A Comparison of two schemes for generating DC-free RLL Sequences

Kees A. Schouhamer Immink
Institute for Experimental
Mathematics, Ellernstrasse 29,
45326 Essen, Germany.
immink@exp-math.uni-essen.de

Wang Yong Hong Wilson
Data Storage Institute, 5
Engineering Drive 1, Singapore
117608, dsiyhw@dsi.nus.edu.sg

Abstract — **We will discuss the generation of dc-free runlength-limited (DCRLL) sequences. We propose to employ standard RLL codes, where dc-control is effectuated by multiplexing the source data or the encoded data with dc-control bits. The dc-control bits offer the degree of freedom required to shape to spectrum. It will be shown that a new technique, called *parity preserving assignment,* will offer great benefits over other constructions.**

I. Introduction

The design of dc-free runlength-limited (DCRLL) codes can, at least in principle, be systematically accomplished by the many design techniques published [1]. Unfortunately, the design of a DCRLL code with a rate close to the Shannon capacity of the constrained channel, is severely hampered by the large number of states of the finite-state machine which models the channel constraints at hand. The large number of states of the underlying FSM, can, at least in principle, be handled by buying a larger computer, but the insight required is too easily lost. Essentially, there are two systematic design approaches that emerged in the literature.

The first method uses a standard method, such as the ACH algorithm to design an RLL code. In the final stage of the ACH algorithm we end up with a graph with the property that from any state of the graph there are at least 2^m (m is assumed to be the source word length) outgoing edges. These *surplus edges* are used as alternative codewords that can be used for dc-control. The rate 8/16, (2,10) EFMPlus code is an example of a DCRLL code used in practice (DVD) that was designed according to these guidelines [1].

In the second method, a given, state-of-the-art, RLL code, is used to generate RLL sequences. The sequences generated under the coding rules of said code are multiplexed with dc-control bits for minimizing the low-frequency components. The user data or alternatively the encoded data are partitioned into segments of ν bits. Between two consecutive ν-bit segments β dc-control bits are inserted, and the β dc-control bits, in turn, are chosen to minimize the low-frequency components.

II. Codes with parity preserving word assignment

In order to make it possible to efficiently control the dc-content in the source date level mode, we have made the assignment between source words and codewords in such a way that the *parity* of both source word and its assigned codeword are the same. The parity, P, of an n-bit word $(x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$, (either source or codewords) is defined by

$$P = \sum_{i=1}^{n} x_i \bmod 2.$$

In other words, if the source word has an even (or odd) number of 'one's then its channel representation also has an even (or odd) number of 'one's. A code with a *parity preserving* assignment has the virtue that when it is used in conjunction with dc-control bits at data level that setting an even (or odd) number of 'one's at data level will result in an even (or odd) number of 'one's at code level. This leads, as we will demonstrate, to an efficient dc-control.

The variable length rate 2/3, $(1, \infty)$ code shown in Table 1, is an example of a code with the parity preserving property. It can easily be verified that indeed the assignment is parity preserving. In the presentation, we will show the difference in performance between various DCRLL codes.

Table 1: Variable-length synchronous rate 2/3, $(1, \infty)$ code with parity preserving assignment.

Data		*Code*
00	$\leftarrow \rightarrow$	000
01	$\leftarrow \rightarrow$	010
10	$\leftarrow \rightarrow$	100
1100	$\leftarrow \rightarrow$	001010
1101	$\leftarrow \rightarrow$	001000
1110	$\leftarrow \rightarrow$	101010
1111	$\leftarrow \rightarrow$	101000

References

[1] K.A.S. Immink, *Codes for Mass Data Storage Systems,* Shannon Foundation Publishers, Eindhoven, The Netherlands, 1999.

0-7803-5857-0/00/$10.00 ©2000 IEEE.