syndrome is 110, or 6). By complementing the bit in position 6 of the received word, we determine that the correct word is 0001011.

A distance-3 Hamming code can easily be modified to increase its minimum distance to 4. We simply add one more check bit, chosen so that the parity of all the bits, including the new one, is even. As in the 1-bit even-parity code, this bit ensures that all errors affecting an odd number of bits are detectable. In particular, any 3-bit error is detectable. We already showed that 1- and 2-bit errors are detected by the other parity bits, so the minimum distance of the modified code must be 4.

Distance-3 and distance-4 Hamming codes are commonly used to detect and correct errors in computer memory systems, especially in large mainframe computers where memory circuits account for the bulk of the system's failures. These codes are especially attractive for very wide memory words, since the required number of parity bits grows slowly with the width of the memory word, as shown in Table 2-15.

Table 2-15 Word sizes of distance-3 and distance-4 Hamming codes.

	Minimum-distance-3 Codes		*Minimum-distance-4 Codes*	
Information Bits	***Parity Bits***	***Total Bits***	***Parity Bits***	***Total Bits***
1	2	3	3	4
≤ 4	3	≤ 7	4	≤ 8
≤ 11	4	≤ 15	5	≤ 16
≤ 26	5	≤ 31	6	≤ 32
≤ 57	6	≤ 63	7	≤ 64
≤ 120	7	≤ 127	8	≤ 128

2.15.4 CRC Codes

Beyond Hamming codes, many other error-detecting and -correcting codes have been developed. The most important codes, which happen to include Hamming codes, are the *cyclic redundancy check (CRC) codes*. A rich set of knowledge has been developed for these codes, focused both on their error detecting and correcting properties and on the design of inexpensive encoders and decoders for them (see References).

cyclic redundancy check (CRC) code

Two important applications of CRC codes are in disk drives and in data networks. In a disk drive, each block of data (typically 512 bytes) is protected by a CRC code, so that errors within a block can be detected and, in some drives, corrected. In a data network, each packet of data ends with check bits in a CRC

Copyright © 1999 by John F. Wakerly Copying Prohibited

code. The CRC codes for both applications were selected because of their burst-error detecting properties. In addition to single-bit errors, they can detect multi-bit errors that are clustered together within the disk block or packet. Such errors are more likely than errors of randomly distributed bits, because of the likely physical causes of errors in the two applications—surface defects in disc drives and noise bursts in communication links.

2.15.5 Two-Dimensional Codes

two-dimensional code

Another way to obtain a code with large minimum distance is to construct a *two-dimensional code*, as illustrated in Figure 2-14(a). The information bits are conceptually arranged in a two-dimensional array, and parity bits are provided to check both the rows and the columns. A code C_{row} with minimum distance d_{row} is used for the rows, and a possibly different code C_{col} with minimum distance d_{col} is used for the columns. That is, the row-parity bits are selected so that each row is a code word in C_{row} and the column-parity bits are selected so that each column is a code word in C_{col}. (The "corner" parity bits can be chosen according to either code.) The minimum distance of the two-dimensional code is the product of d_{row} and d_{col}; in fact, two-dimensional codes are sometimes called *product codes*.

product code

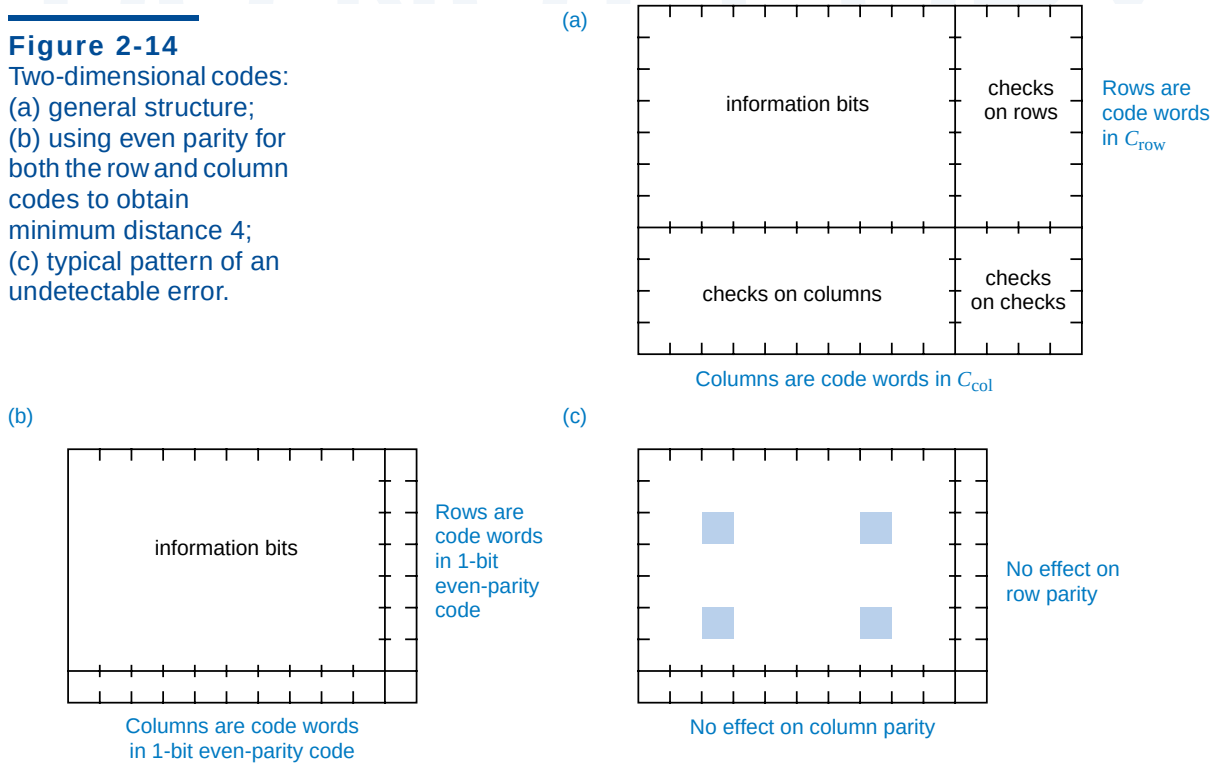

Figure 2-14
Two-dimensional codes: (a) general structure; (b) using even parity for both the row and column codes to obtain minimum distance 4; (c) typical pattern of an undetectable error.

Copyright © 1999 by John F. Wakerly Copying Prohibited

As shown in Figure 2-14(b), the simplest two-dimensional code uses 1-bit even-parity codes for the rows and columns, and has a minimum distance of $2 \cdot 2$, or 4. You can easily prove that the minimum distance is 4 by convincing yourself that any pattern of one, two, or three bits in error causes incorrect parity of a row or a column or both. In order to obtain an undetectable error, at least four bits must be changed in a rectangular pattern as in (c).

The error detecting and correcting procedures for this code are straightforward. Assume we are reading information one row at a time. As we read each row, we check its row code. If an error is detected in a row, we can't tell which bit is wrong from the row check alone. However, assuming only one row is bad, we can reconstruct it by forming the bit-by-bit Exclusive OR of the columns, omitting the bad row, but including the column-check row.

To obtain an even larger minimum distance, a distance-3 or -4 Hamming code can be used for the row or column code or both. It is also possible to construct a code in three or more dimensions, with minimum distance equal to the product of the minimum distances in each dimension.

An important application of two-dimensional codes is in RAID storage systems. *RAID* stands for "redundant array of inexpensive disks." In this scheme, n+1 identical disk drives are used to store n disks worth of data. For example, eight 8-Gigabyte drives could be use to store 64 Gigabytes of nonredundant data, and a ninth 8-gigabyte drive would be used to store checking information.

RAID

Figure 2-15 shows the general scheme of a two-dimensional code for a RAID system; each disk drive is considered to be a row in the code. Each drive stores m blocks of data, where a block typically contains 512 bytes. For example, an 8-gigabyte drive would store about 16 million blocks. As shown in the figure, each block includes its own check bits in a CRC code, to detect errors within that block. The first n drives store the nonredundant data. Each block in drive n+1

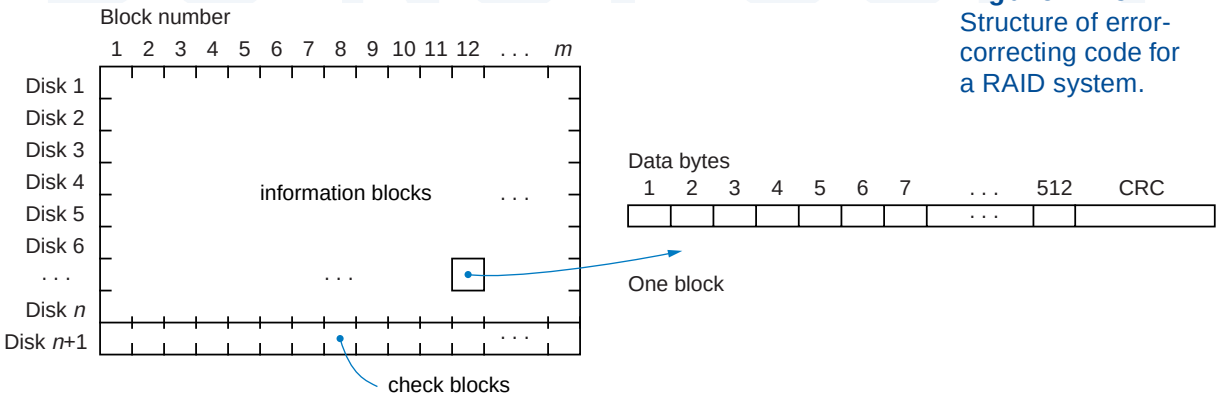

Figure 2-15 Structure of error-correcting code for a RAID system.

Copyright © 1999 by John F. Wakerly Copying Prohibited

stores parity bits for the corresponding blocks in the first n drives. That is, each bit i in drive n+1 block b is chosen so that there are an even number of 1s in block b bit position i across all the drives.

In operation, errors in the information blocks are detected by the CRC code. Whenever an error is detected in a block on one of the drives, the correct contents of that block can be constructed simply by computing the parity of the corresponding blocks in all the other drives, including drive n+1. Although this requires n extra disk read operations, it's better than losing your data! Write operations require extra disk accesses as well, to update the corresponding check block when an information block is written (see Exercise 2.46). Since disk writes are much less frequent than reads in typical applications, this overhead usually is not a problem.

2.15.6 Checksum Codes

The parity-checking operation that we've used in the previous subsections is essentially modulo-2 addition of bits—the sum modulo 2 of a group of bits is 0 if the number of 1s in the group is even, and 1 if it is odd. The technique of modular addition can be extended to other bases besides 2 to form check digits.

For example, a computer stores information as a set of 8-bit bytes. Each byte may be considered to have a decimal value from 0 to 255. Therefore, we can use modulo-256 addition to check the bytes. We form a single check byte, called a *checksum*, that is the sum modulo 256 of all the information bytes. The resulting *checksum code* can detect any single *byte* error, since such an error will cause a recomputed sum of bytes to disagree with the checksum.

checksum
checksum code

Checksum codes can also use a different modulus of addition. In particular, checksum codes using modulo-255, ones'-complement addition are important because of their special computational and error detecting properties, and because they are used to check packet headers in the ubiquitous Internet Protocol (IP) (see References).

ones'-complement checksum code

2.15.7 *m*-out-of-*n* Codes

The 1-out-of-n and m-out-of-n codes that we introduced in Section 2.13 have a minimum distance of 2, since changing only one bit changes the total number of 1s in a code word and therefore produces a noncode word.

These codes have another useful error-detecting property—they detect unidirectional multiple errors. In a *unidirectional error*, all of the erroneous bits change in the same direction (0s change to 1s, or vice versa). This property is very useful in systems where the predominant error mechanism tends to change all bits in the same direction.

unidirectional error

Copyright © 1999 by John F. Wakerly Copying Prohibited