CHAPTER 20

Burst-Error-Correcting Codes

So far we have been concerned primarily with coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise); however, there are communication channels that are affected by disturbances that cause transmission errors to cluster into bursts. In general, codes for correcting random errors are not efficient for correcting burst errors, so it is desirable to design codes specifically for correcting burst errors, namely, burst-error-correcting codes.

20.1 INTRODUCTION

A burst of length l is defined as a vector whose nonzero components are confined to l consecutive digit positions, the first and last of which are nonzero. For example, the error vector $\mathbf{e} = (0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$ is a burst of length 6. A linear code that is capable of correcting all error bursts of length l or less but not all error bursts of length $l + 1$ is called an *l-burst-error-correcting code*, or the code is said to have *burst-error-correcting capability l*.

It is clear that for given code length n and burst-error-correcting capability l, it is desirable to construct an (n, k) code with as small a redundancy $n - k$ as possible. Next, we establish certain restrictions on $n - k$ for given l, or restrictions on l for given $n - k$.

Theorem 20.1 A necessary condition for an (n, k) linear code to be able to correct all burst errors of length l or less is that no burst of length $2l$ or less can be a codeword.

Proof. Suppose that there exists a burst $\mathbf{v}$ of length $2l$ or less as a codeword. This codeword $\mathbf{v}$ can be expressed as a vector sum of two bursts $\mathbf{u}$ and $\mathbf{w}$ of length l or less (except the degenerate case, in which $\mathbf{v}$ is a burst of length 1). Then, $\mathbf{u}$ and $\mathbf{w}$ must be in the same coset of a standard array for this code. If one of these two vectors is used as a coset leader (correctable error pattern), the other will be an uncorrectable error burst. As a result, this code will not be able to correct all error bursts of length l or less. Therefore, in order to correct all error bursts of length l or less, no burst of length $2l$ or less can be a codeword. Q.E.D.

Theorem 20.2 The number of parity-check digits of an (n, k) linear code that has no burst of length b or less as a codeword is at least b (i.e., $n - k \geq b$).

Proof. Consider the vectors whose nonzero components are confined to the first b digit positions. There are a total of 2^b of them. No two such vectors can be in the same coset of a standard array for this code; otherwise, their vector sum, which is a burst of length b or less, would be a codeword. Therefore, these 2^b vectors must be in 2^b distinct cosets. There are a total of 2^{n-k} cosets for an (n, k) code. Thus, $n - k$ must be at least equal to b (i.e., $n - k \geq b$). Q.E.D.

It follows from Theorems 20.1 and 20.2 that there must be a restriction on the number of parity-check digits of an l-burst-error-correcting code.

THEOREM 20.3 The number of parity-check digits of an l-burst-error-correcting code must be at least $2l$; that is,

$$n - k \geq 2l. \tag{20.1}$$

For a given n and k, Theorem 20.3 implies that the burst-error-correcting capability of an (n, k) code is at most $\lfloor (n-k)/2 \rfloor$; that is,

$$l \leq \left\lfloor \frac{n-k}{2} \right\rfloor. \tag{20.2}$$

This is an upper bound on the burst-error-correcting capability of an (n, k) code and is called the *Reiger bound* [5]. Codes that meet the Reiger bound are said to be *optimal*. The ratio

$$z = \frac{2l}{n-k} \tag{20.3}$$

is used as a measure of the *burst-error-correcting efficiency* of a code. An optimal code has burst-error-correcting efficiency equal to 1.

It is possible to show that if an (n, k) code is designed to correct all burst errors of length l or less and simultaneously to detect all burst errors of length $d \geq l$ or less, the number of parity-check digits of the code must be at least $l + d$ (see Problem 20.1).

20.2 DECODING OF SINGLE-BURST-ERROR-CORRECTING CYCLIC CODES

An l-burst-error-correcting cyclic code can most easily be decoded by the error-trapping technique presented in Section 5.7, with a slight variation. Suppose that a codeword $\mathbf{v}(X)$ from an l-burst-error-correcting (n, k) cyclic code is transmitted. Let $\mathbf{r}(X)$ and $\mathbf{e}(X)$ be the received and error vectors, respectively. Let

$$\mathbf{s}(X) = s_0 + s_1 X + \cdots + s_{n-k-1} X^{n-k-1}$$

be the syndrome of $\mathbf{r}(X)$. If the errors in $\mathbf{e}(X)$ are confined to the l high-order parity-check digit positions, $X^{n-k-l}, \cdots, X^{n-k-2}, X^{n-k-1}$, then the l high-order syndrome digits, $s_{n-k-l}, \cdots, s_{n-k-2}, s_{n-k-1}$, match the errors of $\mathbf{e}(X)$, and the $n - k - l$ low-order syndrome digits, $s_0, s_1, \cdots, s_{n-k-l-1}$, are zeros. Suppose that the errors in $\mathbf{e}(X)$ are not confined to the positions $X^{n-k-l}, \cdots, X^{n-k-2}, X^{n-k-1}$ of $\mathbf{r}(X)$ but are confined to l consecutive positions of $\mathbf{r}(X)$ (including the end-around case). Then, after a certain number of cyclic shifts of $\mathbf{r}(X)$, say i cyclic shifts, the errors will be shifted to the positions $X^{n-k-l}, \cdots, X^{n-k-2}, X^{n-k-1}$ of $\mathbf{r}^{(i)}(X)$, the ith shift of $\mathbf{r}(X)$. Let $\mathbf{s}^{(i)}(X)$ be the syndrome of $\mathbf{r}^{(i)}(X)$. Then, the first l high-order digits of $\mathbf{s}^{(i)}(X)$ match the errors at the positions $X^{n-k-l}, \cdots, X^{n-k-2}, X^{n-k-1}$ of $\mathbf{r}^{(i)}(X)$, and the $n - k - l$ low-order digits of $\mathbf{s}^{(i)}(X)$ are zeros. Using these facts, we may trap the errors in the syndrome register by cyclic shifting $\mathbf{r}(X)$.

An error-trapping decoder for an l-burst-correcting cyclic code is shown in Figure 20.1, where the received vector is shifted into the syndrome register from the left end. The decoding procedure is as follows:

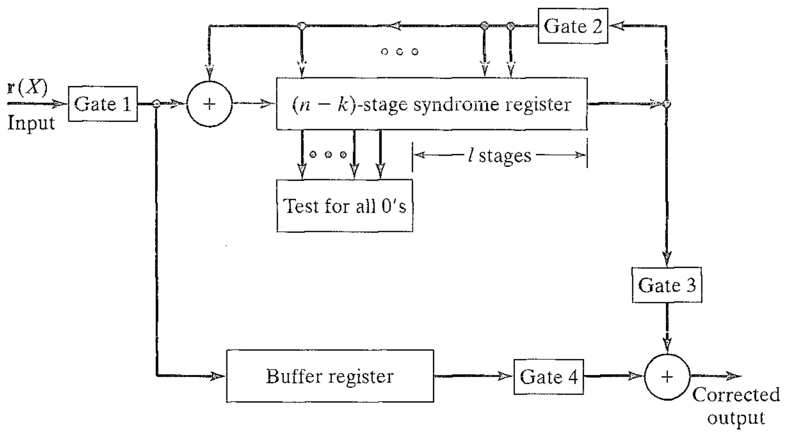

FIGURE 20.1: An error-trapping decoder for burst-error-correcting codes.

Step 1. The received vector $\mathbf{r}(X)$ is shifted into the syndrome and buffer registers simultaneously. (If we do not want to decode the received parity-check digits, the buffer register needs only k stages.) As soon as $\mathbf{r}(X)$ has been shifted into the syndrome register, the syndrome $\mathbf{s}(X)$ is formed.

Step 2. The syndrome register starts to shift with gate 2 on. As soon as its $n - k - l$ leftmost stages contain only zeros, its l rightmost stages contain the burst-error pattern. The error correction begins. There are three cases to be considered.

Step 3. If the $n - k - l$ leftmost stages of the syndrome register contain all zeros after the ith shift for $0 \leq i \leq n - k - l$, the errors of the burst $\mathbf{e}(X)$ are confined to the parity-check positions of $\mathbf{r}(X)$. In this event, the k received information digits in the buffer register are error-free. Gate 4 is then activated, and the k error-free information digits in the buffer are shifted out to the data sink. If the $n - k - l$ leftmost stages of the syndrome register never contain all zeros during the first $n - k - l$ shifts of the syndrome register, the error burst is not confined to the $n - k$ parity-check positions of $\mathbf{r}(X)$.

Step 4. If the $n - k - l$ leftmost stages of the syndrome register contain all zeros after the $(n - k - l + i)$th shift of the syndrome register for $1 \leq i \leq l$, the error burst is confined to positions $X^{n-i}, \cdots, X^{n-1}, X^0, \cdots, X^{l-i-1}$ of $\mathbf{r}(X)$. (This is an end-around burst.) In this event, the $l - i$ digits contained in the $l - i$ rightmost stages of the syndrome register match the errors at the parity-check positions, $X^0, X^1, \cdots, X^{l-i-1}$ of $\mathbf{r}(X)$, and the i digits contained in the next i stages of the syndrome register match the errors at the positions $X^{n-i}, \cdots, X^{n-2}, X^{n-1}$ of $\mathbf{r}(X)$. At this instant, a clock starts to count from $(n-k-l+i+1)$. The syndrome register is then shifted (in step with the clock) with gate 2 turned off. As soon as the clock has counted up to $n - k$, the i rightmost digits in the syndrome register match the errors at the positions $X^{n-i}, \cdots, X^{n-2}, X^{n-1}$ of $\mathbf{r}(X)$. Gates 3 and 4 are then activated. The received information digits are read out of the buffer register and corrected by the error digits shifted out from the syndrome register.

Step 5. If the $n - k - l$ leftmost stages of the syndrome register never contain all zeros by the time that the syndrome register has been shifted $n - k$ times, the received information digits are read out of the buffer register one at a time with gate 4 activated. At the same time, the syndrome register is shifted with gate 2 activated. As soon as the $n - k - l$ leftmost stages of the syndrome register contain all zeros, the digits in the l rightmost stages of the syndrome register match the errors in the next l received information digits to come out of the buffer register. Gate 3 is then activated, and the erroneous information digits are corrected by the digits coming out from the syndrome register with gate 2 disabled.

If the $n - k - l$ leftmost stages of the syndrome register never contain all zeros by the time the k information digits have been read out of the buffer, an uncorrectable burst of errors has been detected. With the decoder just described, the decoding process takes $2n$ clock cycles; the first n clock cycles are required for syndrome computation, and the next n clock cycles are needed for error trapping and error correction. The n clock cycles for syndrome computation are concurrent with the reception of the received vector from the channel; no time delay occurs in this operation. The second n clock cycles for error trapping and correction represent decoding delay.

In this decoder the received vector is shifted into the syndrome register from the left end. If the received vector is shifted into the syndrome register from the right end, the decoding operation will be slightly different (see Problem 20.2).

This decoder corrects only burst errors of length l or less. The number of these burst-error patterns is $n2^{l-1}$, which for large n, is only a small fraction of 2^{n-k} correctable error patterns (coset leaders). It is possible to modify the decoder in such a way that it corrects all the correctable burst errors of length $n - k$ or less. That is, besides correcting all the bursts of length l or less, the decoder also corrects those bursts of length $l + 1$ to $n - k$ that are used as coset leaders. This modified decoder operates as follows. The entire received vector is first shifted into the syndrome register. Before performing the error correction, the syndrome register is cyclically shifted n times (with feedback connections operative). During this cycling the length b of the shortest burst that appears in the b rightmost stages of the syndrome register is recorded by a counter. This burst is assumed to be the error burst added by the channel. Having completed these precorrection shifts, the decoder begins its correction process. The syndrome register starts to shift again. As soon as the shortest burst reappears in the b rightmost stages of the syndrome register, the decoder starts to make corrections as described earlier. This decoding is an optimum decoding for burst-error-correcting codes that was proposed by Gallager [20].

20.3 SINGLE-BURST-ERROR-CORRECTING CODES

20.3.1 Fire Codes

Fire codes were the first class of cyclic codes constructed systematically for correcting burst errors. Let $\mathbf{p}(X)$ be an irreducible polynomial of degree m over $GF(2)$. Let

ρ be the smallest integer such that $\mathbb{p}(X)$ divides $X^\rho + 1$. The integer ρ is called the *period* of $\mathbb{p}(X)$. Let l be a positive integer such that $l \leq m$, and $2l - 1$ is not divisible by ρ. An l-burst-error-correcting Fire code is generated by the following polynomial:

$$\mathbb{g}(X) = (X^{2l-1} + 1)\mathbb{p}(X). \tag{20.4}$$

The length n of this code is the least common multiple (LCM) of $2l - 1$ and the period ρ of $\mathbb{p}(X)$, that is,

$$n = \text{LCM}(2l - 1, \rho). \tag{20.5}$$

The number of parity-check digits of this code is $m + 2l - 1$. Note that the two factors $X^{2l-1} + 1$ and $\mathbb{p}(X)$ of $\mathbb{g}(X)$ are relatively prime.

EXAMPLE 20.1

Consider the irreducible polynomial $\mathbb{p}(X) = 1 + X^2 + X^5$. Because $\mathbb{p}(X)$ is a primitive polynomial, its period is $\rho = 2^5 - 1 = 31$. Let $l = 5$. Clearly, 31 does not divide $2l - 1 = 9$. The Fire code generated by

$$\begin{aligned}\mathbb{g}(X) &= (X^9 + 1)(1 + X^2 + X^5)\\ &= 1 + X^2 + X^5 + X^9 + X^{11} + X^{14}\end{aligned}$$

has length $n = \text{LCM}(9, 31) = 279$. Therefore, it is a (279, 265) cyclic code that is capable of correcting any burst error of length 5 or less.

To prove that the Fire code generated by the polynomial of (20.4) is capable of correcting any burst of length l or less, it is sufficient to show that all bursts of length l or less are in different cosets of the code. Thus, they can be used as coset leaders and form correctable error patterns. The proof is left as a problem (see Problem 20.3).

Fire codes can be decoded with the error-trapping circuit shown in Figure 20.1. The error-trapping decoder for the (279, 265) Fire code considered in Example 20.1 is shown in Figure 20.2.

In a data transmission (or storage) system, if the receiver has some computation capability, a fast decoder for Fire codes may be implemented. Consider a Fire code with generator polynomial $\mathbb{g}(X) = (X^{2l-1} + 1)\mathbb{p}(X)$, where $2l - 1$ and the period ρ of $\mathbb{p}(X)$ are relatively prime. Let $\mathbb{r}(X)$ be the received polynomial. Let $\mathbb{s}_1(X)$ and $\mathbb{s}_2(X)$ be the remainders resulting from dividing $\mathbb{r}(X)$ by $X^{2l-1} + 1$ and $\mathbb{p}(X)$, respectively. Then, we may take

$$[\mathbb{s}_1(X), \mathbb{s}_2(X)]$$

as a syndrome of $\mathbb{r}(X)$. We can readily see that $\mathbb{s}_1(X) = \mathbb{s}_2(X) = 0$ if and only if $\mathbb{r}(X)$ is a code polynomial. If $\mathbb{r}(X)$ contains a nonzero error burst of length l or less, we must have $\mathbb{s}_1(X) \neq 0$ and $\mathbb{s}_2(X) \neq 0$. If $\mathbb{s}_1(X) = 0$ and $\mathbb{s}_2(X) \neq 0$ [or $\mathbb{s}_1(X) \neq 0$ and $\mathbb{s}_2(X) = 0$], then $\mathbb{r}(X)$ must contain a detectable but uncorrectable error burst of length greater than l.

Now, consider an error-trapping decoder as shown in Figure 20.3. This decoder consists of two syndrome registers: the *error-pattern register* and the *error-location register*. The feedback connections of the error-pattern register are based on the

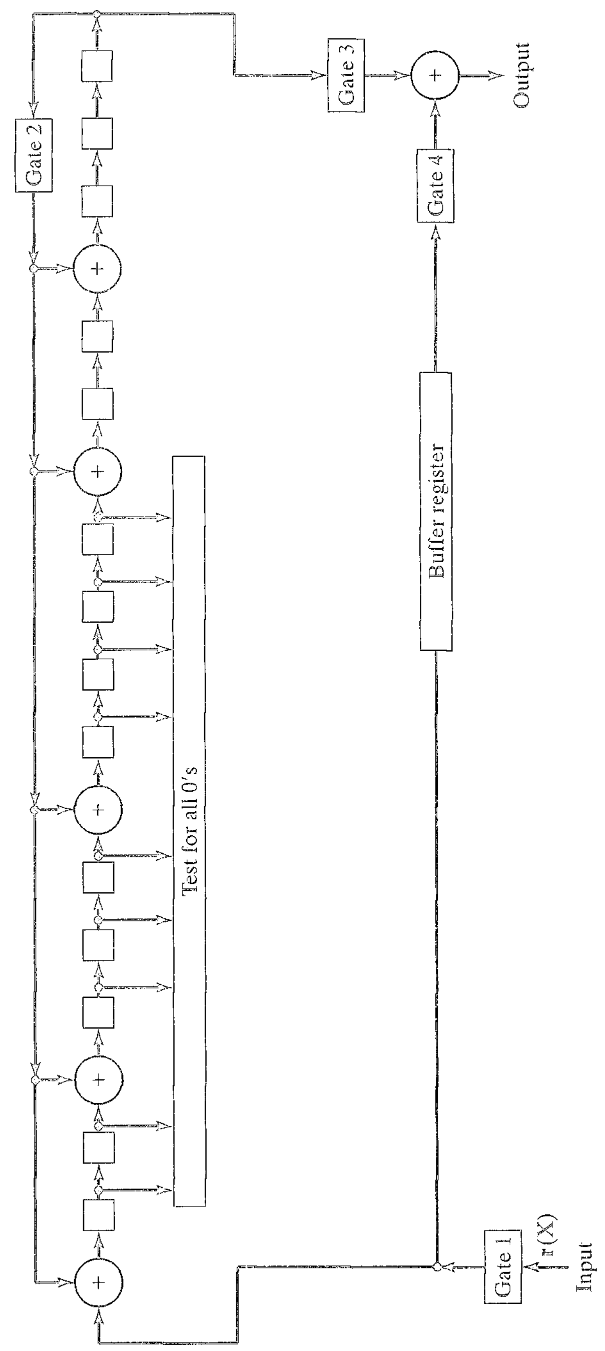

FIGURE 20.2: Error-trapping decoder for the (279, 265) Fire code.

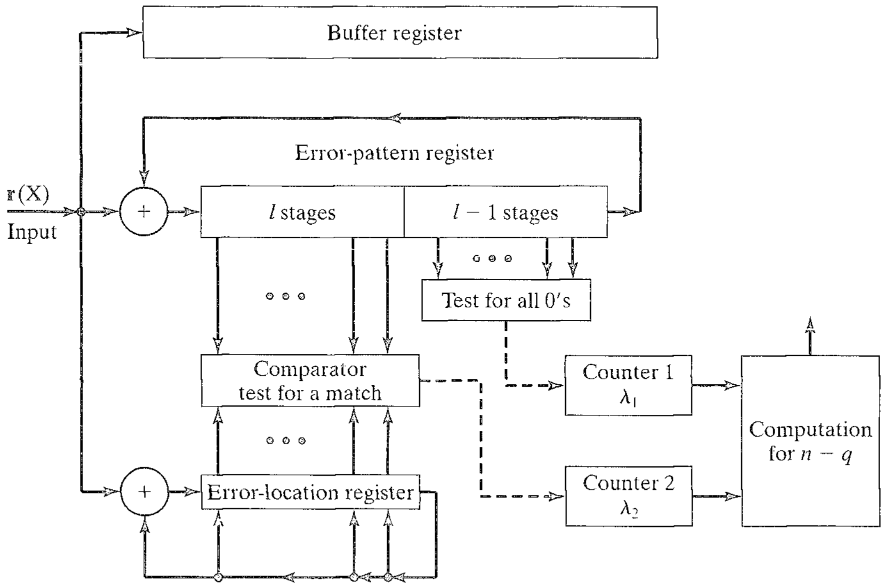

FIGURE 20.3: A high-speed error-trapping decoder for Fire codes.

factor $X^{2l-1}+1$, and the feedback connections of the error-location register are based on the factor $\mathbb{p}(X)$. The received polynomial $\mathbb{r}(X)$ is first read into the two syndrome registers and the buffer register. As soon as the entire $\mathbb{r}(X)$ has been shifted into the two syndrome registers, $\mathbb{s}_1(X)$ and $\mathbb{s}_2(X)$ are formed. The decoder tests $\mathbb{s}_1(X)$ and $\mathbb{s}_2(X)$. If $\mathbb{s}_1(X)=\mathbb{s}_2(X)=0$, the received polynomial $\mathbb{r}(X)$ is assumed to be error-free and is then delivered to the user. If $\mathbb{s}_1(X)=0$ and $\mathbb{s}_2(X)\neq 0$ [or $\mathbb{s}_1(X)\neq 0$ and $\mathbb{s}_2(X)=0$], then $\mathbb{r}(X)$ contains a detectable but uncorrectable error burst and is therefore discarded. If $\mathbb{s}_1(X)\neq 0$ and $\mathbb{s}_2(X)\neq 0$, then $\mathbb{r}(X)$ is assumed to contain a correctable error burst and the decoder starts the following error-correction process:

Step 1. Shift the error-pattern register and test for zeros at the $l-1$ high-order stages. Stop shifting as soon as the $l-1$ high-order stages contain all zeros. The error burst is then trapped in the l low-order stages of the error-pattern register. Let λ_1 be the number of shifts performed (in counter 1). Note that no more than $2l-2$ shifts are needed to trap the error burst.

Step 2. Shift the error-location register until the contents in its l low-order stages match the burst pattern in the l low-order stages of the error-pattern register. Let the number of shifts be λ_2 (in counter 2). In this step, no more than $\rho-1$ shifts are required.

Step 3. Because $2l-1$ and ρ are relatively prime, there exists a unique nonnegative integer q less than n (code length) such that the remainders resulting from dividing q by $2l-1$ and ρ are λ_1 and λ_2, respectively. Determine the integer q by computation. Then, the error burst begins at position X^{n-q} and ends at position $X^{n-q+l-1}$ of $\mathbb{r}(X)$. In the case that $q = 0$, the error burst begins at position X^0 and ends at position X^{l-1} of $\mathbb{r}(X)$.

Step 4. Let $B(X)$ be the burst pattern trapped in the error-pattern register. Add $X^{n-q}B(X)$ to $\mathbb{r}(X)$ in the buffer register. This completes the error-correction process.

If in step 1 the $l - 1$ high-order stages of the error-pattern register never contain all zeros by the time the register has been shifted $2l - 2$ times, an uncorrectable error burst has been detected. In this event, the decoder stops the error-correction process.

The error-location number $n - q$ can be computed easily. Because $2l - 1$ and ρ are relatively prime, there exist two integers A_1 and A_2 such that

$$A_1(2l - 1) + A_2\rho = 1.$$

The q is simply the remainder resulting from dividing

$$A_1(2l - 1)\lambda_2 + A_2\rho\lambda_1$$

by n. Once A_1 and A_2 are determined, the numbers $A_1(2l - 1)$ and $A_2\rho$ can be stored in the receiver permanently for use in each decoding. Therefore, computing $n - q$ requires two multiplications, one addition, one division, and one subtraction.

We note that the error-trapping decoder for Fire codes described here requires at most $2l + \rho - 3$ shifts of the two syndrome registers and five arithmetic operations to carry out the error-correction process; however, the error-trapping decoder described in Section 20.2 takes n shifts (cycle times) to complete the error-correction process. Because $n = \text{LCM}(2l - 1, \rho)$, it is much greater than $2l + \rho - 3$. Therefore, decoding speed is improved. This improvement in decoding speed is possible only when the receiver has some computation capability, or computation facility is available at the receiver. Furthermore, the fast error-trapping decoder requires more logic.

EXAMPLE 20.2

Consider the (279, 265) Fire code considered in Example 20.1. This code is capable of correcting any error burst of length $l = 5$ or less. The fast error-trapping decoder for this code is shown in Figure 20.4. Suppose that the error burst

$$\mathbb{e}(X) = X^2 + X^3 + X^4 + X^5 + X^6$$

has occurred. It is a solid burst of length 5 starting at position $n - q = 2$. The syndromes $\mathbb{s}_1(X)$ and $\mathbb{s}_2(X)$ are remainders resulting from dividing $\mathbb{e}(X)$ by $X^{11} + 1$ and $\mathbb{p}(X) = 1 + X^2 + X^5$, respectively:

$$\mathbb{s}_1(X) = X^2 + X^3 + X^4 + X^5 + X^6,$$

$$\mathbb{s}_2(X) = 1 + X + X^4.$$

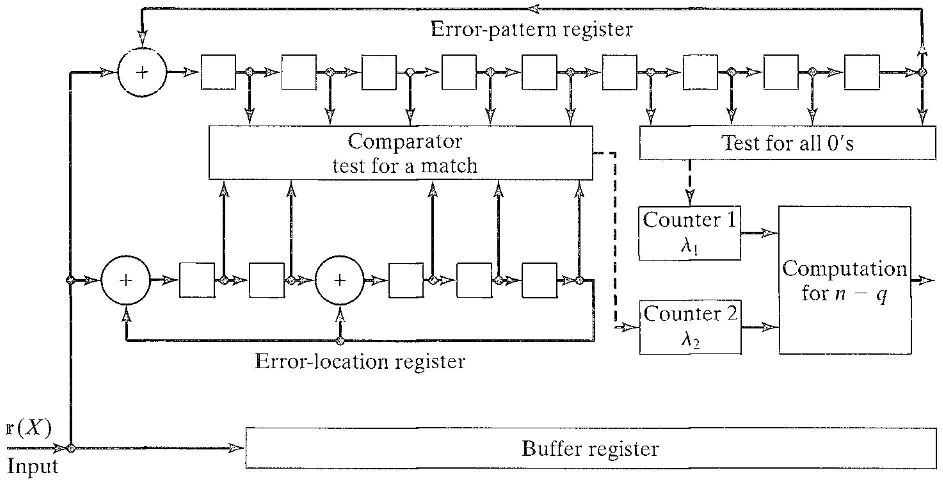

FIGURE 20.4: A high-speed error-trapping decoder for the (279, 265) Fire code.

TABLE 20.1: Contents in the error-pattern register of the decoder shown in Figure 20.4 after each shift.

Shift	Contents
0	0 0 1 1 1 1 1 0 0
1	0 0 0 1 1 1 1 1 0
2	0 0 0 0 1 1 1 1 1
3	1 0 0 0 0 1 1 1 1
4	1 1 0 0 0 0 1 1 1
5	1 1 1 0 0 0 0 1 1
6	1 1 1 1 0 0 0 0 1
7*	1 1 1 1 1 0 0 0 0

*At the seventh shift, all contents in the four high-order stages are all zeros.

As soon as the entire received polynomial $\mathbf{r}(X)$ has been shifted into the error-pattern and error-location registers, the contents in the two registers are $\mathbf{s}_1(X)$ and $\mathbf{s}_2(X)$. Because all four high-order stages of the error-pattern register do not contain all zeros, the error burst is not trapped in the five low-order stages. The error-pattern register starts to shift. Table 20.1 shows the contents in the error-pattern register after each shift. We see that the error burst is trapped in the five low-order stages after $\lambda_1 = 7$ shifts. Now, the error-location register begins to shift. Table 20.2 displays the contents in the error-location register after each shift. At the 29th shift, the contents in the error-location register match the contents in the five low-order stages of the error-pattern register. Therefore, $\lambda_2 = 29$. Next, we need to compute

TABLE 20.2: Contents in the error-location register of the decoder shown in Figure 20.4 after each shift.

Shift	Contents	Shift	Contents
0	1 1 0 0 1	15	0 1 0 0 0
1	1 1 0 0 0	16	0 0 1 0 0
2	0 1 1 0 0	17	0 0 0 1 0
3	0 0 1 1 0	18	0 0 0 0 1
4	0 0 0 0 1	19	1 0 1 0 0
5	1 0 1 0 1	20	0 1 0 1 0
6	1 1 1 1 0	21	0 0 1 0 1
7	0 1 1 1 1	22	1 0 1 1 0
8	1 0 0 1 1	23	0 1 0 1 1
9	1 1 1 0 1	24	1 0 0 0 1
10	1 1 0 1 0	25	1 1 1 0 0
11	0 1 1 0 1	26	0 1 1 1 0
12	1 0 0 1 0	27	0 0 1 1 1
13	0 1 0 0 1	28	1 0 1 1 1
14	1 0 0 0 0	29*	1 1 1 1 1

*At the 29th shift, the contents match the burst pattern in the error-pattern register.

the error-location number $n - q$. First, we find that

$$7 \times 9 + (-2) \times 31 = 1$$

($A_1 = 7$, and $A_2 = -2$). Then, we compute

$$7 \times 9 \times 29 + (-2) \times 31 \times 7 = 1393.$$

Dividing 1393 by $n = 279$, we obtain $q = 277$. Consequently, $n - q = 2$, which is exactly the error-location number. Error correction is achieved by adding the error burst $X^2 + X^3 + X^4 + X^5 + X^6$ to the received polynomial $\mathbf{r}(X)$ in the buffer register. The error-correction process takes at most $8 + 30 = 38$ cycle times. With the decoder shown in Figure 20.2, the error-correction process takes $n = 279$ cycle times.

The fast error-trapping decoder for Fire codes was first devised by Peterson [21] and then refined by Chien [17].

The burst-error-correcting efficiency of a Fire code is $z = 2l/(m + 2l - 1)$. If l is chosen to be equal to m, then $z = 2m/(3m - 1)$. For large m, z is approximately 2/3. Thus, Fire codes are not very efficient with respect to the Reiger bound; however, they can be simply implemented.

A Fire code that is capable of correcting any burst of length l or less and simultaneously detecting any burst of length $d \geq l$ is generated by

$$\mathbf{g}(X) = (X^c + 1)\mathbf{p}(X),$$

where $c \geq l + d - 1$, and c is not divisible by the period ρ of $\mathbf{p}(X)$. The length of this code is the LCM of c and ρ.

20.3.2 Short Efficient Burst-Error-Correcting Codes

Besides Fire codes, some very efficient cyclic codes and shortened cyclic codes for correcting short single bursts have been found either analytically or with the aid of a computer [7, 11, 12]. These codes with their generator polynomials are listed in Table 20.3. These codes and the codes derived from them by interleaving are the most efficient single-burst-error-correction codes known.

TABLE 20.3: Some burst-error-correcting cyclic and shortened cyclic codes.

$n - k - 2l$	Code (n, k)	Burst-error-correcting capability l	Generator polynomial $\mathbf{g}(X)$*
0	(7, 3)	2	35
	(15, 9)	3	171
	(15, 7)	4	721
	(15, 5)	5	2467
	(19, 11)	4	1151
	(21, 9)	6	14515
	(21, 7)	7	47343
	(21, 5)	8	214537
	(21, 3)	9	1647235
	(27, 17)	5	2671
	(34, 22)	6	15173
	(38, 24)	7	114361
	(50, 34)	8	224531
	(56, 38)	9	1505773
	(59, 39)	10	4003351
1	(15, 10)	2	65
	(21, 14)	3	171
	(21, 12)	4	11663
	(21, 10)	5	7707
	(23, 12)	5	5343
	(27, 20)	3	311
	(31, 20)	5	4673
	(38, 29)	4	1151
	(48, 37)	5	4501
	(63, 50)	6	22377
	(63, 48)	7	105437
	(63, 46)	8	730535
	(63, 44)	9	2002353
	(67, 54)	6	36365
	(96, 79)	7	114361
	(103, 88)	8	501001

TABLE 20.3: (*continued*)

$n - k - 2l$	Code (n, k)	Burst-error-correcting capability l	Generator polynomial $g(X)$*
2	(17, 9)	3	471
	(21, 15)	2	123
	(31, 25)	2	161
	(31, 21)	4	3551
	(35, 23)	5	13627
	(39, 27)	5	13617
	(41, 21)	9	6647133
	(51, 41)	4	3501
	(51, 35)	7	304251
	(55, 35)	9	7164555
	(57, 39)	8	1341035
	(63, 55)	3	711
	(63, 53)	4	2263
	(63, 51)	5	16447
	(63, 49)	6	61303
	(73, 63)	4	2343
	(85, 75)	4	2651
	(85, 73)	5	10131
	(105, 91)	6	70521
	(131, 119)	5	15163
	(169, 155)	6	55725
3	(51, 42)	3	1455
	(63, 56)	2	305
	(85, 76)	3	1501
	(89, 78)	4	4303
	(93, 82)	4	6137
	(121, 112)	3	1411
	(151, 136)	6	114371
	(164, 153)	4	6255
	(195, 182)	5	22475
	(217, 202)	6	120247
	(290, 277)	5	24711
4	(43, 29)	5	52225
	(91, 79)	4	10571
	(93, 83)	3	2065
	(117, 105)	4	13413
	(133, 115)	7	1254355
	(255, 245)	3	3523
	(255, 243)	4	17667
	(255, 241)	5	76305
	(255, 239)	6	301565

(*continued overleaf*)

TABLE 20.3: (*continued*)

$n - k - 2l$	Code (n, k)	Burst-error-correcting capability l	Generator polynomial $g(X)$*
4	(273, 261)	4	10743
	(511, 499)	4	10451
	(595, 581)	5	64655
5	(465, 454)	3	7275
	(1023, 1010)	4	22365

*Generator polynomials are given in octal representation. Each digit represents three binary digits according to the following code:

$$\begin{array}{llll} 0 \longleftrightarrow 0\,0\,0 & 2 \longleftrightarrow 0\,1\,0 & 4 \longleftrightarrow 1\,0\,0 & 6 \longleftrightarrow 1\,1\,0 \\ 1 \longleftrightarrow 0\,0\,1 & 3 \longleftrightarrow 0\,1\,1 & 5 \longleftrightarrow 1\,0\,1 & 7 \longleftrightarrow 1\,1\,1 \end{array}$$

The binary digits are then the coefficients of the polynomial, with the high-order coefficients at the left. For example, the binary representation of 171 is 0 0 1 1 1 1 0 0 1, and the corresponding polynomial is $g(X) = X^6 + X^5 + X^4 + X^3 + 1$.

20.3.3 Burst-Error-Correcting Codes Constructed by Interleaving

Code interleaving, presented in Section 4.8, is a powerful technique for constructing long powerful burst-error-correcting codes from short efficient burst-error-correcting codes. Suppose we interleave a burst-error-correcting (n, k) linear code C by a degree λ. An $(\lambda n, \lambda k)$ linear code C^λ results. A code array in C is shown in Figure 4.5. Obviously, a pattern of errors can be corrected for the whole array if and only if the pattern of errors in each row is a correctable pattern for C. No matter where it starts, a burst of length λ will affect no more than one digit in each row. Thus, if C corrects single errors, the interleaved code C^λ will correct single bursts of length λ or less. If C corrects any single burst of length l or less, the interleaved code C^λ will correct any single burst of length λl or less. If C has maximum possible burst-error-correcting capability (i.e., $n - k - 2l = 0$), the interleaved code C^λ also has maximum possible burst-error-correcting capability. By interleaving short codes with maximum possible burst-error-correcting capability, it is possible to construct codes of practically any length with maximum possible burst-error-correcting capability. Therefore, the interleaving technique reduces the problem of searching long efficient burst-error-correcting codes to searching good short codes.

If the original code C is cyclic, the interleaved code C^λ is also cyclic. Let $g(X)$ be the generator polynomial of C. Then, the generator polynomial of the interleaved code C^λ is $g(X^\lambda)$ (see Problem 5.15 or Problem 19.7). Therefore, encoding and syndrome computation can be performed by shift registers. The decoder for the interleaved code can be derived from the decoder of the original code C simply by replacing each register stage of the original decoder by λ stages without changing the other connections. This essentially allows the decoder circuitry to look at successive rows of the code array in successive decoder cycles. Therefore, if the decoder of the original code is simple, so is the decoder for the interleaved code.

EXAMPLE 20.3

Consider the first code given in Table 20.3. It is a (7, 3) cyclic code C generated by

$$\begin{aligned} g(X) &= (X+1)(X^3+X+1) \\ &= 1+X^2+X^3+X^4. \end{aligned}$$

This code is capable of correcting any burst of length 2 or less. It is optimal, since its burst-correcting efficiency is $z = 2l/(n-k) = 2 \times 2/4 = 1$. Suppose we interleave this code to degree $\lambda = 10$. The interleaved code C^{10} is a (70, 30) cyclic code with generator polynomial

$$g(X^{10}) = 1 + X^{20} + X^{30} + X^{40}.$$

This interleaved code is capable of correcting any burst of length 20 or less. The burst-correcting efficiency of this code is $z = 2 \times 20/40 = 1$. Hence, it is also optimal.

In decoding an interleaved code C^λ, the decoder first rearranges the received sequence into an array, then decodes each row of the array based on C. If C is cyclic, error-trapping decoding can be used to decode each row, or a single error-trapping decoder can be devised for the interleaved cyclic code C^λ by modifying the error-trapping decoder for C.

20.3.4 Burst-Error-Correcting Codes Constructed by Product

Code product, presented in Section 4.7, is another powerful technique for constructing long powerful burst-error-correcting codes from short burst-error-correcting codes. Let l_1 and l_2 be the burst-error-correcting capabilities of codes C_1 and C_2, respectively. The burst-error-correcting capability of the product $C_1 \times C_2$ of C_1 and C_2 can be analyzed as follows. Suppose that a code array as shown in Figure 4.3 is transmitted row by row, and at the output of the channel, the received digits are rearranged back into an array row by row. No matter where it starts, any existing error burst of length $n_1 l_2$ or less will affect no more than $l_2 + 1$ consecutive rows; when the received digits are rearranged back into an array, each column will at most affected by a burst of length l_2. Now, if the array is decoded on a column-by-column basis, the burst will be corrected. Therefore, the burst-error-correcting capability of the product code is at least $n_1 l_2$. Suppose that a code array is transmitted on a column-by-column basis and decoded on a row-by-row basis. By a similar argument, it is possible to show that any error burst of length $n_2 l_1$ or less can be corrected. Thus, the burst-error-correcting capability of the product code is at least $n_2 l_1$. Consequently, we may conclude that the burst-error-correcting capability l of the product code is at least max $\{n_1 l_2, n_2 l_1\}$.

Consider a cyclic product code $C_1 \times C_2$ (see Section 5.11). Suppose that the cyclic code C_1 has random-error-correcting capability t_1 and burst-error-correcting capability l_1, and cyclic code C_2 has random-error-correcting capability t_2 and burst-error-correcting capability l_2. Then, the burst-error-correcting capability l of the cyclic product code $C_1 \times C_2$ is at least equal to max $(n_1 t_2 + l_1, n_2 t_1 + l_2)$ [23]; that is,

$$l \geq \max(n_1 t_2 + l_1, n_2 t_1 + l_2). \tag{20.6}$$

This result can be shown as follows. Suppose that an error burst of length $n_2t_1 + l_2$ or less occurred during the transmission of a code array (see Figure 5.23 for transmission of a code array). When the received digits are rearranged back into an array, all except l_2 adjacent rows will contain t_1 or fewer errors. Each of these l_2 adjacent rows will contain $t_1 + 1$ or fewer errors. If the rows are decoded first, these l_2 adjacent rows may contain errors after the row decoding. Therefore, after row decoding, each column of the array contains an error burst of at most length l_2. Because the column code C_2 is capable of correcting any error burst of length l_2 or less, all the remaining errors in the array will be corrected by column decoding. By a similar argument, any error burst of length $n_1t_2 + l_1$ or less will be corrected if the column decoding is performed before the row decoding. Therefore, we obtain the result as stated by (20.6).

20.4 PHASED-BURST-ERROR-CORRECTING CODES

Consider an (n, k) code whose length n is a multiple of m, say $n = \sigma m$. The σm digits of each codeword may be grouped into σ subblocks; each subblock consists of m consecutive code digits. For example, let

$$v = (v_0, v_1, v_2, \cdots, v_{\sigma m-1})$$

be a codeword. Then, the ith subblock consists of the following consecutive code digits:

$$v_{im}, v_{im+1}, \cdots, v_{(i+1)m-1},$$

with $0 \leq i < \sigma$. A burst of length λm or less is called a *phased burst* if and only if it is confined to λ consecutive subblocks, where λ is a positive integer less than σ. A linear code of length $n = \sigma m$ that is capable of correcting all phased error bursts confined to λ or fewer subblocks is called a *λm-phased-burst-error-correcting code*. Because a burst of length $(\lambda - 1)m + 1$, no matter where it starts, can affect at most λ subblocks, it is clear that a λm-phased-burst-error-correcting code is capable of correcting any single burst of length $(\lambda - 1)m + 1$ or less. Thus, a λm-phased-burst-error-correcting code can be used as a $[(\lambda - 1)m + 1]$-single-burst-error-correcting code.

20.4.1 Burton Codes

Next, we present a class of phased-burst-error-correcting cyclic codes similar to the class of Fire codes and was discovered by Burton [18]. Let $\mathbf{p}(X)$ be an irreducible polynomial of degree m and period ρ. Let n be the LCM of m and ρ. Then, $n = \sigma m$. For any positive integer m there exists an m-phased burst-error-correcting Burton code of length $n = \sigma m$ that is generated by

$$\mathbf{g}(X) = (X^m + 1)\mathbf{p}(X). \tag{20.7}$$

The number of parity-check digits of this code is $2m$. Thus, it is a $(\sigma m, (\sigma - 2)m)$ cyclic code. Each codeword consists of σ subblocks. To show that the Burton code generated by $\mathbf{g}(X) = (X^m + 1)\mathbf{p}(X)$ is capable of correcting all phased bursts confined to a single subblock of m digits, it is necessary and sufficient to prove that no two bursts are in the same coset of a standard array for the code. The proof is left as an exercise (see Problem 20.8).

A Burton code can be decoded with the error-trapping decoder described in Section 20.2, except that the contents of the m leftmost stages of the syndrome register are tested for zero at every mth shift. If m and the period ρ of $\mathbb{p}(X)$ are relatively prime, and if the receiver has some computation power, Burton codes can be decoded with the fast error-trapping algorithm described in Section 20.3.

It is possible to interleave an m-phased-burst-error-correcting Burton code in such a way that the interleaved $(\lambda n, \lambda k)$ code is capable of correcting any phased burst that is confined to λ consecutive subblocks. To do this, we arrange λ codewords in the m-phased-burst-error-correcting code into λ rows of a rectangular array as usual. We regard a subblock of each row as a single element. Then, the array consists of σ columns, and each column consists of λ subblocks. The array is transmitted column by column, one subblock at a time from each row. Therefore, a codeword in the interleaved code consists of $\lambda\sigma$ subblocks. No matter where it starts, any phased-error burst confined to λ or fewer subblocks will affect no more than one subblock in each row. Thus, a phased burst of length λm will be corrected if the array is decoded on a row-by-row basis. If the interleaved code is used as a $[(\lambda - 1)m + 1]$-burst-error-correcting code, its burst-error-correcting efficiency is

$$z = \frac{2[(\lambda - 1)m + 1]}{2\lambda m} = 1 - \frac{1}{\lambda}\left(\frac{m-1}{m}\right).$$

As the interleaving degree λ becomes large the burst-error-correcting efficiency of a Burton code approaches 1. Thus, by interleaving the Burton codes, we obtain a class of asymptotically optimal burst-error-correcting codes.

The obvious way to implement an interleaved Burton code is to set up the code array and operate on rows in encoding and decoding. Thus, the encoder of the interleaved code consists of the encoder of the original code and a buffer for the storage of the row vectors of the code array; the decoder consists of the decoder of the original code and a buffer for the storage of the received code array. Of course, the interleaved code can be decoded with the error-trapping decoder of Figure 20.1, in which the contents of the λm leftmost stages are tested for zeros at every mth shift.

20.5 BURST-AND-RANDOM-ERROR-CORRECTING CODES

On many communication channels, errors occur neither independently at random nor in well-defined single bursts but in a mixed manner. Random-error-correcting codes or single-burst-error-correcting codes will be either inefficient or inadequate in combating these mixed errors. Consequently, it is desirable to design codes that are capable of correcting random errors and/or single or multiple error bursts. There are several methods of constructing such codes. The most effective method is the interleaving technique. By interleaving a t-random-error-correcting (n, k) code to degree λ, we obtain a $(\lambda n, \lambda k)$ code capable of correcting any combination of t bursts of length λ or less.

A product code also can be used for simultaneous random-error correction and burst-error correction. Let d_1 and d_2 be the minimum distances of codes C_1 and C_2, respectively. Then, it is possible to show that the product code of C_1 and C_2 is capable of correcting any combination of $t = \lfloor (d_1 d_2 - 1)/2 \rfloor$ or fewer random errors and simultaneously correcting any error burst of length $l = \max(n_1 t_2, n_2 t_1)$ or less, where $t_1 = \lfloor (d_1 - 1)/2 \rfloor$, and $t_2 = \lfloor (d_2 - 1)/2 \rfloor$ [23, 24]. To prove this assertion it is sufficient to show that an error burst of length l or less and a random-error pattern of t or fewer errors cannot be in the same coset of a standard array for the product code. Suppose that $n_1 t_2 \geq n_2 t_1$. Then, $l = n_1 t_2$. Consider a burst of length $n_1 t_2$ or less. When this vector is arranged as an array of n_2 rows and n_1 columns, each column contains at most t_2 errors. Suppose that this burst and some random-error pattern of t or fewer errors are in the same coset of the product code. Then, the sum of these two error patterns (in array form) is a code array in the product code. As a result, each column of the sum array must either have no nonzero components or have at least d_2 nonzero components. Each nonzero column of the sum array must be composed of at least $d_2 - t_2$ errors from the random-error pattern and at most t_2 errors from the burst-error pattern. Because there are at most t random errors, these errors can be distributed among at most $\lfloor t/(d_2 - t_2) \rfloor$ columns. Thus, the sum array contains at most $\lfloor t/(d_2 - t_2) \rfloor t_2 + t$ nonzero components; however,

$$\left\lfloor \frac{t}{d_2 - t_2} \right\rfloor t_2 + t \leq t\left(\frac{t_2}{d_2 - t_2} - 1\right) < 2t.$$

Hence, the sum array contains fewer than $2t < d_1 d_2$ nonzero components and cannot be a code array in the product code. This contradiction implies that a burst of length $l = n_1 t_2$ or less and a random-error pattern of t or fewer errors cannot be in the same coset of a standard array for the product code. Therefore, they can both be used as coset leaders and are correctable error patterns. If $n_2 t_1 > n_1 t_2$, then $l = n_2 t_1$. The same argument can be applied to rows instead of columns of the sum array.

20.5.1 Codes Derived from RS Codes

In Chapter 2 it was pointed out that any element β in the Galois field $GF(2^m)$ can be expressed uniquely as a sum of $1, \alpha, \alpha^2, \cdots, \alpha^{m-1}$ in the following form:

$$\beta = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{m-1}\alpha^{m-1},$$

where α is a primitive element in $GF(2^m)$, and $a_1 = 0$ or 1. Thus, the correspondence between β and $(a_0, a_1, \cdots, a_{m-1})$ is one-to-one. We shall call the m-tuple $(a_0, a_1, \cdots, a_{m-1})$ an *m-bit byte* representation of β.

Consider a t-error-correcting RS code with code symbols from $GF(2^m)$. If each symbol is represented by its corresponding m-bit byte, we obtain a binary linear code with the following parameters:

$$n = m(2^m - 1),$$
$$n - k = 2mt.$$

This binary code is called a *binary image* of the RS code and is capable of correcting any error pattern that affects t or fewer m-bit bytes. It is immaterial whether a byte has one error, or all the m bits are in error; they are counted as one byte error, as follows. At the channel output the binary received vector is divided into $2^m - 1$ bytes; each byte is transformed back into a symbol in $GF(2^m)$. Thus, if an error pattern affects t or fewer bytes, it affects t or fewer symbols in a RS code. Obviously, the error pattern can be corrected by the decoding methods described in Chapter 7. We shall call this binary code a *t-byte correcting code*, but it is actually a multiple-phased-burst-error-correcting code.

Binary codes derived from RS codes are more effective against clustered errors than random errors, since clustered errors usually involve several errors per byte and thus relatively few byte errors. For example, since a burst of length $3m+1$ cannot affect more than 4 bytes, a 4-byte-correcting code can correct any single burst of length $3m+1$ or less. It can also simultaneously correct any combination of two bursts of length $m+1$ or less, because each such burst can affect no more than 2 bytes. At the same time, it can correct any combination of four or fewer random errors. In general, a t-byte-correcting binary RS code is capable of correcting any combination of

$$\lambda = \frac{t}{1 + \lfloor (l+m-2)/m \rfloor}$$

or fewer bursts of length l [or correcting any single burst of length $(t-1)m+1$ or less]. Simultaneously, it corrects any combination of t or fewer random errors.

20.5.2 Concatenated Codes

Concatenated codes, presented in Section 15.1, are also effective against a mixture of random errors and bursts. The pattern of bytes not correctable by the inner code C_1 must form a correctable error pattern for the outer code C_2 if the concatenated code is to correct the error pattern. Scattered random errors are corrected by the inner code C_1. Bursts may affect relative few bytes but probably so badly that the inner code C_1 cannot correct them. These few bytes can then be corrected by the outer code C_2.

20.5.3 Modified Fire Codes for Simultaneous Correction of Burst and Random Errors

Let β be an element of order n in the Galois field $GF(2^m)$. It follows from Theorem 2.5 that n is a factor of 2^m-1. Let $\phi(X)$ be the minimal polynomial of β. The period of $\phi(X)$ is n. The degree of $\phi(X)$, m_0, is either equal to m or a factor of m. Suppose that n has a proper factor b such that

$$\frac{b+1}{2} \le m_0.$$

Let $n = a \cdot b$. Then,

$$X^n + 1 = (X^b+1)(1 + X^b + X^{2b} + \cdots + X^{(a-1)b}).$$

Because the order of β is n, and $b < n$, β cannot be a root of X^b+1 and must be a root of $1 + X^b + X^{2b} + \cdots + X^{(a-1)b}$. Therefore, $\phi(X)$ divides $1 + X^b + X^{2b} + \cdots + X^{(a-1)b}$. The code generated by

$$g_1(X) = (X^b+1)\phi(X) \tag{20.8}$$

is a Fire code C_1 of length n that is capable of correcting any error burst of length $(b+1)/2$ of less.

Let $g_2(X)$ be the generator polynomial of a cyclic code C_2 of length n that is capable of correcting t or fewer random errors. Clearly, $g_2(X)$ is a factor of X^n+1. Let $g(X)$ be the least common multiple of $g_1(X)$ and $g_2(X)$:

$$g(X) = \text{LCM}\,\{g_1(X), g_2(X)\}. \tag{20.9}$$

Clearly, $g(X)$ divides $X^n + 1$ and can be expressed in the form

$$g(X) = (X^b + 1)g_0(X), \tag{20.10}$$

where $g_0(X)$ is a factor of $1 + X^b + X^{2b} + \cdots + X^{(a-1)b}$. Now, we consider the cyclic code C of length n generated by $g(X)$. This code C is a *subcode* of both the Fire code C_1 generated by $g_1(X) = (X^b + 1)\phi(X)$ and the t-error-correcting code C_2 generated by $g_2(X)$. Because C is a subcode of the Fire code C_1, C is capable of correcting any single error burst of length $(b + 1)/2$ or less, and since C is a subcode of the t-error-correcting code C_2, it is capable of correcting any combination of t or fewer random errors. Because $g(X)$ has $(X + 1)$ as a factor, the minimum distance of C is even and is at least $2t + 2$. It is possible to show that C is capable of correcting any single error burst of length $(b + 1)/2$ or less *as well as* any combination of t or fewer random errors. To show this, it is necessary and sufficient to prove that a burst of length $(b + 1)/2$ or less and an error pattern of weight t or less cannot be in the same coset of C unless they are identical (see Problem 20.15).

EXAMPLE 20.4

Let α be a primitive element of the Galois field $GF(2^6)$. The order of α is $2^6 - 1 = 63$, and the minimal polynomial of α is

$$\phi(X) = 1 + X + X^6.$$

The integer 63 can be factored as follows: $63 = 7 \cdot 9$. Thus, we have

$$X^{63} + 1 = (X^9 + 1)(1 + X^9 + X^{18} + X^{27} + X^{36} + X^{45} + X^{54}).$$

The code generated by the polynomial

$$g_1(X) = (X^9 + 1)(1 + X + X^6)$$

is a Fire code of length 63 that is capable of correcting any single error burst of length 5 or less. Let $g_2(X)$ be the generator polynomial of the double-error-correcting BCH code of length 63. From Table 6.4 we find that

$$g_2(X) = (1 + X + X^6)(1 + X + X^2 + X^4 + X^6).$$

Note that both factors of $g_2(X)$ are factors of $1 + X^9 + X^{18} + X^{27} + X^{36} + X^{45} + X^{54}$. The LCM of $g_1(X)$ and $g_2(X)$ is

$$g(X) = (X^9 + 1)(1 + X + X^6)(1 + X + X^2 + X^4 + X^6).$$

Hence, $g(X)$ generates a (63, 42) cyclic code that is a subcode of both the Fire code generated by $g_1(X) = (X^9 + 1)(1 + X + X^6)$ and the double-error-correcting BCH code generated by $g_2(X) = (1 + X + X^6)(1 + X + X^2 + X^4 + X^6)$. Therefore, it is capable of correcting any single error burst of length 5 or less as well as any combination of two or fewer random errors.

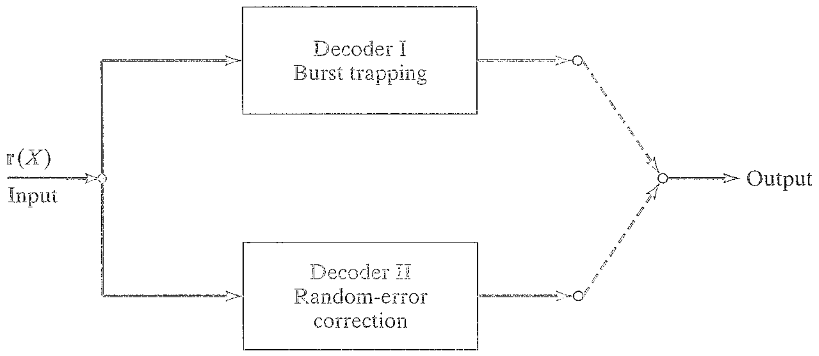

FIGURE 20.5: Parallel decoding for simultaneous correction of burst and random errors.

Decoding of the code defined by (20.9) can be implemented in a straightforward manner, as shown in Figure 20.5, where the two decoders operate in parallel. Decoder I is an error-trapping decoder that is implemented based on the Fire code generated by $g_1(X) = (X^b + 1)\phi(X)$; and decoder II is implemented based on the random-error-correcting code generated by $g(X)$. The received polynomial $r(X)$ is shifted into both decoders simultaneously. Both decoders attempt to decode $r(X)$. The error-trapping decoder gives a decoded message only if the error pattern is either a burst of length $(b+1)/2$ or less or an undetectable burst. The random-error decoder gives a decoded message only if the error pattern either contains t of fewer errors or is an undetectable error pattern. The only time when both decoders will provide decoded messages simultaneously is only when the error pattern is in a coset with a coset leader that is a burst of length $\leq (b+1)/2$ and of weight $\leq t$. In this case the decoded messages from the two decoders are identical. If both decoders fail to decode $r(X)$, errors are detected.

EXAMPLE 20.5

In Example 20.4, let us choose $g_2(X)$ as the generator polynomial of the (1, 3)th-order twofold (63, 45) EG code. From Example 8.21 we find that

$$g_2(X) = (1 + X + X^6)(1 + X + X^2 + X^4 + X^6)(1 + X + X^2 + X^5 + X^6).$$

Then, the LCM of $g_1(X) = (X^9 + 1)(1 + X + X^6)$, and $g_2(X)$ is

$$g(X) = (X^9 + 1)(1 + X + X^6)(1 + X + X^2 + X^4 + X^6)(1 + X + X^2 + X^5 + X^6).$$

Hence, $g(X)$ generates a (63, 36) cyclic code that is capable of correcting any single error burst of length 5 or less as well as any three or fewer random errors with majority-logic decoding.

There is a (63, 36) BCH code that is capable of correcting any combination of five or fewer errors. Clearly, this BCH code is more powerful than the foregoing (63, 36) code; however, decoding for the (63, 36) BCH code is more complex than the decoding of the (63, 36) code here.

By combining Fire codes and BCH codes and with the aid of a computer, Hsu et al. have constructed several classes of shortened cyclic codes that are capable of correcting burst errors as well as random errors [26]. Other works on constructing burst-and-random error-correcting block codes can be found in [11, 19, and 26–28].

PROBLEMS

20.1 Show that if an (n, k) cyclic code is designed to correct all burst errors of length l or less and simultaneously to detect all burst errors of length $d \geq l$ or less, the number of parity-check digits of the code must be at least $l + d$.

20.2 Devise an error-trapping decoder for an l-burst-error-correcting cyclic code. The received polynomial is shifted into the syndrome register from the right end. Describe the decoding operation of your decoder.

20.3 Prove that the Fire code generated by (20.4) is capable of correcting any error burst of length l or less.

20.4 The polynomial $\mathbf{p}(X) = 1 + X + X^4$ is a primitive polynomial over $GF(2)$. Find the generator polynomial of a Fire code that is capable of correcting any single error burst of length 4 or less. What is the length of this code? Devise a simple error-trapping decoder for this code.

20.5 Devise a high-speed error-trapping decoder for the Fire code constructed in Problem 20.4. Describe the decoding operation.

20.6 Use a code from Table 20.3 to derive a new code with burst-error-correcting capability $l = 51$, length $n = 255$, and burst-error-correcting efficiency $z = 1$. Construct a decoder for this new code.

20.7 Let $\mathbf{g}(X)$ be the generator polynomial of an (n, k) cyclic code. Interleave this code to degree λ. The resultant code is a $(\lambda n, \lambda k)$ linear code. Show that this interleaved code is cyclic and its generator polynomial is $\mathbf{g}(X^\lambda)$.

20.8 Show that the Burton code generated by $\mathbf{g}(X) = (X^m + 1)\mathbf{p}(X)$, where $\mathbf{p}(X)$ is an irreducible polynomial of degree m, is capable of correcting all phased bursts confined to a single subblock of m digits.

20.9 Let $m = 5$. Construct a Burton code that is capable of correcting any phased burst confined to a single subblock of five digits. Suppose that this code is interleaved to degree $\lambda = 6$. What are the length, the number of parity-check digits, and the burst-error-correcting capability of this interleaved code?

20.10 Interleave the (164, 153) code in Table 20.3 to degree $\lambda = 6$. Compare this interleaved code with the interleaved Burton code of Problem 20.9. Which code is more efficient?

20.11 Interleave the (15, 7) BCH code to degree 7. Discuss the error-correcting capability of this interleaved code. Devise a decoder for this code and describe the decoding operation.

20.12 Consider the (31, 15) RS code with symbols from $GF(2^5)$. Convert this RS code to a binary code. Discuss the error-correcting capability of the binary RS code.

20.13 Suppose that the Fire code constructed in Problem 20.4 is shortened by deleting the 15 high-order message digits. Devise a decoder for the shortened code such that the 15 extra shifts of the syndrome register after the received vector has entered can be avoided.

20.14 Find a modified Fire code of length 63 that is capable of correcting any single burst of length 4 or less as well as any combination of two or fewer random errors. Determine its generator polynomial.

20.15 Consider the modified Fire code C generated by $\mathbf{g}(X)$ of (20.9). Show that a burst of length $(b + 1)/2$ or less and error pattern of weight t or less cannot be in the same coset.

BIBLIOGRAPHY

1. N. Abramson, "A Class of Systematic Codes for Non-independent Errors," *IRE Trans. Inform. Theory*, IT-4(4): 150–57, December 1959.

2. N. Abramson and B. Elspas, "Double-Error-Correcting Coders and Decoders for Non-independent Binary Errors," presented at the UNESCO Inform. Process. Conf., Paris, 1959.

3. P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-independent Errors," *Sylvania Report No. RSL-E-2*, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, Calif., March 1959.

4. C. M. Melas, "A New Group of Codes for Correction of Dependent Errors in Data Transmission," *IBM J. Res. Dev.*, 4: 58–64, January 1960.

5. S. H. Reiger, "Codes for the Correction of 'Clustered' Errors," *IRE Trans. Inform. Theory*, IT-6: 16–21, March 1960.

6. B. Elspas, "A Note on Binary Adjacent-Error-Correcting Codes," *IRE Trans. Inform. Theory*, IT-6: 13–15, March 1960.

7. B. Elspas and R. A. Short, "A Note on Optimum Burst-Error-Correction Codes," *IRE Trans. Inform. Theory*, IT-8: 39–42, January 1962.

8. J. E. Meggitt, "Error Correcting Codes for Correcting Bursts of Errors," *IBM J. Res. Dev.*, 4: 329–34, July 1960.

9. J. E. Meggitt, "Error Correcting Codes for Correcting Bursts of Errors," *Trans. AIEE*, 80: 708–22, January 1961.

10. C. R. Foulk, "Some Properties of Maximally Efficient Cyclic Burst-Correcting Codes and Results of a Computer Search for Such Codes," File No. 375, Digital Computer Lab., University of Illinois, Urbana, June 12, 1961.

11. J. J. Stone, "Multiple Burst Error Correction," *Inform. Control*, 4: 324–31, December 1961.

12. T. Kasami and S. Matoba, "Some Efficient Shortened Cyclic Codes for Burst-Error Correction," *IEEE Trans. Inform. Theory*, IT-10: 252–53, July 1964.

13. A. J. Gross, "Binary Group Codes Which Correct in Bursts of Three or Less for Odd Redundancy," *IEEE Trans. Inform. Theory*, IT-8: 356–59, October 1962.

14. A. J. Gross, "A Note on Some Binary Group Codes Which Correct Errors in Bursts of Four or Less," *IRE Trans. Inform. Theory*, IT-8: 384, October 1962.

15. A. J. Gross, "Augmented Bose–Chaudhuri Codes Which Correct Single Bursts of Errors," *IEEE Trans. Inform. Theory*, IT-9: 121, April 1963.

16. E. Gorog, "Some New Classes of Cyclic Codes Used for Burst Error Correction," *IBM J. Res. Dev.*, 7: 102–11, 1963.

17. R. T. Chien, "Burst-Correction Codes with High-Speed Decoding," *IEEE Trans. Inform. Theory*, IT-1: 109–13, January 1969.

18. H. O. Burton, "A Class of Asymptotically Optimal Burst Correcting Block Codes," presented at the ICCC, Boulder, Colo., June 1969.

19. S. E. Tavares and S. G. S. Shiva, "Detecting and Correction Multiple Bursts for Binary Cyclic Codes," *IEEE Trans., Inform. Theory*, IT-16: 643–44, 1970.

20. R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley, New York, 1968.

21. W. W. Peterson, *Error-Correcting Codes*, MIT Press, Cambridge, 1961.

22. P. Elias, "Error-Free Coding." *IRE Trans. Inform. Theory*, PGIT-4: 29–37, September 1954.

23. H. O. Burton and E. J. Weldon, Jr., "Cyclic Product Codes," *IEEE Trans. Inform. Theory*, IT-11: 433–40, July 1964.

24. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed., MIT Press, Cambridge, 1970.

25. S. M. Reddy and J. P. Robertson, "Random Error and Burst Correction by Iterated Codes," *IEEE Trans. Inform. Theory*, IT-18: 182–85, January 1972.

26. H. T. Hsu, T. Kasami, and R. T. Chien, "Error-Correction Codes for a Compound Channel," *IEEE Trans. Inform. Theory*, IT-14: 135–39, January 1968.

27. W. Posner, "Simultaneous Error-Correction and Burst-Error Detection Binary Linear Cyclic Codes," *J. Soc. Ind. Appl. Math.*, 13: 1087–95, December 1965.

28. L. Bahl and R. T. Chien, "A Class of Multiple-Burst-Error-Correcting Codes," presented at the IEEE Int. Symp. Inform. Theory, Ellenville, N.Y., 1969.