A. M. Patel
S. J. Hong

Optimal Rectangular Code for High Density Magnetic Tapes

Abstract: IBM's 6250 bpi 3420 series tape units require a powerful error-correcting code for the standard 9-track format. The optimal rectangular code (ORC), presented here, is designed to correct any single-track error or, given erasure pointers, any double-track error in the tape. The code achieves this by conforming to a rectangular codeword of which two orthogonal sides are check bits. The code is specially tailored from a general class of b-adjacent codes. The ORC can be implemented without a buffer for encoding and offers a simple error-correction mechanism. The code can be generalized to multiple-channel applications.

Introduction

Models 4, 6 and 8 of the IBM 3420 series tape units record 6250 bits per inch (bpi), one of the highest densities commercially available on standard ½-inch 9 track tapes. To achieve this density, a number of stringent engineering requirements had to be met. Included was a fast and powerful error-correcting scheme.

The standard ½-inch 9-track tape system evolved through the extensive use of tapes over many years. The most frequently used 8-bit byte of information and a parity check bit are accommodated vertically in these 9 tracks. The ninth parity track is usually called the VRC (vertical redundancy check) track. In low density recording such as IBM 729 series tapes, this VRC and a longitudinal parity byte (LRC) at the end of a record were sufficient. As the bit density increased, another check byte called CRC (cyclic redundancy check) was added at the end of the record to provide track error correction such as in IBM's 800 bpi tape units [1].

In designing new tape products, compatibility with the existing 9-track standard data format is one of the prime considerations if tapes are to be recorded in different machines and interchanged freely. The new products, moreover, require a superior error-correction code because of increased bit densities and/or tape speeds and other similar reasons. The new coding scheme presented in this paper is designed to satisfy these requirements.

Bit density along the direction of tape motion is conventionally much higher than that across the tape. The ratio has steadily increased from about 40 in 800 bpi tapes to two to three orders of magnitude in the current high density tape machines. As a result, most common errors are track erasures. The erroneous tracks are often identified by the loss of signal in the read amplifiers, and/or excessive phase shift in clock and detection circuits or other similar electronic indicators. The coding scheme of this paper is designed to correct this type of track error or erasure. The same scheme, moreover, corrects many random errors which may be spread over many tracks in a record. The codewords are in the form of a rectangle with two orthogonal sides as check bits, and they resemble the diagonal check scheme [2]. The encoding and decoding, however, are carried out in an algebraic manner and the redundancy is minimal. The erroneous bits along a track form a cluster-error that can be corrected as a unit in each rectangular codeword. Each rectangular codeword is processed on the fly, independent of whether cluster errors are from the same track or different tracks.

It is well known that the error-correcting codes for symbols from $GF(2^b)$, the galois field of 2^b elements, can be used for correction of clusters of b-adjacent binary symbols. The generalized Hamming codes [3-5], Reed-Solomon codes [6], and BCH codes [4] with symbols from $GF(2^b)$ are some of the examples. These codes can be described in a binary check matrix as proposed by Cocke [7]. Bossen [3] shows a high-speed implementation using the binary matrix description. Other methods of implementing the $GF(2^b)$ codes for

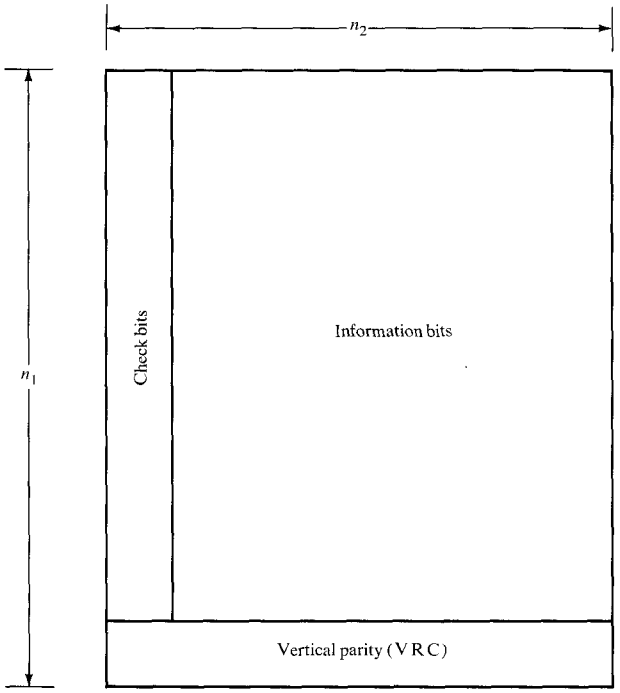

Figure 1 The rectangular format of check bits and information bits on the tape.

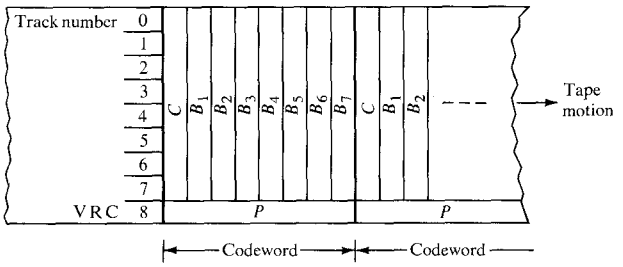

Figure 2 The data format for ORC showing the information bytes as vertical columns.

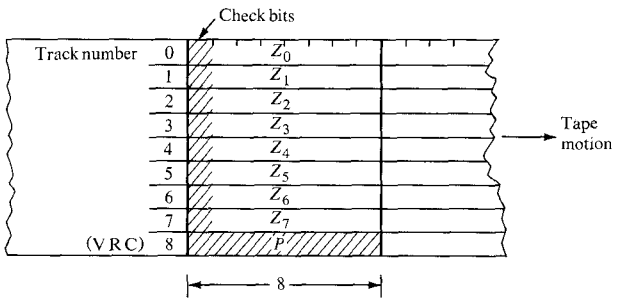

Figure 3 The horizontal track-bytes in ORC—these bytes are subjected to errors and corrections.

b-adjacent errors are proposed by Patel [8, 9] In all these b-adjacent codes, each check symbol and each information symbol in $GF(2^b)$ is replaced by b-binary information digits. All encoding and decoding operations are performed on these clusters of b-binary digits, thus obtaining b-adjacent correction corresponding to the correction of a symbol in $GF(2^b)$.

Hong and Patel [10] obtained a general class of maximal binary codes for correcting a single cluster of b-adjacent errors (or double erasure groups). These codes are constructed directly for binary symbols and, hence, the codewords are not described in terms of the symbols of $GF(2^b)$. One advantage of avoiding symbols from $GF(2^b)$ is that now the binary check bits are no longer required to be clustered for representation of the check symbols in $GF(2^b)$. Instead, each binary parity check acts independently.

Given a parity check matrix of r check bits and the cluster size b, one may choose any linearly independent set of r columns to be the check-bit positions, without affecting the b-adjacent error-correcting capability of the code. Use is made of this property here to mix the binary check digits and the information digits in forming the correctable clusters. This was designed to fit the format of the standard tape recording application. It will be shown that the resultant code, called the optimal rectangular code (ORC), not only meets all the constraints of the tape application but is also easier to implement. It is also optimally efficient in this data format in providing two-track correction capability.

In practice ORC resembles the CRC single-track correcting scheme of Brown and Sellers [1] except that the ORC is also capable of correcting two-track erasures. Moreover, ORC corrects many combinations of cluster or random errors that may be spread over more than two tracks in a record.

Code format and parity check matrix

Codewords of the ORC have a rectangular format of dimensions n_1 and n_2 ($n_1 > n_2$, Fig. 1). Check bits are located on two orthogonal sides of the rectangle. The check bits along the shorter dimension are the overall vertical parity bits, known as VRC in tape applications. Remaining check bits along the vertical column are parity bits over selected positions of information bits. Redundancy is minimum when n_2 is the largest for given n_1, i.e., $n_2 = n_1 - 1$. The special case of $n_1 = 9$ will be described for the standard 9-track $\frac{1}{2}$-inch tape. The code for any other value of n_1 can be constructed in a similar manner.

The data format [11] for the ORC of 9-track tapes is illustrated in Fig. 2. B_1 through B_7 denote the seven bytes of information in standard 8-bit bytes. C denotes the check byte computed from the information bytes. This format shows how the information bytes are written as B_i. The code corrects track errors as errors in

clusters of b-bits along tracks. For a natural description of the code, the track vectors of the codeword will be used as track bytes, denoted by Z_i's in Fig. 3. First, the error-correcting capability of the code is established in this Z_i-notation of Fig. 3. Later, a novel conversion of these coding rules is given in terms of the information bytes in B_i-notation of Fig. 2.

Some well-known but pertinent mathematical background is given here. Let $g(x)$ denote an irreducible polynominal [12] of degree 8 with binary coefficients g_i, i.e., $g(x) = \mathbb{X}_{i=0}^{8} x^i$, where $\mathbb{X}$ denotes the summation modulo 2. The companion matrix T of the polynomial $g(x)$ is defined as the following nonsingular binary matrix:

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & g_1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & g_2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & g_3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & g_4 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & g_5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & g_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & g_7 \end{bmatrix} \tag{1}$$

Let $\underline{\alpha}$ be the element of GF(2^8) representing the residue class x modulo $g(x)$. The sum and multiplication of the elements in GF(2^8) is defined by the polynomial sum and multiplication of the corresponding residue classes modulo $g(x)$. Thus, an element $\underline{\alpha}^i$ for any i, represents the residue class x^i modulo $g(x)$ which can be expressed as an 8-digit column vector α^i of the binary coefficients of the polynomial x^i modulo $g(x)$. For example, $\underline{\alpha}^3$ is represented by the 8-digit column vector

$$\alpha^3 = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]^\tau$$

where superscript τ signifies transposition of a matrix.

Similarly, any 8-digit binary column vector β,

$$\beta = [\beta(0), \beta(1), \beta(2), \beta(3), \beta(4), \beta(5), \beta(6), \beta(7)]^\tau, \tag{2}$$

represents the residue class $\{\mathbb{X}_{i=0}^{7} \beta(i)\ x^i\}$ modulo $g(x)$ which is an element $\underline{\beta}$ of GF(2^8). This 8-digit vector representation will be used for computations involving the elements of GF(2^8). The sum of elements of GF(2^8) is, then, the modulo 2 matrix sum of the column vectors. The products of the elements of GF(2^8) can be computed [4] as a matrix product with help of the companion matrix T of Eq. (1). In particular, multiplication of $\underline{\alpha}$ with any element $\underline{\xi}$ in GF(2^8) can be computed by the matrix multiplication $T\underline{\xi}$. Since β of Eq. (2) can be considered as the matrix sum,

$$\beta = \sum_{i=0}^{7} \beta(i)\ \alpha^i, \tag{3}$$

the product of any two elements $\underline{\beta}$ and $\underline{\xi}$ in GF(2^8) can be computed as the sum of matrix products $\mathbb{X}\ \beta(i)\ T^i\xi$, which is equal to $[\mathbb{X}\beta(i)T^i]\xi$. The matrix M_β given by

$$M_\beta = \left[\sum \beta(i)\ T^i\right], \tag{4}$$

corresponding to each element β in GF(2^8) is, then, the matrix operator for computing the product of β with any element of GF(2^8). It can be shown that the matrix operator for the element β^{-1}, the inverse of β, is the matrix M_β^{-1}, the inverse of matrix M_β. In particular, $M_{\alpha^i} = T^i$ and $M_{\alpha^{-1}} = T^{-1}$.

The companion matrix T of Eq. (1) can be written using the notation of field elements as

$$T = [\alpha\ \alpha^2\ \alpha^3\ \alpha^4\ \alpha^5\ \alpha^6\ \alpha^7\ \alpha^8]. \tag{5}$$

Since $T\alpha = \alpha^2$, by interative computation it follows that

$$T^i = [\alpha^i, \alpha^{i+1}, \alpha^{i+2}, \alpha^{i+3}, \alpha^{i+4}, \alpha^{i+5}, \alpha^{i+6}, \alpha^{i+7}]. \tag{6}$$

The elements α^i form a cyclic subgroup contained in the group of all elements of GF(2^8) under multiplication. If n ($8 \leq n < 2^8$) is the exponent of the polynomial $g(x)$, then $\underline{\alpha}^n = \underline{\alpha}^0$ and

$$T^n = T^0 = I, \tag{7}$$

where I is the identity matrix.

At this point, the mathematical description of the code can be given. Using the matrices of Eq. (6), the parity checking rules of the ORC can be written as the following matrix equations:

$$\left(\sum_{i=0}^{7} Z_i\right) = P, \tag{8}$$

and

$$\left(\sum_{i=0}^{7} T^i Z_i\right) = \varnothing \tag{9}$$

where vector P represents the conventional VRC and $\varnothing$ is the null vector with all zero elements.

Alternatively, the parity checking rules can be written in a concise form $HW^\tau = \varnothing$, where H is the parity check matrix and W is a code word. This parity checking matrix H is, then,

$$H = \begin{bmatrix} I & I & I & I & I & I & I & I & I \\ T^0 & T^1 & T^2 & T^3 & T^4 & T^5 & T^6 & T^7 & 0 \end{bmatrix} \tag{10}$$

for the codeword $W = [Z_0, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, P]$, where I is an identity matrix and 0 is a matrix with all zero elements.

Recall that the first bit of every track-byte Z_i is the i^{th} bit of the vertical check byte $C(i)$. Notice also that the above parity check matrix is a shortened version of the maximal code, $H_{2b,b}$ where $b = 8$, given by Hong and Patel. [10]. $H_{2b,b}$ was given in terms of the companion

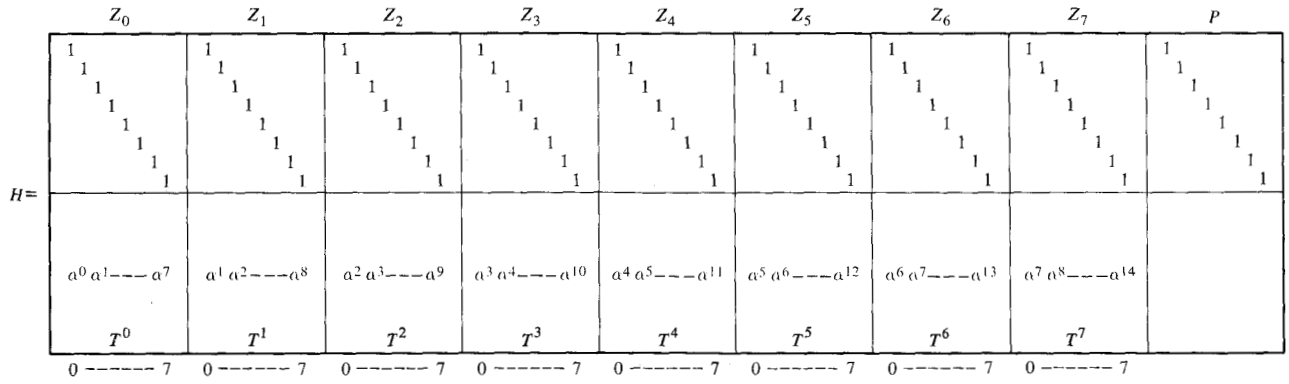

Figure 4 The parity check matrix H for syndrome decoding. Z and P are track-bytes.

matrix of a primitive polynomial to be of maximal length. In ORC a companion matrix of any degree-8 irreducible polynomial is used. A column-by-column representation of the above parity check matrix appears in Fig. 4. The parity check matrix in this format does not suggest an immediate and simple method for computing the check byte from the information bytes. Later an alternate description will be given for the parity check matrix H to facilitate this computation of the check byte. Figure 5 represents the parity check matrix in this alternate format.

Code capability

The ORC given by the parity check matrix of Eq. (10) is a shortened form of a maximal code reported previously, [10] with the same error-correction capaibility, namely, correction of one track-byte error or two track-byte erasure errors in a rectangular codeblock. In the parity check matrix for the ORC, the companion matrix of any irreducible polynomial will be used instead of a primitive polynomial as in Ref. 10. Furthermore, the track bytes Z_i in ORC consists of a mixture of information bits and check bits unlike those in [10]. This, however, does not affect the proof of the theorems on code capability. Here, an independent proof of the ORC capability is provided.

Any correct codeword $W = [Z_0 Z_1 Z_2 \cdots Z_7 P]$ should satisfy the parity check equations given by

$$\left(\sum_{i=0}^{7} Z_i\right) \oplus P = \varnothing, \text{ and} \tag{11}$$

$$\left(\sum_{i=0}^{7} T^i Z_i\right) = \varnothing. \tag{12}$$

When the codeword is corrupted by either single or double track error, the corrupted word is denoted by $\hat{W} = [\hat{Z}_0 \hat{Z}_1 \text{ --- } \hat{Z}_i \text{ --- } \hat{Z}_7 \hat{P}]$. From the corrupted received codeword W, the syndromes S_1 and S_2 of errors are computed as

$$S_1 = \left(\sum_{i=0}^{7} \hat{Z}_i\right) \oplus \hat{P}, \tag{13}$$

and

$$S_2 = \sum_{i=0}^{7} (T^i \hat{Z}_i). \tag{14}$$

Obviously, if there is no error, $S_1 = S_2 = \varnothing$. However, if S_1 or S_2 is not found equal to $\varnothing$, this fact is an indication of errors in the received codeword.

Theorem 1. Any error pattern in any single track byte, either on one of the Z_i or in P, in an ORC codeblock is detectable and correctable.

Proof: Suppose that only the ith track $(0 \leq i \leq 8)$ has erroneous track bytes and the corrupting error pattern is denoted by an 8-digit vector e. That is, the received bytes are error free except in the ith track where

$$\hat{Z}_i = Z_i \oplus e \quad \text{if} \quad 0 \leq i \leq 7,$$

or (15)

$$\hat{P} = P \oplus e \quad \text{if} \quad i = 8.$$

In view of Eqs. (11), (12) and (15), the computed syndromes S_1 and S_2 of Eqs. (13) and (14) represent

$$S_1 = e \tag{16}$$

and

$$S_2 = \begin{cases} T^i e & \text{if} \quad 0 \leq i \leq 7, \\ \varnothing & \text{if} \quad i = 8. \end{cases} \tag{17}$$

Thus, S_1 directly provides the error-pattern vector e. Now, if $S_2 = \varnothing$, the error is in the VRC byte P. If S_2

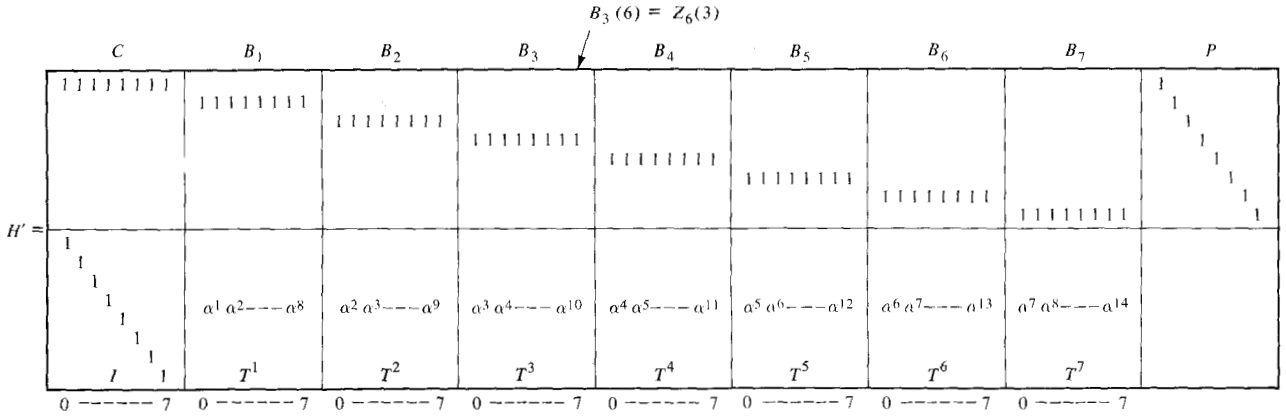

Figure 5 The parity check matrix H' for encoding and syndrome generation. B's are vertical bytes.

$\neq \varnothing$, then the track position i is uniquely determined from the fact that $T^{-i} S_2 = S_1 = e$, where T^{-i} is the operator for the unique element α^{-i} of GF(2^8).

Theorem 2: Any two track bytes in error in an ORC code block are correctable, provided the erroneous tracks i and j are identified by some external pointers [13].

Proof: Let e_1 and e_2 denote the two error-pattern vectors representing errors in the tracks i and j, respectively $(i < j)$. That is, the received bytes are error free except in tracks i and j, where $\hat{Z}_i = Z_i \oplus e_1$, and $\hat{Z}_j = Z_j \oplus e_2$, or $\hat{P} = P \oplus e_2$ if $j = 8$. (The special case of single erasure can be included by allowing $i = j$ in which case e_2 will be assumed to be $\varnothing$.) Then, in view of Eqs. (11) and (12), the computed syndromes S_1 and S_2 of Eqs. (13) and (14) represent

$$S_1 = e_1 \oplus e_2, \tag{18}$$

$$S_2 = \begin{cases} T^i e_1 \oplus T^j e_2 & \text{if} \quad i \neq j \neq 8, \\ T^i e_1 & \text{if} \quad j = 8 \text{ or } j = i. \end{cases} \tag{19}$$

Equations (18) and (19) represent two independent equations involving elements of GF(2^8) in the matrix operator notation. T^i and T^j are matrix operators representing multiplication by the elements α^i and α^j, respectively. These equations uniquely determine the error patterns e_1 and e_2 as

$$e_1 = S_1 \oplus e_2, \text{ and} \tag{20}$$

$$e_2 = \begin{cases} [I \oplus T^{j-i}]^{-1}(S_1 \oplus T^{-i} S_2) & \text{if} \quad i \neq j \neq 8, \\ [S_1 \oplus T^{-i} S_2] & \text{if} \quad j = 8 \text{ or } j = i. \end{cases} \tag{21}$$

where the operators $[I + T^{j-i}]^{-1}$ and T^{-i} represent multiplication by the unique elements $(\alpha^0 \oplus \alpha^{j-i})^{-1}$ and α^{-i}, respectively. This completes the proof.

The above two theorems show the error-correcting capability of ORC. Now, if the code is used purely for error detection, an estimate can be made of how many of the all possible error patterns in the 72-bit code word map to the zero syndrome. Assuming that the inverses of all 2^{16} possible sydromes have roughly the same size domain, i.e., roughly an equal number of 72-bit error patterns map to the same syndrome, $\frac{1}{2}^{16}$ of all possible errors produce zero syndrome. This means 99.998% of all possible error patterns can be detected. In single-track correction mode, exactly 9×2^8 syndromes are used for error correction. The remaining nonzero syndromes indicate an uncorrectable error. Thus, $(1 + 9 \times 2^8)/2^{16}$ of all error patterns in multiple tracks 3.52% gets interpreted as single-track-errors and results in miscorrection. Others, an estimated 96.48% of all multiple-track-errors, will be detected without miscorrection.

Orthogonal symmetry

In the previous sections, the ORC was shown in the track-byte format and its capability was established. Now, an interesting conversion of the parity check matrix into the information-byte format is presented. This alternate form enables direct computation of the check byte and syndromes. In addition, the new format allows a fast implementation of ORC, requiring no buffer for encoding. This is done by observing the orthogonal symmetry of the code with respect to the track bytes and the information bytes, the Z_i's and B_i's.

First, consider the column of the parity check matrix H of Fig. 4 corresponding to the bit $Z_i(j)$ of the track-byte Z_i for all i and j such that $0 \leq i \leq 7$ and $0 \leq j \leq 7$. The lower half of this column is α^k, where $k = i + j$, which is same for the column corresponding to $Z_j(i)$. This property is called the orthogonal symmetry of the code. More graphically, Fig. 6 shows the powers of α that appear in the lower half columns of the H-matrix, corresponding to every bit position in the format.

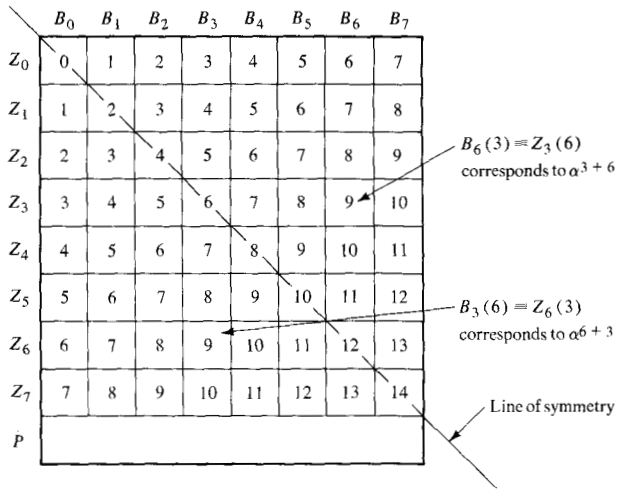

Figure 6 The orthogonal symmetry and powers of α in H-matrix.

To complete the symmetry, let B_0 denote the check byte C. Then $Z_j(i) \equiv B_i(j)$ for all i and j such that $0 \leq i \leq 7$ and $0 \leq j \leq 7$. Now, proceed to rearrange the columns of the H-matrix of Fig. 4 to obtain another parity check matrix H' in Fig. 5, corresponding to a code word W′ in terms of the information bytes written as $W' = [B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7, P]$. Note that this rearrangement does not alter the parity checking rules.

Table 1 shows the lower-half of the matrix H' corresponding to the information byte B_i. Since $B_i(j) \equiv Z_j(i)$ and the lower half of the columns of the H matrix are same for $Z_j(i)$ and $Z_i(j)$, this parity checking multiplier for the information byte B_i and the track byte Z_i remains invariant. Thus, the orthogonal symmetry of the code has produced T^i in the reordered lower half of the matrix H', corresponding to the information byte B_i which is the same as that corresponding to the track byte Z_i! The upper-half of H' is the conventional VRC and can be represented by a matrix G_i, where G_i is an 8×8 all-zero matrix, except the row i which is all ones. Thus, the new parity check matrix H' appears in a compact form as follows:

$$H' = \begin{bmatrix} G_0 & G_1 & G_2 & G_3 & G_4 & G_5 & G_6 & G_7 & I \\ T^0 & T^1 & T^2 & T^3 & T^4 & T^5 & T^6 & T^7 & 0 \end{bmatrix} \quad (22)$$

for the codeword $W' = [B_0\, B_1\, B_2\, B_3\, B_4\, B_5\, B_6\, B_7\, P]$. A bit-by-bit version of H' appears in Fig. 5. The parity checking equation is

$$H'W'^{\tau} = \varnothing.$$

Alternatively, the parity check can be computed from the information bytes as

$$C \equiv B_0 = \sum_{i=1}^{7} T^i B_i, \quad (23)$$

and

$$P(i) = \sum_{j=0}^{7} B_i(j). \quad (24)$$

In polynomial notation, Eq. (23) can be written as

$$C(x) = \sum_{i=1}^{7} x^i B_i(x) \text{ Modulo } g(x), \quad (25)$$

where $C(x)$ and $B_i(x)$ are degree 8 polynomials with binary coefficients given by C and B_i, respectively. These equations form the basis for a fast and simple implementation of ORC. Equation (24) represents the conventional parity computation. Equation (23) or (25) can be implemented by means of a linear feedback shift register connected for Modulo $g(x)$ operation. The implementation is presented in the next section.

Implementation of ORC

The code can be generated using any irreducible polynominal $g(x)$. Table 2 lists all irreducible polynomials of degree 8 with their exponents. This table is taken from Peterson [4]. Choice of $g(x)$ from this set could be arbitrary; however, there are some specific advantages in choosing (i) low exponent polynomial [8] and (ii) self reciprocal polynomial for the read backward facility [1] The polynomials, 8 and 16, in the table are self reciprocal (i.e., $g(x) = x^8 g(1/x)$) and have the lowest value of exponent. Polynomial no. 8 from Table 2 is used for computation of CRC [1] which is retained in the newer tape units for error detection purpose. Therefore, the polynomial no. 16 $g(x) = 1 + x^3 + x^4 + x^5 + x^8$ is chosen for ORC application. The corresponding companion matrix T is, then,

Table 1 Lower-half of H' corresponding to B_i

Bits of Info-byte B_i :	$B_i(0)$,	$B_i(1)$,	$B_i(2)$,	$B_i(3)$,	$B_i(4)$,	$B_i(5)$,	$B_i(6)$,	$B_i(7)$
Corresponding Track-byte bits:	$Z_0(i)$,	$Z_1(i)$,	$Z_2(i)$,	$Z_3(i)$,	$Z_4(i)$,	$Z_5(i)$,	$Z_6(i)$,	$Z_7(i)$
Lower-half column in H′ :	α^i	α^{i+1}	α^{i+2}	α^{i+3}	α^{i+4}	α^{i+5}	α^{i+6}	α^{i+7}

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Encoding. The check byte C is computed from the information bytes according to Eq. (23). This is accomplished by means of a feedback shift register (SR1) as given in Fig. 7. The feedback connections are determined by the binary coefficients g_i's of $g(x)$. $g_i = 1$ implies connection and $g_i = 0$ means no connection at the ith cell. Each stage of the shift register corresponds to the digit position of the check byte C as marked. The shifting operation corresponds to multiplying the content polynomial by x modulo $g(x)$ which is equivalent to multiplying the content vector by the companion matrix T. Input connections are such that the entering information bytes are premultiplied by T. Initially, SR1 contains all zeros. The information bytes B_7, B_6, ---, B_2, B_1 are successively shifted in parallel into SR1 in that order. Thus, at the end of seven shifts, SR1 contains the vector

$$TB_1 \oplus T^2B_2 \oplus T^3B_3 \oplus \text{----} \oplus T^7B_7$$

which is the check byte C gated out onto the tape. The byte parity of B_7, B_6, ---, B_2, B_1 and C is computed by a usual 8-way EXCLUSIVE OR network embodying Eq. (24).

Syndrome Generation. Let $\hat{Z}_i$'s and $\hat{P}$ stand for the received vectors; likewise, $\hat{B}_i$ and $\hat{C}$ now denote the received vectors which may be corrupted by error. Based on Eqs. (23) and (24), the syndrome equations can be written in terms of the information bytes and P, rather than in terms of the track bytes as in Eqs. (10) and (11).

$$\left.\begin{aligned} S_1(0) &= \hat{P}(0) \oplus \bigoplus_{i=0}^{7} \hat{C}(i); \\ S_1(i) &= \hat{P}(i) \oplus \bigoplus_{j=0}^{7} \hat{B}_i(j) \text{ for all } i \neq 0 \end{aligned}\right\} \tag{26}$$

$$S_2 = \hat{C} \oplus \bigoplus_{i=1}^{7} T^i \hat{B}_i \tag{27}$$

The computation of $S_1(i)$ is done by a 9-way EXCLUSIVE OR gate iteratively as each byte is received. S_2 can be generated using a forward shifting register similar to the one used in encoding, but without premultiplication by T and with an added cycle of shift to accommodate the C byte. However, it will be seen later that multiplication by T^{-i} is required in the decoding process and hence, a backward shifting register saves time. Also a backward shifting register is convenient in the read backward mode

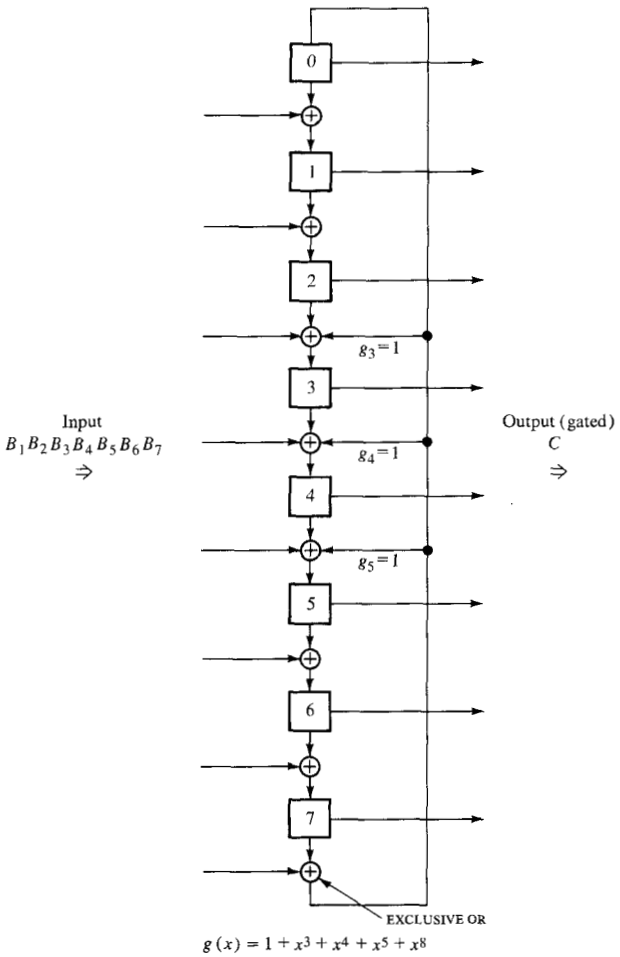

Figure 7 Feedback shift register SR1 for encoding. Information bytes arrive from B_7 to B_1 in parallel into the shift register.

Table 2 Irreducible polynomials of degree 8

No.	*Coefficients:* g_0g_1----g_8									*Exponent: n*
1	1	0	0	0	1	1	1	0	1	255
2	1	0	1	1	1	0	1	1	1	85
3	1	1	1	1	1	0	0	1	1	51
4	1	0	1	1	0	1	0	0	1	255
5	1	1	0	1	1	1	1	0	1	85
6	1	1	1	1	0	0	1	1	1	255
7	1	0	0	1	0	1	0	1	1	255
8	1	1	1	0	1	0	1	1	1	17
9	1	0	1	1	0	0	1	0	1	255
10	1	1	0	0	0	1	0	1	1	85
11	1	0	1	1	0	0	0	1	1	255
12	1	0	0	0	1	1	0	1	1	51
13	1	0	0	1	1	1	1	1	1	85
14	1	0	1	0	1	1	1	1	1	255
15	1	1	1	0	0	0	0	1	1	255
16	1	0	0	1	1	1	0	0	1	17 chosen

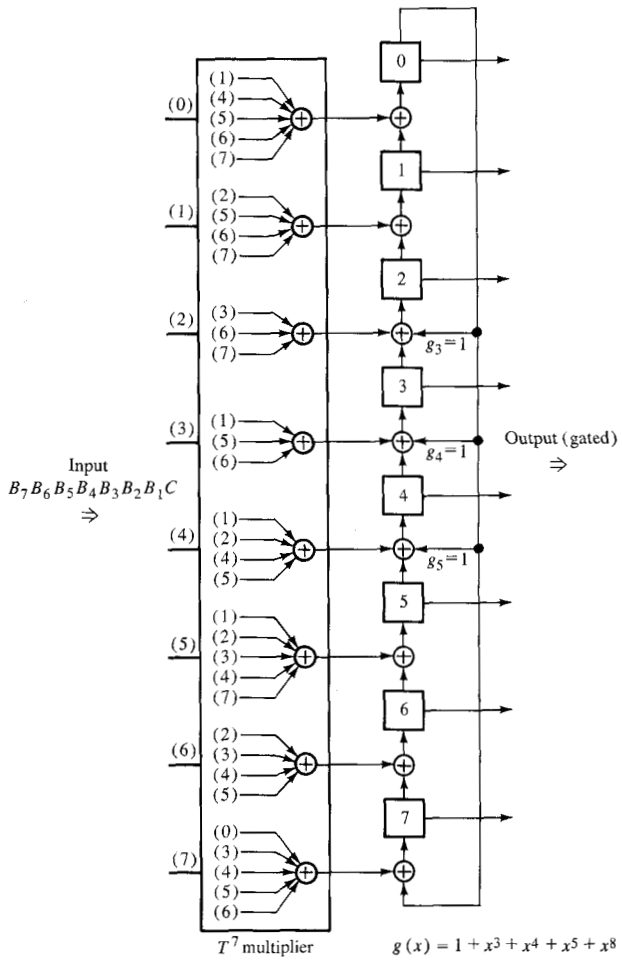

Figure 8 Backward shifting register SR2 with T^7 premultiplier.

and a forward shifting register is convenient in the read forward mode of the tape operations.

Here, the generation of S_2 is presented using the backward shifting register SR2 shown in Fig. 8. For this purpose the syndrome Eq. (27) can be rewritten as

$$S_2 = T^{-7}(T^7C) \oplus \sum_{i=1}^{7} T^{i-7}(T^7B_i) \tag{28}$$

The feedback connections are according to the coefficients of $g(x)$; however, the shifting operation is backwards and corresponds to multiplying the content vector by T^{-1}, the inverse of matrix T. Entering bytes are premultiplied by the matrix T^7, using a network of EXCLUSIVE OR gates. The matrix T^7 appears in Table 3. The received bytes C, B_1, B_2, ---, B_6, B_7 are successively shifted into SR2 in that order. At the end of 8 shifts, SR2 contains the syndrome S_2.

Correction of Single-Track Error. When S_1 and S_2 thus computed are all $\varnothing$, the received word is assumed error free. In single-track correction mode, then, assume that S_1 or $S_2 \neq \varnothing$ indicates a single track error and proceed to correct the error pattern $e = S_1$. S_2 is the content of SR2. If $S_2 = \varnothing$, the erroneous track is the P-track. If $S_2 \neq \varnothing$, then $S_2 = T^i e$ is assumed according to Eq. (17). Shifting SR2 multiples S_2 by T^{-1} each time. Hence, if the error occurred at the i^{th} track Z_i, the contents of SR2 after i shifts should match $S_1 = e$. Thus, when a match occurs, the number of shifts determines the track position. If $S_2 \neq \varnothing$ and the contents of SR2 do not match S_1 after a maximum of 7 shifts, there are two or more tracks in error. This is the additional detection power of ORC.

Alternately, a forward shifting register can be used, such as SR1 for determination of error track position i. Because $T^{-i} = T^{n-j}$, this requires a maximum of n shifts to determine the index i. The polynomial $g(x)$ with the lowest exponent n, in this case, saves correction time.

Table 3 Erasure decoding matrices M_1 through M_7.

$$M_1 = \begin{bmatrix} 01111111 \\ 00111111 \\ 00011111 \\ 11110000 \\ 00000111 \\ 11111100 \\ 11111110 \\ 11111111 \end{bmatrix} \quad M_4 = \begin{bmatrix} 01011101 \\ 10101110 \\ 01010111 \\ 01110110 \\ 01100110 \\ 11101110 \\ 01110111 \\ 10111011 \end{bmatrix} \quad M_7 = \begin{bmatrix} 10011100 \\ 01001110 \\ 10100111 \\ 11001111 \\ 11111011 \\ 11100001 \\ 01110000 \\ 00111000 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 00101010 \\ 00010101 \\ 00001010 \\ 10101111 \\ 11111101 \\ 01010100 \\ 10101010 \\ 01010101 \end{bmatrix} \quad M_5 = \begin{bmatrix} 10001000 \\ 11000100 \\ 01100010 \\ 00111001 \\ 10010100 \\ 01000010 \\ 00100001 \\ 00010000 \end{bmatrix} \quad T = \begin{bmatrix} 00000001 \\ 10000000 \\ 01000000 \\ 00100001 \\ 00010001 \\ 00001001 \\ 00000100 \\ 00000010 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 11001001 \\ 01100100 \\ 10110010 \\ 10010000 \\ 00000001 \\ 01001001 \\ 00100100 \\ 10010010 \end{bmatrix} \quad M_6 = \begin{bmatrix} 00111110 \\ 10011111 \\ 11001111 \\ 01011001 \\ 10010010 \\ 11110111 \\ 11111011 \\ 01111101 \end{bmatrix} \quad T^7 = \begin{bmatrix} 01001111 \\ 00100111 \\ 00010011 \\ 01000110 \\ 01101100 \\ 01111001 \\ 00111100 \\ 10011110 \end{bmatrix}$$

Correction of Double Erasure Tracks. Given the erasure-track pointers as indices i and j, the code can determine the error patterns e_1 and e_2 ($e_2 = \varnothing$ if $i = j$, which is actually a single erasure error as a special case of double erasure error). With no loss of generality, assume $0 \leq i \leq j \leq 8$. The track number 8 denotes the VRC track with P. Rewriting Eqs. (20) and (21),

$$e_1 = S_1 \oplus e_2, \text{ and} \tag{29}$$

$$e_2 = \begin{cases} [I \oplus T^{j-i}]^{-1}(S_1 \oplus T^{-i}S_2) & \text{if } i \neq j \neq 8, \\ S_1 \oplus T^{-i}S_2 & \text{if } j = 8 \text{ or } i. \end{cases} \tag{30}$$

In case of $i = j$, the single erasure error, the error pattern e_1 is given by S_1; however, the computation for e_2 should return $e_2 = \varnothing$. Otherwise, either the erasure pointer is in error or some uncorrectable error is detected. Equation (30) for e_2 can be rewritten as

$$e_2 = M_{j-i}[S_1 \oplus T^{-i}S_2], \tag{31}$$

where

$$M_{j-i} = \begin{cases} [I \oplus T^{j-i}]^{-1}, & \text{if } i \neq j \neq 8, \\ I & \text{if } j = i \text{ or } 8. \end{cases} \tag{32}$$

The matrix M_{j-i} can be computed from T^{j-i} for $j - i = 1, 2, 3, ---, 6, 7$. Table 3 gives these M matrices for all $j - i$ values using the specific companion matrix T of the ORC tape application.

Equation (30) can be realized in the following manner: SR2 with S_2 shifted i times yields $T^{-i}S_2$. Figure 9 gives the block diagram of the circuit which computes e_1 and e_2 from inputs S_1, $T^{-i}S_2$ and the number $j - i$. The blocks M_{j-i} in Fig. 9 for $j - i = 1, 2, \cdots, 6, 7$ are exclusive-or networks realizing multiplication by the respective matrix M_{j-i} of Eq. (32) as given in Table 3. Each of the eight outputs of an M_{j-i} block is the modulo-2 sum of selected inputs indicated by 1's positions in the corresponding row in matrix M_{j-i}.

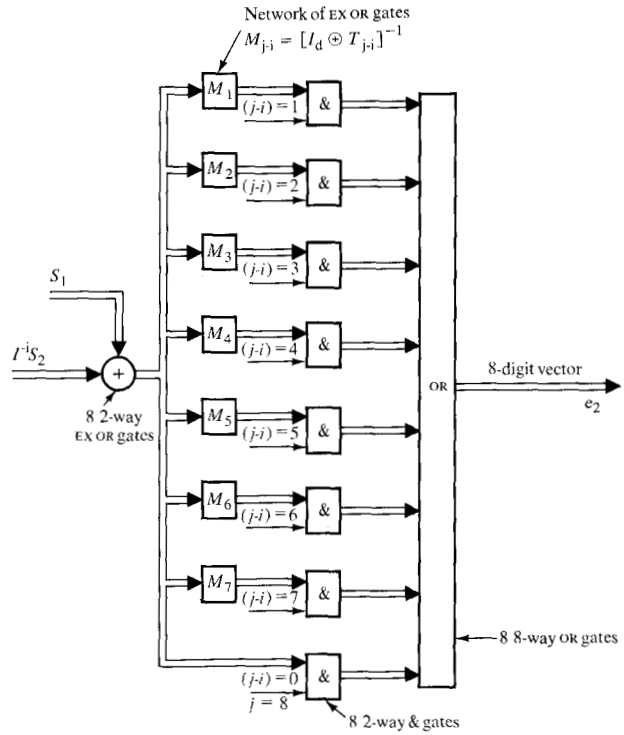

Figure 9 Double erasure decoding circuit.

Summary

The optimal rectangular code satisfied the constraints of the standard 9-track format of $\frac{1}{2}$-inch magnetic tapes. It also meets the error-correction requirements for the 6250-bpi density tape application with high-speed implementation. Data is recorded in the conventional manner with one VRC track. A check byte is introduced for every code word. The orthogonal symmetry in the parity check matrix allows the check byte be generated as the information bytes and their parities (VRC) are being recorded. Hence, the encoding process does not require a buffer.

Error correction is performed on each code word while the next code word is being received. The code corrects any error pattern in any single track. It also corrects any error patterns confined in two known erroneous tracks indicated by the erasure pointers. In addition, the code detects an estimated 96.5% of all possible errors while providing single-track error correction; corrects wrong erasure pointer in case of a single erasure error; and can detect 99.998% of all errors if operated for detection only. The amount of encoding and decoding hardware needed is modest.

The general results of this paper are applicable to any set of parallel channels. For n_1 parallel channels, the code requires one check character for n_2 ($n_2 < n_1$) information characters. One of the n_1 channels is used for transmitting the VRC. The check character is computed using a shift register connected according to an irreducible polynomial of degree $n_1 - 1$ in the same manner as it is illustrated here for the case $n_1 = 9$. The proofs of theorems 1 and 2 apply in the general case using the operations involving elements of $GF(2^{n_1-1})$.

Acknowledgment

The authors express their gratitude to Dr. M. Y. Hsiao for his encouragement and support during the development of this work.

References and notes

1. D. T. Brown and F. F. Sellers, Jr., "Error Correction for IBM 800-bit-per-inch Magnetic Tape," *IBM J. Res. Develop.* **14,** 384 (1970).
2. W. H. Kautz, "A Class of Multiple Error-correcting Codes for Data Transmission and Recording," *Stanford Research Institute Report 5*, (SRI Project 2124), Palo Alto, CA, 1959.
3. D. C. Bossen, "b-Adjacent Error Correction," *IBM J. Res. Develop.* **14,** 402 (1970).
4. W. W. Peterson, *Error Correction Codes*, Massachusetts Institute of Technology Press, Cambridge, MA, 1961.
5. M. J. E. Golay, "Notes on Digital Coding," *Proc. IRE* **37,** 657 (1949).
6. J. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *J. Soc. Ind. Appl. Math.* **8,** 300 (1960).
7. J. Cocke, "Lossless Symbol Coding with Nonprimes," *IRE Trans. Elect. Comput. and Info. Theory* **IT-5,** 33 (1959).

8. A. M. Patel, "Shift Register Implementation of b-Adjacent Codes," *Technical Report 00-2117*, IBM Corporation, Poughkeepsie, NY, 1970, also IEEE Computer Society Repository R72-260.
9. A. M. Patel, "The b-Adjacent Codes in Cyclic Form," *Technical Report 00-2082*, IBM Corporation, Poughkeepsie, NY, 1970, also IEEE Computer Society Repository R72-259.
10. S. J. Hong and A. M. Patel, "A General Class of Maximal Codes for Computer Applications," *IEEE Trans. Comput.* **C-21,** 1322 (1972).
11. The track numbers shown in Fig. 2 are algebraic track numbers for code description and computations. The physical placement of the tracks need not be the same. In IBM 3420 tapes and in the proposed American National Standard for 6250 BPI Group Coded Recording, the physical track numbers correspond to the ORC track number in the following manner:
    Algebraic track number: 1 4 7 P 3 6 0 2 5
    Physical track number: 1 2 3 4 5 6 7 8 9
12. For every r, there exists at least one irreducible polynomial of degree r[4].
13. The external pointers could be hardware indicators for inadequate amplitude or phase of the received signal corresponding to a particular track. This could also be any other signals indicating errors in a track.

Received April 2, 1974

A. M. Patel is located at the IBM General Products Division laboratory, San Jose, CA 95193; S. J. Hong is located at the IBM System Products Division Laboratory, Poughkeepsie, NY 12602; he is on a temporary teaching assignment at the University of Illinois for the scholastic year 1974-75.