

ON THE SIZE OF A TYPE OF RLL CODES

Kees A. Schouhamer Immink

Turing Machines Inc,

Willemskade 15b-d, 3016 DK Rotterdam, The Netherlands.

E-mail: immink@turing-machines.com.

&

Cai Kui

Data Storage Institute,

5 Engineering Drive 1, Singapore 117608, Republic of Singapore.

Summary - *We will report on a relationship between the size of certain runlength-limited (RLL) codes and encoder complexity expressed as the number of encoder states, where the number of encoder states equals a (generalized) Fibonacci number.*

Key words: optical recording, channel capacity, constrained code, runlength-limited, RLL sequence, (d, k) sequence.

INTRODUCTION

Runlength-limited codes, generically designated as (d, k) RLL codes, have been widely and successfully applied in modern magnetic and optical recording systems. Binary sequences generated by a (d, k) RLL encoder have at least d and at most k , $k > d$, '0's between successive '1'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 $C(d, k)$. Thus for any (d, k) code we have $R \leq C(d, k)$.

Finite-state RLL encoders have become very popular in recording practice as the rate of such well-designed transducers approaches capacity. Immink *et al.* [1] have introduced a new family of simple and efficient finite-state RLL codes. It is of engineering interest to have a knowledge of the relationship between the encoder's hardware complexity measured in the number of encoder states and the code's efficiency. In this article, we will report on a relationship between the code size and the number of encoder states. For ease of presentation we will first focus on RLL codes with $d = 1$, which we will later extend to the design of codes with $d = 2$.

ENCODER DESCRIPTION

We start with a few definitions. A *codeword* is a binary string of length n that satisfies the $d(= 1)$ constraint. The set of codewords, E , is divided into four subsets E_{00} , E_{01} , E_{10} , and E_{11} . The four subsets are characterized as follows. Codewords in E_{00} start and end with a '0', codewords in E_{01} start with a '0' and end with a '1', etc. The encoder has r states, which are divided into two state subsets of a first and second type. The encoder has r_1 states of the first type and $r_2(= r - r_1)$ states of the second type. The two types of coding states are characterized by the fact that all codewords in the r_1 states of the first type must start with a '0', while codewords in the r_2 states of the second type are free to start with a '1' or a '0'.

We are now in the position to describe the encoder state-transition rules. Codewords that end with a '0', i.e., codewords in subsets E_{00} and E_{10} may enter any of the $r = r_1 + r_2$ encoder states. Codewords that end with a '1' may only enter the r_1 states of the 1st type only (and not the states of the 2nd kind). Note that, by definition, the codewords in states of the first type start with a '0', and codewords in states of the 2nd type may start with a '1', which prohibits that a codeword ending with '1' may enter states of the 2nd type.

The sets of codewords that belong to a given state (of any type) must be disjoint. This attribute implies that any codeword can unambiguously be identified to the state from which it emerged. The decoder can, by observing both the current and the next codeword uniquely decide which of the information words was actually transmitted. Codewords ending with a '0' can be followed by codewords in any of r states, and can, thus, be assigned r times to different information words. Similarly, codewords ending with a '1' can only be followed by the r_1 states of the 1st kind, and can therefore be assigned r_1 times to different information words. Given the above encoder model, we can write down two existential conditions of a $d = 1$ code of size M , where usually $M \geq 2^m$.

Let $|E_{xy}|$ denote the size of E_{xy} . Then, following the above arguments, there are at maximum $r|E_{00}| + r_1|E_{01}|$ codewords leaving the r_1 states of the first type. For a code of size M , there should be at least r_1M codewords leaving the r_1 states of the first type. Thus we can write down the first condition

$$r|E_{00}| + r_1|E_{01}| \geq r_1M. \quad (1)$$

Similarly, the second condition follows from the fact that there should be a sufficient amount of codewords leaving the r states. We find

$$r(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) \geq rM. \quad (2)$$

Note that the above two conditions are necessary conditions for the construction of such RLL code. In the sequel, we will study various properties of the new code constructions. Specifically, we will study the code size for a specific selection of r and r_1 , where r and r_1 are Fibonacci numbers. To that end, let $N_d(n)$ denote the number of distinct (d) sequences of length n and define

$$\begin{aligned} N_d(n) &= 0, \quad n < 0, \\ N_d(0) &= 1. \end{aligned} \quad (3)$$

The number of (d) sequences of length $n > 0$ is found by

$$\begin{aligned} N_d(n) &= n + 1, \quad 1 \leq n \leq d + 1, \\ N_d(n) &= N_d(n - 1) + N_d(n - d - 1), \quad n > d + 1. \end{aligned} \quad (4)$$

The numbers $N_1(n)$

$$1, 2, 3, 5, 8, 13, \dots,$$

where each number is the sum of its two predecessors, are called *Fibonacci numbers*. We simply find that the numbers of sequences in the four subsets are given by

$$\begin{aligned} |E_{00}| &= N_1(n - 2), \\ |E_{01}| &= |E_{10}| = N_1(n - 3) \\ |E_{11}| &= N_1(n - 4). \end{aligned} \quad (5)$$

Thus, the left-hand side of Conditions (1) and (2) now read

$$rN_1(n - 2) + r_1N_1(n - 3) \quad (6)$$

and

$$rN_1(n - 1) + r_1N_1(n - 2). \quad (7)$$

We can freely choose any convenient r and r_1 , but for reasons of mathematical elegance we select $r = N_1(q)$, $q \geq 0$, and $r_1 = N_1(q - 1)$. Then with (4), we find $r_2 = r - r_1 = N_1(q - 2)$. In other words, the numbers of code states, r , r_1 , and

r_2 are Fibonacci numbers. By recursive elimination we can proof that [2]

$$\begin{aligned} rN_1(n-2) + r_1N_1(n-3) &= N_1(q)N_1(n-2) + N_1(q-1)N_1(n-3) \\ &= N_1(n+q-1). \end{aligned} \quad (8)$$

The size of the ($d = 1$) code, denoted by M , not necessarily a power of two, is given by

$$M = \min(\lfloor \frac{N_1(n+q-1)}{N_1(q-1)} \rfloor, \lfloor \frac{N_1(n+q)}{N_1(q)} \rfloor). \quad (9)$$

According to d'Ocagne's identity [2], we have

$$N_1(n+q-1)N_1(q) - N_1(n+q)N_1(q-1) = (-1)^{(q-1)}N_1(n-2). \quad (10)$$

Then, we find

$$\lfloor \frac{N_1(n+q-1)}{N_1(q-1)} \rfloor \geq \lfloor \frac{N_1(n+q)}{N_1(q)} \rfloor, q \text{ odd,}$$

and

$$\lfloor \frac{N_1(n+q)}{N_1(q)} \rfloor \geq \lfloor \frac{N_1(n+q-1)}{N_1(q-1)} \rfloor, q \text{ even.}$$

So that we obtain the following simple expression for the code size

$$M = \begin{cases} \lfloor \frac{N_1(n+q)}{N_1(q)} \rfloor, & q \text{ odd,} \\ \lfloor \frac{N_1(n+q-1)}{N_1(q-1)} \rfloor, & q \text{ even.} \end{cases} \quad (11)$$

For a 2-state encoder, i.e. $r = 2$ and $q = 1$, we find

$$M = \min(N_1(n), \lfloor \frac{N_1(n+1)}{2} \rfloor) = \lfloor \frac{N_1(n+1)}{2} \rfloor. \quad (12)$$

For a 3-state encoder, i.e. $r = 3$ and $q = 2$, we simply find

$$M = \min(\lfloor \frac{N_1(n+1)}{2} \rfloor, \lfloor \frac{N_1(n+2)}{3} \rfloor) = \lfloor \frac{N_1(n+1)}{2} \rfloor, \quad (13)$$

which shows that an increase of the encoder complexity from two to three states does not increase the code size (and as we can deduce from (11) this holds for all even numbers of encoder states). As an example, Table 1 shows the code size, M , as a function of the number of encoder states, $r = N_1(q)$, for a codeword length $n = 12$.

Table 1: Code size M as a function of the number of coding states r . Code-word length $n = 12$.

q	r	$\lfloor \frac{N_1(n+q-1)}{N_1(q-1)} \rfloor$	$\lfloor \frac{N_1(n+q)}{N_1(q)} \rfloor$	M
0	1	233	377	233
1	2	377	305	305
2	3	305	329	305
3	5	329	319	319
4	8	319	323	319
5	13	323	321	321
6	21	321	322	321

We may observe that in case the number of encoder states equals $r = 13$, the encoder achieves the maximum possible code size $\lfloor 2^{(nC(d,\infty))} \rfloor = 321$ (this code could be implemented in practice). For an RLL code, a large number of surplus codewords, i.e. larger than a power of two, is very beneficial for implementing a dc-control as normally used in optical recording disc systems. The surplus words can be used as alternative channel representations, which are selected by the encoder to reduce the low-frequency components [3].

Table 1 shows the code size M as a function of the number of encoder states r , where r is a Fibonacci number. Clearly we are also interested in M when r does not equal a Fibonacci number. Table 2 shows, for $n = 12$, M as a function of $r \leq 14$. The table has been compiled using an exhaustive search for the best r_1 (and r_2) given r . It is surprising to notice that choosing r to be a Fibonacci number yields such relatively high values of the code size. For example, the smallest number of code states, r , not equal to a Fibonacci number, where the encoder can accommodate the maximum numbers of 321 codewords, equals 18 (not listed in Table 2). Computer experiments conducted with other values of n show the same behavior.

Table 2: Maximal code size M as a function of the numbers of coding states r_1 , r_2 , and r . Codeword length $n = 12$.

r_1	r_2	r	M
2	2	4	305
3	2	5	319
4	2	6	305
4	3	7	315
5	3	8	319
5	4	9	313
6	4	10	319
7	4	11	315
7	5	12	317
8	5	13	321
8	6	14	315

Case $d=2$

For the case of the $d = 2$ constraint we can derive similar equations. 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. A codeword is a binary string of length n that satisfies the $d = 2$ constraint. The set of codewords is divided into nine subsets denoted by E_{0000} , E_{0001} , E_{0010} , E_{0011} , E_{0100} etc, where the two first symbols of the subset subscript denote the first two symbols of the codeword, and the last two symbols of the subset subscript denote the last two symbols of the codeword. Thus, codewords in E_{0000} start and end with '00'; codewords in E_{0001} start with '00' and end with a '01', etc. The codewords in the various subsets are distributed over the various states of the three types such that codewords in states of the first type start with '00'; codewords in states of the second type start with '01' or '00'; and codewords in states of the third type start with '10', '01' or '00'. The state-transition rules are now easily described. Codewords that end with the string '00', i.e., codewords in subsets E_{0000} , E_{0100} , and E_{1000} 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.

The state sets of codewords are assumed to be disjoint. As a result, it is possible to assign the same codeword to different information words. Codewords that end with '00' may enter any state so that these codewords can be assigned $r = r_1 + r_2 + r_3$ times to different information words. Codewords that end with '10' may enter states of the 1st and 2nd type so that these codewords can be assigned $(r_1 + r_2)$ times to different information words. Similarly, codewords that end with a '1' can be assigned r_1 times. Given the above encoder model, it is straightforward to write down three conditions for the existence of such a rate m/n $d = 2$ code. Define the short notation

$$A_1 = r|E_{0000}| + (r_1 + r_2)|E_{0010}| + r_1|E_{0001}|,$$

$$A_2 = r|E_{0100}| + (r_1 + r_2)|E_{0110}| + r_1|E_{0101}|,$$

and

$$A_3 = r|E_{1000}| + (r_1 + r_2)|E_{1010}| + r_1|E_{1001}|.$$

Then the three conditions are

$$A_1 \geq r_1 M, \tag{14}$$

$$A_1 + A_2 \geq (r_1 + r_2) M, \tag{15}$$

$$A_1 + A_2 + A_3 \geq (r_1 + r_2 + r_3) M. \tag{16}$$

Along the same lines as in the $d = 1$ case above, we can analyze the $d = 2$ case when we assume that the number of encoder states of the 1st, 2nd, and 3rd type be chosen such that $r = r_1 + r_2 + r_3 = N_2(q)$, $r_1 + r_2 = N_2(q - 1)$, and $r_1 = N_2(q - 2)$, $q > 1$, respectively. The size of the ($d = 2$) code, M , not necessarily a power of two, equals

$$M = \min(\lfloor \frac{N_2(n + q - 2)}{N_2(q - 2)} \rfloor, \lfloor \frac{N_2(n + q - 1)}{N_2(q - 1)} \rfloor, \lfloor \frac{N_2(n + q)}{N_2(q)} \rfloor). \tag{17}$$

For a 3-state encoder, i.e. $r = 3$ and $q = 2$, we find

$$M = \lfloor \frac{N_2(n + 2)}{3} \rfloor. \tag{18}$$

As an example, Table 3 shows the code size, M , as a function of the number of code states $r = N_2(q)$ for a codeword length $n = 15$. We may observe that

at an encoder size, $r = 9$, the encoder could achieve the code size, 305. A 129-state encoder (not shown in Table 3) is the smallest encoder that achieves the maximum code size 309.

Table 3: Code size M as a function of the number of coding states r . Code-word length $n = 15$.

q	r	$\lfloor \frac{N_2(n+q-2)}{N_2(q-2)} \rfloor$	$\lfloor \frac{N_2(n+q-1)}{N_2(q-1)} \rfloor$	$\lfloor \frac{N_2(n+q)}{N_2(q)} \rfloor$	M
2	3	406	297	290	290
3	4	297	290	319	290
4	6	290	319	312	290
5	9	319	312	305	305
6	13	312	305	309	305
7	19	305	309	310	305
8	28	309	310	308	308

CONCLUSIONS

We have evaluated a relationship between the size of a certain type of sliding-block decodable RLL codes ($d = 1$ and $d = 2$) and the number of encoder states, where the number of encoder states equals a (generalized) Fibonacci number.

REFERENCES

- [1] K.A.S. Immink, J.Y. Kim, S.W. Suh, and S.K. Ahn, 'Extremely Efficient Dc-free RLL codes for Optical Recording', *IEEE Trans. Commun.*, vol COM-51, no. 3, pp. 326-331, March 2003.
- [2] S. Vajda, *Fibonacci and Lucas numbers, and the Golden Section: Theory and Applications*, Halsted Press, 1989.
- [3] K.A.S. Immink, 'EFMPlus: The Coding Format of the MultiMedia Compact Disc', *IEEE Trans. Consumer Electr.*, vol. CE-41, pp. 491-497, Aug. 1995.