The decoding function can be accomplished with a simple logic array. Note that the code above can easily be transformed into a \((d = 1, k = 11)\) RLL code by representing the source word "2" by "0000000" or "1111111." If \(x_5 x_6 = 00\), then represent the source word "2" by "000000." It should be appreciated that the smallest block-decodable conventional rate \(2/3\) \((d = 1, k = 11)\) code requires a codeword length of \(n = 18\). This clearly shows that a design of an RLL code which is not a \((d, k)\) code plus precoder can be quite advantageous.

IV. CONCLUSIONS

We have presented a new rate \(4/6\) \((d = 1, k = 11)\) runlength-limited code. The code is block-decodable, and it is particularly attractive as many commercially available Reed-Solomon codes operate in \(\text{GF}(2^m)\). The encoder can be implemented with a simple 6-bit ROM, and decoding can be accomplished with a logic array.

REFERENCES


Tang and Bahl also showed how \(dk\)-sequences can be cascaded without violating, at the boundaries, the prescribed constraints. A more efficient method for cascading \(dk\)-sequences was described by Beekker and Immink [5] using \(dklr\)-sequences. A \(dklr\)-limited sequence is a \(dk\)-sequence that starts with at most \(t\) "zeros." Also for enumerating \(dklr\)-sequences the prior art requires two sets of coefficients.

We will present a new method for enumerating \(dklr\)-sequences requiring only one set of coefficients. The major part of the electronics that implements the enumeration technique is taken by the storage of the coefficients. As a result, the new method presented will almost halve the electronics needed.

II. CODING AND DECODING USING ENUMERATION

Tang and Bahl [2] developed a general algebraic technique for encoding and decoding \(dk\)-sequences. They defined a 1–1 mapping from the set \(S\) of all \(dk\)-sequences of length \(n\) onto the set of integers \(\{0, 1, \cdots, |S| - 1\}\) and they presented an algorithm for converting \(dk\)-sequences to integers and vice versa. Tang and Bahl’s method is briefly described below. Let \(\{0, 1\}^n\) denote the set of binary sequences of length \(n\) and let \(S\) be any subset of \(\{0, 1\}^n\). The set \(S\) can be ordered lexicographically as follows: if \(x = (x_1, \cdots, x_n) \in S\) and \(y = (y_1, \cdots, y_n) \in S\), then \(y \leq x\), if there exists an \(i, 1 \leq i \leq n\), such that \(y_s < x_s\) and \(x_j = y_j, 1 \leq j < i\). For example, \(001101 < 001011\). The position of \(x\) in the lexicographical ordering of \(S\) is defined to be the rank of \(x\) denoted by \(i_s(x)\), i.e., \(i_s(x)\) is the number of all \(y\) in \(S\) with \(y < x\). Tang and Bahl showed that in the specific case of \(dk\)-sequences the rank can be computed by

\[
i_s(x) = \sum_{j=1}^{n} x_i N(n-j)
\]

where \(N(n)\) is the number of \(dk\)-sequences of length \(n, n > 0\). By definition we have \(N(0) = 1\). The above algorithm is usually called the decoding operation. In an implementation of this algorithm, the weight coefficients \(N(n)\) can be precalculated and stored in memory or they can be calculated “on the fly.” In the more specific case of \(dklr\)-sequences, a similar relationship can be written down. Translation of an integer \(I, 0 \leq I \leq N(n)-1\), into the corresponding \(x\), the encoding operation, is done by the following algorithm:

\[
\hat{I} := a + I_j \\
\text{for } j = 1 \text{ to } n \text{ do} \\
\text{if } \hat{I} \geq T(n-j) \text{ then} \\
x_j := 1, \hat{I} := \hat{I} - N(n-j) \text{ else } x_j := 0
\]
integer coefficients \( a \) and \( T(j) \), \( 1 \leq j \leq n \), are a function of the runlength constraints in force and are not relevant in this context. The interested reader is referred to the literature [3]. In the general \( dk \)-limited case, two sets of coefficients are needed for accomplishing the encoding operation. For \( d \)-limited sequences, we have \( a = 0 \) and \( T(n) = N(n) \), and as a result only one set of coefficients is required for encoding. In the \( k \)-limited case we need two coefficient sets. However, as remarked by Tang and Bahl, if the "complement" form of \( x \) is dealt with [1], only one set is required. This motivated us to attempt finding a method for encoding \( dklr \)-sequences with only one set. The outcome is given below.

III. DESCRIPTION OF THE NEW ENUMERATION SCHEME

Another scheme to enumerate vectors of the set \( S \in \{0,1\}^n \) was given by Cover [4]. Let \( n_s(x_1, x_2, \ldots, x_n) \) be the number of elements in \( S \) for which the first \( k \) coordinates are \( (x_1, x_2, \ldots, x_n) \). The rank of \( x \in S \) can be obtained by

\[
i'_S(x) = \sum_{j=1}^{n} x_j n_s(x_1, x_2, \ldots, x_j-1, 0).
\]

Although the above sum looks similar to (1), applying Cover's method for enumerating \( dk \)-sequences yields different weights that are dependent of the sequence coded so far. It can be shown that, as opposed to Tang and Bahl's algorithm, Cover's enumeration needs approximately \( kn \) weights for encoding and decoding \( dklr \)-sequences.

An alternative of Cover's enumeration scheme can be given by counting the number of elements of \( S \) that have a higher lexicographic index than \( x \), the inverse rank of \( x \).

Proposition 1: The inverse rank of \( x \in S \) is given by

\[
i'_S(x) = \sum_{j=1}^{n} x_j n_s(x_1, x_2, \ldots, x_j-1, 0)
\]

where \( \bar{x}_j = 1-x_j \), the complement of \( x_j \).

Proof: Words with prefix \((x_1, x_2, \ldots, x_j-1, 0)\) lexicographically precede words with prefix \((x_1, x_2, \ldots, x_j-1, 1)\). For each \( j \) such that \( x_j = 0 \), \( n_s(x_1, x_2, \ldots, x_j-1, 1) \) gives the number of elements of \( S \) that first differ from \( x \) in the \( j \)th term and therefore have a higher lexicographic index. By adding these numbers for \( j = 1, 2, \ldots, n \), we eventually count all the elements in \( S \) of higher index than \( x \).

Given \( S \) and the lexicographic index \( I \), the decoding algorithm of Proposition 1 runs as follows:

1) If \( I \geq n_s(1) \) set \( x_1 = 0 \) and set \( I = I - n_s(1) \); otherwise set \( x_1 = 1 \).

2) For \( j = 2, \ldots, n \), if \( I \geq n_s(x_1, x_2, \ldots, x_j-1, 1) \) set \( x_j = 0 \) and set \( I = I - n_s(x_1, x_2, \ldots, x_j-1, 1) \); otherwise set \( x_j = 1 \).

IV. ENCODING AND DECODING OF \( dklr \)-SEQUENCES

In the following we first give an algorithm to enumerate \( dklr \)-constrained sequences, for which the length of the leading zero-run is not constrained. From the results obtained, enumeration of \( dklr \)-sequences follows easily.

To use Proposition 1, we introduce some notations. Given a \( dklr \)-codeword \( x \), let \( x_j^* = (x_1, x_2, \ldots, x_j-1, 0) \). Denote by \( N^0(\cdot) \) the number of \( dklr \)-constrained sequences of length \( i \) whose first element equals \( 1 \). We define the quantity \( a_j(x) \) as the length of the trailing zero-run of the subvector \((x_1, \ldots, x_j)\) if it is not the all-zero sequence. Hence

\[
a_j(x) = \begin{cases} 
\min \{ (j-i-1) : 1 \leq i < j, x_i = 1 \}, & \text{if } j > 1 \\
1 & \text{if } j = 1 \\
d, & \text{otherwise.}
\end{cases}
\]

Applying Proposition 1 for \( dklr \)-sequences, we can observe that for a given \( j \), \( 1 \leq j \leq n \), \( n_s(x_j^*) \) can take one of two different values, depending on \( x_j \). If \( a_j(x) < d \) then \( n_s(x_j^*) = 0 \), because in this case \( x_j^* \) contains at least one "one" in the first \((j-1)\) positions, and the length of the zero-run between the last two 1's of \( x_j^* \) is less than \( d \), violating the \( d \)-constraint. Therefore, no \( dklr \)-codeword can begin with \( x_j^* \), so \( n_s(x_j^*) = 0 \). If \( a_j(x) \geq d \), then \( x_j^* \) does not violate the \( d \)-constraint, so that \( n_s(x_j^*) \) equals the number of \( dklr \)-sequences beginning with \( x_j^* \). Considering that \( x_j^* \) ends with a 1, the value of \( n_s(x_j^*) \) is independent of \( x_j \) and is given by \( N^0(n-j+1) \). By noting the above we get the next proposition for the enumeration of \( dklr \)-sequences.

Proposition 2: The inverse rank of a \( dklr \)-sequence \( x \) of length \( n \) is

\[
i'_S(x) = \sum_{j=1}^{n} n_s(x_1, x_2, \ldots, x_j, 0)\delta_j(x)N^0(n-j+1)
\]

where

\[
\delta_j(x) = \begin{cases} 
1, & \text{if } x_j = 0 \text{ and } a_j(x) \geq d \\
0, & \text{otherwise.}
\end{cases}
\]

To put in a simple way, the inverse rank of \( x \) can be obtained by summing the weight of each "zero" of \( x \) that is either contained in the leading zero-run or is preceded by at least \( d \) zeroes.

We can observe that \( dklr \)-sequences violating the \( l \)-constraint have lower lexicographic indexes than \( dklr \)-sequences, therefore their inverse rank is higher. If we denote by \( x \) the \( dklr \) codeword having the highest inverse rank in the set of \( dklr \)-sequences \( S \), then the set \( \{0, \ldots, i'_S(x)\} \) of inverse ranks corresponds to the set of all \( dklr \)-codewords of length \( n \). This property is very favorable, because using the new algorithm no subtraction of a constant is required during encoding.

Proposition 1 provides the algorithm for decoding \( dklr \)-sequences. The encoding operation, i.e., given the inverse lexicographic index \( I \) find the corresponding \( x \), is described by the following program:

\[
i := I, \quad a := d;
\]

for \( j = 1 \) to \( n \) do

if \( I \geq N^0(n-j+1) \) and \( a \geq d \)

then \( x_j := 0, \quad I := I - N^0(n-j+1) \)

else if \( a < d \)

then \( x_j := 0 \)

else \( x_j := 1, \quad a := a + 1 \);

end for

V. CONCLUSIONS

We have presented a new method of enumerating runlength-limited sequences that requires only one set of weight coefficients.

REFERENCES