CHANNEL CODING

Introduction:

The purpose of channel coding is-

1-either to protect information from channel noise, distortion and jamming, which is the subject of error detecting and correcting codes. Or,

2-to protect information from the 3rd party (enemy) which is the subject of encryption, scrambling.

In this course, only error detecting and correcting codes are discussed.

ERROR DETECTING AND CORRECTING CODES

Concept:

The basic idea behind error detecting or correcting codes is to add extra bits (or digits) to the information such that the receiver can use it to detect or correct errors with limited capabilities. These extra redundant bits are called parity or check or correction bits. So, if for each k digits, r parity digits are added then, the transmitted k+r=n digits will have r redundant digits and the code is called (n,k) code with code efficiency or rate of (k/n). In general, the ability of detection or correction depends on the techniques used and the n, k parameters.

ERROR DETECTING CODES

Simple error detecting codes:

The simplest error detection schemes are the well- known even and odd parity generators. For even parity, an extra bit is added for each k information bits such that the total number of 1's is even. At the receiver, an error is detected if the number of 1's is odd. However, if the number of 1's is even, then either no error occurs or even number of errors occur. Hence:

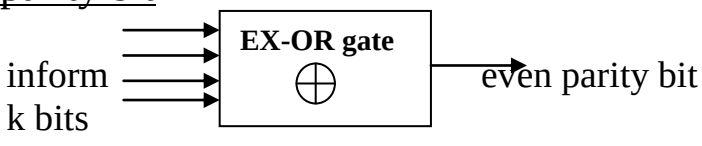
probability(detecting errors)=probability(odd number of errors) and :

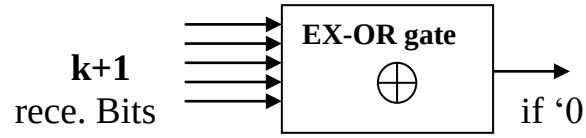
probability.(undetected errors)=probability(even number of errors).

The same idea can be applied when number of 1's is adjusted to be odd. The code rate (efficiency) is k/(k+1).

To implement these parity generators, simple Ex-OR gates are used at TX and RX as shown below:

Information bits	even parity bit
1 0 1 0 1 1 1	1
0 1 1 0 1 0 1	0
1 0 1 1 1 1 1	0
1 1 0 1 0 0 0	1





Hence, we can conclude that error detection is not ideal. It does not detect errors with 100% probability , since even number of errors behaves exactly the same as no error.

Example: for even parity check code, if K=7 and bit error probability (bit error rate BER) is 10^{-3} , find the prob.of detected and undetected errors.

$$P(undet \quad error) = C_2^{\ 8} p^2 (1-p)^6 + C_4^{\ 8} p^4 (1-p)^4 + C_6^{\ 8} p^6 (1-p)^2 + C_8^{\ 8} p^8 \approx 28 \times 10^{-6}$$

for p<<1. $C_n^{\ m}$ is the combination factor $C_n^{\ m} = \dfrac{m!}{n!\,(m-n)!}$

The probability of detected errors will be:

$$P(detected \quad errors) = C_1^{\ 8} p^1 (1-p)^7 + C_3^{\ 8} p^3 (1-p)^5 + C_5^{\ 8} p^5 (1-p)^3 + C_7^{\ 8} p^7 (1-p) \approx 8 \times 10^{-3}$$

P(undetected errors) < P(detected errors)

Note that, although the code used for detection is so simple (few EX-OR gates) but still we have big advantage since probability of detecting errors is much higher than probability of undetected errors. The advantage of error detection is clear when used together with ARQ (Automatic Repeat Query) systems. In these systems, two channels are used, the usual forward channel with error detection and a backward channel. Data are transmitted through the forward channel. These data are protected against errors with parity error detection. If the receiver detects errors then a backward channel will be used to inform the transmitter to retransmit(repeat) the same data so that in the next transmission, data is received correctly since errors occur randomly (may occur or may not occur).

ERROR CORRECTING CODES

In order to make the receiver have the ability to detect and correct errors, then not only a single parity bit is used, but in stead r bits are used giving what is called the (n,k) code.

Basic definitions:

1-systematic and nonsystematic codes: If information bits (a's) are unchanged in their values and positions at the transmitted codeword, then this code is said to be systematic.

Input data [D]=[a_1 a_2 a_3 …….a_k],

Output systematic (n,k) codeword is [C]=[a_1 a_2 a_3 …….a_k c_1 c_2 c_3 …….c_r]

However if data bits are spread or changed at the output codeword then, the code is said to be nonsystematic:

Output nonsystematic(7,4) codeword is [C]=[c_2 a_1 c_3 a_2 c_1 a_4 a_3]

2- Hamming distance: it is the number of differences between corresponding bits and the ability of error detection and correction codes depends on this parameter. The Hamming distance between two codewords C_i and C_j is denoted by d_{ij} which is the number of bits that differ. For a binary (n,k) code with 2^k possible codewords then the minimum Hamming distance (HD) is the min(d_{ij}). Of course

$n \geq d_{ij} \geq 0$.

Example: Find the Hamming distance between the two codewords: [C1]=[1011100] and

[C2]=[1011001].

Solution: Here, the no. of bits that differ is 2, hence $d_{12}=2$.

Example: Find the minimum Hamming distance for the 3 codewords:

[C1]=[1011100],

[C2]=[1011001]

[C3]=[1011000].

Solution: Here $d_{12}=2$, $d_{13}=1$ and $d_{23}=1$. Hence min(d_{ij})=1=(HD). Note that the calculation of HD becomes more difficult if no of codewords increases.

3-Hamming weight: This is the number of 1's in the non zero codeward C_i. It is denoted by ω_i . As will be shown later, and for linear codes, ω_{min}=HD=min(d_{ij}). This simplifies the calculation of HD. As an example, if [C1]=[1011000], then $\omega_1=3$, and for [C2]=[0001010], then $\omega_2=2$, and so on.

4-Linear and nonlinear codes: when the r parity bits are obtained from a linear function of the k information bits then the code is said to be linear, otherwise it is a nonlinear code.

Hamming Bound: The purpose of Hamming bound is either
1) to choose the number of parity bits (r) so that a certain error correction capability is obtained. Or
2) to find the error correction capability (t) if the number of parity bits (r) is known
For binary codes, this is given by:

$$\mathbf{2^{n-k} = 2^r \geq} \sum_{j=0}^{t} C_j^{\,n}$$

where t is the number of bits to be corrected.
Example: for a single correction code with k=4 find the no. of parity bits that should be added.
$2^r \geq C_0^{\,4+r} + C_1^{\,4+r}$. This gives $2^r \geq 1+(4+r)$ and the minimum r is r=3 (take minimum r to have max code efficiency). This is the (7,4) code. the code is said to be perfect code.
Perfect code:in Hamming bound ,if the equality is satisfied then this code is said to be a perfect code.
Example if k=5 and up to 3 errors are to be corrected, find the no. of check bits that should be added.
:
$2^r \geq C_0^{\,5+r} + C_1^{\,5+r} + C_2^{\,5+r} + C_3^{\,5+r}$ that gives:
$2^r \geq 1+(5+r)+(5+r)(4+r)/2+(5+r)(4+r)(3+r)/6$, then min r here is r=9, and the code is the (14,5) non perfect code(equal sign is not satisfied).

Note: If the (n,k) codewords are trans. through a channel having error prob=p_e, then prob. of decoding a correct word at the Rx for t-error correcting code will be:
P(correct words)=p(no error)+p(1 error)+……..p(t errors)

$$P(correct\ word) = \sum_{i=0}^{t} C_i^{\,n}\ p_e^{\,i}\ (1-p_e)^{n-i}$$

and prob(erroneous word)=1-P(correct word).

Hamming code:
The first example given above is the Hamming code. It is a single error correcting perfect code with the following parameters: $n=2^r-1$, HD=3, t=1. The (7,4), (15,11), (31,26) …..are examples of Hamming codes. Hamming codes are encoded and decoded as a linear block codes.

Notes:

1-A linear code can correct t=Int[(HD-1)/2] of random (isolated) errors and detect (HD-1) random(isolated errors).

2- HD is the min Hamming distance= ω_{min}

Linear Block Codes:

Only systematic binary codes will be described. The r parity bits are obtained using a linear function of the a's data. Mathematically, this can be described by the set of equations:

$$\left.\begin{aligned} C_1&=h_{11}a_1+h_{12}a_2+h_{13}a_3+\ldots\ldots..+h_{1k}a_k \\ C_2&=h_{21}a_1+h_{22}a_2+h_{23}a_3+\ldots\ldots..+h_{2k}a_k \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ C_r&=h_{r1}a_1+h_{r2}a_2+h_{r3}a_3+\ldots\ldots .+h_{rk}a_k \end{aligned}\right| \qquad \ldots\ldots\ldots\ldots\ldots..(1)$$

Where + is mod-2 addition (EX-OR), product is the AND multiplication and h_{ij} coefficients are binary variables for a binary coding. The complete output codeword can be written in matrix form as:

$$[C]= [D][G] \ldots\ldots\ldots(1) \qquad \text{, where:}$$

$$[G]=\begin{bmatrix} 1 & 0 & 0 & 0 & h_{11} & h_{21} & h_{31} & . & h_{r1} \\ 0 & 1 & 0 & 0 & h_{12} & h_{22} & h_{32} & . & h_{r2} \\ 0 & 0 & 1 & 0 & . & . & . & . & . \\ 0 & 0 & 0 & 1 & h_{1k} & h_{2k} & h_{3k} & . & h_{rk} \end{bmatrix} = [\ I_k : P_{kxr}\ ]$$ which is kxn matrix.

This matrix is called the generator matrix of the linear block code (LBC). Equation(1) can also be written in matrix form as:

$$[\ H\ ]\ [C]^T=[0] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

where: $[C]=[\ a_1\ a_2\ a_3\ \ldots\ldots a_k\ c_1\ c_2\ c_3\ \ldots\ldots c_r]$ and $[\ H\ ]$ matrix is in fact related with [G] matrix by:

$[\ H\ ]=[-P^T : I_r]$, and for binary coding this – sign drops out. This rxn [H] matrix is called the parity check matrix. As will be shown, encoding can be done either using eq(1) ([G] matrix) or eq(2) ([H] matrix), but decoding is done using [H] matrix only.

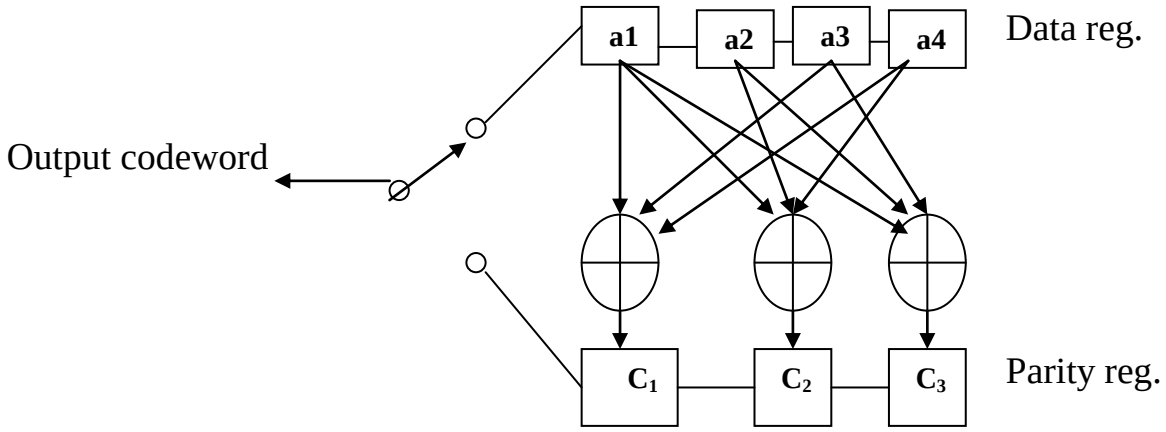
Encoding of Linear Block codes:

Example: a given binary (7,4) Hamming code with a parity check matrix:

$$[H]=\begin{bmatrix}1&0&1&1&1&0&0\\1&1&0&1&0&1&0\\1&1&1&0&0&0&1\end{bmatrix}$$, find: 1) encoding circuit 2) all possible codewords 3) error correction capability.

Solution : using eq(2), $[H][C]^T=[0]$ will give:

$C_1=a_1+a_3+a_4$, $C_2=a_1+a_2+a_4$, $C_3=a_1+a_2+a_3$.



Above equations for C's are used to find the code table for this code as:

a_1	a_2	a_3	a_4	c_1	c_2	c_3	ω_i
0	0	0	0	0	0	0	--
0	0	0	1	1	1	0	3
0	0	1	0	1	0	1	3
0	0	1	1	0	1	1	4
0	1	0	0	0	1	1	3
0	1	0	1	1	0	1	4
0	1	1	0	1	1	0	4
0	1	1	1	0	0	0	3
1	0	0	0	1	1	1	4
1	0	0	1	0	0	1	3
1	0	1	0	0	1	0	3
1	0	1	1	1	0	0	4
1	1	0	0	1	0	0	3
1	1	0	1	0	1	0	4
1	1	1	0	0	0	1	4
1	1	1	1	1	1	1	7

ωi(min)=3=HD, i.e. t= int(3-1)/2=1 bit. Hence, this is a single error correcting code(Hamming code).

Example: Find the generator matrix for the previous LBC.

Solution:

$$[G]=[I_k \;\; P^T]=\begin{bmatrix}1&0&0&0&1&1&1\\0&1&0&0&0&1&1\\0&0&1&0&1&0&1\\0&0&0&1&1&1&0\end{bmatrix}$$

Note that the equation [C]=[D][G] gives:

[C]=[a_1 a_2 a_3 a_4 (a_1+a_3+a_4) (a_1+a_2+a_4) (a_1+a_2+a_3)]=[a_1 a_2 a_3 a_4 c_1 c_2 c_3] as obtained before.

Decoding of linear block codes:

If [R]=[C]+[E] is the received codeword, where [E] is the error word, if [E]=[0] then no error occurs but if [E]=[0 0 …0 0 1 0] then single error occurs at 2nd position(from the right), or if [E]=[0 0 0…0 1 0 0 1 0 1] then triple errors occur at 1st, 3rd and 6th positions. Depending on t not all of these errors can be corrected. If [R] is multiplied by [H] (the receiver must know [H]) then:

[H] $[R]^T$=[H]$[C]^T$+ [H] $[E]^T$= [H]$[E]^T$ since [H]$[C]^T$ is set to [0] at the TX. Then define [S] vector :

[S]=[H] $[R]^T$= [H]$[E]^T$ ……..(3)

This [S] r-vector is called the syndrome. If [S]=[0], the RX decides on no error but if [S]≠[0], then the receiver must use [S] to find [E] and hence the corrected [C]=[R]+[E] binary coding). Of course, [S] is calculated from [R]. The problem is now how to find [E] from [S]?. In Eqs(4) we have n unknowns in r equations (n>r). To solve this problem, maximum likelihood criterion is used. i.e, most probable error words are chosen and usually the most probable errors are those with less number of errors. So the RX finds [E] that matches [S] such that the less number of errors solution is chosen.

Simple decoding procedure for single error: For single error Hamming codes, above mathematical solution is reduced into comparing the [S] r-vector with all columns of the [H] matrix (2^r-1 non zero and non repeated columns). That column similar to [S] is the position of error. This is mathematically equivalent to multiply [H]$[E]^T$ such that [E] has only one nonzero element at the ith position or at the ith column. Hence, for single error correction, **the parity check matrix [H] must satisfy the following:**

i. **No all zero columns so as not to mix with the no error case.**

ii. No repeated columns so that the decoder can decode any received word correctly with single error assumption.

Example: For previous example,[1]-Find the corrected word at the receiver, for the previous example, if the received word [R]=[1001111]. [2]-Find the syndrome vector if double errors occur at 1st and last positions, comment. [3]- Draw the decoder cct used to find the syndrome vector[S].
Solution:[1] If the received word is [R]=[1001111]

$$\text{then, } [H][R]^T = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = [S]$$

which is similar to the 4th column in [H].
Hence the corrected word=[R]+[0001000]= [1 0 0 0 1 1 1]
which checks with the table shown besides.

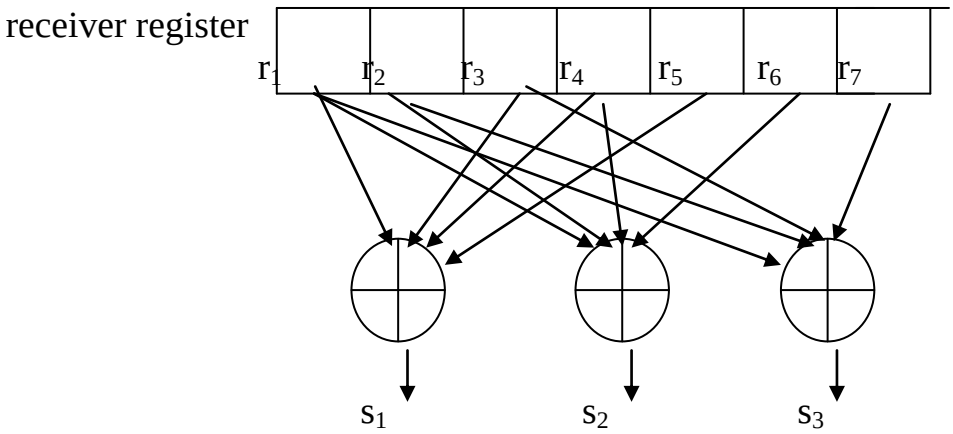
[2]-To find the syndrome vector[S] for double errors, then $[S]=[H][E]^T$.
Where [E]=[1000001] corresponding to double errors at 1st and last positions. Then:

$$[S] = [H][E]^T = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Note that the syndrome for single error at the 4th position is the same as the syndrome for double errors at 1st and last positions. This indicates that the code is only capable of correcting single error as expected.
[3]- To draw the decoder cct, then :

$$[S]=[H][R]^T=\begin{bmatrix}1&0&1&1&1&0&0\\1&1&0&1&0&1&0\\1&1&1&0&0&0&1\end{bmatrix}\begin{bmatrix}r_1\\r_2\\r_3\\r_4\\r_5\\r_6\\r_7\end{bmatrix}=\begin{bmatrix}s_1\\s_2\\s_3\end{bmatrix}$$ which gives:

$s_1=r_1+r_3+r_4+r_5$, $s_2=r_1+r_2+r_4+r_6$, $s_3=r_1+r_2+r_3+r_7$ implemented as shown:



Example:

The generator matrix of a LBC is given by:

$$[G]=\begin{bmatrix}1&0&0&1&1&0&1&1&0&0\\0&1&0&0&1&1&0&1&1&0\\0&0&1&0&0&1&1&1&0&1\end{bmatrix}$$

a-Use Hamming bound to find error correction capability. b-Find the parity check matrix. c-find the code table, Hamming weight and the error correction capability then compare with part(a). d-If the received word is [R]=[1011110011], find the corrected word at the Rx.

Solution: (a) n=10, k=3, r=7 , (10,3) code. Using Hamming bound, then:

$2^7 \geq C_0^{10}+C_1^{10}+C_2^{10}+\ldots\ldots\ldots+C_t^{10}$ that gives 128>1+10+(10*9/2), i.e t=2 double error correction.

b-

$$[H]=[P^T\ I]=\begin{bmatrix}1&0&0&1&0&0&0&0&0&0\\1&1&0&0&1&0&0&0&0&0\\0&1&1&0&0&1&0&0&0&0\\1&0&1&0&0&0&1&0&0&0\\1&1&1&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&0&1&0\\0&0&1&0&0&0&0&0&0&1\end{bmatrix}$$ with no 'zero' or repeated columns.

The equation $[H][C]^T=[0]$ gives $c_1=a_1$, $c_2=a_1+a_2$ and $c_3=a_2+a_3$, $c_4=a_1+a_3$, $c_5=a_1+a_2+a_3$ $c_6=a_2$, $c_7=a_3$.

a1	a2	a3	c1	C2	c3	c4	c5	c6	c7	wi
0	0	0	0	0	0	0	0	0	0	---
0	0	1	0	0	1	1	1	0	1	5
0	1	0	0	1	1	0	1	1	0	5
0	1	1	0	1	0	1	0	1	1	6
1	0	0	1	1	0	1	1	0	0	5
1	0	1	1	1	1	0	0	0	1	6
1	1	0	1	0	1	1	0	1	0	6
1	1	1	1	0	0	0	1	1	1	7

ωi(min)=5=HD, i.e. t= int(5-1)/2=2 bits. Hence, this is a double error correcting code which checks with part(a).

d-If [R]=[1011110011], then:

$$[S]=[H][R]^T=\begin{bmatrix}1&0&0&1&0&0&0&0&0&0\\1&1&0&0&1&0&0&0&0&0\\0&1&1&0&0&1&0&0&0&0\\1&0&1&0&0&0&1&0&0&0\\1&1&1&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&0&1&0\\0&0&1&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}1\\0\\1\\1\\1\\0\\0\\1\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\end{bmatrix}$$

which is similar to the 9th. column in [H](from the left), hence corrected word will be [1011110001].