3) Maximum Likelihood Decoding (MLD): Decoding

The decoder must produce an estimate $\hat{u}$ of the inf sequence u based on the received sequence r, (or an estimate $\hat{v}$ of v since there is a one to one correspondance between u & v).
This is done by minimizing the distance $d(r, v)$ bet r & v; that is it chooses the codeword that differs from the received sequence in the fewest number of positions.

- distance bet 2 codewords = number of bit locations in which the codewords differ:

s_1 101101 s_2 100111

$d(s_1, s_2) = 2$.

(A MLD for the BSC is sometimes called a minimum distance decoder).

4) Types of Errors:

Transmission errors in a digital com. system are caused by the noise in the com. channel. Generally two kinds of noise can be distinguished in a com. channel. The first, Gaussian noise. Its sources are thermal noise in the channel and radiation picked up by the receiving antenna. (The p.s.d. of Gaussian noise at the Rx input is white). The transmission errors caused by white noise are such that noise affects each bit (transmitted symbol) independently. These are random errors. Examples of random error channels are the deep-space channel & many satellite channels. The codes devised for correcting random errors are called random error correcting codes.

A second type of noise is called impulse noise, which is characterized by long quiet intervals followed by high amplitude noise bursts. As a consequence, transmission errors occur in clusters or bursts (more than one bit). Examples of burst error channels are radio channels where the error bursts are caused by signal fading (fading = random variations in amplitude & phase of the signal), and in cable transmission which is affected by impulsive switching noise and crosstalk.
The codes devised for correcting burst errors are called burst-error correcting codes.

* Some channels contain a combination of both random and burst errors. ⟶ compound channels. Codes used are called burst and random error correcting codes.

5) Error Control Strategies:

The block diagram of a com. system represents a one-way system (from Tx to Rx). Error control for a one-way syst. must be accomplished using forward error correction (FEC), that is, by employing error correcting codes that automatically correct errors detected at the Rx (e.g. in deep space com. systems, where the relatively simple encoding equipment can be placed abroad the space craft, but the much more complex decoding equipment must be performed on earth).
In some cases, a transmission system can be two-way; that is the inf. can be sent in both directions & the Tx acts as a Rx (transceiver), and vice versa. Examples of two-way systems are telephone

channels and some satellite com. systems.
Error control for a two-way syst can be accomplished using error detection & retransmission called ARQ (Automatic Repeat Request). In an ARQ system, when errors are detected at the Rx, a request is sent for the Tx to repeat the message, and this continues until the message is received correctly.

Linear Block Codes.

1) Introduction :

For ease of code synthesis & implementation, we restrict our attention to a subclass of the class of all block codes, the linear block codes (strictly binary ⟶ the theory developed for binary codes can be generalized to codes from a nonbinary field in a straightforward manner),

[If a b.c. is linear, the encoding complexity is greatly reduced].

Def 1: A binary b.c. is linear iff the modulo-2 sum of any two codewords is also a c.w.

*A code is said to be linear if any 2 c.w.s in the code can be added in mod-2 arithmetic to produce a third c.w. in the code.

* It is possible to find k linearly independent codewords, $g_0, g_1, \dots, g_{k-1}$ such that every codeword in C is a linear combination of these k codewords, that is :

$$v = u_0 g_0 + u_1 g_1 + \dots + u_{k-1} g_{k-1} \qquad (1)$$

where $u_i = 0$ or 1 for $0 \le i \le k-1$: Let us arrange these k linearly indep codewords as the rows of a $k \times n$ matrix as follows.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & & g_{1,n-1} \\ \vdots & & & \\ g_{k-1,0} & g_{k-1,1} & & g_{k-1,n-1} \end{bmatrix}_{k \times n} \quad (2)$$

If $u = (u_0, u_1, \ldots, u_{k-1})$ is the message to be encoded, the corresponding c.w. is given as follows:

$$v = u \cdot G = (u_0, u_1, \ldots, u_{k-1}) \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} \quad (3)$$

$$= u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1}$$

(Clearly, the rows of G generate (or span) the (n, k) linear code C. For this reason, G is called the generator matrix. Note that any k linearly indep. codewords of an (n, k) linear code can be used to form a gen. matrix.

Example:

The $(7, 4)$ linear code given in table 1, has the following gen matrix G:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

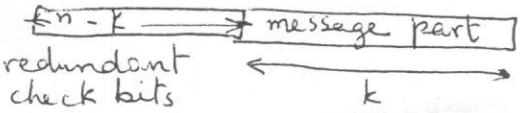
If $u = 1\ 1\ 0\ 1$ is the message to be encoded,

DATE
SUBJ

its corresponding c.w. is

$$v = 1 \cdot g_0 + 1 \cdot g_1 + 0 \cdot g_2 + 1 \cdot g_3$$
$$= (0\ 0\ 0\ 1\ 1\ 0\ 1)$$

* A desirable property for a linear b.c. to possess is the systematic structure of the codewords as shown:



(They are linear sums of the inf digits)

- A linear b.c. with this structure is referred to as a linear systematic b.c. The (7,4) b.c. given in table (1) is a linear systematic b.c.

* A linear systematic b.c. (n,k) is completely specified by a G of the form:

$$G = [\underbrace{P}_{k \times n-k} \quad \underbrace{I_k}_{\text{identity matrix } k \times k}] \quad (4)$$

$$P = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \vdots & & & \\ p_{k-1,0} & & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

$$v = (v_0, v_1, \ldots v_{n-1}) = (u_0, u_1, \ldots u_{k-1}) \cdot G$$

$$v_{n-k+i} = u_i \quad \text{for} \quad 0 \le i \le k-1$$

& $v_j = u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k-1} p_{k-1,j}$ (5)

for $0 \le j \le n-k-1$ [parity check eqs. of the code].

* There is another useful matrix associated with every linear block code : $H_{(n-k)\times n}$ parity check matrix with $n-k$ linearly indep rows

An n-tuple v is a codeword in the code generated by G iff $v \cdot H^T = 0$.

H in systematic form:

$$= [I_{n-k} \mid P^T] \quad (6)$$

$$= \left[\begin{array}{ccc|ccc} 1 & 0 & \cdots & 0 & p_{00} & \cdots & p_{k-1,0} \\ 0 & 1 & & 0 & p_{01} & & p_{k-1,1} \\ 0 & 0 & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{0,n-k-1} & & p_{k-1,n-k-1} \end{array}\right] \quad (7)$$

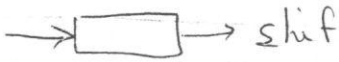
$G \cdot H^T = 0$, (An (n,k) linear b.c. is completely specified by its H matrix]

ex 2: The corresponding H for G in ex (1) is

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

(2) The encoding operation :

Based on eqs (5), the encoding ct of an (n,k) linear syst code can be implemented easily :



* The message $u = (u_0, u_1, \cdots u_{k-1})$ is shifted into the message register & simultaneously into the channel.

As soon as the entire message has been ~~shifted~~ entered the message register, the n-k parity check digits are formed at the outputs of the n-k mod-2 adders. Those parity check digits are then serialized and shifted into the channel.

For example given in table (1) of the (7,4) code, the encoder is given in fig where the connection is based on the parity check eqs given in (5)

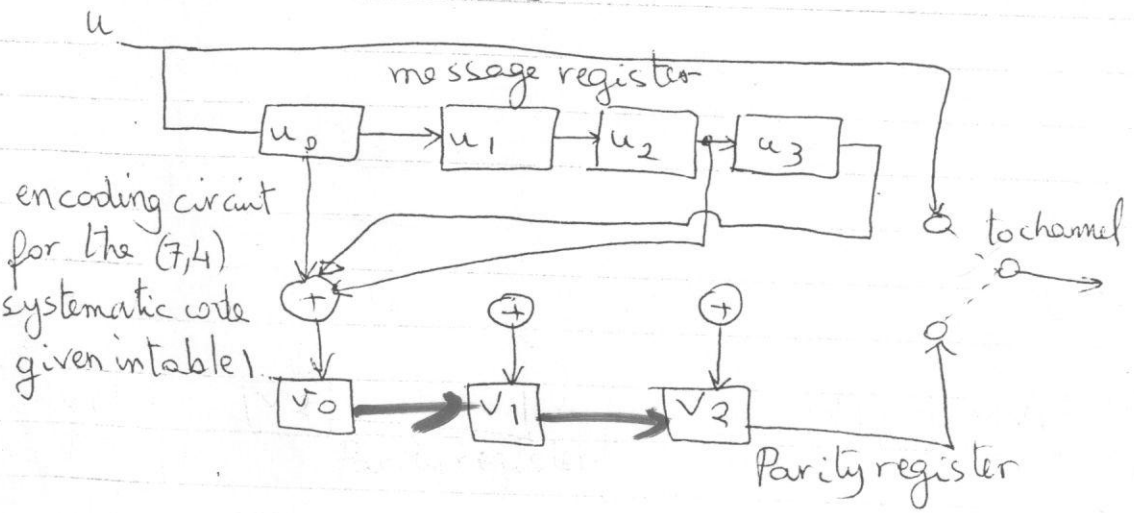
The matrix G for the above ex is given in its systematic form, hence:

$$V = (u_0, u_1, u_2, u_3) \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$(v_0, v_1, v_2, v_3, v_4, v_5, v_6)$

By matrix multiplication we obtain the following digits of the codeword V:

$$v_6 = u_3, \quad v_5 = u_2, \quad v_4 = u_1, \quad v_3 = u_0,$$
$$v_2 = u_1 + u_2 + u_3, \quad v_1 = u_0 + u_1 + u_2, \quad v_0 = u_0 + u_2 + u_3$$



3) Syndrome and error detection :
Consider an (n,K) linear code with gen matrix G & parity check matrix H : $r = v + e$.
$e = (r+v)$ vector sum (5)
$v = (v_0, v_1, \dots, v_{n-1})$ $r = (r_0, r_1, \dots, r_{n-1})$
$e_i = 1$ for $r_i \neq v_i$ & $e_i = 0$ for $r_i = v_i$
↳ error in transmission. e error vector or pattern.

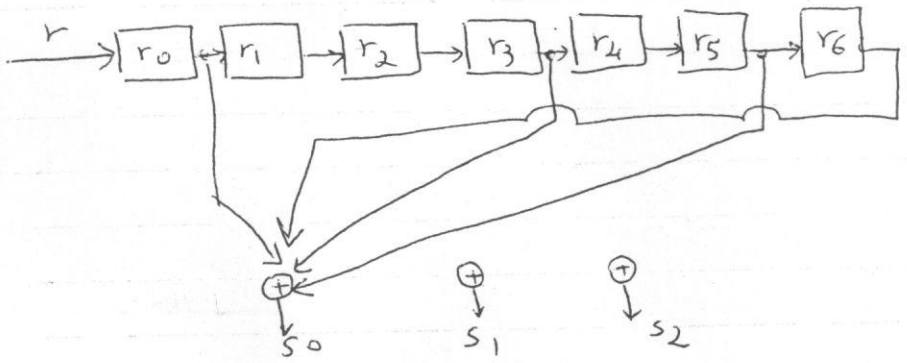
* The decoder doesn't know either e or v, upon receiving r, the decoder must determine whether r contains transmission errors. If the presence of errors is detected, the decoder will either take actions to locate the errors & correct them (FEC) or request for retransmission (ARQ)

Step 1. Compute $s = r \cdot H^T$ $(n-K)$ tuple : (9)
$(s_0, s_1, \dots, s_{n-K-1})$
$s = 0$ iff r is a codeword.

* Therefore, when $s \neq 0$, we detect the presence of errors. (sometimes, $r \cdot H^T = 0$ but r contains errors → undetectable error pattern, the dec makes a dec error). this can be made very small).
* The syndrome ct is constructed based on H matrix (7) & (9)
example: for the (7,4) linear code : $r = (r_0, r_1, r_2, r_3, r_4, r_5, r_6)$

$$s = (s_0, s_1, s_2) = (r_0, r_1, r_2, r_3, r_4, r_5, r_6) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$s_0 = r_0 + r_3 + r_5 + r_6$$
$$s_1 = r_1 + r_3 + r_4 + r_5$$
$$s_2 = r_2 + r_4 + r_5 + r_6$$



* The syndrome s computed from the vector $\underline{r}$ actually depends on the error pattern $\underline{e}$ and not $\underline{v}$

$$s = r \cdot H^T = (v + e) H^T = v_0 H^T + e \cdot H^T = e \cdot H^T \quad (10)$$

based on (7) { H in its systematic form } we have:

$$\left.\begin{aligned} s_0 &= e_0 + e_{n-k} p_{00} + e_{n-k+1} p_{10} + \cdots + e_{n-1} p_{k-1,0} \\ s_1 &= e_1 + e_{n-k} p_{01} + \cdots + e_{n-1} p_{k-1,1} \\ &\vdots \\ s_{n-k-1} &= e_{n-k-1} + e_{n-k} p_{0,n-k-1} + \cdots + e_{n-1} p_{k-1,n-k-1} \end{aligned}\right\} \text{ } n-k \text{ eqs} \quad (11)$$

The syndrome digits are simply linear combinations of the error digits.

The $(n-k)$ linear eqs do not have a unique solution but have 2^k solutions (theorem), i.e there are 2^k error patterns that result in the same syndrome & the true error pattern is just one of them. Various methods to determine the true error pattern from the $(n-k)$ linear eqs of (11) exist.

4) The minimum distance of a block code:
It is a parameter which determines the random error correcting and random error detecting capabilities of a code.

* Let $v = (v_0, v_1, \ldots, v_{n-1})$ be a binary n-tuple. The Hamming weight (or simply weight) of v, denoted by $w(v)$, is defined as the number of non-zero components of v. e.g. w of $v = (1001011) = 4$.

* Let v & w be 2 n-tuples. The Hamming distance bet v & w, $d(v, w)$. {number of places in which they differ}:

$$d(v,w) = w(v+w) \quad (12)$$ [from H.d definition & def. of mod. 2 addition).

for e.g. H.d. bet $v = (1001011)$ & $w = (1110010)$ is 4 & $w(v+w) = w(0111001)$ is also 4.

* The min dist of C denoted as d_{min} is defined as:

$$d_{min} = \min \{ d(v, w); v, w \in C, v \neq w) \quad (13)$$

If C is a linear block code, the sum of 2 codevectors is also a codevector. It follows from (12) that the H.d. bet 2 codevectors in C is equal to the Hamming weight of a third codevector in C. Then it follows from (13) that:

$$d_{min} = \min \{ w(v+w) : v, w \in C, v \neq w \}$$
$$= \min \{ w(x) : x \in C, x \neq 0 \} \quad (14)$$
$$\triangleq w_{min}.$$

called the min weight of the linear code C.
Summarizing the result above we have the following theorem:

Theorem 1: The min. distance of a linear block code is equal to the min weight of its non zero codewords.

* The (7,4) code given in Table (1) has a min weight of 3, thus the min distance is 3.

5) Error Correction & error detection capabilities of linear block codes:

Theorem 2:
A linear b.c. with a min distance d_{min} can correct up to $\lfloor (d_{min}-1)/2 \rfloor$ errors & detect up to $d_{min}-1$ errors in each codeword, where $\lfloor (d_{min}-1)/2 \rfloor$ denotes the largest integer no greater than $(d_{min}-1)/2$.

6) Single error-correcting Hamming codes:

Hamming codes are the first class of linear codes devised for error correction. These codes and their variations have been widely used for error control in digital com. & data storage systems.

* For any positive integer $m \geq 3$, there exists a H.C. with the following parameters:

codelength $n = 2^m - 1$
Number of inf. symbols $k = 2^m - m - 1$
" " parity check symbols $n - k = m$.

Error correcting capability: $t = 1$ ($d_{min} = 3$)

* When a single error occurs, say in the ith bit of the codeword, the syndrome of the received vector is equal to the ith row of H^T. Hence, if we choose the n rows of H^T to be distinct, then the syndrome of all single errors will be distinct & we can correct single errors. (don't use a 0's row).

example: for H matrix given in the last ex:

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

If $v = (1111111)$ was sent,
$r = (0111111)$ was received,
then

$s_0 = 1$
$s_1 = 0$
$s_2 = 0$

→ 1st row → hence error is in first bit.

* if $r = (1110111) \rightarrow s_0 = 1, s_1 = 1, s_2 = 0 \rightarrow 4^{th}$ row
hence error is in the 4^{th} bit →
$e = (0001000) \rightarrow \hat{v} = r + e = (1111111)$

PAGE
DATE

Syndrome Decoding :

We describe a syndrome-based decoding scheme for linear block codes. Let $c_1, c_2, \ldots c_{2^k}$ denote the 2^k code vectors of the (n,k) linear block code. Let r denote the received vector, which may have one of 2^n possible values. The receiver has the task of partitioning the 2^n possible received vectors into 2^k disjoint subsets $D_1, D_2, \ldots D_{2^k}$ in such a way that the i^{th} subset D_i corresponds to the code vector c_i for $1 \leq i \leq 2^k$. The received vector r is decoded into c_i if it is in the i^{th} subset. For the decoding to be correct, r must be in the subset that belongs to the code vector c_i that was actually sent.

The 2^k subsets described herein constitute a standard array of the linear block code.

To construct it,

1. The 2^k code vectors are placed in a row with the all-zero code vector c_1 as the left most element.
2. An error pattern e_2 is picked & placed under c_1, and a 2^{nd} row is formed by adding e_2 to each of the remaining code vectors in the first row; it is important that the error pattern chosen as the 1^{st} element in a row

not have previously appeared in the S.A.
3. Step 2 is repeated until all possible error pattern have been accounted for.

$c_1 = 0$	c_2	c_3	c_i	c_{2^k}
e_2	$e_2 + c_2$	$e_2 + c_3$	$c_i + e_2$	$c_{2^k} + e_2$
e_3	$e_3 + c_2$			
⋮				
$e_{2^{n-k}}$	$e_{2^{n-k}} + c_2$			$c_{2^k} + e_{2^{n-k}}$

S.A. for an (n, k) block code

The 2^k columns of this array represent the disjoint subsets $D_1, D_2, \ldots D_{2^k}$.
The 2^{n-k} rows of the array represent the cosets of the code, and their first elements are called coset leaders.

For a given channel, the prob. of decoding error is minimized when the most likely error patterns (i.e., those with the largest prob. of occurrence) are chosen as the coset leaders. In the case of a BSC, the smaller the H. weight of an error pattern the more likely it is to occur. Accordingly, the S.A. should be constructed with each coset leader having the min. H. weight in its coset.

Cyclic codes

* Binary cyclic codes form a subclass of linear block codes described above. Cyclic codes are attractive for two reasons: First, encoding & synodrome calculations can be implemented easily using simple shift registers with F.B. connections. Secondly, because they have considerable inherent algebraic structure, it is possible to find various simple and efficient decoding methods.

1) Description of Cyclic Codes:

Def: An (n,K) linear code C is called a cyclic code if shifting v to the right cyclically i places, the vector resulting is also a codeword (every cyclic shift of a code vector is also a codevector).

- A polynomial representation is developed for cyclic codes:

$$v = (v_0, v_1, - - - - v_{n-1})$$

$$v(x) = v_0 + v_1 x + v_2 x^2 + - - - - + v_{n-1} x^{n-1} \quad (1)$$

$$\left[v^{(1)} = v_{n-1}, v_0, v_1, - - - - v_{n-2} \right] \quad (2)$$

Thus, each codevector corresponds one-to-one ~~correspon~~ to a polynomial of degree n-1 or less.

2) Generator polynomial :

In an (n, K) cyclic code, there exists one and only one code polynomial $g(x)$ of degree $n-K$:

$$g(x) = 1 + g_1 x + g_2 x^2 + \cdots + g_{n-K-1} x^{n-K-1} + x^{n-K} \quad (3)$$

Each code pol $v(x)$ is a multiple of $g(x)$ & every pol of degree $n-1$ or less which is a multiple of $g(x)$ must be a codepolynomial.

$$v(x) = m(x)\, g(x) = (m_0 + m_1 x + \cdots + m_{K-1} x^{K-1})\, g(x) \quad (4)$$

Thus, the encoding of a message $m(x)$ is equivalent to multiplying the message $m(x)$ by $g(x)$ in eq. (3). Hence, an (n, K) cyclic code is completely specified by $g(x)$ (generator polynomial)

* To obtain $v(x)$ in the systematic form :

- we first multiply $m(x)$ by x^{n-K}

$$m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{K-1} x^{K-1}$$

$$x^{n-K} m(x) = m_0 x^{n-K} + m_1 x^{n-K+1} + \cdots + m_{K-1} x^{n-1} \quad (5)$$

- Dividing $x^{n-K} m(x)$ by $g(x)$, we have

$$x^{n-K} m(x) = q(x)\, g(x) + r(x) \quad (6)$$

$q(x)$: quotient ; $r(x)$: remainder

($r(x)$ must have a degree of $n-k-1$ or less since the degree of $g(x)$ is $n-k$)

— rearranging (6)

$$\underbrace{r(x) + x^{n-k} m(x)}_{\text{multiple of } g(x) \text{ \& has a degree } n-1 \text{ or les}} = q(x) \cdot g(x) \qquad (7)$$

Therefore, $r(x) + x^{n-k} m(x)$ is a code pol of the cyclic code generated by $g(x)$:

$$r(x) + x^{n-k} m(x) = r_0 + r_1 x + \cdots + r_{n-k-1} x^{n-k-1} + m_0 x^{n-k} + m_1 x^{n-k+1} + \cdots + m_{k-1} x^{n-1} \qquad (8)$$

which corresponds to the codevector:

$$\underbrace{(r_0, r_1, \ldots, r_{n-k-1}}_{\text{parity check digits}}, \underbrace{m_0, m_1, \ldots, m_{k-1}}_{k \text{ inf digits}})$$

$n-k$

*Example: consider the $(7,4)$ cyclic code generated by $g(x) = 1 + x + x^3$. Let u be $(1\ 0\ 1\ 1)$ the message to be encoded.

To obtain $v(x)$, we first multiply $m(x)$ by x^{n-k} (x^3) $\rightarrow x^3 m(x) = x^3 + x^5 + x^6$

$(m(x) = 1 + x^2 + x^3)$

second: divide $x^3 m(x) = x^3 + x^5 + x^6$ by $g(x)$.

$$
\begin{array}{r|l}
 & x^3+x^2+x+1 \\
\hline
x^3+x+1 & x^6+x^5+\quad +x^3 \\
 & x^6+\quad +x^4+x^3 \\
\hline
 & x^5+x^4 \\
 & x^5+\quad +x^3+x^2 \\
\hline
 & x^4+x^3+x^2 \\
 & x^4+\quad +x^2+x \\
\hline
 & x^3+x \\
 & x^3+x+1 \\
\hline
 & (1)\ \text{remainder}
\end{array}
$$

$\therefore V(x) = r(x) + x^3 m(x) = 1 + x^3 + x^5 + x^6$

$= (1\ 0\ 0\ 1\ 0\ 1\ 1)$

(parity check digits) (inf digits)

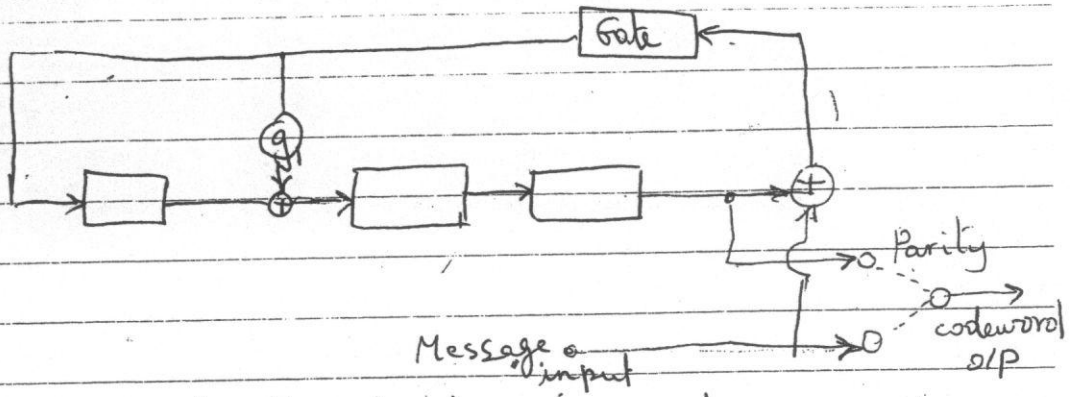
3) Encoding procedure:

The encoding of a message block $m(x)$ of k digits is equivalent to calculating the parity check section $r(x)$ which is the remainder of dividing $x^{n-k} m(x)$ by $g(x)$. All stage above can be accomplished with a division ct. which is a linear $(n-k)$ stage S.R. with F.B. connections based on the gen. pol $g(x) = 1 + g_1 x + \cdots + g_{n-k-1} x^{n-k-1} + x^{n-k}$.

Ex: Consider the $(7,4)$ binary cyclic code generated by $g(x) = 1 + x + x^3$.

DATE
SUBJ.

The encoding ckt is:



An encoder for the (7,4) cyclic code

Step 1: With the gate turned on, the k inf digits are shifted into the reg. S simultaneously into the channel. As soon as the k inf digits have entered the S.R., the n-k digits in the register are the parity check digits.

step 2: Break the F.B. connection by turning off the gate.

step 3: Shift the contents of the S.R. out & send them into the channel.

Suppose $m = (1\,0\,1\,1)$ $\quad m(x) = 1 + X^2 + X^3$

Input	Register contents.	
	0 0 0	initial state
1	1 1 0	1st shift
1	1 0 1	
0	1 0 0	
1	1 0 0	4th shift.

} k shifts

After 4 shifts (k = 4), the contents are (1 0 0), thus the complete codeword is

(1 0 0 1 0 1 1)