INFORMATION AND CONTROL 9, 196–209 (1966)

Cyclic Product Codes and Their Implementation*

D. T. TANG AND R. T. CHIEN†

IBM Watson Research Center, Yorktown Heights, New York

In many communication systems, variable-redundancy coding schemes are highly desirable either due to the fact that messages of various degrees of importance are present, or due to some change in the real communication channel. A class of "cyclic product codes" presented in this paper is capable of operating in variable redundancy modes. In the simplest case, the generator polynomial in the high redundancy mode is $g(x) = g_1(x)g_2(x)$, while the generator polynomial in the low redundancy mode is only $g_1(x)$. It is shown that efficient product codes can be constructed offering different degrees of protection against independent errors, burst errors, and multiple burst errors. It is also shown that particularly simple implementation for cyclic product codes is possible. In fact, the complexity of the entire decoder can be made roughly the same as that of the decoder for the high redundancy code alone. Hence, the implementation of low-redundancy codes is accomplished with little extra cost.

I. INTRODUCTION

Practical situations in military and commercial communications systems call for the transmission of messages with various degrees of urgency and reliability requirements. To meet such needs with economy and efficiency, one turns to coding techniques which employ a variable amount of redundancy that is adjusted to suit the situation at hand. Two outstanding problems that demand solutions in this area are (1) the selection of suitable codes with variable redundancy and (2) the determination of efficient methods of informing the receiver of a mode change.

The paper is concerned with the first problem, that of selecting efficient, variable redundancy, and easily implementable codes. In particular, a class of codes, called the cyclic product codes, are inves-

* This work was sponsored by the Rome Air Development Center, Rome, New York, under contract AF 30(602)2958.

† Presently at the University of Illinois on temporary leave from IBM.

tigated in detail. A simplest cyclic product code is a cyclic code pair that operates in a variable redundancy mode. The high redundancy code is generated with the generator polynomial $g(x) = g_1(x)g_2(x)$. The low redundancy code is generated with the polynomial $g_1(x)$ only. We shall assume that the period of $g_1(x)g_2(x)$ is the same as that of $g_1(x)$. Several classes of cyclic product codes are constructed for the correction of independent errors, burst errors, and multiple bursts. They are all implementable with relatively simple hardware.

All discussions in the paper concern binary codes, the extension to nonbinary cases is of a straightforward nature in most cases. The generalization of a cyclic product code pair to a product code with three or more modes of operation is also immediate.

II. CYCLIC PRODUCT CODES

Let $g_1(x)$ be the generator polynomial of a cyclic code and n be the period of $g_1(x)$, then $g_1(x)$ divides $x^n + 1$. Furthermore, if $g_1(x)$ divides $x^k + 1$, k is a multiple of n. For most cases of interest n is odd, and therefore, $x^n + 1$ does not contain any repeated factors. As $g_1(x)g_2(x)$ has the same period n, one may conclude that $g_2(x)$ divides $x^n + 1$ and $g_1(x)$ is relatively prime to $g_2(x)$.

A. Optimum Cyclic Product Code for Correction of Burst Errors

Suppose $g(x)$ is a generator polynomial of a b-burst correcting code of length n, the period of $g(x)$, then for any two error patterns $e_1(x)$ and $e_2(x)$ that are correctible,

$$e_1(x) + x^i e_2(x) \equiv 0 \qquad (g(x)) \tag{1}$$

implies

$$e_1(x) + x^i e_2(x) \equiv 0 \qquad (x^n + 1), \tag{2}$$

where

$$\deg (e_1) \leqq b - 1, \qquad \deg (e_2) \leqq b - 1.$$

The following theorem establishes a class of cyclic product codes for burst errors.

Theorem 1. *Let $g(x) = g_1(x)g_2(x)$ be the generator polynomial of a b-burst correcting code of length n, the period of $g(x)$. Let the degree, r_2, of $g_2(x)$ be less than b. Then, $g_1(x)$ is the generator polynomial of a b_1-burst*

correcting code of length n_1, *the period of* $g_1(x)$, *where* $b_1 \geqq b - r_2$ and n_1 *divides* n.

Proof: $g_1(x)$ generates a code of length n_1. Suppose this code does not have the ability to correct all b_1-burst errors, then there exists two error patterns $e_1(x)$ and $e_2(x)$, such that

$$e_1(x) + x^i e_2(x) \equiv 0 \qquad (g_1(x)) \tag{3}$$

and

$$e_1(x) + x^i e_2(x) \not\equiv 0 \qquad (x^{n_1} + 1) \tag{4}$$

where n_1 divides n, deg $(e_1) < b_1$, and deg $(e_2) < b_1$. If we multiply both sides of (3) by $g_2(x)$,

$$g_2(x)[e_1(x) + x^i e_2(x)] \equiv 0 \qquad (g(x)). \tag{5}$$

As deg $(g_2 \cdot e_1) < b_1 + r_2 = b$, and deg $(g_2 \cdot e_2) < b_1 + r_2 = b$, by the error correcting properties of $g(x)$, (5) implies

$$g_2(x)[e_1(x) + x^i e_2(x)] \equiv 0 \qquad (x^n + 1). \tag{6}$$

As both $g_2 \cdot e_1$ and $g_2 \cdot e_2$ are bursts of maximum length b and $2b < n$, it can easily be shown that (6) implies

$$g_2 \cdot e_1 = g_2 \cdot e_2 \tag{7}$$

and that n divides i. Hence, $e_1 = e_2$ and n_1 divides i. This is a contradiction to the assumption of the existence of e_1 and e_2 that satisfy both (3) and (4). The theorem therefore follows.

With the use of Theorem 1, one may take a b-burst correcting code and obtain from it a b_1-burst correcting code by dropping a factor in the generator polynomial. The result is a pair of cyclic product codes that are suitable for variable redundancy applications.

Elspas and Short (1962) have studied the construction of burst-error codes with minimum redundancy. For burst length b and r check bits, their codes have a length of $n = 2^{r-b+1} - 1$. Tables of minimum redundancy codes have been presented for n up to 4095 and b up to four. These codes are optimum when used as cyclic product codes.

Example: The code generated by

$$g_4(x) = (1 + x)(1 + x + x^2)(1 + x^2 + x^3 + x^4 + x^6 + x^{10} + x^{12})$$

corrects all bursts of four bits or less with $n = 4095$. By dropping some factors we may obtain a burst-3 correcting code with

$$g_3(x) = (1 + x + x^2)(1 + x^2 + x^3 + x^4 + x^6 + x^{10} + 10^{12})$$

or a burst-2 correcting code with

$$g_2(x) = (1 + x)(1 + x^2 + x^3 + x^4 + x^6 + x^{10} + x^{12})$$

or a burst-1 correcting code with

$$g_1(x) = (1 + x^2 + x^3 + x^4 + x^6 + x^{10} + x^{12}).$$

The fact that these codes are all optimum as claimed follows directly from Theorem 1. We list in Table I a number of codes selected from Elspas and Short (1962), which are suitable for cyclic product codes. It should be noted that Theorem 1 could also be used to limit the amount of labor required in searching for minimum redundancy codes in general.

B. Other Cyclic Product Codes for Burst-Error Correction

The Fire codes (1959) are defined by the generator polynomial

$$g(x) = (x^c + 1)p(x) \tag{8}$$

where $p(x)$ is a polynomial of degree $\geqq b$ and period e, and $c \geqq 2b - 1$. This code is capable of correcting a single burst of maximum length b with a block length n which is L.C.M. (c, e). For most cases in practice, c is taken to be $2b - 1$ and $p(x)$ chosen to be a primitive irreducible polynomial of degree b, hence the block length is $c(2^b - 1)$ if $2b - 1$ and $2^b - 1$ are relatively prime.

It is now easy to see that the class of cyclic product codes may be constructed by taking the pair

$$g_1(x)g_2(x) = (x^c + 1)p(x) \tag{9}$$

$$g_1(x) = (x^{c_1} + 1)p(x) \tag{10}$$

where c_1 divides c. The code generated with $g_1(x)$ alone will correct a shorter burst, $b_1 = (c_1 + 1)/2$. However, as this code is longer than the normal length of a Fire code for bursts of length b_1, it is closer to optimality in redundancy.

Another class of cyclic product codes can be obtained by applying Theorem 1 to Fire codes in a more general manner. This will be illustrated

by the following generator polynomials

$$g_1(x) = (x^7 + x^6 + x^4 + x^3 + x + 1)(x^5 + x^2 + 1)$$

$$g_1(x)g_2(x) = (x^9 + 1)(x^5 + x^2 + 1).$$

Thus, $g_1(x)$ generates a code for bursts-3 correction which has a length of $9 \times 31 = 279$. $g_1(x)g_2(x)$ generates a burst-5 Fire code of length 279. It should be noted that $(x^7 + 1)(x^5 + x^2 + 1)$ generates a burst-4 Fire code of length 217. Comparing with the results obtained by applying Theorem 1 to codes of Table I, it is clear that the most advantageous situation of applying Theorem 1 is when the higher redundancy code of the pair is close to optimum.

C. Product Codes for Independent-Error and Multiple-Burst Correction

The BCH (Bose-Chaudhuri-Hocquenghem) codes are most conveniently described in terms of the finite field $GF(2^m)$. (See (Bose and Chaudhuri, 1960) and (Hocquenghem, 1959).) A t-error correcting BCH code is generated by the polynomial $g(x)$ such that

$$g(\alpha^i) = 0 \qquad i = 1, 2, \cdots, 2t \tag{11}$$

where α is an element of $GF(2^m)$ and the length of the codes is the order of α. When α is a primitive element, the code length is $2^m - 1$.

Let $M_i(x)$ be the minimal polynomial of α^i, then the generator of a t-error correcting BCH code can be written as

$$g(x) = \text{LCM } (M_1(x), \cdots, M_{2t}(x)). \tag{12}$$

The generator of a t_1-error correcting BCH code (where $t_1 < t$) can be written as

$$g_1(x) = \text{LCM } (M_1(x), M_2(x), \cdots, M_{2t_1}(x)). \tag{13}$$

It follows that $g_1(x) \mid g(x)$, and BCH codes can be immediately written in the form of product codes if variable redundancy is required.

The Reed-Solomon codes can be used for multiple-burst correction. Being a special type of BCH codes, they can be also immediately written in the form of product codes.

IMPLEMENTATION OF CYCLIC PRODUCT CODES

A. General Considerations

Let $g_1(x)$ be the generator polynomial of a (n, k_1) code C_1 and let $g(x) = g_1(x)g_2(x)$ be the generator polynomial of a (n, k) code C where the degree of $g_1(x)$ is $r_1 = n - k_1$, and the degree of $g(x)$ is $r = n - k$. To illustrate the implementation of the product codes, a specific example will be used throughout this paper for encoding and burst-error correction. The cyclic product code pair operate in the following two modes:

1. Mode 1: C_1 is a (15, 11) code generated by

$$g_1(x) = x^4 + x + 1.$$

This code corrects single errors.

2. Mode 2: C_2 is a (15, 9) code generated by

$$\begin{aligned} g(x) &= g_1(x)g_2(x) \\ &= (x^4 + x + 1)(x^2 + x + 1). \end{aligned}$$

This code corrects burst errors up to length = 3.

The above product code pair has the property that both C_1 and C are optimum codes. Other product codes with the same optimality property can be obtained from Table I.

TABLE I
CYCLIC PRODUCT CODES FOR BURST ERRORS

n	r	g(x) (Listing Powers with Units Coefficient for Each Factor)
1023	13	(01) (012) (0235, 10)
		(01) (012) (012467, 10)
		(01) (012) (023458, 10)
		(01) (012) (012369, 10)
		(01) (012) (013469, 10)
4095	15	(01) (012) (0347, 12)
		(01) (012) (012459, 12)
		(01) (012) (034589, 12)
		(01) (012) (01234589, 12)
		(01) (012) (02346, 10, 12)
		(01) (012) (02348, 10, 12)
		(01) (012) (01478, 10, 12)
		(01) (012) (0234678, 10, 12)
		(01) (012) (01269, 10, 12)
		(01) (012) (01246, 11, 12)
		(01) (012) (0125789, 11, 12)
		(01) (012) (0124, 10, 11, 12)
		(01) (012) (012347, 10, 11, 12)

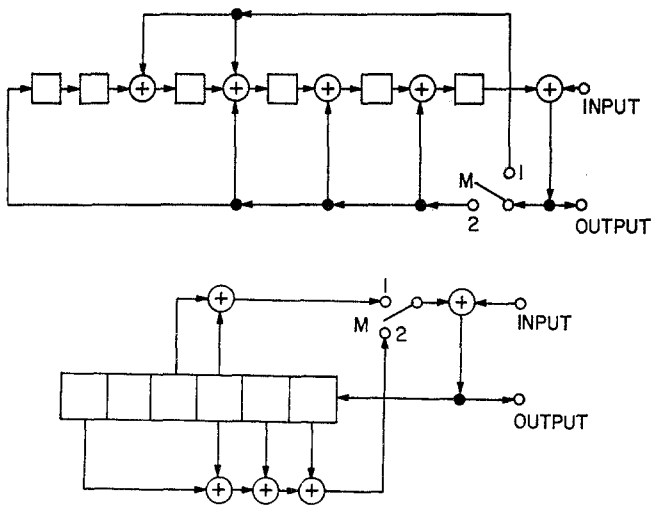

FIG. 1. Two division circuits each with separate feedback connections for multimode operations.

To implement a cyclic product code capable of operating in two or more modes, we should take advantage of the properties of the product code so that some sharing of the circuitry when operated in different modes is possible. An immediate approach is to share the shift-registers in different modes and switch the interconnection from one configuration to another to obtain a change in its function required by the different modes of operation.

Figure 1 shows two division circuits designed according to the above philosophy. Switch M is at position "1" for Mode 1 and at position "2" for Mode 2. Such division circuits can be parts of the encoding or decoding circuitry.

It can be seen from the circuits of Fig. 1 that, because of an extra mode of operation, some new mod-2 adders must be installed, and some mod-2 adders originally with two inputs now become adders with three inputs. Generally speaking, the complication thus introduced is roughly proportional to the degree of $g_1(x)$ or $g(x)$.

B. Basic Register Blocks

A different approach which is both practically more appealing and theoretically more interesting is to combine the basic register blocks

corresponding to $g_1(x)$ and $g_2(x)$ in obtaining the basic block corresponding to $g(x) = g_1(x)g_2(x)$. The complication introduced due to multiple modes of operation can be kept at its minimum and is independent of the sizes of $g_1(x)$ and $g_2(x)$.

A basic block corresponding to $g(x)$ here refers to a shift-register circuit whose input and output sequence is related by the transfer function

$$\frac{t}{s} = \frac{g(x)}{x^r} + 1 = g^*(D) + 1,$$

where $x^{-1} = D$ is considered as a delay operator and $g^*(D)$ is the reciprocal polynomial of $g(D)$. Such a register block can be connected either as a multiplication circuit or a division circuit.

Figure 2 shows a multiplication circuit where

$$\begin{aligned} \text{input} &= s, \\ \text{output} &= \text{input} + t \\ &= \text{input}\,(1 + g^*(D) + 1) \\ &= \text{input} \cdot g^*(D). \end{aligned} \tag{14}$$

Figure 3 shows a division circuit where

$$\begin{aligned} \text{input} &= s + t = s \cdot g^*(D) \\ \text{output} &= s = \text{input}/g^*(D). \end{aligned} \tag{15}$$

It is interesting to see how such basic blocks with $t_1/s_1 = g_1^*(D) + 1$ and $t_2/s_2 = g_2^*(D) + 1$ can be connected to form a new block with

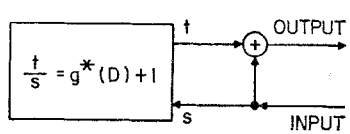

FIG. 2. A basic register block connected as a multiplication circuit

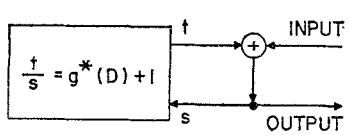

FIG. 3. A basic register block connected as a division circuit

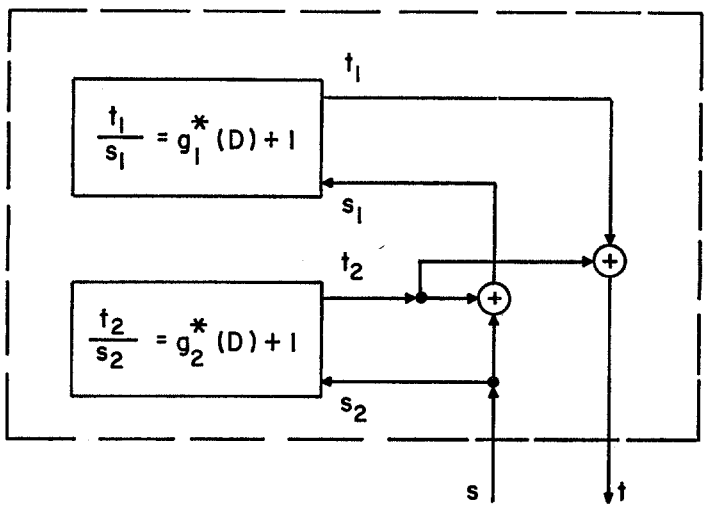

FIG. 4. Combination of two basic blocks into one where the defining polynomials multiply.

$t/s = g^*(D) + 1$ where $g^*(D) = g_1^*(D)g_2^*(D)$. This is shown in Figure 4 where we have

$$\begin{aligned} t &= t_1 + t_2 = s_1(g_1^*(D) + 1) + s_2(g_2^*(D) + 1) \\ &= s(g_2^*(D) + 1) + s \cdot g_2^*(D)(g_1^*(D) + 1) \\ &= s(g_1^*(D)g_2^*(D) + 1). \end{aligned} \tag{16}$$

It will be clear in the next section that such a circuit is very useful in the implementation of a product code.

C. Encoding Circuits

Figure 5 shows a general encoding circuit for any cyclic product code-pair. The switches are controlled according to the following:

Mode 1: Generator polynomial $= g_1(x)$

$M = 1$ for all n bits

$K = 0$ for the first k_1 bits

$\quad\; 1$ for the remaining r_1 bits.

Mode 3: Generator polynomial $= g(x) = g_1(x)g_2(x)$

$M = 2$ for all n bits

$K = 0$ for the first k bits

$\quad\; 1$ for the remaining r bits.

The code vectors obtained from this encoder are separable. That is, a k-bit message $M(x)$ is coded as

$$C(x) = x^r M(x) + r(x)$$

where

$$r(x) \equiv x^r M(x) \qquad (g(x))$$

has degree no more than $r - 1$.

The basic block is connected as a division circuit when the k bits of $x^r M(x)$ enter the circuit. The r-bit residue $r(x)$ is obtained at the output by switching K to the "1" position.

Figure 6 shows two possible encoding circuits for the specific cyclic product code pair described earlier.

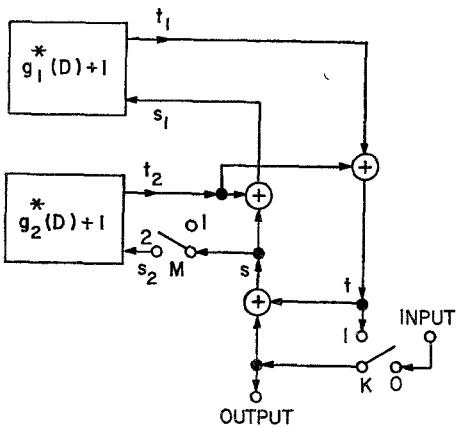

FIG. 5. A general product code encoder

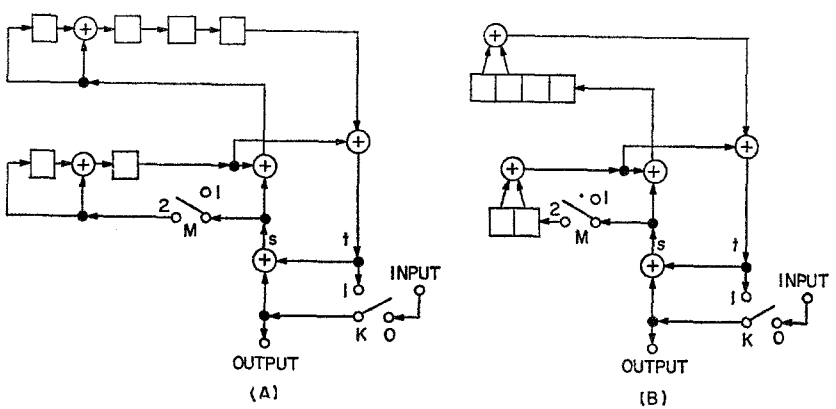

FIG. 6. Two encoders for the product code pair with generators $g_1(x) = x^4 + x + 1$ and $g(x) = (x^2 + x + 1) \cdot g_1(x)$.

D. Decoding Circuits for Burst-Error-Correcting

An ordinary decoding circuit contains an n-stage buffer storage and a division circuit corresponding to $g(x)$. The vector with shift-register contents as components forms an error syndrome, which does not necessarily correspond to the residue of the error polynomial modulo

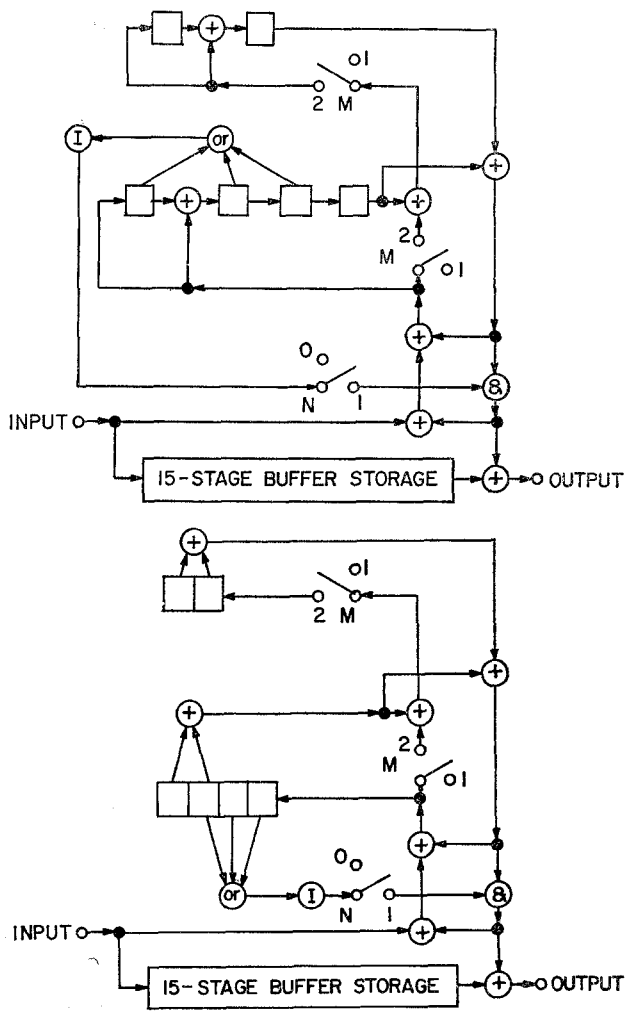

FIG. 7. Two decoders for the product code pair with generators $g_1(x) = x^4 + x + 1$ and $g(x) = (x^2 + x + 1) \cdot g_1(x)$. Switch $N = 0$ for the first n bits and $N = 1$ for the next n bits.

$g(x)$. This basic approach is the same in that of Peterson (1961) and Meggitt (1961).

In order for the circuit to be capable of correcting burst errors in different modes, the division circuit must contain individual basic blocks corresponding to $g_1(x)$ and $g_2(x)$ so that one of them can be disconnected in the low redundancy mode. The syndromes should go through an error cycle and should be recognized when the first bit of a correctible burst error is ready to leave the n-stage buffer storage.

Figure 7 shows two decoding circuits for the product code with $g(x) = (x^4 + x + 1)(x^2 + x + 1)$. Note that since the difference in the maximum correctible burst lengths is the same as the difference in the degrees of generator polynomials in two modes, the same zero-detecting circuit can be used in both modes. This applies to all burst-error correcting product codes which are obtained by the use of Theorem 1, such as those shown in Table I. In general, the zero-detecting circuit may require a different set of input connections in different modes.

In terms of matrix notations, the shift-register contents form a syndrome in the form of the row vector determined by multiplying the received vector (a code vector C plus an error vector E) and the transposed parity-check matrix. That is,

$$S = (C + E)H^T = EH^T. \tag{17}$$

The parity-check matrix corresponding to the decoding circuit of Fig. 7(a) is

$$H_a = \left|\begin{array}{cccccccccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{array}\right|, \tag{18}$$

and the parity-check matrix corresponding to the decoding circuit of Fig. 7(b) is

$$H_b = \left|\begin{array}{cccccccccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{array}\right|. \tag{19}$$

E. Decoding Cirucits for Independent-Error Correction

The decoding circuits described previously can be used for independent-error correction if a combinational circuit is used as syndrome recognizer. However, this may be limited to cases where the minimum distance of the code is relatively small. For codes with larger distance the size of the combinational syndrome recognizer may become prohibitively large.

For Bose-Chaudhuri-Hocquenghem codes, a cyclic decoder can be used. This improved decoding procedure takes advantage of the cyclic property of the code, and the errors are corrected sequentially at the output.

Upon receiving a polynomial $r(x)$, syndromes $S_1 = r(\alpha)$, $S_3 = r(\alpha^3)$, $\cdots$, $S_{2t-1} = r(\alpha^{2t-1})$ are first used to obtain the elementary symmetric functions $\sigma_1, \sigma_2, \cdots, \sigma_t$. The nonzero roots of the polynomial

$$\Sigma(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_{t-1}x + \sigma_t \tag{20}$$

represent the errors.

Instead of the usual exhaustive approach in finding the roots, the following procedure is carried out: The polynomial is sequentially transformed such that at the ith step, we have

$$\Sigma^{(i)}(x) = x^t + \sigma_1\alpha^i x^{t-1} + \cdots + \alpha_{t-1}\alpha^{(t-1)i}x + \sigma_t\alpha^{ti}. \tag{21}$$

Therefore, if β is a root of $\Sigma(x)$, $\alpha^i\beta$ is a root of $\Sigma^{(i)}(x)$. Consider an error at the jth position, thus $\beta = \alpha^{j-1}$. After $i = n - j + 1$ transformations, $\alpha^{i+j-1} = \alpha^n = 1$ becomes a root of $\Sigma^{n-j+1}(x)$. By checking whether "1" is a root of $\Sigma^{(i)}(x)$ at each step, the bit in error can be corrected sequentially at the output.

It is clear that if the decoder is designed for the code with generator polynomial $g(x) = g_1(x)g_2(x)$, the low redundancy mode of operation with generator polynomial $g_1(x)$ requires essentially no change of the circuit. For if $g_1(x)$ generates a t_1-error correcting code, $\sigma_i = 0$ for $i > t_1$.

For detailed discussions of this cyclic decoding approach, see Chien (1964).

CONCLUSION

This paper presents original results on the theory and implementation of cyclic product codes. Various classes of codes are shown to exist for the correction of burst errors, independent errors, and multiple bursts. Simple decoding circuits with little extra hardware for mode changes

are obtained and shown to be very suitable for burst-error correction. An algebraic-cyclic procedure is suggested for correction of independent errors and multiple bursts.

The concept of basic blocks can be used in more general cases where algebraic operations are required with respect to a set of polynomials over a finite field and their products. Using this approach in the implementation of cyclic codes, it becomes evident that Peterson's decoder (1961) and Meggitt's decoder (1961) are equivalent.

RECEIVED: December 17, 1964

REFERENCES

BOSE, R. C. AND RAY-CHAUDHURI, D. K. (1960), On a class of error-correcting binary group codes. *Inform. Control* **3,** 68–79.

CHIEN, R. T. (1964), A cyclic decoder for Bose-Chaudhuri-Hocquenghem codes. *IEEE Trans.* **IT-10,** 357–363.

ELSPAS, B. AND SHORT, R. A. (1962), A note on optimum burst-error correcting codes. *IRE Trans.* **IT-8,** 39–42.

FIRE, P. (1959), A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors, Rpt. RSL-E-2 Sylvania Electric Products, Inc., Mt. View, Calif.

HOCQUENGHEM, A. (1959), Codes correcteurs derreurs. *Chiffres* **2,** 147–156.

MEGGITT, J. E. (1961), Error-correcting codes and their implementation for data transmission systems. *IRE Trans.* **IT-7,** 234–244.

PETERSON, W. W. (1961), "Error Correcting Codes." MIT Technology Press, Cambridge, Mass., and Wiley, New York.

TANG, D. T. (1965), Dual codes as variable redundancy codes. *IEEE Intern. Conv. Record,* **13,** part 7, 220–226.