

Two-Dimensional Dot Codes for Product Identification

WIL J. VAN GILS, MEMBER, IEEE

Abstract—In automated manufacturing the two-dimensional representation of product identification numbers by means of square dot codes offers a number of advantages over the use of bar codes. In this paper a method is described for the encoding of product identification numbers into square matrices of round dots on a contrasting background. It consists of an optimal source coding scheme for the encoding of identification numbers into channel message words, and a channel coding scheme offering error-detection and error-correction for random bit errors (dot corruptions). For the channel coding scheme, a special class of error-correcting codes is introduced, called *square-cyclic codes*. It is shown how to transform quasi-cyclic and (shortened) cyclic codes into square-cyclic codes.

1. INTRODUCTION

THE WIDESPREAD USE of bar codes in automated manufacturing clearly shows the need for an automatically readable product identification code. A bar code is built up from a number of parallel bars. The relative widths and mutual distances of these bars determine the meaning of the bar code.

We believe, however, that dot codes provide a better alternative to bar codes in this area of technology. A dot code consists of a square matrix of round dots on a contrasting background. The meaning of the dot code is determined by the absence or presence of dots. In a dot code the information is recorded in two dimensions, whereas in a bar code only one direction is used to encode information. This difference enables the dot code to offer higher information density, thereby allowing smaller product identification areas (see Fig. 1). For example, at the flat top of an electric motor shaft there is not enough room for a bar code. Furthermore, in automated manufacturing it is easy to write the dot codes onto the mechanical parts by an engraving process. With bar codes this would be more complicated. The dot codes can be read by a standard TV camera and can be recognized by a relatively inexpensive picture processing system.

Therefore, we shall introduce a method for the transmission of numbers from one point to another point by means of square matrices of round dots. These square dot matrices can be translated into square binary matrices by repre-

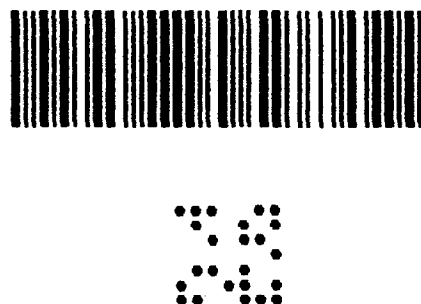


Fig. 1. Comparison of 7×7 dot code and standard "code 10" bar code with same information capacity. Diameter of dots has been chosen equal to linewidth of wide black bars in bar code. Both codes have been printed with dot-addressable matrix printer.

senting the presence of a dot by a one (1) and the absence of a dot by a zero (0). In a practical situation it was observed that only random dot corruptions (causing random bit errors) occurred in the dot squares. These errors were due to printing imperfections, dust particles, and reading failures. Furthermore, because of the possibly random rotation of the mechanical parts during the manufacturing process, recognition of the dot matrices should be possible irrespective of the orientation of the matrices. For example, one should again think of a square dot matrix on the flat top of a rotated shaft of an electric motor, without any synchronization indication outside the dot matrix.

In this paper we give a possible solution to this transmission problem. To this end, we distinguish the following steps in the transmission process.

- 1) *Source Encoding*: The encoding of the source message set (the integers i , $0 \leq i \leq M-1$) into the channel message set (a subset of the set of all binary vectors of length k).
- 2) *Channel Encoding*: The encoding of channel message words into codewords (square binary matrices) of a binary linear $[n, k]$ code such that random bit errors can be corrected and detected, and such that rotation of a codeword does not cause a decoding error.
- 3) *Transmission*: The actual transmission of the dot matrix through the noisy channel. During transmission dot matrices can also be rotated over 90, 180, and 270 degrees.
- 4) *Channel Decoding*: The decoding of a rotated corrupted codeword into an estimate of the transmitted rotated channel message word.

Manuscript received May 29, 1985; revised September 30, 1986. This paper was presented in part at the Sixth Symposium on Information Theory in the Benelux, Mierlo, The Netherlands, May 23–24, 1985, and at the IEEE International Symposium on Information Theory, Brighton, England, June 24–28, 1985.

The author is with Philips Research Laboratories, P.O. Box 80.000, 5600 JA Eindhoven, The Netherlands.

IEEE Log Number 8613582.

- 5) *Source Decoding*: The decoding of the estimate of the rotated channel message word into an estimate of the source message.

This scheme is illustrated in Fig. 2.

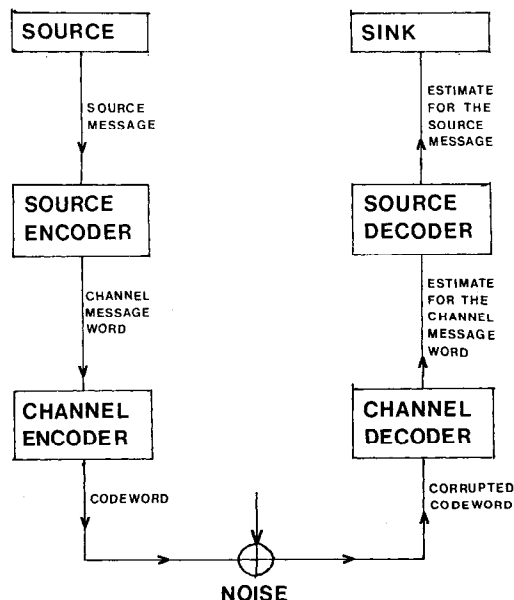


Fig. 2. Transmission scheme of square binary matrices.

We shall start by considering the structure the error-correcting code should have. This will result in the definition of the class of so-called *square-cyclic codes*. In Section III we describe two source coding schemes that nicely fit the concept of square-cyclic codes. One is the optimal one in the sense that it uses the minimum number of bits to encode a source message into a channel message word. The other scheme is not optimal, but gives rise to a very simple and fast encoding/decoding algorithm.

In general it is not possible to transform a square-cyclic generator matrix into a systematic square-cyclic generator matrix. However, it is possible to transform it into a *canonical* square-cyclic generator matrix that makes transformation of a codeword into its corresponding channel message word relatively easy. This canonical square-cyclic generator matrix is described in Section IV. In Section V we give the construction of square-cyclic codes from well-known quasi-cyclic and (shortened) cyclic codes, respectively. In fact, a square-cyclic code is a quasi-cyclic or an extended quasi-cyclic code itself. In this paper no new error-correcting codes are introduced, but a number of well-known, well-structured codes are transformed such that they can be used in our particular application.

II. DEFINITION OF SQUARE-CYCLIC CODES

If we use an arbitrary t -error-correcting code on our channel, then the decoder has to deal with the corruption as well as with the rotation of the transmitted codeword. One possibility for the decoder is to apply the decoding algorithm to all four rotations of the received word and check which one gives a codeword. In this case it is very

possible that among the four estimates of a received word with at most t errors in it there is more than one codeword. Furthermore, in this approach the decoding algorithm is applied four times for each received word. Hence, it seems useful to choose a linear error-correcting code, whose codewords have the form of square binary matrices, such that rotations over 90, 180, and 270 degrees of any codeword again give codewords in the code. In that case we can first correct the errors in the corrupted and rotated codeword, and afterwards we can deal with the rotation.

This approach implies that a codeword together with its rotations over 90, 180, and 270 degrees should represent the same channel message word. Hence, the number of channel message words is less than or equal to the number of equivalence classes of the relation defined by the rotation of codewords in the above-mentioned code. We will come back to this source coding problem in Section III.

We next define the class of codes, called square-cyclic codes, having the property that a rotation of any codeword of such a code is again a codeword of that code.

Definition 1: A binary linear $[n, k]$ code C is called *square-cyclic* if it is possible to transform the codewords of C into square $m \times m$ matrices for some $m, m^2 \geq n$, allowing empty positions, such that the rotations of any codeword in square matrix form over 90, 180, and 270 degrees again give codewords of the code C in square matrix form.

Example: Consider the $[48, 24]$ binary linear code C with generator matrix

$$\begin{bmatrix} I & O & O & O & O & O & A & B & C & D & E & F \\ O & I & O & O & O & O & PF & A & B & C & D & E \\ O & O & I & O & O & O & PE & PF & A & B & C & D \\ O & O & O & I & O & O & PD & PE & PF & A & B & C \\ O & O & O & O & I & O & PC & PD & PE & PF & A & B \\ O & O & O & O & O & I & PB & PC & PD & PE & PF & A \end{bmatrix}$$

where all submatrices are 4×4 matrices, I is the identity matrix, O is the all-zero matrix, and A, B, C, D, E, F, P are 4×4 circulant matrices with first rows (1011) , (1101) , (0101) , (1110) , (0100) , (1011) , (0100) , respectively. The 24 information bits of a codeword c are $m_0, m_1, \dots, m_{23}$, the 24 parity bits are $p_0, p_1, \dots, p_{23}$, and $c = (m_0, m_1, \dots, m_{23}, p_0, p_1, \dots, p_{23})$. These 48 bits are put into a 7×7 square according to the scheme shown in Fig. 3. The middle point is not used in this case. From Fig. 3 we see that the information bits are put at the border of the square, and the check bits are put in the inner 5×5 square. Of course, other distributions are also possible.

m_0	m_{23}	m_{19}	m_{15}	m_{11}	m_7	m_3
m_4	p_0	p_{15}	p_{11}	p_7	p_3	m_{22}
m_8	p_4	p_{16}	p_{23}	p_{19}	p_{14}	m_{18}
m_{12}	p_8	p_{20}		p_{22}	p_{10}	m_{14}
m_{16}	p_{12}	p_{17}	p_{21}	p_{18}	p_6	m_{10}
m_{20}	p_1	p_5	p_9	p_{13}	p_2	m_6
m_1	m_5	m_9	m_{13}	m_{17}	m_{21}	m_2

Fig. 3. Distribution of message bits and check bits in square matrix for $[48, 24]$ code.

The code has minimum Hamming distance 12. Its construction will be given in Section V together with an elegant decoding algorithm. From the special structure of the generator matrix and the mapping of the code bits into 7×7 square matrices according to the scheme in Fig. 3, it is easy to see that the rotations of a codeword in C (in square matrix form) over 90, 180, and 270 degrees also give codewords in C . For an example, see Fig. 4.

```

1010011  1111100
1000101  1001011
1001101  0111110
111 111  001 111
1111101  1001100
0101110  0001111
0101010  1111100

```

Fig. 4. Codeword in $[48, 24]$ code and its rotation over 90 degrees.

From Definition 1 it is immediately clear that the length n of a square-cyclic code should be a multiple of four ($n = 4s$) or a multiple of four plus one ($n = 4s + 1$) if the middle position of the square is also used (that is only possible if m and n are odd). Without loss of generality we order the bits in codewords of a square-cyclic code such that positions that are mapped onto each other by rotations of the square matrices are grouped together into four-tuples. In other words, if the code bits $c_0, c_1, \dots, c_{n-1}$ of a codeword $(c_0, c_1, \dots, c_{n-1})$ are put in an $m \times m$ square matrix, where m is the smallest integer such that $n \leq m^2$ and m is odd if n is odd, then if code bit c_{4f} is put in position (i, j) , the code bits c_{4f+1} , c_{4f+2} , and c_{4f+3} are put in positions $(m+1-j, i)$, $(m+1-i, m+1-j)$, and $(j, m+1-i)$, respectively. Note that positions in an $m \times m$ square matrix are numbered by the pairs (i, j) , where $i, j = 1, 2, \dots, m$. Furthermore, if m and n are both odd, the last code bit c_{n-1} is put in the middle of the square matrix, i.e., in position $((m+1)/2, (m+1)/2)$. The four rotations of a codeword (in square matrix form) of a square-cyclic $[n, k]$ code give 4, 2, or 1 different codewords. For $i = 1, 2$, and 4, define $C(i)$ to be the set of codewords c of C whose four rotations give at most i mutually different codewords. It is easy to see that the $C(i)$, $i = 1, 2, 4$, are linear subspaces of the code C . In particular $C(4)$ is equal to C . Let p , q , and r be (the unique) nonnegative integers such that the following equations hold:

$$k = \dim C(4) = 4p + 2q + r$$

$$\dim C(2) = 2p + 2q + r$$

$$\dim C(1) = p + q + r$$

where $\dim C(i)$ denotes the dimension of $C(i)$. Hence the square-cyclic $[n, k]$ code C of length n and dimension k

has a generator matrix of the form

$$G = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1s} \\ A_{21} & A_{22} & \cdots & A_{2s} \\ \vdots & \vdots & & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{ps} \\ B_{11} & B_{12} & \cdots & B_{1s} \\ \vdots & \vdots & & \vdots \\ B_{q1} & B_{q2} & \cdots & B_{qs} \\ C_{11} & C_{12} & \cdots & C_{1s} \\ \vdots & \vdots & & \vdots \\ C_{r1} & C_{r2} & \cdots & C_{rs} \end{bmatrix}$$

if n is even, and the same matrix as above with an extra column

$$(d_1, d_2, \dots, d_p, e_1, e_2, \dots, e_q, f_1, \dots, f_r)^T$$

added to it, if n is odd. The matrices $A_{11}, A_{12}, \dots, A_{1s}$, $A_{21}, \dots, A_{ps}$ are 4×4 circulant matrices, $B_{11}, B_{12}, \dots, B_{1s}, B_{21}, \dots, B_{ps}$ are 2×4 circulant matrices, $C_{11}, \dots, C_{1s}, C_{21}, \dots, C_{rs}$ are 1×4 circulant matrices, $d_1, \dots, d_p$ are 4×1 all-zero or all-one row vectors, $e_1, \dots, e_q$ are 2×1 all-zero or all-one row vectors, and $f_1, \dots, f_r$ are 1×1 all-zero or all-one row vectors. Here a $u \times v$ matrix is called circulant if, for all $i = 1, 2, \dots, u-1$, row $(i+1)$ is the cyclic shift of one position to the right of row i , and row 1 is the cyclic shift of one position to the right of row u . By this definition a 3×4 binary circulant matrix consists of twelve zeros or twelve ones; that is, it is a repetition of three 1×4 binary circulant matrices. If u is smaller (larger) than v , a $u \times v$ circulant matrix is a repetition of $u \times u$ ($v \times v$) circulant matrices.

Of course, parity-check matrices of square-cyclic codes also have an identical form. Generator and parity-check matrices of this form are called *square-cyclic generator* and *parity-check matrices*, respectively. The most interesting square-cyclic codes are the ones that fill a square matrix completely ($n = m^2$) or nearly completely if m is odd ($n = m^2 - 1$). It should be noted that in certain applications, some positions in a dot matrix, for example the corner dots, need to be reserved for other purposes.

III. SOURCE ENCODING AND DECODING

Because we use a square-cyclic code on our channel, and because codewords can be rotated during transmission, it is not possible to attach different source messages to the four rotations of a particular codeword. A codeword and its rotations should all represent the same source message.

To a rotation of a codeword in a code with a generator matrix like the ones in Section II there corresponds a "rotation" of the k -bit information vector associated with it. Let a k -bit ($k = 4p + 2q + r$) information vector m be denoted by

$$m = (u_1, u_2, \dots, u_p, v_1, v_2, \dots, v_q, w_1, w_2, \dots, w_r)$$

where the vectors u_i ($i=1, \dots, p$) are row vectors of length four, the vectors v_i ($i=1, \dots, q$) are row vectors of length two, and the vectors w_i ($i=1, \dots, r$) are row vectors of length one. The partitioning of the message m depends on the parameters p , q , and r , which are parameters determined by the channel code (Section II). By the rotation $R^i(m)$ of m over i positions is meant the cyclic shifts of the individual components $u_1, \dots, u_p, v_1, \dots, v_q, w_1, \dots, w_r$ over i positions to the right simultaneously. So the rotation over one position of $m = (1000, 0011, 10, 00, 11, 0, 1, 0)$ is $R(m) = (0100, 1001, 01, 00, 11, 0, 1, 0)$.

We define the relation $\sim$ on the set F_2^k of k -bit binary vectors by

$$m_1 \sim m_2: \Leftrightarrow (R^i(m_1) = m_2, \text{ for } i = 0, 1, 2, \text{ or } 3).$$

This relation is an equivalence relation. Its equivalence classes contain one, two, or four elements. As we have seen above, the maximum size of the set of source messages equals the number of equivalence classes of the relation $\sim$ in F_2^k . In the source encoder we add to any source message a representative of an equivalence class. This set of representatives is called the *channel message set*. In the channel encoder these channel message words are mapped onto codewords of the error-correcting code. Because the set of channel message words has less than 2^k elements, the set of possible channel encoder outputs is not the entire code space, but only a part of it. However, due to the rotations during transmission, any codeword can appear as input of the channel decoder.

A k -bit vector $(u_1, \dots, u_p, v_1, \dots, v_q, w_1, \dots, w_r)$ is said to have period i whenever it is contained in an equivalence class having i elements. The set F_2^k contains 2^{p+q+r} elements of period one, $2^{2p+2q+r}$ elements of period at most two, and $2^{4p+2q+r}$ elements of period at most four. So, it contains $2^{2p+2q+r} - 2^{p+q+r}$ elements of period two and $2^{4p+2q+r} - 2^{2p+2q+r}$ elements of period four. Hence, the number of equivalence classes equals

$$\begin{aligned} M(p, q, r) &:= (2^{4p+2q+r} - 2^{2p+2q+r})/4 \\ &\quad + (2^{2p+2q+r} - 2^{p+q+r})/2 + 2^{p+q+r} \\ &= 2^{4p+2q+r-2} + 2^{2p+2q+r-2} + 2^{p+q+r-1} \end{aligned}$$

Next we build a channel message set by choosing $M(p, q, r)$ representatives of the equivalence classes. Therefore we define the following sets:

$$\begin{aligned} P &= \{(1000), (1100), (1110)\} \\ Q &= \{(0000), (1111)\} \\ S &= \{(1010)\} \\ T &= \{(1000), (0100), (1100), (0110), (1110), (0111)\} \\ U &= \{(1010), (0101), (0000), (1111)\} \\ V &= \{(10)\} \\ W &= \{(00), (11)\}. \end{aligned}$$

It is easy to see that we can take representatives whose first components u_1 are in the set $P \cup Q \cup S$. Therefore,

our first rule requires all representatives to have a first component u_1 in the set $P \cup Q \cup S$. If u_1 is in P this rule determines a unique representative for the corresponding equivalence class. If u_1 is in $Q \cup S$ we could have more than one candidate. Additional rules are necessary to determine a unique representative.

In an equivalence class containing elements with first component u_1 in Q , at least one element has a second component u_2 in $P \cup Q \cup S$. In this case (u_1 in Q) our second rule is the same as the first one, but now for the second component u_2 ; i.e., u_2 should be in $P \cup Q \cup S$.

In an equivalence class that contains an element with first component u_1 in S , at least one element has a second component u_2 in $T \cup U$. In this case (u_1 in S) our second rule requires a representative to have a second component u_2 in $T \cup U$. If u_2 is in T this rule determines a unique representative. If u_2 is in U our third rule requires that the third component u_3 of a representative is again in $T \cup U$ (of course only if $p \geq 3$).

The above process and a similar process for the components v_j ($j=1, \dots, q$) is repeated until a unique representative is found for all equivalence classes. Formally the process is described by the following rules.

The set F of representatives is recursively defined by

$$\begin{aligned} F &= \bigcup_{u_1 \in P \cup Q \cup S} F(u_1), \quad \text{for } p > 0, \\ F &= \bigcup_{v_1 \in V \cup W} H(v_1), \quad \text{for } p = 0, q > 0, \\ F &= F_2^r, \quad \text{for } p = 0, q = 0, r > 0. \end{aligned}$$

For $1 \leq i < p$

$$F(u_1, \dots, u_i) = \{(u_1, \dots, u_i, x) | x \in F_2^{4(p-i)+2q+r}\},$$

for $u_i \in P$

$$F(u_1, \dots, u_i) = \bigcup_{u_{i+1} \in P \cup Q \cup S} F(u_1, \dots, u_i, u_{i+1}),$$

for $u_i \in Q$

$$F(u_1, \dots, u_i) = \bigcup_{u_{i+1} \in T \cup U} G(u_1, \dots, u_i, u_{i+1}),$$

for $u_i \in S$

$$G(u_1, \dots, u_i) = \{(u_1, \dots, u_i, x) | x \in F_2^{4(p-i)+2q+r}\},$$

for $u_i \in T$

$$G(u_1, \dots, u_i) = \bigcup_{u_{i+1} \in T \cup U} G(u_1, \dots, u_i, u_{i+1}),$$

for $u_i \in U$

$$F(u_1, \dots, u_p) =$$

$$G(u_1, \dots, u_p) = \bigcup_{v_1 \in V \cup W} H(u_1, \dots, u_p, v_1), \quad \text{for } q > 0$$

$$F(u_1, \dots, u_p) =$$

$$G(u_1, \dots, u_p) = \{(u_1, \dots, u_p, z) | z \in F_2^r\}, \quad \text{for } q = 0.$$

For $1 \leq j < q$

$$H(u_1, \dots, u_p, v_1, \dots, v_j) \\ = \{(u_1, \dots, u_p, v_1, \dots, v_j, y) | y \in F_2^{2(q-j)+r}\}, \\ \text{for } v_j \in V.$$

$$H(u_1, \dots, u_p, v_1, \dots, v_j) \\ = \bigcup_{v_{j+1} \in V \cup W} H(u_1, \dots, u_p, v_1, \dots, v_j, v_{j+1}), \\ \text{for } v_j \in W.$$

$$H(u_1, \dots, u_p, v_1, \dots, v_q) \\ = \{(u_1, \dots, u_p, v_1, \dots, v_q, z) | z \in F_2^r\}.$$

From this definition of the channel message set it is easy to derive algorithms for the encoding of source messages into channel message words and for the decoding of channel message words into source messages. The encoding and decoding algorithms are shown in Figs. 5 and 6, respectively. In these algorithms the following definitions are used.

Definition 2:

$$A(x, y, z) := 2^{4x+2y+z}$$

$$C(x, y, z) := 2^{4x+2y+z-1} + 2^{2x+2y+z-1}$$

$$M(x, y, z) := 2^{4x+2y+z-2} + 2^{2x+2y+z-2} + 2^{x+y+z-1}$$

$$B(y, z) := 2^{2y+z}.$$

For a binary vector $y = (y_0, y_1, \dots, y_{t-1})$, $\text{bin}(y)$ is defined by

$$\text{bin}(y) = \sum_{i=0}^{t-1} y_i 2^{t-1-i}.$$

For natural numbers s and t such that $s < 2^t$, $\text{invbin}_t(s)$ is defined by

$$\text{invbin}_t(s) = (y_0, y_1, \dots, y_{t-1})$$

where

$$s = \sum_{i=0}^{t-1} y_i 2^{t-1-i}.$$

For a binary vector $x = (x_0, x_1, x_2, x_3)$ of length four, the function $f(x)$ is defined as the minimum number of positions x has to be shifted cyclically to the left such that it matches one of the elements of $P \cup Q \cup S$, and the function $g(x)$ as the minimum number of even positions x has to be shifted cyclically to the left such that it matches one of the elements of $T \cup U$. For a binary vector $z = (z_0, z_1)$ of length two, the function $h(z)$ is defined as the minimum number of positions z has to be shifted cyclically to the left such that it matches one of the elements of $V \cup W$. The vector $x^{(s)}$ is defined as the result of cyclically shifting x s positions to the left.

Note that the numbers $A(\cdot)$, $C(\cdot)$, $M(\cdot)$, and $B(\cdot)$ correspond to the sizes of the sets $F(\cdot)$, $G(\cdot)$, and $H(\cdot)$ defined above. The input to the encoding algorithm is a source message m , $0 \leq m \leq M(p, q, r) - 1$, and the out-

put is a channel message word $(u_1, \dots, u_p, v_1, \dots, v_q, w_1, \dots, w_r)$. The input to the source decoding algorithm is a rotation of the channel message word, and the output is the corresponding source message.

We call the encoding and decoding schemes described above optimal because they use the largest channel message set possible. In this scheme the encoding of source messages costs $4p + 2q + r - \log_2 M(p, q, r) < 2$ bits.

Example: In the [48, 24] square-cyclic code defined above, $p = 6$, $q = r = 0$. So $M(p, q, r) = M(6, 0, 0) = 2^{22} + 2^{10} + 2^5 = 4\,195\,360$. For example, the source message 1 375 211 corresponds to the channel message (1100, 0100, 1111, 0100, 0001, 1011), and the source message 3 374 855 corresponds to the channel message (0000, 1111, 1010, 0101, 0100, 0111).

Example: Consider a channel code with dimension $k = 24$ and suppose the corresponding 3-tuple (p, q, r) of this code equals $(4, 3, 2)$. So $M(p, q, r) = M(4, 3, 2) = 4\,210\,944$. For example, the source message 1 375 211 corresponds to the channel message (1100, 0100, 1111, 1011, 11, 10, 10, 1, 1), and the source message 3 376 658 corresponds to the channel message (0000, 1111, 1010, 0101, 11, 00, 10, 1, 0).

If the number M of source messages is at most 2^{k-3} , or the encoding/decoding algorithms described above take too much time, then a simpler source coding scheme can be used. This encoding scheme adds at most three bits to a source message to build a channel message word. The encoding and decoding are described below.

1) $p > 0$: Encoding—Set the first three bits of u_1 equal to 1, 0, 0 and use the remaining $4p + 2q + r - 3$ bits of the channel message word for the $4p + 2q + r - 3$ source message bits.

Decoding—Rotate the channel message word until the first three positions of the channel message word are equal to 1, 0, 0. The remaining bits are a binary representation of the source message.

2) $p = 0$ and $q > 0$: Encoding—Set v_1 equal to (10) and use the remaining $2q + r - 2$ bits of the channel message word for the $2q + r - 2$ source message bits.

Decoding—Rotate the channel message word until the first two positions of the channel message word are equal to 1, 0. The remaining bits are a binary representation of the source message.

3) $p = 0$ and $q = 0$: The channel message word is the binary representation of the source message.

So in the most general case ($p > 0$), three bits are added to a source message to produce a channel message word.

IV. A CANONICAL GENERATOR MATRIX OF A SQUARE-CYCLIC CODE

In general it is necessary to multiply a part (k bits) of a codeword by a regular matrix to retrieve the corresponding information vector. When a systematic generator matrix for the code is used this is not necessary, the information vector being a part of the codeword. Unfortunately, a square-cyclic code does not always have a square-cyclic systematic generator matrix. However, it does have another

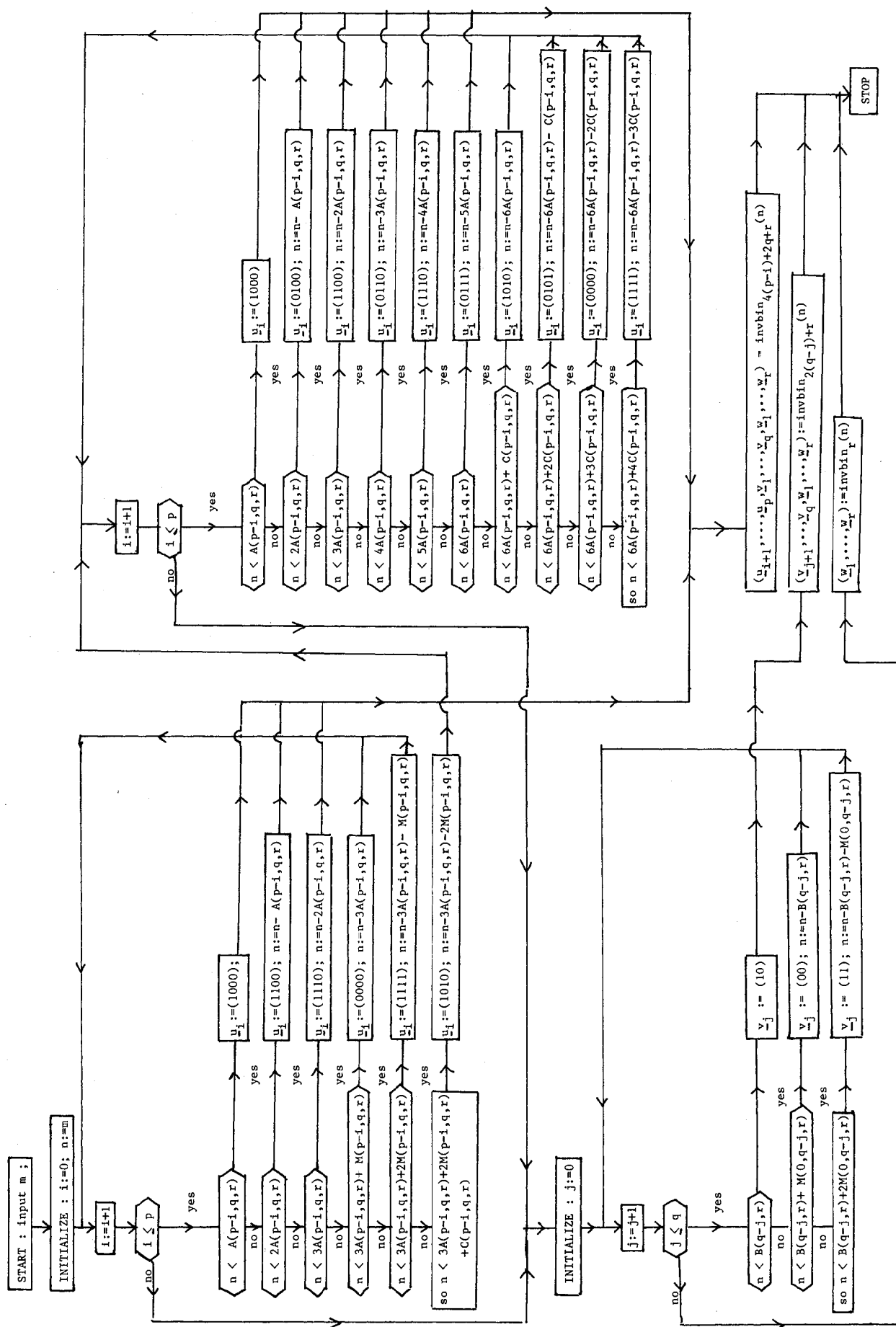


Fig. 5. Encoding algorithm for optimal source code.

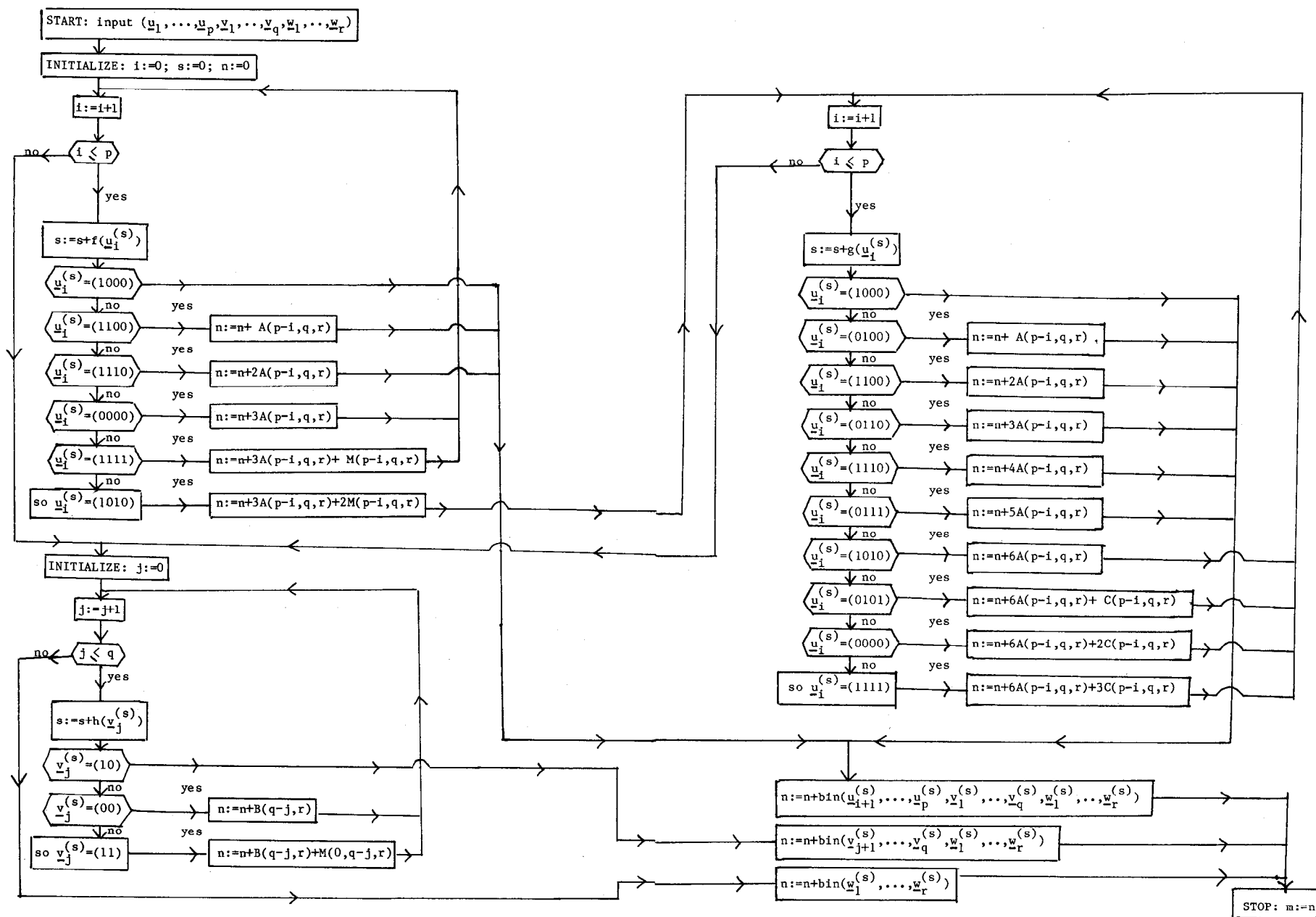


Fig. 6. Decoding algorithm for optimal source code.

nicely structured square-cyclic generator matrix, the so-called canonical form. When this generator matrix is used, the transformation of codewords into information vectors does not require multiplication by a $k \times k$ regular matrix. In this section we will show how to transform an arbitrary square-cyclic generator matrix into such a canonical form. It should be noted that a square-cyclic code of course also has a systematic generator matrix, but if it is not square-cyclic then there is no nice correspondence between the rotation of a codeword and the rotation of a channel message word.

Let G be a generator matrix of a square-cyclic code of length $4s$, G having the form given in Section II. The 4×4 circulant matrices A_{ij} can be divided into six classes by considering their first rows. The first row of a 4×4 circulant matrix is a cyclic shift of one of the following six vectors: (0000), (1111), (1100), (1010), (1000), or (1110). The first four types yield singular matrices, the last two types yield regular matrices whose inverses are also circulant.

Now consider the generator matrix G . In the first four rows ($A_{11}, A_{12}, \dots, A_{1s}$) at least one 4×4 circulant is regular; otherwise G would not have full rank. Without loss of generality, let us suppose that A_{11} is regular. By premultiplying G by the proper regular matrix we obtain a matrix of the following form:

$$G' = \begin{bmatrix} I & D_{12} & \dots & D_{1s} \\ O & D_{22} & \dots & D_{2s} \\ \vdots & \vdots & & \vdots \\ O & D_{p2} & \dots & D_{ps} \\ O & E_{12} & \dots & E_{1s} \\ \vdots & \vdots & & \vdots \\ O & E_{q2} & \dots & E_{qs} \\ O & F_{12} & \dots & F_{1s} \\ \vdots & \vdots & & \vdots \\ O & F_{r2} & \dots & F_{rs} \end{bmatrix}$$

where $I, O, D_{ij}, E_{ij}, F_{ij}$ are the 4×4 identity matrix, the all-zero matrix, 4×4 circulants, 2×4 circulants, and 1×4 circulants, respectively. By repeating this process p times for successive four-tuples of rows, after a proper permutation of the columns, we get a generator matrix of the form

$$G'' = \begin{bmatrix} I & O & O & \dots & O & K_{1p+1} & \dots & K_{1s} \\ O & I & O & \dots & O & K_{2p+1} & \dots & K_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ O & O & O & \dots & I & K_{pp+1} & \dots & K_{ps} \\ O & O & O & \dots & O & L_{1p+1} & \dots & L_{1s} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ O & O & O & \dots & O & L_{qp+1} & \dots & L_{qs} \\ O & O & O & \dots & O & M_{1p+1} & \dots & M_{1s} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ O & O & O & \dots & O & M_{rp+1} & \dots & M_{rs} \end{bmatrix}$$

where the matrices K_{ij} , L_{ij} , and M_{ij} are 4×4 , 2×4 , and 1×4 circulants, respectively.

At least one of the 2×4 circulant matrices $L_{1,p+1}, \dots, L_{1s}$ has rank two; i.e., it has the form

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

or

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix};$$

otherwise the generator matrix G would not have full rank. In an analogous way to that above, the matrix G'' can be transformed into the matrix

$$G''' = \begin{bmatrix} I & O & \dots & O & X_{1p+1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & X_{1s} \\ O & I & & \vdots & X_{2p+1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & X_{2s} \\ \vdots & & \ddots & O & \vdots & & & & & & & & & \vdots \\ O & \dots & O & I & X_{pp+1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & X_{ps} \\ OO & \dots & \dots & OO & II & OO & \dots & OO & Y_{1p+q+1} & \dots & \dots & \dots & \dots & Y_{1s} \\ \vdots & \ddots & & \vdots & OO & II & & \vdots & Y_{2p+q+1} & \dots & \dots & \dots & \dots & Y_{2s} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & OO & \vdots & & & & & \vdots \\ OO & \dots & \dots & OO & OO & \dots & OO & II & Y_{qp+q+1} & \dots & \dots & \dots & \dots & Y_{qs} \\ 0000 & \dots & \dots & 0000 & 0000 & \dots & \dots & 0000 & 1111 & 0000 & \dots & 0000 & Z_{1p+q+r+1} & Z_{1s} \\ \vdots & & & \vdots & \vdots & & & \vdots & 0000 & 1111 & & \vdots & Z_{2p+q+r+1} & Z_{2s} \\ \vdots & & & \vdots & \vdots & & & \vdots & \vdots & & \ddots & 0000 & \vdots & \vdots \\ 0000 & \dots & \dots & 0000 & 0000 & \dots & \dots & 0000 & 0000 & \dots & 0000 & 1111 & Z_{rp+q+r+1} & Z_{rs} \end{bmatrix},$$

where X_{ij} , Y_{ij} , and Z_{ij} are 4×4 , 2×4 , and 1×4 circulant matrices, respectively. The matrix G''' is called a *canonical generator matrix* of the code C . For a square-cyclic code of length $4s+1$ the procedure is similar.

We have shown that a square-cyclic generator matrix of a square-cyclic code can be transformed into a matrix which, after proper column permutations, has the above canonical form. By using a canonical generator matrix, it is easy to retrieve a channel message word from a codeword. If $q = r = 0$, then the matrix G''' is systematic, so it is easy to retrieve the information vector $(u_1, \dots, u_p) = (c_1, \dots, c_p)$ from the codeword $(c_1, \dots, c_s)$, where c_i ($i=1, \dots, s$) has length four. If $q \neq 0$, then we have to subtract from the vector $(c_{p+1}, \dots, c_{p+q})$ the contribution of the vector $(c_1, \dots, c_p)$ to obtain the vector $(v_1, v_1, v_2, v_2, \dots, v_q, v_q)$, i.e., two times the information vector $(v_1, \dots, v_q)$. The case $r \neq 0$ is similar.

V. CONSTRUCTION OF SQUARE-CYCLIC CODES

In this section we will show how square-cyclic codes can be constructed by transforming quasi-cyclic and (shortened) cyclic codes. In the following we shall allow the possibility of transforming an $f^*g \times f^*h$ circulant matrix A into an $f^*g \times f^*h$ matrix A' , consisting of $f \times f$ circulants, by permuting the rows and columns of A . For example, this can be done by the following permutation of rows and columns. For $i=1, 2, \dots, f$, $j=1, 2, \dots, g$, row $(i-1)g+j$ is mapped onto row $(j-1)f+i$, and for $i=1, 2, \dots, f$, $j=1, 2, \dots, h$, column $(i-1)h+j$ is mapped onto column $(j-1)f+i$. By applying transformations of this kind, many nicely structured codes, such as (shortened) cyclic and quasi-cyclic codes, can be transformed into square-cyclic codes. Among them are many optimal codes; i.e., they have the maximum minimum distance for a binary linear code of the same length and dimension. Because of their (quasi-) cyclic structure, well-known decoding techniques can be used to decode these codes.

In Subsections A and B we consider the construction of square-cyclic codes from quasi-cyclic and (shortened) cyclic codes, respectively. We shall frequently use the following permutations.

Definition 3: The permutation $P(fg, f)$ maps $(i-1)g+j$ onto $(j-1)f+i$ for $i=1, 2, \dots, f$ and $j=1, 2, \dots, g$. The permutation $\text{REV}(f)$ maps i onto $f+1-i$ for $i=1, 2, \dots, f$.

A. CONSTRUCTION OF SQUARE-CYCLIC CODES FROM QUASI-CYCLIC CODES

A code is called quasi-cyclic if there is some integer v such that every cyclic shift of a codeword by v places is again a codeword. So, after a proper permutation of the columns of the generator matrix of a quasi-cyclic code, we get a generator matrix which is composed of circulant matrices. We have seen that a circulant matrix can be transformed into a matrix composed of smaller circulants. If the sizes of the circulants in the generator matrix of a quasi-cyclic code are proper, then the generator matrix can

be transformed into a square-cyclic generator matrix by applying permutations of its rows and columns.

In [1], [5], [6], [10], [11] quasi-cyclic codes are constructed whose generator matrices are composed of $g \times g$ circulants, where g is a multiple of four. Those circulants can thus be transformed into compositions of 4×4 circulants by applying the permutation $P(4g, 4)$ to the rows and columns. These constructions give square-cyclic codes with $p \neq 0, q = r = 0$.

The very elegant decoding methods for quasi-cyclic codes can be used in these cases. For example, the decoding method of double-circulant codes described by Karlin [6] gives rise to a very efficient decoding algorithm when decoding is performed by software in a microprocessor.

Example: Consider the double-circulant [48, 24, 12] quadratic residue (QR) code [5]. This code has a generator matrix $G = [I \ A]$, where I is the 24×24 identity matrix and A is the 24×24 circulant matrix with top row $a = (a_0, a_1, \dots, a_{23}) = (11010101110100101111001)$ [5]. By performing the permutation $P(24, 4)$ on the rows and columns of I and A , we bring the generator matrix G into square-cyclic form. The resulting generator matrix was given in Section II. The very special structure of the [48, 24] QR code (self-dual and double circulant) makes decoding relatively easy. The first step in the decoding process is to apply the inverse of the permutation $P(24, 4)$ to the left and right half of a received word. This brings the code back to its original double-circulant form. For decoding of the double-circulant code we use the "redundant" parity-check matrix

$$H = \begin{bmatrix} I & A \\ A^{-1} & I \end{bmatrix}$$

and the corresponding "redundant" syndromes. For convenience, we represent vectors $b = (b_1, b_2)$, where b_1, b_2 have length 24, by the polynomial pair $(b_1(x), b_2(x))$, where $b_1(x)$ and $b_2(x)$ are polynomials of degree at most 23:

$$b_i = (b_{i0}, b_{i1}, \dots, b_{i23}), \quad b_i(x) = \sum_{j=0}^{23} b_{ij}x^j, \quad \text{for } i=1, 2.$$

An error pattern is then represented by $(e_1(x), e_2(x))$, the corrupted received codeword by $(r_1(x), r_2(x))$, and the syndrome by $(s_1(x), s_2(x)) = (r_1(x) + r_2(x)\bar{a}(x), r_1(x)a(x) + r_2(x))$, where $a(x) := a_0 + a_1x + a_2x^2 + \dots + a_{23}x^{23}$ and $\bar{a}(x) := x^{24}a(x^{-1})$. The following decoding algorithm, which is t -error-correcting and $(11-t)$ -error-detecting for $t \leq 5$, can easily be implemented by software in a microprocessor. In this algorithm one should skip the steps that start with $(t \geq i)$ if the desired correction capability is less than i . The function $\text{wt}(\cdot)$ denotes the Hamming weight function.

Decoding algorithm (t -error-correcting, $(11-t)$ -error-detecting):

- 1) input $(r_1(x), r_2(x))$
- 2) compute the syndrome

$$s_1(x) = r_1(x) + r_2(x)\bar{a}(x),$$

$$s_2(x) = r_1(x)a(x) + r_2(x)$$

- 3) if $\text{wt}(s_1(x)) \leq t$, then $\hat{e}_1(x) := s_1(x)$; else
- 4) if $\text{wt}(s_2(x)) \leq t$, then $\hat{e}_1(x) := 0$; else
- 5) ($t \geq 2$): if for some $j = 0, 1, \dots, 23$, $\text{wt}(s_1(x) + x^j \bar{a}(x)) \leq t-1$, then $\hat{e}_1(x) := s_1(x) + x^j \bar{a}(x)$; else
- 6) ($t \geq 3$): if for some $j = 0, 1, \dots, 23$, $\text{wt}(s_2(x) + x^j \bar{a}(x)) \leq t-1$, then $\hat{e}_1(x) := x^j$; else
- 7) ($t \geq 4$): if for some $i, j = 0, 1, \dots, 23, i \neq j$, $\text{wt}(s_1(x) + (x^i + x^j) \bar{a}(x)) \leq t-2$, then $\hat{e}_1(x) := s_1(x) + (x^i + x^j) \bar{a}(x)$; else
- 8) ($t \geq 5$): if for some $i, j = 0, 1, \dots, 23, i \neq j$, $\text{wt}(s_2(x) + (x^i + x^j) \bar{a}(x)) \leq t-2$, then $\hat{e}_1(x) = x^i + x^j$; else
- 9) detection of an error.

What is used in this algorithm is the fact that if the weight of the left or right half of a modified syndrome is small enough, then it is known where the errors can be found [7, ch. 16, Theorem 19, p. 513]. The decoding algorithm above is similar to the one described in Karlin [6].

B. Construction of Square-Cyclic Codes from Shortened Cyclic Codes

In [4] it is shown that generator matrices and parity-check matrices of cyclic codes can be partitioned into circulant matrices. The way this is done is described below.

Let n be an odd integer, and let b be minimal such that n divides $2^b - 1$. Let $\beta \in GF(2^b)$ be a primitive n th root of unity. Let $\{C_i: i \in A\}$, where A denotes a set of indices, be the set of cyclotomic cosets modulo n . Now consider a binary cyclic code C of length n having zeros $\gamma_i = \beta^i, i \in \bigcup_{j \in Z} C_j$, where Z is a subset of A .

If we number the positions in a codeword by the n th roots of unity $\beta^0, \beta^1, \dots, \beta^{n-1}$ such that conjugates of elements occupy adjacent positions in order of increasing powers of two, the parity-check matrix H of C has the following form:

$$H = \begin{array}{c|cccc|cccc|cccc|} 0 & d & 2d & 4d & \dots & e & 2e & 4e & \dots & \dots & \dots & f & 2f & 4f & \dots \\ \hline 1 & \gamma_i^d \gamma_i^{2d} \gamma_i^{4d} & \dots & \gamma_i^e \gamma_i^{2e} \gamma_i^{4e} & \dots & \dots & \dots & \gamma_i^f \gamma_i^{2f} \gamma_i^{4f} & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \gamma_j^d \gamma_j^{2d} \gamma_j^{4d} & \dots & \gamma_j^e \gamma_j^{2e} \gamma_j^{4e} & \dots & \dots & \dots & \gamma_j^f \gamma_j^{2f} \gamma_j^{4f} & \dots & \dots & \dots & \dots & \dots & \dots \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ 1 & \gamma_k^d \gamma_k^{2d} \gamma_k^{4d} & \dots & \gamma_k^e \gamma_k^{2e} \gamma_k^{4e} & \dots & \dots & \dots & \gamma_k^f \gamma_k^{2f} \gamma_k^{4f} & \dots & \dots & \dots & \dots & \dots & \dots \end{array}$$

where $d, e, \dots, f$ are representatives of the cyclotomic cosets modulo n and $i, j, \dots, k$ are in Z .

Every Galois field has a normal basis [7, p. 122]. Let $\alpha, \alpha^2, \alpha^4, \dots, \alpha^{2^{b-1}}$ be a normal basis of $GF(2^b)$ with respect to $GF(2)$. Now we map the parity-check matrix H onto its binary image H_{bin} with respect to the abovementioned normal basis. Every component in H is replaced by its binary representation (a column vector of length b) with respect to the normal basis $\alpha, \alpha^2, \alpha^4, \dots, \alpha^{2^{b-1}}$. From the resulting matrix all rows occurring more than once are deleted. The resulting matrix H_{bin} is a full-rank parity-

check matrix of the code C , and it is composed of circulant matrices

$$H_{\text{bin}} = \begin{array}{c|cccc|cccc|cccc|} 0 & d & 2d & 4d & \dots & e & 2e & 4e & \dots & \dots & \dots & f & 2f & 4f & \dots \\ \hline g & 1 & & & & & & & & & & & & & \\ 2g & 1 & & & & & & & & & & & & & \\ 4g & 1 & A_{gd} & & & A_{ge} & & & \dots & & & & A_{gf} & & \\ \vdots & \vdots & & & & \vdots & & & & & & & & & \\ \vdots & 1 & & & & & & & & & & & & & \\ - & 1 & & & & & & & & & & & & & \\ h & 1 & & & & & & & & & & & & & \\ 2h & 1 & & & & & & & & & & & & & \\ 4h & 1 & A_{hd} & & & A_{he} & & & \dots & & & & A_{hf} & & \\ \vdots & \vdots & & & & \vdots & & & & & & & & & \\ \vdots & 1 & & & & & & & & & & & & & \\ - & 1 & & & & & & & & & & & & & \\ \vdots & \vdots & & & & \vdots & & & & & & & & & \\ \vdots & 1 & & & & & & & & & & & & & \\ i & 1 & & & & & & & & & & & & & \\ 2i & 1 & & & & & & & & & & & & & \\ 4i & 1 & A_{id} & & & A_{ie} & & & \dots & & & & A_{if} & & \\ \vdots & \vdots & & & & \vdots & & & & & & & & & \\ \vdots & 1 & & & & & & & & & & & & & \\ - & & & & & & & & & & & & & & \end{array}$$

in which the rows are numbered in the same way as the columns. The size of a cyclotomic coset modulo n is a divisor of b . The matrix A_{gd} is a repetition of circulant matrices of row and column size minimum $\{|C_g|, |C_d|\}$ (for a set A , $|A|$ denotes the number of elements in A).

We shall transform a number of parity-check matrices of shortened cyclic codes such that the codes become square-cyclic. A set of rows numbered by the elements of a cyclotomic coset modulo n will be called a row cluster. A

set of columns numbered by the elements of a cyclotomic coset modulo n will be called a column cluster.

It is immediately clear that if we want to construct square-cyclic codes from cyclic codes by the abovementioned method, the number of elements in at least a few cyclotomic cosets modulo n have to be multiples of four. The number of elements in a coset modulo n is a divisor of b , so four has to be a divisor of b .

We now consider all binary cyclic codes of length n , $n \leq 99$, such that $b := \min\{a: n \text{ is a divisor of } 2^a - 1\}$ is a

multiple of four. The lengths, dimensions, and minimum distances of these codes are listed in [8, Appendix D] and [9]. We also consider the binary cyclic codes of length 255.

Table I shows how the parity-check matrices H_{bin} of the abovementioned cyclic codes can be transformed into square-cyclic parity-check matrices. Table I is divided into seven parts numbered by the values of b . The second column, with heading " n_{cycl} ", gives the lengths of the cyclic codes we consider. The third column (heading " $n_{\text{sq-cycl}}$ ") gives the lengths of the square-cyclic codes derived from them. For example, for $b = 8$ we consider the cyclic codes of lengths 51, 85, and 255. The cyclic codes of lengths 51, 85, and 255 are transformed into square-cyclic codes of lengths 49, 81, and 241, respectively. How this is done can be seen in the last three columns. Column four (heading "size cc") gives the sizes of the cyclotomic cosets modulo $(2^b - 1)$ that are of interest for the codes we consider. For example, the cyclotomic cosets modulo $2^8 - 1$ have sizes 1, 2, 4, and 8. Column five (heading "rows") and column six (heading "columns") provide the transformations of the parity-check matrices. If in the sixth column of Table I there is the word "del" in a row corresponding to a cyclotomic coset size i , it means that all columns in column clusters of size i should be deleted from the parity-check matrix H_{bin} . If in the fifth (sixth) column of Table I there is a permutation in a row corresponding to a cyclotomic coset size i , it means that in the parity-check matrices H_{bin} the rows (columns) in all row (column) clusters of size i should be permuted according

to that permutation. For example, consider the cyclic codes of length 85. The corresponding value of b is 8. There exist ten cyclotomic cosets modulo 85 containing eight elements, one containing four elements, none containing two elements, and one containing one element. According to Table I, from the parity-check matrices of the cyclic codes of length 85, all columns in column clusters of size four should be deleted. This reduces the length of the codes to 81. Next, in the resulting parity-check matrices the rows and columns in row and column clusters of size eight should be permuted according to the permutation $P(8,4)$. Furthermore, in all parity-check matrices the rows in the row cluster of size four should be permuted according to the permutation $P(4,2)$. The resulting parity-check matrices are square-cyclic parity-check matrices of square-cyclic codes of length 81.

For the decoding of the square-cyclic codes constructed in this subsection, the many well-known decoding algorithms for (shortened) cyclic codes can be used. Shortening a square-cyclic code in a proper way again gives a square-cyclic code. So by shortening the codes constructed above until their lengths are squares or squares minus one, we get square-cyclic codes that give completely filled square dot matrices.

CONCLUSION

We have described the transmission of information by means of two-dimensional matrices of dots on a contrasting background. In the automated manufacturing area, for example, product identification with dot codes is considered to be a good or even better alternative to product identification with bar codes.

The dot code transmission scheme is partitioned into two subschemes, the source coding scheme and the channel coding scheme. Due to the fact that dot matrices can be rotated during transmission, certain measures in the source and channel coding schemes are taken. To deal with these rotations in the source encoding, we have described two source encoding methods. These source encoding schemes, one in a very simple way and the other in the optimal way, encode the source messages (binary numbers) into channel message words (binary numbers with special structures of somewhat larger length). These channel message words are encoded into codewords by the channel encoder. Because we have to deal with the rotation of codewords and the corruption by errors simultaneously, we use a special class of error-correcting codes, called square-cyclic codes. Noise is caused by printing imperfections, dust particles, and reading failures. A large number of well-known codes, such as quasi-cyclic and (shortened) cyclic codes, were transformed into square-cyclic codes. Those codes have the advantage that a large class of elegant decoding methods for them is known. Finally, we want to remark that it is also possible to transform (shortened) cyclic unequal error protection (UEP) codes [2] into square-cyclic UEP codes and that the results in this paper can easily be extended to dot code patterns in the form of regular n -gons, where $n = 3, 5, 6, 7, \dots$.

TABLE I
CONSTRUCTION PARAMETERS FOR SQUARE-CYCLIC
CODES FROM CYCLIC CODES.

b	n_{cycl}	$n_{\text{sq-cycl}}$	size cc	rows	columns
4	15	13	1		
			2		del
			4		
8	51	49	1		
	85	81	2		del
	255	241	4	$P(4,2)$	del
			8	$P(8,4)$	$P(8,4)$
12	35	29	1		
	39	37	2		del
	45	36	3		del
	65	65	4	REV(4)	REV(4)
	91	85	6	$P(6,2)$	del
			12	$P(12,4)$	$P(12,4)$
20	41	41	1		
	55	45	2		del
	75	73	4		
			10	$P(10,2)$	del
			20	$P(20,4)$	$P(20,4)$
28	87	85	1		
			2		del
			28	$P(28,4)$	$P(28,4)$
36	95	77	1		
			4		
			18	$P(18,2)$	del
			36	$P(36,4)$	$P(36,4)$
48	97	97	1		
			48	$P(48,4)$	$P(48,4)$

ACKNOWLEDGMENT

I wish to thank C. P. M. J. Baggen, G. F. M. Beenker, J. W. Brands, C. J. L. van Driel, and L. M. G. M. Tolhuizen for their helpful suggestions during the preparation of this paper. Special thanks are due to C. P. M. J. Baggen for the fruitful cooperation that resulted in the discovery of an optimal source coding scheme.

REFERENCES

- [1] C. L. Chen, W. W. Peterson, and E. J. Weldon, "Some results on quasi-cyclic codes," *Inform. Contr.*, vol. 15, pp. 407-423, 1969.
- [2] W. J. van Gils, "Two topics on linear unequal error protection codes: Bounds on their length and cyclic code classes," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 866-876, Nov. 1983.
- [3] H. J. Helgert and R. D. Stinaff, "Minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 344-356, May 1973.
- [4] C. W. Hoffner, II and S. M. Reddy, "Circulant bases for cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 511-512, July 1970.
- [5] R. A. Jenson, "A double circulant presentation for quadratic residue codes," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 223-227, Mar. 1980.
- [6] M. Karlin, "Decoding of circulant codes," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 797-802, Nov. 1970.
- [7] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1978.
- [8] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*, 2nd ed. Cambridge, MA: M.I.T. Press, 1972.
- [9] G. Promhouse and S. E. Tavares, "The minimum distance of all binary cyclic codes of odd lengths from 69 to 99," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 438-442, July 1978.
- [10] S. E. Tavares, V. K. Bhargava, and S. G. S. Shiva, "Some rate- $p/(p+1)$ quasi-cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 133-135, Jan. 1974.
- [11] H. van Tilborg, "On quasi-cyclic codes with rate $1/m$," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 628-630, Sept. 1978.