $T_{\min}=(d+1)T_s$이고 $k>d$인 조건에서 $T_{\max}=(k+1)T_s$를 나타내며 $1/T_s$는 signaling rate를 의미한다.

Shannon capacity (constraint capacity) $C=\log_2\lambda$로 정의되며 $\lambda$는 다음 특성방정식의 가장 큰 실수 근이다.

$$z^{k+2}-z^{k+1}-(M-1)z^{k-d+1}+M-1=0 \qquad (1)$$

$M=6$이고 $d=3$인 조건에서 효율적인 code를 작성하기 위하여 $R=m/n$ 값의 변화에 따른 효율을 계산하여 <표 1>에 나타내었다. $k$가 충분히 큰 값이라고 가정하고 이 조건에서 구한 constraint capacity $C(3,\infty)=\log_2^{\lambda}=\log_2^{1.8240}=0.8671$ 이다.

Density ratio $D$는 다음 식으로 정의 된다.

$$D=\frac{(1+d)R}{T_{\min}} \ (\text{bits}/T_{\min}) \qquad (2)$$

본 논문의 경우 $d=3$이고 효율을 높이기 위하여 설계한 $R=m/n=6/7$과 $R=13/15$인 경우 각각 $D=3.4286$ bits/$T_{\min}$과 $D=3.4667$ bits/$T_{\min}$이다. 도달할 수 있는 가장 큰 density $D_C=\frac{(1+d)C}{T_{\min}}\geq D$로 표시된다. Code efficiency $e=R/C=D/D_C$로 정의되고 code가 이룰 수 있는 가장 큰 density에 근접하는 척도를 나타낸다[1].

<표 1>의 결과를 참조하여 $n$값이 비교적 작고 효율이 높은 $R=6/7$과 $R=13/15$의 경우에 대하여 각각 설계하도록 한다. 우선 $R=6/7$의 경우 $k$의 조건은 일단 무시하고 $d=3$의 제한조건을 만족시키는 finite-state encoder를 설계한다. Codeword는 $d=3$의 조건을 만족하며 길이가 $n=7$인 6-ary symbol로 구성한다. 이 codeword 들은 또한 $E_{0000000}$, $E_{000X00}$, $E_{0000X0}$, $E_{00000X}$, $E_{00X000}$, $E_{00XX00}$, $E_{00X0X0}$, $E_{00X00X}$, $E_{0X0000}$, $E_{0X0X00}$, $E_{0X00X0}$, $E_{0X000X}$, $E_{X00000}$, $E_{X00X00}$, $E_{X000X0}$, $E_{X0000X}$의 16개 codeword subset으로 나누는데 이는 $d=3$의 조건을 만족시키는 code를 구하기 위하여 구성한 subset이다. 여기서 $X$는 1, 2, 3, 4 또는 5 중 하나의 값을 나타낸다. 각 codeword subset의 아래첨자 6개의 symbol은 codeword 앞의 3개와 뒤의 3개를 각각 나타낸다. Encoder는 $r$개의 state를 지니고 있다고 가정하고 네 가지 형태의 state subset으로 나눈다. 각 state subset에 속해있는 state의 수를 각각 $r_1, r_2, r_3$와 $r_4(=r-r_1-r_2-r_3)$로 정의한다. 16 개

표 1. 각 R값에 따른 효율성 비교

Table 1. Code efficiency for the various R.

| m | n | R=m/n | $\frac{C(3,\infty)-R}{C(3,\infty)}\times 100$ |
|---|---|---|---|
| 5 | 6 | 0.8333 | 3.898 % |
| 6 | 7 | 0.8571 | 1.153 % |
| 11 | 13 | 0.8462 | 2.410 % |
| 13 | 15 | 0.8667 | 0.046 % |
| 16 | 19 | 0.8421 | 2.883 % |
| 17 | 20 | 0.8500 | 1.972 % |
| 19 | 22 | 0.8636 | 0.473 % |
| 23 | 27 | 0.8519 | 1.750 % |
| 25 | 29 | 0.8621 | 0.577 % |

의 codeword subset에 속해있는 각 codeword들은 네 가지 형태의 state subset으로 구분된다. 네 가지 형태의 state subset들은 다음과 같다. 첫 번째 형태의 state subset에 속해있는 codeword들은 "000"으로 시작한다. 두 번째 형태의 state subset에 있는 codeword는 "000" 또는 "00X"로 시작한다. 세 번째 형태의 state subset에 속해있는 codewod들은 "000" 또는 "00X" 또는 "0X0"로 시작한다. 또한 네 번째 형태의 state subset에 속해있는 codeword들은 "000" 또는 "00X" 또는 "0X0" 또는 "X00"로 시작한다. 이를 그림 1의 화살표 우측에 나타내었다.

State-transition은 $d=3$의 조건을 만족하기 위하여 다음의 rule을 따른다. 그림 1의 좌측에서 보여주듯이 $E_{000000}$, $E_{00X000}$, $E_{0X0000}$, $E_{X00000}$와 같이 "000"으로 끝나는 codeword들은 $r$ 개의 모든 encoder next state로 연결될 수 있다. "X00"($E_{000X00}$, $E_{00XX00}$, $E_{0X0X00}$, $E_{X00X00}$)로 끝나는 codeword는 네 번째 형태의 next state(NS)에 이어질 수 없다.

"0X0"($E_{0000X0}$, $E_{00X0X0}$, $E_{0X00X0}$, $E_{X000X0}$)로 끝나는 codeword는 세 번째와 네 번째 형태의 NS에 이어질 수 없다. 같은 방법으로 "00X"($E_{00000X}$, $E_{00X00X}$, $E_{0X000X}$, $E_{X0000X}$)로 끝나는 codeword는 오직 첫 번째 형태의 NS로만 이어진다.

NS가 다르면 동일한 codeword에 다른 information word를 할당할 수 있다. 즉, $E_{000000}$, $E_{00X000}$, $E_{0X0000}$, $E_{X00000}$와 같이 "000"으로 끝나는 codeword

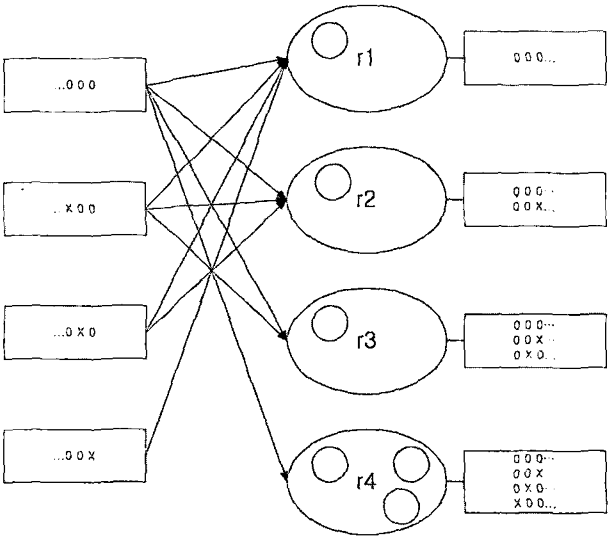


그림 1. $n=7$인 경우 encoder의 transition 방법
Fig. 1. Encoder transition rule($n=7$).

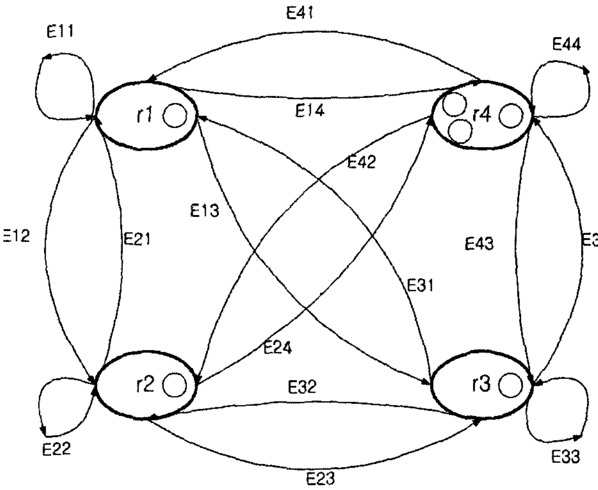


그림 2. $n=7$인 경우 state간의 transition diagram
Fig. 2 M-ary state transition diagram($n=7$).

는 모든 $r$개의 NS와 연결될 수 있으므로 서로 다른 information word를 $r=r_1+r_2+r_3+r_4$ 배만큼 할당할 수 있다. 또한 $E_{000X00}$, $E_{00XX00}$, $E_{0X0X00}$, $E_{X00X00}$와 같이 "X00"으로 끝나는 codeword는 네 번째 형태의 NS를 제외한 모든 NS와 연결될 수 있으므로 서로 다른 information word를 $r_1+r_2+r_3$배만큼 할당 할 수 있다. $E_{0000X0}$, $E_{00X0X0}$, $E_{0X00X0}$, $E_{X000X0}$와 같이 "0X0"으로 끝나는 codeword의 경우는 첫 번째와 두 번째 형태의 NS와 연결될 수 있으므로 $r_1+r_2$배만큼 할당 할 수 있다. 같은 방법으로 $E_{00000X}$, $E_{00X00X}$, $E_{0X000X}$, $E_{X0000X}$와 같이 "00X"으로 끝나는 codeword의 경우는 첫 번째 형태의 NS로만 연결될 수 있으므로 $r_1$배만큼 할당 할 수 있다. 위의 관계를 state transition diagram으로 나타내면 그림 2가 된다.

그림 2에서

$E_{11}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00000X}$;

$E_{12}: E_{000000}, E_{000X00}, E_{0000X0}$;

$E_{13}: E_{000000}, E_{000X00}$; $E_{14}: E_{000000}$;

$E_{21}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00000X}, E_{00X000}$;

$E_{22}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00X000}$;

$E_{23}: E_{000000}, E_{000X00}, E_{00X000}$; $E_{24}: E_{000000}, E_{00X000}$;

$E_{31}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00000X}, E_{0X0000}, E_{0X000X}, E_{00X000}$;

$E_{32}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00X000}, E_{0X0000}$;

$E_{33}: E_{000000}, E_{000X00}, E_{00X000}, E_{0X0000}$;

$E_{34}: E_{000000}, E_{00X000}, E_{0X0000}$;

$E_{41}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00000X}, E_{00X000}, E_{0X0000}, E_{0X000X}, E_{X00000}, E_{X000X0}, E_{X0000X}$;

$E_{42}: E_{000000}, E_{000X00}, E_{0000X0}, E_{00X000}, E_{0X0000}, E_{X00000}, E_{X000X0}$;

$E_{43}: E_{000000}, E_{000X00}, E_{00X000}, E_{0X0000}, E_{X00000}$;

$E_{44}: E_{000000}, E_{00X000}, E_{0X0000}, E_{X00000}$ 를 나타낸다.

위에서 설명한 encoder model을 기반으로 $R = m/n = 6/7$ 인 6-ary RLL code를 설계하기 위하여 다음 식을 정의한다. 아래 식에서 $|E_{WXYZ}|$는 $E_{WXYZ}$의 크기 즉 codeword subset에 속해있는 codeword 수를 의미한다.

$$A_1 = r|E_{000000}| + (r_1 + r_2 + r_3)|E_{000X00}| + (r_1 + r_2)|E_{0000X0}| + r_1|E_{00000X}| \tag{3}$$

$$A_2 = r|E_{00X000}| + (r_1 + r_2 + r_3)|E_{00XX00}| + (r_1 + r_2)|E_{00X0X0}| + r_1|E_{00X00X}| \tag{4}$$

$$A_3 = r|E_{0X0000}| + (r_1 + r_2 + r_3)|E_{0X0X00}| + (r_1 + r_2)|E_{0X00X0}| + r_1|E_{0X000X}| \tag{5}$$

$$A_4 = r|E_{X00000}| + (r_1 + r_2 + r_3)|E_{X00X00}| + (r_1 + r_2)|E_{X000X0}| + r_1|E_{X0000X}| \tag{6}$$

$d = 3$ 인 제한조건에서 $n = 7$ 인 경우에 codeword subset의 크기를 구하면 다음과 같다.

$$|E_{000000}| = 6, \tag{7}$$

$$|E_{000X00}| = |E_{0000X0}| = |E_{00000X}| = |E_{00X000}| = |E_{0X0000}| = |E_{X00000}| = 5, \tag{8}$$

$$|E_{00X00X}| = |E_{0X00X0}| = |E_{0X000X}| = |E_{X00X00}| = |E_{X000X0}| = |E_{X0000X}| = 25. \tag{9}$$

$$|E_{00XX00}| = |E_{00X0X0}| = |E_{0X0X00}| = 0. \tag{10}$$

$m = 6$ 인 경우에 encoder model은 다음 조건을 만족하여야 한다.

$$A_1 \ge r_1 2^m = 64r_1, \tag{11}$$

$$A_1 + A_2 \ge 64(r_1 + r_2), \tag{12}$$

$$A_1 + A_2 + A_3 \ge 64(r_1 + r_2 + r_3), \tag{13}$$

$$A_1 + A_2 + A_3 + A_4 \ge 64(r_1 + r_2 + r_3 + r_4). \tag{14}$$

위에서 구한 값들을 대입하여 정리하면 다음식이 된다.

$$-43r_1 + 16r_2 + 11r_3 + 6r_4 \ge 0 \tag{11'}$$

$$-13r_1 - 43r_2 + 16r_3 + 11r_4 \ge 0 \tag{12'}$$

$$42r_1 - 13r_2 - 43r_3 + 16r_4 \ge 0 \tag{13'}$$

$$122r_1 + 42r_2 - 13r_3 - 43r_4 \ge 0 \tag{14'}$$

$(11)' - (14)'$를 만족하면서 $r$의 값을 최소화하는 $r_1, r_2, r_3, r_4$과 $r = r_1 + r_2 + r_3 + r_4$를 구하면 다음과 같다.

$$r_1 = r_2 = 2,\ r_3 = 3,\ r_4 = 6,\ r = 13 \tag{15}$$

Code 구성을 위한 다음 과정은 위에서 구한 13개의 state에 codeword를 할당하는 것이다. Trial and error 방법으로 codeword를 할당하기 때문에 다양한 결과를 얻을 수 있고, 한 예를 <표 2>에 나타내었다. <표 2>의 $a \times b$에서 $a$는 codeword subset에 속해있는 codeword의 수를 $b$는 이어서 연결될 수 있는 NS의 수를 나타낸다. <표 2>에서 보여주는 바와 같이 크기가 $5 \times 7$인 $E_{000X00}$은 state 2에 22 개의 codeword를 할당하고 state 3에 13 개의 codeword를 할당한다. 즉, 각 행의 합이 각 codeword subset의 크기와 같도록 할당한다. <표 2>의 state 열에는 실제로 사용한 codeword 개수만을 나타냈기 때문에 각 행의 합이 codeword subset 보다 적은 경우도 있다.

$E_{000000}, E_{00X000}, E_{0X0000}$와 $E_{X00000}$ 같이 "000"으로 끝나는 각 codeword에는 서로 다른 information word를 이어서 연결할 수 있는 NS의 수 만큼인 13번 할당할 수 있고 $E_{000X00}$와 같이 "X00"로 끝나는 각 codeword에는 서로 다른 information word를 7번 할당할 수 있다. 마찬가지로 $E_{0000X0}$와 $E_{X000X0}$같이 "0X0"로 끝나는 codeword에는 서로 다른 information word를 4번 할당할 수 있고 $E_{00000X}, E_{0X000X}$와 $E_{X0000X}$같이 "00X"로 끝나는 codeword에는 information word를 2번만 할당할 수 있다. State 2에 할당된 총 codeword 수는 $12 + 22 + 20 + 10 = 64 = 2^6$이 된다. 나머지 state에 할당되는 codeword 수도 64개 이상이 되도록 구성하여야 한다.

각 state에 필요한 64 codeword 보다 더 많은 여분의 code들이 존재하는 경우 codeword "0000000"의 앞과 뒤에 오는 codeword 중에서 많은 수의 "0"이 연속적으

표 2. 각 codeword subset과 state의 분포($R=6/7$)
Table 2. Distribution of the various codeword subsets and states($R=6/7$).

| | $1(\tau_1)$ | $2(\tau_1)$ | $3(\tau_2)$ | $4(\tau_2)$ | $5(\tau_3)$ | $6(\tau_3)$ | $7(\tau_3)$ | $8(\tau_4)$ | $9(\tau_4)$ | $10(\tau_4)$ | $11(\tau_4)$ | $12(\tau_4)$ | $13(\tau_4)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $E_{000000}(6\times13)$ | 64 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{000X00}(5\times7)$ | 0 | 22 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{0000X0}(5\times4)$ | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{00000X}(5\times2)$ | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{00X000}(5\times13)$ | | | 51 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{00X00X}(25\times2)$ | | | 0 | 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{0X0000}(5\times13)$ | | | | | 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{0X00X0}(25\times4)$ | | | | | 4 | 64 | 32 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{0X000X}(25\times2)$ | | | | | 0 | 0 | 32 | 18 | 0 | 0 | 0 | 0 | 0 |
| $E_{X00000}(5\times13)$ | | | | | | | | 46 | 14 | 0 | 0 | 0 | 0 |
| $E_{X00X00}(25\times7)$ | | | | | | | | 0 | 50 | 64 | 61 | 0 | 0 |
| $E_{X000X0}(25\times4)$ | | | | | | | | 0 | 0 | 0 | 3 | 64 | 33 |
| $E_{X0000X}(25\times2)$ | | | | | | | | 0 | 0 | 0 | 0 | 0 | 31(19) |

로 이어지는 여분의 codeword를 제거하여 $k$값을 줄인다. 예를 들면 codeword "0000000" 다음에 codeword "0000000"나 "000000X"가 이어지는 것을 방지하도록 구성하여 많은 수의 "0"이 뒤에 나타나지 않도록 한다. 이는 <표 2>의 state 1열에서 output "0000000"일 때, NS 1과 2를 사용하지 않는 것이 된다. 또한 $E_{X00000}$과 $E_{0X0000}$가 "0000000"으로 이어지는 것을 방지하기 위하여 NS가 1인 여분의 10가지 codeword를 사용하지 않았다. 그 밖에 사용하지 않은 19개의 codeword를 괄호 안에 나타내었으며 필요한 경우에는 이를 이용하여 $X$(1, 2, 3, 4 or 5)를 포함하는 codeword를 구성할 수 있다. 이런 결과로 state가 13개이며 $k=15$인 13-state 6-ary(3, 15)를 구성할 수 있다. <표 2>에 근거하여 구성한 encoding table은 양이 방대하여 생략한다. 효율이 높은 $R=6/7$인 code의 경우 $d=3$이고 $k=15$이므로 $C=0.8669$이며 $D_C=3.4676/T_{\min}$이고 code efficiency $e=R/C=98.87\%$가 된다.

다음은 효율이 가장 높은 $R=m/n=13/15$인 6-ary RLL code를 설계한다. $d=3$인 제한조건에서 $n=15$인 경우에 codeword subset의 크기를 구하면 다음과 같다.

$$|E_{000000}|=546,\ |E_{000X00}|=|E_{00X000}|=530,$$
$$|E_{0000X0}|=|E_{0X0000}|=930,$$
$$|E_{00000X}|=|E_{X00000}|=1455,\ |E_{00XX00}|=400,$$
$$|E_{00X0X0}|=|E_{0X0X00}|=525,$$
$$|E_{00X00X}|=|E_{X00X00}|=|E_{0X00X0}|=1275.$$
$$|E_{0X000X}|=|E_{X000X0}|=2650.\ |E_{X0000X}|=4650,$$

$m=13$인 경우에 encoder model은 다음 조건을 만족하여야 한다.

$$A_1\ge r_1 2^{13}=8192r_1,$$
$$A_1+A_2\ge 8192(r_1+r_2),$$
$$A_1+A_2+A_3\ge 8192(r_1+r_2+r_3),$$
$$A_1+A_2+A_3+A_4\ge 8192(r_1+r_2+r_3+r_4).$$

위에서 구한 값들을 대입하여 정리하면 다음식이 된다.

$$\begin{aligned}-4731r_1+2006r_2+1076r_3+\ 546r_4&\ge 0\\ -2001r_1-4731r_2+2006r_3+1076r_4&\ge 0\\ 3379r_1-2001r_2-4731r_3+2006r_4&\ge 0\\ 1340r_1+3379r_2-2001r_3-4731r_4&\ge 0\end{aligned}$$

윗 식들을 만족하면서 $r$의 값을 최소화하는 $r_1, r_2, r_3, r_4$과 $r=r_1+r_2+r_3+r_4$를 구하면 다음과 같다.

$$r_1=12,\ r_2=10,\ r_3=18,\ r_4=33,\ r=73$$

Code 구성을 위한 다음 과정은 위에서 구한 73개의 state에 codeword를 할당하는 것이다. Trial and error 방법으로 codeword를 할당하기 때문에 다양한 결과를 얻을 수 있다. Codeword를 할당하는 방법은 state의 수가 73개로 증가한 것을 제외하고는 앞의 $R=6/7$인 경

표 3. 각 codeword subset과 state의 분포($R=13/15$)
Table 3. Distribution of the various codeword subsets and states($R=13/15$).

| subset state | 1-4($\tau_1$) | 5($\tau_1$) | 6-7($\tau_1$) | 8($\tau_1$) | 9($\tau_1$) | 10($\tau_1$) | 11($\tau_1$) | 12($\tau_1$) |
|---|---|---|---|---|---|---|---|---|
| $E_{000000}(546\times73)$ | $8192\times4$ | 5822 | 0 | 0 | 0 | 0 | 0 | 1095 |
| $E_{000X00}(530\times140)$ | 0 | 2370 | $8192\times2$ | 2246 | 0 | 0 | 0 | 200 |
| $E_{0000X0}(930\times22)$ | 0 | 0 | 0 | 5946 | 8192 | 5497 | 0 | 825 |
| $E_{00000X}(1455\times12)$ | 0 | 0 | 0 | 0 | 0 | 2695 | 8192 | 6072 |
| $E_{00X000}(530\times73)$ | | | | | | | | |
| $E_{00XX00}(400\times40)$ | | | | | | | | |
| $E_{00X0X0}(525\times22)$ | | | | | | | | |
| $E_{00X00X}(1275\times22)$ | | | | | | | | |
| subset state | 23-25($\tau_3$) | 26-28($\tau_3$) | 29-30($\tau_3$) | 31-38($\tau_3$) | 39($\tau_3$) | 40($\tau_3$) | 41-46($\tau_4$) | 47-53($\tau_4$) |
| $E_{0X0000}(930\times73)$ | 0 | 0 | 0 | $8192\times8$ | 0 | 1070(1284) | 0 | 0 |
| $E_{0X0X00}(525\times40)$ | 0 | 0 | $8192\times2$ | 0 | 0 | 4616 | 0 | 0 |
| $E_{0X00X0}(1275\times22)$ | 0 | $8192\times3$ | 0 | 0 | 968 | 2506 | 0 | 0 |
| $E_{0X000X}(2650\times12)$ | $8192\times3$ | 0 | 0 | 0 | 7224 | 0 | 0 | 0 |
| $E_{X00000}(1455\times73)$ | | | | | | | 0 | 0 |
| $E_{X00X00}(1275\times40)$ | | | | | | | 0 | 0 |
| $E_{X000X0}(2650\times22)$ | | | | | | | 0 | $8192\times7$ |
| $E_{X0000X}(4650\times12)$ | | | | | | | $8192\times6$ | 0 |

| subset state | 13-16($\tau_2$) | 17($\tau_2$) | 18($\tau_2$) | 19($\tau_2$) | 20($\tau_2$) | 21($\tau_2$) | 22($\tau_2$) |
|---|---|---|---|---|---|---|---|
| $E_{000000}(546\times73)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{000X00}(530\times140)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{0000X0}(930\times22)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_{00000X}(1455\times12)$ | 0 | 0 | 0 | 0 | 0 | 0 | 485(16) |
| $E_{00X000}(530\times73)$ | $8192\times4$ | 0 | 0 | 0 | 5817 | 0 | 0 |
| $E_{00XX00}(400\times40)$ | 0 | 8192 | 0 | 0 | 0 | 7808 | 0 |
| $E_{00X0X0}(525\times22)$ | 0 | 0 | 8192 | 0 | 0 | 0 | 3358 |
| $E_{00X00X}(1275\times22)$ | 0 | 0 | 0 | 8192 | 2375 | 384 | 4349 |
| subset state | 54-59($\tau_4$) | 60-71($\tau_4$) | 72($\tau_4$) | 73($\tau_4$) | | | |
| $E_{0X0000}(930\times73)$ | 0 | 0 | 0 | 0 | | | |
| $E_{0X0X00}(525\times40)$ | 0 | 0 | 0 | 0 | | | |
| $E_{0X00X0}(1275\times22)$ | 0 | 0 | 0 | 0 | | | |
| $E_{0X000X}(2650\times12)$ | 0 | 0 | 0 | 0 | | | |
| $E_{X00000}(1455\times73)$ | 0 | $8192\times12$ | 0 | 6932(979) | | | |
| $E_{X00X00}(1275\times40)$ | $8192\times6$ | 0 | 588 | 1260 | | | |
| $E_{X000X0}(2650\times22)$ | 0 | 0 | 956 | 0 | | | |
| $E_{X0000X}(4650\times12)$ | 0 | 0 | 6648 | 0 | | | |

우와 동일하다. Codeword subset과 state의 분포도는 <표 3>에 나타내었다. 가장 높은 효율을 나타낼 수 있는 $R=13/15$인 경우 $d=3$이고 $k=20$이므로 $C=0.8671$이며 $D_C=3.4684/T_{min}$이고 code efficiency는 Shannon capacity에 근접하는 무려 $e=R/C=99.95\%$가 된다.

ML-RLL을 효율적으로 검출하기 위하여 partial response(PR) equalizer를 사용한다[5~6].

참고문헌 [5]의 연구결과에 따르면 ML-RLL의 성능 향상을 위하여 $(1+2D+2D^2+2D^3+D^4)$ PR mode와 $(1+D+D^2+D^3)$ PR mode의 사용이 바람직함을 나타내었다. Three-level의 경우 $(1+2D+2D^2$

$+2D^3+D^4$) PR mode를 Viterbi detector와 사용하여 가장 좋은 성능을 보였고 trellis diagram의 state 수로 나타낸 구성의 복잡성은 가장 컸다. ($1+D+D^2+D^3$) PR mode를 Viterbi detector와 사용한 경우 성능은 약간 떨어졌으나 trellis diagram의 state 수로 나타낸 구성의 복잡성은 줄어들어 ($1+D+D^2+D^3$) PR mode의 사용을 권장하였다.

본 논문에서 6-ary RLL의 경우 ($1+2D+2D^2+2D^3+D^4$) PR mode와 ($1+D+D^2+D^3$) PR mode의 Viterbi detector의 복잡성을 trellis diagram의 state 수를 구하여 비교하였다. $d=3$인 6-ray RLL의 경우 ($1+2D+2D^2+2D^3+D^4$) PR mode의 Viterbi detector의 state 수는 96개 이고 ($1+D+D^2+D^3$) PR mode의 Viterbi detector의 state 수는 66개로 ($1+D+D^2+D^3$) PR mode의 Viterbi detector가 비교적 간단함을 보여주었다.

Decoder에 연속하여 들어오는 codeword를 information word로 정확하게 decode하기 위해서는 현재의 codeword뿐 아니라 이어 오는 다음 codeword도 알아야 한다. 따라서 single channel error는 많아야 두 개의 decoded $n$-bit symbol에 영향을 미친다. 이는 decoder가 next-state look-up table과 data look-up table이라는 두 개의 look-up table로 이루어져야 함을 의미한다. Look-up table의 크기는 encoder의 state 수가 커짐에 따라 증가하므로 작은 state 수로 encoder를 구성함은 그만큼 look-up table의 크기도 작아짐을 의미한다. 98.87 %의 효율을 보이는 $R=6/7$의 경우 총 state 수는 13개이고 99.95 %로 가장 효율이 좋은 $R=13/15$의 경우 총 state 수는 73개로 증가함을 알 수 있다.

## IV. 결 론

본 논문은 optical recording을 위한 고효율의 $d=3$인 6-ary RLL code를 구성하였다. $R=6/7$인 경우 98.87 %의 효율을 보이고 13개의 state 수를 갖는 6-ary (3, 15) code를 설계할 수 있다. $R=13/15$인 경우 Shannon capacity에 근접하는 99.95 %의 높은 효율을 보이고 73개의 state 수를 갖는 6-ary (3, 20) code를 설계할 수 있다. 또한 6-ary RLL code를 효과적으로 검출하기 위한 partial response mode에 관하여 고찰하였다.

## 참 고 문 헌


[1] S. W. McLaughlin, "Five Runlength-Limited Codes for M-ary Recording Channels," *IEEE Trans. Magn.*, vol. MAG-33, no. 3, pp.2442-2450, May 1997.

[2] B. Marcus, P. H. Siegel and J. K. Wolf, "Finite-State Modulation Codes for Data Storage," *IEEE J. on Selected Areas in Communications*, vol. 10, no. 1, pp.5-37, January 1992.

[3] K.A.S. Immink, J. Kim, S. Suh and S. Ahn, "Efficient dc-free RLL Codes for Optical Recording," *IEEE Trans. Commun.*, vol. 51, no. 3, pp.326-331, March 2003.

[4] H. Hu, L. Pan, and D. Xu, "3-ary (2, 10) Run-Length Limited Code for Optical Storage Channels," *Electronics Letters*, vol. 41, no. 17, pp.51-52, August 2005.

[5] S. H. Jiang and F. H. Lo, "PRML Process of Multilevel Run-Length-Limited Modulation Recording on Optical Disc," *IEEE Trans. on Magnetics*, vol. 41, no. 2, pp.1070-1072 February 2005.

[6] H. Hu, J. Pei, and L. F. Pan, "M-ary Even Nonzero Symbol and Run-Length Limited Code for Multilevel Read Only Memory," *Electronics Letters*, vol. 42, no. 5, pp.294-295, March 2006.


## 저 자 소 개


지 윤 규(정회원)
1978년 서울대학교
전자공학과 학사 졸업.
1980년 서울대학교
전자공학과 석사 졸업.
1984년 The University of Texas at Austin 박사 졸업.
<주관심분야 : 광통신, 광신호처리>