

Combinatorial Construction of High Rate Runlength-limited Codes

A.J. van Wijngaarden and K.A. Schouhamer Immink *

Institute for Experimental Mathematics, University of Essen, Ellernstr. 29, 45326 Essen, Germany

* Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

Abstract — New combinatorial construction techniques are proposed which convert binary user information into a $(0, k)$ constrained sequence having the virtue that at most k ‘zeroes’ between logical ‘ones’ will occur. In this way sequences are constructed which have a limited runlength. These codes find application in optical and magnetic recording systems. The new construction methods provide efficient, high rate codes with a low complexity. The low complex combinatorial structure of the encoder and the decoder ensure a very fast and efficient parallel conversion of binary information to code words and vice versa. Specifically, we present the combinatorial structures to convert 16 data bits into a 17 bit constrained sequence to obtain an optimum $(0, 4)$ code, a $(0, 6)$ code with at most one byte error propagation, and a $(0, 6/6)$ -code, respectively. Serious error propagation is avoided by using constrained codes with several unconstrained positions, which are reserved to store the parity bits of an error control code which protects the constrained code word.

I. INTRODUCTION

A runlength-limited sequence is a sequence of binary symbols characterized by two parameters, d and k , which constrain the minimum and maximum runlength of consecutive like symbols in the sequence to $(d + 1)$ and $(k + 1)$ respectively. Runlength-limited codes are widely used in optical and magnetic recording systems [1; 2; 3]. The parameter d controls the highest transition frequency, and the maximum runlength parameter k ensures adequate frequency of transitions for synchronization of the read clock. Sequence construction usually takes place by representing the sequence by its transitions. Each transition is represented by a one. In this case there should be at least d and at most k zeroes between two consecutive ones. This code is referred to as a (d, k) -code.

In recent years, interest has grown in the application of $(0, k)$ codes with a small redundancy for magnetic recording systems. In these systems, the code words are usually concatenated. In order to maintain the runlength constraints between consecutive code words, the number of leading and trailing zeroes of a code word has been restricted to l and r , with $l + r \leq k$. The corresponding codes will be referred to as $(0, k, l, r)$ -codes. Several $(0, k)$ -codes and $(0, k, l, r)$ -codes have been constructed in the past, using look-up tables or simple combinatorial circuitry [1]. Examples of codes employed in magnetic recording systems are the rate $4/5$, Group-Coded Recording [4] and the rate $8/9$, $(0, 3)$ code [1; 5]. Other $(0, k)$ -constrained codes with a rate higher than $8/9$ have, to the authors’ knowledge, not been used in recording systems. It is not unrealistic to say that the presumed hardware requirements of high-rate codes have hampered their

introduction thus far.

As an alternative to combinatorial constructions, enumerative constructions could be used [7; 8]. These have thus far been proposed for the construction of prefix synchronized codes [6; 9]. A disadvantage of enumerative coding is the bitwise conversion of data words into code words of length n and vice versa, and the need of comparisons and additions with n -bit coefficients.

We consider the combinatorial construction of $(0, k)$ -codes, and show that it is possible to construct runlength-limited codes with a very high rate for relevant values of k . We first develop a method to determine the maximum code size of $(0, k)$ -codes and $(0, k, l, r)$ -codes. Three techniques are presented to construct high rate constrained codes with very low coding complexity. In addition, we will consider the issue of propagation of errors and propose efficient methods to overcome corresponding problems by reversing the order of application of error control coding and modulation coding. Finally, we consider application of the three new techniques to the construction of so-called $(0, G/I)$ -codes [3], where a $(0, I)$ constraint is imposed on the subsequences of odd and even positions, and a $(0, G)$ constraint on the overall code word.

II. CAPACITY OF CONSTRAINED CODES

In order to determine the cardinality of $(0, k)$ -codes of word length n , generalized Fibonacci sequences can be used, while for asymptotic values the corresponding generating functions can be employed.

A $(0, k)$ constrained code of length n , denoted by $\mathcal{C}^n(0, k)$ is a subset of \mathcal{A}_2^n , the set of binary sequences of length n . Let $\mathcal{F}_k^{(m)}$ denote the set of constrained sequences of length m in which runs of more than k zeroes do not occur as a subsequence. The cardinality of $\mathcal{F}_k^{(m)}$ is denoted by $F_k^{(m)}$. Obviously, the cardinality of $\mathcal{C}^n(0, k)$ is at most $F_k^{(n)}$. For a sequence $P \in \mathcal{A}_2^p$ and a set of sequences $\mathcal{S} \subset \mathcal{A}_2^s$, the set $PS = \{PX | X \in \mathcal{S}\}$ denotes the set of concatenated sequences of length $p + s$ with prefix P . A sequence of length 0 is denoted by ϕ , for which $X\phi = \phi X = X$ holds for any sequence X . Let α^m denote a (sub)sequence of m consecutive symbols α , $\alpha \in \mathcal{A}_2$.

For $m \leq k$, $\mathcal{F}_k^{(m)} = \mathcal{A}_2^m$, and $\mathcal{F}_k^{(k+1)} = \mathcal{A}_2^m \setminus \{0^{k+1}\}$. For any $m > k + 1$, the following relation holds.

$$\mathcal{F}_k^{(m)} = \bigcup_{i=0}^k 0^i 1 \mathcal{F}_k^{(m-i-1)} \quad (1)$$

Since this union of subsets is non-intersecting, the following recursive expression for the cardinality of the $\mathcal{F}_k^{(m)}$ can be immediately obtained.

$$F_k^{(m)} = \begin{cases} 2^m & 0 \leq m \leq k \\ 2^m - 1 & m = k + 1 \\ 2 F_k^{(m-1)} - F_k^{(m-k-2)} & m \geq k + 2. \end{cases} \quad (2)$$

To determine the cardinality of a $(0, k, l, r)$ -code $G_{k,l,r}^{(m)}$, it is useful to write this set as follows.

$$G_{k,l,r}^{(m)} = \bigcup_{i=0}^l \bigcup_{j=0}^r 0^i 1 \mathcal{F}_k^{(m-i-j-2)} 10^j \quad (3)$$

The cardinality of this code, denoted by $G_{k,l,r}^{(m)}$, equals

$$G_{k,l,r}^{(m)} = \sum_{i=0}^l \sum_{j=0}^r F_k^{(m-i-j-2)} \quad (4)$$

Since each term $F_k^{(m)}$ is non-negative, it is easy to show that the cardinality of a $(0, k, l, r)$ -code is maximized for $l = \lfloor k/2 \rfloor$, and $r = k - l$.

For practical applications, it is often not necessary to use the entire code set. Usually, an m -bit data word has to be converted into a code word of length n . This implies that the inequality $2^m \leq C^n(0, k)$ should be fulfilled. We will consider the construction of encoding methods which are capable to convert an $(n-1)$ -bit data word to an n -bit constrained code word. The redundancy R is therefore equal to $1/n$. For given k , the maximum code word length n for which a constrained code with redundancy $1/n$ exists is shown in Table I. The corresponding redundancy R is specified as well.

Table I. Maximum code length for given k , for which a constrained code exists with at least 2^{n-1} code words ($l = \lfloor k/2 \rfloor$, and $r = k - l$).

k	$(0, k)$ -code		$(0, k, l, r)$ -code	
	n_k	R_k	$n_{k,l,r}$	$R_{k,l,r}$
2	9	$1.1 \cdot 10^{-1}$	5	$2.0 \cdot 10^{-1}$
3	21	$4.8 \cdot 10^{-2}$	12	$8.3 \cdot 10^{-2}$
4	43	$2.3 \cdot 10^{-2}$	31	$3.2 \cdot 10^{-2}$
5	88	$1.1 \cdot 10^{-2}$	67	$1.5 \cdot 10^{-2}$
6	177	$5.6 \cdot 10^{-3}$	148	$6.8 \cdot 10^{-3}$

III. ERROR CONTROL ASPECTS

Data protected by an error control code will be vulnerable for errors after conversion into a constrained code word, since the distance properties of the error control code are partly lost after conversion. The effect of errors in the modulated code can be reduced by applying error control codes which protect the constrained code word. A constrained code can be constructed where each code word has the same fixed, unconstrained positions, i.e., the data values at these positions can be changed without violating the code constraints. After the data word is converted into the constrained code word,

the parity check bits are computed for the constrained code word, and stored at the unconstrained positions. At the receiver side, the decoder first checks the received code word for errors, and then decodes the constraint code word. In this way the distance properties of the error control code will be maintained.

IV. COMBINATORIAL CONSTRUCTION METHODS

Combinatorial constructions can be used to obtain $(0, k)$ -constrained codes and $(0, k, l, r)$ -codes. We present three different types: block constrained codes, sub-block constrained codes, and interleaved codes. The first construction gives constrained codes with minimum k , while the other two techniques are particularly suited to be used in combination with the error control scheme described in section III.

A. Block constrained codes

Block constrained codes map an $(n-1)$ -bit data word onto an n -bit constrained $(0, k)$ -code. If $C^n(0, k) \geq 2^{n-1}$, it is always possible to perform this mapping, though the complexity may be high. For several code word lengths, it turns out to be possible to perform this mapping procedure using only a small amount of combinatorial circuitry. As a specific example, a rate 16/17, $(0, 4, 2, 2)$ code, having a minimum k -constraint (Table I), has been constructed using a symmetric combinatorial structure, which will be presented in Section V. These constructions have the following characteristics:

- a symmetric data mapping structure is used to simplify the combinatorial structure and the design process.
- one central position in the code word, y_c , is used to indicate whether or not the data sequence violates the constraints.
- in case the constraints are not violated, leave the data sequence unchanged. Position y_c is set to 1, to indicate that no violations occurred.
- in case a violation occurs, this is caused by several data bits being zero. These bits are replaced by a short code which identifies the type of the violation, and by some rearranged data bits. In this way a sequence is formed which fulfills the constraints. Position y_c is set to 0 to indicate that a violation occurred.

This construction technique makes it relatively simple to construct constrained codes with a minimum k constraint. The complexity of constructing a rate $(n-1)/n$, $(0, k)$ -constrained codes increases significantly if the maximum code size $G_{k,l,r}^{(n)}$ is only slightly larger than 2^{n-1} .

B. Interleaved codes

One option to avoid that the mapping technique changes all data symbols is to use an interleaving technique. Assume that a $(0, k, l, r)$ code of length n is available. If i arbitrary bits are inserted between each pair of consecutive positions of the $(0, k, l, r)$ -code, the resulting $(0, k_i, l_i, r_i)$ code of length

n_i and rate $(1 - 1/n_i)$ is constructed, where

$$n_i = (n - 1) \cdot i + n$$

$$k_i = (k + 1) \cdot i + k$$

$$l_i = l \cdot i + l$$

$$r_i = r \cdot i + r$$

Several of the unconstrained bits could be used to protect the constrained $(0, k)$ code word and the other constrained positions. In this way error protection is ensured *after* data mapping. For larger values of i , the resulting code is far from optimum with respect to the k constraint, but for small i reasonably good codes can be very easily constructed, and error propagation effects can be easily suppressed.

C. Sub-block constrained codes

The third option is to use subblocks in the code word to ensure the $(0, k)$ constraints, while the other positions are unconstrained.

Let B denote the number of subblocks, and b_i be the number of positions of the i -th subblock. Then the following relation holds for the cardinality C_B of this B -block code

$$C_B \leq 2^u \prod_{i=1}^B (2^{b_i} - 1) \quad (5)$$

where u denotes the number of unconstrained positions (the positions which do not belong to one of the subblocks). As an example, consider a 3-block code, as shown in Figure 1.

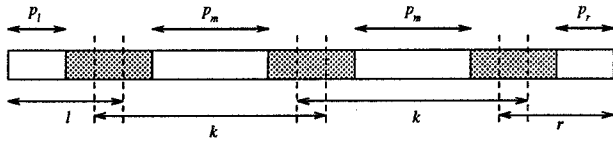


Fig. 1. General construction of a 3-block $(0, k)$ code

Three blocks of three bits each are used to fulfill the $(0, k)$ -constraints, irrespective of the value of the unconstrained data positions surrounding the blocks. This can be achieved if none of the sub-blocks contains the 3-tuple $(0, 0, 0)$. The number of unconstrained positions between the blocks is denoted by p_m , on the lefthand side by p_l and on the righthand side by p_r . These three parameters determine the actual constraints as follows:

$$n = 2 \cdot p_m + p_l + p_r + 9$$

$$k \leq p_m + 4$$

$$l \leq p_l + 2$$

$$r \leq p_r + 2$$

There are $7 \times 7 \times 7 = 343$ code words, of which only 256 code words are needed. This gives some opportunity to further reduce the maximum runlength between two blocks.

The 3-block $(0, k)$ -code can be realized using the symmetric mapping scheme shown in Table II. The symmetry is

shown by splitting the data word and the code word into a left and a right part, which is denoted by superscripts l, r . The 8 data symbols are thus grouped into two subvectors (x_1^l, \dots, x_4^l) and (x_4^r, \dots, x_1^r) , which are mapped onto 9 output positions $(y_1^l, \dots, y_4^l), y_c, (y_4^r, \dots, y_1^r)$. If the tuple (x_1^l, x_2^l, x_3^l) , and (x_3^r, x_2^r, x_1^r) are both unequal to $(0, 0, 0)$, the data bits are left unchanged and y_c is set to 1 (see last pair of rows in Table II). In this way, 196 out of 2^8 code words are generated without effort. The remaining 60 code words are formed according to the mapping scheme specified in Table II. The first three pairs of rows in Table II show how data is mapped in case one or more 3-tuples are $(0, 0, 0)$. With this 3-block code, $(0, k)$ -constrained codes can be constructed for which $n = 3k - 1$.

As a specific example, a rate 16/17, $(0, 6, 3, 3)$ -code can be constructed by choosing $p_m = 3$, and $p_l = p_r = 1$. Each code word has 8 unconstrained bits. If one of the 17 bits is erroneously received, error propagation will in any case be limited to one byte, which makes this code very useful in byte-oriented systems. Since a $(0, 5, 2, 3)$ -code of length 17 with 8 unconstrained positions has at most $58850 \leq 2^{16}$ code words, this code has the minimum k constraint among codes of length 17 with 8 unconstrained positions.

Table II. Mapping scheme of a 3-block $(0, k)$ -code.

y_1^l	y_2^l	y_3^l	y_4^l	y_c	y_4^r	y_3^r	y_2^r	y_1^r
0	0	0	x_4^l	1	x_4^r	0	0	0
x_4^l	x_4^r	1	1	0	1	1	x_4^l	x_4^r
0	0	0	x_4^l	1	x_4^r	x_3^r	x_2^r	x_1^r
x_4^l	x_4^r	1	0	0	1	x_3^r	x_2^r	x_1^r
x_1^l	x_2^l	x_3^l	x_4^l	1	x_4^r	0	0	0
x_1^l	x_2^l	x_3^l	1	0	0	1	x_4^l	x_4^r
x_1^l	x_2^l	x_3^l	x_4^l	1	x_4^r	x_3^r	x_2^r	x_1^r
x_1^l	x_2^l	x_3^l	x_4^l	1	x_4^r	x_3^r	x_2^r	x_1^r

Another example of a sub-block constrained codes is to use a 5-block code. The mapping scheme for this code is given in Table III. The number of possible code words equals $7 \times 15 \times 7 \times 15 \times 7 = 77175 > 2^{16}$. The mapping technique is similar as in Table II. For reasons of symmetry, it is only shown how a violation of the constraints on the lefthand side is remapped. If violations occur on both sides, the centre positions (y_8^l, y_c, y_8^r) will become $(1, 0, 1)$.

Table III. Mapping scheme for a 5-block $(0, k)$ -code

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_c	y_8^r	y_7^r	y_6^r	y_5^r	y_4^r	y_3^r	y_2^r	y_1^r
0	0	0	0	0	0	0	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
1	0	0	1	x_8^l	x_8^r	1	0	0	1	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
0	0	0	x_4^l	x_5^l	x_6^l	x_7^l	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
x_8^l	x_8^r	1	x_4^l	x_5^l	x_6^l	x_7^l	0	0	1	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
x_1^l	x_2^l	x_3^l	0	0	0	0	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
x_8^l	1	0	x_1^l	x_2^l	x_3^l	x_8^r	0	0	1	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r

Instead of using a combinatorial construction of the sub-blocks, it is possible to represent an arbitrary data word by B

tuples, which are all non-zero. Since $0 \leq x_i \leq 2^{b_i} - 1$, block i will not contain zeroes only, if it will be filled with the inverted binary representation of x_i . Consider for example a 3-block code, where an arbitrary data word with index x , $0 \leq x < 2^8$, can be uniquely represented by the 3-tuple (x_1, x_2, x_3) , for which $x_3 = \lfloor \frac{x}{49} \rfloor$, $x_2 = \lfloor \frac{x}{7} \rfloor - 7x_3$, and $x_1 = x - 49x_3 - 7x_2$.

V. SPECIFIC COMBINATORIAL CONSTRUCTIONS

For practical applications, it is often useful to consider data to be stored as bytes. For this reason, it is interesting to consider mapping techniques for data word sequences of which the length is a multiple of 8. Only the rate 8/9, (0, 3)-code has been considered thus far [5]. We will consider the previously proposed construction techniques for the conversion of a two byte data word into a constrained code word of length 17.

Using the interleaving technique, an arbitrary rate 8/9, (0, 3, 1, 2)-code, e.g., [5], can be used to obtain a rate 16/17, (0, 7, 2, 4)-code. Using the 3-block conversion technique, a rate 16/17, (0, 6, 3, 3)-code can be constructed. Finally, we will now show that it is possible to obtain a rate 16/17, (0, 4, 2, 2)-code using the following (symmetric) block conversion method. This code is optimum in the sense that it is not possible to construct a concatenable rate 16/17, (0, k)-code with $k < 4$.

The rate 16/17, (0, 4, 2, 2) code

We will now present the structure of the encoder and the decoder of the rate 16/17, (0, 4, 2, 2) code. Due to the symmetric structure we represent the data sequence by two 8-bit sequences $x_1^l \dots x_8^l$ and $x_1^r \dots x_8^r$. The mapping scheme is shown in Table IV. Each pair of rows indicates a constraint violation, denoted by v_1, \dots, v_5 , and the corresponding data mapping to fulfill the constraints. The last violation, v_c^l , occurs if a violation occurs on the righthand side.

Table IV. Mapping scheme of the rate 16/17, (0, 4, 2, 2)-code of length 17.

v_1^l	0 0 0 0 0 0 0 0	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r						
	1 0 0 0 0 1	x_8^l	x_8^l	1	0	v_6^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r				
v_2^l	0 0 0 0	x_4^l	x_5^l	x_6^l	x_7^l	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r		
	x_8^r	1	x_8^l	x_4^l	x_5^l	x_6^l	x_7^l	1	0	v_6^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
v_3^l	1 0 0 0 0 0 0 0	x_7^l	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r					
	1 0 0 0 1	x_8^l	x_8^l	x_7^l	1	0	v_6^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r			
v_4^l	x_1^l	1 0 0 0 0 0 0 0	x_8^l	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r					
	x_1^l	0 1 1 1	x_8^l	x_8^l	1	0	v_6^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r			
v_5^l	x_1^l	x_2^l	1 0 0 0 0 0 0 0	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r					
	x_1^l	0 1 0 1	x_8^l	x_2^l	1	0	v_6^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r			
v_c^l	x_1^l	x_2^l	x_3^l	x_4^l	0 0 0 0 0 0 0 1	y_7^r	y_6^r	y_5^r	y_4^r	y_3^r	y_2^r	y_1^r					
	x_1^l	x_2^l	x_3^l	x_4^l	y_7^r	1 0 0 0 0 0 0 1	y_6^r	y_5^r	y_4^r	y_3^r	y_2^r	y_1^r					

The following expressions identify the type of violation:

$$\begin{aligned} v_1 &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 \bar{x}_7 \\ v_2 &= \bar{x}_1 \bar{x}_2 \bar{x}_3 (x_4 + x_5 + x_6 + x_7) \\ v_3 &= x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 \end{aligned}$$

$$\begin{aligned} v_4 &= x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 \bar{x}_7 \\ v_5 &= x_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 \bar{x}_7 \bar{x}_8 \\ v_6 &= \bar{v}_1 + \bar{v}_2 + \bar{v}_3 + \bar{v}_4 + \bar{v}_5 \\ v_c &= \bar{x}_5 \bar{x}_6 \bar{x}_7 \bar{v}_6 v_6^s \end{aligned}$$

where v_6 is used to detect the presence of a violation. Note that the indication of the left and right side are omitted. A symbol of the opposite side will be identified by the superscript s . The outputs $y_1 \dots y_8$ can be immediately derived. The following expressions have been obtained.

$$\begin{aligned} y_1 &= \bar{v}_2 x_1 + v_1 + v_2 x_8^s + v_3 \\ y_2 &= \bar{v}_4 \bar{v}_5 x_2 + v_2 \\ y_3 &= \bar{v}_2 \bar{v}_4 x_3 + v_2 x_8 \\ y_4 &= x_4 + v_3 + v_4 \\ y_5 &= \bar{v}_3 \bar{v}_c x_5 + v_1 + v_3 x_8 + v_4 + v_5 + v_c y_7^s \\ y_6 &= \bar{v}_1 \bar{v}_3 x_6 + \bar{v}_2 v_6 x_8^s + v_c \\ y_7 &= \bar{v}_1 \bar{v}_4 \bar{v}_5 x_7 + (v_1 + v_4) x_8 + v_5 x_2 + v_c^s \\ y_8 &= x_8 + v_6 \\ y_c &= \bar{v}_6 \bar{v}_6^s \end{aligned}$$

Similarly, expressions have been derived for the decoder which extracts the data bits $z_1 \dots z_8$ from the code words.

$$\begin{aligned} v_1 &= \bar{y}_2 \bar{y}_3 \bar{y}_4 y_8 \bar{y}_c \\ v_2 &= y_2 y_8 \bar{y}_c \\ v_3 &= \bar{y}_2 \bar{y}_3 y_4 y_8 \bar{y}_c \\ v_4 &= \bar{y}_2 y_3 y_4 y_8 \bar{y}_c \\ v_5 &= \bar{y}_2 \bar{y}_3 \bar{y}_4 y_8 \bar{y}_c \\ v_6 &= y_8 \bar{y}_c \\ v_c &= \bar{y}_7 \bar{y}_8 \bar{y}_c \\ z_1 &= \bar{v}_2 y_1 \\ z_2 &= \bar{v}_2 \bar{v}_5 y_2 + v_4 + v_5 y_7 \\ z_3 &= \bar{v}_2 \bar{v}_4 y_3 \\ z_4 &= \bar{v}_3 \bar{v}_4 y_4 \\ z_5 &= \bar{v}_1 \bar{v}_3 \bar{v}_4 \bar{v}_5 \bar{v}_c y_5 \\ z_6 &= \bar{v}_1 \bar{v}_3 \bar{v}_4 \bar{v}_5 \bar{v}_c y_6 \\ z_7 &= \bar{v}_1 \bar{v}_4 \bar{v}_5 \bar{v}_c^s y_7 + v_c^s y_5^s \\ z_8 &= y_c x_8 + (v_1 + v_4) y_7 + v_2 y_3 + \bar{v}_2^s v_6^s y_6^s + v_2^s y_1^s \end{aligned}$$

The encoder can be realized with combinatorial circuitry of 128 gate equivalents, having a propagation delay of about 9.8 ns. The decoder can be realized with 107 gate equivalents, having a propagation delay of about 9.7 ns.

VI. CONSTRUCTION OF (0, G/I) CODES

The construction of a (0, G/I) code [3; 10] is in general more difficult than the construction of a (0, k) code, since both the global runlength constraint, and the constraint for the even and odd subsequences have to be fulfilled. In [11], a combinatorial construction method of a rate 8/9, (0, 4/4) code and a rate 8/9, (0, 3/6) code have been developed. To the authors' knowledge, (0, G/I) codes with a rate $> 8/9$ have not been used thus far.

With several modifications, it is possible to use the pre-

viously discussed sub-block and block constrained code construction. As an example, we present a rate 16/17, (0, 6/6)-code using the block constrained code technique.

The rate 16/17, (0, 6/6) encoder

A rate 16/17, (0, 6/6) code has been constructed using a symmetric combinatorial mapping according to Table V. The central position in the code word, y_c , is used to indicate whether or not a conversion of the data is necessary.

Table V. Mapping scheme of the rate 16/17, (0, 6/6)-code of length 17.

	y_1^i	y_2^i	y_3^i	y_4^i	y_5^i	y_6^i	y_7^i	y_8^i	y_c	y_8^r	y_7^r	y_6^r	y_5^r	y_4^r	y_3^r	y_2^r	y_1^r
v_1^i	0	0	0	0	0	x_6^i	0	x_8^i	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
	1	x_7^r	0	1	0	x_6^i	v_5^r	x_8^i	0	x_8^r	1	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
v_2^i	x_1^i	0	x_3^i	0	x_5^i	0	x_7^i	0	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
	x_7^r	x_1^r	1	x_3^r	1	x_5^i	v_5^r	x_7^i	0	x_8^r	1	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
v_3^i	0	x_2^i	0	x_4^i	0	x_6^i	0	x_8^i	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
	x_7^r	x_2^r	0	x_4^r	1	x_6^i	v_5^r	x_8^i	0	x_8^r	1	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
v_4^i	0	0	0	0	x_5^i	x_6^i	x_7^i	x_8^i	1	x_8^r	x_7^r	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r
	x_7^r	x_6^i	1	x_8^i	0	x_5^i	v_5^r	x_7^i	0	x_8^r	1	x_6^r	x_5^r	x_4^r	x_3^r	x_2^r	x_1^r

The mapping function of the encoder can be achieved using combinatorial circuitry. The following combinatorial description allows the detection for the presence of a violation, v_5 , and the violation types v_1, v_2, v_3 , or v_4 :

$$\begin{aligned} v_1 &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_7 \\ v_2 &= \bar{x}_2 \bar{x}_4 \bar{x}_6 \bar{x}_8 (x_1 + x_3) \\ v_3 &= \bar{x}_1 \bar{x}_3 \bar{x}_5 \bar{x}_7 (x_2 + x_4) \\ v_4 &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 (x_5 + x_7) \\ v_5 &= \bar{v}_1 + \bar{v}_2 + \bar{v}_3 + \bar{v}_4 \end{aligned}$$

The encoder outputs y_1, \dots, y_8 can now be immediately derived. The following expressions have been obtained.

$$\begin{aligned} y_1 &= \bar{v}_5 x_1 + v_5 x_7^s + v_1 \\ y_2 &= (\bar{v}_5 + v_3) x_2 + v_2 x_1 + v_4 x_6 + v_1 x_7^s \\ y_3 &= \bar{v}_5 x_3 + v_2 + v_4 \\ y_4 &= (\bar{v}_5 + v_3) x_4 + v_1 + v_2 x_3 + v_4 x_8 \\ y_5 &= \bar{v}_5 x_5 + v_2 + v_3 \\ y_6 &= (\bar{v}_5 + v_1 + v_3) x_6 + (v_2 + v_4) x_5 \\ y_7 &= \bar{v}_5 x_7 + v_5^s \\ y_8 &= (\bar{v}_5 + v_1 + v_3) x_8 + (v_2 + v_3) x_7 \\ y_c &= \bar{v}_5 \bar{v}_5^s \end{aligned}$$

The rate 16/17, (0, 6/6) decoder

The objective of the decoder is to perform the inverse mapping procedure. The approach is similar as for the encoder. First the type of violation is determined, and next the data is recovered.

$$\begin{aligned} v_1 &= \bar{y}_3 \bar{y}_5 \bar{y}_c y_7^s \\ v_2 &= y_3 y_5 \bar{y}_c y_7^s \\ v_3 &= \bar{y}_3 y_5 \bar{y}_c y_7^s \\ v_4 &= y_3 \bar{y}_5 \bar{y}_c y_7^s \end{aligned}$$

$$v_5 = \bar{y}_c y_7^s$$

The decoder outputs z_1, \dots, z_8 can now be derived:

$$\begin{aligned} z_1 &= \bar{v}_5 y_1 + v_2 y_2 \\ z_2 &= (\bar{v}_5 + v_3) y_2 \\ z_3 &= \bar{v}_5 y_3 + v_2 y_4 \\ z_4 &= (\bar{v}_5 + v_3) y_4 \\ z_5 &= \bar{v}_5 y_5 + (v_2 + v_4) y_6 \\ z_6 &= \bar{v}_2 \bar{v}_4 y_6 + v_4 y_2 \\ z_7 &= y_c y_7 + (v_2 + v_4) y_8 + v_5^s (v_1^s y_2^s + \bar{v}_1^s y_1^s) \\ z_8 &= \bar{v}_2 \bar{v}_4 y_8 + v_4 y_4 \end{aligned}$$

The encoder can be realized with combinatorial circuitry of 97 gate equivalents, having a propagation delay of the critical path in the order of 7.5 ns. Similarly, the decoder can be realized with combinatorial circuitry of 77 gates equivalents, having a propagation delay of about 3.5 ns.

VII. CONCLUSIONS

Combinatorial techniques show to be very useful for the construction of (0, k)-codes, (0, k, l, r)-codes, and (0, G/I)-codes. The redundancy and the constraints are optimal or close to optimal. Main advantages of the combinatorial structure are the possibility of a parallel conversion of data into constrained code words and vice versa. The complexity of the combinatorial structure tends to be low, and the encoder and decoder will be very fast, due to the combinatorial, parallel structure. We have shown that additional error protection can be achieved by using a structure for which it is possible to reverse the order of application of the error control code and the runlength-limited code.

REFERENCES

- [1] C. Mee and E. Daniel, *Magnetic Recording*. McGraw-Hill, New York, 1987.
- [2] K.A.S. Immink, *Coding Techniques for Digital Recorders*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.
- [3] P. Siegel and J. Wolf, "Modulation and coding for information storage," *IEEE Comm. Magazine*, vol. 29, pp. 68-86, Dec. 1991.
- [4] G. Milligan, "Self-clocking five bit record playback system," *US Patent 3,641,525*, Feb. 1972.
- [5] A. Patel, "Improved encoder and decoder for a byte-oriented (0, 3) s_8 code," *IBM Techn. Discl. Bull.*, vol. 28, pp. 1938-1940, Oct. 1985.
- [6] E. Gilbert, "Synchronization of binary messages," *IRE Trans. Inform. Theory*, pp. 470-477, Sept. 1960.
- [7] W. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inform. Theory*, pp. 284-292, Apr. 1965.
- [8] D. Mandelbaum, "Synchronization of codes by means of Kautz's Fibonacci encoding," *IEEE Trans. Inform. Theory*, vol. 18, pp. 281-285, Mar. 1972.
- [9] H. Morita, A.J. van Wijngaarden, and A.J. Han Vinck, "On the construction of maximal prefix-synchronized codes," *IEEE Trans. Inform. Theory*, Nov. 1996 (to appear).
- [10] K. Abdel-Ghaffar and J. Weber, "Constrained block codes for Class-IV partial-response channels with maximum-likelihood sequence estimation," *IEEE Trans. Inform. Theory*, Mar. 1995 (submitted).
- [11] J. Eggenberger and A. Patel, "Method and apparatus for implementing optimum PRML codes," *US Patent 4,707,681*, Nov. 1987.