

# Construction of Maximum Run-Length Limited Codes using Sequence Replacement Techniques

Adriaan J. van Wijngaarden, *Senior Member, IEEE*, and Kees A. Schouhamer Immink, *Fellow, IEEE*

**Abstract**—Several algorithms for the construction of maximum run-length limited codes are presented that are based on the sequence replacement technique. This technique effectively converts an input sequence into a constrained sequence in which a prescribed subsequence is forbidden to occur. The proposed algorithms show how all forbidden subsequences can be successively or iteratively removed to obtain a constrained sequence and how special subsequences can be inserted at predefined positions in the constrained sequence to represent the indices of the positions where the forbidden subsequences were removed. Several enhancements for providing effective error control are presented as well. The proposed algorithms prove to be very efficient and the rates of the constructed codes are close to their theoretical maximum. The proposed algorithms prove to be very efficient and are thus of practical importance for use in storage systems and data networks.

**Index Terms**—modulation coding, run length codes, block codes, variable length codes, error analysis, error correction coding.

## I. INTRODUCTION

MANY digital communication and storage systems require the use of codes that ensure the exclusion of certain prescribed sequences from a stream of symbols that is modulated and transmitted [1]. The exclusion of a prescribed sequence is required in situations where this particular sequence is reserved for specific purposes, or in situations where certain sequences cause excessive errors or lead to loss of synchronization. This problem has been studied extensively, in particular in the context of *modulation codes*.

Modulation codes are required in many communication systems, in particular in magnetic and optical recording systems and data networks. An important class of modulation codes are maximum run-length limited (RLL) codes, where long runs of consecutive like symbols are excluded. These codes are equivalent to  $(0, k)$  codes in NRZI notation [1]. A  $(0, k)$  code is a set of binary sequences (codewords) with the property that at least 0 and at most  $k$  “zeros” occur between two consecutive “ones”. In NRZI, a “one” corresponds to a reversal and a “zero” to a non-reversal of the polarization in a binary differential signalling scheme. The parameter  $k$  defines the maximum distance between transitions and therefore determines the self-clocking properties. Long sequences that fulfill  $(0, k)$  constraints are typically constructed by concatenating the codewords of a  $(0, k, k_l, k_r)$  code, where  $k_l$  and  $k_r$  denote the maximum number of leading and trailing zeros of a codeword ( $k_l + k_r \leq k$ ). In practical systems one often finds

A. J. (de Lind) van Wijngaarden is with the Mathematics of Networks and Communications Department, Bell Laboratories, Alcatel-Lucent, Murray Hill, NJ. E-mail: alw@research.bell-labs.com.

K. A. Schouhamer Immink is with Turing Machines, Inc., The Netherlands, and is affiliated with the Inst. for Experimental Mathematics, Essen, Germany, as well as the Data Storage Institute, Singapore.

$(0, k, k_l, k_r)$  codes that are encoded and decoded using look-up tables or combinatorial circuitry [1]–[6]. The rates are typically low, e.g.,  $8/9$  and sometimes  $16/17$ . Several schemes have been proposed to construct higher rate codes [7]–[9], but their introduction in practical applications is hampered due to coding complexity, latency, and the risk of error propagation.

We apply the *sequence replacement technique*, which successively removes all forbidden subsequences to obtain a constrained sequence and inserts special subsequences at predefined positions in the sequence to represent the indices of the positions where the forbidden subsequences were removed [10], [11]. In Section II, we discuss the main concept and characteristics of the sequence replacement technique. In Section III we propose three methods to construct high rate  $(0, k)$  codes by replacing runs of more than  $k$  zeros using the sequence replacement technique. The efficiency of the codes is significantly higher than for the codes currently used in recording systems. In Section IV, we analyse the effects of transmission errors on the decoding process and discuss options to further reduce error propagation and provide error control.

## II. THE SEQUENCE REPLACEMENT TECHNIQUE

To describe the properties of the sequence replacement technique we need to introduce some notation. Let  $X = x_1x_2 \cdots x_w \in \mathbb{Z}_2^w$  represent a binary string of length  $w$ , where  $\mathbb{Z}_2$  denotes the binary alphabet. The concatenation of two sequences  $X$  and  $Y$  is denoted by  $XY$ , and a set of sequences of length  $p+w$  that have a common prefix  $P \in \mathbb{Z}_2^p$  is denoted by  $PS^{(w)} = \{PY | Y \in \mathcal{S}^{(w)}\}$ , where  $\mathcal{S}^{(w)} \subseteq \mathbb{Z}_2^w$ . Similarly,  $\mathcal{S}^{(w)}P = \{YP | Y \in \mathcal{S}^{(w)}\}$ . The null string, denoted by  $\Lambda$ , represents a string of length 0 for which  $\Lambda X = X \Lambda = X$ . A run of  $w$  consecutive symbols  $a$  is written as  $a^w$ . The character  $*$  denotes an arbitrary symbol of  $\mathbb{Z}_2$ , and the sequence  $(*)^w$  denotes an arbitrary element of  $\mathbb{Z}_2^w$ . We use the notation  $\bar{a}$  to represent the inverse of  $a$ .

*Definition 1:* Let  $\mathcal{F}_Q^{(m)}$  denote the set of sequences of length  $m$  where sequence  $Q$  of length  $h$  does not appear at any position as a subsequence of  $h$  consecutive symbols.

The cardinality  $F_Q^{(m)} = |\mathcal{F}_Q^{(m)}|$  is, according to Theorem 1 in [12], given by the recursion relation

$$F_Q^{(m)} = \begin{cases} 2^m & \text{for } m < h, \\ \sum_{i=1}^h (2b_i - b_{i+1}) F_Q^{(m-i)} & \text{for } m \geq h, \end{cases} \quad (1)$$

where  $b_1 = 1$  and  $b_{h+1} = 1$ . For  $2 \leq s \leq h$ ,  $b_s = 1$  if the  $s$ -symbol prefix of  $Q$  equals the  $s$ -symbol suffix of  $Q$ , i.e., if  $q_1 \cdots q_{h-s+1} = q_s \cdots q_h$ , and  $b_s = 0$  otherwise.

The set of sequences that satisfy the  $(0, k)$  constraint, i.e., the set of sequences where  $Q = 0^{k+1}$  does not occur, is denoted by  $\mathcal{G}_k^{(m)} = \mathcal{F}_Q^{(m)}$ , and its cardinality is given by

$$G_k^{(m)} = \begin{cases} 2^m & \text{for } m < k + 1, \\ 2G_k^{(m-1)} - G_k^{(m-k-1)} & \text{for } m \geq k + 1. \end{cases} \quad (2)$$

The *redundancy* (in bits) of a code  $\mathcal{C}^{(n)}$  of length  $n$  is defined as the quantity  $R = n - \log_2(|\mathcal{C}^{(n)}|)$ . Consequently, the rate of the code is given by  $1 - R/n$ .

We consider the problem of converting an arbitrary sequence  $X \in \mathbb{Z}_2^m$  into a sequence  $Y \in \mathcal{C}_Q^{(n)} \subseteq \mathcal{F}_Q^{(n)}$ . Consequently,  $2^m \leq |\mathcal{C}_Q^{(n)}| \leq F_Q^{(n)}$ . The objective is to provide a mapping  $\zeta_Q : \mathbb{Z}_2^m \rightarrow \mathbb{Z}_2^n$ , such that the redundancy  $R = n - m$  is minimized and the translation of  $X \in \mathbb{Z}_2^m$  into  $\zeta_Q(X) \in \mathcal{C}_Q^{(n)} \subseteq \mathcal{F}_Q^{(n)}$  and vice versa is efficient and has a low computational complexity. In addition, we aim to make the mapping  $\zeta_Q$  resilient for transmission errors. Transmission errors are said to cause *error propagation* if the Hamming distance  $d_H(Y, \hat{Y})$  between the transmitted codeword  $Y = \zeta_Q(X)$  and the received word  $\hat{Y}$ , i.e., the number of transmission errors, is much smaller than  $d_H(X, \hat{X})$ , where  $\hat{X} = \zeta_Q^{-1}(\hat{Y})$  is the decoded received word.

The *concept* of the sequence replacement technique is to successively remove all forbidden subsequences and to represent their positions by special subsequences which are inserted at predefined positions of the constrained sequence. This concept is best explained with three example source words  $X_A$ ,  $X_B$  and  $X_C$  of length  $n - 1$ , where 0, 1, and 3 forbidden subsequences  $Q$  occur in the source word; their conversion into the codewords  $Y_A$ ,  $Y_B$ ,  $Y_C \in \mathcal{F}_Q^{(n)}$  of length  $n$  is illustrated in Fig. 1. The source word  $X_A = A_1 \in \mathcal{F}_Q^{(n-1)}$

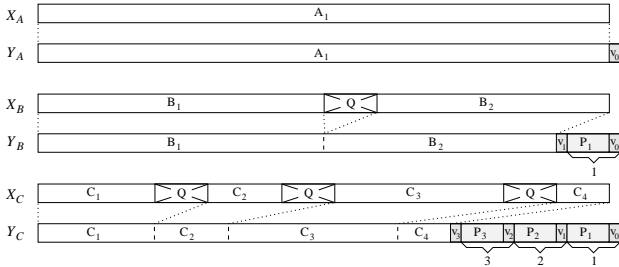


Fig. 1. The conversion of three source words  $X_A$ ,  $X_B$  and  $X_C$  of length  $n - 1$  into codewords  $Y_A$ ,  $Y_B$ ,  $Y_C \in \mathcal{F}_Q^{(n)}$  using the sequence replacement technique for the case where 0, 1, and 3 forbidden subsequences  $Q$  occur in the source word. The segments marked with ‘ $v_i$ ’ are used to indicate the number of replacements and the segments marked with ‘ $P_i$ ’ and ‘ $v_i$ ’ uniquely identify the positions where the  $i$ th forbidden sequence was removed.

does not need to be converted, and therefore the first  $n - 1$  positions of the codeword  $Y_A = \zeta_Q(X_A) = A_1 v_0 \in \mathcal{F}_Q^{(n)}$  are identical to the source word and the rightmost position ‘ $v_0$ ’ is used to indicate that the source word has remained unchanged.

We now consider the situation where  $Q$  is detected one or several times in the source word. In this case ‘ $v_0$ ’ indicates that at least one replacement was required to obtain the codeword. Consider the source word  $X_B = B_1 Q B_2$ , where  $Q$  occurs once at position  $t_1$  and  $B_1 \in \mathcal{F}_Q^{(t_1-1)}$  and  $B_2 \in \mathcal{F}_Q^{(n-t_1-q)}$ .

The subsequence  $Q$  of length  $q$  is removed to satisfy the constraints. If  $B_1 B_2 \in \mathcal{F}_Q^{(n-q-1)}$ , no further replacements are necessary. We now use ‘ $v_0$ ’ to indicate that  $Q$  was replaced at least once. The codeword  $Y_B = \zeta_Q(X_B) = B_1 B_2 v_1 P_1 v_0$  specifies the index  $t_1$  using the tuple  $(v_1, P_1)$  of length  $q$ , and uses ‘ $v_1$ ’ to indicate that not more than one sequence was replaced. The tuple  $(v_1, P_1)$  can represent at most  $2^{q-1}$  indices, because one bit value is used to let  $v_1$  indicate the number of replacements. Since  $1 \leq t_1 \leq n - q$ , The wordlength  $n$  is not to exceed a maximum value  $n_{\max}$ , since the index  $t_1$  in the range from one to  $n - q$  is to be represented unambiguously.

The sequence replacement technique is applied  $s$  times on the (shortened) source word until all forbidden subsequences have been removed. At the  $i$ th transformation the forbidden sequence at a position  $t_i$  is removed and position  $t_i$  is represented by  $(v_i, P_i)$ , which occupy the same number of symbols as the sequence  $Q$ . The codeword length therefore remains unchanged irrespective of the number of replacements. The symbols  $v_1, \dots, v_s$  indicate that the codeword was obtained with exactly  $s$  replacements. Figure 1 illustrates the transformation of source word  $X_C$  into the codeword  $Y_C$ , which requires three replacements.

To represent the indices of the positions, we use the natural lexicographical order of a set of sequences of fixed length. Any set  $\mathcal{S}^{(m)} \subseteq \mathbb{Z}_2^m$  can be ordered lexicographically by sorting the elements of  $\mathcal{S}^{(m)}$  in the following way [13]. The sequence  $X = x_1 x_2 \dots x_m \in \mathcal{S}^{(m)}$  is smaller than  $Y = y_1 y_2 \dots y_m \in \mathcal{S}^{(m)}$ , in short  $X < Y$ , if and only if there exists an index  $j$ ,  $1 \leq j \leq m$ , such that  $x_i = y_i$  for any  $1 \leq i < j$ , and  $x_j < y_j$ . Let  $\mathcal{N}_X^{(m)} = \{Z \in \mathbb{Z}_2^m | Z < X\}$ . The cardinality  $|\mathcal{N}_X^{(m)}|$  of this set  $\mathcal{N}_X^{(m)}$  is given by

$$|\mathcal{N}_X^{(m)}| = \sum_{i=1}^m x_i 2^{m-i}. \quad (3)$$

The function  $\text{idx}(\mathcal{S}^{(m)}, X)$  denotes the cardinality of the set  $\mathcal{N}_X \cap \mathcal{S}^{(m)}$  and thus corresponds to the position of  $X \in \mathcal{S}^{(m)}$  in the lexicographically ordered set  $\mathcal{S}^{(m)}$ . The function  $\text{idx}(\mathcal{S}^{(m)}, X)$  is therefore called the *index function* of the set  $\mathcal{S}^{(m)}$ . The inverse operation  $\text{idx}^{-1}(\mathcal{S}^{(m)}, i)$  returns the  $i$ th element of the lexicographically ordered set  $\mathcal{S}^{(m)}$ .

In the next section several constructions will be proposed. Each construction is specified by a mapping based on which the encoding and decoding are detailed. The maximum wordlength  $n_{\max}$  and the maximum number of replacements  $s_{\max}$  will be determined.

### III. CONSTRUCTION OF MAXIMUM RLL CODES

In this section we employ the sequence replacement technique to construct  $(0, k)$  codes and  $(0, k, k_l, k_r)$  codes. The main objective is to develop methods to transform a binary sequence of length  $m$  into a constrained sequence of length  $n = m + 1$  where the maximum number of consecutive zeros in the sequence is  $k$ , i.e., sequence  $0^{k+1}$  is not allowed to occur as a subsequence. A second objective is to minimize the  $k$  constraint for the given word length  $n$ .

*Method A. One-pass  $(0, k)$  code construction*

The first method to construct a  $(0, k)$  code of length  $n$  is specified by the mapping  $\xi_k : \mathbb{Z}_2^{n-1} \rightarrow \mathbb{Z}_2^n$ , for each  $X \in \mathbb{Z}_2^{n-1}$  defined by

$$\xi_k(X) = \begin{cases} X1 & \text{if } X1 \in \mathcal{G}_k^{(n)} \\ Z1A_110A_210\cdots A_s10 & \text{otherwise} \end{cases}$$

where  $s$  denotes the number of disjoint occurrences of subsequence  $0^{k+1}$  in  $X$ , and each  $t_i$  is the index of the first position of the  $i$ th subsequence  $0^{k+1}$  in  $X$ , encountered by scanning  $X$  from left to right. Each sequence  $A_i$  of length  $k-1$  is the binary representation of the number  $(t_i - 1)$  in reversed order, and  $Z$  is the sequence obtained after all  $s$  subsequences  $0^{k+1}$  in  $X$  have been removed. Note that  $1 \leq s \leq \lfloor (n-1)/(k+1) \rfloor$ .

Let  $Y = y_1 \dots y_n = \xi_k(X)$ . A property of the construction that can easily be verified is  $y_{n-i(k+1)} = 0$  for  $1 \leq i < s$ , and  $y_{n-s(k+1)} = 1$ . This property is used by the decoder to determine the number,  $s$ , of sequence replacements that were made by the encoder. Once  $s$  has been retrieved the  $s$  indices can be recovered, and the sequence  $X$  is reconstructed by inserting  $0^{k+1}$  at the specified positions.

*Theorem 1:* For  $n \leq 2^{k-1} + k + 1$  and  $k > 1$ , a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code can be constructed using mapping  $\xi_k$ .

*Proof:* If  $X1 \in \mathcal{G}_k^{(n)}$ , the  $(0, k)$  constraint is fulfilled. Otherwise,  $X1 \notin \mathcal{G}_k^{(n)}$ , and in this case  $X$  is uniquely represented by  $X = Z_0 0^{k+1} Z_1 0^{k+1} \dots Z_{s-1} 0^{k+1} Z_s$ , with  $1 \leq s \leq \frac{n-1}{k+1}$ , and  $Z_i \in \mathcal{G}_k^{(|Z_i|)}$  for  $0 \leq i \leq s$ . Sequence  $Z = Z_0 Z_1 \dots Z_s$  is an element of  $\mathcal{G}_k^{(n-s(k+1)-1)}$ , since the ones act as a delimiter between non-empty segments  $Z_i$ . The indices  $t_1$  up to  $t_s$  are uniquely represented by  $1A_110\cdots A_s1$ , and since each  $A_i \in \mathbb{Z}_2^{k-1}$  is on both sides separated by a ‘‘one’’, the transformed source word  $Z1A_110\cdots A_s10$  is a codeword of a  $(0, k)$  code. Sequence  $A_i$  allows the representation of  $2^{k-1}$  different values of  $t_i$ , and since  $1 \leq t_i \leq n - k - 1$ , we obtain the condition  $n \leq 2^{k-1} + k + 1$ .  $\square$

By slightly modifying  $\xi_k$  to obtain  $\xi'_k$ , it is possible to construct concatenatable rate  $\frac{n-1}{n}$ ,  $(0, k, k-1, 1)$  codes. Special attention has to be paid to ensure that the condition  $k_l = k-1$  is fulfilled. If a sequence  $0^k$  occurs at the first position of the source word, this sequence is removed, and this violation of the constraint is identified by  $A_i = 0^{k-2}$ , which is one bit shorter than usual. If  $t_i \neq 1$ ,  $A_i$  of length  $(k-1)$  equals the binary representation in reversed order of number  $t_i$ . The decoder can easily identify this difference in length, since the first  $(k-2)$  positions of  $A_1$  are zeros, and for  $t_i \neq 1$ , at least one of these positions is non-zero.

*Corollary 1:* For  $n \leq 2^{k-1} + k$  and  $k \geq 2$ , a rate  $\frac{n-1}{n}$ ,  $(0, k, k-1, 1)$  code can be constructed using mapping  $\xi'_k$ .

*Example 1:* We use method A to transform source word  $X = 00000100000011000000 \in \mathbb{Z}_2^{20}$  into a codeword of a rate  $\frac{20}{21}$ ,  $(0, 5)$  code. Subsequence  $0^6$  occurs at positions  $t_1 = 7$  and  $t_2 = 15$ , and therefore  $s = 2$ ,  $A_1 = 0110$ ,  $A_2 = 0111$  and  $Z = 00000111$ . We obtain codeword  $\xi_k(X) = 000001111011010011110$ . Similarly, a rate  $\frac{20}{21}$ ,  $(0, 5, 4, 1)$  code can be constructed with the aid of mapping  $\xi'_k(X)$ . With  $t_1 = 1$ ,  $t_2 = 7$ , and  $t_3 = 15$ ,  $s = 3$ ,

$A_1 = 000$ ,  $A_2 = 1110$ , and  $A_3 = 1111$ . Codeword  $\xi'_k(X) = 111100010111010111110$ .

*Method B. Enhanced one-pass  $(0, k)$  code construction*

The second construction is described by mapping  $\mu_k : \mathbb{Z}_2^{n-1} \rightarrow \mathbb{Z}_2^n$ , for each  $X \in \mathbb{Z}_2^{n-1}$  defined by

$$\mu_k(X) = \begin{cases} X1 & \text{if } X1 \in \mathcal{G}_k^{(n)} \\ Z'_s1A_110A_210\cdots A_s10 & \text{otherwise} \end{cases}$$

where  $s$  denotes the number of steps required to convert  $Z_0 = X1$  into sequence  $Z_s$ , with  $Z_s = Z'_s1 \in \mathcal{G}_k^{(|Z_s|)}$ , and  $A_1 \dots A_s$ . Every sequence  $A_i$  is an element of  $\mathbb{Z}_2^k \setminus \{0^k, 1^{k-1*}\} \cup \{1^{k-1}\}$ .

Sequence  $Z_0 = X1$  is scanned from left to right for the presence of subsequence  $0^{k+1}$ . If  $X1 \in \mathcal{G}_k^{(n)}$ ,  $\mu_k(X) = X1$ . Otherwise,  $0^{k+1}$  occurs at position  $t_1$ , and  $Z_0$  has the form  $D_0 0^{k+1} 1 H_0$ , where  $D_0 \in \mathcal{F}_{0^{k+1}}^{(t_1-1)}$ , and  $H_0 \in \mathbb{Z}_2^{n-t_1-k-1}$ . If  $0^{k+1}$  occurs at the end of  $Z_0$ ,  $t_1 = e = |Z_0| - k + 1$ ,  $H_0 = \Lambda$ , and we remove  $0^{k+1}$  at position  $t_1$  to obtain  $Z_1 = D_01$ . Position  $t_1 = e$  is represented by  $A_1 = 1^{k-1}$ . Otherwise,  $1 \leq t_1 < e$ , and we remove  $0^{k+1}$  at  $t_1$  from  $Z_0$  to obtain  $Z_1 = D_0 H_0$ . Sequence  $A_1$  of length  $k$  is the binary representation in reversed order of  $t_1$ , which means that the less significant bits are positioned on the left-hand side of sequence  $A_1$ . The scanning procedure now continues, taking into account that the first  $t_1 - k - 1$  positions of  $Z_1$  do not contain  $0^{k+1}$ . After  $1 \leq s \leq \lfloor n/(k+1) \rfloor$  transformations,  $Z_s = Z'_s1 \in \mathcal{G}_k^{(|Z_s|)}$ , as well as  $A_1 \dots A_s$  have been determined, and  $\mu_k(X) = Z'_s1A_110A_210\cdots A_s10$ . This encoding procedure is illustrated by the following example.

*Example 2:* Let  $X = 000001011100100000 \in \mathbb{Z}_2^{18}$ , and  $k = 4$ . Since  $0^{k+1}$  occurs at position  $t_1 = 1$ , sequence  $Z_0 = X1 \notin \mathcal{G}_k^{(m+1)}$ , and therefore it is converted into  $Z_1 = 0111001000001$ , which, because of the occurrence of  $0^{k+1}$  at the end of the sequence, is converted into  $Z_2 = 01110011 \in \mathcal{G}_k^{(8)}$ . With  $s = 2$ ,  $A_1 = 1000$ , and  $A_2 = 111$ , we obtain  $\mu_k(X) = 0111001110001011110$ .

*Theorem 2:* For  $n \leq 2^k + k - 1$  and  $k \geq 2$ , a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code can be constructed using mapping  $\mu_k$ .

*Proof:* If  $X1 \in \mathcal{G}_k^{(n)}$ ,  $\mu_k(X) = X1$  is a codeword of a  $(0, k)$  code. Otherwise,  $X1 \notin \mathcal{G}_k^{(n)}$ , and the conditions imposed on  $Z'_s$  and  $A_1 \dots A_s$  imply that sequence  $\mu_k(X) = Z'_s1A_110A_210\cdots A_s10$  is also a codeword of a  $(0, k)$  code.

The subsequence  $0^{k+1}$  can occur at  $n - k - 1$  different positions in  $X1$ , and since there are  $2^k - 2$  different sequences  $A_i$  to represent  $t_i$ , condition  $n \leq 2^k + k - 1$  is obtained.  $\square$

By modifying mapping  $\mu_k$  to obtain  $\mu'_k$ , rate  $\frac{n-1}{n}$ ,  $(0, k, k-1, 1)$  codes can be constructed. If sequence  $Z_i$  starts with  $0^{k-1}$ , this subsequence is removed to ensure that the  $k_l = k-1$  constraint is fulfilled. The corresponding index  $t_i = 1$  is represented by  $A_i = 0^{k-1}$ , and for  $t_i = e$ ,  $A_i = 1^{k-1}$ . All other  $A_i \in \mathbb{Z}_2^k \setminus \{0^{k-1*}, 1^{k-1*}\}$  can be used to represent index  $2 \leq t_i < e$ , and every  $t_i$  can be uniquely identified when  $A_i$  corresponds to the binary representation of the number  $(t_i + 1)$ .

There are  $2^k - 2$  different sequences  $A_i$ , and there are  $n - k$  different indices. Therefore  $n \leq 2^k + k - 2$ .

*Corollary 2:* For  $n \leq 2^k + k - 2$  and  $k \geq 2$ , a rate  $\frac{n-1}{n}$ ,  $(0, k, k-1, 1)$  code can be constructed using mapping  $\mu'_k$ .

### Method C. Quasi-optimum $(0, k)$ code construction

This method aims to treat the segment that represents the index of a forbidden subsequence as if it is a part of the source word in the next step. One difficulty is the unique representation of the number of transformation steps  $s$ , since for every replacement of a forbidden sequence of length  $q$  only one bit will be used to indicate the additional transformation, and the other  $(q-1)$  positions will be used to represent the position of the forbidden sequence. Therefore the technique used in methods A and B cannot be applied here. For that reason we will use sequence  $10^s$  to represent  $s$  transformations, and replace part of this sequence as soon as  $s > k$ . The proposed method is described by mapping  $\lambda_k : \mathbb{Z}_2^{n-1} \rightarrow \mathbb{Z}_2^n$ , for each  $X \in \mathbb{Z}_2^{n-1}$  defined by

$$\lambda_k(X) = \begin{cases} X1 & \text{if } X1 \in \mathcal{G}_k^{(n)} \\ Z'_s A_s 1 W'_s 0 & \text{otherwise} \end{cases}$$

where  $s$  denotes the number of steps required to convert sequence  $Y_0 = X1$  into sequence  $Z'_s A_s 1 \in \mathcal{G}_k^{(n-s)}$ . Sequence  $A_s$  represents  $t_s$ , the position of the forbidden subsequence in  $Z_{s-1}$ , and sequence  $W_s = W'_s 0$  uniquely represents  $s$ . If  $s \leq k$ ,  $W_s = 0^s$ , and otherwise sequence  $W_s = 0^{s-v} (1^k)^v$  ( $1^k 0^v$ , where  $v = \lfloor s/(k+1) \rfloor$ ).

Since  $1W_{k+1} = 1^{k+1}0$  and for  $s > k+1$ , sequence  $W_s = W'_{s-k-1} 01^k 0$ , sequence  $*1^k$  and sequence  $*01^{k-1}$  are excluded from the set of valid sequences  $A_i$  to be able to distinguish between  $A_i 10$  and  $W_s$ .

Each position  $t_i$  is represented by  $A_i$ , the binary representation of number  $(t_i + 3)$ , and, if  $t_i = e = |Z_i| - k + 1$ ,  $A_i = 10^k$ .

*Theorem 3:* For  $n \leq 2^{k+1} + k - 4$ , a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code can be constructed using mapping  $\lambda_k$ .

*Proof:* If  $X1 \in \mathcal{G}_k^{(n)}$ , it follows that  $\lambda_k(X) = X1$  fulfills the  $(0, k)$  constraints. Otherwise,  $X$  is converted into  $\lambda_k(X) = Z'_s A_s 1 W'_s 0$ . Since  $Z'_s A_s \in \mathcal{G}_k^{(n-s-1)}$  and  $W_s \in \mathcal{G}_k^{(s)}$ ,  $Z'_s A_s 1 W'_s 0 \in \mathcal{G}_k^{(n)}$  is a valid codeword of a  $(0, k)$  code.

Each index  $t_i$  is identified by a sequence  $A_i$ , where  $A_i \in \mathbb{Z}_2^{k+1} \setminus \{ *10^{k-1}, *01^{k-1}, *1^k \} \cup \{ 10^{k-1} \}$ . Since a forbidden sequence can occur at  $n - k - 1$  different positions, and there are  $2^{k+1} - 5$  sequences  $A_i$ , we obtain  $n \leq 2^{k+1} + k - 4$ .  $\square$

*Example 3:* Let  $X = 000000000101100000000100 \in \mathbb{Z}_2^{24}$ , and  $k = 4$ . Source word  $X$  is in  $s = 7$  steps converted into a  $(0, 4)$  code of length 25:

$i$	$Z_i A_i 1 W_i$	$t_{i+1}$	$A_{i+1}$
0	0000000001011000000001001	5	00010
1	0000011000000001000001010	1	00100
2	1000000001000001000100100	5	00010
3	1000000001000100000101000	5	00010
4	1000000100000100001010000	3	01100
5	1000000100001001100111110	3	01100
6	1000001001100011001011110	2	10100
7	1001100011001010010011110	-	—

We thus obtain  $\lambda_k(X) = 1001100011001010010011110$ , which is obviously an element of a rate  $\frac{24}{25}$ ,  $(0, 4)$  code.

The suffix of the codeword changes for  $i = 5$ . In this way the suffix will not violate the imposed  $k$  constraint. Inspection shows that there are only 55 out of  $2^{24}$  sequences for which  $s = 7$  transformations are needed, and therefore probability  $\Pr[s=7] = \frac{55}{2^{24}} \approx 3.3 \cdot 10^{-6}$ . Since only very few sequences require a lot of transformations, it may be appropriate to develop an alternative concurrent detection and coding scheme for these sequences.

Rate  $\frac{n-1}{n}$ ,  $(0, k, k_l, k_r)$  codes can be constructed with the modified mapping  $\lambda'_k$ . It is not possible to construct a  $(0, k, k-1, 1)$  code using method C, because of the right-hand constraint  $k_r > 1$ . This means that the value of constraint  $k_l = k - k_r$  has to be decreased. We now will have to indicate both the position of the replacement and the kind of sequence that has been removed. It is required to uniquely represent a replacement of sequence  $0^{k+1}1$  on one out of  $n - k - 1$  positions, or a replacement of a sequence  $0^{k_i+1}1$ ,  $1 \leq i \leq (k - k_l)$  violating the  $k_l$ -constraint, or one replacement of sequence  $0^{k+1}$ , or the resolution of the  $k_r$ -constraint by transforming  $q - k_r + 1$  positions in the suffix  $W_s$  of the codeword.

The Kraft inequality [14] states that the necessary and sufficient condition for the existence of a binary code with codewords having lengths  $q_1 \leq q_2 \leq \dots \leq q_L$  that satisfy the prefix condition is

$$\sum_{i=1}^L 2^{q-q_i} \leq 2^q.$$

There are  $(n - k - 2)$  indices which have a representation of  $q = k + 1$  symbols, one representation of length  $(k_l + i)$  is needed for each of the  $1 \leq i \leq (k - k_l)$  violations of the  $k_l$ -constraint, one of length  $k$  for the overlap with the position indicator, and there is one sequence of length  $(k_r - 1)$ . We now obtain the inequality

$$\sum_{i=1}^{k-k_l} 2^{(k+1)-(k_l+i)} + 2^1 + 2^{k-k_r+2} + (n-k-2) \leq 2^{k+1}$$

By rewriting this inequality as  $2^{k-k_l+1} + 2^{k-k_r+2} + (n-k-2) \leq 2^{k+1}$ , we obtain the inequality  $n \leq 2^{k+1} - 2^{k-k_l+1} - 2^{k-k_r+2} + k + 2$ . The right-hand side is maximized if  $k_r = \lceil k/2 \rceil$ , and  $k_l = k - k_r$ . The sequences  $A_i$  can be associated to  $t_i$  such that  $A_i$  corresponds to the binary representation of the index  $t_i$ .

*Corollary 3:* For  $n \leq 2^{k+1} + k - 2^{k-k_l+1} - 2^{k-k_r+2} + 2$  and  $k \geq 3$ , a rate  $\frac{n-1}{n}$ ,  $(0, k, k_l, k_r)$  code can be constructed with mapping  $\lambda'_k$ .

Table I lists the maximum value of  $n$  for which a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code exists, denoted by  $n_{\max}$ , as well as the maximum values of  $n$  for which a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code can be constructed by using methods A, B or C. This table shows that RLL codes with very high rates of more than 99% can be constructed for relevant values of  $k$ . Table I also shows the corresponding values for rate  $\frac{n-1}{n}$ , concatenatable  $(0, k, k_l, k_r)$  codes. It shows that the constructed codes are very efficient in terms of redundancy. The proposed methods enable the construction of (concatenatable) constrained codes for arbitrary word lengths. The complexity is almost independent of the word length, and the efficiency is very high.

TABLE I

MAXIMUM LENGTHS  $n$  FOR WHICH A RATE  $\frac{n-1}{n}$ ,  $(0, k)$  CODE AND A RATE  $\frac{n-1}{n}$ , CONCATENATABLE  $(0, k)$  CODE CAN BE CONSTRUCTED USING METHODS A, B, C.

$k$	$(0, k)$ code				concat. $(0, k)$ code			
	$n_{\max}$	$n_A$	$n_B$	$n_C$	$n_{\max}$	$n_{A'}$	$n_{B'}$	$n_{C'}$
2	9	5	5	6	5	4	4	-
3	21	8	10	15	12	7	9	5
4	43	13	19	32	31	12	18	14
5	88	22	36	65	67	21	35	39
6	177	39	69	130	148	38	68	88
7	355	72	134	259	310	71	133	201
8	710	137	263	516	649	136	262	426
9	1420	266	520	1029	1327	265	519	907
10	2840	523	1033	2054	2715	522	1032	1868
11	5679	1036	2058	4103	5490	1035	2057	3853
12	11358	2061	4107	8200	11105	2060	4106	7822

#### IV. ERROR CONTROL ASPECTS

Having presented several new construction methods based on sequence replacement techniques in the previous sections, we now consider their susceptibility to transmission errors. Errors are detectable if the constraints imposed on the codeword are violated, or when the decoder detects an inconsistency, for instance an index being out of range. Otherwise, the erroneous codeword will be decoded and the errors may propagate. We will analyse this situation and propose error control schemes which reduce or avoid error propagation.

##### A. Error Propagation

Constructions based on the sequence replacement technique are to some extent susceptible to error propagation. We will analyse method B; the effects are similar for the other methods. Consider a rate  $\frac{n-1}{n}$ ,  $(0, k)$  code, and let source word  $X$  be of the form  $D0^{k+1}1H$ , where  $D \in \mathcal{F}_{0^{k+1}}^{(t_1-1)}$ , and  $H \in \mathbb{Z}_2^{n-t_1-k-2}$ . Source word  $X$  is converted into  $Z = DH1A_110$ . This transformation is depicted in Fig. 2. Segment  $D$  remains at the same position, but segment  $H$  is shifted  $k+2$  positions to the left. If an error occurs in  $A_1$ , sequence  $X' = DH10^{k+1}1H_2$  will be formed, i.e.,  $H_1$  will not be shifted backwards to its original position. This causes a burst error of length  $|H_1| + k + 2 \leq 2^{k-1} + k + 2$ . The position

and length of the burst due to an error in a high-significant position of the index representation segment  $A_1$  is indicated in Fig. 2 by a long, thin dark gray bar.

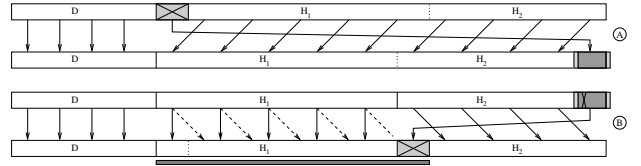


Fig. 2. Transformation of a source word into a constrained codeword (A) and the inverse transformation in the presence of an error in the index representation part (B). The dark gray bar indicates the position and maximum length of the burst error.

Shifts of segments can be avoided by slightly modifying the construction methods. Consider the same source word  $X = D0^{k+1}1H = D0^{k+1}1H_1H_2$ , where  $|H_1| = (n - t_1 - k - 2) - (k + 2)$  and  $|H_2| = k + 2$ . We replace the forbidden sequence  $0^{k+1}$  by  $H_2$ , and in doing so, we prevent the shifting of sequence  $H_1$ . The resulting codeword is  $Z = DH_2H_11A_110$ . This transformation technique is depicted in Fig. 3. It can be clearly seen that only small blocks of the sequence are exchanged. The other segments remain unchanged at their original positions.

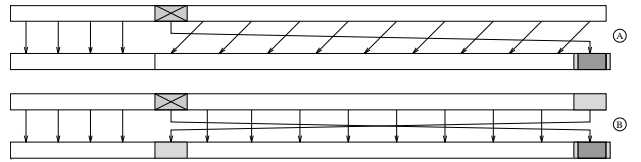


Fig. 3. Transformation of a source word into a constrained codeword using the standard method (A) and the modified technique (B).

*Example 4:* Source word  $X$  of length 64 is converted into a  $(0, 6)$  constrained codeword using method B to obtain  $Y$ , and using the modified method B to obtain  $Z$ . The codewords  $Y$  and  $Z$  are affected by one error in the part where the index of the violation at position  $t_1 = 6$ , represented by  $A_1 = 011000$ , is stored. Due to this error, the decoded index is 38 instead of 6. The decoded words are identified by  $\tilde{X}$ . The mapping  $Y = \mu_k(X)$  and  $Z = \mu'_k(X)$ , and the inverse mapping  $\tilde{Z} = \mu_k^{-1}(Y)$  and  $\tilde{X} = (\mu'_k)^{-1}(\tilde{Z})$  are as follows:

```

X  0110000000010110110000110111100110111010101110110110000011011
Y  0110001101100001101111001101110101011101101110000011011100010
Y-tilde  011000110110000110111100110111010101110111000001101110100110
X-tilde  0110001101100001101111001101110101010000001110110110000011011

X  01100000000101101100001101111001101110101011101101110000011011
Z  0110000011011011011000011011110011011101010111011011100101100010
Z-tilde  0110000011011011011000011011110011011101010111011011100101100110
X-tilde  011000001101101101100001101111001101100000011101101110011010101

```

Error propagation occurs in both situations. In the first case, the error causes a burst error of length 37 with 22 bit errors. With the modified method error propagation is limited to three bursts of lengths 4, 6, and 7, with 12 bit errors. In byte-oriented error control systems, 6 out of the 8 bytes have been destroyed in the first case, whereas 4 out of 8 bytes are destroyed in the second case. The differences in error propagation become more significant for larger codeword lengths.

There will be only one burst error when standard techniques are used, which is advantageous in certain error control schemes and for small wordlength. The modified construction significantly reduces the number of bit errors for long word lengths.

Error propagation only occurs due to errors in the index representation part. There are four different kinds of errors (see Fig. 4). If an error occurs in a segment representing an index,  $t_i$ , the position of the error in this segment is of importance. An error at a low-significant position causes a small shift resulting in two burst errors of maximum lengths  $k+1$  and one of length  $k+|t_i-\tilde{p}_i|+1 \leq 2k+2$ , whereas an error at a highly significant position causes three burst errors of maximum lengths  $k+1$ . It is possible to detect a wrongly decoded index if it is out of range, or if there is another inconsistency. Finally, if an error occurs at a position where information about the number of replacements is stored, one or more index segments will be created or removed, and each wrong index segment will cause two bursts of maximum lengths  $k+1$ . If an inconsistency is detected during the decoding process, and an error is expected, it is better not to decode this part, since the resulting bursts will be smaller than in the case where a forbidden sequence is inserted at the wrong position. This also underlines the importance of avoiding the segments to shift.

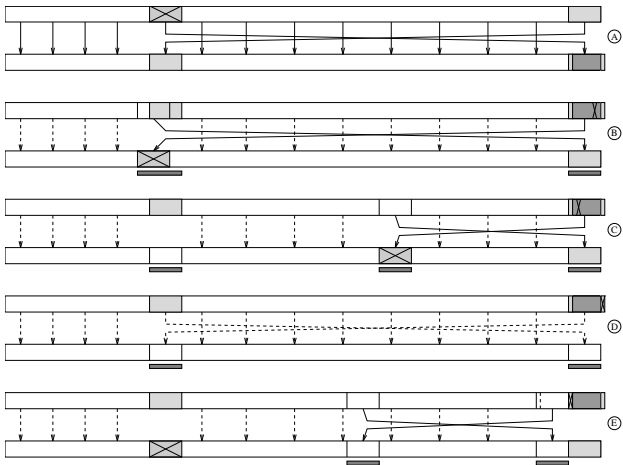


Fig. 4. Error control aspects for the modified replacement technique: the encoder (A), the operation of the decoder in the presence of an error at a low-significant or a high-significant position of the index segment (B,C), or the effect of a deletion or creation of an index segment due to an error at the position which indicates the number of replacements (D,E). The locations and the lengths of the burst errors in the decoded word are indicated.

The modified method limits error propagation to a few short bursts. Each of the presented methods can be modified in a similar fashion without increasing their complexity.

Errors in constrained sequences of length  $n$ , converted by using enumerative coding techniques, can cause bursts of any length up to  $n$ . Enumerative coding techniques use weights of up to  $n$  symbols for encoding and decoding. If an error occurs in the codeword at position  $i$ , a weight of  $i$  symbols is wrongly added, thereby causing a burst of errors of the  $i$  less significant positions of the source word. Therefore, bursts may occur which may be as long as the source word. In this respect, the methods based on the sequence replacement

technique involve significantly less error propagation than the enumerative coding techniques. This is of importance in systems where no or only limited error control is provided.

We compare the error propagation effects of a rate  $\frac{1024}{1025}$ ,  $(0, 10)$  code, constructed using the sequence replacement technique (method B), and a code with the same parameters, constructed using a coding algorithm based on enumeration with (12-bit) truncated coefficients [8]. The decoded bit error probability of words received over a binary symmetric channel is measured. The results are depicted in Fig. 5. The figure

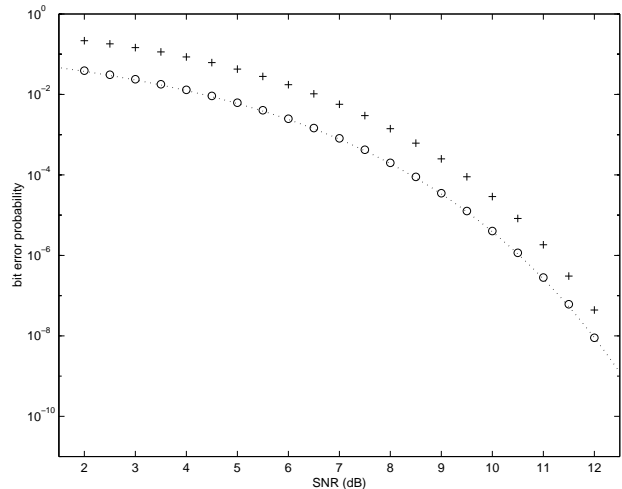


Fig. 5. Bit error probability measured at the output of the sequence replacement decoder (○) and the decoder based on enumeration with truncated coefficients (+). The dotted line indicates the uncoded bit error probability.

shows that the bit error probability measured at the output of the sequence replacement decoder is almost indistinguishable from the uncoded situation.

Simulations were performed to determine the bit error and byte error probability distribution as a result of a single error event. The parameters for the simulation were:  $n = 1025$ ,  $k = 10$ . The distribution for  $10^8$  words is shown in Fig. 6 and Fig. 7.

## B. Error-resilient Code Constructions

The error control capabilities of codes constructed using replacement techniques can be improved by slightly increasing the redundancy of the coding method in return for better error control. Error propagation only occurs if errors corrupt the segments in the codeword which indicate the number of replacements and to some extent the index representation parts. This is typically a small fraction of the sequence and can therefore be well protected using *unequal error protection* schemes. Another option is to *partition* the source words into smaller, separately coded, segments to restrict error propagation to the length of one sub-block. If the imposed constraints can be *relaxed*, the probability that a forbidden subsequence occurs in a source word is significantly reduced, and since the index representation of a replaced sequence is shorter, extra positions will become available to protect each index segment.

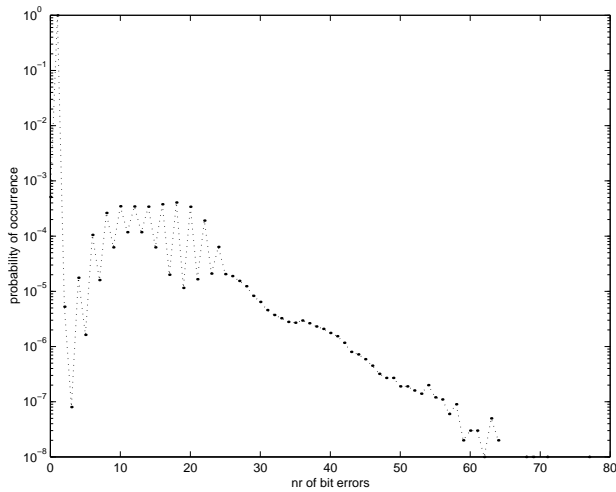


Fig. 6. Bit error probability distribution measured at the output of the decoder as a result of a single error event at the input.

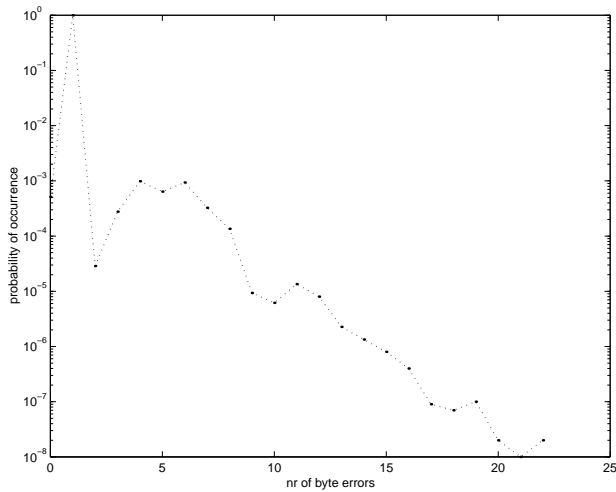


Fig. 7. Byte error probability distribution measured at the output of the decoder as a result of a single error event at the input.

### C. Error Control

The proposed modifications suppress but do not eliminate error propagation. If the received codewords are corrupted by errors, a common requirement is to detect the errors with a high probability, or to correct the errors. In [15] we presented efficient methods to protect constrained sequences against errors, thereby avoiding error propagation. These schemes enable the application of high rate constrained codes and provide error control with practically the same capabilities as for unconstrained sequences.

## V. CONCLUSIONS

It has been demonstrated that the proposed sequence replacement technique is a powerful method for the construction of maximum run-length limited codes. The underlying coding algorithms are very efficient and have a low complexity. Several techniques have been proposed for significantly reducing the impact of errors. It has been demonstrated that

the proposed constructions have clear advantages in terms of error propagation and achievable code rate over the modulation codes employed in recording systems.

## REFERENCES

- [1] K. A. S. Immink, *Codes for Mass Data Storage Systems*, 2nd ed. Eindhoven, The Netherlands: Shannon Foundation Publishers, 2004.
- [2] C. D. Mee and E. D. Daniel, *Magnetic Recording*. McGraw-Hill, New York, 1987.
- [3] P. H. Siegel and J. K. Wolf, "Modulation and coding for information storage," *IEEE Commun. Mag.*, vol. 29, no. 12, pp. 68–68, Dec. 1991.
- [4] K. A. S. Immink and H. D. L. Hollmann, "Prefix-synchronized runlength-limited sequences," *IEEE J. Sel. Areas Commun.*, vol. 10, no. 1, pp. 214–222, Jan. 1992.
- [5] A. J. van Wijngaarden and K. A. S. Immink, "Combinatorial construction of high rate runlength-limited codes," in *Proc. IEEE Global Telecommun. Conf.*, London, U.K., Nov. 1996, pp. 343–347.
- [6] A. J. van Wijngaarden and E. Soljanin, "A combinatorial technique for constructing high-rate MTR-RLL codes," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 582–588, Apr. 2001.
- [7] H. Morita, A. J. van Wijngaarden, and A. J. Han Vinck, "On the construction of maximal prefix-synchronized codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2158–2166, Nov. 1996.
- [8] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, vol. 43, no. 5, pp. 1389–1399, Sep. 1997.
- [9] Y. Sankarasubramaniam and S. W. McLaughlin, "Fixed-rate maximum-runlength-limited codes from variable-rate bit stuffing," *IEEE Trans. Inf. Theory*, vol. 53, no. 8, pp. 2769–2790, Aug. 2007.
- [10] A. J. van Wijngaarden and K. A. S. Immink, "On the construction of constrained codes employing sequence replacement techniques," in *Proc. IEEE Int. Symp. Information Theory*, Ulm, Germany, Jun. 1997, p. 144.
- [11] A. J. de Lind van Wijngaarden, "Frame synchronization techniques," Ph.D. dissertation, Univ. of Essen, Essen, Germany, Mar. 1998.
- [12] L. J. Guibas and A. M. Odlyzko, "String overlap, pattern matching, and nontransitive games," *J. Comb. Theory*, vol. 30, pp. 183–208, 1981.
- [13] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.
- [14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ: Wiley, 2006.
- [15] A. J. van Wijngaarden and K. A. S. Immink, "Maximum runlength-limited codes with error control capabilities," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 602–611, Apr. 2001.