Unit 8 Error detection codes.

- Errors of transmission or errors in memory storage can be detected by checking parity bits.
- ASCII coding is used for transmission and storage of characters.
- The Hamming distance is a measure of the difference between two code segments.
- Hamming codes contain more error check bits which allow two or more errors to be detected and corrected.

The Gray code, which was discussed in Unit 7, is used to avoid apparent discontinuous glitches which can occur in the measurement of angular or linear position. In the case of the Gray code and the other codes which were described in Units 6 and 7, it has been presumed that no errors occur in the transmission or storage of the numbers. Unfortunately, errors do occur and it is important to detect the errors and, if possible, to correct the errors.

The essential feature of any error detecting coding system is that extra information, usually in the form of extra bits or parity bits, is included within an extended number and this information is used for error checking and possible correction.

One of the most frequently used error detection systems is based on the use of **parity** checking. If, for instance, the data word is 8 bits long, then an extra bit is added to give a 9 bit word. The value of this bit is assigned to be 0 or 1 so that the sum of the 9 bits, including the parity bit, is either odd for an odd parity system and even for an even parity system. Thus, if an even parity convention is in operation and the data word is 10011011, we add the 1s in the word to get an odd number (5). We then have to put a 1 in the parity position to obtain an overall even parity word so the data word with parity appended then becomes 100110111, nine bits for which the sum of the 1s is even.

When parity checking is used on a communications link, both ends of the link agree to employ a particular parity, say odd parity. The transmitting computer then puts a 0 or a 1 bit in the parity bit position so as to make the parity odd. At the receiving end the computer checks the parity of the received word and compares the value with the previously agreed convention. If there is agreement, then the received word is assumed to be valid. This system is not infallible, however. Consider the situation where two bits are corrupted during the transmission. The parity check is valid even though the word was corrupted. Once an error is detected the system halts as there is no possibility of correcting the error with parity checking alone.

One specific use of parity checking is with the ASCII (American Standard Code for Information Interchange) where 8 binary bit or two digit hexadecimal numbers from 00 to FF (numbers between 0 and 255 decimal) are used to represent the digits 0 to 9, the characters a to z and capitals A to Z together with other punctuation and control characters and foreign language characters. In other words, the ASCII set represents the full alphanumeric character set. The use of the ASCII codes allows the transmission of text between computers using modems for the transmission of text between computers or from computers and printers. For reference, the ASCII codes are given in Appendix A. Note that the codes for the numbers begin with 30_{H} for "0"; the codes for the letters begin with 41_{H} for "A" and 61_{H} for "a". The ASCII code only defines the assignment of numbers to represent text characters. Because communication links often introduce errors there is usually an extra parity bit used to check the integrity of the transmission link. The parity generation at the transmission end and the parity checking at the receive end is carried out by custom chips designed for use in communication links.

While parity checking is the most common error checking method and is also the most efficient in terms of the numbers of bits which are used, there are other error systems in use. One system is the "M out of N code" system in which each code word has n bits and m of these bits are 1 s. The number of distinct codes is then the number of combinations of n objects taken m at a time or C_m^n. In the case of 2 out of 5 code, there are 10 distinct codes which can be used to represent the decimal digits. This m out of n scheme will always detect an error caused by a single 1 to 0 or 0 to 1 conversion and will detect some but not all double conversion errors.

The Berger codes are a development of the m out of n codes in which the number of 1 s in the data part of the code is obtained in binary. This binary number is complemented and appended to the code word. For example the data 0110111 contains 5 1 s represented by 0101 in binary which when complemented is 1010. This is now appended to the data word to give the Berger code representation, 01101111010.

We have seen that, if some redundancy is permitted, it is possible to detect the occurrence of errors but that it is not possible to identify where the error has occurred. Therefore it is not possible to repair the errors. Transmitting the data twice followed by comparison of the two received versions of the data will allow errors to be detected but will still not allow correction of errors since it is not possible to decide which version of the data contains the error. If parity checks are used and a retransmit request is sent in the event of a parity error being detected then the data can be corrected but this requires two way communication and there may not be enough time available for retransmission. If the data and parity checks are sent twice then it is possible to do parity checks on each version and also decide which version contains the errors. The data rate transmission is however reduced by a factor of two or more and this strategy is only feasible for very important data transmission or storage. What is really needed is a method of adding on a small extra segment of code which can be used to correct errors but which will not cause a significant increase in the transmission time or storage requirement. One such strategy is based on the use of the **Hamming distance** which is the measure of the difference between two data words.

If there are n bits in a code word then there are 2^n possible distinct words. If a subset is chosen, as representing valid words, which all differ in at least 2 or more bit positions, then it is possible to detect the occurrence of a single bit error. If extra check bits are added then it is possible to detect which bit is in error and repair that bit.

The fundamental concept which is used is that of the Hamming distance between two words which is the number of bit differences between two code words. The **minimum Hamming distance** of a set of code words is the minimum number of bits of a code word that must be changed in order to convert one valid code word of the set into another valid code word of the set.

Consider the subset of 4 bit code words shown below:

x		0100
y		0011
z		1010
w		1001
then x and y	differ in	3 bit positions
x and z	differ in	3 bit positions
x and w	differ in	3 bit positions
y and z	differ in	2 bit positions
y and w	differ in	2 bit positions
z and w	differ in	2 bit positions

and this set of code words has a minimum Hamming distance of 2, that is $d = 2$. It can be shown by a theoretical analysis that if C is the number of bit errors that can be corrected and D is the number of bit errors that can be detected then:

$$d = C + D + 1 \text{ and } D \geq C$$

In the 4 bit code words shown above, $d = 2$ so that $C + D = 2 - 1 = 1$ and since $D \geq C$ the only valid solution is $C = 0$ and $D = 1$ so that we can detect a single error bit but we cannot correct the error.

If the number of bits in the data word is increased and the size of the selected valid code word set is also increased, then larger Hamming distances between the code words are obtained.

Extra check bits are appended to the data word which are used to correct for errors. If there are m bits in the data word and there are p check bits appended then the required number of bits for correction of errors in the $m + p$ bits of the code word is given by the value of p which satisfies the **Hamming relationship**:

$$2^p \geq m + p + 1$$

The resulting code is called the **Hamming code**. Note that this criterion applies for correction of single errors only. Other criteria apply for correction of greater numbers of errors. The table below shows the values of p for various sizes of data words, m.

m	4	5	6	7	8	9	10	16	32	200
p	3	3	4	4	4	4	4	5	6	8

It can be seen from this table that there is no great advantage of using Hamming codes for small code words but that the efficiency in minimizing the amount of redundant data that must be transmitted and the efficiency in correcting errors is greater for larger data words than for smaller data words. There is, however, the constraint that the probability of multiple errors must be vanishingly small because this criteria is valid only for single error correction.

The parity bits which comprise the check bits can be generated from the bits of the Hamming code word using a variety of schemes but the two principal requirements are that the set of parity checks should be independent of each other (the columns of the Hamming code matrix, $\mathcal{H}$, should be linearity independent) and that the codes should permit the correction of errors.

In constructing the Hamming code, the convention is that the kth check bit is put in the $2^{(k-1)}$ th position in the Hamming code word and therefore, if the data bits are designated d and the parity bits designated d, the Hamming code will be in the form:

Bit function	p_1	p_2	d_1	p_3	d_2	d_3	d_4
Position number	1	2	3	4	5	6	7
Position number	001	010	011	100	101	110	111
Parity of bits numbered	1357	2367		4567			

To form a parity bit; take the parity bit position number (row 3) and locate the single 1 bit. The parity is formed using the bits in the code for which the position number (in binary) has a 1 in the same place as the parity bit. The fourth row shows the numbers of the bits which are used to form the parity bit at that position. The procedures for forming the Hamming code for a data word and also for correcting the code when an error has occurred are given in Examples 8.1 and 8.2.

The Hamming code is only one of many types of error correcting codes which are used. For more detailed treatments the student is referred to the more specialist works on error correcting codes given in the references and in particular, the text by MacWilliams and Sloane.

8.1 References

Floyd Thomas L (1990), *Digital Fundamentals*, Maxwell MacMillan.

Rhee, Man Young (1989), *Error Correcting Coding Theory*, McGraw Hill.

Rao T. R. N., Fujiwara E. (1989), *Error-control Coding for Computer Systems*, Prentice Hall.

MacWilliams, F.J., Sloane NJA (1977), *The Theory of Error-Correcting Codes*, North Holland.

8.2 Examples

8.1 Form the 7 bit Hamming code for the data word 0111 using even parity.

Write down the data bits and the unknown parity bits represented by p_1, p_2, p_3, as shown in row 1 of the table. Write in rows 2, 3, 4 the bits from which the parity is to be calculated with p_x representing the unknown parity. Determine the value of p_x which makes the row even parity. Form the Hamming code as shown in row 5 of the table.

p_1	p_2	0	p_3	1	1	1
p_1		0		1		1
	p_2	0			1	1
			p_3	1	1	1
0	0	0	1	1	1	1

8.2 Recover the correct data word from the received 7 bit code 0111001, given that odd parity is in use.

The procedure is the reverse of the encoding shown in Example 8.1.

0	1	1	1	0	0	1	received
0		1		0		1	error
	1	1			0	1	no error
			1	0	0	1	error
0	1	1	1	1	0	1	corrected

Put the received Hamming code in row 1 of the table. Rows 2, 3 and 4 now contain the bits for which parity is to be checked. Rows 2 and 4 show parity errors. To identify the location of the error, note that the no error in row 3 implies that bits 2, 3, 6 and 7 are correct. Bit 5 is the only bit which is common to rows 2 and 4 and has not already been shown to be correct. Bit 5 is therefore the incorrect bit.
The corrected Hamming code word is then shown in row 5.

8.3 Problems

8.1 If a communication link has a probability of an error in a particular bit of 10^{-6} , calculate the probability of 1 error occurring in an 8 bit code word.

8.2 If a (bad) communication link has a probability of an error in a particular bit of 0.001, calculate the probabilities of 1 and 2 errors occurring in an 8 bit code word.

8.3 Give the 10 binary representations for the possible 2 out of 5 codes. Assign weights on a 63210 scheme and assign the digits 1 to 9 to the appropriate code.

8.4 Generate the 7 bit Hamming code word from the data word 0110 using odd parity.