Optical Storage Technology

Error Correction

Introduction

- With analog audio, there is no opportunity for error correction. With digital audio, the **nature of binary data** lends itself to recovery in the event of damage.
- When audio data is stored, it can be specially **coded** and accompanied by **redundancy**. This enables the reproduced data to be checked for error.
- **Error correction** is an opportunity to **preserve data integrity**, and it is absolutely necessary to ensure the success of digital audio storage.
- With proper design, CD and DVD can be reduced to an error rate of $\mathbf{10^{-12}}$, that is, less than one uncorrectable error in 10^{12} bits.

Sources of Errors

- Optical media can affected by **pit asymmetry**, **bubbles or defects in substrate**, and **coating defects**.
- The most significant cause of error in digital media are **dropouts**, essentially a **defect** in the media that causes a **momentary drop** in signal strength.
- Dropouts can be traced to two causes : a **manufactured defect** in the media or a **defect** introduced **during use**.
- A loss of data or invalid data can provoke a **click** or **pop**.
- An error in the **least significant bit** of PCM word might pass unnoticed, but an error in the **most significant bit** word would create a drastic change in amplitude.

Sources of Errors

- Errors that have no relation to each other are called **random-bit errors**.
- A **burst error** is a large error, disrupting perhaps **thousands of bits**.
- An important **characteristic** of any error-correction system is the **burst length**, that is, the **maximum** number of **adjacent erroneous bits** that can be corrected.
- Error-correction design is influenced by the kind of **modulation code** used to convey the data.
- With normal **wear** and **tear**, **oxide particles** coming form the backing and other foreign particles, such as **dust**, **dirt**, and **oil from fingerprints**, can contribute to the number of dropouts.

Sources of Errors

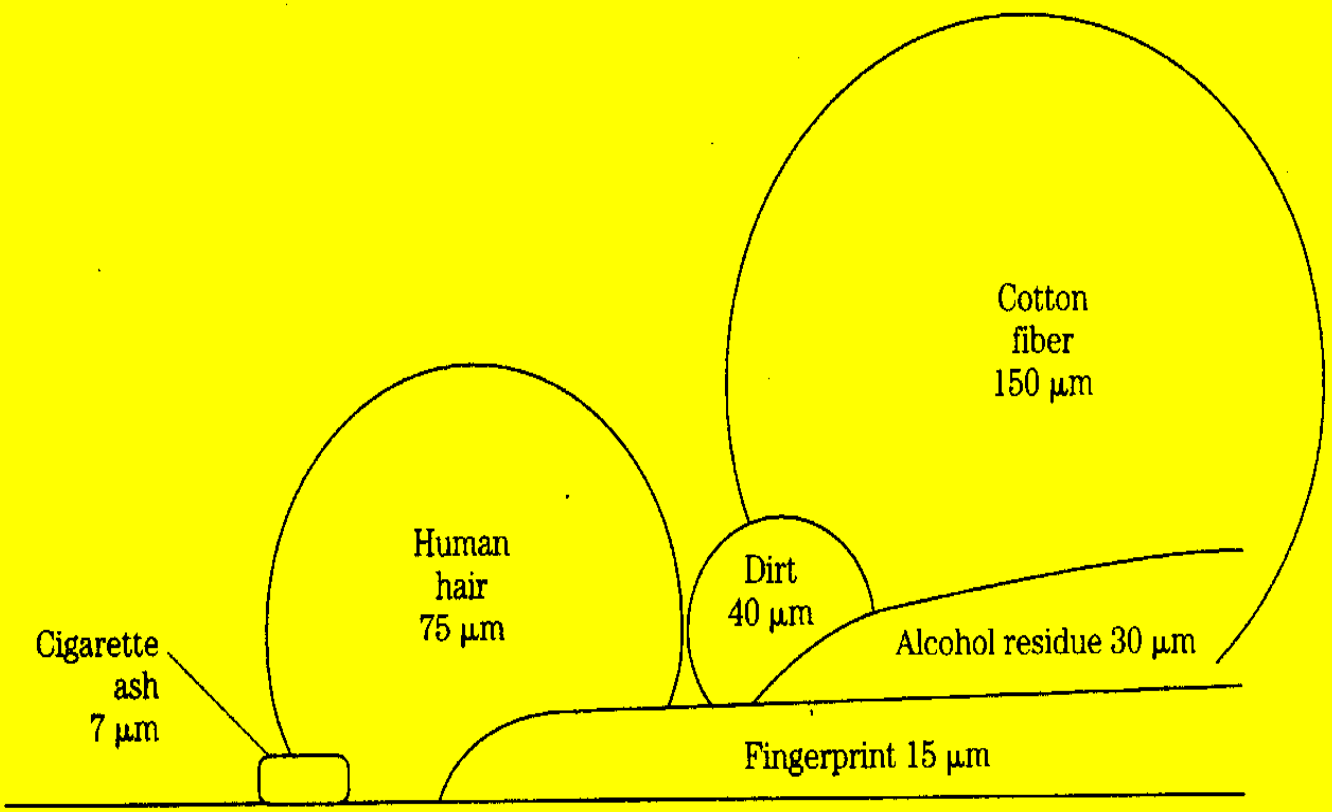


Figure 5.1 Small physical objects are large in the context of 1-μm recorded bit size.

Sources of Errors

- The scratches **perpendicular to tracks** are easier to correct, but a scratch **along one track** could be impossible to correct.
- The **bit-error rate (BER)** is the number of bits received in error divided by the total number of bit received.
- An optical disc system can obtain error-correction algorithms able to handle a **BER of 10^{-5} to 10^{-4}**.
- The **block error rate (BLER)** measures the number of blocks or frames of data per second that have at least one occurrence of uncorrected data.
- The **burst-error length (BERL)** counts the number of consecutive blocks in error.

Objectives of errors correction

- **Redundancy** alone will not ensure accuracy of the recovered information; appropriate **error detection** and **correction coding** must be used.

- An error-correction system comprises three operations :

1. **Error detection** uses redundancy to permit data to be checked for validity.
2. **Error correction** uses redundancy to replace erroneous data with newly calculated valid data.
3. In the event of large errors or insufficient data for correction, **error concealment techniques** substitute approximately correct data for invalid data.

Errors Detection- Single-bit parity

- All error-detection and correction techniques are based on the **redundancy** of data.
- Practical **error detection** uses technique in which redundan data is coded so it can be used to efficiently check for errors
- The technique of casting out **9s** can be used to cast out any number, and forms the basis for a binary error detection method called **parity**.
- Given a binary number, a residue bit can be formed by casting out 2s. This extra bit, known as **a parity bit**, permits **error detection**, but **not correction**.
- An even parity bit is formed with a simple rule : if the numbe of 1s in the data word is **even**, the parity bit is a **0**; if the number of 1s in the word is **odd**, the parity bit is a **1**.
- An **8-bit** data word, made into a **9-bit** word with an even parity bit, will always have **an even number of 1s**.

Errors Detection- Single-bit parity

Casting out 9s:

$240 + 578 \stackrel{?}{=} 818 \rightarrow$ $(2 + 4 + 0 = 6)$
$(5 + 7 + 8 = 20, 2 + 0 = 2)$
$(8 + 1 + 8 = 17, 1 + 7 = 8)$

$6 + 2 \stackrel{\checkmark}{=} 8$ Casting out 9s sum agrees, thus no error

$227 \times 67 \stackrel{?}{=} 15209$
$2 \times 4 \stackrel{\checkmark}{=} 8$ Casting out 9s product agrees, thus no error

$154 \times 95 \stackrel{?}{=} 14613$
$1 \times 5 \neq 6$ Casting out 9s product does not agree calculation is in error

Casting out 2s:

Sum of 11001011 is 5, which is odd. Cast out 2s to get 1 and append to word:110010111. In this way, the number of 1s is always even.

Figure 5.2 Casting out of 9s and 2s provides simple error detection.

Errors Detection- Single-bit parity

8 DATA BITS	PARITY BIT (EVEN PARITY)
00000000	0
01011100	0
00100110	1
11111111	0
00001101	1
11010110	1

8 DATA BITS	PARITY BIT (ODD PARITY)
00000000	1
01011100	1
00100110	0
11111111	1
00001101	0
11010110	0

Figure 5.3 Parity can be formed through the modulo 2 addition of data bits.

Transmitted word		Received word		Parity calculated from received data word	
Data	Parity	Data	Parity		
00011001	1	00001001	1	0	Error detected
10101011	1	11001011	1	1	Errors not detected
01110100	0	01110100	1	0	Parity error detected
01101011	1	00000011	0	0	Errors not detected

Figure 5.4 Examples of single-bit parity error detection.

Errors Detection- ISBN

- An **ISBN (International Standard Book Number)** is not just a series of numbers. (Ex. 0-14-044118-2)
- The first digit is a country code; the next two digits is a publisher code; the next six digits is the book number code; the **last** digit is a **check digit**.
- To form the weighted checksum of a ten-digit number *abcdefghij*, compute the weighted sum of the numbers by :

  $$10a + 9b + 8c + 7d + 6e + 5f + 4g + 3h + 2i + 1j$$

- For the ISBN number 0-14-044118-2, the weighted sum is equal to 110.
- The weighted checksum **modulo 11** is found by taking the remainder after dividing by 11 : 110/11 = 10, with a 0 remainder.
- The **0 remainder** suggest that the ISBN is **correct.**

Cyclic Redundancy Check Code

- The **cyclic redundancy check code (CRCC)** is an error detection method preferred in audio applications.
- The CRCC is cyclic block code that generates a parity check word. In *1011011010*, the six binary 1s are added together t form binary 0110 (6), and this check word is appended to the data word to form the code word for storage.

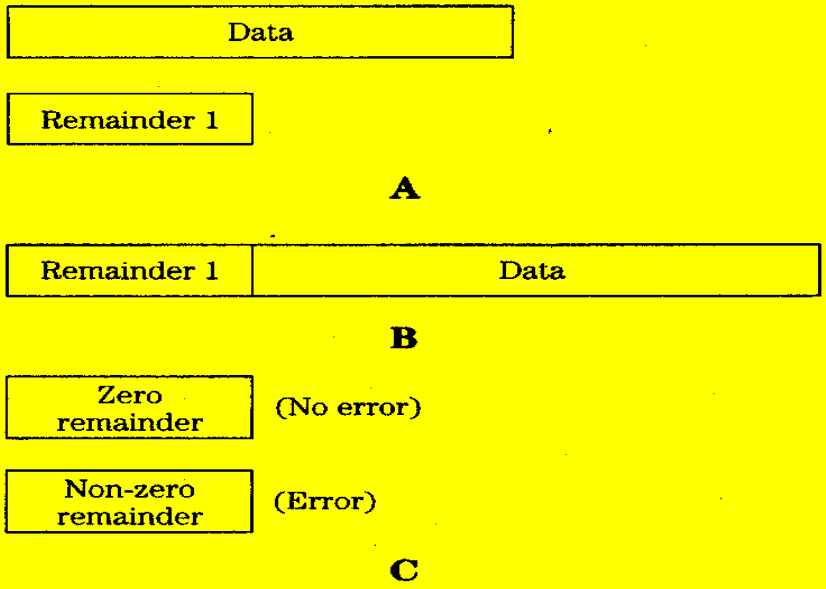


Figure 5.5 CRCC in simplified form showing generation of remainder. A. Original block of data is divided to produce a remainder. The quotient is discarded. B. The remainder is appended to the data word and both words are transmitted or stored. C. The received data is again divided to produce a remainder, used to check for errors.

Cyclic Redundancy Check Code

- Original **k-bit** data block, m(x) $\rightarrow$ multiplied by X^{n-k} , X^{n-k} m(x $\rightarrow$ divided by the **generation polynomial *g*** to form the quotient ***q*** and remainder ***r***, $\boldsymbol{X^{n-k}}$ **m(x) =q(x)g(x)** + **r(x),** however, r(X) + r(X)=0 $\rightarrow$ **r(x) +** $\boldsymbol{X^{n-k}}$ **m(x) =q(x)g(x)** $\rightarrow$ the transmission polynomial **v = q(x)g(x)** is transmitted or stored $\rightarrow$ the received data u undergoes error detection by calculating a **syndrome**, u(x)=p(x)g(x)+s(x) $\rightarrow$ **s(x)=0**, **error free; s(x)$\neq$0, error**.
- **Error correction** can be accomplished by forming an **error pattern** that is the difference between the received data and the original data to be recovered.
- u(x) = v(x) + e(x) = q(x)g(x) + e(x) = p(x)g(x) + s(x) $\rightarrow$ e(x) = [p(x) – q(x)] g(x) + s(x)

Cyclic Redundancy Check Code

- In practice, CRCC and other detection and correction code words are described in **mathematical terms**, where the data bits are treated as the coefficients of a binary polynomial.
- **1001011** $\rightarrow$ $2^6 + 2^3 + 2^1 + 2^0 \rightarrow$ **$x^6 + x^3 + x + 1$**
- Given a k-bit data word with m (where m = n-k) bits of CRCC, a code word of n bits is formed, and the following are true :

1. Burst error **less than or equal** to **m** bits are always **detectable**.
2. Detection **probability** of burst errors of **m+1 bits** is **$1 - 2^{-m+}$**
3. Detection **probability** of burst errors **longer than m+1 bits** is **$1 - 2^{-m}$**
4. **Random errors** up to **three consecutive bits** long can be detected.

Example of cyclic code encoding

Given a message m = (1001) to be encoded, message polynomial $m(x) = x^3 + 1$. Multiplying by x^{n-k}, $x^3 m(x) = x^6 + x^3$.

0	0	0	1	0	0	1	m(x)

$x^3 + 1$

1	0	0	1	0	0	0	$x^3 m(x)$

$x^6 + x^3$

Given a generation polynomial $g(x) = x^3 + x^2 + 1$ division by g(x) is performed:

$$
\begin{array}{r}
x^3 + x^2 + x + 1 \\
x^3 + x^2 + 1 \;\overline{)\; x^6 \qquad\quad x^3} \\
\underline{x^6 + x^5 \quad + x^3} \\
x^5 \\
\underline{x^5 + x^4 \quad\quad + x^2} \\
x^4 \quad + x^2 \\
\underline{x^4 + x^3 \quad\quad + x} \\
x^3 + x^2 + x \\
\underline{x^3 + x^2 \quad\quad + 1} \\
x + 1
\end{array}
$$

Remainder = r(x)

The code word polynomial

$(x) = x^3 m(x) + r(x)$

$= x^6 + x^3 + x + 1$

1	0	0	1	0	1	1

$x^6 + x^3 + x+1$

Message Parity

Figure 5.7 An example of cyclic code encoding, with message 1001 written as the polynomial $x^3 + 1$. The encoder outputs the original message and parity word.

Error-Correction Codes

- With the use of redundant data, it is possible to correct errors that occur during storage or transmission. However, there are many types of codes, different in their designs and functions.
- The field of error-correction codes is **highly mathematical** one.
- In general, two approaches are used : **block codes** using algebraic methods, and **convolutional codes** using probabilistic methods.
- In some cases, algorithms use a block code in a convolutional structure known as a **cross-interleave code**. Such codes are used in the CD format.

Block codes

- **Block error-correction encoders** assemble a number of data words to form a block and, operating over that block, generate one or more parity words and append them to the block.

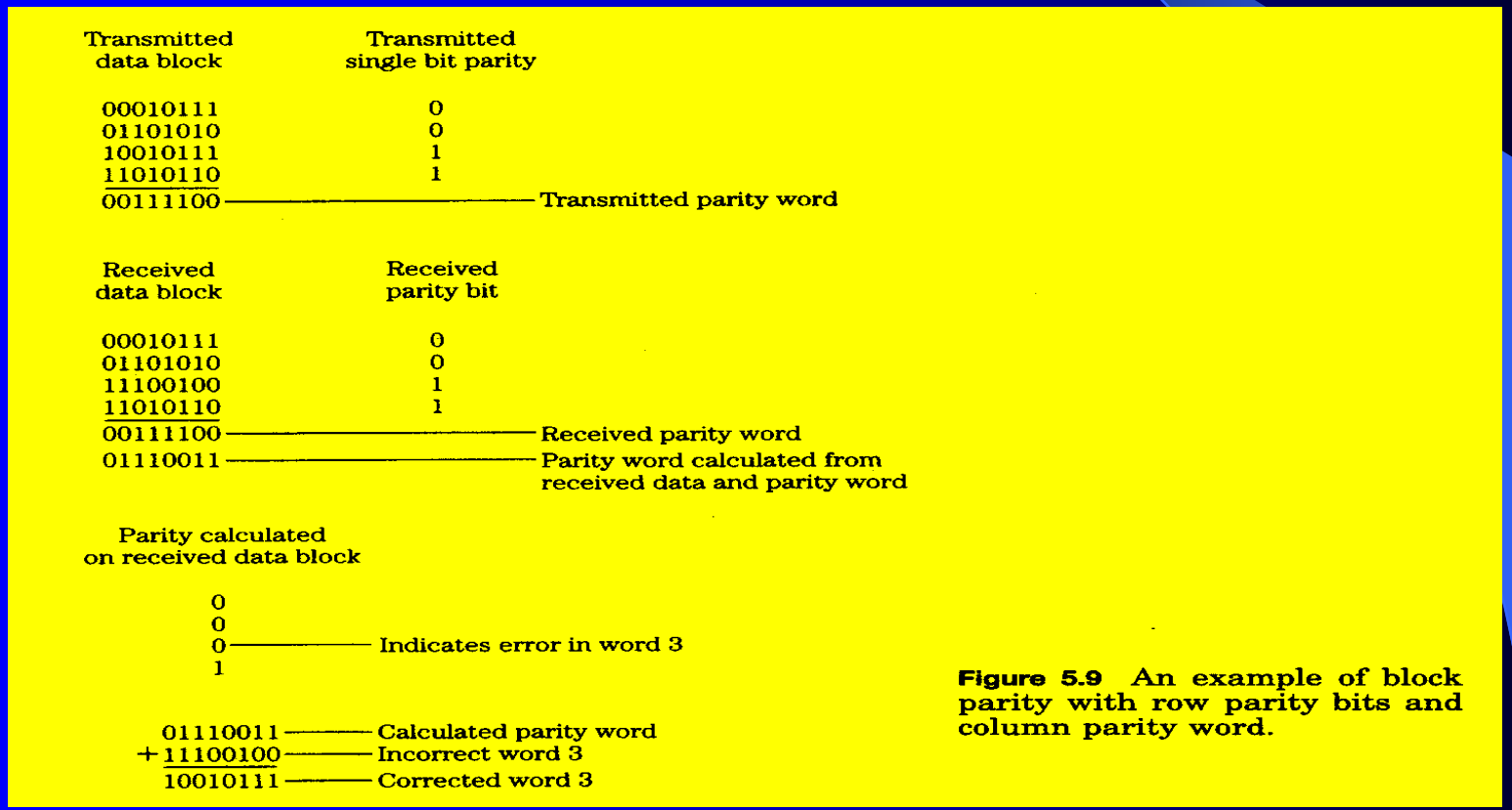


Figure 5.9 An example of block parity with row parity bits and column parity word.

Block codes

Original data words and parity

W_1	10
W_2	30
W_3	20
W_4	25
W_5	30
W_6	15
P	130 = $W_1 + W_2 + W_3 + W_4 + W_5 + W_6$

Received data words and parity

W_1	10
W_2	30
W_3	20
W_4	25
W_5	30
W_6	15
P	130

Syndrome $S = W_1 + W_2 + W_3 + W_4 + W_5 + W_6 - P$
$= 10 + 30 + 20 + 25 + 30 + 15 - 130 = 0$

Thus no error is indicated

A Block correction code showing no error condition

Received data and parity word

W_1	10
W_2	30
W_3	20
W_4	15
W_5	30
W_6	15
P	130

CRCC error pointer

Syndrome $S = 10 + 30 + 20 + 15 + 30 + 15 - 130 = -10$

Error correction: $W_4 = W_4' - S$
$= 15 - (-10)$
$= 25$

B Block correction code showing correction with pointer

Received data and parity word

W_1	10
W_2	30
W_3	20
W_4	25
W_5	30
W_6	15
P	130

False error pointer

Syndrome $S = 10 + 30 + 20 + 25 + 30 + 15 - 130 = 0$

Error correction: $W_5 = W_5' - S$
$= 30 - 0$
$= 30$

C Block correction code showing false pointer

Figure 5.10 Examples of single-parity block coding.

Block codes – double parity

Received data and two parity words

W_1	10
W_2	30
W_3	20
W_4	25
W_5	30
W_6	15
P	$130 = W_1 + W_2 + W_3 + W_4 + W_5 + W_6$
Q	$440 = 6W_1 + 5W_2 + 4W_3 + 3W_4 + 2W_5 + W_6$

Syndrome $S_1 = W_1 + W_2 + W_3 + W_4 + W_5 + W_6 - P = 10 + 30 + 20 + 25 + 30 + 15 - 130 = 0$
$S_2 = 6W_1 + 5W_2 + 4W_3 + 3W_4 + 2W_5 + W_6 - Q = 60 + 150 + 80 + 75 + 60 + 15 - 440 = 0$

A Block correction code with double parity showing no error condition

Received data and two parity words

W_1	10
W_2	30
W_3	20
W_4	25
W_5	10
W_6	15
P	130
Q	440

$S_1 = -20$
$S_2 = -40$

Algebraically we see that

If $6S_1 = S_2$ then W_1 is erroneous
If $5S_1 = S_2$ then W_2 is erroneous
If $4S_1 = S_2$ then W_3 is erroneous
If $3S_1 = S_2$ then W_4 is erroneous
If $2S_1 = S_2$ then W_5 is erroneous
If $S_1 = S_2$ then W_6 is erroneous
If $S_1 \neq 0$ and $S_2 = 0$ then P is erroneous
If $S_1 = 0$ and $S_2 \neq 0$ then Q is erroneous

$6S_1 = 6W_1' + 6W_2' + 6W_3' + 6W_4' + 6W_5' + 6W_6' - 6(W_1 + W_2 + W_3 + W_4 + W_5 + W_6)$

$= 6W_1' + 5W_2' + 4W_3' + 3W_4' + 2W_5' + W_6' - (6W_1 + 5W_2 + 4W_3 + 3W_4 + 2W_5 + W_6)$

$\Rightarrow W_2' + 2W_3' + 3W_4' + 4W_5' + 5W_6' = W_2 + 2W_3 + 3W_4 + 4W_5 + 5W_6$

In this case $2S_1 = S_2$, W5 is erroneous, thus (as in single erasure case):

$S_1 = 10 + 30 + 20 + 25 + 0 + 15 - 130 = -30$

$W_5 = W_5' - S$
$= 0 - (-30)$
$= 30$ Corrected

B Block correction code with double parity showing single error without pointer

Hamming Codes

X_0	X_1	X_2	X_3	X_4	X_5	X_6
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	0	0	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	0
1	1	1	1	1	1	1

A

X_0, X_1, X_2, X_3	Data bits
$X_4 = X_1 + X_2 + X_3$	(Mod 2) parity check bits
$X_5 = X_0 + X_2 + X_3$	(Mod 2)
$X_6 = X_0 + X_1 + X_3$	(Mod 2)
$X_0, X_1, X_2, X_3, X_4, X_5, X_6$	Transmitted code word

B

Hamming Codes

$$
\begin{aligned}
X_1 + X_2 + X_3 + X_4 &= 0 \\
X_0 + X_2 + X_3 + X_5 &= 0 \\
X_0 + X_1 + X_3 + X_6 &= 0
\end{aligned}
$$

Decoding algorithm

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

Parity-check matrix

C

Example: ~~1100110 Transmitted word~~
~~1000110 Received word~~

$$
\begin{aligned}
P_4 &= X_1 + X_2 + X_3 = 0 + 0 + 0 = 0 \\
P_5 &= X_0 + X_2 + X_3 = 1 + 0 + 0 = 1 \\
P_6 &= X_0 + X_1 + X_3 = 1 + 0 + 0 = 1
\end{aligned}
$$

Parity of received data

Syndromes are calculated with mod 2 addition of parity of received data and received parity bits:

$$
\begin{aligned}
P_4 = 0, \quad X_4 = 1, \quad 0 + 1 &= 1 \quad \text{(Error)} \\
P_5 = 1, \quad X_5 = 1, \quad 1 + 1 &= 0 \quad \text{(Correct)} \\
P_6 = 1, \quad X_6 = 0, \quad 1 + 0 &= 1 \quad \text{(Error)}
\end{aligned}
$$

The resulting syndromes form the error pattern $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ which corresponds to the second column of H

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

Thus bit X_1, is in error

D

Convolutional codes

- Convolutional codes do **not** partition data into **blocks**.
- Message digits k are taken a few at a time and used to generate code digits n, formed **not only** from **those k message digits**, **but** from many **previous k digits** as well.
- The coded output contains a history of the previous input data. Such a code is called an **(n,k) convolutional code**.
- Upon retrieval, the correction decoder uses **syndromes** to check code words for errors.
- **Shift registers** can be used to implement the delay memories required in the encoder and decoder.

Convolutional codes

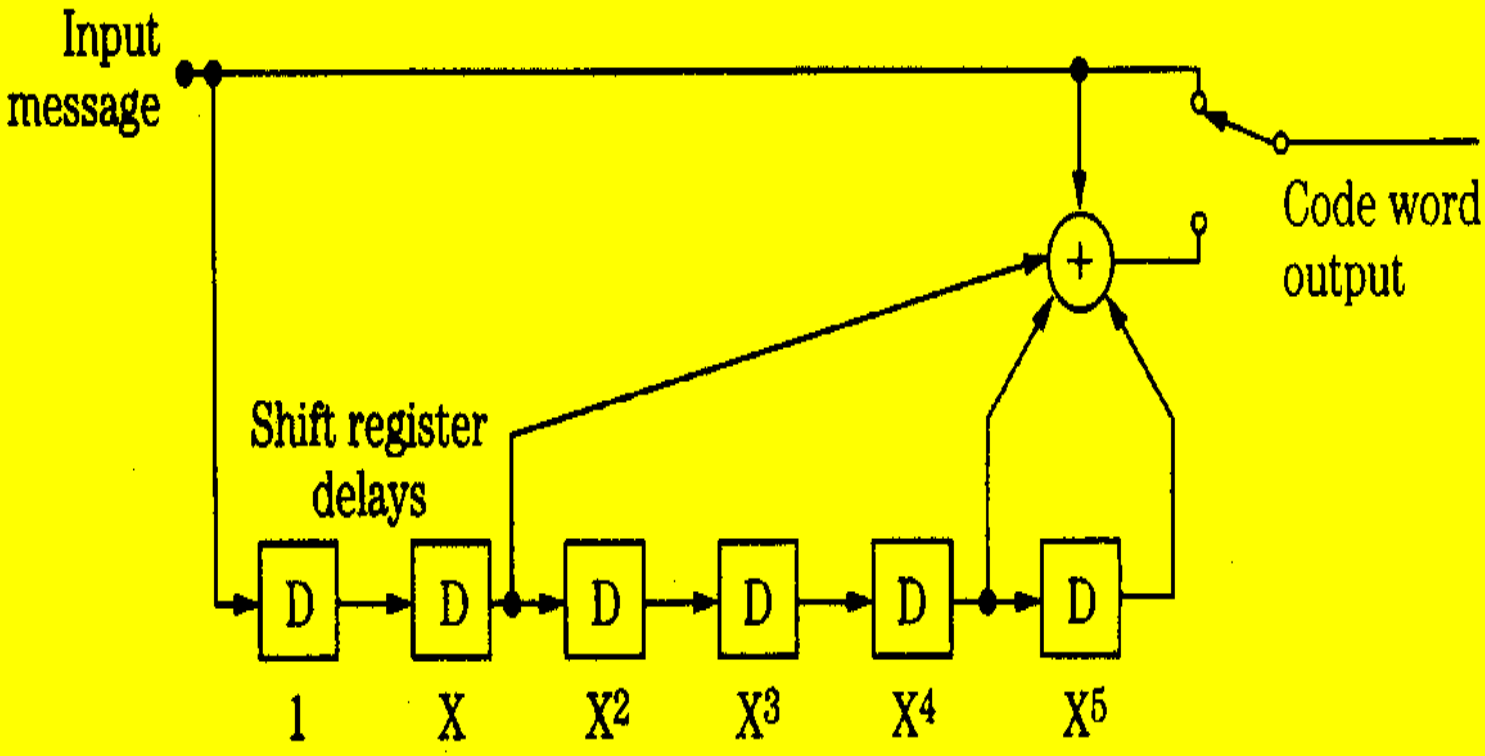


Figure 5.13 A convolutional code encoder with six delay blocks.

Convolutional codes

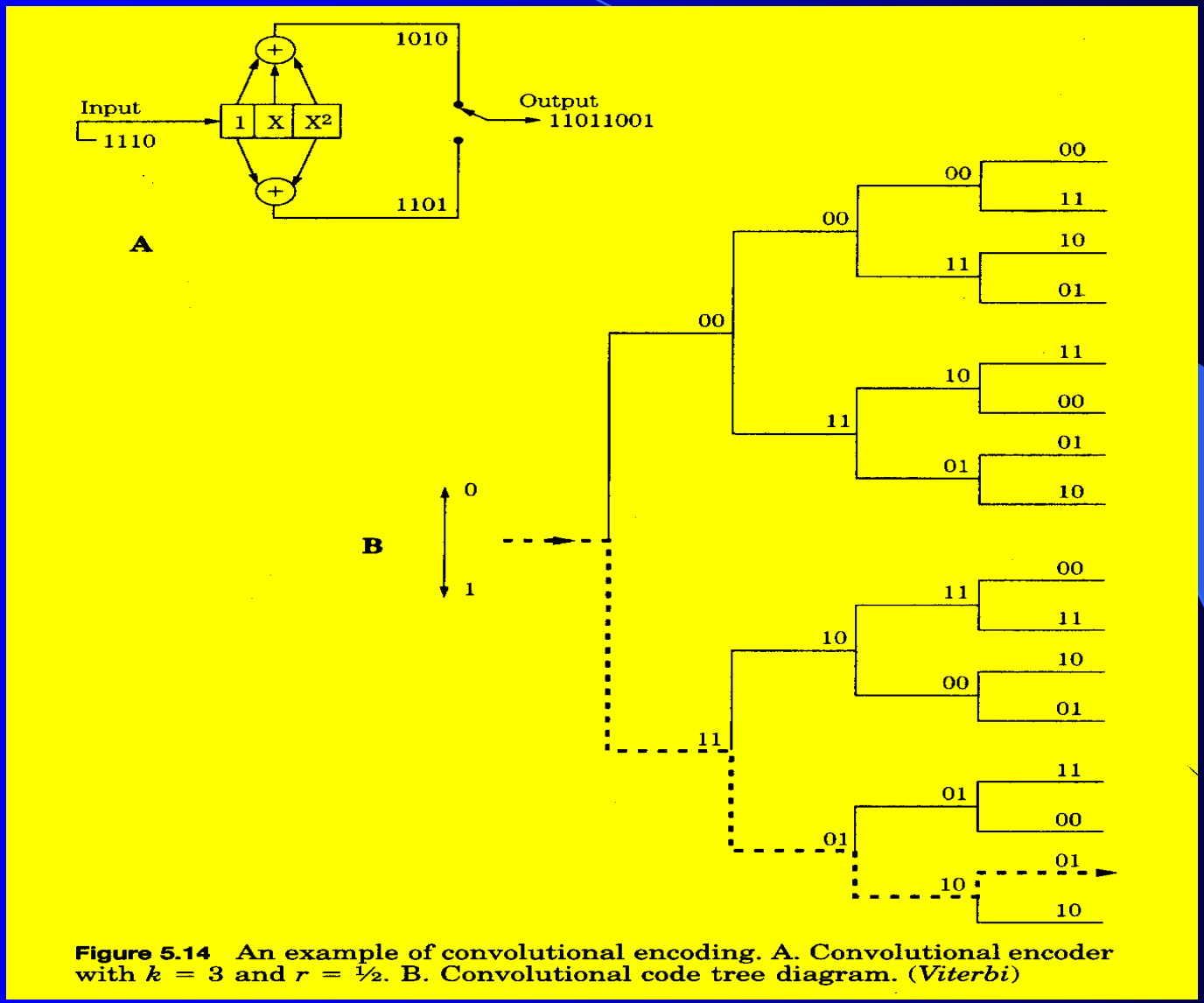


Figure 5.14 An example of convolutional encoding. A. Convolutional encoder with $k = 3$ and $r = ½$. B. Convolutional code tree diagram. (*Viterbi*)

Interleaving

- Error correction depends on an **algorithm's ability** to efficiently use **redundant data** to reconstruct invalid data.
- When the error is sustained, as in the case of a **burst error**, both the data and the redundant data are lost, and correction becomes difficult or impossible.
- Data is **interleaved** or **dispersed** through the data stream prior to storage or transmission.
- With interleaving, the largest error that can occur in any block is limited to the **size of the interleaved section**.
- Interleaving greatly **increases burst error correctability** of the block codes.

Interleaving

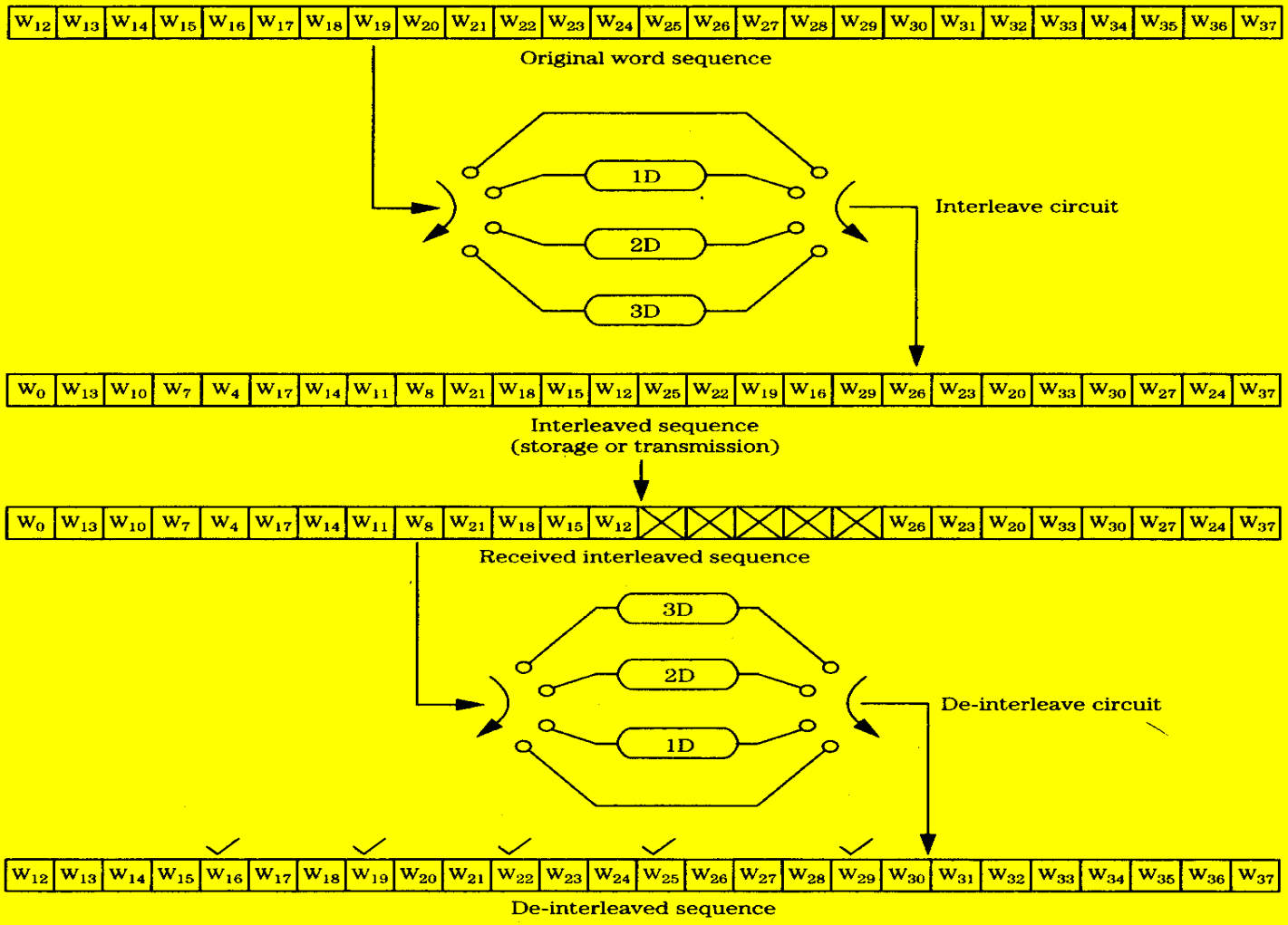


Figure 5.16 Zero-, one-, two-, and three-word delays perform interleaving and de-interleaving for error dispersion prior to correction. (*Doi*)

Cross-interleaving

- Although the burst is scattered, the random errors add additional errors in a given word, perhaps overloading the correction algorithm.
- One solution is to generate **two** correction codes, separated by an interleave and delay.
- When block codes are arranged in rows and columns two-dimensionally, the code is called a **product code**. (in DVD)
- When two block codes are separated by both **interleaving** and **delay**, cross-interleaving results.
- A cross-interleaved code comprises two (or more) block codes assembled with a convolutional structure.

Interleaving

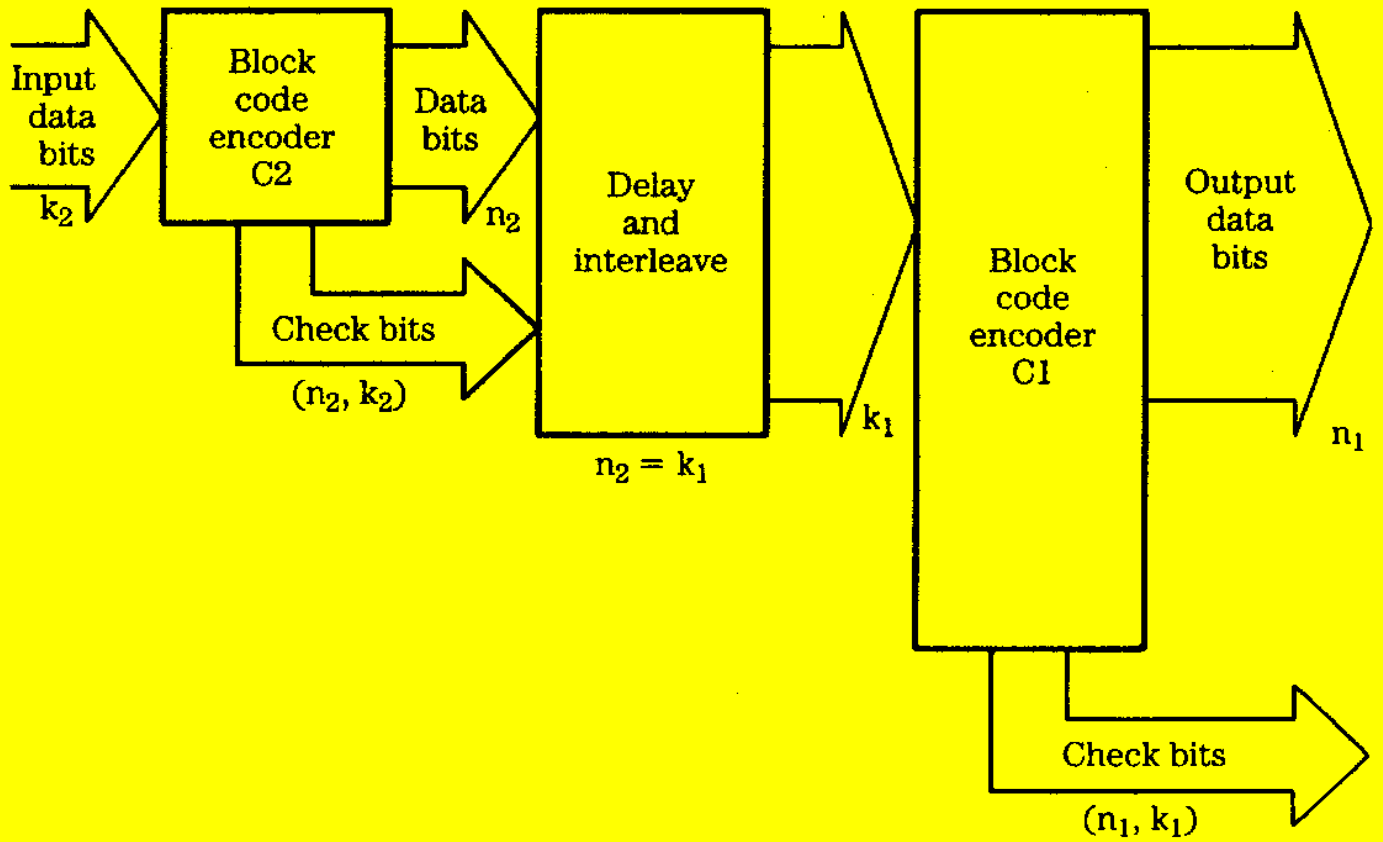


Figure 5.17 A cross-interleave code encoder. Syndromes from the first block are used as error pointers in the second block. In the CD format, $k_2 = 24$, $n_2 = 28$, $k_1 = 28$, and $n_1 = 32$; the C_1 and C_2 codes are Reed-Solomon codes.

Reed-Solomon Codes

- For compact discs, Reed-Solomon codes are used and the algorithm is known as the **Cross-Interleave Reed-Solomon code** (**CIRC**).
- Reed-Solomon codes exclusively use polynomials derived using **finite field** mathematic known as **Galois Field** to encode and decode block data.
- Either **multiplication** or **addition** can be used to combine elements, and the result of adding or multiplying two elements is always a **third element** contained **in** the field.
- In addition, there exists at least one element called a **primitive** such that every other element can be expressed as a power of this element.

Reed-Solomon Codes

- The **size** of the Galois Field, which determine the number of symbols in the code, is based on the **number of bits** comprising a symbol; **8-bit** symbols are commonly used.
- A primitive polynomial often used in GF(2^8) systems is **$x^8+x^4+x^3+x^2+1$**.
- The code can use the input word to generate two types of **parity**, **P** and **Q**. The P parity can be a modulo 2 sum of the symbols. The Q parity multiplies each input word by a different power of the GF primitive element.
- If one symbol is erroneous, the P parity gives a nonzero syndrome S1. The Q parity yields a syndrome S2. By checking this **relationship between S1 and S2**, the RS code can **locate the error**. Correction is performed by adding S1 to the designated location.

Reed-Solomon Codes - example

- Consider a GF(2^3) code comprising 3-bit symbols. α **is the primitive element** and is the solution to the equation :

  $F(x) = x^3 + x + 1$, that is $\alpha^3 + \alpha + 1 = 0$.

- The element can be represented as ordinary polynomials :

000 = 0	001 = 1
010 = x	011 = x + 1
100 = x^2	101 = x^2 + 1
110 = x^2 + X	111 = x^2 + x + 1

- Using the properties of the Galois Field and **modulo 2** (where **1+1 = α + α = α^2 + α^2 = 0**)

  0 = 000, **1** = 001, α = 010, α^2 = 100, $\alpha^3 = \alpha + 1$ = 011

  $\alpha^4 = \alpha(\alpha+1) = \alpha^2+\alpha$ = 110, $\alpha^5 = \alpha^2+\alpha+1$ = 111,

  $\alpha^6 = (\alpha^3)^2 = \alpha^2+1$ = 101, $\alpha^7 = \alpha^3+\alpha = 2\alpha+1 = 1$ = 001

Reed-Solomon Codes - example

- Elements can be **multiplied** by simply adding exponents, always resulting in **another element** in the Galois Field.
- $\alpha \cdot \alpha = \alpha^2 = (010)(010) = 100$, $\alpha^2 \cdot \alpha^3 = \alpha^5 = (100)(011) = 111$

Bits		000	001	010	011	100	101	110	111
	Elements	0	1	α	α^3	α^2	α^6	α^4	α^5
000	0	0	0	0	0	0	0	0	0
001	$\alpha^7 = 1$	0	1	α	α^3	α^2	α^6	α^4	α^5
010	α	0	α	α^2	α^6	α^3	1	α^5	α^6
011	α^3	0	α^3	α^4	α^6	α^5	α^2	1	α
100	α^2	0	α^2	α^3	α^5	α^4	α	α^6	1
101	α^6	0	α^6	1	α^2	α	α^5	α^3	α^4
110	α^4	0	α^4	α^5	1	α^6	α^3	α	α^2
111	α^5	0	α^5	α^6	α	1	α^4	α^2	α^3

Reed-Solomon Codes - example

- Suppose that **A**, **B**, **C**, and **D** are **data** symbols and **P** and **Q** are **parity** symbols. The RS code will satisfy the following equations :

$$A + B + C + D + P + Q = 0$$

$$\alpha^6 A + \alpha^5 B + \alpha^4 C + \alpha^3 D + \alpha^2 P + \alpha^1 Q = 0$$

$$\rightarrow \quad P = \alpha^1 A + \alpha^2 B + \alpha^5 C + \alpha^3 D$$

$$Q = \alpha^3 A + \alpha^6 B + \alpha^4 C + \alpha^1 D$$

- If A = **001** = 1, B = **101** = α^6, C = **011** = α^3, D = **100** = α^2

$$\rightarrow \quad P = 101 = \alpha^6, \quad Q = 110 = \alpha^4$$

- Assume A', B', C', D', P', and Q' are the received data

$$\rightarrow \quad S1 = A' + B' + C' + D' + P' + Q'$$

$$S2 = \alpha^6 A' + \alpha^5 B' + \alpha^4 C' + \alpha^3 D' + \alpha^2 P' + \alpha^1 Q'$$

Reed-Solomon Codes - example

- Each possible error pattern is expressed by E_i :

  $S1 = E_A + E_B + E_C + E_D + E_P + E_Q$

  $S2 = \alpha^6 E_A + \alpha^5 E_B + \alpha^4 E_C + \alpha^3 E_D + \alpha^2 E_P + \alpha^1 E_Q$
- If there is no error, then S1 = S2 = 0

  If symbol A' is erroneous, $S1 = E_A$ and $S2 = \alpha^6 S1$

  If symbol B' is erroneous, $S1 = E_B$ and $S2 = \alpha^5 S1$

  If symbol C' is erroneous, $S1 = E_C$ and $S2 = \alpha^4 S1$

  If symbol D' is erroneous, $S1 = E_D$ and $S2 = \alpha^3 S1$

  If symbol P' is erroneous, $S1 = E_P$ and $S2 = \alpha^2 S1$

  If symbol Q' is erroneous, $S1 = E_Q$ and $S2 = \alpha^1 S1$
- In other word, an **error** results in **nonzero syndromes**; the value of the erroneous symbols can be determined by the difference of the weighting between S1 and S2..

Reed-Solomon Codes - example

- If the received data is : A' = 001 = 1, B' = 101 = α^6, C' = 001 = 1 (erroneous), D' = 100 = α^2, P' = 101 = α^6, Q' = 110 = α^4

  $\rightarrow$ S1 = α = 010, S2 = $\alpha^2 + \alpha + 1 = \alpha^5$ = 111
- Because S2 = α^4S1, symbol C' must be erroneous and because S1 = E_C = 010, C = C' + E_C = 001 + 010 = 011.
- In practice, the polynomials used in CIRC are :

  $P = \alpha^6A + \alpha^1B + \alpha^2C + \alpha^5D + \alpha^3E$

  $Q = \alpha^2A + \alpha^3B + \alpha^6C + \alpha^4D + \alpha^1E$

  and the syndromes are :

  $S1 = A' + B' + C' + D' + E' + P' + Q'$

  $S2 = \alpha^7A' + \alpha^6B' + \alpha^5C' + \alpha^4D' + \alpha^3E' + \alpha^2P' + \alpha^1Q'$
- Reed-Solomon codes are used for error correction in the compact disc, DAT, DVD, direct broadcast satellite, digital radio and digital television applications.

Cross-Interleave Reed-Solomon Codes

- A **quadruple erasure correction** RS code, known as the **cross-interleave Reed-Solomon code** (**CIRC**) has been adopted for the compact disc.
- Encoding carries data through the **C2** encoder, then the **C1** encoder. Decoding reverses the process.
- **C2** is a **(28, 24) code**, that is the encoder input 24 symbols, and outputs 28 symbols, including 4 symbols of Q parity.
- Q parity is designed to correct **one erroneous symbol**, or up to **four erasures** in one word.
- **C1** is a **(32, 28)** code producing four P parity symbols.
- P parity is designed to correct **single-symbol errors** and **detect** and **flag** double and triple errors for Q correction.
- Up to **four** symbols can be corrected if the error location is known, and **two** symbols can be corrected if the location is not known.

Cross-Interleave Reed-Solomon Codes

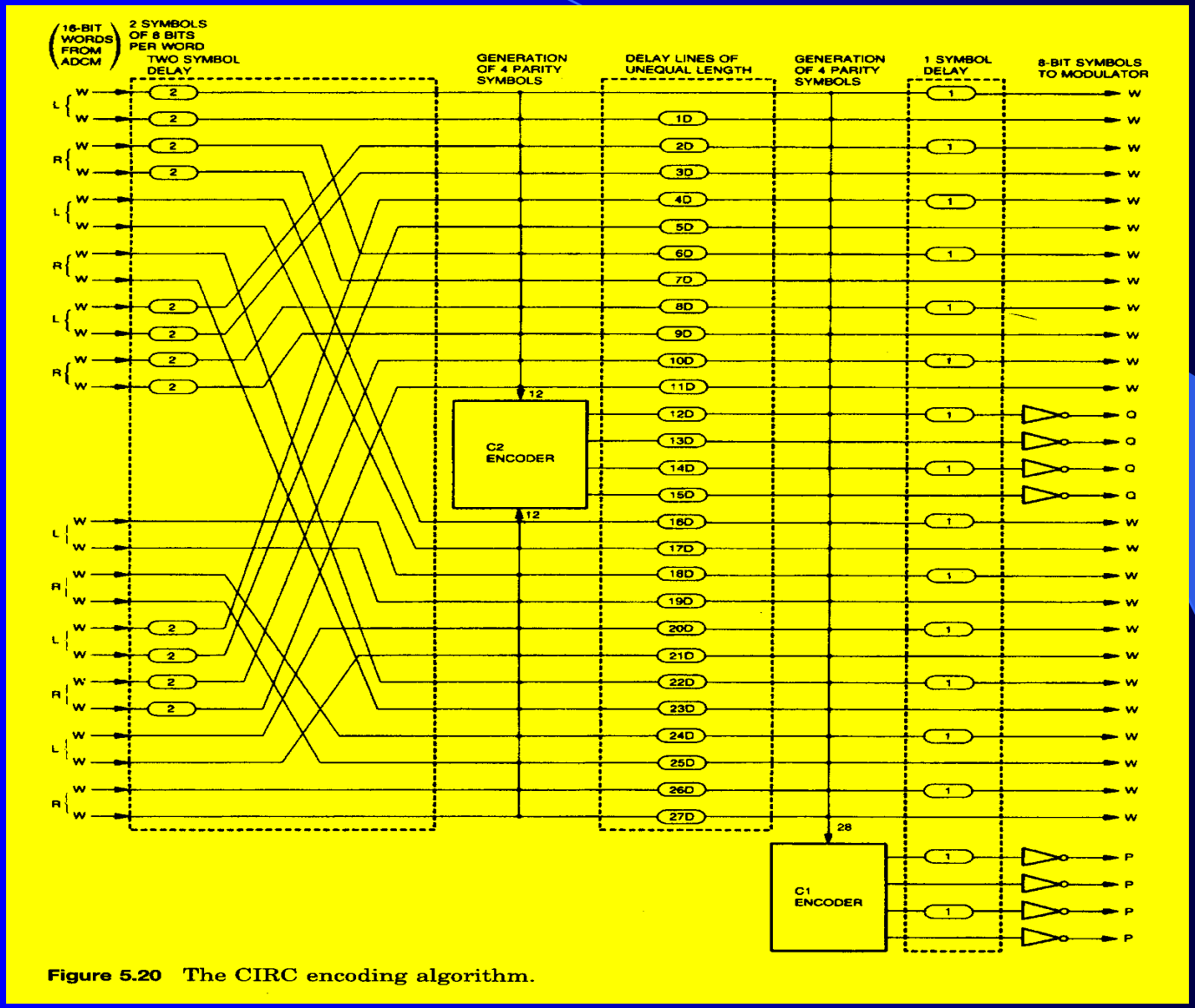


Figure 5.20 The CIRC encoding algorithm.

Cross-Interleave Reed-Solomon Codes

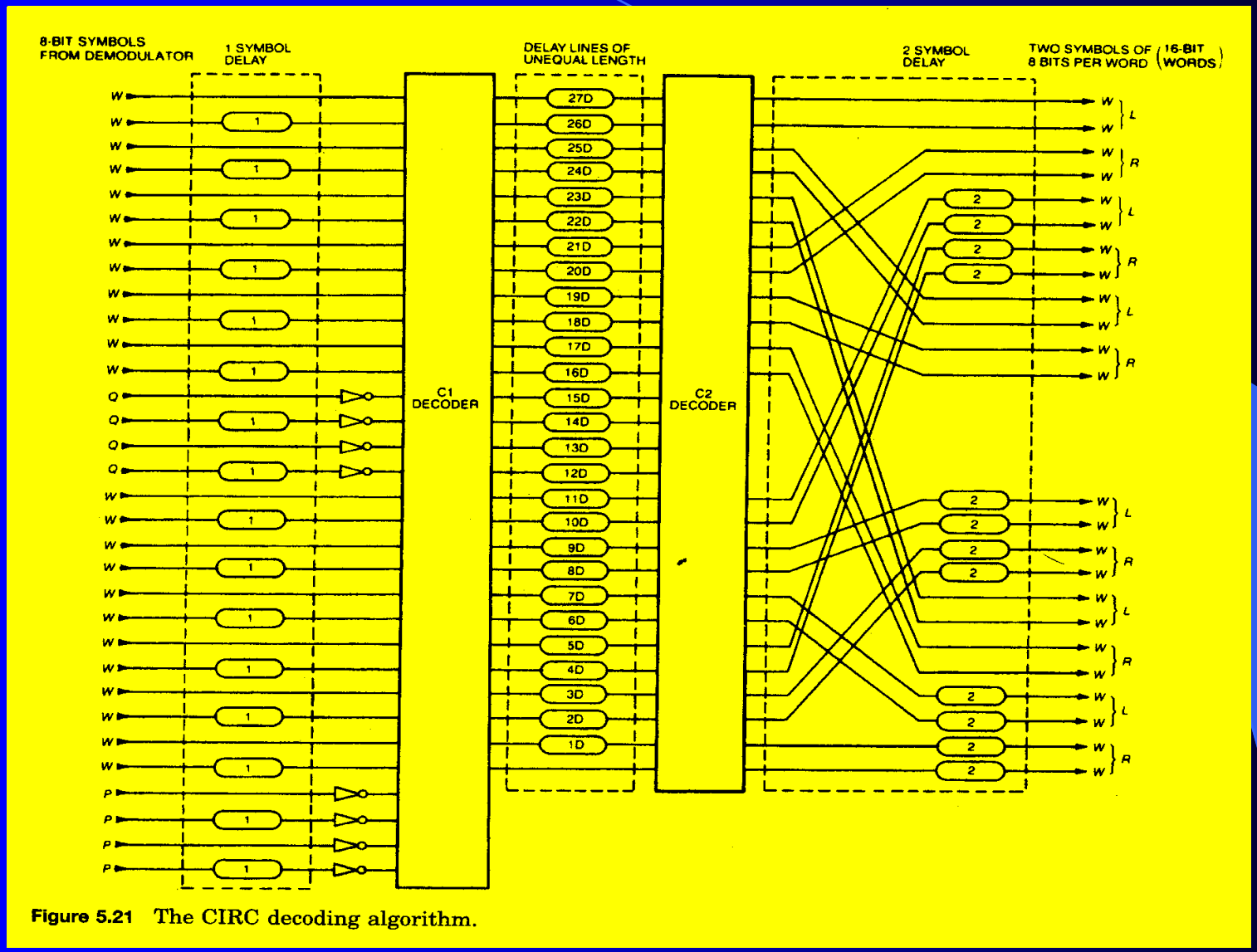


Figure 5.21 The CIRC decoding algorithm.

Cross-Interleave Reed-Solomon Codes

- Using four P parity symbols, the **C1** decoder corrects **random errors** and **detect burst**. The C1 decoder can correct **one erroneous symbol** in each frame.
- If there is more than one erroneous symbol, the 28 data symbols are **marked** with an erasure flag and passed to C2.
- Given pre-corrected data, and help from de-interleave, C2 can correct **burst errors** as well as **random errors** that C1 was unable to correct.
- When C2 cannot accomplish correction, the 24 data symbols are **flagged** and passed on for **interpolation**.
- For the compact discs, a two-digit nomenclature is used. The **first digit** specifies the **number of erroneous** symbols, and the **second digit** specifies at **which decoder** they occurred.

Cross-Interleave Reed-Solomon Codes

- Three error counts (**E11, E21** and **E31**) are measured at the output of the **C1 decoder**.

1. E11 : the frequency of occurrence of single symbol (correctable) errors per second in the C1 decoder.
2. E21 : the frequency of occurrence of double symbol (correctable) errors in the C1 decoder.
3. E31 : the frequency of triple symbol (uncorrectable) errors in the C1 decoder.

- The **block error rate** (**BLER**) = **E11 + E21 + E31**
- The CD standard sets a maximum of **220** BLER errors per second averaged over 10 seconds.
- The CD block rate is 7350 blocks per second; hence 220 BLER errors per second shows that **3%** of frames contains a defect.

Cross-Interleave Reed-Solomon Codes

- There are three error count (**E12, E22**, and **E32**) at the **C2 decoder**.
- **E12** count indicates the frequency of occurrence of a single symbol (correctable) error in the C2 decoder.
- A high E12 is not problematic because one E31 error can generate up to 30 E12 errors due to interleaving.
- **E22** count indicates the frequency of double symbol (correctable) error in the C2 decoder.
- E22 errors are the **worst correctable errors**.
- **E32** count indicates triple-symbol (uncorrectable) errors in the C2 decoder. **E32 should never occur** on a disc.
- The E21 and E22 signals can be combined to form a **burst error** (**BST**) count. This count is often tabulated as a **total number over an entire disc**.

Error Concealment

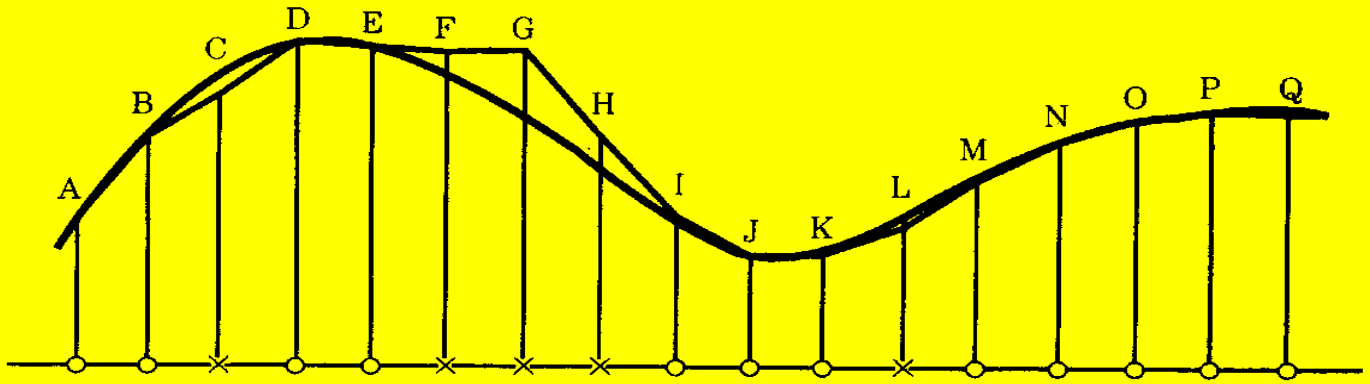
- A practical error-correction method allows **severe errors** to remain uncorrected. However, subsequent processing – an **error-concealment system –** compensates for those errors and ensures that they are **not audible**.
- There are two kinds of uncorrectable errors :

1. **Detected but not corrected** : can be **concealed** with properly designed concealment methods.
2. **Undetected and miscorrected** : cannot be concealed and might result in an **audible click** in the audio output.

Interpolation

- Following de-interleaving, most error are **interspersed** with valid data word. It is thus reasonable to use techniques in which **surrounding valid data** is used to calculate new data to replace the missing or incorrect data.
- **Zero-orde**r or **pervious-value** interpolation : Interpolation holds the previous sample value and replaces it to cover the missing or incorrect sample.
- **First-order** or **linear-order** interpolation : the erroneous sample is replaced with a new sample derived from the **mean value** of the previous and subsequent samples.
- In many digital audio systems, a **combination** of zero- and first-order interpolation is used.
- Other higher-order interpolation is sometimes used.

Interpolation



(Thick lines for analog input data. × is for error data)

Interpolation by maintenance of previous value

$G = F(=E)$

Interpolation by maintenance of mean value

$$C = \frac{1}{2}(B + D)$$

$$H = \frac{1}{2}(G + I) = \frac{1}{2}(E + I)$$

$$L = \frac{1}{2}(K + M)$$

Figure 5.22 Interpolation is used to conceal errors. For example, a previous value can be held, followed by calculation of the mean value.

Muting

- **Muting** is the simple process of setting the value of missing or uncorrected words to zero.
- Muting might be used in the case of **uncorrected errors**, which would otherwise cause an **audible click** at the output.
- Also in the case of **severe data damage** or **player malfunction**, it is preferable to mute the data output.
- To minimize audibility of a mute, muting algorithms **gradually attenuate** the output signal's amplitude prior to a mute, and then gradually restore the amplitude afterward.
- Such muting for **1 to 4 ms** durations cannot be perceived by the human ear.