논문 2006-43SD-7-2

광 기록 채널을 위한 간단한 6-ary Runlength-Limited Code

(Simple 6-ary Runlength-Limited Code for Optical Storage Channels)

지 윤 규*

(Yoon Kyoo Jhee)

요 약

광 기록 채널을 위한 간단한 6-ary(3, 10) runlength-limited(RLL) code를 연구하였다. 이 제안하는 코드는 6개의 state로 나타낼 수 있어 look-up table의 크기를 작게 할 수 있다. 또한 coding density가 3.33 bits/T_{min} 이며 96.6%의 code efficiency를 나타내었다.

Abstract

A simple 6-ary(3, 10) runlength-limited(RLL) code for a six-level optical recording channel is presented. The proposed code can be represented by a finite state diagram with six states. The code is given achieving coding density of 3.33 bits/T_{min}, which is 96.6% efficient.

Keywords: 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 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].

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

RLL code를 구성하기 위하여 일반적으로 state splitting/merging 방법을 많이 사용한다. 이는 $p/q = R \leq C$ 를 만족하는 양의 정수 p와 q가 존재하면 state splitting /merging 방법으로 sliding block decoder를 갖는 finite state encoder를 항상 구성할 수 있다는 정리에 기인한다[2]. R값을 constraint capacity C의 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

* 정회원, 이화여자대학교 정보통신학과
(Dept. of Information Electronics Engineering, Ewha Womans University)
접수일자: 2006년3월2일, 수정완료일: 2006년6월14일

$d=2$ 일 때 구성하였다[4]. 이 방법은 k의 제한이 유동적인 경우에 매우 효과적이다.

본 논문에서는 위의 방법을 확장하여 $d=3$ 일 경우 6-ary (3, 10) constraint code를 구성하였다. 이는 state splitting/merging 방법을 이용한 6-ary(3, 8)의 결과 [1]와 비교할 때 code efficiency를 97%에서 96.6%로 낮추는 대신에 state의 수를 10개에서 6개로 줄일 수 있었고 따라서 look-up table의 크기를 감소시키는 효과를 얻을 수 있었다. Ⅱ절에서는 6-ary (3, 10) RLL code를 설계하는 방법을 설명하고 Ⅲ절에서 결론을 맺는다.

Ⅱ. State 수를 감소시키는 6-ary (3, 10) RLL code의 설계

Multiamplitude (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. Density ratio D는 다음 식으로 정의된다.

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

본 논문의 경우 $d=3$이고 $R=m/n=5/6$이므로 $D=3.33$ bits/$T_{\min}$이다. Constraint capacity $C=\log_2\lambda$로 정의되며 λ는 다음 특성방정식의 가장 큰 실수 근이다.

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

도달할 수 있는 가장 큰 density $D_C=\frac{(1+d)C}{T_{\min}}\geq\frac{(1+d)R}{T_{\min}}=D$로 표시된다. Code efficiency $e=R/C=D/D_C$로 정의되고 code가 이룰 수 있는 가장 큰 density에 근접하는 척도를 나타낸다 [1]. 이 절에서 구성하려는 code의 경우 $d=3$이고 $k=10$이므로 $C=0.863$이며 $D_C=3.45/T_{\min}$이고 code efficiency $e=R/C=96.6\%$가 된다.

이 절에서는 k의 조건은 일단 무시하고 $d=3$의 제한조건을 만족시키는 finite-state encoder의 설계에 대하여 설명한다. Encoder는 r개의 state를 지니고 있다고 가정하고 네 가지 형태의 subset으로 나눈다. 각 subset에 속해있는 state의 크기를 각각 r_1, r_2, r_3와 $r_4(=r-r_1-r_2-r_3)$로 정의한다. Codeword는 $d=3$의 조건을 만족하며 길이가 $n=6$인 6-ary symbol로 구성한다. 이 codeword 또한

$$E_{000000}, E_{000X00}, E_{0000X0}, E_{00000X},$$
$$E_{00X000}, E_{00XX00}, E_{00X0X0}, E_{00X00X},$$
$$E_{0X0000}, E_{0X0X00}, E_{0X00X0}, E_{0X000X}, E_{X00000}, E_{X00X00},$$
$$E_{X000X0}, E_{X0000X}$$

의 16개 subset으로 나눈다. 여기서 X는 1, 2, 3, 4 또는 5 중 하나의 값을 나타낸다. 16 개의 subset에 속해있는 codeword들은 네 가지 형태의 state에 분배된다. 네 가지 형태의 state들은 다음과 같다. 첫 번째 형태의 state에 속해있는 codeword들은 "000"으로 시작한다. 두 번째 형태의 state에 있는 codeword는 "000" 또는 "00X"로 시작한다. 세 번째 형태의 state에 속해있는 codewod들은 "000" 또는 "00X" 또는 "0X0"로 시작한다. 또한 네 번째 형태의 state에 속해있는 codeword들은 "000" 또는 "00X" 또는 "0X0" 또는 "X00"로 시작한다. 이를 그림으로 나타내면 그림 1과 같다.

State-transition은 다음의 rule을 따른다. 그림 1이 보여주듯이 $E_{000000}, E_{00X000}, E_{0X0000}, E_{X00000}$와 같이 "000"으로 끝나는 codeword들은 r개의 모든 encoder state로 연결될 수 있다. "X00" (E_{000X00}, E_{00XX00},

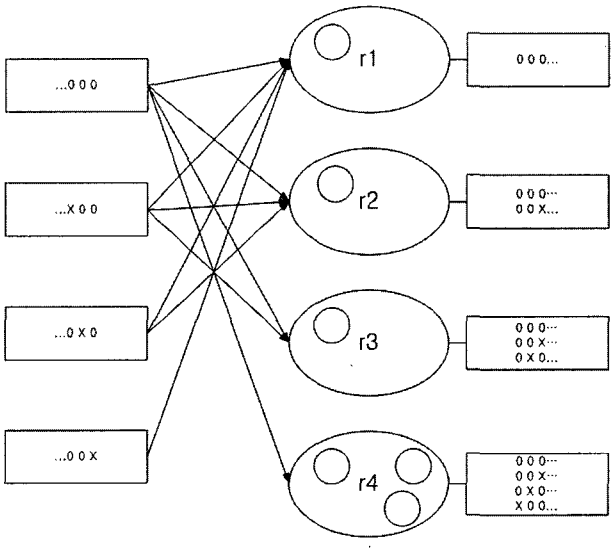

그림 1. $n=6$인 경우 encoder의 transition 방법
Fig. 1. Encoder transition rule($n=6$).

E_{0X0X00}, E_{X00X00})로 끝나는 codeword는 네 번째 형태의 state에 속해 있는 codeword에 이어질 수 없다. "0X0"(E_{0000X0}, E_{00X0X0}, E_{0X00X0}, E_{X000X0})로 끝나는 codeword는 세 번째와 네 번째 형태의 state에 속해있는 codeword에 이어질 수 없다. 같은 방법으로 "00X"(E_{00000X}, E_{00X00X}, E_{0X000X}, E_{X0000X})로 끝나는 codeword는 오직 첫 번째 형태의 state에 속해 있는 codeword로만 이어진다.

Next state(NS)가 다르면 동일한 codeword에 다른 information word를 할당할 수 있다. 즉, E_{000000}, E_{00X000}, E_{0X0000}, E_{X00000}와 같이 "000"으로 끝나는 codeword는 모든 r개의 state와 연결될 수 있으므로 서로 다른 information word를 $r=r_1+r_2+r_3+r_4$ 배만큼 할당 할 수 있다. 또한 E_{000X00}, E_{00XX00}, E_{0X0X00}, E_{X00X00}와 같이 "X00"으로 끝나는 codeword는 네 번째 형태의 state를 제외한 모든 state와 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 state와 연결될 수 있으므로 r_1+r_2배만큼 할당 할 수 있다.

같은 방법으로 E_{00000X}, E_{00X00X}, E_{0X000X}, E_{X0000X}와 같이 "00X"으로 끝나는 codeword의 경우는 첫 번째 형태의 state로만 연결될 수 있으므로 r_1배만큼 할당 할 수 있다. 위의 관계를 state transition diagram으로 나타내면 그림 2가 된다.

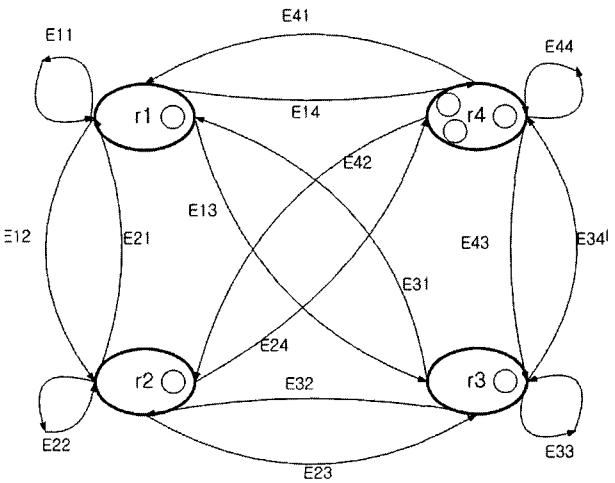

그림 2. $n=6$ 인 경우 state간의 transition diagram
Fig. 2 M-ary state transition diagram($n=6$).

그림 2에서

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

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

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

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

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

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

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

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

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

$E_{34}: E_{000000}, E_{000X00}, E_{00X000}, E_{0X0000}$;

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

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

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

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

를 나타낸다.

위에서 설명한 encoder model을 기반으로 $R=m/n=5/6$ 인 RLL code를 설계하기 위하여 다음 식을 정의한다. 아래 식에서 $|E_{WXYZ}|$는 E_{WXYZ}의 크기를 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 $n=6$ 인 경우에 codeword subset의 크기를 구하면 다음과 같다.

$$|E_{000000}| = 1, \tag{7}$$

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

$$|E_{0X000X}| = |E_{X000X0}| = |E_{X0000X}| = 25, \quad (9)$$

$$|E_{00XX00}| = |E_{00X0X0}| = |E_{00X00X}| = |E_{0X0X00}| = |E_{0X00X0}| = |E_{X00X00}| = 0. \quad (10)$$

$m = 5$ 인 경우에 encoder model은 다음 조건을 만족하여야 한다.

$$A_1 \geq r_1 2^m = 32r_1, \quad (11)$$

$$A_1 + A_2 \geq (r_1 + r_2)2^m = 32(r_1 + r_2), \quad (12)$$

$$A_1 + A_2 + A_3 \geq 32(r_1 + r_2 + r_3), \quad (13)$$

$$A_1 + A_2 + A_3 + A_4 \geq 32(r_1 + r_2 + r_3 + r_4). \quad (14)$$

(11) 식은 r_1 state에서 나가는 최소의 codeword 수는 $r_1 2^m = 32r_1$ 이어야 함을 의미한다. 위에서 구한 값들을 대입하여 정리하면 다음식이 된다.

$$-16r_1 + 10r_2 + 5r_3 + r_4 \geq 0 \quad (11)'$$

$$-11r_1 - 17r_2 + 10r_3 + 6r_4 \geq 0 \quad (12)'$$

$$19r_1 - 12r_2 - 17r_3 + 11r_4 \geq 0 \quad (13)'$$

$$74r_1 + 18r_2 - 12r_3 - 16r_4 \geq 0 \quad (14)'$$

(11)'-(14)'를 만족하면서 r의 값을 최소화하는 r_1, r_2, r_3, r_4과 $r = r_1 + r_2 + r_3 + r_4$를 구하면 다음과 같다.

$$r_1 = r_2 = r_3 = 1,\ r_4 = 3,\ r = 6. \quad (15)$$

Code 구성을 위한 다음 과정은 위에서 구한 6개의 state에 codeword를 할당하는 것이다. Trial and error 방법으로 codeword를 할당하기 때문에 다양한 결과를 얻을 수 있고, 한 예를 <표 1>에 나타내었다. <표 1>의 $a \times b$에서 a은 codeword의 수를 b는 연결되는 state의 수를 나타낸다. <표 1>에서 보여주는 바와 같이 크기가 5인 E_{00000X}은 state 1에 세 개의 codeword를 할당하고 state 2에 두 개의 codeword를 할당한다. 즉, 각 행의 합이 각 codeword subset의 크기와 같도록 할당한다. <표 1>의 state 열에는 실제로 사용한 codeword 개수만을 나타냈기 때문에 각 행의 합이 codeword subset 보다 적은 경우도 있다.

표 1. 각 subset과 state의 분포
Table 1. Distribution of the various subsets and states.

subset \ state	$1(r_1)$	$2(r_2)$	$3(r_3)$	$4(r_4)$	$5(r_4)$	$6(r_4)$
$E_{000000}(1\times6)$	1×4	0	0	0	0	0
$E_{000X00}(5\times3)$	5×3	0	0	0	0	0
$E_{0000X0}(5\times2)$	5×2	0	0	0	0	0
$E_{00000X}(5\times1)$	3×1	2×1	0	0	0	0
$E_{00X000}(5\times6)$		5×6	0	0	0	0
$E_{0X0000}(5\times6)$			5×5	0	0	0
E_{0X000X} (25×1)			7×1	6×1	6×1	6×1
$E_{X00000}(5\times6)$				2×4	2×4	1×4
E_{X000X0} (25×2)				9×2	9×2	7×2
E_{X0000X} (25×1)				0	0	$24\times$

E_{000000}, E_{00X000}, E_{0X0000}와 E_{X00000} 같이 "000"으로 끝나는 codeword에는 서로 다른 information word를 6번 할당할 수 있고 E_{000X00}와 같이 "X00"로 끝나는 codeword에는 서로 다른 information word를 3번 할당할 수 있다. 마찬가지로 E_{0000X0}와 E_{X000X0}같이 "0X0"로 끝나는 codeword에는 서로 다른 information word를 2번 할당할 수 있고 E_{00000X}, E_{0X000X}와 E_{X0000X} 같이 "00X"로 끝나는 codeword에는 information word를 1번만 할당할 수 있다. 따라서 state 1에 할당된 총 codeword 수는 $1\times6+5\times3+5\times2+3\times1=34 \geq 32=2^5$ 이 된다. 나머지 state에 할당되는 codeword 수도 32개 이상이 되도록 구성하여야 한다.

위에서 설명한 바와 같이 각 state에 필요한 32 codeword보다 더 많은 code를 할당할 수 있으므로 초과되는 최악의 codeword를 제거하여 k의 제한조건을 최소화하도록 한다. 예를 들면 codeword "000000" 다음에 codeword "000000"나 "00000X"가 이어지는 것을 방지하도록 구성하여 많은 수의 "0"이 나타나지 않도록 한다. 이는 <표 2>의 state 1열에서 output "000000"일 때, NS 1과 2를 사용하지 않는 것이 된다. 결과적으로 state가 6개이며 $k=10$인 6-state 6-ary(3, 10)를 구성할 수 있다. <표 1>에 근거하여 구성한 encoding table

표 2. Encoding을 위한 테이블
Table 2. Encoding table.

input	state 1		state 2		state 3		state 4		state 5		state 6	
	output	NS	output	NS	output	NS	output	NS	output	NS	output	NS
1	000000	3	000004	1	010000	2	020003	1	030004	1	040005	1
2	000000	4	000005	1	010000	3	020004	1	030005	1	050001	1
3	000000	5	001000	1	010000	4	020005	1	040001	1	050002	1
4	000000	6	001000	2	010000	5	030001	1	040002	1	050003	1
5	000001	1	001000	3	010000	6	030002	1	040003	1	050004	1
6	000002	1	001000	4	020000	2	030003	1	040004	1	050005	1
7	000003	1	001000	5	020000	3	100000	3	300000	3	500000	3
8	000010	1	001000	6	020000	4	100000	4	300000	4	500000	4
9	000010	2	002000	1	020000	5	100000	5	300000	5	500000	5
10	000020	1	002000	2	020000	6	100000	6	300000	6	500000	6
11	000020	2	002000	3	030000	2	200000	3	400000	3	400040	1
12	000030	1	002000	4	030000	3	200000	4	400000	4	400040	2
13	000030	2	002000	5	030000	4	200000	5	400000	5	400050	1
14	000040	1	002000	6	030000	5	200000	6	400000	6	400050	2
15	000040	2	003000	1	030000	6	100010	1	200050	1	500010	1
16	000050	1	003000	2	040000	2	100010	2	200050	2	500010	2
17	000050	2	003000	3	040000	3	100020	1	300010	1	500020	1
18	000100	1	003000	4	040000	4	100020	2	300010	2	500020	2
19	000100	2	003000	5	040000	5	100030	1	300020	1	500030	1
20	000100	3	003000	6	040000	6	100030	2	300020	2	500030	2
21	000200	1	004000	1	050000	2	100040	1	300030	1	500040	1
22	000200	2	004000	2	050000	3	100040	2	300030	2	500040	2
23	000200	3	004000	3	050000	4	100050	1	300040	1	500050	1
24	000300	1	004000	4	050000	5	100050	2	300040	2	500050	2
25	000300	2	004000	5	050000	6	200010	1	300050	1	100001	1
26	000300	3	004000	6	010001	1	200010	2	300050	2	100002	1
27	000400	1	005000	1	010002	1	200020	1	400010	1	100003	1
28	000400	2	005000	2	010003	1	200020	2	400010	2	100004	1
29	000400	3	005000	3	010004	1	200030	1	400020	1	20000X	1
30	000500	1	005000	4	010005	1	200030	2	400020	2	30000X	1
31	000500	2	005000	5	020001	1	200040	1	400030	1	40000X	1
32	000500	3	005000	6	020002	1	200040	2	400030	2	50000X	1

이 <표 2>에 나타나 있다. 이 표에서 NS는 next state를 의미한다.

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 5-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 6개의 state 수로 encoder를 구성함은 그만큼 look-up table의 크기도 작아짐을 의미한다. State splitting/merging 방법으로 구성한 6-ary (3, 8) encoder는 동일한 rate 5/6를 구현하기 위하여 10개의 state가 필요하였다[1]. 이 논문과 비교하여 보면 code efficiency는 97%에서 96.6%로 낮아지는 대신에 state의 수를 10개에서 6개로 줄일 수 있었고 따라서 look-up table의 크기를 감소시키는 효과를 얻을 수 있었다. 이는 binary search를 가정할 경우 complete search를 위한 최대 시행 횟수가 ($\lfloor \log_2 N \rfloor + 1$)이므로 look-up table의 entry 수가 320개에서 192개로 줄어든 것은 최대 시행 횟수가 9에서 8로 감소하였음을 의미한다.

Ⅳ. 결 론

본 논문은 optical recording을 위한 간단한 5/6 rate의 6-ary(3, 10) RLL code를 구성하였다. 이 code의 capacity $C=0.863$이고 coding density $D=3.33$ bits/$T_{\min}$이다. 기존의 논문과 비교하면 본 논문의 결과는 code efficiency는 97%에서 96.6%로 낮아지는 대신에 state의 수를 10개에서 6개로 줄일 수 있었고 따라서 look-up table의 크기를 감소시키는 효과를 얻을 수 있었다.

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.

저 자 소 개

지 윤 규(정회원)

1978년 서울대학교 전자공학과 학사 졸업.

1980년 서울대학교 전자공학과 석사 졸업.

1984년 University of Texas at Austin 전자공학과 박사 졸업.

<주관심분야 : 광통신, optical recording>