Tutorially presented are theoretical and practical concepts that underlie error-control coding for data computing, storage, and transmission systems.

Emphasis is on cyclic codes, the most deeply studied and widely used of the many available codes. Operations of typical binary shift registers illustrate the encoding and decoding processes.

Strategic considerations for applying coding to computer-communication systems are discussed. Actual applications further exemplify the basis for code selection.

Coding for error control

by D. T. Tang and R. T. Chien

Error rates associated with current digital systems are usually extremely low in spite of the increasingly high speed of processing and transmission. Recent developments in error-correcting codes have contributed toward achieving the high reliability required by today's digital systems, and it is evident that the use of coding methods for error control has become an integral part in the design of modern computers and communications systems.

This paper is intended as an introduction to the theory and applications of error-control codes, involving both error detection and error correction. The first two parts of this paper are concerned with fundamental definitions in coding and digital data channels. In the following sections, concepts of errors, code structures for error control, and some general properties of shift-register circuits are introduced. Methods of implementing encoders and decoders as well as the functional classes of error-control codes are also described. The last two sections deal with coding strategy and applications of error-control schemes in existing data-transmission and storage systems.

Basic definitions

Coding is the representation of information (signals, numbers, messages, etc.) by code symbols or sequences of code symbols (often called *code words*). The set of code words and their mapping, which determines the set, characterize a *code*. Information is said to be placed into code form by *encoding* and extracted from code

form by *decoding*. Certain codes may have a larger average code length than others. Such codes are said to contain "redundancy," which can be used to advantage for error control.

redundancy

The development of redundancy schemes, in the form of coding suitable for modern digital systems, took place after the inspiration of Shannon's basic theorem in 1948.[1] Among other things, Shannon showed that even in a noisy channel, errors in data transmission can be reduced to any desired level if a certain minimum percentage of redundancy is maintained by means of proper encoding and decoding of the data. Although Shannon's theorem does not suggest any procedure for constructing such codes, the work of Golay,[2] Hamming,[3] Slepian,[4] Prange,[5] and many others have contributed a whole body of new knowledge—coding theory.[6,7] Mathematical structures have been used to construct codes with various types of error control, and these structures provide means of analysis as well as sophisticated encoding and decoding procedures.

source codes

Since encoding is no more than the digital representation of information, a code does not necessarily have error-control capability. *Source codes*, for example, are designed to represent information with sequences of code symbols in the most efficient way, i.e., using the smallest possible number of code symbols on the average.[8] Therefore, source codes usually contain negligible redundancy and should not be confused with the error-control *channel codes* used under noisy situations. Typically, a source code is first used to represent the output of an information source. Then an error-control coding scheme is implemented to cope with the noisy condition in which the resulting code sequence is to be transmitted or stored.

block codes

An important class of error-control codes is that of *block codes*. A block code consists of "code words," which are sequences of code symbols of fixed length n, often referred to as *n-tuples* or *n-vectors*. In most cases, the information sequence to be encoded contains k digits, which are encoded as an n-tuple code word. The *redundancy* (normalized) is $(n - k)/n$, or r/n, where $r = n - k$. Such a block code is often denoted as an (n, k) code.

binary codes

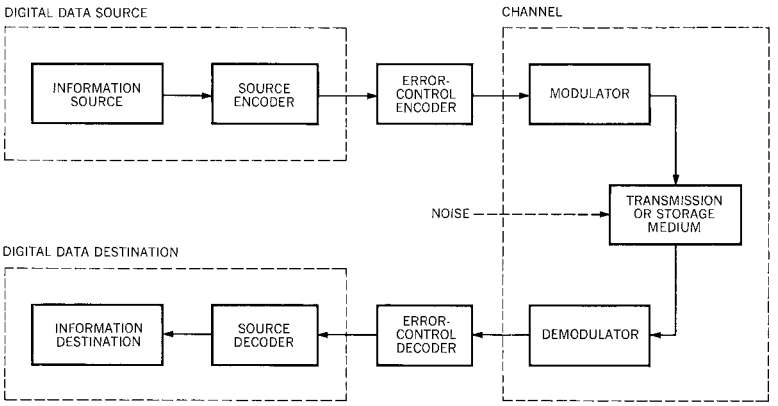
Because of their applications in digital data transmission, storage, and processing systems, *binary codes* are by far the most important codes used. The simplicity of the binary representation of information lends itself conveniently to mathematical treatments, and as a result, we now know much more about binary codes than others. We deal almost exclusively with binary codes in this paper. Although familiarity with basic matrix operations is assumed, other concepts of modern algebra are described as they are used.

Errors in digital data channels

transmission and storage

The transmission and storage of digital data have much in common. They both accomplish the transfer of digital data from

Figure 1 Generalized data transmission or storage system



a source to a destination. For transmission, the source and destination are mainly separated in space, and for storage, they are mainly separated in time. Transmitting lunar photographs from a distant satellite back to earth, transferring data from one computer component to another only inches away, and writing and reading data on magnetic tape can all be described by the same general process consisting of the steps shown in the block diagram in Figure 1.

source encoding

The purpose of the source encoder is to produce the best digital representation of data originating at the information source. Source encoding often requires redundancy removal. When the information at the source is in analog form, the quantization of analog signals must also be performed. This part of the system is normally independent of the channel characteristics or noise statistics. After the error-control encoder (or channel encoder) adds the appropriate amount of redundancy, the modulator then transforms the digital code symbols into physical signals, such as voltage waveforms, ready for transmission or storage via the noisy channel. On the other end of the channel, the exact reversal of the above procedure is performed in complementary steps.

modulation and demodulation

Both the modulator and demodulator must be considered as parts of the digital data channel, since an error-control code can only protect against errors corresponding to the wrong identifications of digital symbols. Modulation and demodulation techniques designed to produce the fewest possible errors are usually analog in nature.

Although the analysis of modulation-demodulation techniques are basically communications problems, which are not discussed in this paper, several related facts are mentioned here. In order to demodulate properly, the demodulation must be able to establish the synchronization of received signals so that the detection of a digital symbol is based on the proper portion of

the detected waveform. Any small change in detection threshold level or sampling delay would, strictly speaking, result in a different digital data channel. However, we may assume that the system parameters do not change greatly during a typical operating period. All temporary effects of changes can be regarded as noise and included in the error statistics. In the final analysis, the error statistics of the demodulated signals characterize the digital data channel.

Error sources

error statistics

The distribution of error statistics depends heavily on the following sources of errors:

- *Modulator and demodulator circuit noise* is predominantly thermal in origin and results mostly in uncorrelated errors.
- *Physical disturbances in terminal components* include changing air gap and changing surface velocity in magnetic surface recording. Errors caused by physical disturbances are highly correlated and tend to cluster in bursts.
- *Physical disturbances in transmission or storage media* are usually sources of burst errors.

The first two error sources are self-explanatory, but there are many causes of transmission and storage disturbances. The most common cause of errors in telephone lines, for example, is switching-impulse noise. The duration of such impulses is in the order of milliseconds, resulting in short error bursts. For microwave and radio links, typical fading or dropouts may last from milliseconds to seconds or even to minutes. The resulting bursts thus tend to be much longer than those caused by switching impulses, and they are often difficult to control by codes, unless extremely long blocks are used.

storage

In storage media, such as magnetic tapes, surface defects include loss of oxide, scratches, dirt particles, and wrinkles. The effect of such disturbances can accumulate until a tape is no longer usable. Many of these defects are also common to magnetic disks or drums. These defects typically assume sizes up to several mils, resulting again in short bursts of errors. Core storage arrays usually remain reliable after they are tested, although breakage or other accidental defects may later cause independent errors. Generally speaking, burst errors are much more likely to be caused by physical disturbances. Background noises do exist, but become significant only in special cases such as space communications.

channel models

A digital data channel is characterized by the error statistics associated with the input and output alphabets of the channel. Therefore, it is often desirable to represent the error statistics in terms of a certain simple mathematical model. List all the conditional probabilities of receiving the symbols in the output alphabet, for all possible transmitted symbols in the input alphabet.

If these probabilities are independent of the locations of symbols, then we have a model completely characterizing a digital *memoryless channel*.[1] In such a channel, probabilities of erroneous symbols received are independent of the neighboring transmitted or received sequences of symbols.

When most errors tend to cluster, the channel is no longer a memoryless one. A memoryless model can at best be considered as an approximation of the real channel. If the clustering of errors is independent of the transmitted symbols, a Markov model is the appropriate one. Such a model consists of states identified by one or more preceding symbols from the "error sequence" (the difference between the transmitted and received sequences).

When error bursts are not necessarily solid, or when bursts themselves tend to cluster, such as in a fading channel, one must either go to Markov models of higher orders or use a different model, such as one in which the probability distribution of the number of digits between errors is described by a certain simple function.[9]

Mathematical structures in coding

Some basic concepts of code structure and requirements of error-control are now discussed. We choose a subset from the set of all n-tuples to form a code set. This code set has some error-control capability, since the receiver can detect the occurrence of an error when the received n-tuple is not in the chosen code set. For errors to be corrected, we must also have a decoding procedure that determines the supposedly transmitted code word when an unacceptable n-tuple is received. This can be done by a table lookup procedure at the receiving end.

A mathematical treatment of the encoding-decoding process is needed to (1) select a set of n-tuple code words with a specified error-control capability, and (2) build a structure so that the code set can be decoded systematically without table lookup (which is clearly impractical for large code sets). Such structures yield properties of code sets that facilitate analysis and simplification of the encoding-decoding procedure.

linear separable codes

It is desirable to divide a code word into an information part and a redundant checking part. A code with this feature is a *separable code*. In the case of the *linear separable codes*, each of the check symbols is a certain linear combination of the information symbols. For example, a binary information 4-tuple, (i_1, i_2, i_3, i_4) can be coded as a binary 7-tuple with three binary check symbols (c_1, c_2, c_3). Here, a 7-tuple code word may take the general form $(i_1, i_2, i_3, i_4, c_1, c_2, c_3)$, with

$$c_1 = i_1 + i_2 + i_3, \quad c_2 = i_2 + i_3 + i_4, \quad c_3 = i_1 + i_2 + i_4$$

where additions are binary operations.[10] The relationship can

be conveniently illustrated by an example expressed in matrix form as shown in Equation 1. A code word vector results when a binary information 4-tuple operates on the code generator matrix. The configuration of the generator matrix is obtained from coefficients of the corresponding simultaneous equations, which depend upon the nature of the code selected.

$$[i_1 i_2 i_3 i_4]\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [v_1, v_2, \cdots, v_7] \tag{1}$$

An equivalent way to characterize a linear code is to specify a set of simultaneous *parity equations* that must be satisfied by the code symbols. Using the example in Equation 1, the following three equations must be satisfied by all the code words that take the form $(v_1, v_2, \cdots, v_7)$:

$$v_1 + v_2 + v_3 + v_5 = 0$$

$$v_2 + v_3 + v_4 + v_6 = 0$$

$$v_1 + v_2 + v_4 + v_7 = 0$$

Again, this set of linear simultaneous equations can be conveniently written in matrix form as follows:

$$[v_1, v_2, \cdots, v_7]\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{0}$$

In general, a k-tuple information part can be coded into an n-tuple code word according to the equation

$$\mathbf{iG} = \mathbf{v}$$

where the matrix $\mathbf{i}$ is 1 by k, $\mathbf{G}$ is k by n, and $\mathbf{v}$ is 1 by n. The matrix $\mathbf{G}$ is called the *generator matrix* of the code. Alternatively, the parity equations may be written in the form

$$\mathbf{vH}^T = 0$$

where $\mathbf{v}$ is 1 by n, $\mathbf{H}^T$ is n by $r(=n-k)$, and $\mathbf{0}$ is 1 by r. $\mathbf{H}^T$ is the transpose of $\mathbf{H}$, which is called the *parity-check matrix* of the code. For some basic structural features of linear codes, see Appendix A.

polynomial cyclic codes

One way to represent an n-tuple is to consider the symbols of the n-tuple to be coefficients of a polynomial of degree $n-1$

or less. Specifically, an n-tuple $(a_1, a_2, \cdots, a_n)$ gives rise to a polynomial representation $a_1x^{n-1} + a_2x^{n-2} + \cdots + a_n$.

When the addition and multiplication are both defined on the symbols used as the coefficients of polynomials, the addition and multiplication of polynomials can be carried out in the ordinary manner. The addition of two polynomials of degree $n - 1$ or less does not differ from the addition of corresponding n-tuples. The product of two polynomials $a(x)$ and $b(x)$ of degree $n - 1$ or less can be defined as another polynomial $c(x)$, also of degree $n - 1$ or less, which is the residue of the usual product when divided by $x^n + 1$. This operation is written in the form

$$a(x)b(x) \equiv c(x) \quad \text{modulo} \quad (x^n + 1)$$

We use the symbol $\equiv$ in this paper to mean "is congruent to."

Any binary polynomial $g(x)$ must divide $x^n + 1$ for some positive integer n. The set of all polynomials that are distinct multiples of $g(x)$ modulo $x^n + 1$ constitutes a *cyclic (polynomial) code* in the sense that if $a(x)$ and $b(x)$ are code polynomials then $a(x) + b(x)$ is also a code polynomial.

Furthermore, any cyclic (end-around) shift of a code word is also a code word, since a cyclic shift of a code word is equivalent to the multiplication of x^i by the code polynomial modulo $(x^n + 1)$, resulting in another polynomial in the code set. The polynomial $g(x)$ is called the *generator polynomial* of the code, and such a polynomial uniquely characterizes a cyclic code. The polynomial $h(x) = (x^n + 1)/g(x)$ is called the *recursive polynomial* of the same code.

If the degree of $g(x)$ is r, then there are 2^k distinct multiples of $g(x)$ of degree $n - 1$ or less, where $k = n - r$ is also the degree of $h(x)$. Some basic structural features of polynomial codes are included in Appendix B.

General requirements for encoding and decoding

Thus far we have discussed the generation of linear separable and cyclic codes and have appended some basic structural features of these codes. Now, we briefly discuss certain general requirements of linear and cyclic codes. It was stated that the encoding procedure consists of essentially the selection of an n-tuple code word, given any number of information symbols. At the same time, the decoding procedure essentially consists of determining what these information symbols should be, when receiving any n-tuple. Without any code structure, decoding can only be done by table lookup.

error syndromes

When linear codes are used, however, the correctable error patterns become separable from the code words and can thus be identified independently of the code words transmitted. To show this, let $\mathbf{v}$ be a code word, and $\mathbf{H}$ be a parity-check matrix. If the error is $\mathbf{e}$ (an n-tuple), then the received n-tuple

is $\mathbf{v}' = \mathbf{v} + \mathbf{e}$. If we calculate the *syndrome*, defined as

$$\mathbf{S} = \mathbf{v}'\mathbf{H}^T = (\mathbf{v} + \mathbf{e})\mathbf{H}^T = \mathbf{v}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

we see that it is an r-tuple independent of the code word $\mathbf{v}$. The syndrome $\mathbf{S}$ contains all the information regarding the error that has been added to the code word during the transmission. For a deterministic correction scheme, each syndrome must be identified with a unique error n-tuple. Since the zero syndrome always means "no error," a nonzero syndrome is necessary for the detection of any error n-tuple.

The following observation can now be made for a binary code. Since the syndromes are r-tuples, there are 2^r distinct forms. Clearly, we cannot expect the code to correct more than 2^r distinct errors (including no error). Furthermore, if two errors result in the same syndrome, at most one of them can be corrected. A condition for a set of errors to be correctable is for any two errors $\mathbf{e}_1$ and $\mathbf{e}_2$ from the set to satisfy

$$\mathbf{e}_1\mathbf{H}^T - \mathbf{e}_2\mathbf{H}^T = (\mathbf{e}_1 - \mathbf{e}_2)\mathbf{H}^T \neq 0$$

In terms of polynomials, the condition becomes

$$e_1(x) - e_2(x) \not\equiv 0 \quad \text{modulo} \quad g(x)$$

where $e_1(x)$ and $e_2(x)$ are any two correctable error polynomials. In particular, if an error takes the same form as a code word, then it cannot be distinguished from zero error.

For a cyclic code, the syndrome of an error $e(x)$ usually means the residue of $e(x)$ modulo $g(x)$, the generator polynomial. However, depending on the specific decoding procedure chosen, the syndrome may take other forms such as the residue of $x^r e(x)$ modulo $g(x)$.

conditional maximum likelihood decoding

It should be noted that the performance of a decoding scheme depends on the characteristics of the information source and the channel, as well as that of the code used. Generally speaking, if we want to minimize the decoding error with a specific code, the *conditional maximum likelihood* decision scheme should be used. With this scheme, a code word $\mathbf{v}_i$ is selected as the decoded message upon receiving $\mathbf{v}'$, such that the conditional probability $P(\mathbf{v}_i \mid \mathbf{v}')$ is maximum for all $\mathbf{v}_i$. In evaluating these conditional probabilities, accurate source statistics must be used. This introduces an immediate difficulty since such detailed source statistics are usually not available. Furthermore, the calculation of $P(\mathbf{v}_i \mid \mathbf{v}')$ for all $\mathbf{v}_i$ is impractical for most cases.

maximum likelihood decoding

An alternative method of decoding is to use the *maximum likelihood* decision rule, which selects a code word $\mathbf{v}_i$, upon receiving $\mathbf{v}'$, such that the conditional probability $P(\mathbf{v}' \mid \mathbf{v}_i)$ is maximum for all possible code words. The calculation of conditional probabilities $P(\mathbf{v}' \mid \mathbf{v}_i)$ no longer depends on the source statistics. This rule is equivalent to the conditional maximum likelihood decision rule when all source symbols are equally likely. For linear codes, this decoding method requires that, among all error n-tuples resulting in the syndrome calculated, the

one with the highest probability of occurrence should be taken as the error that occurred. Note that the error can be identified independently of the code transmitted.

minimum distance decoding

We may consider all possible n-tuples to be points in an n-dimensional space, and define a *distance function* $D(\mathbf{x}, \mathbf{y})$ between two points (n-tuples) $\mathbf{x}$ and $\mathbf{y}$ to be the number of places where the two n-tuples differ. (In binary cases, this is usually called the "Hamming distance.") We may then use the following *minimum distance* decoding scheme: upon receiving $\mathbf{v}'$, select a code word $\mathbf{v}_i$ that minimizes $D(\mathbf{v}', \mathbf{v}_i)$ among all code words. Minimum distance decoding is equivalent to that obtained by using the maximum likelihood decision rule, provided that the errors are independent. This geometrical interpretation of the coding and decoding procedure is often very useful.

The distance function previously defined has the following "triangular" property: for any three points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, then $D(\mathbf{x}, \mathbf{y}) + D(\mathbf{y}, \mathbf{z}) \geq D(\mathbf{x}, \mathbf{z})$. From this property, one can show that, if for a given code the minimum distance between any pair of code words is D_m, then this code is capable of correcting t errors and simultaneously detecting d errors ($d > t$) as long as $d + t < D_m$. On the other hand, if t-error correction is desired, then $D_m \geq 2t + 1$.

Linear switching circuits and shift registers

A properly designed electronic linear switching circuit is capable of storing and manipulating a given digital message sequence algebraically, and hence can be used for encoding or decoding purposes. The basic elements of a linear switching circuit are: delay units, adders, and multipliers. In binary cases, no multipliers are necessary because the multiplication of 1 implies a direct connection, and the multiplication of 0 implies no connection. A switching circuit with modulo 2 adders and delay units (or registers) is referred to as a shift-register circuit.

The relationships between input and output sequences of a linear switching circuit depends upon the connections among the basic elements previously described. With respect to a pair of input/output points, the behavior of such a circuit can be described by its unit response. This response is the output sequence caused by an input sequence wherein the first symbol is 1, and all the following symbols are 0. (The initial contents of all delay units must be 0.)

polynomials in delay operator D

We may denote a sequence $\mathbf{s} = (s_1, s_2, \cdots)$ in terms of its transform,[11] which is a power series in the delay operator D

$$s(D) = s_1 + s_2 D + s_3 D^2 + \cdots$$

For ease of algebraic manipulation, a code polynomial $A(x)$ representing an n-tuple is usually transmitted with the higher-order-first convention. Writing

$$\begin{aligned} A(x) &= a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n \\ &= x^{n-1}(a_1 + a_2 x^{-1} + \cdots + a_{n-1} x^{2-n} + a_n x^{1-n}) \end{aligned}$$

we see that, in a sense, x^{-1} becomes equivalent to the delay operator D.

Consider the binary shift-register circuits as shown in Figures 2A and 2B. If $s(D) = 1$, one can see from the paths directed from the input to the output that, in both cases,

$$t(D) \mid_{s(D)=1} = T(D) = 1 + D^2 + D^3$$

If we let $D = x^{-1}$ in $T(D)$, we have

$$T(1/x) = 1 + x^{-2} + x^{-3}$$

or

$$x^3 T(1/x) = x^3 + x + 1 = T^*(x)$$

Here, $T^*(x)$ denotes the reciprocal of $T(x)$ and is obtained by reversing the order of coefficients in $T(x)$. Thus, in terms of polynomials in x, the circuits in Figures 2A and 2B are both circuits for multiplying the input polynomial by the polynomial $x^3 + x + 1$. The coefficient of the highest degree term in the product is obtained at the output without any delay.

Figures 3A and 3B show two division circuits whose functions can be easily analyzed by first observing that the following relationships hold in both circuits

$$x(D) = (D^3 + D^2)t(D)$$

and

$$x(D) + s(D) = t(D)$$

Combining the above two equations, we have

$$s(D) = (D^3 + D^2 + 1)t(D)$$

or

$$T(D) = \frac{t(D)}{s(D)} = \frac{1}{D^3 + D^2 + 1}$$

Similarly, for both circuits in Figures 4A and 4B,

$$\frac{t(D)}{s(D)} = \frac{D^3}{D^3 + D^2 + 1}$$

In terms of polynomial representations, all circuits in Figures 3A, 3B, 4A, and 4B are circuits for dividing the input polynomial by the polynomial $x^3 + x + 1$. The first bit of the quotient (coefficient of the highest degree term) is obtained at the output either without any delay or after three units of delay depending upon whether 1 or D^3 appears in the numerator of the transfer function.

Figures 5A and 5B show circuits for respectively multiplying and dividing the input polynomial by an arbitrary polynomial of degree m,

$$A(x) = x^m + a_{m-1}x^{m-1} + \cdots + a_1 x_1 + a_0$$

Figure 2 Multiplication circuits

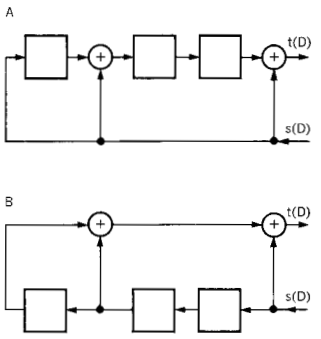


Figure 3 Division circuits

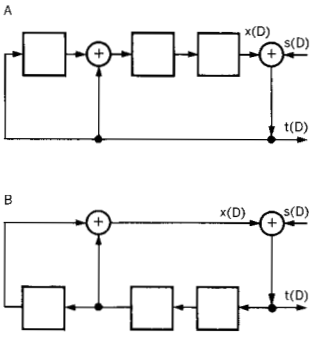


Figure 4 Division circuits that produce residues

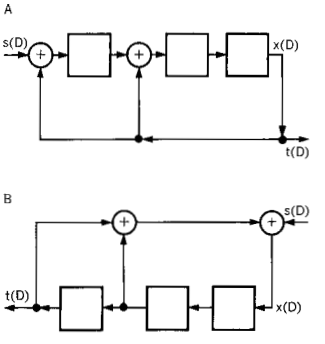
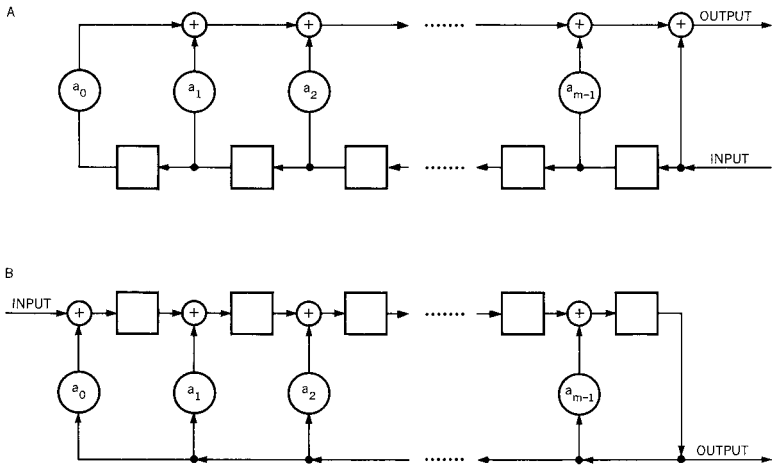


Figure 5 Generalized multiplication and division circuits



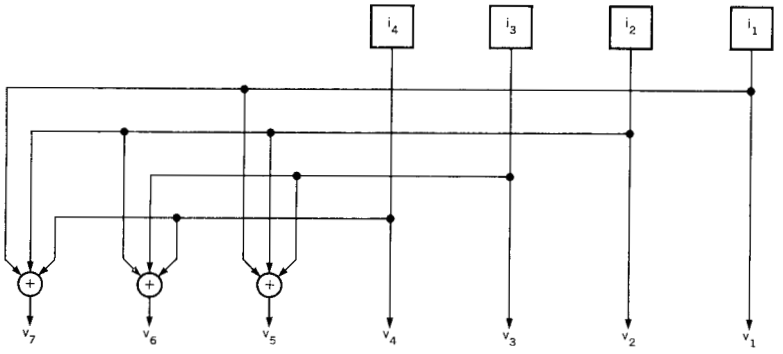
The division circuits described, perform essentially the long division process. Thus, if the input polynomial is a multiple of the dividing polynomial, the output sequence is the quotient followed by zeros. Otherwise, the output sequence is the infinite sequence corresponding exactly to what one obtains in the long division process. For example, if we divide x^3 by $x^3 + x + 1$ as follows:

$$
\begin{array}{l}
\qquad\qquad 1 + x^{-2} + x^{-3} + x^{-4} \qquad\qquad + x^{-7} + \cdots \\
x^3 + x + 1)\overline{x^3} \\
\qquad\quad x^3 + x + 1 \\
\qquad\quad \overline{\qquad\quad x + 1} \\
\qquad\qquad\quad x \quad + x^{-1} + x^{-2} \\
\qquad\qquad\quad \overline{\qquad\quad 1 + x^{-1} + x^{-2}} \\
\qquad\qquad\qquad\quad 1 \quad\quad + x^{-2} + x^{-3} \\
\qquad\qquad\qquad\quad \overline{\qquad\quad x^{-1} \quad\quad + x^{-3}} \\
\qquad\qquad\qquad\qquad\quad x^{-1} \quad\quad + x^{-3} + x^{-4} \\
\qquad\qquad\qquad\qquad\quad \overline{\qquad\qquad\quad x^{-4}} \\
\qquad\qquad\qquad\qquad\qquad\quad x^{-4} + x^{-6} + x^{-7} \\
\qquad\qquad\qquad\qquad\qquad\quad \overline{\qquad\qquad\qquad\quad \vdots}
\end{array}
$$

the outcome is $1 + x^{-2} + x^{-3} + x^{-4} + x^{-7} + \cdots$. The division circuit of Figure 3A or 3B, on the other hand, gives a corresponding output sequence $1 + D^2 + D^3 + D^4 + D^7 + \cdots$.

The circuit in Figure 4A (or the general circuit of Figure 5B) has the following special property. The contents of the registers represent the residue of the division after the last term of the input polynomial has entered the circuit. For example, if we consider the register contents from left to right as coefficients of 1, x, and x^2, respectively, in Figure 4A, then a shift to the right is equivalent to multiplication by x. The feedback connec-

Figure 6 Combinatorial encoder for a linear code



tions add the output from the third register to the contents of the first and the second registers, thus effecting $x^3 = 1 + x$ (or $x^3 + x + 1 = 0$) whenever this reduction becomes possible. The final contents of the registers clearly represent the input polynomial minus all multiples of $x^3 + x + 1$, that is, the residue of the input polynomial modulo $x^3 + x + 1$.

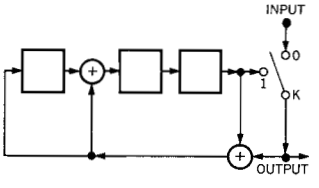
The shift-register contents of other types of division circuits do not necessarily correspond to residues. For example, the shift-register contents of the circuit in Figure 3A represent the residue of the input polynomial multiplied by x^3 modulo $x^3 + x + 1$. In general, register contents represent linear transforms of the residue coefficients described above.

Encoders and decoders

An encoder for an (n, k) linear code produces an n-tuple code word when an information k-tuple is given. This fact is illustrated by writing the symbols in the n-tuple code word as functions of the given k-tuple and implementing each of these functions (as Boolean functions) with logic circuits. For example, the linear code specified by Equation 1 may be implemented by the circuit in Figure 6. Note that the information symbols remain unchanged; thus, the code obtained is separable.

When a cyclic polynomial code is used, it is convenient to generate the code polynomials in a sequential manner. The binary cyclic code with the generator polynomial $g(x) = x^3 + x + 1$, for example, may be encoded with the shift-register (multiplication) circuits in Figures 2A and 2B yielding a nonseparable code structure. When the code is separable, a division circuit capable of producing the residue of the input polynomial modulo $g(x)$ can be used to produce the check symbols. Figure 7 shows such an encoder. During the transmission of the first k bits, information symbols are fed into the encoders shown in Figure 7. The switch K is in the 0 position, allowing the same symbols to appear unchanged at the output. At the end of k bits, the

Figure 7 Sequential encoder for a linear code



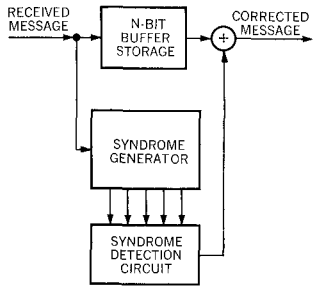
desired residue has formed in the registers and is obtained by throwing the switch K to the 1 position. Since the feedback of the division circuit is now nullified, the register contents will next appear at the output.

The cyclic code generated by the polynomial $g(x) = x^3 + x + 1$ is identical to the linear code described by the generator matrix of Equation 1. Encoders shown in Figures 6 and 7, therefore, yield the same code words when fed with the same k-tuple input.

The basic function of a decoder is to establish mapping from the syndrome (r-tuple) of the received message to an error n-tuple. By subtracting the error from the received message, one obtains the transmitted code word which, in the case of separable codes, contains the original information k-tuple.

Since the mapping being implemented can be completely specified by a table, an immediate approach to the design of a decoder is via a logic circuit that implements the table lookup procedure. When decoding delay must be minimized, the logic circuit approach in decoding can be quite attractive. The obvious limit to this approach is that the complexity of the decoding circuit tends to grow exponentially with the capability of the code used.

Figure 8 General-purpose decoder for cyclic code



With cyclic codes, simplification in the decoding circuitry is possible. Figure 8 shows a general-purpose decoder which consists of the following components: a division circuit that serves as a syndrome generator, an n-stage buffer storage that retains the message received, and a syndrome recognition circuit that usually recognizes the syndromes of error vectors that include an erroneous highest-degree digit.

To see how this decoder works, let $a(x) = a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ be the code polynomial and let $a'(s) = a(x) + e(x)$ be the received polynomial, where $e(x)$ represents the error. As mentioned earlier, the syndrome of $a'(x)$ generated here by a division circuit of $g(x)$ is independent of $a(x)$. The syndrome is obtained when the last digit of the code word, a_n, has entered the decoder. If the first digit is not in error, the syndrome detection circuit maintains a zero output and the highest order bit is obtained unaltered at the output. After a shift, the transformed syndrome corresponds to $xe(x)$, which is the original error with the coefficients advanced one position toward the high-degree end. The syndrome recognition circuit then recognizes the syndrome if the second digit in the original received message is in error. Since the same argument applies to the subsequent shifts and subsequent errors, we see that erroneous digits of a correctable error pattern can all be corrected.

The decoding circuit of Figure 8 requires a delay of n digits before the decoded message is received. The errors are corrected sequentially. Although generally applicable to all types of cyclic codes, the syndrome recognition circuit may in many cases still be too complicated (in spite of the relative simplicity in comparison with the pure combinatorial circuit). However, remarkably

simple decoding circuits of this type are possible with cyclic burst-error-correcting codes (including the Hamming codes).

If a code is used for error detection only, one merely needs a recognition circuit to determine whether the residue is zero. A nonzero residue indicates that an error has been detected.

Functional classes of error-control codes

Several functional classes of cyclic polynomial codes have been found:

- *Single-error-correcting codes.* A single-error-correcting code of length n is capable of correcting any error affecting no more than one symbol in a code block of n symbols.
- *Burst-error-correctings codes.* A burst-error-correcting code of length n is one that can correct any span of errors of fixed length b or less in a code block of n symbols.
- *Independent-error-correcting codes.* An independent- (or multiple-) error-correcting code is a code of length n that is capable of correcting up to a multiple of t errors within a code block of length n.
- *Multiple-character-correcting codes.* A multiple-character-correcting code is a code of length n characters, where a character is a group of bits with fixed length. Any combination up to a fixed number of character errors within a block may be corrected.

Depending upon channel characteristics, members of these code classes may be selected. Methods for finding generator polynomials for these codes are given in Appendix C.

Certain specialized codes are modifications of some members of previously mentioned functional classes of codes. Interleaved codes, N-dimensional codes, and shortened codes, for example, are methods of constructing stronger codes based on weaker ones. Self-orthogonal codes are characterized by their threshold logic decodability, which leads to simple decoding circuits. Synchronization codes add framing capability to error-control. Convolutional codes form a class of nonblock codes with various possible error-control capabilities and are often used in conjunction with the sequential decoding technique. Constant-weight codes are useful in channels with some special properties. Arithmetic codes are based on arithmetic operations and are useful in channels which include arithmetic processors. Certain basic properties of such specialized codes are included in Appendix D.

Coding strategy

When an error-control code is considered in a digital transmission or storage system, one should ask not only what can this code do, but also what is needed to achieve the capability of the code.

Generally speaking, the longer the block length (i.e., larger n), the more storage the decoder requires, and the greater the mini-

mum decoding delay. It is also generally true that the longer the code block the larger the class of errors to be corrected, hence the more complicated the decoding circuits. However, the distribution of errors in longer code blocks becomes much more predictable, thereby permitting the use of codes with smaller redundancy while maintaining the same reliability.

The data flow in a complex computer system may take different forms at different levels corresponding to the channels described previously. Therefore, basic requirements for error-control codes may also change in emphasis from one case to another. For example, intermachine data transmission may go through many conventional communications channels. The primary requirements of the preferred error-control scheme are high reliability and high information rate. Since decoding delay does not reduce throughput, one would tend to use longer codes with lower redundancy even though they require more decoding complexity.

For intramachine transmission, such as going in and out of an internal random-access storage, the primary coding requirements are high reliability and speed. Thus, simple decoding by circuitry is essential in keeping storage access-time small. Another feature of the codes used for intramachine transmission is that an error-control code is often used in the detecting mode, since retransmission can usually be effected by simple instructions based on the outcome of error detection. There are exceptions to such general rules. An optimum coding strategy can be achieved, and the best code obtained, only after a design engineer evaluates several alternatives.

We now outline several different courses of action he may prefer as an alternative of forward-acting full-power correction with block codes.

error detection

The main advantage of error detection is the simplicity of its implementation. An error is detected if the received message yields a nonzero syndrome. For cyclic codes, a division circuit plus a test for zero constitute a complete decoder.

The detection capability of a code is closely related to its correction capability. If a code is capable of correcting a set $\{\mathbf{e}_i\}$ of error n-tuples, then the syndromes of any two errors, $\mathbf{e}_i$ and $\mathbf{e}_j$ from the set must be distinct. This implies that any error of the form $\mathbf{e}_i + \mathbf{e}_j$ must be detectable. It should be pointed out that the code also detects many other errors. Any error of the form $\mathbf{e}_i + \mathbf{e}_j + \mathbf{v}$ is clearly detectable if $\mathbf{v}$ is a code vector (and $\mathbf{vH}^T = \mathbf{0}$). This often results in a significant reduction in the undetected and uncorrected error rate.

From the preceding, we observe that a t-error-correcting code is capable of detecting all combinations of $2t$ errors, and a burst-b-correcting code is capable of detecting any two bursts of length b or less. A Fire code generated by $g(x) = (x^c + 1)p(x)$—as described in Appendix C—when used for detection only, is capable of detecting any combination of two bursts of which the length of the shorter burst is no greater than the degree of $p(x)$. Any cyclic

code of degree r is capable of detecting all single bursts of length up to r.

Error detection is an attractive means of error control provided it is possible to effect retransmission. In the case of data transmission, this implies the existence of a reliable feedback channel, which is used to relay the request-for-retransmission message back to the sender.[12-14] Many data links within a computer system have the ability to regenerate a message at the sending end when it is not cleared at the receiving end. On the other hand, an error detected during a readback process from storage may not be successfully avoided by rereading the same message when the error is due to a permanent damage in the storage medium or when the error occurred during the writing process.

When a feedback channel is available, one should calculate, from available statistics, the probability of requests for retransmission and the average time the system is tied up because of the requests. Performance of the detection-retransmission method can then be evaluated within the context of given system parameters.[15] In general, detection and retransmission is effective against highly clustered errors. For random errors or for a combination of random and burst errors, some error will tend to appear regularly in every block. In such cases, some forward-acting error correction is necessary to maintain the performance of the transmission system.

partial correction

We have seen that, even where a feedback channel is available, some forward error correction is often needed to combat random errors. For most codes, there is a trade-off between the numbers of correctable and detectable errors. A multiple-error-correcting code is capable of correcting t errors and simultaneously detecting d errors as long as the minimum distance of this code is at least $t + d + 1$. A Fire code generated by $g(x) = (x^c - 1)g(x)$ is capable of correcting a burst of length up to b and simultaneously detecting any other burst of length up to $dc \geq b$ as long as $b + d - 1 \leq c$ and $b \leq m$, the degree of $p(x)$. See Appendix C.

Aside from the need to use partial correction in conjunction with the detection-retransmission method, there may be other reasons for the use of partial correction in the overall error-control scheme, namely, to minimize the decoding complexity. We mention here two situations wherein partial correction may prove useful.

1. In the case of multiple-error correction, decoding complexity grows exponentially with the number of errors corrected. Thus, even if a given code can correct $t > 1$ errors, one may still want to go through a single-error-correction procedure and test the syndrome for possible erroneous correction. If single errors account for a large portion of the overall error rate, considerable reduction in average decoding delay can thereby be achieved. Success of single-error correction eliminates the need to go through the more complicated t-error-correction. If two or more errors occur, the single-error-correction procedure may

make an erroneous correction in some cases. However, due to the minimum distance of the code, the result is still a detectable error. The correction algorithm specifies returning to the original message received and trying a more powerful correction procedure. A similar approach also applies to the partial correction of multiple errors up to the maximum number of correctable errors.

2. For certain classes of multiple-error-correcting codes, simple circuit implementation is possible for correcting a small number of errors. Since threshold-logic decoding has error detection and correction capabilities approaching those of multiple-error correcting codes, the combination of partial correction by logic circuitry and detection may prove very useful.

erasures

Erasures usually correspond to detected signals that are considered to be in a certain "no-confidence zone". In the case of binary level detection, the erasure zone is intermediate between the 1- and the 0-zone. In general, an erasure implies an unknown symbol (or character) at a known location.

In a pure erasure channel, locations of errors are always known. The error-correction capability of a code in an erasure channel is similar to its detection capability in a nonerasure channel. An erasure pattern is correctable if (and only if), by substituting all possible combinations of symbols at these erased digits, only one results in a code word. With a t-error-correcting code, any pattern of $2t$ erasures is correctable. This follows immediately from the fact that, with $2t$ erasures, any two n-tuple resulting from different substitutions can differ at most at $2t$ digits. However, a t-error-correcting code must have a minimum distance at least $2t + 1$, which means these two n-tuples cannot both be code words. Similarly, with a burst-b-correcting code, any pattern consisting of two erasure bursts of length b or less is correctable.

In more realistic channels, erasures are often compounded with nonerasure errors. Again, there is a trade-off between the numbers of correctable errors and erasures. For example, a multiple-error-correcting code is capable of correcting any combination of t errors and e erasures as long as the minimum distance of the code is at least $2t + e + 1$.

Generally speaking, the use of erasures tends to reduce the uncorrectable-error rate. The amount of improvement is a function of the detailed statistics of the detected signals and of the thresholds that define the erasures. The price of improvement here is a probable increase in decoding complexity. When correcting combinations of errors and erasures with a multiple-error-correcting code, one must perform the additional step of transforming the error syndromes in order to separate the erasures from nonerasures before the ordinary decoding procedures can be applied.[16] At least part of this added effort is compensated by a reduction in the number of errors to be corrected, as compared with forcing all erasures into decisions of code symbols. The erasure concept can be generalized as an increased number of levels at the detector output whereby further gain in reliability is possible.[17]

adaptive coding schemes

If the noise characteristics of a digital data channel tend to change from time to time, an adaptive coding scheme may be desirable. In the method of detection and retransmission, certain forward-acting partial correction becomes necessary if a small number of errors tend to occur regularly. The amount of partial correction can be monitored at the receiving end to cope with the varying error rate. Recently, an interesting method of adaptive decoding without feedback has been developed.[18] With this method, a received message is analyzed to determine whether the burst-error correction or the independent-error correction should be performed. Methods have been studied for changing the code used (as well as the decoding algorithm) in such a way as to minimize implementation complexity.[19,20]

sequential decoding

Although sequential decoding has been successfully applied to space communications, its use in computer systems is still in an exploratory stage. Quantitative performance evaluation of a sequential decoding algorithm is difficult without actual implementation and testing. As we have indicated previously, since the decoding algorithm can only be implemented by a computer, sequential decoding is not applicable where sufficient processing capability is not provided. Another factor that may limit the use of sequential decoding is that decoding effort is a random variable without an upper bound. However, the sequential decoding algorithm is applicable to a wide range of conditions, including those in which other block coding schemes do not perform satisfactorily. Such conditions exist, for example, where the initial error rate is high, or where high reliability is required at a high information rate.

Some error-control applications

data communications

Many IBM terminals use cyclic codes for error detection. Because of their relatively low error rates, the codes are mostly burst-detecting codes that usually have very little redundancy.

The IBM 1050 data communication system uses an interleaved code, generated by $g(x) = (x^6 + 1)$, in which six check digits form a character at the end of each message. Single burst-errors of length up to six are detectable, as are many other error patterns.

The Binary Synchronous Communication (BSC)[21] convention uses a burst-2-correcting code generated by $g(x) = (x + 1)p(x)$, where $p(x)$ is a primitive polynomial of degree 15. The BSC code is capable of detecting two bursts of length two. Also, because the minimum distance is four, BSC can detect any three or fewer independent errors in messages up to a length of $2^{15} - 1$.

Although errors on microwave links used for voice-grade channels are effectively eliminated by the use of pulse code modulation and repeaters, encoders and decoders for additional error control are provided. For example, private lines are available with additional coding equipment, wherein the code used is a shortened (200, 175) BCH type with a minimum distance equal

to eight. The generator polynomial of this code is of the form $g(x) = (x + 1)m_1(x)m_3(x)m_5(x)$, where $m_1(x)$, $m_3(x)$, and $m_5(x)$ are polynomials of degree eight. The (200, 175) code is obtained by shortening a full-length (255, 230) code. This code is capable of correcting three independent errors and, in addition, detecting four errors. Retransmission is requested if an uncorrectable error is detected. The use of a convolutional code with one-sixth redundancy is also an option with the direct-distance-dialing switched network.

data storage

Although magnetic cores are highly reliable, such storage elements as drivers, sense amplifiers, and read-write gates, which control the storage operation, are subject to occasional failures. The use of an error-control code in the CPU of a computing system not only helps to locate failures, but also keeps the CPU in operation when the effect of a failure is within the correction capability of the code used.

The IBM 650 central processing unit uses a "bi-quinary" code, which encodes a decimal digit into seven binary digits with two 1's. This code, like the four-of-eight code, detects all odd numbers of errors.

auxiliary storage

The IBM 7030 (STRETCH) computer uses a single-error-correcting double-error-detecting code with 64 data bits and eight check bits. The encoding and decoding are implemented by logic circuits.

The IBM 7070 data processing system uses a "two-of-five" code with an additional overall parity check. Many other CPU's, including SYSTEM/360, use single parity checks for error detection.

Disk files, like other magnetic surface-recording systems, are vulnerable to surface irregularities. Therefore, protection against burst error is usually needed. As the recording density increases, more powerful coding schemes are needed. The IBM 1300-series disk storage uses a cyclic code for burst detection, in which there are 13 check digits at the end of every record. The IBM 2301 drum storage unit also uses a cyclic code with 19 check digits for error detection. Most of the other disk files use similar cyclic codes for error detection.

Magnetic tape units used today contain several tracks, and a character or a byte is obtained by reading one bit from each track. Error control is necessary since tapes are relatively less reliable than magnetic cores. Control can be achieved in a number of ways. The tractor tape unit has 22 tracks, 16 of which are information bits and six are check bits. Each character is a (22, 16) code obtained by shortening a (31, 25) BCH code with minimum distance of four. The IBM 727 and 729-series magnetic tape units use a two-dimensional coding scheme. One track, which provides a *vertical redundancy check* (VRC), is used for an overall check on each character. Also, one character at the end of each record is used for an overall check on each track and is known as the *longitudinal redundancy check* (LRC). The overall code detects errors in a single track, plus many other errors.

The IBM 2400-series magnetic tape units use a coding scheme involving another character next to LRC as a check based on a cyclic code, in addition to the VRC and LRC already described. This check is called *cyclic redundancy check* (CRC) and is discussed in greater detail in Appendix E.

Digital Cypress error control

The photo-digital storage for the IBM 1360 computer, known as Digital Cypress,[45] uses a (366, 300) Reed-Solomon code, which is one of the most sophisticated codes ever used for storage. With six bits in each character, this code is a multiple-character-error-correcting code with a minimum distance (on the character basis) equal to 12, which requires 11 check characters (66 bits). The full length of the code is $2^6 - 1$ characters (i.e., 63 characters or 378 bits). There are 300 bits (or 50 characters) of data plus two characters for line number and 11 check characters. The code is capable of correcting any combination of independent and burst errors representable by five characters. A sixth character error, plus many others, can be detected.

Except for the encoder and the syndrome-generating circuit, the Digital Cypress decoding procedure is implemented by programming, the strategy for which may be outlined as follows. When a nonzero syndrome is detected, a rescan is called for first. If the error is still present, the program goes to a single-error partial-correction subroutine. If that procedure is unsuccessful in correcting the error, a two-error partial-correction subroutine is called. The full-power correction routine is used only when both the single-error- and the double-error-correction subroutines are unsuccessful.

Concluding remarks

We have developed basic concepts of error-control coding, with emphasis on the use of cyclic codes, which form a subclass of linear block codes. The use of an error-control scheme should be an integral part of the overall system design, rather than a "remedy" or a "bonus" for a system with unsatisfactory reliability. To achieve a proper error-control scheme, a systems engineer needs an extensive knowledge of existing coding methods and their implementations. Since this paper is not intended to give a full treatment of the theory and applications of all types of codes, the aim has been to expose some of the underlying principles involved in selecting an error-control coding scheme for a realistic computer or communication system.

The demands on overall data-processing and communications capacities have been increasing and are expected to grow. This implies a prevailing need to fully utilize every communication or memory channel available. One approach is by way of error-control coding. With advances in integrated circuit technology, costs of logic and storage elements are declining in comparison with increasing rates of data-processing. Thus, circuit-implemented error-control schemes are expected to become increasingly

attractive. One objective of system designers is to achieve "ultra reliable" components, in which error-control capabilities are an integral part of the monolithic circuit design.

As applications of more sophisticated error-control coding schemes for computer and communications systems become more extensive, one may expect coding principles to be applied to other types of problems. For example, algebraic procedures typical of encoding and decoding can be used to obtain solutions in such problem areas as file organization and document retrieval.[22,23] Since a document in a file is usually characterized by a list of "descriptors" contained in a "dictionary," a binary n-vector can identify a document, wherein each position of the n-vector represents a descriptor. Storage required for such a dictionary becomes too large to be practical in most cases. However, if we regard the n-vectors as errors, the vectors can be transformed into r-tuples (syndromes) appropriate to the code selected. The r-tuples can then be used to identify documents in the file. Requests for retrieval can be handled with the help of the corresponding decoding algorithm.

The design of matrix switches, such as those used in main storage arrays, is another example. It has been shown that certain codes can be used to determine selection patterns in a matrix switch so that all driving power is channeled to the selected output only.[24,25]

Coding concepts and techniques are also potentially useful in such other areas as signal design, digital modulation, pattern recognition, fault diagnosis, image processing, and cryptography.

Appendix A: Structure of linear codes

The first four columns of the four-by-seven coefficient matrix in Equation 1 form an identity submatrix. In general, the generator matrix of a separable code is a k by n matrix containing a k by k identity submatrix. The columns of the submatrix correspond to information positions.

A fundamental property of a linear code is that if $\mathbf{v}_i$ and $\mathbf{v}_j$ are two code words, then $\mathbf{v}_k(=\mathbf{v}_i + \mathbf{v}_j)$ must also be a code word, since

$$\mathbf{v}_i + \mathbf{v}_j = \mathbf{x}_i\mathbf{G} + \mathbf{x}_j\mathbf{G} = (\mathbf{x}_i + \mathbf{x}_j)\mathbf{G} = \mathbf{x}_k\mathbf{G} = \mathbf{v}_k$$

The use of a generator matrix to represent a code eliminates the need to list all the n-tuples in the code set. In the binary case, a k by n generator matrix uniquely specifies the code set containing 2^k n-tuples.

With respect to every linear code set V, it is possible to find a set U of n-tuples such that U and V are "orthogonal" in the sense that for any n-tuple code word $\mathbf{v}$ in V and any n-tuple code word $\mathbf{u}$ in U,

$$\mathbf{v}\mathbf{u}^T = \mathbf{0}$$

Here, $\mathbf{v}$ and $\mathbf{u}$ are row matrices, and $\mathbf{u}^T$ denotes the transpose of $\mathbf{u}$.

The set U is obtained by summing all possible combinations of rows of an r by n parity-check matrix. The orthogonality requirement can, therefore, be written as

$$\mathbf{GH}^T = \mathbf{0}$$

Given the code word $\mathbf{v} = (v_1, v_2, \cdots, v_n)$ in V that satisfies the equation

$$\mathbf{vH}^T = [v_1, \cdots, v_n]\begin{bmatrix} h_{11} & h_{21} & \cdots & h_{r1} \\ h_{12} & h_{22} & \cdots & h_{r2} \\ \vdots & \vdots & & \vdots \\ h_{1n} & h_{2n} & \cdots & h_{rn} \end{bmatrix} = \mathbf{0}$$

then the following set of linear simultaneous equations is obtained:

$$\left.\begin{aligned} h_{11}v_1 + h_{12}v_2 + \cdots + h_{1n}v_n &= 0 \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots & \\ h_{r1}v_1 + h_{r2}v_2 + \cdots + h_{rn}v_n &= 0 \end{aligned}\right\}$$

A parity-check matrix $\mathbf{H}$ specifies r linear simultaneous parity-check equations that must be satisfied by the symbols of every code word from V.

To obtain the parity-check matrix, we can write the generator matrix in the standard form $\mathbf{G} = [\mathbf{I}_k \ \mathbf{P}]$, where $\mathbf{I}_k$ is a k by k identity submatrix and $\mathbf{P}$ is a k by r submatrix that describes the interdependence between information and parity-check symbols. The parity-check matrix can then be written as $\mathbf{H} = [\mathbf{P}^T \ \mathbf{I}_r]$. One can check to see that

$$\mathbf{GH}^T = [\mathbf{I}_k \ \mathbf{P}]\begin{bmatrix} \mathbf{P} \\ \mathbf{I}_r \end{bmatrix} = \mathbf{0}$$

Although the specification of either a generator matrix or a parity-check matrix uniquely determines a linear code, neither the generator matrix nor the parity-check matrix is unique. In general, different generator or parity-check matrices for the same code are obtainable from one another by means of nonsingular linear transformations.

Appendix B: Structure of polynomial codes

Given a generator polynomial $g(x)$ of a cyclic code, a corresponding generator matrix $\mathbf{G}$ can be written by listing k n-tuples (corresponding to k code polynomials), none of which can be obtained by a linear combination of the others. For example, n-tuples corresponding to $x^i g(x)$, $i = k - 1, k - 2, \cdots, 0$ constitute k rows of a generator matrix of the same code. The generator matrix of the specific form $\mathbf{G} = [\mathbf{I}_k \ \mathbf{P}]$ can be determined as follows. For each x^i, where $i = n - 1, n - 2, \cdots, r$, find the residue $p_i(x) \equiv x^i$, modulo $g(x)$. The k polynomials

$$x^i + p_i(x) \qquad \text{(where } i = n-1, n-2, \cdots, r\text{)}$$

are multiples of $g(x)$ and are, therefore, code words. Also, by writing the corresponding n-tuples as rows, the result is a generator matrix of the form

$$\mathbf{G} = [\mathbf{I}_k \ \mathbf{P}]$$

To obtain a *parity-check matrix* $\mathbf{H}$, simply write each n-tuple corresponding to $x^i h(x)$, $i = 0, 1, \cdots, r-1$ in the reverse order. The r rows thus obtained form a parity-check matrix. This procedure can be checked by identifying the product of any row of $\mathbf{G}$ corresponding to $x^i g(x)$, where $0 \leq i \leq k-1$, and any row of the previously mentioned $\mathbf{H}$ to be identical to one of the missing coefficients in the equation $g(x)h(x) = x^n + 1$. To obtain the specific form $\mathbf{H} = [\mathbf{P}^T \ \mathbf{I}_r]$, we find the residue $q_i(x) \equiv x^i$, modulo $h(x)$, for each x^i, where $i = k, k+1, \cdots, n$. The reversal of each n-tuple corresponding to the polynomials $x^i + q_i(x)$, where $i = k, k+1, \cdots, n$, which are all multiples of $h(x)$, gives the r rows of the parity-check matrix in the desired form $\mathbf{H} = [\mathbf{P}^T \ \mathbf{I}_r]$.

For example, consider the primitive polynomial $g(x) = x^3 + x + 1$, which as a generator polynomial, generates a code of length $2^3 - 1 = 7$. To write the corresponding generator matrix, calculate residues of x^i as follows:

$$p_3(x) \equiv x^3 \equiv x+1, \quad p_4(x) \equiv x^4 \equiv x^2+x, \quad p_5(x) \equiv x^5 \equiv x^3+x^2 \equiv x^2+x+1,$$

$$p_6(x) \equiv x^6 \equiv x^3+x^2+x \equiv x^2+1, \quad \text{modulo} \quad (x^3+x+1)$$

The following generator matrix contains rows corresponding to the vector representation of polynomials $x^i + p_i(x)$, $i = 6, 5, 4, 3$:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

To write the parity-check matrix, first calculate $h(x) = (x^7 + 1)/(x^3 + x + 1)$ as follows:

$$\begin{array}{r} x^4 + x^2 + x + 1 \\ x^3 + x + 1 \overline{)\, x^7 \qquad\qquad\qquad + 1} \\ \underline{x^7 + x^5 + x^4} \\ x^5 + x^4 \qquad\qquad + 1 \\ \underline{x^5 \qquad + x^3 + x^2} \\ x^4 + x^3 + x^2 \qquad + 1 \\ \underline{x^4 \qquad + x^2 + x} \\ x^3 \qquad + x + 1 \\ \underline{x^3 \qquad + x + 1} \\ 0 \end{array}$$

Thus, $h(x) = x^4 + x^2 + x + 1$, and

$$q_4(x) \equiv x^4 \equiv x^2+x+1, \qquad q_5(x) \equiv x^5 \equiv x^3+x^2+x,$$

$$q_6(x) \equiv x^6 \equiv x^4+x^3+x^2 \equiv x^3+x+1, \quad \text{modulo} \quad (x^4+x^2+x+1)$$

Writing, in reverse order, the vector representation of polynomials $x^i + q_i(x)$, where $i = 4, 5, 6$, we have the parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

It can be seen that the cyclic code in this example is identical to the linear code of the last example.

Appendix C: Methods for finding generator polynomials

single-error-corecting codes

Single-error-correcting codes are often referred to as Hamming codes.[3] In such a code, any two distinct single errors x^i and x^j must yield distinct syndromes. Let $\mathbf{e}_i$ and $\mathbf{e}_j$ be row vectors corresponding to x^i and x^j respectively

$$\mathbf{e}_i\mathbf{H}^T \neq \mathbf{e}_j\mathbf{H}^T$$

or

$$(\mathbf{e}_i + \mathbf{e}_j)\mathbf{H}^T \neq 0$$

Thus, the generator polynomial $g(x)$ never divides $x^i + x^j$ for any i and j. This condition can be satisfied if we choose the code length n to be e, where e is the *period* of $g(x)$. The period e is the smallest integer such that $g(x)$ divides $x^e + 1$. With i and j both smaller than n, $g(x)$ can never divide $(x^i + x^j) = x^j(x^{i-j} + 1)$. In particular, if an rth degree $g(x)$ is *irreducible* (i.e., not divisible by any other polynomial except 1), then the period of $g(x)$ divides $2^r - 1$. Then, if the period of $g(x)$ is $2^r - 1$, $g(x)$ is said to be *primitive*. A single-error-correcting code generated by a primitive polynomial is "close-packed" in the sense that all 2^r syndromes are used for the prescribed correctable errors, $2^r - 1$ single errors and one zero error. Since primitive polynomials are known to exist for all degrees, Hamming codes of length $2^r - 1$ exist for all r.

burst-error-correcting codes

One way to generalize the class of single-error-correcting codes is to obtain codes to correct any error burst within a span of b digits. Such codes are called burst-b correcting codes[26,27] and are suitable for channels with occasional error bursts.

A class of burst-error-correcting codes, known as *Fire codes*,[27] is best defined as the class cyclic codes wherein the generator polynomials take the form

$$g(x) = (x^c + 1)p(x)$$

Here, $c \geq 2b + 1$, the length of the code is the least common

multiple (LCM) of c and the period of $p(x)$, and the degree of $p(x)$ is at least b. When these conditions are satisfied, the resulting code is capable of distinguishing syndromes resulting from any two burst errors each of length no greater than b.

There are burst-error-correcting codes other than the class of Fire codes; many are optimum codes, which are more efficient than the Fire codes of the same length and maximum correctable bursts.[28]

independent-error-correcting codes

It was pointed out earlier that an irreducible polynomial $p(x)$ can be used to generate a single-error-correcting code of a length equal to the period e of the polynomial $p(x)$, where e is the smallest integer such that $p(x)$ divides $x^e + 1$. If we properly combine several irreducible factors of $x^e + 1$, we can obtain the generator polynomial of an independent (or multiple)-error-correcting code. Given that some α is a root of $m_1(x) = p(x)$, i.e., $p(\alpha) = 0$. Then for any i, only one among these factors, denoted by $m_i(x)$, satisfies $m_i(\alpha^i) = 0$. These $m_i(x)$, called the *minimum polynomials* of x^i, are not necessarily distinct for different i's.

BCH codes

The binary BCH (Bose-Chaudhuri-Hoquenghem) codes form a class of multiple-error-correcting codes[6,29-31] that can be described in terms of the mininum polynomials $m_i(x)$ as follows. Let the generator polynomial be defined as

$$g(x) = \text{LCM}\ [m_1(x), m_3(x), \cdots, m_{2t-1}(x)] \tag{2}$$

then the code generated by $g(x)$ is a t-error correcting code with a minimum distance at least $2t + 1$ and a length $n = e_1$, where e_1 is the period of $m_1(x)$.

If the generator polynomial is

$$g(x) = \text{LCM}\ [m_0(x), m_1(x), m_3(x), \cdots, m_{2t-1}(x)] \tag{3}$$

the corresponding code has a minimum distance of at least $2t + 2$. The length of this code is again $n = e_1$ for $t \geq 1$. For $t = 0$, $g(x) = m_0(x) = x + 1$. The code generated by $g(x) = x + 1$ has a minimum distance of 2. This is a code with a single parity digit, and the code length can be arbitrary.

Given any $m_1(x)$, one could obtain $m_i(x)$ for any i by using algebraic procedures.[32,33] However, this is generally time consuming and unnecessary since tables of binary minimum polynomials are available.[34]

examples

As an example, assume that we are generating a binary double-error-correcting code of length $n = 2^6 - 1 = 63$. Since a primitive polynomial of degree six has a period equal to 63, we select $m_1(x)$ as a primitive polynomial. From Reference 34, if the primitive polynomial $x^6 + x + 1$ is chosen as $m_1(x)$, then $m_3(x) = x^6 + x^4 + x^2 + x + 1$. From Equation 3, the generator polynomial

$$\begin{aligned} g(x) &= \text{LCM}\ [m_1(x), m_3(x)] = m_1(x)m_3(x) \\ &= (x^6 + x + 1)(x^6 + x^4 + x^2 + x + 1) \\ &= x^{12} + x^{10} + x^8 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

generates a (63, 51) code with a minimum distance at least 5, good for double independent-error correction. Note that the coefficients in the product can be obtained by first writing the product in the ordinary fashion. Then all even coefficients are transformed to 0's and all the odd ones to 1's.

The period of $m_i(x)$ may be smaller than that of $m_1(x)$; the degree of $m_i(x)$ may also be smaller than that of $m_1(x)$. Such properties are sometimes useful, as shown in the following example.

With the same $m_i(x)$ as used in the last example, if we let $m_1'(x) = m_3(x)$ and $\beta = \alpha^3$, such that $m_1'(\beta) = m_3(\alpha^3) = 0$, then $m_3'(x) = m_9(x)$, where $m_3'(\beta^3) = m_9(\alpha^9) = 0$. From Reference 34 we find that $m_9(x) = x^3 + x^2 + 1$. From Equation 3, the generator polynomial is

$$g(x) = (x + 1)(x^6 + x^4 + x^2 + x + 1)(x^3 + x^2 + 1)$$

$$= x^{10} + x^7 + x^6 + x^4 + x^2 + 1 \tag{4}$$

which generates a (21, 11) code with a minimum distance of 6.

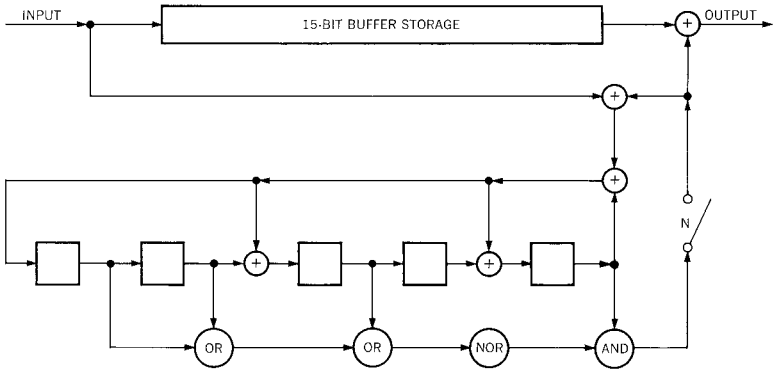
It should be pointed out that the minimum distance d guaranteed by the BCH code in Equation 2 is just a lower bound to the actual minimum distance of the code. For example, the primitive binary polynomial $m_1(x) = x^{11} + x^2 + 1$ has a period $2^{11} - 1 = 89 \times 23$. The polynomial $m_{89}(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ has a period of 23. Assuming $\beta = \alpha^{89}$ and $m_1'(x) = m_{89}(x)$, then the roots of $m_1'(x)$ are β, β^2, β^4, β^8, β^{16}, $(\beta^{32} = \beta^9)$, β^{18}, $(\beta^{36} = \beta^{13})$, $(\beta^{26} = \beta^3)$, β^6, β^{12}. Since $m_1'(x) = m_2'(x) = m_3'(x) = m_4'(x)$, as a BCH code, $m_1(x)$ generates a (23, 12) code of minimum distance at least 5. However, the (23, 12) code is equivalent to the Golay code[35] with a minimum distance equal to 7. Other BCH codes have also been found to have actual minimum distances exceeding those guaranteed by the theory of BCH codes.[36]

Error-correction procedures of BCH codes are rather complicated. They generally involve solving the roots of a t-degree polynomial and a set of t simultaneous equations, where t is the number of correctable errors. The number of operations needed to perform these procedures grows exponentially with respect to t. Recent research suggests ways of significantly reducing the decoding complexity of BCH codes.[37-39] Perhaps decoding complexity will eventually increase only linearly with t.

For many applications where the number of errors to be corrected in a code block is small, logic implementation of table lookup is a practical solution to the decoding of BCH codes. Another attractive method of implementation by means of majority gates can be used for a class called "self-orthogonal" codes, which includes certain BCH codes. This subject is covered later in Appendix D.

Another well-known class of multiple-error-correcting code is the class of Reed-Muller codes.[40,41] Although not originally formulated in terms of cyclic codes, Reed-Muller codes have been shown to be obtainable from a special class of BCH codes.[42]

Figure 9 Decoder for a burst-2 code



multiple-burst-correcting codes

The BCH codes described earlier exist in other than binary cases. A q-nary BCH code can be generated by a q-nary polynomial (a polynomial with q-nary coefficients), provided the q symbols can be identified as elements in a field.[43] A character (or a byte) consisting of a binary m-tuple, for example, may be considered as belonging to a field of 2^m elements.

Reed-Solomon codes

Reed-Solomon codes are a special class of BCH codes where the message symbols are m-tuples.[44] When used for binary messages, binary symbols must be grouped as m-tuples (or characters). A generator polynomial taking the form

$$g(x) = (x - \alpha)(x - \alpha^2) \cdots (x - \alpha^{d-1}) \tag{5}$$

generates a code with minimum distance of at least d. Note that the coefficients of the generator polynomial and code polynomials are now m-tuples and the distance between two code words is the number of places wherein corresponding m-tuples differ. The length of this code is $e = 2^m - 1$ characters, or $m(2^m - 1)$ binary digits.

Because of their independent-character-error correcting capability, Reed-Solomon codes are effective against multiple bursts of error if they occur within a code block. The decoding procedure is rather complex and usually requires program implementation. The code efficiency is usually attractive when compared with the efficiency of competitive schemes, such as the use of interleaved codes. A Reed-Solomon code with a minimum distance equal to 12 has been used in Digital Cypress.[45]

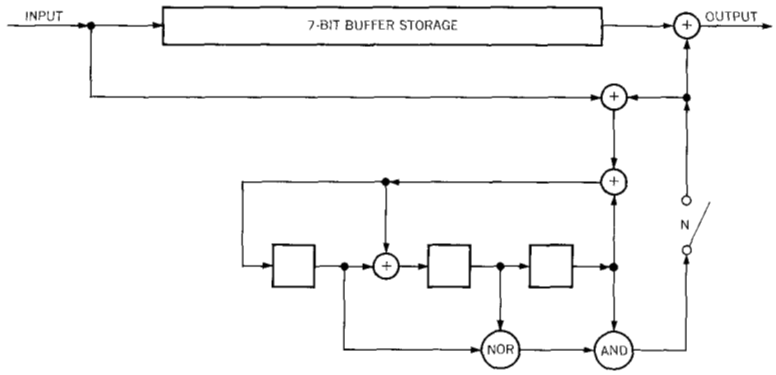
example decoders

We indicated previously that burst-error-correcting codes can easily be implemented. This is illustrated in the following example. A binary code having as its generator polynomial

$$g(x) = (x + 1)(x^4 + x + 1) = x^5 + x^4 + x^2 + 1$$

is a burst-2 correcting code of length 15, a decoder for which is

Figure 10 Decoder for a single-error-correcting code



shown in Figure 9. The registers in the division circuit contain the residue of $x^3e(x)$ modulo $g(x)$, where $e(x)$ is the error polynomial. Since the syndrome detection circuit must recognize the syndrome when the error burst is located at the high-degree end, we may write the corresponding error polynomial as

$$e(x) = x^{15-2}\, b(x)$$

where $b(x)$ is the error-burst polynomial of degree $b - 1 = 1$. The syndrome of this $e(x)$ is the residue of $x^{15-2+5}\, b(x)$ modulo $(x^5 + x^4 + x^2 + 1)$, which is simply $x^3b(x)$. The existence of three zeros in this syndrome is taken as the basis of syndrome detection as shown in Figure 9. Once the burst location is determined, feedbacks in the division circuit can be cut off or, as shown in Figure 9, nullified by establishing an additional feedback path. The detected error pattern (including no error) is then gated through and removed from the received message coming out of the 15-bit buffer storage. Switch N is closed only during the second n-bit cycle.

Another example is the single-error-correcting code generated by $g(x) = x^3 + x + 1$, a decoder for which is shown in Figure 10. The operation of this decoder is similar to that shown in Figure 9.

In some applications, the input message may not be in the exact serial form. Combinatorial decoders or decoders that combine serial and parallel operations then become distinct possibilities.[46,47]

Appendix D: Specialized error-control codes

interleaved codes

The interleaving of codes is just like the time-division multiplexing of a number of messages. Each "subcode" consists of symbols separated periodically by m digits; there are m such subcodes. Usually all m subcodes are generated by the same polynomial

$g'(x)$. Clearly, if the length of the subcode is n', the overall code length is $n = mn'$. The generator polynomial of the interleaved code can be shown to be

$$g(x) = g'(x^m)$$

where $g'(x)$ is the generator polynomial of individual subcodes.

Interleaved codes tend to break up error bursts, and subcodes interpret them as independent errors. Thus, one can use independent-error-correcting codes of acceptable decoding complexity against burst or multiple-burst errors, which might otherwise require a multiple-burst correcting code with impractical decoding complexity. On the other hand, a single-burst-correcting code with simple implementation cannot handle long bursts (e.g., drop-outs) unless the code is long. In that case, long code words would be exposed to some additional errors not protected by the code. The main disadvantage of interleaved codes is that the redundancy requirement is relatively high in comparison with that of multiple-burst-error-correcting codes.

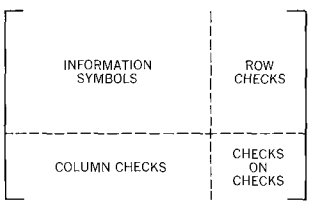
N-dimensional codes

The N-dimensional codes are, as the name suggests, best discussed in geometric terms. Figure 11 shows a two-dimensional code format in which each row belongs to a subcode and each column belongs to another (not necessarily distinct) subcode.

If d_1 and d_2 are respectively the minimum distances of row and column subcodes, then the two-dimensional code has a minimum distance $d = d_1 d_2$. More dimensions can be added to the code to further strengthen the correction capability.

The above two-dimensional code is equivalent to a two-level interleaved code. Columns of information symbols can be considered as being interleaved with the row subcode, and N iterations of interleaving clearly result in an N-dimensional code. It is from this point of view that N-dimensional codes are often referred to as iterated codes.[48] The geometrical interpretation of N-dimensional codes also enables one to obtain simple implementations of such codes especially for such storage devices as tapes and core arrays whose geometrical configurations are ideal.

Figure 11 Two-dimensional code format



An N-dimensional code may also suffer from the high redundancy requirement when used in burst channels because of interleaving. Nevertheless, such a code has the attractive feature that as long as the error rate is reduced in each level of iteration, more and more iterations will, in theory, make the error rate diminish while keeping the information rate nonzero.[49]

shortened codes

We have seen that in any cyclic code capable of correcting single errors, the code length should not exceed e, the period of the generator polynomial. However, an (n, k) code can be shortened to become an $(n - s, k - s)$ code by constraining the s high-degree digits of the code polynomial to be always zero. These s digits are then omitted from all code words. The linear sequential encoder of Figure 7 can be used for shortened codes without change. However, if the decoding delay is to be $n' = n - s$ digits

instead of n digits, the input of the division circuit in the decoder of Figure 8 should be premultiplied by x^s. The same syndrome detection circuit can then be used.[50]

Shortened codes are often used because natural code lengths may not be suitable in some applications. They can also be used to improve reliability, since with the reduced code length $(n - s)$, the expected number of errors is reduced by a factor $(n - s)/n$. The most attractive feature of shortened codes, however, is that the maximum correctable errors may now exceed what was originally possible with full-length codes.[51] This feature is particularly desirable with burst-error correcting codes, since the increased correcting capability presents no extra decoding complexity. In applications to variable length messages, codes that have increased capabilities at shorter lengths can achieve additional reduction in overall error rate.

threshold-logic-decodable codes

We have seen that decoding complexity is a severe limitation to the application of powerful BCH codes. It is, therefore, desirable to find new classes of codes with structures that enable one to use simple decoding procedure. Codes obtained from projective and Euclidean geometries have recently been shown to be decodable by threshold logic.[52] We shall illustrate the basic concept with a special class of binary "self-orthogonal" codes.[53]

Self-orthogonality is defined on the parity-check matrix as follows: the set of rows $(h_1, h_2, \cdots h_J)$ in a parity-check matrix **H** with 1's in a particular column i are self-orthogonal on the ith column if, in this set (considered as a submatrix), no other column contains two or more 1's. To decode the digit corresponding to the ith column of **H**, we first assume that the error at this digit is unknown, and that each of the J parity equations from the set gives an "estimate" of this error. The majority determines the final error value. Since an error corresponding to the ith column has J votes, while an error at any other position has at most one vote (because of the self-orthogonality), the majority decision must be correct as long as the total number of errors does not exceed $J/2$. If the self-orthogonality condition can be established for every digit (not necessarily with the same parity-check matrix), the code is threshold decodable with a minimum distance at least $J + 1$.

The most interesting case occurs when the code is cyclic, because a decoder with the general form shown in Figure 8 can be used. The syndrome detection circuit, in this case, contains majority logic with inputs from J modulo-2 adders performing the set of J parity checks found to be self-orthogonal on the highest-degree digit. We now demonstrate this with an example.

self-orthogonal decoding example

The code generated by $g(x) = x^{10} + x^7 + x^6 + x^4 + x^2 + 1$ of Equation 4 was shown to be a (21, 11)-code with minimum distance equal to six. Using the division circuit of Figure 4A in the decoder of Figure 8, the contents of the shift-registers (considered as an r-tuple) give the syndrome of the error, which, in this case, is the residue of the received polynomial modulo $g(x)$.

The ith column of the parity-check matrix can be written as the residue of x^{i-1} modulo $g(x)$ as follows:

$$\mathbf{H} = \left[\begin{array}{cccccccccc|cccccccccc} 0 & 0 & \cdot & \cdot & \cdot & & & 0 & 1 & & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & & & & & & & 1 & 0 & & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ & & & & & & 1 & & & & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \cdot & & & & & 1 & & & & \cdot & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ \cdot & & & & 1 & & & & & \cdot & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ \cdot & & & 1 & & & & & & \cdot & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ & & 1 & & & & & & & & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ & 1 & & & & & & & & & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & & & & & & & & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & \cdot & \cdot & \cdot & & & 0 & 0 & & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \tag{6}$$

This matrix does not satisfy the desired "self-orthogonality" condition. However, an equivalent parity-check matrix can be obtained by cyclicly shifting the first row of $\mathbf{H}$ in Equation 6. There are five such cyclic shifts with a 1 in the right-most column (because the row has five 1's) as shown in Equation 7.

$$\mathbf{H}' = \left[\begin{array}{cccccccccc|cccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{7}$$

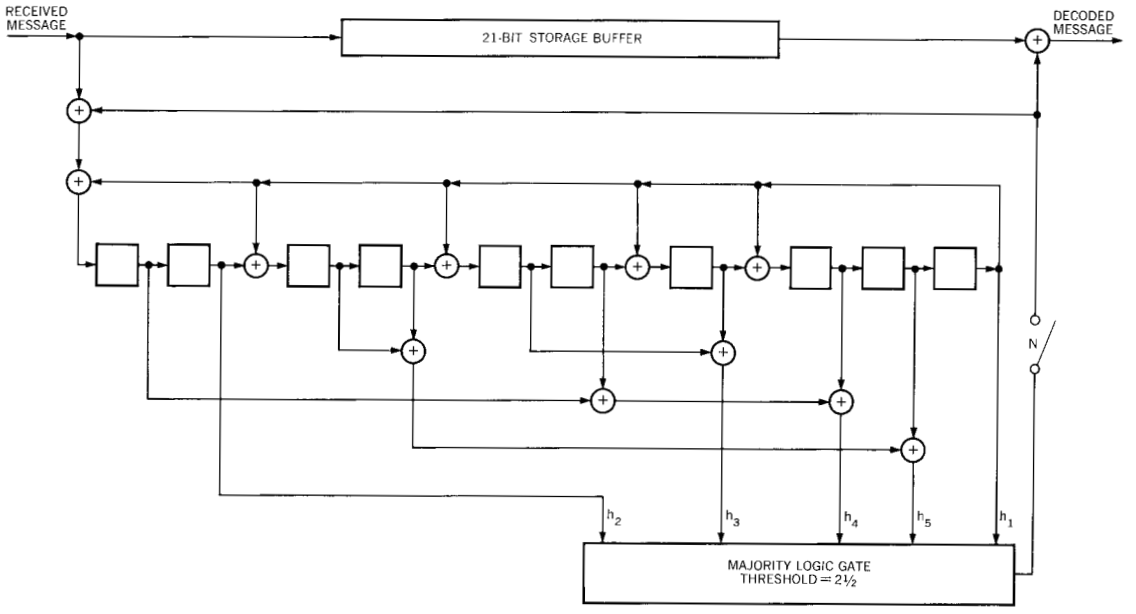
The five rows of $\mathbf{H}'$ are self-orthogonal on the right-most column, since no other column contains two 1's. The minimum distance is 6. Any row of $\mathbf{H}'$, denoted by $\mathbf{h}_i$, is a linear combination of a unique collection of rows in $\mathbf{H}$ and can be "synthesized" from the ten left-most digits by adding rows of $\mathbf{H}$ (of Equation 6) with 1's at the desired position. These sums are equivalent to modulo-2 additions of the contents of the corresponding shift-registers. A complete implementation of the decoder is shown in Figure 12.

Self-orthogonal codes, such as the one just discussed, belong to a general class of threshold-logic decodable codes, which are derived from finite geometries. For more details regarding the recent developments in threshold decodable codes, see References 54 and 55.

synchronization codes

The error-control codes discussed thus far deal with additive errors, and we assume that there is no misidentification of locations of symbols. In real transmission or storage systems, however, synchronization errors can occur at a bit level, character level, and even at a higher level, where the framing of code words is involved.

Figure 12 Decoder for a self-orthogonal code



Various methods of controlling synchronization errors have been suggested. The use of a synchronization sequence with a sharp autocorrelation function[56] sets up the word-framing. To avoid subsequent loss of word synchronization due to the possible loss of bit synchronization, such special sequences may be inserted before each code word, or periodically at longer intervals to avoid the need for excessive redundancy.

When a cyclic code is to be used for error control, it is possible to incorporate synchronization-error control in the code capability. Since, in that case, a cyclic shift of a code word is also a code word, ordinary coding schemes must be modified if a slip in word framing is to be controlled within the context of a code. There are three possibilities:

1. Add a fixed n-tuple, with a special synchronization property, to every code word. (Such a code is known as a "coset code.") The same n-tuple is subtracted from the received message after the word-framing is established.[57]
2. Use a shortened cyclic code to control word-framing.[57]
3. Use an extended cyclic code for the same purpose.[58,59]

Recovering errors due to the loss or insertion of bits within a code block is a different problem and has yielded relatively few results.[60,61] A more practical method is the detection of this type of errors accompanied by a possible request for retransmission.

convolutional codes

The relationship between information symbols and code symbols need not be confined to disjoint blocks. In a *convolutional* (or *recurrent*) *code*, check digits in a given block, check some of

the information digits in other blocks as well. One may describe a convolutional code as one that has overlapping blocks. In a separable linear code, the generator matrix may be written in the standard form $\mathbf{G} = [\mathbf{I}_k \ \mathbf{P}]$. Similarly, we may write the generator matrix for a truncated convolutional code of length $n' = m(k + r)$ as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_k & \mathbf{P}_0 & \mathbf{0} & \mathbf{P}_1 & \cdots & \mathbf{0} & \mathbf{P}_{m-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_k & \mathbf{P}_0 & \cdots & \mathbf{0} & \mathbf{P}_{m-2} \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & \mathbf{I}_k & \mathbf{P}_0 \end{bmatrix}$$

Here, the first k information digits are related to the r following check digits in the same block by $\mathbf{P}_0$ and are related to the check digits in the $m - 1$ following blocks by $\mathbf{P}_1, \cdots, \mathbf{P}_{m-1}$. The corresponding parity-check matrix is the following:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^T & \mathbf{I}_r & \mathbf{0} & \mathbf{0} & & \\ \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I}_r & & \\ \vdots & \vdots & \vdots & \vdots & \cdot & \\ \vdots & \vdots & \vdots & \vdots & & \cdot \\ \mathbf{P}_{m-1}^T & \mathbf{0} & \mathbf{P}_{m-2}^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I}_r \end{bmatrix}$$

Although convolutional codes for correcting burst errors[63–65] and independent errors[66,67] have been studied, at present they are not as well understood as block codes. As far as theoretical error-control capability is concerned, there appears to be no significant difference between block codes and convolutional codes.[68]

There are two different approaches in decoding a convolutional code. The first is "deterministic decoding," in which syndromes are calculated and algebraic procedures are carried out to determine the error sequence, similar to the decoding of block codes. However, if the decoding results of previous blocks are fed back to modify syndromes that determine the following blocks, any decoding error may "propagate" to succeeding blocks. Although the error propagation problem may not be serious, it must be analyzed and evaluated when convolutional codes are used.

Another method of decoding a convolutional code is known as the "sequential decoding" technique.[69–71] With sequential decoding, one evaluates the accumulated likelihood of correct decisions at each digit and accepts a digit only after a certain number of succeeding digits tend to confirm (in terms of accumulated likelihood measure) that the first digit is correct. If succeeding digits indicate that the first digit is in error, a search through the code tree, based on a predetermined algorithm follows, with corresponding likelihood evaluated, until a satisfactory decoding of the digit is found.

The following can be said about sequential decoding in general:

- The decoding algorithm is usually flexible enough to be used on a variety of channels.
- Randomly chosen convolutional codes can be used.
- A computer with large storage is required.
- In theory, given sufficient redundancy, the decoding error decreases exponentially with the constraint length n'.
- The decoding effort (in terms of computations or storage required) is a random variable without an upper bound, although the expected decoding effort is bounded.

constant-weight codes

A constant-weight code consists of all n-vectors of a certain fixed weight (number of 1's) w. Since two n-vectors of weight w do not always result in a vector sum of the same weight, such codes are generally not linear codes. Constant-weight codes are useful in asymmetric channels in which errors of one polarity dominate, since such errors always change the weight of the code vectors and, thus, can be detected.[72,73] The minimum Hamming distance between any two code vectors is two. Therefore, any combination of an odd number of errors can also be detected. When $n = 2w$, the code vectors can be used directly to specify the exact bipolar signal sequences to be used in the channel. Such signals would contain no dc component. This is a desirable feature, since it is common for a channel frequency characteristic to assume a zero value at the zero frequency.

arithmetic codes

Arithmetic codes have been proposed for use with computers to control errors that occur in arithmetic operations as well as in transmission and storage.[62] Code words are considered integer numbers, and ordinary arithmetic operations apply. There is a generator A, similar to that of cyclic polynomial codes, and the code words are all integer multiples of A, within a certain range of n digits. For binary arithmetic codes, the number of redundant digits is the smallest integer $r \geq \log_2 A$. Such a code is linear with respect to arithmetic operations, i.e., $AN_1 + AN_2 = A(N_1 + N_2) = AN_3$.

An error in arithmetic code is defined by subtracting the transmitted code "number" from the received number arithmetically. Because of carries, a "single" arithmetic error may appear as a burst of errors in the vector representation.

A single-error-detecting arithmetic code can be obtained by letting $A = 3$. Since a single error must assume a magnitude of the form $\pm 2^i$, no single error can change one code number to another because code numbers must differ by a multiple of three. Such a multiple can never assume the form $\pm 2^i$. The arithmetic code length can be arbitrary.

For single-error correction, the residues of $\pm 2^i$ ($i = 0, \cdots, n - 1$) modulo A (which are similar to the syndromes in polynomial codes) must be distinct. For example, with $A = 19$, we have the following residues:

$$\left.\begin{aligned} &2^0 \equiv 1,\ 2^1 \equiv 2,\ 2^2 \equiv 4,\ 2^3 \equiv 8,\ 2^4 \equiv 16, \\ &2^5 \equiv 13,\ 2^6 \equiv 7,\ 2^7 \equiv 14,\ 2^8 \equiv 9,\ -2^0 \equiv 18, \\ &-2^1 \equiv 17,\ -2^2 \equiv 15,\ -2^3 \equiv 11,\ -2^4 \equiv 3, \\ &-2^5 \equiv 6,\ -2^6 \equiv 12,\ -2^7 \equiv 5,\ -2^8 \equiv 10 \end{aligned}\right\} \text{modulo } 19$$

The code length is nine digits with five redundancy digits. Despite their attractive features for error control in computer-communication systems, there are few known classes of arithmetic codes. However, some encouraging results in the theory of arithmetic codes for multiple-error correction have recently been obtained.[74]

Appendix E: Cyclic redundancy checking

Operation of the CRC in IBM 2400-series magnetic tapes is now illustrated. If $a_i(x)$, $i = 0, \cdots, 8$, indicates the message polynomials on the ith track, then the CRC character contains the residue

$$c(x) = c_0 + c_1x + \cdots + c_8x^8 \equiv \sum_{i=0}^{8} x^{i+1}a_i(x) \quad \text{modulo} \quad g(x)$$

where

$$g(x) = x^9 + x^6 + x^5 + x^4 + x^3 + 1$$

The residue $c(x)$ can be obtained from the division circuit shown in Figure 13. Comparing Figure 13 with Figure 4a, it is clear that the contribution of $xa_0(x)$ in the residue $c(x)$ is $c^{(0)}(x) \equiv xa_0(x)$ modulo $g(x)$. Similarly, because of the successively advanced input points, the contribution of $xa_i(x)$ in $c(x)$ is $c^{(i)}(x) \equiv x^{i+1}a_i(x)$ modulo $g(x)$. The complete residue is $\sum_{i=0}^{8} c^{(i)}(x) = c(x)$.

The effect of the premultiplication by x at all inputs is equivalent to an additional shift after the last digits in $a_i(x)$ are in the division circuit. The CRC-coded message in the ith track is $m_i(x) = xa_i(x) + c_i$. When the error-free coded messages in nine tracks are fed into the division circuit of Figure 13, the shift-register contents correspond to the residue of

$$\sum_{i=0}^{8} x^i m_i(x) = \sum_{i=0}^{8} x^{i+1}a_i(x) + \sum_{i=0}^{8} x^i c_i$$

$$\equiv c(x) + c(x) = 0 \quad \text{modulo} \quad g(x)$$

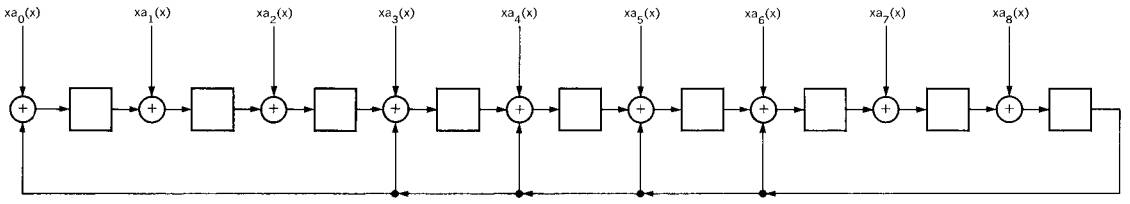
If an error $e_i(x)$ occurs in the ith track, then the register contents correspond to

$$s(x) \equiv x^i e_i(x) \qquad \text{modulo } g(x) \tag{8}$$

which is not zero if $g(x)$ does not divide $e_i(x)$.

To determine the track i, feed the VRC error sequence into a second division circuit similar to the one shown in Figure 13, but with the input point corresponding to that of the eighth track. An error appears in the VRC sequence when $e(x)$ occurs

Figure 13 Nine-bit division circuit



in any single track. Therefore, the register contents of the second division circuit correspond to

$$s'(x) \equiv x^8 e_i(x) \qquad \text{modulo } g(x)$$

Making $8 - i$ additional shifts in the original divider after $s(x)$ is obtained (and referring to Equation 8), the register contents are

$$x^{8-i}s(x) = x^{8-i}x^i e_i(x) \equiv s'(x) \qquad \text{modulo } g(x)$$

matching the register contents of the second divider. After the error track is determined by shifting and matching the divider register contents, the track is reread with the VRC error sequence added to the message. Many errors not correctable by the above procedure, including any single-track error $e_i(x)$ that is divisible by $g(x)$ and any combination of odd numbers of bit errors, are still detectable.

CITED REFERENCES AND FOOTNOTES

1. C. E. Shannon, "A mathematical theory of communications," *Bell System Technical Journal* **27**, 379–423, 623–656 (1948).
2. M. J. E. Golay, "Notes on digital coding," *Proceedings of the IRE* **37**, 657 (1949).
3. W. R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal* **29**, 147–160 (1950).
4. D. Slepian, "A class of binary signalling alphabets," *Bell System Technical Journal* **35**, 203–234 (1956).
5. E. Prange, *Cyclic Error Correcting Codes in Two Symbols*, U. S. Air Force Cambridge Research Center, Technical Report AFCRC-TN-58-156, Bedford, Massachusetts (1957).
6. W. W. Peterson, *Error-Correcting Codes*, The MIT Press and John Wiley and Sons, Inc. (1961).
7. R. W. Lucky, J. Salz, and E. J. Weldon, Jr., *Principles of Data Communication*, McGraw-Hill Book Company, New York, New York (1968).
8. D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE* **40**, No. 9, 1098–1101 (September 1952).
9. J. M. Berger and B. Mandelbrot, "A new model for error clustering in telephone circuits," *IBM Journal of Research and Development* **7**, No. 3, 224–236 (July 1963).
10. The addition $(+)$ and the multiplication $(\cdot)$ in a binary field are defined by the following equations: $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$, $0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0$, and $1 \cdot 1 = 1$.
11. D. A. Huffman, "The synthesis of linear sequential coding networks," *Information Theory*, 77–95, Academic Press, New York, New York (1956).

12. C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell System Technical Journal* **38**, No. 3, 611–656 (1959).
13. G. L. Turin, "Signal design for sequential detection systems with feedback," *IEEE Transactions on Information Theory* **IT-11**, 401–408 (July 1965).
14. J. P. Schalkwijk and T. Kailath, "A coding scheme for additive noise channels with feedback—Part I: no bandwidth constraint," *IEEE Transactions on Information Theory* **IT-12**, 172–188 (April 1966).
15. A. H. Frey, Jr. and R. J. Benice, "An analysis of retransmission systems," *IEEE Transactions on Communication Technology* **COM-12**, 135–146 (1964).
16. G. D. Forney, Jr., "On decoding BCH codes," *IEEE Transactions on Information Theory* **IT-11**, No. 4, 549–557 (October 1965).
17. G. D. Forney, "Generalized minimum distance coding," *IEEE Transactions on Information Theory* **IT-12**, No. 2, 125–131 (April 1966).
18. A. H. Frey, Jr., "Adaptive decoding without feedback," *Proceedings of the International Symposium on Information Theory* (Athens, September 1967).
19. D. T. Tang, "Dual codes as variable redundancy codes," *IEEE International Convention Record* **13**, Part 7, 220–226 (March 1965).
20. D. T. Tang and R. T. Chien, "Cyclic product codes and their implementation," *Information and Control* **9**, No. 2, 196–209 (April 1966).
21. J. L. Eisenbies, "Conventions for digital data communication design," *IBM Systems Journal* **6**, No. 4, 267–302 (1967).
22. R. T. Chien and D. Frazer, "An application of coding theory to document retrieval," *IEEE Transactions on Information Theory* **IT-12**, No. 2, 92–96 (April 1966).
23. C. T. Abraham, S. P. Ghosh, and D. K. Ray-Chaudhuri, "File organization schemes based on finite geometries," *Information and Control* **12**, No. 2, 143–163 (February 1968).
24. G. Constantine, Jr., "A load-sharing matrix switch," *IBM Journal of Research and Development* **2**, No. 3, 204–211 (July 1958).
25. R. T. Chien, "A class of optimal noiseless load-sharing matrix switches," *IBM Journal of Research and Development* **4**, No. 4, 414–417 (October 1960).
26. N. M. Abramson, "A class of systematic codes for non-independent errors," *IRE Transactions on Information Theory* **IT-5**, No. 4, 150–157 (December 1959).
27. P. Fire, *A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors*, Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Laboratory, Mountain View, California (1959).
28. B. Elspas and R. A. Short, "A note on optimum burst-error-correcting codes, *IRE Transactions on Information Theory* **IT-8**, No. 1, 39–42 (January 1962).
29. R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control* **3**, 68–79 (1960).
30. R. C. Bose and D. K. Ray-Chaudhuri, "Further results on error correcting binary group codes," *Information and Control* **3**, 279–290 (1960).
31. A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres* **2**, 147–156 (1959).
32. W. W. Peterson, *op. cit.*, 141–142.
33. A. A. Albert, *Fundamental Concepts of Higher Algebra*, 143 (Theorem 25), The University of Chicago Press (1956).
34. W. W. Peterson, *op. cit.*, 254–270.
35. M. J. E. Golay, "Notes on the penny-weighing problem, lossless symbol coding with nonprimes, etc.," *IRE Transactions on Information Theory* **IT-4**, No. 3, 103–109 (September 1958).
36. V. Y. Lum, *On Bose–Chaudhuri–Hocquenghem Codes Over GF(q)*, Report R-306, University of Illinois, Urbana, Illinois (July 1966).
37. R. T. Chien, "Cyclic decoding procedures for Bose–Chaudhuri–Hoc-

quenghem codes, *IEEE Transactions on Information Theory* **IT-10**, No. 4, 357–363 (October 1964).
38. E. R. Berlecamp, *Algebraic Coding Theory*, Chapters 7 and 10, McGraw-Hill Book Company, New York, New York (1968).
39. J. L. Massey, "Feedback shift-register synthesis and BCH decoding," submitted for publication in the *IEEE Transactions on Information Theory*.
40. D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Transactions on Electronic Computers*, **EC-3**, No. 3, 6–12 (September 1954).
41. I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Transactions—Professional Group on Information Theory* **PGIT-4**, 38–49 (September 1954).
42. T. Kasami, S. Lin, and W. Peterson, "New generalizations of the Reed-Muller codes part I: primative codes," *IEEE Transactions on Information Theory* **IT-14**, No. 2, 189–199 (March 1968), and E. J. Weldon, "New generalizations of the Reed-Muller codes part II: nonprimative codes," *IEEE Transactions on Information Theory* **IT-14**, No. 2, 199–205 (March 1968).
43. There are two operations defined in a field, addition and multiplication. If 2^m binary m-tuples are represented by corresponding polynomials, the addition and multiplication of binary polynomials can be taken as field operations, provided that we always reduce a product polynomial of degree m or higher to its residue modulo a fixed, irreducible polynomial of degree m. For a rigorous treatment on the theory of finite fields, see Chapter 6 in Reference 6.
44. I. S. Reed and G. Soloman, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics* **8**, No. 2, 300–304 (1960).
45. I. B. Oldham, R. T. Chien, and D. T. Tang, "Error detection and correction in a photo-digital memory system," *IBM Journal of Research and Development* **12**, No. 6 (November 1968).
46. K. Y. Sih and M. Y. Hsiao, "Cyclic codes in multiple channel parallel systems," *IEEE Transactions on Electronic Computers* **EC-15**, No. 6, 927–930 (December 1966).
47. A. Gill, "On the series-to-parallel transformations of linear sequential circuits," *IEEE Transactions on Electronic Computers* **EC-15**, No. 1, 107–108 (February 1966).
48. T. G. Birdsall and M. P. Ristenblatt, *Introduction to Linear Shift-Register Generated Sequences*, EDG Technical Report No. 90, University of Michigan Research Institute (1958).
49. P. Elias, "Error-free coding," *IRE Transactions of the Professional Group on Information Theory* **PGIT-4**, 29–37 (1954).
50. W. W. Peterson, *op. cit.*, 194–195.
51. T. Kasami, "Optimum shortened cyclic codes for burst-error-correction," *IEEE Transactions on Information Theory* **IT-9**, No. 2, 105–109 (April 1963).
52. L. D. Rudolph, "A class of majority logic decodable codes," *IEEE Transactions on Information Theory* **IT-13**, No. 12, 305–306 (April 1967).
53. J. Massey, *Threshold Decoding*, The MIT Press, Cambridge, Massachusetts (1963).
54. E. J. Weldon, Jr., "Difference-set cyclic codes," *Bell System Technical Journal* **45**, No. 7, 1045–1057 (1966).
55. D. K. Chow, *A Geometric Approach to Coding Theory with Application to Information Retrieval*, Report R-368 (Doctoral Dissertation), University of Illinois, Urbana, Illinois.
56. R. H. Barker, "Group synchronizing of binary digital systems," *Communication Theory*, W. Jackson, Editor, 273–287, Academic Press, Inc., New York, New York (1953).

57. S. Y. Tong, "Synchronization recovery techniques for binary cyclic codes," *Bell System Technical Journal* **45**, 561–596 (1966).
58. R. C. Bose and J. G. Caldwell, "Synchronizable error-correcting codes," *Information and Control* **10**, 616–630 (1967).
59. E. J. Weldon, Jr., "A note on synchronization recovery with extended cyclic codes," *Proceedings of the First Annual Princeton Conference on Information Sciences and Systems*, Department of Electrical Engineering, 233, Princeton, New Jersey (1967).
60. F. F. Sellers, "Bit loss and gain correction code," *IEEE Transactions on Information Theory* **IT-8**, No. 1, 35–38 (January 1962).
61. J. D. Ullman, "Near-optimal, single-synchronization-error-correcting code," *IEEE Transactions on Information Theory* **IT-12**, No. 4, 418–425 (October 1966).
62. D. T. Brown, "Error detecting and correcting binary codes for arithmetic operations," *IRE Transactions on Electronic Computers* **EC-9**, 333–337 (1960).
63. A. D. Wyner and R. B. Ash, "Analysis of recurrent codes," *IEEE Transactions on Information Theory* **IT-11**, No. 3, 143–156 (July 1963).
64. D. W. Hagelbarger, "Recurrent codes: easily mechanized, burst-correcting, binary codes," *Bell System Technical Journal* **38**, 969–984 (1959).
65. E. R. Berlekamp, "Note on recurrent codes," *IEEE Transactions on Information Theory* **IT-10**, No. 3, 257–259 (July 1964).
66. J. J. Bussgang, "Some properties of binary convolutional code generators," *IEEE Transactions on Information Theory* **IT-11**, No. 1, 90–100 (January 1965).
67. J. P. Robinson, "An upper bound on minimal distance of convolutional code," *IEEE Transactions on Information Theory* **IT-11**, No. 4, 567–571 (October 1965).
68. C. V. Freiman and J. P. Robinson, "A comparison of block and recurrent codes for the correction of independent errors," *IEEE Transactions on Information Theory* **IT-11**, No. 3, 445–449 (July 1965).
69. J. M. Wozencraft, *Sequential Decoding for Reliable Communication*, Massachusetts Institute of Technology, Research Laboratory for Electronics, Technical Report TR325 (1957).
70. R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory* **IT-9**, No. 2, 64–74 (January 1963).
71. F. Jelinek, *Probabilistic Information Theory—Discrete and Memoryless Models*, McGraw-Hill Book Company (1968).
72. J. M. Berger, "A note on burst error detection codes for asymmetric channels," *Information and Control* **4**, No. 3, 68–73 (March 1961).
73. C. V. Freiman, "Optimal error detection codes for completely asymmetric binary channels," *Information and Control* **5**, No. 1 (March 1961).
74. R. T. Chien, S. J. Hong, and F. P. Preparata, "Some contributions to the theory of arithmetic codes," *Proceedings of the Hawaii International Conference on Systems Sciences*, Honolulu, Hawaii (1968).