Lecture

Technologies and Services on Digital Broadcasting (7)

Error Correction Methods

1. Digital Transmission and Error Correction

Error correction technology, with which the errors that occur during transmission are corrected, is essential when high-quality digital video and audio are to be provided in digital broadcasting The principle of error correction is that the improvement in the reliability of the received information as a result of error correction will more than compensate for the increase in data caused by the addition of redundant data in the coding process.

The channel capacity C (bps) for a digital transmission channel of bandwidth B (Hz) on which errors are caused by random noise can be given by the following equation (Shannon's theorem)[1]:

$$C = B\cdot\log_2\left(1+\frac{S}{N}\right) = B\cdot\log_2\left(1+\frac{S}{N_0B}\right) \quad (1)$$

where S is signal power (W), N is noise power (W), and N_0 is noise power spectral density (W/Hz). Shannon's theorem says that coding for which data can be transmitted without errors will exist so long as the transmission rate does not exceed this channel capacity. Figure 1 shows the change in channel capacity as a function of bandwidth. Here, if B were unlimited, C would asymptotically approach the constant value C_∞ given below.

$$C_\infty = \lim_{B\to\infty} B\cdot\log_2\left(1+\frac{S}{N_0B}\right) = \left(\frac{S}{N_0}\right)\cdot\log_2 e \quad (2)$$

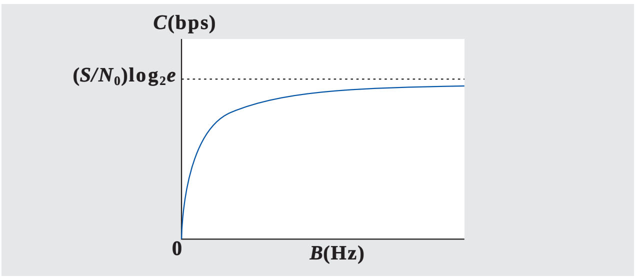


Figure 1

When the transmission bandwidth is limited, the transmission rate can be increased without incurring errors only if the transmit power is increased to improve S/N_0. Conversely, if power is limited, channel capacity can be increased by increasing the transmission bandwidth. However, even if the bandwidth can be extended, the error-free transmission rate will reach a limiting value determined by the S/N_0 of the transmission channel. Accordingly, the issue in error correction is the extent to which the expansion of bandwidth due to coding can efficiently raise the reliability of information and bring the error-free transmission rate close to Shannon's channel capacity.[2]

Error correcting codes can be distinguished into two types: block code and convolutional code. A block code generates a single codeword from a fixed-length piece of information, called a block, whereas a convolutional code generates a code series from multiple blocks of information.

2. Block Code

An important type of block code is "cyclic code" in which the cyclic shifting of codeword components also produces a codeword. One feature of cyclic code is that coding and decoding can be achieved relatively easily using a shift register. The following provides an explanation of block codes, with cyclic code presumed.

2.1 Code expressions

A block code is generated by adding parity check bits for detecting and correcting errors in information bits. A block code featuring a code length of n bits and k bits of information is denoted as an (n, k) code. Here, the coding rate, which indicates the efficiency of coding, is defined as k/n. A block code can be represented by a matrix or polynomial, as explained below.

(1) Matrix representation

We consider the coding of a set of information (called here just information) i of length k bits to a codeword c of length n bits.

$$i = (i_{k-1}, i_{k-2}, \cdots, i_0) \quad (3)$$

$$c = (c_{n-1}, c_{n-2}, \cdots\cdots, c_0) \quad (4)$$

The components of codeword c are expressed by the following equation.

$$c_i = \begin{cases} i_i & (0 \le i \le k-1) \\ \sum_{j=1}^{k} h_{ij} i_j & (k \le i \le n-1) \end{cases} \quad (5)$$

Here, h_{ij} indicates the coefficients (components of the matrix expression) in the linear sums used for calculating parity check bits for the information. The value of h_{ij} is 0 or 1. All sums here are Exclusive OR operations.

Denoting the number of parity bits as m (= n-k), the matrix relating the codeword and information is as follows.

$$(c_{n-1}, c_{n-2}, \cdots, c_0) = (i_{k-1}, i_{k-2}, \cdots, i_0) \begin{pmatrix} 1 & 0 & \cdots & 0 & h_{11} & \cdots & h_{m1} \\ 0 & 1 & \cdots & 0 & h_{12} & \cdots & h_{m2} \\ \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_{1k} & \cdots & h_{mk} \end{pmatrix} \quad (6)$$

This $k \times n$ matrix consisting of a $k \times k$ unit matrix and coefficients h_{ij} is called a generator matrix. A matrix H of order $m \times k$ is created by permutating the components of the generator matrix as follows.

$$H = \begin{pmatrix} h_{11} & \cdots & h_{1k} & 1 & 0 & \cdots & 0 \\ h_{21} & \cdots & h_{2k} & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{m1} & \cdots & h_{mk} & 0 & 0 & \cdots & 1 \end{pmatrix} \quad (7)$$

H is called a parity check matrix.

The decoding side calculates a "syndrome" by multiplying the received words by the transpose matrix of H. The size of the syndrome is only m bits, the same as the number of parity check bits. Errors can be inferred from the pattern of this m-bit syndrome since the pattern has a one-to-one correspondence with the error locations.

(2) Polynomial representation

If the length of a block code is long, the matrix representation can be difficult to handle, and a polynomial expression may be more convenient.

Letting each bit of the information and codeword be a coefficient of a polynomial, we get the following equations.

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \cdots + i_1 x + i_0 \quad (8)$$

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots\ldots + c_1 x + c_0 \quad (9)$$

Equations (8) and (9) are related as follows using a polynomial $g(x)$ of degree m (= n-k).

$$\begin{aligned} c(x) &= i(x)g(x) \\ &= i(x)\left(g_m x^m + g_{m-1}x^{m-1} + \cdots + g_1 x + g_0\right) \end{aligned} \quad (10)$$

$g(x)$ is called a generator polynomial.

However, simply multiplying the information polynomial $i(x)$ by $g(x)$ does not preserve the information's original form within the codeword. The following operation is therefore necessary.

First, the product of the information polynomial $i(x)$ and x^m can be expressed as follows, where $A(x)$ and $B(x)$ are the quotient polynomial and residual polynomial, respectively, obtained by dividing that product by $g(x)$.

$$i(x) \cdot x^m = A(x) \cdot g(x) + B(x) \quad (11)$$

Since $g(x)$ is of degree m, $A(x)$ and $B(x)$ become polynomials of degrees k-1 or less and m-1 or less, respectively, and can be denoted as follows.

$$A(x) = A_{k-1}x^{k-1} + A_{k-2}x^{k-2} + \cdots + A_1 x + A_0 \quad (12)$$

$$B(x) = B_{m-1}x^{m-1} + B_{m-2}x^{m-2} + \cdots + B_1 x + B_0 \quad (13)$$

The codeword polynomial $c(x)$ becomes

$$\begin{aligned} c(x) &= A(x) \cdot g(x) \\ &= i(x) \cdot x^m + B(x) \\ &= i_{n-1}x^{n-1} + i_{n-2}x^{n-2} + \cdots + i_1 x^{m+1} + i_0 x^m + B_{m-1}x^{m-1} \\ &\quad + B_{m-2}x^{m-2} + \cdots + B_1 x + B_0 \end{aligned} \quad (14)$$

As shown by Eq. (14), the codeword polynomial has the information in its original form in k bits of the higher degree terms. This kind of code is called a "systematic code".

The codeword polynomial has the property of being divisible by the generator polynomial. This means that a received word in which an error has been added to the codeword cannot be divided evenly by the generator polynomial. The residual polynomial obtained by dividing the received word polynomial by the generator polynomial becomes the syndrome polynomial $s(x)$ as a result. The pattern of this syndrome polynomial has a one-to-one relationship with that of the errors and can therefore be used to correct them.

2.2 Coding and decoding

Figure 2 shows coding and decoding configurations for a cyclic code using shift registers.

In the coding circuit of Fig. 2(a), *SW1* is initially set to its lower position so that the k bits of information are output in their original form. *SW2* is set in the closed

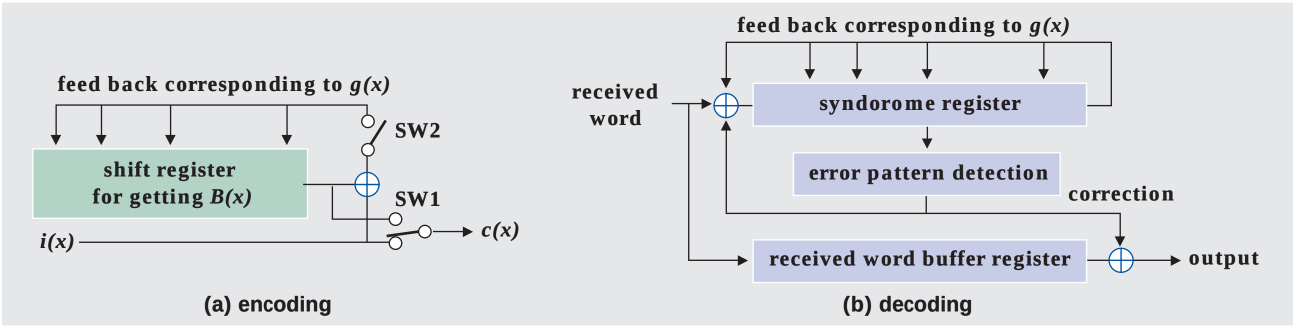

feed back corresponding to g(x)
shift register for getting B(x)
SW2
SW1
i(x)
c(x)
(a) encoding
feed back corresponding to g(x)
received word
syndrome register
error pattern detection
correction
received word buffer register
output
(b) decoding

Figure 2: Coding and decoding configurations for a cyclic code

position to perform shifting while receiving feedback corresponding to the generator polynomial. This shifting determines the residual polynomial $B(x)$ obtained by dividing the product of $i(x)$ and x^m by $g(x)$. The coefficients of $B(x)$ are left in the shift register after the above k bits have been output. At this point in time, *SW1* is set to its upper position, *SW2* is set in an open position, and the coefficients of $B(x)$ are output in order from the shift register to complete the codeword.

Next, in the decoding circuit of Fig. 2(b), the received word is simultaneously input to the buffer register and the syndrome register. The latter register is used to calculate the residual polynomial obtained by dividing the received word by the generator polynomial. Once all bits of the received word have been input, the absence of error in the received word can be inferred if the syndrome register contains only zeros.

In error correction, when the value of the syndrome register corresponds to a pattern in which the leading bit of the received word is erroneous, that bit will be corrected and then output. The feedback bit of the syndrome register will also be corrected at this time. This operation repeats until all data in the received-word buffer register have been output.

2.3 Galois field

The concept of a Galois field is important when thinking about block codes in mathematical terms. A Galois field is a set with a finite number of elements that can be added, subtracted, multiplied, and divided (except by 0). A Galois field of q elements is denoted as $GF(q)$.

Accordingly, a Galois field cannot have an arbitrary (infinite) number of elements. Furthermore, the number of elements q must be of the form p^m, where p is a prime number and m is a positive integer. A field whose number of elements is equal to a prime number is called a prime field and one whose number of elements is equal to a power of a prime is called an extension field of a base prime field. Of particular importance to error correcting code is the prime field $GF(2)$ for the prime number 2 and its extension field $GF(2^m)$.

There are two elements in $GF(2)$: 0 and 1. These elements can be subjected to the four algebraic operations in the same way as an integer with the provision that any result equal to 2 or greater takes on the remainder of that result divided by 2. This means that addition in $GF(2)$ is an Exclusive OR operation.

When deriving an extension field from a certain prime field, the field is expanded by adding on the roots of an irreducible polynomial. This is similar to affixing root i (imaginary unit) of the irreducible polynomial $x^2+1=0$ to a real number when deriving a complex number field from a real number field. For example, in deriving $GF(2^8)$ from $GF(2)$, the 256 elements of $GF(2^8)$ would be $0, 1, \alpha, \alpha^2, \ldots, \alpha^{254}$, where α is a root of the irreducible polynomial $x^8+x^4+x^3+x^2+1=0$.

In a block code, digital data corresponds to the elements of a Galois field, and coding and decoding are performed by operating on those elements.

2.4 Examples of block codes

(1) Bose-Chaudhuri-Hocquenghem (BCH) code[3)4)]

The BCH code is used in a variety of applications due to its especially strong ability for correcting random errors as well as its wide range of code lengths. The following describes a 2-element (binary) BCH code that handles data on $GF(2)$.

Denoting the number of bits that can be corrected within one block as t, then, for any positive integer m, a BCH code with code length n of 2^m-1 and parity length k of mt or less can be generated. Now, given that one element of the Galois field $GF(2^m)$ is α, the minimal polynomial over $GF(2)$ having $\alpha, \alpha^2, \ldots, \alpha^{2t}$ as roots becomes the generator polynomial of the BCH code.

However, because a polynomial over $GF(2)$ also has α^i to the second power, that is, α^{2i}, as a root as well, the roots of the generator polynomial of BCH code may be the t roots $\alpha, \alpha^3, \ldots, \alpha^{2t-1}$ of odd powers only. The generator polynomial of BCH code is consequently defined by the following expression:

$$g(x) = LCM\left[m_1(x), m_3(x), \cdots, m_{2t-1}(x)\right] \qquad (15)$$

where $m_i(x)$ is the minimal polynomial over $GF(2)$ having α^i as a root and $LCM[\ ]$ indicates the least common multiple polynomial. Table 1 shows examples of generator polynomials for 2-element BCH code.

Table 1

n	k	t	generator polynomials
7	4	1	x^3+x+1
15	11	1	x^4+x+1
	7	2	$x^8+x^7+x^6+x^4+1$
31	26	1	x^5+x^2+1
	16	3	$x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$
63	57	1	x^6+x+1
	39	4	$x^{24}+x^{23}+x^{22}+x^{20}+x^{19}+x^{17}+x^{16}+x^{13}+x^{10}+x^9+x^8$ $+x^6+x^5+x^4+x^2+x+1$

(2) Reed-Solomon code [5)]

The Reed-Solomon (RS) code performs data coding and decoding in units of multiple bits (byte). Consequently, while the 2-element BCH code described in the previous section processes data in bit units of $GF(2)$, the extension of this to a $GF(2^m)$ extension field is the RS code.

Given that α is one element of $GF(2^m)$, the generator polynomial of RS code for correcting up to t bytes within a block is defined as follows:

$$g(x) = (x-1)(x-\alpha)\left(x-\alpha^2\right)\cdots\left(x-\alpha^{2t-1}\right) \qquad (16)$$

where code length is 2^m-1 bytes or less and parity length is $2t$ bytes. In contrast to a 2-element code, where polynomial coefficients are the elements of $GF(2)$ and roots are the elements of $GF(2^m)$, in the case of RS code, both polynomial coefficients and roots are the elements of $GF(2^m)$.

Reed Solomon coding is performed in a manner similar to that of 2-element code, that is, the parity check bytes are taken to be the residual polynomial obtained by dividing the product of the information polynomial and x^m by the generator polynomial.

Decoding is performed as a 5-step process of error correction: syndrome calculation, derivation of the error locator polynomial and error evaluator polynomial, error locator calculation, and error value calculation.

1) Syndrome calculation

The first step in determining the location of errors in the received data and their values is to calculate syndromes from the received word, where the number of syndromes is the same as the number of parity bytes. The received word polynomial is described as follows.

$$R(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \cdots + r_1 x + r_0 \quad (17)$$

Substituting the root α^i (i=0,1,...,2t-1) of the generator polynomial in the received word polynomial gives us the syndrome s_i as follows.

$$\begin{aligned} s_i &= R(\alpha^i) \\ &= r_{n-1}\alpha^{(n-1)i} + r_{n-2}\alpha^{(n-2)i} + \cdots + r_1\alpha^i + r_0 \\ &= \left(\left(\cdots\left(r_{n-1}\alpha^i + r_{n-2}\right)\alpha^i + r_{n-3}\right)\cdots\right) + r_1\right)\alpha^i + r_0 \end{aligned} \quad (18)$$

This can be calculated by repeated multiplication and addition, as shown in Fig. 3.

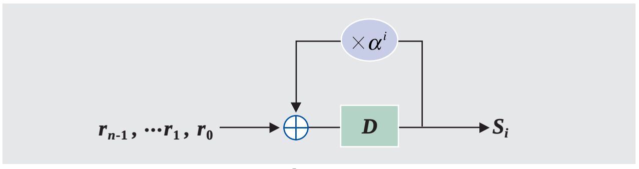


Figure 3

2) Derivation of error locator polynomial and error evaluator polynomial

Assuming that errors in the received word have occurred in the $j_1, j_2,...,j_l$ bytes from high-order bytes of the received word, we consider a polynomial $\sigma(z)$, whose roots are the inverse of roots α^{j1}, α^{j2},...,α^{jl}, and a polynomial $\omega(z)$ that includes errors e_i.

$$\begin{aligned} \sigma(z) &= \left(1-\alpha^{j1}z\right)\left(1-\alpha^{j2}z\right)\cdots\left(1-\alpha^{jl}z\right) \\ &= \sigma_l z^l + \sigma_{l-1}z^{l-1} + \cdots + \sigma_1 z + 1 \end{aligned} \quad (19)$$

$$\begin{aligned} \omega(z) &= \omega_{l-1}z^{l-1} + \cdots + \omega_1 z + \omega_0 \\ &= \sum_{i=1}^{l} e_i \prod_{k\neq i}\left(1-\alpha^{jk}z\right) \end{aligned} \quad (20)$$

The polynomials $\sigma(z)$ and $\omega(z)$ are called the error locator polynomial and error evaluator polynomial, respectively.

The syndrome polynomial $S(z)$ is defined as follows.

$$S(z) = s_{2t-1}z^{2t-1} + \cdots + s_1 z + s_0 \quad (21)$$

$S(z)$ is related to $\sigma(z)$ and $\omega(z)$ in the following way.

$$\sigma(z)S(z) = \omega(z) \bmod z^{2t} \quad (22)$$

The error locator polynomial $\sigma(z)$ and error evaluator polynomial $\omega(z)$ are derived in the process of determining the greatest common divisor polynomial of $S(z)$ and z^{2t} by the Euclidean division algorithm.

The process of deriving $\sigma(z)$ and $\omega(z)$ by the Euclidean division algorithm is shown in Fig. 4.

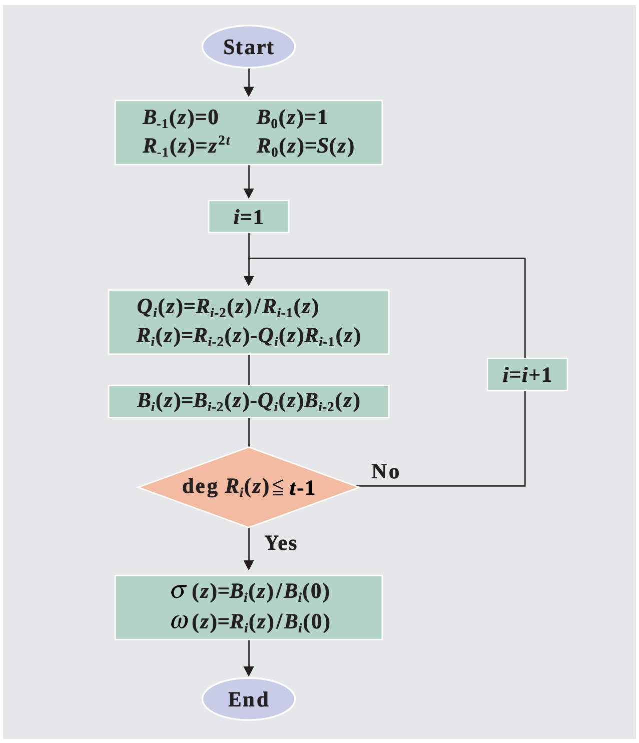

Start
$B_{-1}(z)=0$ $B_0(z)=1$
$R_{-1}(z)=z^{2t}$ $R_0(z)=S(z)$
i=1
$Q_i(z)=R_{i-2}(z)/R_{i-1}(z)$
$R_i(z)=R_{i-2}(z)-Q_i(z)R_{i-1}(z)$
$B_i(z)=B_{i-2}(z)-Q_i(z)B_{i-2}(z)$
$i=i+1$
deg $R_i(z) \leqq t-1$
No
Yes
$\sigma(z)=B_i(z)/B_i(0)$
$\omega(z)=R_i(z)/B_i(0)$
End

Figure 4

3) Error locator calculation

Because the error locator polynomial's roots have degrees inverse to those of the error locations, we can substitute α^{-1}(i=0,...,n-1) in order in the error locator polynomial to get the following equation from which i, that is, the error locations, can be determined by the Chien search algorithm shown in Fig. 5.

$$\sigma(\alpha^{-i}) = 0 \quad (23)$$

The $\sigma(z)$ coefficients σ_1,...,σ_l are set in each register as initial values. Shifting is then performed in sequence, and the number of shifts needed for the sum total of register-outputs to be zero indicates the error location.

4) Error value calculation

Once the error locations $j_1,j_2,...,j_l$ are known, we substitute α^{-j_i} in the error evaluator polynomial $\omega(z)$ and obtain the following expression.

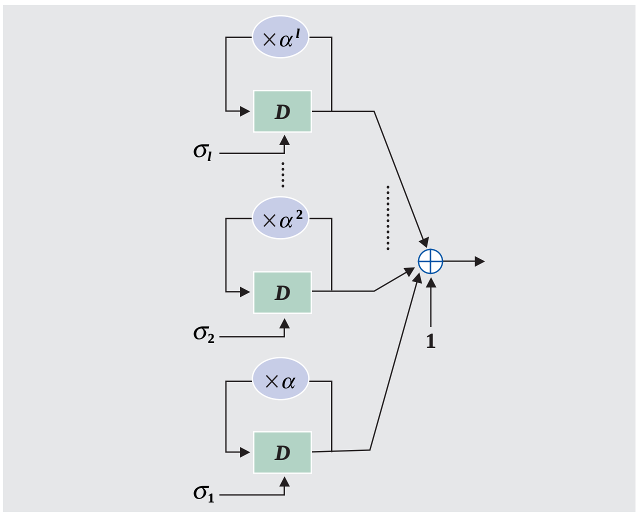


Figure 5

$$\omega\left(\alpha^{-ji}\right)=e_i\prod_{k\neq i}\left(1-\alpha^{jk}\alpha^{-ji}\right) \tag{24}$$

This can be rewritten as follows.

$$e_i=\frac{\omega\left(\alpha^{-ji}\right)}{\prod_{k\neq i}\left(1-\alpha^{jk}\alpha^{-ji}\right)} \tag{25}$$

The error values $e_{j1},e_{j2},\dots,e_{jl}$ at locations $j_1,j_2,\dots,j_l$ can thus be calculated.

5) Error correction

Erroneous bites are corrected by adding error values $e_{j1},e_{j2},\dots,e_{jl}$ to bytes corresponding to error locations $j_1,j_2,\dots,j_l$ of the received word.

Because data in digital broadcasts are transmitted in units of transport stream (TS) packets in MPEG-2 systems, coding will be performed by either (204, 188) RS coding for every 188-byte TS packet or by (207, 187) RS coding for every 187 bytes, excluding one synchronizing byte at the beginning of the TS packet.

(3) Difference-set cyclic code[6)]

In block code error correction, the existence of an error is typically determined by calculating syndromes from received words. However, if the block length is long and the number of parity bits is large, the number of syndrome patterns will be enormous, forcing an increase in the scale of the decoding circuit. In this regard, a decoding method that enables a decoding circuit to be configured in a relatively simple manner is "majority logic decoding."

Majority logic decoding calculates several syndrome sums and examines the results of those sums to determine whether 0 or 1 is in the majority. Error correction is performed if there are more 1's than 0's. Such a syndrome sum is called a "parity check sum."

An example of majority logic decodable code is the "difference-set cyclic code." This code is so named because it is derived from a set of numbers called a "perfect difference set."

A perfect difference set is a set of integers such that

$$D=\{d_1,d_2,\cdots,d_j\leq q(q+1)\} \tag{26}$$

Specifically, given a set D having J (= q +1) integers, we can define difference Δ as follows.

$$\Delta=d_i-d_j \quad i\neq j \tag{27}$$

D is called a perfect difference set if Δ satisfies ① to ③ below.

①The positive values of Δ are all different.

②The negative values of Δ are all different.

③For negative Δ, $q(q+1)+1+\Delta$ does not equal the absolute value of Δ.

The polynomial for which the powers of its terms correspond to the integers of a perfect difference set is expressed as follows.

$$Z(x)=x^{d1}+x^{d2}+\cdots+x^{dj} \tag{28}$$

Furthermore, $g(x)$ of the difference-set cyclic code is given as follows:

$$g(x)=\frac{x^n-1}{GCD\left[Z(x),x^n-1\right]} \tag{29}$$

where n is the code length and GCD[ ] indicates the greatest common divisor polynomial.

Given that J denotes the number of integers in a perfect difference set, a difference-set cyclic code with good efficiency can be derived for J=2^s+1, where s is any positive integer. The block length n is 2^{2s}+2^s+1 bits, parity length k is 3^s+1 bits, and the number of parity check sums is J. In addition, the number of bits t that can be corrected within one block is (J - 1)/2 bits. Table 2 shows the generator polynomials and perfect difference sets of the difference-set cyclic code.

Figure 6 shows an example of majority logic decoding in the difference-set cyclic code.

Referring to the figure, the bits of the received word will be corrected if there are more 1's than 0's among the three parity check sums (A_1, A_2, A_3).

Table 2

s	*n*	*k*	*j*	*t*	generator polynomials	perfect difference sets
1	7	3	3	1	$x^4+x^3+x^2+1$	0,1,3
2	21	11	5	2	$x^{10}+x^7+x^6+x^4+x^2+1$	0,2,7,8,11
3	73	45	9	4	$x^{28}+x^{25}+x^{22}+x^{16}+x^{12}+x^8+x^6+x^4+x^2+1$	0,2,10,24,25,29,36,42,45
4	273	191	17	8	$x^{82}+x^{77}+x^{76}+x^{71}+x^{67}+x^{66}+x^{56}+x^{52}+x^{48}+x^{40}+x^{36}+x^{34}+x^{24}+x^{22}+x^{18}+x^{10}+x^4+1$	0,18,24,46,50,67,103,112,115,126,128,159,166,167,186,196,201
5	1057	813	33	16	(omitted)	0,1,3,7,15,31,54,63,109,127,138,219,255,277,298,338,348,439,452,511,528,555,597,677,697,702,754,792,879,905,924,990,1023

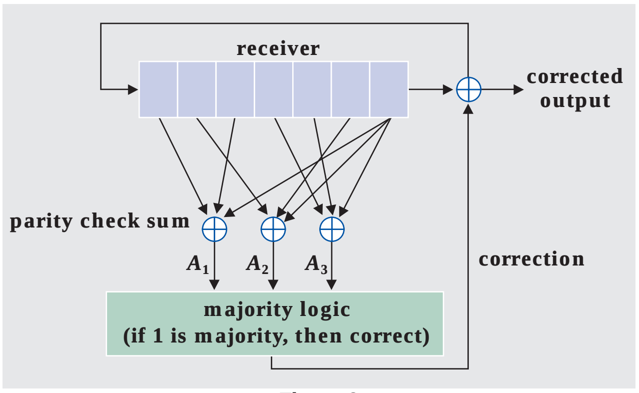


Figure 6

In majority logic decoding, there is also a way to improve error correcting performance beyond conventional levels by devising a particular means of decoding. This decoding method is called "variable threshold majority logic decoding." In ordinary majority logic decoding, error correction is performed if the number of 1's is greater than the threshold value $T=J/2$ for J parity check sums. In variable threshold majority logic decoding, decoding is performed recursively, starting with a threshold value of $T>J/2$ and continuing until $T=J/2$ while decreasing T by 1 each cycle. This decoding method improves error correcting performance by correcting bits with a high possibility of error first.

3. Convolutional Code

In contrast to block code, convolutional code is difficult to handle from a mathematical point of view since it refers to past information bits to determine the codewords. It is nevertheless a useful error correcting code in practice and has found a wide range of application. The following describes convolutional coding and decoding (Viterbi decoding algorithm) on the basis of a simple example.

3.1 Convolutional coding

We consider the simple example of convolutional code shown in Fig. 7. Here, the coding rate is 1/2 because two bits of output data are generated for every one bit of input data. Specifically, two bits of code c_1 and c_2 are generated from one bit of input data a_1 and two bits of previously input data a_2 and a_3 left in the registers. In other words, all output values are systematically determined by the value of one input bit and the values of the two previous bits. The number of registers in the coding circuit is called the constraint length of the convolutional code.

Denoting each of the possible states determined by the values of the two bits left in the coding circuit registers as A (a_2=0, a_3=0), B (a_2=1, a_3=0), C (a_2=0, a_3=1), and D (a_2=1, a_3=1), the transitions from each of these states to a following state due to input bit a_1 can be represented by the state transition diagram shown in Fig. 8.

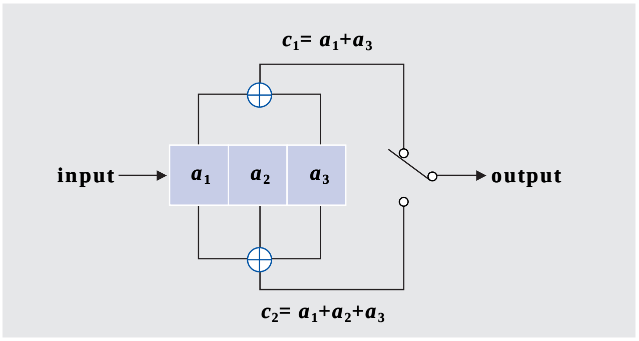


Figure 7

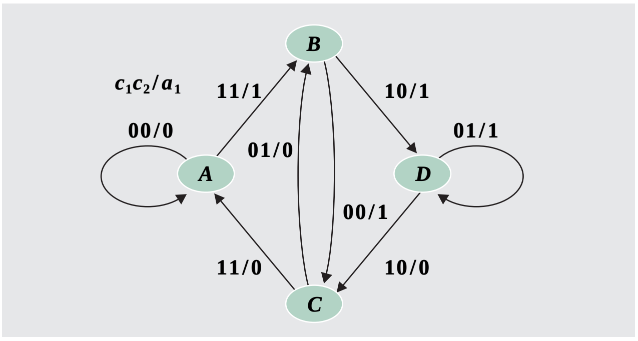


Figure 8

3.2 Viterbi decoding [7)]

Viterbi decoding is one method of decoding convolutional code. It corrects errors by observing the received data series and determining the code series closest to that data series. As an example, we consider the information series (1 0 1 0 1 0 0) input to the coding circuit of Fig. 7. Here, the last two bits of the information series are dummy bits used for terminating decoding, and the code series output from the coding circuit is (11 01 00 01 00 01 11). The received series, moreover, is assumed to be (01 00 00 01 00 01 11) in which two bits, the 1st and 4th bit, are different as a result of errors incurred on the transmission channel.

We explain the decoding algorithm by using the Trellis diagram shown in Fig. 9. This diagram shows the transitions that can be made between the states described above. Starting from state A, we calculate the total number of bits that differ between the code output and the received series for each path of the Trellis diagram. This total number of different bits is called the "Hamming distance." In the figure, the numerical values within the circles for each state indicate the total Hamming distance with respect to the received series up to that point on the path. In this process, two paths enter each state but only the one with the smaller distance survives. This remaining path is called the "survivor." In this example, the above operation is performed until the paths converge to state A as a result of the dummy bits making up the last two bits of the information series.

Consequently, the path with the smallest total Hamming distance ($A \rightarrow B \rightarrow C \rightarrow B \rightarrow C \rightarrow B \rightarrow C \rightarrow A$) survives. The transmitted information series can be inferred to be (1 0 1 0 1 0 0), and the result of decoding is correct despite the occurrence of two transmission errors. Correcting errors in

information	1	0	1	0	1	0	0
convolutional code	11	01	00	01	00	01	11
received code	01	00	00	01	00	01	11

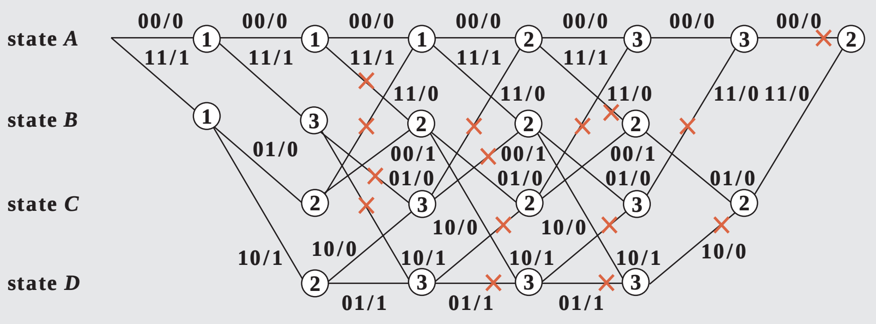


Figure 9

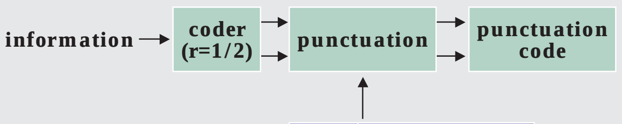
this way, by making an efficient search for the closest code series using a Trellis diagram, is Viterbi decoding.

The above example presented a case in which dummy bits are added to the information series to be transmitted to indicate termination of decoding. In general, however, a path is selected when reaching a path of fixed length on the decoding side without adding dummy bits. In this case, the path length terminating decoding must be five or six times the code constraint length.

Error correcting performance can be improved by using the results of a multi-value judgment (soft decision) made on the received signal as a value for selecting the surviving path instead of the Hamming distance, which is determined from binary data (hard decision). For example, by dividing binary data (0, 1) into 8-level data (011,010,001,000,100,101,110,111) represented in three bits according to the magnitude of that received signal, the reliability of that binary data can be applied to the decoding process. Because Viterbi decoding selects a survivor by quantifying the difference between each path and the received series and comparing their totals, a soft decision is relatively easy to apply. Viterbi decoding with a soft decision improves characteristics by about 2 dB over those of hard-decision decoding.

3.3 Punctured coding

The coding rate of a convolutional code can be modified by a "punctured coding" that systematically removes data from the code series to be transmitted. On the receive side, the removed data can be ignored and decoding can be performed by a decoder identical to the one for the original convolutional code. Figure 10 shows examples of data removing patterns when creating code with coding rates of 3/4 and 7/8 from convolutional code of coding rate 1/2.



rate	eraser pattern
3/4	1 0 1 0 1
7/8	1 1 1 1 0 1 0 1 0 0 0 1 0 1

Figure 10

While punctured coding increases transmission efficiency, the uncertainty in selecting the survivor increases with the amount of removed data, and thus, error correcting performance deteriorates. Figure 11 shows the error correcting characteristics at coding rates of 1/2, 3/4, and 7/8 for convolutional code.

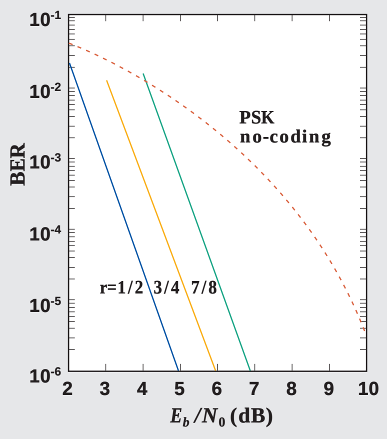


Figure 11

4. Concatenated Coding [8)]

When applying error correcting code to actual digital broadcasting systems, error correction may be applied twice to the digital data to be transmitted in a process called concatenated coding.

In particular, concatenated coding that combines code over $GF(p)$ and code over $GF(p^m)$ has been found to be effective. We give an example of concatenated coding that combines convolutional code over $GF(2)$ and RS code over $GF(2^8)$, as used in actual digital broadcasting systems and

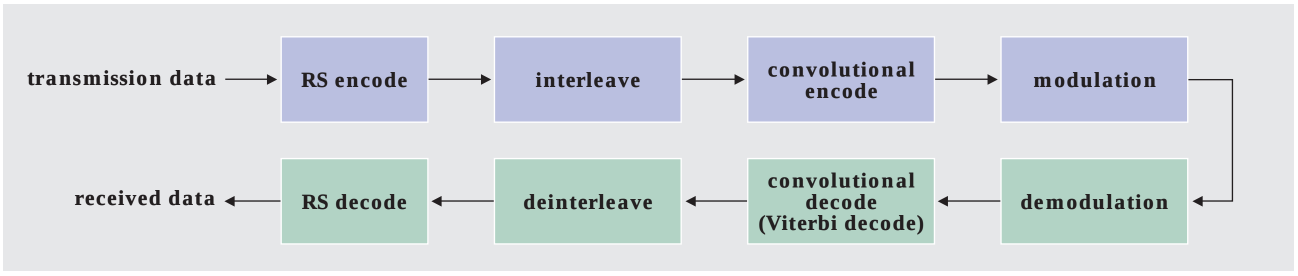


Figure 12

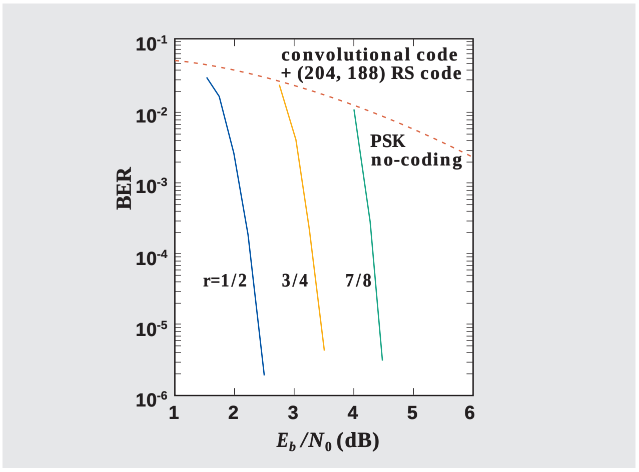


Figure 13

program-material transmission systems.

Figure 12 shows the flow of basic concatenated coding.

The convolutional code is called the "inner code" and the RS code the "outer code." In addition, considering that the error after Viterbi decoding of the inner code generally occurs in burst form, burst-error diffusion processing is performed by adding an interleaving function between the inner and outer codes. In this example, byte interleaving is performed in units of 8 bits, since the outer code is RS code over $GF(2^8)$. Figure 13 shows the bit error rate (BER) characteristics when using a convolutional code with coding rates 1/2, 3/4, and 7/8 as the inner code and (204,188) RS code as the outer code. (Dr. Shigeki Moriyama)

References

1) Shannon, C.E.: A Mathematical Theory of Communication, Bell System Tech. J., Vol. 27, pp. 379-423, pp. 623-656 (1948)
2) Peterson, W.W. and Weldon, E.J.,Jr.: Error-Correcting Codes, 2nd ed., MIT press (1972)
3) Hocquenghem, A.: Codes Correcteurs d'Erreurs, Chiffres, Vol. 2, pp. 147-156 (1959)
4) Bose, R.C. and Ray-Chaudhure, D.K.: On a Class of Error Correcting Binary Group Codes, Inform. and Control, Vol. 3, pp. 68-79 (1960)
5) Reed, I.S. and Solomon, G.: Polynomial Codes over Certain Finite Fields, J.Soc.Indust.Apply.Math., Vol. 8, pp. 300-304 (1960)
6) Weldom, E.J., Jr.: Difference-Set Cyclic Codes, Bell System Tech. J., Vol. 45, pp. 1045-1055 (1996)
7) Viterbi, A.J.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, IEEE Trans. Inform. Theory, Vol. IT-13, pp. 260-269 (1967)
8) Forney, G.D., Jr.: Concatenated Codes, MIT Press, Cambridge, Mass. (1966)