

Construction of Efficient Dc-Free Runlength-Limited Codes for Optical Recording

Kees A. Schouhamer Immink¹, Jin-Yong Kim², Sang-Woon Suh², Seong Keun Ahn²

1. Turing Machines Inc, 15 W. Alexanderlaan, 5664 AN Geldrop, The Netherlands. E-mail: immink@turing-machines.com
2. DCT Team, Multi-Media Labs, LG Electronics Inc., 16 Woomyeon-Dong, Seocho-Gu, Seoul 137-724, Korea

Abstract

We will report on new dc-free runlength-limited codes (DCRLL) intended for the next generation of digital versatile disc (DVD). The efficiency of the newly developed DCRLL schemes is extremely close to the theoretical maximum, and as a result, significant density gains can be obtained with respect to prior art coding schemes. Preferred embodiments of the codes (footprints) and respective hardware realization will be considered.

Keywords: optical recording, capacity, constrained code, runlength-limited, RLL sequence, (d, k) sequence, dc-free code

1 Introduction

Optical recording, developed in the late 60s and early 70s, is the enabling technology of a series of very successful products for digital consumer electronics systems such as Compact Disc (CD), CD-ROM, CD-R, DVD, and many other high-density products that are still on the drawing board. Notably spectral shaping (dc-free) and runlength-limited (RLL) codes have found widespread usage in consumer-type mass storage systems such as compact disc, DAT, digital versatile disc (DVD), and so on, see ref. 1. The design of codes for optical recording is essentially the design of combined *dc-free* and *runlength limited* (DCRLL) codes. Eight to fourteen modulation (EFM) developed by Immink & Ogawa in the early eighties, ref. 2, was adopted as the recording code for the Compact Disc (CD). EFMplus, ref. 3, used in the DVD, is a code with the same basic parameters as EFM and a useful six

percent higher efficiency. Table I gives a survey of recording codes, which are part of consumer-type optical recording products. Binary sequences generated by a (d, k) RLL encoder have at least d and at most k 'zero's between successive 'one's. Let the integers m and n denote the information word length and codeword length, respectively. The *code rate*, $R = m/n$, is a measure of the code's efficiency. The maximum rate of an RLL code, given values of d and k , is called the *Shannon capacity*, and it is denoted by $C(d, k)$.

As an example, Table II tabulates $C(d = 1, k)$ and $C(d = 2, k)$ for relevant values of k . We may observe, for example, that for $d = 1$ and $k = 7$ the Shannon capacity, $C(1, 7)$, has a value of 0.6793. Thus, an encoder that translates arbitrary sequences into sequences that have at least $d = 1$ and at most $k = 7$ 0's between successive 1's, cannot have a rate larger than 0.6793.

Information recording has a constant need for enhancing the information density on the record carrier, and a possible solution to this end is an increase of the rate of the code.

2 Very Efficient Coding Schemes

For ease of presentation we will first focus on the design of RLL codes with $d = 1$. Later we will extend the ideas to the design of codes with $d = 2$.

Rate $2/3$, $(1, 7)$ codes are known in the art for more than a quarter of a century, see for example refs. 5 and 6. The code rate, $2/3$, of the $(1, 7)$ code is slightly less than the Shannon capacity, 0.6793, and the code is therefore a highly efficient one. The efficiency of an RLL code is usually measured by a quantity called *code efficiency*, η , defined by

$$\eta = R/C(d, k). \tag{1}$$

There are only two approaches for constructing a $(1, k)$ RLL code, whose rate is larger than two-thirds. Firstly, we may relax the maximum runlength k to a value larger than 7. Note that a $(1, 7)$ code was first put to practical use in the early seventies, and that since the advent of hard-disk drives (HDDs), significant improvements in signal processing for timing recovery circuits have made it possible to employ codes with a much larger maximum runlength k . Secondly, on top of that we may endeavor to design a more efficient code.

The efficiency of the rate $2/3$, $(1, 7)$ code is $0.6667/0.6793 = 0.981$, which reveals that we can at most gain 1.9% in rate by an alternative, more efficient, code redesign. If we fully relax the k constraint, i.e. set $k = \infty$, we can at

most gain 3.97% in code rate. In other words, a viable improvement in code rate of a ($d = 1$) encoder ranges from 1.9 to 3.97%. To the best of the author's knowledge, extremely efficient ($d = 1$) codes having a rate exceeding two-thirds have not been reported in the literature. In the sequel of the paper, we will systematically design such extremely efficient codes. We start, in the next subsection, with a simple problem, namely finding integers m and n that improve the rate, $2/3$, of the industry standard code.

2.1 Suitable integers m and n for $d = 1$

We will start with a simple, but very illuminating exercise, namely a search for pairs of integers m and n that are suitable candidates for a coding rate exceeding $2/3$. All pairs of integers $2/3 < m/n < C(1, \infty)$, $n < 50$, are shown in Table III. Surprisingly there are just six m and n pairs whose quotient is larger than $2/3$. We omitted trivial pairs, such as 18 and 26 etc., that are multiples of given smaller pairs. Perusal of the table reveals that the code rate $m/n = 9/13$ is highly attractive as it is just 0.28% below the Shannon capacity $C(1, \infty)$. The fact that the quotient $9/13$ is less than capacity does not mean that a code with that rate can be *practically* constructed.

2.2 Encoder description

We start with a few ubiquitous definitions. The encoder has r states, which are divided into two state subsets of a first and second type. The state subsets are of size r_1 and $r_2 (= r - r_1)$, respectively. A codeword is a binary string of length n that satisfies the $d = 1$ constraint. The encoder state-transition rules are easily described. Codewords that end with a '0', i.e., codewords in subsets E_{00} and E_{10} may enter any of the r encoder states. Codewords that end with a '1' may be followed by codewords in the r_1 states of the first type only. With the above model we were able to construct many new codes including a rate $9/13$, (1,14) code. Clearly this new code improves the rate of the traditional rate $2/3$, (1,7) code by a factor of $27/26 (\approx 1.038)$ without seriously compromising the timing regeneration.

3 Efficient $d = 2$ Codes

Up till now we have concentrated on the design of efficient $d = 1$ codes, and as both code parameters, $d = 1$ and $d = 2$, are of great practical interest for optical recording, we will now repeat the exercise for the case $d = 2$.

3.1 Suitable integers m and n for $d = 2$

RLL codes with minimum runlength parameter $d = 2$ have been widely published. The highest reported rate of such a ($d = 2$) code is $8/15$. (At press time, the author became aware that Kim, ref. 7, has been granted a U.S. Patent on an example of a rate $7/13$, (2,25) code.) Table II tabulates $C(2, k)$ as a function of k , and from this table the reader can easily discern the head room available for the design of a code of rate $R = m/n > 8/15$. The rate $8/15$ is, see Table II, 3.3% below channel capacity $C(2, \infty)$. Table IV shows values of m and n , where $8/15 \leq m/n < C(2, \infty)$ and $n < 50$. The pairs of integers are ordered according to their efficiency $R/C(2, \infty)$.

Clearly, the quotients $11/20$, $6/11$, and $7/13$ are suitable candidate rates for the creation of small ($d = 2$) codes.

3.2 Encoder description

In this section we will describe a finite-state encoder that generates sequences that satisfy the $d = 2$ constraint (note that the k constraint will be ignored for a while). The encoder is assumed to have r states, which are divided into three state subsets of states of a first, second, and third type. The state subsets are of size r_1 , r_2 , and $r_3 (= r - r_1 - r_2)$, respectively. Codewords that end with the string '00' may enter any of the r encoder states. Codewords that end with a '10' may not be followed by codewords in a state of the third type. Similarly, codewords that end with a '1' may only be followed by codewords belonging to states of the first type. Table V summarizes the new RLL codes, $d = 1$ and $d = 2$, we have found. As we can see, the efficiency of the majority of the new codes is just a few tenths of a percent below capacity.

At this junction, we have completed the description of the new RLL codes, and we are in the position to describe how we can turn the newly developed RLL codes into DCRL codes.

4 Guided Scrambling

There are various methods to transform an RLL code into a DCRL code, ref. 1, by adding redundant dc-control bits, which are chosen by the encoder to optimize the spectral performance of the generated sequence. Obviously, we can multiplex, either at data or channel level, the data stream with the dc-control bits. Alternatively, an excellent method for extending an RLL code is *Guided Scrambling (GS)*, ref. 1. In GS, each information word can be represented by a member of a selection set consisting of $L = 2^p$, $p \geq 1$, codewords. The encoder generates the selection set, and the “best” (according to a predefined penalty function) codeword in the selection set is selected for transmission. The RLL codes, listed in Table V, will be employed in conjunction with GS for achieving two goals:

- spectral shaping;
- rejection of long runs of '0's: k constraint;

The maximum runlength constraint, k , imposed by the GS penalty function can be made smaller than that of the inner RLL code. Naturally, the GS method cannot fully guarantee the k constraint, but the probability of occurrence of runs exceeding the set k constraint can be made small.

5 Implementation Aspects

In the GS format, m_1 user bits are multiplexed with p redundant bits, which are a part of the input of the channel encoder. The p redundant bits are used to generate a selection set of size $L = 2^p$. Each member of the selection set is generated and tested by the encoder with respect to the penalty function. In the proposed coding format, the channel encoder input comprises p redundant bits plus m_1 user bits that from a *super* block. The integers p and m_1 are integers chosen such that

$$Km = p + m_1, \tag{2}$$

where K is an integer that denotes the number of m -bit information words in a super block. In a practical environment of a byte-oriented system, m_1 is preferably a multiple of eight, i.e. $m_1 \bmod 8 = 0$. Under the rules of the

RLL code, the $p + m_1 = Km$ -bit super block is translated into Kn channel bits. Thus, the overall rate, R_o , of the code is

$$R_o = \frac{m_1}{Kn} = \frac{Km - p}{Kn}. \quad (3)$$

In order to compare the performance of the newly developed codes with that of prior art codes, we have worked out many examples. Below we will describe two examples of proposed footprints.

5.1 Footprint $d = 1$

The electronics of our experimental optical recording set up is based on the DVD format (scrambling, product ECC code, and EFMPlus). In the DVD format, 91 bytes, or 728 bits, of user data are translated into EFMPlus frames under the directives of EFMPlus. For the sake of convenience we also adopted the same user block of $m_1 = 91$ bytes, for the new experimental format. We employed the 5-state, rate 9/13 ($d = 1, k = 14$) code in conjunction with Guided Scrambling (GS) described above *in lieu* of the standard EFMPlus code. We split the fame of user information into two equal segments of $728/2=364$ bits and add $p = 5$ redundant bits to each segment, so that $K = 369 = 9 \times 41$. The 364 user bits are translated into $41 \times 13 = 533$ channel bits. Then the overall rate is $R_o = 364/533 = 0.6829$. The maximum runlength constraint, k , imposed by the GS penalty function was set to $k = 11$. The probability of occurrence of a maximum runlength, $k = 11$, violation is very small ($< 10^{-9}$). We have implemented the (1,11) GS code in hardware and amply tested its real-time performance. The hardware size of the ROM encoder tables is about 9.3 Kbyte and the extra gate size is 40,000. The decoder gate size is approximately 25,000.

A second footprint is based on the proposed DVR format, ref. 4. The overall rate of the so-called (1,7)PP scheme is according to ref. 4, $45/69=0.6523$, and we therefore conclude that the rate of the new code is 4.7% greater than that of the prior art (1,7)PP scheme. In addition, we may have extra gain as the GS format allows a more efficient redesign of the sync formats. The lf suppression of the new code is 2 dB better than that of the (1,7)PP scheme.

5.2 Footprint $d = 2$

For the $d = 2$ case, we have the same input data per frame, i.e. 91 bytes. The 10-state, rate 6/11, (2,12) is used as the inner RLL code. The frame

of 728 user bits is split up into two segments of 361 and 367 user bits. To each segment we add $p = 5$ redundant bits. The segments are translated into 671 and 682, channel bits, respectively. Thus the overall rate is $R_o = 728/(671 + 682) = 0.538$. Note that the rate of EFMPPlus is $8/16$ so that the newly developed scheme gains 7.65% in density. The maximum runlength constraint, k , imposed by the GS penalty function was set to $k = 12$.

6 Experimental Results

Optical recording experiments with the new ($d = 1$) footprint have been conducted. The experiments showed that essential parameters such as jitter, bit error rate, and so on, are, as anticipated, the same as those of their prior art counterparts. In view of the many recording experiments we conducted, we may therefore safely conclude that the new codes significantly gain in coding rate, i.e. recording density, without compromising other essential recording parameters. Figure 1 displays an eye pattern obtained with the conventional (1,7)PP code using a blue laser experimental set-up. Figure 2 shows, with the same physical conditions, but at a 4.7% higher information density, a typical eye pattern obtained with the new (1,11)GS code.

7 Conclusions

We have studied the construction and implementation aspects of extremely efficient runlength-limited (RLL) codes. We have shown that there is a very limited number of pairs of integers m and n , whose quotient m/n form a suitable coding rate for ($d = 1$) and ($d = 2$) RLL codes that are more efficient than prior art codes. Suitable values for the rate of a ($d = 1$) code are $9/13$ and $11/16$, while for ($d = 2$) codes we have $11/20$, $7/13$, and $6/11$.

The above, and other, RLL codes can be employed in conjunction with Guided Scrambling (GS), or other techniques, to turn them into DC-free RLL codes, which suppress the low-frequency (lf) components. We have presented footprints (coding formats) of the new codes. With the newly developed rate $9/13$, $d = 1$ code as an inner code, we have achieved a 4.6% higher overall rate than possible with the prior art (1,7)PP code. With the newly developed rate $6/11$, $d = 2$ code we have achieved a 7.6% higher overall rate than that of EFMPPlus. Higher coding gains are possible for $d = 1$ and $d = 2$ as the new

coding format allows a more efficient synchronization. The lf suppression is similar to the prior art codes, but overall coding rate and lf suppression can easily be traded. Recording experiments with the new codes have shown that jitter and bit error rate of the new codes are essentially the same as those measured with the prior art codes.

References

- [1] K.A.S. Immink: *Codes for Mass Data Storage Systems*, (Shannon Foundation Publishers, Netherlands, 1999).
- [2] K.A.S. Immink and H. Ogawa: US Patent 4,501,000, Feb. 1985.
- [3] K.A.S. Immink: *IEEE Trans. Consumer Electron.*, CE-41, (1995) 491.
- [4] K.A.S. Immink, J.A.H. Kahlman, G. van den Enden, T. Nakagawa, Y. Shinpuku, T. Narahara and K. Nakamura: Patent Application WO 9963671A1, Dec. 1999.
- [5] P.A. Franaszek: *IBM J. Res. Develop.*, 23, (1979) 75.
- [6] G.V. Jacoby and R. Kost: *IEEE Trans. Magn.*, MAG-20, (1984) 709.
- [7] M.J. Kim: US Patent 6,188,336, Feb. 2001.
- [8] K.A.S. Immink: *IEEE Trans. Consumer Electron.*, CE-43, (1997) 491.

Figure Captions

Figure 1: Eye pattern obtained with the conventional (1,7)PP code. Jitter 7.5%.

Figure 2: Eye pattern obtained with the new (1,11) GS code at a 4.7% higher density than with the conventional (1,7)PP code. Jitter 8%.

Table I: Survey of recording codes and their application area

<i>Device</i>	<i>Code</i>	<i>Type</i>	<i>d,k</i>	<i>Ref.</i>
CD	EFM	DCRLL	1,10	2
MiniDisc	EFM	DCRLL	1,10	
DVD	EFMPlus	DCRLL	1,10	3
DVR	(1,7)PP	DCRLL	1,7	4

Table II: Capacity $C(1, k)$ and $C(2, k)$ as a function of k .

k	$C(1, k)$	$C(2, k)$
7	0.6793	0.5174
8	0.6853	0.5293
9	0.6888	0.5369
10	0.6909	0.5418
11	0.6922	0.5450
∞	0.6942	0.5515

Table III: Integers m and n such that $2/3 < R = m/n < C(1, \infty)$.

m	n	$1 - \eta$ %
34	49	0.0525
9	13	0.2786
11	16	0.9711
13	19	1.4449
15	22	1.7895
17	25	2.0514

Table IV: Integers m and n such that $8/15 < R = m/n < C(2, \infty)$.

m	n	$1 - \eta$ %
11	20	0.2720
17	31	0.5644
6	11	1.0962
19	35	1.5672
13	24	1.7830
20	37	1.9872
7	13	2.3642
15	28	2.8623
8	15	3.2940

Table V: Survey of newly developed codes.

m	n	d	k	states	$\eta = R/C(d, k)$
11	20	2	23	9	0.9975
7	13	2	11	9	0.9880
6	11	2	15	9	0.9915
9	13	1	14	13	0.9979
9	13	1	18	5	0.9973
11	16	1	10	13	0.9951