

Maximum Runlength-Limited Codes with Error Control Capabilities

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

Abstract—New methods are presented to protect maximum runlength-limited sequences against random and burst errors and to avoid error propagation. The methods employ parallel conversion techniques and enumerative coding algorithms that transform binary user information into constrained codewords. The new schemes have a low complexity and are very efficient. The approach can be used for modulation coding in recording systems and for synchronization and line coding in communication systems. The schemes enable the usage of high-rate constrained codes, as error control can be provided with similar capabilities as for unconstrained sequences.

Index Terms—Burst correction codes, enumerative coding, forward error correction, modulation coding, Reed–Solomon codes, runlength codes, synchronization.

I. INTRODUCTION

MAXIMUM runlength-limited sequences have the property that long runs of consecutive like symbols do not occur. Maximum runlength-limited codes, in short, MRL 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 nonreversal of the polarization in a binary differential signaling scheme. The parameter k defines the maximum distance between transitions. The $(0, k)$ codes are used in recording systems [1], [2] and in digital communications systems as line codes for clock recovery or to align fixed and variable length data packets, also known as frame synchronization. Frame synchronization can be achieved by converting user information into a $(0, k)$ sequence of length m , which is prefixed by the sync word $0 \cdots 01$ of length $k + 2$ to uniquely identify the beginning of each frame [3].

Long sequences that fulfill $(0, k)$ constraints are typically constructed by concatenating the codewords of a $(0, k, k_l, k_r)$ block 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 practice, $(0, k, k_l, k_r)$ codes are often constructed with look-up tables or combinatorial circuitry [1], [2], [4], [5]. The rate is typically 8/9 and sometimes 16/17. The introduction of higher rate codes is hampered by the coding complexity, latency, and error propagation. In data networks, the bit-stuffing procedure [6] is widely employed. The drawbacks of this method are the data-dependent codeword length and error propagation [7]. Recently, algorithmic conversion methods based on enumeration [3], [8], [9] and on the sequence replacement technique [10], [11] have been proposed. These algorithms have a computational complexity proportional to the length of the codewords and attain the achievable rate for the given constraints.

The $(0, k, k_l, k_r)$ codes are usually not designed to correct transmission errors. Errors that transform the transmitted word into a noncodeword can be detected, but this is insufficient for reliable error control. Due to the variety of error types and their statistics for a particular communication channel, it is not practical to design a specific code for every possible constraint and error control requirement. Instead, we use existing constrained coding algorithms and error control codes. We propose to convert user information into a constrained sequence, preferably by using a very efficient coding algorithm [11], [12], and to protect the resulting sequence using a systematic error control code [13]. We present special techniques to add the error control information, i.e., the parity check bits, without violating the imposed constraints. In Section II, we describe the properties of MRL sequences, and in Section III, we propose three combined modulation and error control coding schemes. In Section IV, we present enumerative techniques for the analysis and construction of $(0, k, k_l, k_r)$ codes with additional constraints to facilitate error control. In Section V, we propose several combinatorial constructions of specific high-rate constrained codes. In Section VI, the methods are applied to the class of $(0, G/I)$ codes [2], [14], which fulfill global $(0, G)$ constraints as well as $(0, I)$ constraints for the two interleaved subsequences of elements with odd and even indexes, respectively. The $(0, G/I)$ codes arise in the context of coding for channels with a $1 - D^2$ impulse response, also known as PRML channels, to aid timing and gain recovery and simplify the design of the Viterbi detector for the PRML channel [2]. Concluding remarks are given in Section VII.

II. MAXIMUM RUNLENGTH-LIMITED SEQUENCES

To describe the properties of the techniques for constrained and error control coding, we need to introduce some notation. Let \mathcal{A}^w denote the set of sequences of w symbols from the

Manuscript received March 1, 2000; revised August 28, 2000. This work was supported in part by the German Science Foundation DFG. Parts of the material in this paper have been presented at the IEEE Global Telecommunications Conference GLOBECOM'96, London, U.K., November 1996, and at the 1998 IEEE International Symposium on Information Theory, Boston, MA, August 1998.

A. J. (de Lind) van Wijngaarden was with the Digital Communications Department, Institute for Experimental Mathematics, University of Essen, Essen, Germany. He is now with Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 USA.

K. A. Schouhamer Immink was with Philips Research Laboratories, Eindhoven, The Netherlands. He is now with the Digital Communications Department, Institute for Experimental Mathematics, University of Essen, 45326 Essen, Germany.

Publisher Item Identifier S 0733-8716(01)01754-1.

binary alphabet \mathcal{A} . A sequence $X \in \mathcal{A}^w$ is represented as a string of symbols $x_1x_2 \cdots x_w$ of length w . The concatenation of two sequences X and Y is denoted by XY , and $PS^{(w)} = \{PX|X \in \mathcal{S}^{(w)}\}$ denotes a set of sequences of length $p+w$ with prefix $P \in \mathcal{A}^p$ and suffix $X \in \mathcal{S}^{(w)}$, where $\mathcal{S}^{(w)} \subseteq \mathcal{A}^w$. Similarly, $\mathcal{S}^{(w)}P = \{XP|X \in \mathcal{S}^{(w)}\}$. The null string, denoted by Λ , 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 . We use $*$ and $(*)^w$ to denote any element of \mathcal{A} and \mathcal{A}^w , respectively.

1) *Lexicographical Order*: Any set $\mathcal{S}^{(m)} \subseteq \mathcal{A}^m$ can be ordered lexicographically by sorting the elements of $\mathcal{S}^{(m)}$ in the following way [9]. The sequence $X = x_1x_2 \cdots x_m \in \mathcal{S}^{(m)}$ is smaller than $Y = y_1y_2 \cdots y_m \in \mathcal{S}^{(m)}$, in short, $X < Y$, if and only an index j , $1 \leq j \leq m$ exists, such that $x_i = y_i$ for any $1 \leq i < j$, and $x_j < y_j$. Let \mathcal{N}_X be the set $\{Z \in \mathcal{A}^m | Z < X\}$. The cardinality N_X of the set \mathcal{N}_X is

$$N_X = \sum_{i=1}^m x_i 2^{m-i}. \quad (1)$$

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 $\mathcal{S}^{(m)}$.

2) *MRL Constrained Sequences*: Let $\mathcal{G}_{k,k_l,k_r}^{(n)}$ denote the set of binary sequences of length n that fulfill the $(0, k, k_l, k_r)$ constraint. This set can be written as [3]

$$\mathcal{G}_{k,k_l,k_r}^{(n)} = \bigcup_{s=0}^{k_l} 0^s 1 \mathcal{G}_{k,k,k_r}^{(n-s-1)}. \quad (2)$$

The disjoint subsets $\mathcal{G}_{k,k,k_r}^{(m)}$ are determined using recursion relations. For $0 \leq m \leq k_r$, $\mathcal{G}_{k,k,k_r}^{(m)} = \mathcal{A}^m$, and for $k_r < m \leq k$, $\mathcal{G}_{k,k,k_r}^{(m)} = \{P\mathcal{G}_{k,k,k_r}^{(k_r+1)} | P \in \mathcal{A}^{m-k_r-1}\}$. For $k < m < n$, $\mathcal{G}_{k,k,k_r}^{(m)}$ satisfies the recursion relation

$$\mathcal{G}_{k,k,k_r}^{(m)} = \bigcup_{s=0}^k 0^s 1 \mathcal{G}_{k,k,k_r}^{(m-s-1)}. \quad (3)$$

It follows that the cardinality of the set $\mathcal{G}_{k,k_l,k_r}^{(n)}$, denoted by $G_{k,k_l,k_r}^{(n)}$, is determined by two equations

$$G_{k,k,k_r}^{(m)} = \begin{cases} 2^m, & \text{if } 0 \leq m \leq k_r, \\ 2^{m-k_r-1}(2^{k_r+1} - 1), & \text{if } k_r < m \leq k, \\ G_{k,k,k_r}^{(m-1)} + \cdots + G_{k,k,k_r}^{(m-k-1)}, & \text{if } m \geq k+1 \end{cases}$$

and

$$G_{k,k_l,k_r}^{(n)} = \sum_{s=0}^{k_l} G_{k,k,k_r}^{(n-s-1)}. \quad (4)$$

Coding algorithms based on enumeration can be directly applied [12], [15].

3) *Constrained and Unconstrained Positions*: Consider a code $\mathcal{C}^{(n)} \subseteq \mathcal{A}^n$. The positions of the sequence $X \in \mathcal{A}^n$

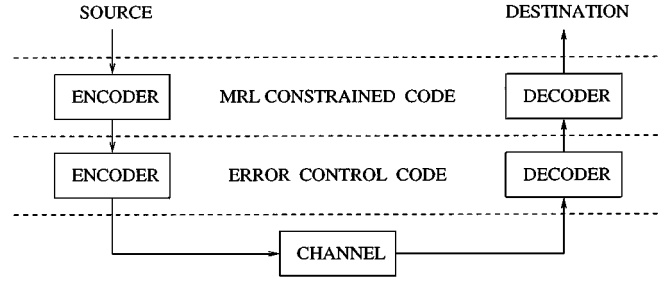


Fig. 1. Block diagram of a CC-EC scheme.

that determine whether X is an element of $\mathcal{C}^{(n)}$ are called the constrained positions of the code $\mathcal{C}^{(n)}$; the other positions are called the unconstrained positions. The symbol values at the unconstrained positions can be chosen or changed arbitrarily, and the resulting sequences are guaranteed to be elements of $\mathcal{C}^{(n)}$. Let \mathcal{U} and \mathcal{P} denote the respective ordered sets of u unconstrained and $n-u$ constrained positions of $\mathcal{C}^{(n)}$. The relative location of the unconstrained and the constrained positions of the code $\mathcal{C}^{(n)}$ can be specified by a string of length n , consisting of u characters and $n-u$ characters. This notation has the advantage that the relative location of the constrained and unconstrained positions is clearly visible. For example, the code

$$\mathcal{C}^{(5)} = \{01010, 01011, 01110, 01111, 10010, 10011, 10110, 10111\}$$

has two unconstrained positions (the third and the fifth) and three constrained positions. Thus, $\mathcal{P} = \{1, 2, 4\}$ and $\mathcal{U} = \{3, 5\}$, and $\mathcal{C}^{(5)}$ can be represented by $\{01*1*, 10*1*\}$. The “profile” of $\mathcal{C}^{(5)}$ is $\boxtimes \boxtimes \square \boxtimes \square$.

We define $\mathcal{C}_{\mathcal{P}}^{(n-u)}$ and $\mathcal{C}_{\mathcal{U}}^{(u)}$ as the set of all sequences in the respective positions \mathcal{P} and \mathcal{U} of $\mathcal{C}^{(n)}$. Note that $\mathcal{C}^{(n)} = \mathcal{C}_{\mathcal{P}}^{(n-u)} \times \mathcal{C}_{\mathcal{U}}^{(u)}$, and that $\mathcal{C}^{(n)}$ is determined by $\mathcal{C}_{\mathcal{P}}^{(n-u)}$, because according to the definitions, $\mathcal{C}_{\mathcal{U}}^{(u)} = \mathcal{A}^u$. The cardinality of $\mathcal{C}^{(n)}$ is equal to $|\mathcal{C}_{\mathcal{P}}^{(n-u)}| \cdot 2^u$. For the above example, we have $\mathcal{C}_{\mathcal{P}}^{(3)} = \{011, 101\}$ and $|\mathcal{C}^{(5)}| = 2 \cdot 2^2 = 8$.

4) *Code Efficiency*: For practical applications, it is often not necessary to use the entire code set. Usually, an m -bit source word has to be converted into a codeword of length n . This implies that the inequality $2^m \leq G_{k,k,k_r}^{(n)}$ should be fulfilled. 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 $1 - R/n = (1/n) \log_2(|\mathcal{C}^{(n)}|)$.

III. ERROR CONTROL CODING SCHEMES

The objective is to develop schemes to provide constrained sequences with effective error control capabilities. We propose three schemes that convert user information into constrained sequences and add error control information without violating the imposed constraints. The schemes have the common characteristic that an MRL constrained code is used prior to an error control code, as shown in Fig. 1. Such schemes are referred to as CC-EC schemes to indicate the order in which the constrained code and the error control code are applied.

The proposed schemes are in contrast to the typical approach employed in recording and communication systems, where every source word is encoded into a codeword of an error control code and successively converted into a constrained sequence [16]. At the receiver, the constrained sequence must be decoded first, and the resulting decoded sequence is checked for the presence of errors using the error control code. A major disadvantage of these so-called EC-CC schemes is error propagation. The codewords of the constrained code that are affected by transmission errors may be decoded incorrectly, causing burst errors in the decoded sequence. Hence, the error correcting code must be a burst correcting code [13], even when noise in the channel is dominated by random errors. It should be noted that in general, more redundancy is needed to correct t burst errors than to correct t random errors. The EC-CC scheme requires the wordlength of the constrained code to be small to restrict error propagation. Typical block codes such as the rate 8/9, $(0, 3, 1, 2)$ code [17] convert 8-bit source words into nine-bit codewords using a look-up table or a combinatorial circuit consisting of a few logic gates. In this case, error propagation is limited to one byte, and this allows the restoration of the codewords by using byte-oriented Reed-Solomon codes that can correct multiple byte errors in a codeword at high speed [2].

The proposed CC-EC schemes avoid error propagation by first checking for transmission errors before decoding the constrained sequence. The choice of a particular systematic error control code depends mainly on the channel type, the relative error statistics, the required error detection and/or error correction capabilities, and the required level of reliability of the received data after decoding. The schemes facilitate the usage of soft decision and other side information, e.g., violations of the imposed constraints. As long as the error correcting capabilities are not exceeded, there will not be any errors left in the constrained sequence. The restriction of employing small constrained block codes can therefore be dispensed with. The proposed CC-EC schemes use several methods to insert the check symbols of the systematic error control code in the constrained sequence without violating the imposed $(0, k, k_l, k_r)$ constraints.

A. CC-EC Scheme I

The first scheme that we will consider is a three-step constrained coding and error control coding scheme. The block diagram of this scheme, which is shown in Fig. 2, is similar to the schemes proposed in [12], [15], [18]–[20], and references therein, but by applying this scheme for the protection of $(0, k)$ codes, important modifications can be made.

This scheme converts source words into constrained codewords in three successive steps. Let m be the length of the source word. In the first stage, a source word of length m is converted into a $(0, k, k_l, k_r)$ sequence of length $m + v_c$, which is in the second stage protected against errors by using a systematic error control code. The m_e parity check bits of the error control code are separately converted into a $(0, k, k_l, k_r)$ constrained sequence of length $m_e + v_e$ and appended to form a $(0, k, k_l, k_r)$ constrained sequence of length $n = m + v_c + m_e + v_e$.

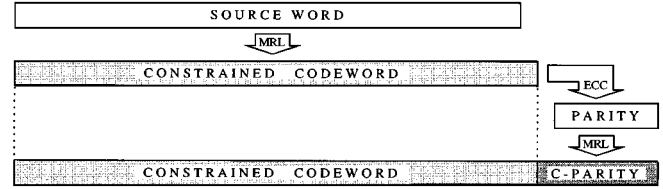


Fig. 2. Block diagram of CC-EC Scheme I, a three-step modulation and error control coding scheme to convert source words into constrained sequences. First, a source word is converted into a $(0, k, k_l, k_r)$ MRL sequence. Second, a systematic error control code (ECC) is used and the corresponding parity check bits are determined. Third, the parity check bits are converted into a $(0, k, k_l, k_r)$ MRL sequence to fulfill the maximum runlength constraints.

The parity check sequence of length m_e has to be carefully converted into a constrained sequence to avoid or at least to limit error propagation. Error propagation can be avoided by interspersing the parity check sequence with t “ones” in order to satisfy the $(0, k, k_l, k_r)$ constraints. Consequently, a $(0, k, k_l, k_r)$ code $\mathcal{C}^{(m_e+t)}$ is constructed, where $\mathcal{C}_{\mathcal{U}}^{(m_e)} = \mathcal{A}^{m_e}$ contains the parity check sequence, and $\mathcal{C}_{\mathcal{P}}^{(t)} = \{1^t\}$. The minimum value of t is given by

$$t = \left\lceil \frac{m_e - k_l - k_r}{k} \right\rceil + 1. \quad (5)$$

If $m_e = (t-1)k + k_l + k_r$, the positions of $\mathcal{C}^{(m_e+t)}$ are assigned as follows:

$$\square^{k_l} (\square^k)^{t-1} \square^{k_r}. \quad (6)$$

Otherwise, $(t-1)k + k_l + k_r - m_e$ unconstrained positions can be removed arbitrarily to obtain the appropriate length.

If burst error control codes are used, the sequence of parity check symbols may as well be converted into a $(0, k, k_l, k_r)$ codeword with limited error propagation. It is, however, of paramount importance that the error propagation of this code be limited to a few bits and aligned, preferably, to one byte in a byte-oriented system. Codes with these attributes can be designed, as will be discussed in the next sections.

The redundancy of this CC-EC scheme is $R = v_c + m_e + v_e$. The number of parity bits is commonly a small fraction of the length of the source word. The extra redundancy that has to be spent to avoid error propagation in the check sequence by interspersing “ones” is moderate. If, on the other hand, the parity check bits are converted into a constrained sequence of a short $(0, k, k_l, k_r)$ code with limited error propagation, the scheme is only a partial CC-EC scheme. The next two schemes are genuine CC-EC schemes.

B. CC-EC Scheme II

The second error control scheme converts user information into a $(0, k, k_l, k_r)$ constrained sequence in two consecutive steps. The block diagram of this scheme is shown in Fig. 3.

A source word of length m' is converted into a codeword of a $(0, k', k'_l, k'_r)$ constrained sequence of length n' , where $k' \leq k$, $k'_l \leq k_l$, and $k'_r \leq k_r$. This codeword is protected by a systematic error control code. The e parity bits are inserted in the codeword in a specific manner [21], such that the resulting sequence

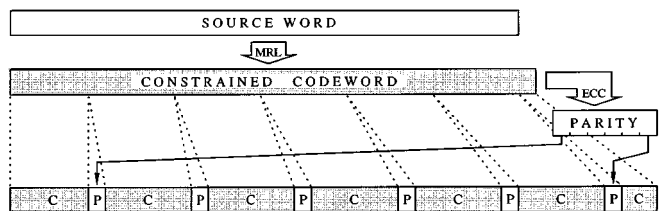


Fig. 3. Block diagram of CC-EC Scheme II, a two-step modulation and error control coding scheme. Information is converted into a constrained sequence with tighter constraints than required to allow the parity bits to be inserted without violating the imposed constraints.

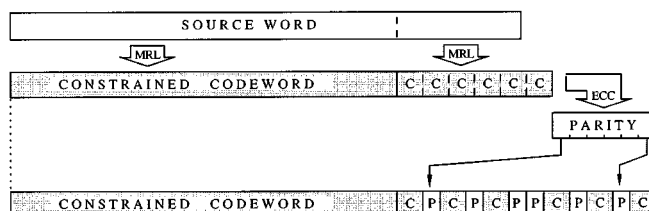


Fig. 4. Block diagram of CC-EC Scheme III, a two-step modulation and error control coding scheme. Information is converted into a sequence of a constrained code, which provides unconstrained positions. These positions, which are initially set to zero, are used to store the parity check symbols.

fulfills the $(0, k, k_l, k_r)$ constraints. This can be accomplished as follows.

We will consider the case where a $(0, k', k'_l, k'_r)$ code with codewords $c_1 \cdots c_{n'}$ of length n' is used to construct a $(0, k, k_l, k_r)$ code of length n . Let $\delta_k = k - k'$, $\delta_l = k_l - k'_l$, and $\delta_r = k_r - k'_r$.

We can add up to δ_l unconstrained positions at the beginning of the sequence because this limits the number of leading zeros to $k'_l + \delta_l = k_l$. Using the same argument, we can append at most δ_r unconstrained positions at the end of the sequence.

The positions are assigned as follows:

$$\square^{\delta_l} \boxtimes^{k'_l+1} \square^{\delta_k} (\boxtimes^{k'+1} \square^{\delta_k})^w \boxtimes^{v-w(k'+1)} \boxtimes^{k'_r+1} \square^{\delta_r}, \quad (7)$$

where $v = n' - k'_l - k'_r - 2$ and $w = \lfloor v/(k' + 1) \rfloor$. This code has $u = (w + 1)\delta_k + \delta_l + \delta_r$ unconstrained positions and a total length $n = n' + u$. It can be easily verified that the sequence of this form satisfies the $(0, k, k_l, k_r)$ constraints. The number of leading zeros is at most $k'l' + \delta_l = k_l$, and the number of trailing zeros is at most $k'r' + \delta_r = k_r$. The runlength constraint at every other position is at most $k' + \delta_k = k$, because every segment $\boxtimes^s \square^{\delta_k} \boxtimes^{k'+2-s}$ is in the worst case occupied by $10^s 0^{\delta_k} 0^{k'-s} 1$ for $0 \leq s \leq k'$. The blocks \square^{δ_k} are separated by constrained segments $\boxtimes^{k'+1}$ of length $k' + 1$, and therefore, the constrained segment will not be all-zero.

Example 1: A rate 16/17, $(0, 4, 2, 2)$ code, presented in [21], can be used as a component code to obtain, using the previously discussed interleaving technique, a rate 24/25, $(0, 6, 3, 3)$ code with $u = 8$ unconstrained positions. The structure of this code is specified by

$$\square \boxtimes \boxtimes \boxtimes \square \square \boxtimes \boxtimes \boxtimes \square \square \boxtimes \boxtimes \boxtimes \square \square \boxtimes \boxtimes \boxtimes \square \square \square \square \square \square \boxtimes \boxtimes \boxtimes \square.$$

In the next section, it will be shown that there are at most $u_{\max} = 12$ unconstrained positions for the given values $n, m, k, k_l,$ and k_r . These unconstrained positions are distributed in the following way:

$$\square \boxtimes \boxtimes \boxtimes \square \square \square \boxtimes \boxtimes \boxtimes \square \square \square \boxtimes \boxtimes \boxtimes \square \square \square \square \square \square \boxtimes \boxtimes \boxtimes \square.$$

This code has $17047552 = 4162 \cdot 2^{12}$ codewords.

C. CC-EC Scheme III

The third CC-EC scheme reserves unconstrained positions in the constrained sequence. These positions are to be filled by error check symbols in the second stage. This scheme, illustrated in Fig. 4, combines the idea of splitting the constrained sequences into several subsequences, thereby protecting longer

and efficiently coded sequences by shorter subsequences, and by using unconstrained positions to store the error check symbols. At the encoder side, the user information is split into subsequences and encoded in such a manner that the codewords still satisfy the constraints, if all unconstrained positions contain “zeros.” This is the worst-case situation, and therefore, the maximum runlength constraints will be automatically fulfilled for any combination of error check symbols. In the second stage, the error check symbols of the error control code that protect the constrained sequence are stored at the unconstrained positions. This does not affect the imposed constraints, and therefore, block error control codes that match the channel impairment characteristics can be selected.

At the receiver side, the decoder first extracts the parity check symbols and then it checks the codeword for the presence of errors. After verification and possibly correction of the constrained sequence by the error control code, the constrained sequence is decoded.

The unconstrained positions could be incorporated in long codewords, but they could also be part of one or several shorter constrained subsequences. Several techniques for the construction of constrained sequences with unconstrained positions will be presented in the next section.

D. Properties of CC-EC Schemes

The three CC-EC schemes offer effective protection of constrained sequences of fixed and variable lengths. Of importance for the selection of one of the CC-EC schemes are the actual constraints and the required level of error protection. The CC-EC schemes allow soft information from the detector and the MRL decoder to be used to aid error correction. For instance, the segments of the received word where the constraints are violated can be declared as an erasure.

IV. CONSTRAINED CODES WITH UNCONSTRAINED POSITIONS

In this section, we present a general technique to construct a rate m/n , $(0, k, k_l, k_r)$ code with u unconstrained positions. We first develop a method to determine the maximum number of codewords of a $(0, k, k_l, k_r)$ code of length n with u unconstrained positions. Next, we will develop coding algorithms based on enumeration that achieve the upper bound on the cardinality for the given parameters. In Section V, we will present several constrained codes that can be constructed with look-up tables or with combinatorial circuitry. The efficiency of the constructed codes will be discussed.

binary conversion. Instead of the usual powers of two, the integer weights $L_{\kappa}^{(i)}$, $0 \leq i < n$, are used. The following algorithm describes the conversion of X into $Y = y_1 \cdots y_n$:

```

Y := function Encode( $\xi$ )
begin
  for  $i$  from 1 to  $n$  do begin
    if ( $\xi \geq L_{\kappa}^{(n-1)}$ ) then begin
       $y_i := 0$ ;
       $\xi := \xi - L_{\kappa}^{(i)}$ ;
    end else  $y_i := 1$ ;
    end;
  return( $y_1 \cdots y_n$ );
end;

```

Reconstruction of the source word given the codeword $Y = y_1 \cdots y_n$ is easily accomplished using the following procedure:

```

 $\xi :=$  function Decode( $y_1 \cdots y_n$ )
begin
   $\xi' := 0$ ;
  for  $i$  from 1 to  $n$  do begin
    if ( $y_i = 0$ ) then  $\xi' := \xi' + L_{\kappa}^{(n-i)}$ ;
    end;
  return( $\xi'$ );
end;

```

It can be easily shown that every word that fulfills the constraint can be generated with the enumerative coding technique. The main advantage of this technique is the property that both the encoding and the decoding operation can be performed with linear complexity using only n weights. Every weight is represented by at most n bits: $W_i = \text{id}x^{-1}(\mathcal{A}^n, L_{\kappa}^{(i)})$, $1 \leq i \leq n$. We can truncate the coefficients to simplify the computations and to limit error propagation [12].

V. COMBINATORIAL CONSTRUCTION METHODS

In this section, we consider the construction of $(0, k, k_l, k_r)$ codes of length n with u unconstrained positions. The constraints are, as previously discussed, imposed by a constraint-imposing component code $\mathcal{C}_P^{(n-u)}$, in short, a CIC code. We will propose seven combinatorial constructions (A–G) of CIC codes for the composition of a rate $(n - 1)/n, (0, k, \lfloor k/2 \rfloor, \lfloor k/2 \rfloor)$ code with the maximum number of unconstrained positions. The underlying combinatorial techniques that are used are discussed in detail in [5]. The seven different constructions are used to span particular ranges of the parameters n, k , and u .

The maximum number of unconstrained positions u_{\max} for which a rate $(n - 1)/n, (0, k, k_l, k_r)$ code exists can be determined with (10) or (12). Table I lists the maximum number of unconstrained positions u_{\max} of a rate $(n - 1)/n, (0, k, \lfloor k/2 \rfloor, \lfloor k/2 \rfloor)$ code of length n for $3 \leq k \leq 10$ and $8 \leq n \leq 28$. As is to be expected, the maximum number of unconstrained positions for a given value of k first increases with the wordlength n , and then it decreases to zero. By loosening the k constraint, the maximum number of unconstrained positions increases rapidly.

TABLE I
MAXIMUM NUMBER OF UNCONSTRAINED POSITIONS u_{\max} OF A RATE $(n - 1)/n, (0, k, \lfloor k/2 \rfloor, \lfloor k/2 \rfloor)$ CODE OF LENGTH n AND THE APPLICABLE CONSTRUCTION A–G

n	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
8	1 C	4 B	4 B	4 B	7 A	7 A	7 A	7 A
9	1 D	5 B	5 B	5 B	5 B	8 A	8 A	8 A
10	1 E	3 C	6 B	6 B	6 B	6 B	9 A	9 A
11	0	3 D	7 B	7 B	7 B	7 B	7 B	10 A
12	0	4 D	5 C	8 B	8 B	8 B	8 B	8 B
13	–	4 E	5 D	9 B	9 B	9 B	9 B	9 B
14	–	3 F ₁	6 D	7 C	10 B	10 B	10 B	10 B
15	–	3 G	7 D	7 D	11 B	11 B	11 B	11 B
16	–	4 G	7 E	8 D	9 C	12 B	12 B	12 B
17	–	4	6 F ₁	9 D	9 D	13 B	13 B	13 B
18	–	3	7 F ₁	10 D	10 D	11 C	14 B	14 B
19	–	3	7 G	10 E	11 D	11 D	15 B	15 B
20	–	3	8 G	9 F ₁	12 D	12 D	13 C	16 B
21	–	3	8	10 F ₁	13 D	13 D	13 D	17 B
22	–	3	7	10 G	13 E	14 D	14 D	15 C
23	–	2	7	11 G	12 F ₂	15 D	15 D	15 D
24	–	2	7	12 G	13 F ₂	16 D	16 D	16 D
25	–	2	8	12	14 F ₂	16 E	17 D	17 D
26	–	2	8	11	14 G	15 F ₂	18 D	18 D
27	–	1	8	11	15 G	16 F ₂	19 D	19 D
28	–	1	7	12	16 G	17 F ₂	19 E	20 D

The letters A–G in Table I are used to indicate which of the constructions is to be used to obtain a code with the given parameters.

We describe the construction of $(0, k, k_l, k_r)$ codes by a set of rules that prescribe the mapping of source words of length $n - 1$ onto n -bit codewords of a $(0, k, k_l, k_r)$ code. If the number of sequences of length n that satisfy the $(0, k)$ constraints exceeds 2^{n-1} , it is always possible to perform this mapping, although the complexity may be high. For several codeword lengths, it is possible to realize this mapping procedure using a small, digital combinatorial circuit. The codes that have been developed are characterized by a set of source-dependent mapping rules. The encoder first checks whether the source word would be a valid codeword. If this is the case, the encoder copies the source word and inserts one extra bit that indicates that the other bits of the codeword are a copy of the source word. Otherwise, the extra bit indicates that a mapping rule had to be applied to transform the source word into an $(n - 1)$ -bit coded sequence that satisfies the imposed $(0, k, k_l, k_r)$ constraints. The coded sequence, which uniquely represents the source word, is composed of a mapping rule identifier and rearranged source symbols. The mapping structure is often symmetric to simplify the design process and the structure of the digital combinatorial circuitry [5].

A. Construction A

The first construction is based on the insertion of “ones” to fulfill the $(0, k, k_l, k_r)$ constraints. The placement of the unconstrained and constrained positions is specified by (6). For a rate $(n - 1)/n, (0, k, k_l, k_r)$ code $\mathcal{C}^{(n)}$, this reduces for $n = n_A = k_l + k_r$ to $\square^{k_l} \boxtimes \square^{k_r}$, where $\mathcal{C}_P^{(1)} = \{1\}$ is the CIC code. For $n < n_A$, we can remove $n_A - n$ unconstrained positions. It follows that $u_A = n - 1$.

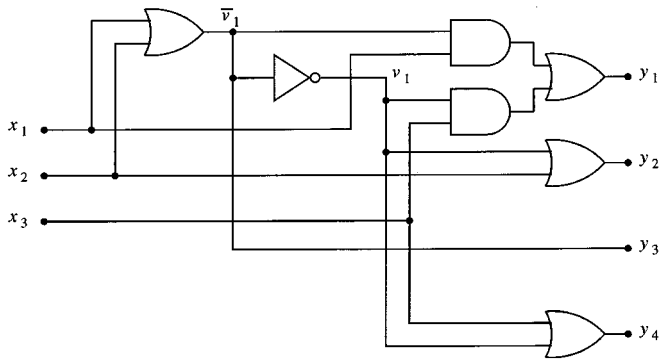


Fig. 5. Construction B. Combinatorial circuit for the encoder.

B. Construction B

A rate $(n-1)/n$, $(0, k, k_l, k_r)$ code $\mathcal{C}^{(n)}$ with $u_B = n-4$ unconstrained positions can be constructed by using a rate $3/4$, $(0, 1, 1, 1)$ code as the CIC code $\mathcal{C}_P^{(4)}$ and by placing the symbols, in the case where $n = n_B = k + k_l + k_r + 1$, as follows:

$$\square^{k_l-1} \boxtimes \boxtimes \square^{k-1} \boxtimes \boxtimes \square^{k_r-1}.$$

For $n < n_B$, we can remove $n_B - n$ unconstrained positions.

The mapping $\theta_B: \mathcal{A}^3 \rightarrow \mathcal{A}^4$ of a source word $X = x_1x_2x_3$ onto a codeword $Y = y_1y_2y_3y_4$ of the rate $3/4$, $(0, 1, 1, 1)$ code is specified by

$$Y = \theta_B(X) = \begin{cases} x_3 \ 1 \ 0 \ 1, & \text{if } x_1 = 0 \text{ and } x_2 = 0, \\ x_1 \ x_2 \ 1 \ x_3, & \text{otherwise.} \end{cases} \quad (13)$$

This can be described in terms of logic equations as follows:

$$Y = [\bar{v}_1 \cdot x_1 + v_1 \cdot x_3, x_2 + v_1, \bar{v}_1, x_3 + v_1],$$

where $v_1 = \overline{x_1 + x_2}$. In these logic equations, the addition is equivalent to a binary OR operation, and a multiplication is equivalent to a binary AND operation. The inverse of a binary value x is denoted by \bar{x} , and the inverse of a binary expression $p(x_1, x_2, \dots, x_n)$ is denoted by

$$\overline{p(x_1, x_2, \dots, x_n)}.$$

The encoder can be realized with 4 OR gates, 2 AND gates, and 1 INV gate. The circuit is shown in Fig. 5. The delay is $2T_{\text{OR}} + T_{\text{AND}} + T_{\text{INV}}$. The decoder performs the reverse operation by mapping the codeword Y onto the word $Z = z_1z_2z_3$. This mapping can be described in terms of logic equations by

$$Z = [y_3 \cdot y_1, y_3 \cdot y_2, y_3 \cdot y_4 + \bar{y}_3 \cdot y_1].$$

The decoder can, as shown in Fig. 6, be realized with 1 OR gate, 4 AND gates, and 1 INV gate. The delay is $T_{\text{OR}} + T_{\text{AND}} + T_{\text{INV}}$.

C. Construction C

A rate $(n-1)/n$, $(0, k, k_l, k_r)$ code $\mathcal{C}^{(n)}$ with $u_C = n-7$ unconstrained positions can be constructed by using a rate $6/7$, $(0, 2, 1, 2)$ code as the CIC code $\mathcal{C}_P^{(7)}$ and by placing the symbols, in the case where $n = n_C = k + k_l + k_r + 2$, as follows:

$$\square^{k_l-1} \boxtimes \boxtimes \square^{k-2} \boxtimes \boxtimes \boxtimes \boxtimes \square^{k_r-2}.$$

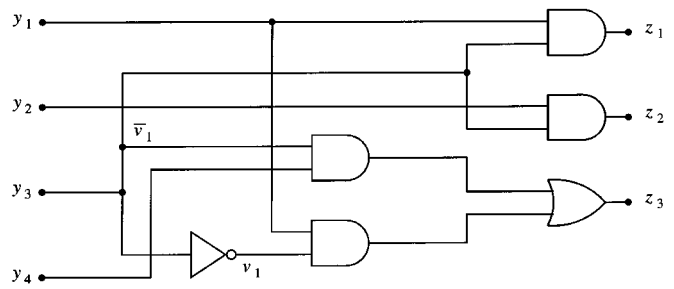


Fig. 6. Construction B. Combinatorial circuit of the decoder.

TABLE II
CONSTRUCTION C—MAPPING SCHEME OF A RATE $6/7$, $(0, 2, 1, 2)$ CODE

	source word X	y_1	y_2	y_3	y_4	y_5	y_6	y_7
v_1	0 0 x_3 x_4 x_5 x_6	x_4	1 x_3	<u>0</u>	<u>1</u>	x_5	x_6	
v_2	x_1 x_2 x_3 0 0 0	x_1	x_2	1	<u>0</u>	<u>0</u>	1	x_3
v_0	x_1 x_2 x_3 x_4 x_5 x_6	x_1	x_2	x_3	<u>1</u>	x_4	x_5	x_6

The mapping $\theta_C: \mathcal{A}^6 \rightarrow \mathcal{A}^7$ of a source word $X = x_1x_2 \dots x_6$ onto a codeword $Y = y_1y_2 \dots y_7$ of the rate $6/7$, $(0, 2, 1, 2)$ code is specified in Table II.

The subsequences that violate the constraints are bold-faced to emphasize the kind of violation that is under consideration. If the source word matches the first word, the corresponding mapping rule will be applied; else, if the source word matches the second word, the second rule is applied. This process continues until a match is found and the corresponding rule is applied. The underlined zeros and ones indicate which positions uniquely specify the inverse mapping. The decoder checks these positions, and if there is a match, the mapping will be reversed to reconstruct the source word.

The mapping given in Table II can be realized by a small combinatorial circuit, which is specified by

$$\begin{aligned} y_1 &= \bar{v}_1 \cdot x_1 + v_1 \cdot x_4, & y_5 &= \bar{v}_2 \cdot x_4 + v_1, \\ y_2 &= x_2 + v_1, & y_6 &= x_5 + v_2, \\ y_3 &= x_3 + v_2, & y_7 &= \bar{v}_2 \cdot x_6 + v_2 \cdot x_3, \\ y_4 &= \bar{v}_1 \cdot \bar{v}_2, \end{aligned}$$

where $v_1 = \overline{x_1 + x_2}$ and $v_2 = \overline{x_4 + x_5 + x_6} \cdot \bar{v}_1$. The encoder consists of 9 OR gates, 7 AND gates, and 2 INV gates. The delay is $3T_{\text{OR}} + 2T_{\text{AND}} + 2T_{\text{INV}}$.

The inverse mapping is given in terms of logic equations by

$$Z = [\bar{v}_1y_1, \bar{v}_1y_2, \bar{v}_2y_3 + v_2y_7, (y_1 + y_4)y_5, \bar{v}_2y_6, \bar{v}_2y_7]$$

where $v_1 = \bar{y}_4 \cdot y_5$ and $v_2 = \overline{y_4 + y_5}$. The decoder can be realized with 7 AND gates, 3 OR gates, and 1 INV gate. The delay is $2T_{\text{OR}} + T_{\text{AND}} + T_{\text{INV}}$.

D. Construction D

A rate $(n-1)/n$, $(0, k, k_l, k_r)$ code $\mathcal{C}^{(n)}$ with $u_D = n-8$ unconstrained positions can be constructed by using a rate $7/8$ code as the CIC code $\mathcal{C}_P^{(8)}$ and by placing the symbols, in the case where $n = n_D = 2k + k_l + k_r$, as follows:

$$\square^{k_l-2} \boxtimes \boxtimes \boxtimes \square^{k-3} \boxtimes \boxtimes \boxtimes \square^{k-2} \boxtimes \boxtimes \square^{k_r-1}.$$

The mapping for the rate $7/8$ code is specified in Table III.

TABLE III
CONSTRUCTION D—MAPPING SCHEME

	source word X	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
$w_{1,1}$	0 0 0 x_4 x_5 0 0	x_5	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	x_4
$w_{0,1}$	x_1 x_2 x_3 x_4 x_5 0 0	x_1	x_2	x_3	<u>1</u>	<u>0</u>	x_5	<u>1</u>	x_4
$w_{1,0}$	0 0 0 x_4 x_5 x_6 x_7	x_5	x_4	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_6	x_7
$w_{0,0}$	x_1 x_2 x_3 x_4 x_5 x_6 x_7	x_1	x_2	x_3	x_4	<u>1</u>	x_5	x_6	x_7

Let $v_1 = \overline{x_1 + x_2 + x_3}$ and $v_2 = \overline{x_6 + x_7}$. By evaluating v_1 and v_2 , we can easily determine which mapping is to be applied. The combinatorial circuitry of the encoder is specified by the following equations:

$$\begin{aligned} y_1 &= \overline{v_1} \cdot x_1 + v_1 \cdot x_5, & y_5 &= \overline{v_1} \cdot \overline{v_2}, \\ y_2 &= \overline{v_1} \cdot x_2 + v_1(v_2 + x_4), & y_6 &= \overline{v_1} \cdot x_5 + v_1, \\ y_3 &= \overline{v_1} \cdot x_3 + v_1 \cdot \overline{v_2}, & y_7 &= \overline{v_2} \cdot x_6 + v_2, \\ y_4 &= \overline{v_1}(\overline{v_2} \cdot x_4 + v_2), & y_8 &= \overline{v_2} \cdot x_7 + v_2 \cdot x_4. \end{aligned}$$

The encoder can be realized with 13 AND gates, 11 OR gates, and 2 INV gates. The delay is $2T_{\text{OR}} + T_{\text{AND}} + T_{\text{INV}}$. The logic equations for the decoder can be obtained in a similar fashion.

E. Construction E

A rate 8/9, (0, 2, 2, 2) code can be used as a component code $C_{\mathcal{P}}^{(9)}$ of the rate $(n-1)/n$, (0, k , k_l , k_r) code $C^{(n)}$, which has the following structure in the case where $n = n_E = 2k + k_l + k_r + 1$:

$$\square^{k_l-2} \boxtimes \boxtimes \boxtimes \square^{k-2} \boxtimes \boxtimes \boxtimes \square^{k-2} \boxtimes \boxtimes \boxtimes \square^{k_r-2}.$$

For $n < n_E$, we can remove $n_E - n$ unconstrained positions. It follows that $u_{\max} = n - 9$.

The mapping for the rate 8/9, (0, 2, 2, 2) code is specified in Table IV.

F. Construction F_1

A rate 10/11, CIC code with $\kappa = [2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 1]$ can be used to construct a rate $(n-1)/n$, (0, k , k_l , k_r) code. The location of the unconstrained and constrained positions are specified for $4 \leq k \leq 6$ by

$$\begin{aligned} k=4: & \boxtimes \boxtimes \boxtimes \square \boxtimes \boxtimes \square \boxtimes \boxtimes \boxtimes \square \boxtimes \boxtimes \boxtimes, \\ k=5: & \boxtimes \boxtimes \boxtimes \square \square \boxtimes \boxtimes \square \boxtimes \boxtimes \boxtimes \square \square \boxtimes \boxtimes \boxtimes \square, \\ k=6: & \square \boxtimes \boxtimes \boxtimes \square \square \square \boxtimes \boxtimes \square \boxtimes \boxtimes \square \square \square \boxtimes \boxtimes \boxtimes \square. \end{aligned}$$

There are $1066 > 2^{10}$ valid constrained sequences of length 11. The mapping for the rate 10/11 CIC code is specified in Table V. The encoder checks whether the word $x_1^l x_2^l x_3^l x_4^l x_5^l x_6^l x_7^l x_8^l x_9^l x_{10}^l x_{11}^l$ fulfills the constraints. This is the case if subsequence $x_1 x_2 x_3 \neq 000$ and subsequence $x_3 x_4 x_5 \neq 000$.

The mapping given in Table V can be implemented with approximately 50 logic gates.

G. Construction F_2

The construction F_1 does not give the maximum number of unconstrained positions. Instead, the maximum is obtained by a rate 10/11 CIC code for which $\kappa = [2, 4, 3, 2, 4, 3, 2, 3, 2, 1, 1]$. This CIC code can be used to construct a rate $(n-1)/n$,

TABLE IV
CONSTRUCTION E—MAPPING SCHEME

	source word X	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
$w_{1,1}$	0 0 0 x_4 x_5 0 0 0	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	x_5	x_4
$w_{1,0}$	0 0 0 x_4 x_5 x_6 x_7 x_8	x_5	x_4	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_6	x_7	x_8
$w_{0,1}$	x_1 x_2 x_3 x_4 x_5 0 0 0	x_1	x_2	x_3	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_5	x_4
$w_{2,0}$	1 0 0 0 x_5 x_6 x_7 x_8	x_5	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	x_6	x_7	x_8
$w_{0,2}$	x_1 x_2 x_3 x_4 0 0 0 1	x_4	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	x_1	x_2	x_3
$w_{0,0}$	x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8	x_1	x_2	x_3	x_4	<u>1</u>	x_5	x_6	x_7	x_8

TABLE V
CONSTRUCTION F_1 —MAPPING SCHEME

	source word X	y_1^l	y_2^l	y_3^l	y_4^l	y_5^l	y_6^l	y_7^l	y_8^l	y_9^l	y_1^r	y_2^r	y_3^r	y_4^r	y_5^r	y_6^r	y_7^r	y_8^r	y_9^r
$w_{1,1}$	$x_1^l x_2^l$ 0 0 0 0 0 0 $x_7^r x_8^r$	$x_1^l x_2^l$	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	$x_5^r x_6^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{2,1}$	0 0 0 $x_4^l x_5^l$ 0 0 0 $x_7^r x_8^r$	$x_4^l x_5^l$	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	$x_5^r x_6^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{1,2}$	$x_1^l x_2^l$ 0 0 0 $x_5^r x_6^r$ 0 0 0	$x_1^l x_2^l$	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	$x_5^r x_6^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{2,2}$	0 0 0 $x_4^l x_5^l x_6^l x_7^l$ 0 0 0	$x_4^l x_5^l$	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	$x_5^r x_6^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{1,0}$	$x_1^l x_2^l$ 0 0 0 $x_5^r x_6^r x_7^r x_8^r$	$x_1^l x_2^l$	<u>1</u>	<u>x_5^r</u>	<u>1</u>	<u>0</u>	<u>1</u>	$x_4^r x_5^r$	$x_6^r x_7^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r	
$w_{2,0}$	0 0 0 $x_4^l x_5^l x_6^l x_7^l x_8^l x_9^l$	$x_4^l x_5^l$	<u>0</u>	<u>x_5^r</u>	<u>1</u>	<u>0</u>	<u>1</u>	$x_4^r x_5^r$	$x_6^r x_7^r$	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r	
$w_{0,1}^{[1]}$	$x_1^l x_2^l x_3^l$ <u>1</u> x_5^l 0 0 0 $x_7^r x_8^r$	$x_1^l x_2^l x_3^l$	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_5^r	<u>1</u>	x_5^r	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{0,1}^{[2]}$	$x_1^l x_2^l x_3^l$ <u>0</u> x_5^l 0 0 0 $x_7^r x_8^r$	$x_1^l x_2^l x_3^l$	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_5^r	<u>0</u>	x_5^r	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{0,2}^{[1]}$	$x_1^l x_2^l x_3^l$ <u>1</u> $x_5^l x_6^l$ 0 0 0	$x_1^l x_2^l x_3^l$	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_5^l	<u>0</u>	x_5^l	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{0,2}^{[2]}$	$x_1^l x_2^l x_3^l$ <u>0</u> $x_5^l x_6^l$ 0 0 0	$x_1^l x_2^l x_3^l$	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	x_5^l	<u>0</u>	x_5^l	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r
$w_{0,0}$	$x_1^l x_2^l x_3^l x_4^l x_5^l x_6^l x_7^l x_8^l x_9^l$	$x_1^l x_2^l x_3^l x_4^l$	x_5^l	x_6^l	x_7^l	x_8^l	x_9^l	x_1^r	x_2^r	x_3^r	x_4^r	x_5^r	x_6^r	x_7^r	x_8^r	x_9^r			

TABLE VI
CONSTRUCTION F_2 —MAPPING SCHEME

	source word X	y_1^l	y_2^l	y_3^l	y_4^l	y_5^l	y_6^l	y_7^l	y_8^l	y_9^l	y_1^r	y_2^r	y_3^r	y_4^r	y_5^r	y_6^r	y_7^r	y_8^r	y_9^r
$w_{1,1,1}$	0 0 0 0 0 0 0 0 0 x_7^r	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{0,1,1}$	$x_1^l x_2^l x_3^l$ 0 0 0 0 0 0 0 0 x_7^r	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{1,0,1}$	0 0 0 $x_4^l x_5^l x_6^l$ 0 0 0 0 x_7^r	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{1,1,0}$	0 0 0 0 0 0 0 $x_4^r x_5^r x_6^r x_7^r$	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{1,0,0}$	0 0 0 $x_4^l x_5^l x_6^l x_7^l x_8^l x_9^l$	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{0,1,0}$	$x_1^l x_2^l x_3^l$ 0 0 0 $x_4^r x_5^r x_6^r x_7^r$	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{0,0,1}$	$x_1^l x_2^l x_3^l x_4^l x_5^l x_6^l$ 0 0 0 x_7^r	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
$w_{0,0,0}$	$x_1^l x_2^l x_3^l x_4^l x_5^l x_6^l x_7^l x_8^l x_9^l$	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>

(0, k , k_l , k_r) code, which has the following structure in the case where $n = n_{F_2} = 3k + k_l + k_r - 3$:

$$\square^{k_l-2} \boxtimes \boxtimes \boxtimes \square^{k-4} \boxtimes \boxtimes \boxtimes \square^{k-4} \boxtimes \boxtimes \boxtimes \square^{k-3} \boxtimes \boxtimes \square^{k_r-1}.$$

There are $1029 > 2^{10}$ valid constrained sequences of length 11. The underlying rate 10/11 CIC code is specified in Table VI.

Table VI uniquely specifies the operation of the encoder and the decoder. The combinatorial circuitry and the logic equations can be easily obtained with hardware design tools. The number of gates is low.

H. Construction G

A rate 11/12 CIC code with $\kappa = [2, 3, 3, 2, 3, 3, 2, 2, 2, 2, 1]$ can be used to construct a rate $(n-1)/n$, (0, k , k_l , k_r) code, which has the following structure in the case where $n = n_G = 3k + k_l + k_r$:

$$\square^{k_l-2} \boxtimes \boxtimes \boxtimes \square^{k-3} \boxtimes \boxtimes \boxtimes \square^{k-3} \boxtimes \boxtimes \boxtimes \square^{k-2} \boxtimes \boxtimes \boxtimes \square^{k_r-2}.$$

There are $2063 > 2^{11}$ valid constrained sequences of length 12. The CIC code can be constructed in a similar fashion as for the

construction of high-rate constrained codes with limited error propagation.

REFERENCES

- [1] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
- [2] P. H. Siegel and J. K. Wolf, "Modulation and coding for information storage," *IEEE Commun. Mag.*, vol. 29, no. 12, pp. 68–86, Dec. 1991.
- [3] H. Morita, A. J. van Wijngaarden, and A. J. Han Vinck, "On the construction of maximal prefix-synchronized codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2158–2166, Nov. 1996.
- [4] C. D. Mee and E. D. Daniel, *Magnetic Recording*. New York: McGraw-Hill, 1987.
- [5] A. J. van Wijngaarden and E. Soljanin, "A combinatorial technique for constructing high rate MTR–RLL codes," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 582–588, Apr. 2001.
- [6] D. Bertsekas and R. G. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [7] G. Kabatjanskii, A. J. Han Vinck, and A. J. van Wijngaarden, "On combined synchronization and error control coding," in *Proc. 1994 IEEE Int. Symp. on Information Theory*, June 1994, p. 62.
- [8] W. H. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 284–292, Apr. 1965.
- [9] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 73–77, Jan. 1973.
- [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. on Inform. Theory*, Ulm, Germany, Sept. 1997, p. 144.
- [11] —, "Construction of constrained codes using sequence replacement techniques," *IEEE Trans. Inform. Theory*, to be published.
- [12] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1389–1399, Sept. 1997.
- [13] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [14] C. D. Mee and E. D. Daniel, *Magnetic Storage Handbook*. New York: McGraw-Hill, 1996.
- [15] K. A. S. Immink, *Codes for Mass Data Storage Systems*. The Netherlands: Shannon Foundation, 1999.
- [16] K. A. S. Abdel-Ghaffar, M. Blaum, and J. H. Weber, "Analysis of coding schemes for modulation and error control," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1955–1968, Nov. 1995.
- [17] A. M. Patel, "Improved encoder and decoder for a byte-oriented (0, 3) 8/9 code," *IBM Technol. Discl. Bull.*, vol. 28, pp. 1938–1940, Oct. 1985.
- [18] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Technol. Discl. Bul.*, vol. 23, pp. 4633–4634, 1981.
- [19] B. Fitingof and M. Mansuripur, "Method and apparatus for implementing post-modulation error correction coding scheme," U.S. Patent 5,311,521, May 1994.
- [20] J. L. Fan and A. R. Calderbank, "A modified concatenated coding scheme with applications to magnetic data storage," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1565–1574, July 1998.
- [21] A. J. van Wijngaarden and K. A. S. Immink, "Combinatorial construction of high rate runlength-limited codes," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM '96*, Nov. 1996, pp. 343–347.
- [22] K. A. S. Abdel-Ghaffar and J. H. Weber, "Constrained block codes for Class-IV partial-response channels with maximum-likelihood sequence estimation," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1405–1424, Sept. 1996.
- [23] J. S. Eggenberger and A. M. Patel, "Method and apparatus for implementing optimum PRML codes," U.S. Patent 4,707,681, Nov. 1987.
- [24] A. M. Patel, "Rate 16/17 (0,6/6) code," *IBM Technol. Discl. Bull.*, vol. 31, no. 8, 1989.



Adriaan J. (de Lind) van Wijngaarden (S'87–M'98) was born in Amsterdam, The Netherlands, in 1968. He received the engineering (M.Sc. equivalent) degree in electrical engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1992, and the doctorate degree in engineering from the University of Essen, Essen, Germany, in 1998.

From 1992 to 1998, he was a Research Engineer of the Digital Communications Group at the Institute for Experimental Mathematics, University of Essen.

During this period, he has also been a Visiting Scientist at the Communications Research Centre, Ottawa, Ont. Canada, and at the University of Electro-Communications, Tokyo, Japan. As of 1998, he is with the Mathematics of Communications Research Department, Bell Laboratories, Lucent Technologies, Murray Hill, NJ. His research interests include communications, in particular, error control coding, synchronization and modulation coding with applications in recording, and optical and wireless communication systems.



Kees A. Schouhamer Immink (M'81–SM'86–F'90) was born in Rotterdam, The Netherlands. He received the M.Sc. (EE) and Ph.D. degrees from the Eindhoven University of Technology.

He is President and Founder of Turing Machines Inc. and serves as a Guest Professor at the Institute of Experimental Mathematics, Essen University, Germany, and the National University of Singapore. He has contributed to the design and development of a variety of digital recorders such as the compact disk, compact disk video, DAT, DCC, and recently, the DVD. Immink was granted 35 U.S. patents, is (co)author of three books and has written numerous papers in the field of digital recorders.

Dr. Immink is Vice-President of the Audio Engineering Society (AES), a Governor of the IEEE Consumer Electronics Society, and Trustee of the Shannon Foundation. He served as Program Chairman and Conference Chairman of various international conferences over the years. He holds fellowships of the AES, IEE, and SMPTE, and is an elected member of the Royal Netherlands Academy of Arts and Sciences. For his part in the digital audio and video revolution, he was honored with the AES Gold and Silver Medal, IEE Thomson Medal, SMPTE Ponyatoff Gold Medal, IEEE Information Theory Society Golden Jubilee Award for Technical Innovation, IEEE Masaru Ibuka consumer electronics award, a Knighthood in the Order of Orange-Nassau, and the IEEE Edison Medal "For a career of creative contributions to the technologies of video, audio, and data recording."