The Channel Coding Theorem

It maybe stated in a different form as below:

$$R \le C \text{ or } r_s H(S) \le r_c I(X,Y)_{Max} \text{ or} \{ H(S)/T_s \} \le \{ I(X,Y)_{Max}/T_c \}$$

If a discrete memoryless source with an alphabet ‘’S‟ has an entropy H(S) and producessymbols everyT seconds and a discrete memoryless channel has a capacity I(X,Y)Max and isused once every Tc seconds then ifthere exists a coding scheme for which the source output can be transmitted over the channel andbe reconstructed with an arbitrarily small probability of error. The parameter C/Tc is called thecritical rate. When this condition is satisfied with the equality sign, the system is said to besignaling at the critical rate.

SOURCE CODING

Encoding of the Source Output:

Need for encoding

Encoding involves the use of a code to change original data into a form that can be used by an external process.

Encoding is the process of converting data into a format required for a number of information processing needs, including:

- Program compiling and execution
- Data transmission, storage and compression/decompression
- Application data processing, such as file conversion

Encoding can have two meanings:

- In computer technology, encoding is the process of applying a specific code, such as letters, symbols and numbers, to data for conversion into an equivalent cipher.
- In electronics, encoding refers to analog to digital conversion.

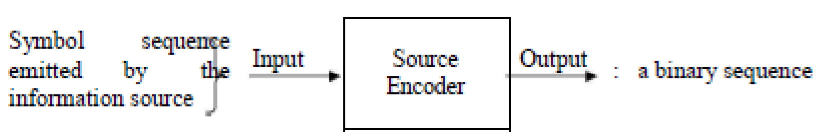
Encoding is also used to reduce the size of audio and video files.

It reduces redundancy thereby power consumption during transmission and space in storage.

It is used to enable error detection and error correction in communication

Let M – messages = 2^N, which are equally likely to occur.

Then recall that averageinformation per messages interval in H = N.Say further that each message is coded into **N** bits,If the messages are not equally likely, then „H‟ will be les s than „N‟ and each bit will carryless than one bit of information,so encoding is needed to improve the si



If the encoder operates on blocks of „N‟ symbols, Produces an average bit rate of GN bits / symbol

$$\text{Where, } G_N = -\frac{1}{N}\sum_i p(m_i)\log p(m_i)$$

$p(m_i)$ = Probability of sequence 'm_i' of 'N' symbols from the source, Sum is over all sequences 'm_i' containing 'N' symbols.

G_N in a monotonic decreasing function of N and

$$\lim_{N\to\infty} G_N = H \text{ bits / symbol}$$

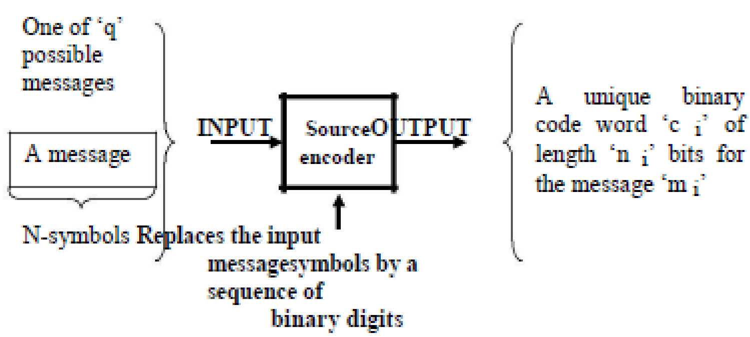
Performance measuring factor for the encoder

Coding efficiency: η_c

$$\text{Definition of } \eta_c = \frac{\text{Source information rate}}{\text{Average output bit rate of the encoder}}$$

$$\boxed{\eta_c = \frac{H(S)}{\hat{H}_N}}$$

Shannon"s Encoding Algorithm:



„q" messages : $m_1, m_2, \ldots m_i, \ldots, m_q$

Probs. of messages : $p_1, p_2, \ldots p_i, \ldots, p_q$

n_i: an integer

- **The objective of the designer**

To find „n_i" and „c_i" for i = 1, 2, ..., q such that the average number of bits per symbol $\hat{H}_N$ used in the coding scheme is as close to G_N as possible.

Where, $\hat{H}_N = \frac{1}{N}\sum_{i=1}^{q} l_i p_i$

and $G_N = \frac{1}{N}\sum_{i=1}^{q} p_i \log \frac{1}{p_i}$

i.e., the objective is to have

$\boxed{\hat{H}_N \quad G_N}$ as closely as possible

Shannon binary encoding procedure:

We present, first, the Shannon"s procedure for generating binary codes mainly because of itshistorical significance.

The procedure is as follows:

1. List the source symbols in the order of decreasing probability of occurrence.

$S = \{s_1, s_2 \ldots s_q\};$ $\quad P = \{p_1, p_2 \ldots p_q\}: p_1 \geq p_2 \geq \ldots\ldots \geq p_q$

2. Compute the sequence:

$\alpha_0 = 0,$

$\alpha_1 = p_1,$

$\alpha_2 = p_2 + p_1 = p_2 + \alpha_1$

$\alpha_3 = p_3 + p_2 + p_1 = p_3 + \alpha_2$

.

$\alpha_{q-1} = p_{q-1} + p_{q-2} + \ldots\ldots + p_1 = p_{q-1} + \alpha_{q-2}.$

$\alpha_q = p_q + p_{q-1} + \ldots\ldots + p_1 = p_q + \alpha_{q-1} = 1$

3. Determine the set of integers, l_k, which are the smallest integer's solution of the inequalities. $2^{l_k} p_k \geq 1$, $k = 1, 2, 3 \ldots q$. Or alternatively, find l_k such that $2^{-l_k} \leq p_k$.

4. Expand the decimal numbers α_k in binary form to l_k places. i.e., neglect expansion beyond l_k digits

5. Removal of the decimal point would result in the desired code.

BLOCK CODES

Linear Block Codes:

A block code is said to be linear (n ,k) code if and only if the 2^kcode words from a k-dimensional sub space over a vector space of all **n**-Tuples over the field **GF(2)**.Fields with $\mathbf{2^m}$symbols are called „Galois Fields" (pronounced as Galva fields), $GF^{(2m)}$.Their arithmetic involves binary additions and subtractions. For two valued variables, (**0, 1**).Themodulo – 2 addition and multiplication is defined in Fig below

		Y	
		0	1
X	0	0	1
	1	1	0

$X \oplus Y$

⊗		Y	
		0	1
X	0	0	0
	0	1	1

$X \otimes Y$

⊕	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

⊗	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

The binary alphabet (**0, 1**) is called a **field** of two elements (a binary field and is denoted by**GF (2)**.

$\mathbf{X^3+X^2+1}$**,** $\mathbf{X^3+X+1}$**,** $\mathbf{X^4+X^3+1}$**,** $\mathbf{X^5+X^2+1}$ etc. are irreducible polynomials, whereas $\mathbf{f(X)=X^4+X^3+X^2+1}$ is not as **f(1)** = **0** and hence has afactor **X+1**) then **p(X)** is said to be a „ **primitive polynomial**".

If **vn**represents a vector space of all **n**-tuples, then a subset **S** of **vn**is called a subspace if (i)the all Zero vector is in **S** (ii) the sum of any two vectors in **S** is also a vector in **S**. To be more

specific, a block code is said to be linear if the following is satisfied. "If **v1** and **v2** are any two codewords of length **n** of the block code then **v1□v2** is also a code word length **n** of the block code".

Example: Linear Block code with**k= 3**, and**n = 6**

Messages		*Code words*		*Weight (No. of 1`s in the code word)*
m_1	000	v_1	0 0 0 0 0 0	0
m_2	001	v_2	0 0 1 1 1 0	3
m_3	010	v_3	0 1 0 1 0 1	3
m_4	100	v_4	1 0 0 0 1 1	3
m_5	011	v_5	0 1 1 0 1 1	4
m_6	101	v_6	1 0 1 1 0 1	4
m_7	110	v_7	1 1 0 1 1 0	4
m_8	111	v_8	1 1 1 0 0 0	3

Observe the linearity property:

$v_3 = (010\ 101)$ and $v_4 = (100\ 011)$, $v_3 \oplus v_4 = (110\ 110) = v_7$.

Remember that **n** represents the word length of the code words and **k** represents the number ofinformation digits and hence the block code is represented as **(n,k)** block code.

Thus by definition of a linear block code it follows that if **g1**, **g2**… **gk**are the **k** linearly independent code words then every code vector, **v**, of our code is a combination of these code words,i.e.

$$v = u_1 g_1 \oplus u_2 g_2 \oplus \ldots \oplus u_k g_k$$

Where $u_j = 0$ or 1, $\forall 1 \le j \le k$

The eqn. can be arranged in matrix form by nothing that each **gj**is an n-tuple, i.e.
gj= (gj1, gj2,…. gjn)
Thus we have **v = u G**Where: **u = (u1, u2… uk)** represents the data vector and

$$G = \begin{matrix} g_1 \\ g_2 \\ g_3 \end{matrix} \begin{matrix} g_{11} & g_{12} & \ldots & g_{1n} \\ g_{21} & g_{22} & \ldots & g_{2n} \\ \vdots & \vdots & & \\ g_{k1} & g_{k2} & \ldots & g_{kn} \end{matrix}$$

Systematic Block Codes (Group Property):

Here a code word isdivided into two parts –Message part and the redund ant part. If either the first **k** digits or the last **k**digits of the code word correspond to the message part then we say that the code is a "SystematicBlock Code".

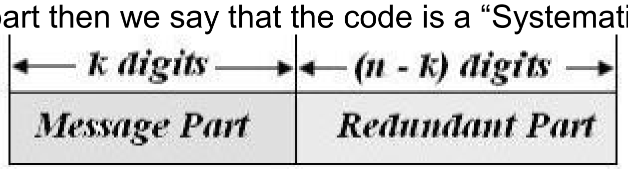


fig format of sys. codes

IFis the ***k*** × ***k***identity matrix (unit matrix), ***P*** is the ***k*** ×(***n*** – ***k***)„ ***parity generator matrix***", in which ***pi,j***are either ***0*** or ***1*** and ***G*** is a ***k***×***n***matrix. The (***n***–***k***) equations given are referred to as ***parity check equations***. Observe that the ***G*** matrix of Example 6.2 is in the systematic format. The ***n***-vectors ***a= (a1,a2…an)*** and ***b= (b1,b2…bn)*** are said to be orthogonal if their inner product defined by: ***a.b = (a1, a2…a n) (b1, b2 …b n)^T= 0.***
where, „ ***T***" represents transposition. Accordingly for any ***k***×***n***matrix, ***G***, with ***k*** linearly independent rows there exists a (***n-k***)×***n*** matrix ***H*** with (***n-k***) linearly independent rows such that any vector in the row space of ***G*** is orthogonal to the rows of ***H*** and that any vector that is orthogonal to the rows of ***H*** is in the row space of ***G***. Therefore, we can describe an(***n***, ***k***)linear code generated by ***G*** alternatively asfollows:
"An n – tuple, v is a code word generated by G, if and only if v.H^T= O".
(**Orepresents an all zero row vector**.)
This matrix ***H*** is called a " ***parity check matrix***" of the code. Its dimension is (***n–k***)×***n.***

If the generator matrix has a systematic format, the parity check matrix takes the following form.
This matrix **H** is called a " **parity check matrix**" of the code.Its dimension is (***n–k***)×***n.***
If the generator matrix has a systematic format, the parity check matrix takes the following form.

$$H = [P^T \; I_{n-k}] = \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{k1} & 1 & 0 & 0 & \cdots & 0 \\ p_{12} & p_{22} & \cdots & p_{k2} & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ p_{1,n-k} & p_{2,n-k} & \cdots & p_{k,n-k} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

The i^{th} row of G is:

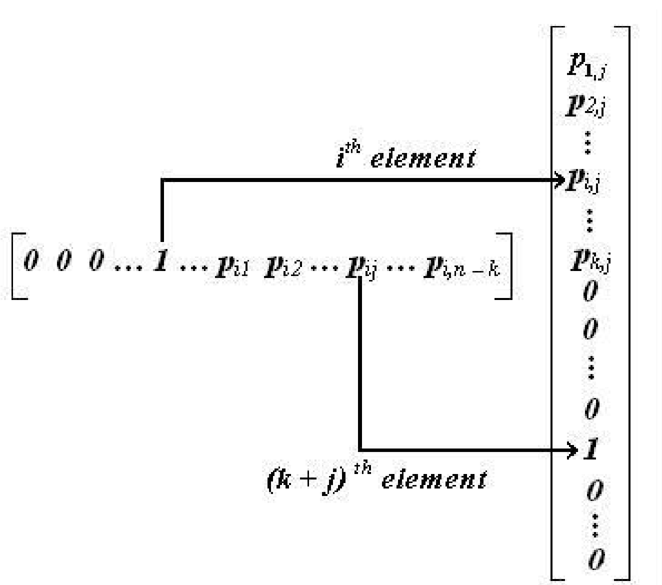
$$g_i = (\underset{i^{th}\text{ element}}{0\,0 \ldots \underline{1} \ldots 0 \ldots 0} \quad \underset{(k+j)^{th}\text{ element}}{p_{i,1} \; p_{i,2} \ldots \underline{p_{i,j}} \ldots p_{i,n-k}})$$

The j^{th} row of H is:

$$h_j = (\underset{\substack{\downarrow \\ i^{th}\text{ element}}}{p_{1,j}\; p_{2,j} \ldots p_{i,j}} \ldots p_{k,j}\; \underset{\substack{\downarrow \\ (k+j)^{th}\text{ element}}}{0\,0 \ldots 0\,1\,0 \ldots 0})$$

Accordingly the inner product of the above n – vectors is:

$$g_i \times h_j^T = (\underset{i^{th}\text{ element}}{0\,0 \ldots \underline{1} \ldots 0 \ldots 0} \quad \underset{(k+j)^{th}\text{ element}}{p_{i,1} \; p_{i,2} \ldots \underline{p_{i,j}} \ldots p_{i,n-k}})\,(\underset{i^{th}\text{ element}}{p_{1,j} \; p_{2,j} \ldots \underline{p_{i,j}}} \ldots p_{k,j}\; \underset{(k+j)^{th}\text{ element}}{0\,0 \ldots 0\,\underline{1}\,0 \ldots 0})^T$$



Where **O**$_{k\times(n-k)}$ is an all zero matrix of dimension $k\times(n-k)$.
Further, since the **$(n-k)$** rows of the matrix ***H*** are linearly independent, the ***H*** matrix is a parity check matrix of the **(n, k)** linear systematic code generated by ***G***.

Syndrome and Error Detection:
Suppose **v= (v1,v2…vn)** be a code word transmitted over a noisy channel and let:
r = (r1, r2…. rn)be the received vector. Clearly, **r** may be different from **v** owing to the channelnoise.The vector sum
e = r – v = (e1, e2… en) is an **n**-tuple, where ***ej***= ***1*** if ***rj≠vj***and ***ej***= ***0*** if ***rj=vj***. This ***n*** – tuple is called the “ ***error vecto*r**”or “ ***error pattern***”. The ***1***‟s in***e*** are the transmission errors caused by the channel noise.

r = v ⊕ e

When ***r*** is received, the decoder computes the following **(*n-k*)** tuple:

s = r. H^T= (s1, s2… sn-k)

s = 0 if and only if ***r*** is a code word and ***s≠0*** iff***r*** isnot a code word. This vector s is called . Thus if ***s = 0***, the receiveraccepts ***r*** as a valid code word. Notice that there are possibilities of errors undetected, whichhappens when ***e*** is identical to a nonzero code word. In this case ***r*** is the sum of two code wordswhich according to our linearity property is again a code word. This type of error pattern isreferred to an " ***undetectable errorpattern***". Since there are ***2^{k-1}***nonzero code words, it followsthat there are ***2^{k-1}***error patterns aswell. Hence when an undetectable error pattern occurs thedecoder makes a " ***decoding error***".as below:

$$s = r.H^T = (s_1, s_2 \ldots s_{n-k}) = (r_1, r_2, \ldots, r_n)\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n-k} \\ p_{21} & p_{22} & \cdots & p_{2,n-k} \\ \vdots & & & \\ p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

From which we have

$$s_1 = r_1 p_{11} + r_2 p_{21} + \ldots + r_k p_{k1} + r_{k+1}$$

$$s_2 = r_1 p_{12} + r_2 p_{22} + \ldots + r_k p_{k2} + r_{k+2}$$

$$\mathrm{M} \quad \mathrm{M} \quad \mathrm{M} \quad \mathrm{M} \quad \mathrm{M}$$

$$s_{n-k} = r_1 p_{1,n-k} + r_2 p_{2,n-k} + \ldots + r_k p_{k,n-k} + r_n$$

The syndrome is simply thevector sum of the received parity digits **(rk+1, rk+2...rn)** and the parity check digits recomputedfrom the received information digits **(r1, r2… rn)**.

Example

We shall compute the syndrome for the (**6, 3**) systematic code of Example 5.2. We have

$$\text{or } s_1 = r_2 + r_3 s_2 + r_4$$

$$= r_1 + r_3\ s_3 + r_5$$

$$= r_1 + r_2 \quad + r_6$$

$$s = r.H^T = (v \oplus e)\, H^T$$

$$= v.H^T \oplus e.H^T$$

$$\text{or } s = e.H^T$$

as $v.H^T = O$. we have the following relationshipbetween the syndrome digits and the error digits.

$$
\begin{aligned}
s_1 &= e_1 p_{11} + e_2 p_{21} + \dots + e_k p_{k,1} + e_{k+1} \\
s_2 &= e_1 p_{12} + e_2 p_{22} + \dots + e_k p_{k,2} + e_{k+2} \\
&\vdots \\
s_{n-k} &= e_1 p_{1,n-k} + e_2 p_{2,n-k} + \dots + e_k p_{k,n-k} + e_n
\end{aligned}
$$

Thus, the syndrome digits are linear combinations of error digits.

3. Uniquely decodable codes:

A non-singular code is uniquely decipherable, if every word immersed in a sequence of words can be uniquely identified. The **n**thextension of a code, that maps each message into the codewords **C**, is defined as a code which maps the sequence of messages into a sequence of code words.

This is also a block code, as illustrated in the following example.

Example: Second extension of the code set given in **C = {0, 11, 10, 01}**

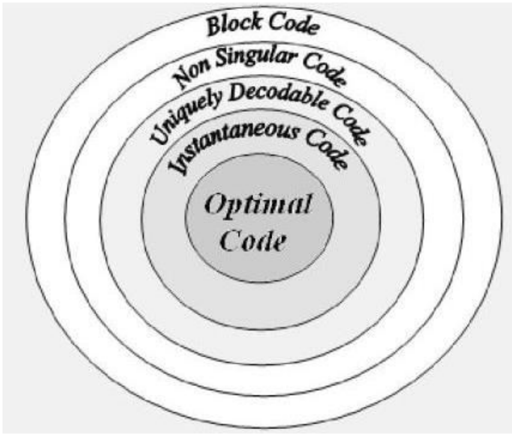
S^2={s1s1,s1s2,s1s3,s1s4;s2s1,s2s2,s2s3,s2s4,s3s1,s3s2,s3s3,s3s4,s4s1,s4s2,s4s3,s4s4}

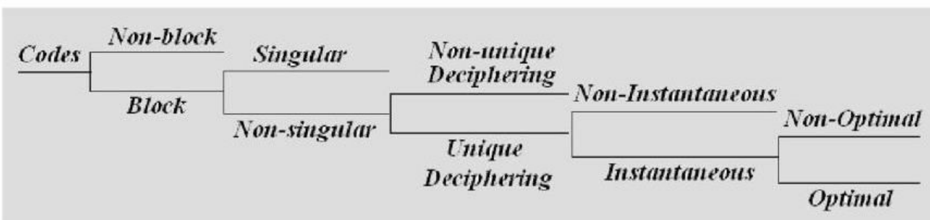
Source Symbols	*Codes*	*Source Symbols*	*Codes*	*Source Symbols*	*Codes*	*Source Symbols*	*Codes*
s_1s_1	*0 0*	s_2s_1	*1 1 0*	s_3s_1	*1 0 0*	s_4s_1	*0 1 0*
s_1s_2	*0 1 1*	s_2s_2	*1 1 1 1*	s_3s_2	*1 0 1 1*	s_4s_2	*0 1 1 1*
s_1s_3	*0 1 0*	s_2s_3	*1 1 1 0*	s_3s_3	*1 0 1 0*	s_4s_3	*0 1 1 0*
s_1s_4	*0 0 1*	s_2s_4	*1 1 0 1*	s_3s_4	*1 0 0 1*	s_4s_4	*0 1 0 1*

Notice that, in the above example, the codes for the source sequences, **s1s3** and **s4s**1 are notdistinct and hence the code is "**Singular in the Large**". Since such singularity properties introduceambiguity in the decoding stage, we therefore require, in general, for unique decidability of our codesthat **"The nthextension of the code be non-singular for every finite n."**

4. Instantaneous Codes:

A uniquely decodable code is said to be " **instantaneous**" if the end of any code word is recognizablewith out the need of inspection of succeeding code symbols. That is **there is no timelagin the process of decoding**. To understand the concept, consider the following codes:





Kraft Inequality:
Given a source S = {**s1, s2…s q**}.Let the word lengths of the codes corresponding to these symbols be**l1, l2 …….l q** and let the code alphabet be X = {**x1, x2…x r**}. Then, an instantaneous code for thesource existsiff

$$\sum_{k=1}^{q} r^{-lK} \leq 1$$

The above Eq is called **Kraft Inequality** (Kraft – 1949).
Example:
A six symbol source is encoded into Binary codes shown below. Which of these codes areinstantaneous?

Source symbol	Code A	Code B	Code C	Code D	Code E
s_1	0 0	0	0	0	0
s_2	0 1	1 0 0 0	1 0	1 0 0 0	1 0
s_3	1 0	1 1 0 0	1 1 0	1 1 1 0	1 1 0
s_4	1 1 0	1 1 1 0	1 1 1 0	1 1 1	1 1 1 0
s_5	1 1 1 0	1 1 0 1	1 1 1 1 0	1 0 1 1	1 1 1 1 0
s_6	1 1 1 1	1 1 1 1	1 1 1 1 1	1 1 0 0	1 1 1 1
$\sum_{k=1}^{6} 2^{-l_k}$	1	$\frac{13}{16} < 1$	1	$\frac{7}{8} < 1$	$1\frac{1}{32} > 1$

As a first test we apply the Kraft Inequality and the result is accordingly tabulated. **Code E doesNotsatisfy Kraft Inequality and it is not an instantaneous code**.

Next we test the prefix property. For **Code D**, notice that the complete code word for the symbol **s4** isa prefix of the code word for the symbol **s3**. Hence it is not an instantaneous code. However, **Code A**,**Code B** and **Code C** satisfy the prefix property and are therefore they are instantaneous codes.

Code Efficiency and Redundancy:

Consider a zero memory source, **S** with **q**-symbols {**s1**, **s2**… **sq**} and symbol probabilities {**p1**, **p2**… **pq**} respectively. Let us encode these symbols into **r**- ary codes (Using a code alphabet ofsymbols) with word lengths **l1, l2…l q**.We shall find a lower bound for the average length of thecodewords and hence define efficiency and redundancy of the code.**Q**Let **Q1, Q2** … **Qq**be any set of numbers such that **Qk**≥ 0 and

$$\sum_{k=1}^{q} Q_k = 1.$$

Consider the quantity

Equality holds iff**Qk**= **pk**. Eq. (5.21) is valid for any set of numbers **Qk**that are non negative andsum to unity. We may then choose:

$$Q_k = \frac{r^{-l_k}}{\sum_{k=1}^{q} r^{-l_k}}$$

and obtain

$$H(s) \leq \sum_{k=1}^{q} p_k \log r^{l_k} \sum_{k=1}^{q} r^{-l_k}$$

$$\leq \sum_{k=1}^{q} p_k\, l_k \log r + \log \sum_{k=1}^{q} r^{-l_k}$$

$$i.e \quad H(S) \leq \log r \sum_{k=1}^{q} p_k\, l_k + \log \sum_{k=1}^{q} r^{-l_k}$$

Defining

$$L = \sum_{k=1}^{q} p_k\, l_k$$

Which gives the average length of the code words, and

$$\frac{H(S)}{L} \leq \log r$$

LHS of the Eq. is simply (The entropy of the source in bits per source symbol) **÷** (no. of codesymbols per source symbol) or bits per code symbol; which is nothing but the actual entropy of thecode symbols. RHS is the maximum value of this entropy when the code symbols are all equiprobable.Thus we can define the code efficiency as

"**Code efficiency is the ratio of the average information per symbol of the encoded language tothemaximum possible information per code symbol**". Mathematically, w ewrite

Code efficiency

$$\eta_c \triangleq \frac{H(S)}{L} : \log r$$

or

$$\eta_c \triangleq \frac{H(S)}{L \log r}$$

Accordingly, Redundancy of the code,

$E_c = 1 - \eta_c$

Example:

Let the source have four messages **S=** {**s1, s2, s3, s4**} with **P=1/2,1/4,1/8,1/8**

$\therefore H(S) = \frac{1}{2}\log 2 + \log 4 + 2\,x\,\frac{1}{8}\log 8 = 1.75$ *bits/sym.*

If S itself is assumed to be the alphabet then we have

$L=1$, $r=4$ and $\eta_c = \frac{1.75}{\log 4} = 0.875$, i.e. $\eta_c = 87.5\%$, and $E_c=12.5\%$

Suppose the messages are encoded into a binary alphabet, $X=\{0, 1\}$ as

	p_k	Code	l_k
s_1	1/2	0	$l_1=1$
s_2	1/4	1 0	$l_2=2$
s_3	1/8	1 1 0	$l_3=3$
s_4	1/8	1 1 1	$l_4=3$

We have $L=\sum_{k=1}^{4} l_k p_k = 1.\frac{1}{2} + 2.\frac{1}{4} + 3.\frac{1}{8} + 3.\frac{1}{8} = 1.75$ *binits/symbol*

Since $r=2$, $\eta_c = \frac{H(S)}{L \log r} = \frac{1.75}{1.75 \log_2 2} = 1$

i.e. $\eta_c=100\%$, and hence $Ec=1-\eta_c=0\%$

Thus by proper encoding, the efficiency can be increased.

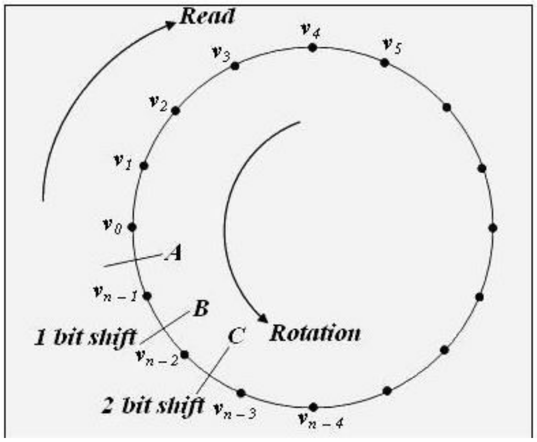
So,the equality **L= H(S) /log r is strict since lk**= **log1/pk**

CYCLIC CODES

A binary code is said to be "**cyclic**" if it satisfies:

1. Linearity property – sum of two code words is also a code word.
2. Cyclic property – Any lateral shift of a code word is also a code word.

For example, if we move in a counter clockwise direction then starting at „ **A**" the code wordis **110001100** while if we start at **B** it would be **011001100**. Clearly, the two code words are related inthatone is obtained from the other by a cyclic shift.



Illustrating the Cyclic Property.

If the **n** - tuple, read from „ **A**" in the **CW** direction in Fig illustrating cyclic property,
v = (vo, v1, v2, v3, vn-2, vn-1)
is a code vector, then the code vector, read from **B**, in the **CW** direction, obtained by a one bit cyclicright shift:
V^1 = (vo, v1, v2, v3, vn-2, vn-1)
is also a code vector. In this way, the **n** - tuples obtained by successive cyclic right shifts:
$v^{(2)}$ = (vn-2, vn-1, vn, v0, v1… vn-3)
$v^{(3)}$ = (vn-3 ,vn-2, vn-1, vn, ... vo, v1, vn-4)

$v^{(i)}$= (vn-i, vn-i+1,…v n-1, vo, v1,…. v n-i-1)

$$V(X) = v_0 + v_1 X + v_2 X^2 + v_3 X^3 + \ldots + v_{i-1} X^{i-1} + \ldots + v_{n-3} X^{n-3} + v_{n-2} X^{n-2} + v_{n-1} X^{n-1} \qquad \ldots\ldots$$

are all code vectors. This property of cyclic codes enables us to treat the elements of each code vectoras the co-efficients of a polynomial of degree **(n-1).**
This is the property that is extremely useful in the analysis and implementation of these codes. Thus we write the "code polynomial' **V(X)** for the codeas a vector polynomial as:

Thus it turns out that

$$V^{(1)}(X) = v_{n-1} + v_0 X + v_1 X^2 + \ldots + v_{n-2} X^{n-1} \qquad \ldots\ldots\ldots\ldots$$

is the code polynomial for $v^{(1)}$. We can continue in this way to arrive at a general format:

$$X^i V(X) = V^{(i)}(X) + q(X)(X^n + 1) \qquad \ldots\ldots\ldots\ldots$$

Remainder Quotient

Where

$$V^{(i)}(X) = v_{n-i} + v_{n-i+1}X + v_{n-i+2}X^2 + \dots v_{n-1}X^i + \dots v_0 X^{i-1} + v_1 X^{i+1} + \dots v_{n-i-2}X^{n-2} + v_{n-i-1}X^{n-} \quad \dots\dots$$

Generator Polynomial for Cyclic Codes:

An **(n, k)** cyclic code is specified by the complete set of code polynomials of degree ≤**(n-1)**and contains a polynomial **g(X)**, of degree **(n-k)** as a factor, called the "**generator polynomial**" of thecode. This polynomial is equivalent to the generator matrix **G**, of block codes. Further, it is the onlypolynomial of minimum degree and is unique. Thus we have an important theorem

Theorem 1 "If**g(X)**is a polynomial of degree**(n-k)**and is a factor of**(Xn+1)**then**g(X)**generatesan**(n, k)**cyclic code in which the code polynomial **V(X)** for a data vector **u = (u0, u1… uk -1)** is generated by

V(X) = U(X) ×**g(X)**

Where

$$U(X) = u_0 + u_1 X + u_2 X^2 + \dots + u_{k-1} X^{k-1} \quad \dots\dots\dots\dots\dots\dots\dots$$

is the data polynomial of degree **(k-1)**.The theorem can be justified by Contradiction: - If there is another polynomial of same degree, thenadd the two polynomials to get a polynomial of degree <**(n, k)** (use linearity property and binaryarithmetic). Not possible because minimum degree is (**n-k**). Hence **g(X)** is uniqueClearly, there are $\mathbf{2^k}$code polynomials corresponding to $\mathbf{2^k}$data vectors. The code vectorscorresponding to these code polynomials form a linear (**n, k**) code. We have then, from the theorem

$$g(X) = 1 + \sum_{i=1}^{n-k-1} g_i X^i + X^{n-k}$$

$$\text{As } g(X) = g_0 + g_1 X + g_2 X^2 + \dots\dots \qquad + g_{n-k-1} X^{n-k-1} \quad + g_{n-k} X^{n-k}$$

Suppose **u0=1** and **u1=u2= …=uk-1=0.** Then it follows **g(X)** is a code word polynomialof degree**(n-k)**. This is treated as a „ **basis code polynomial**‟ (All rows of the **G** matrix of a block code,being linearly independent, are also valid code vectors and form „ **Basis vectors**‟ of the code).

is a polynomial of minimum degree, it follows that $g_0 = g_{n-k} = 1$ always and the remaining co-efficients may be either' *0*' of '*1*'. Performing the multiplication we have:

$$U(X)\, g(X) = u_0\, g(X) + u_1 X\, g(X) + \dots + u_{k-1} X^{k-1} g(X)$$

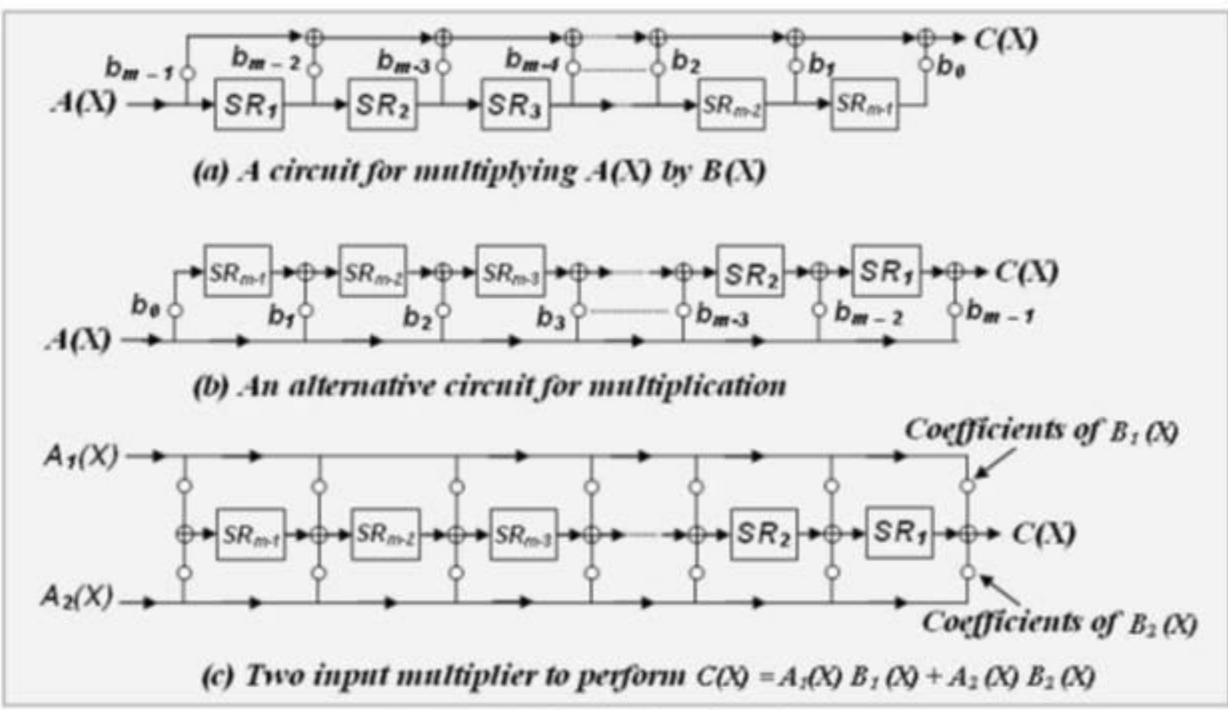
Therefore from cyclic property $\mathbf{X^i g(X)}$ is also a code polynomial. Moreover, from the linearityproperty - a linear combination of code polynomials is also a code polynomial. It follows thereforethat any multiple of **g(X)** as a code polynomial. Conversely, any binarypolynomial of degree **(n-1)** is a code polynomial if and only if it is a multiple of **g(X).** The codewords generated are in non-systematic form. Non systematic cyclic codes can begenerated by simple binary multiplication circuits using shift registers. here we have described cyclic codes with right shift operation. Left shift version canbe obtained by simply re-writing the polynomials. Thus, for left shift operations, the variouspolynomials take the following form

$U(X) = uoX^{k-1} + u1X^{k-2} + \ldots\ldots + u^{k-2}X + u^{k-1}$ …………………. (a)
$V(X) = v0\,X^{n-1} + v1X^{n-2} + \ldots. + v^{n-2}X + v^{n-1}$ ……………… (b)
$g(X) = g0X^{n-k} + g1X^{n-k-1} + \ldots..+ g^{n-k-1}\,X + g^{n-k}$……………… (c)

$$= X_{n-k} + \sum_{i=1}^{n-k} g_i X^{n-k-i} + g_{n-k}$$
…………..(d)

Multiplication Circuits:
Construction of encoders and decoders for linear block codes are usually constructed withcombinational logic circuits with mod-2 adders. Multiplication of two polynomials **A(X)** and **B(X)**and the division of one by the other are realized by using sequential logic circuits, mod-2 adders andshift registers. In this section we shall consider multiplication circuits.
For the polynomial: **A(X) = a0+ a1X + a2X2+...+ an-1Xn-1**where **ai"s**are either a ' **0**' or a '**1**', the right most bit in the sequence **(a0, a1, a2... an-1)** is transmittedfirst in any operation. The product of the two polynomials **A(X)** and **B(X)** yield:
C(X) = A(X) *B(X)
$= (a0 + a1\,X + a2\,X^2 + \ldots \ldots\ldots\ldots + a\,n\text{-}1X^{n-1})\,(b0 + b1\,X + b2X^2 + \ldots + b\,m\text{-}1\,X^{m-1})$
$= a^0b^0 + (a1b0+a0b1)\,X + (a0b2 + b0a2+a1b1)\,X^2 + \ldots. + (a\,n\text{-}2bm\text{-}1 + an\text{-}1bm\text{-}2)\,X^{n+m-3} + an\text{-}1bm\text{-}1X^{n+m-2}$

This product may be realized with the circuits as in fig below illustrate the concepts described so far.



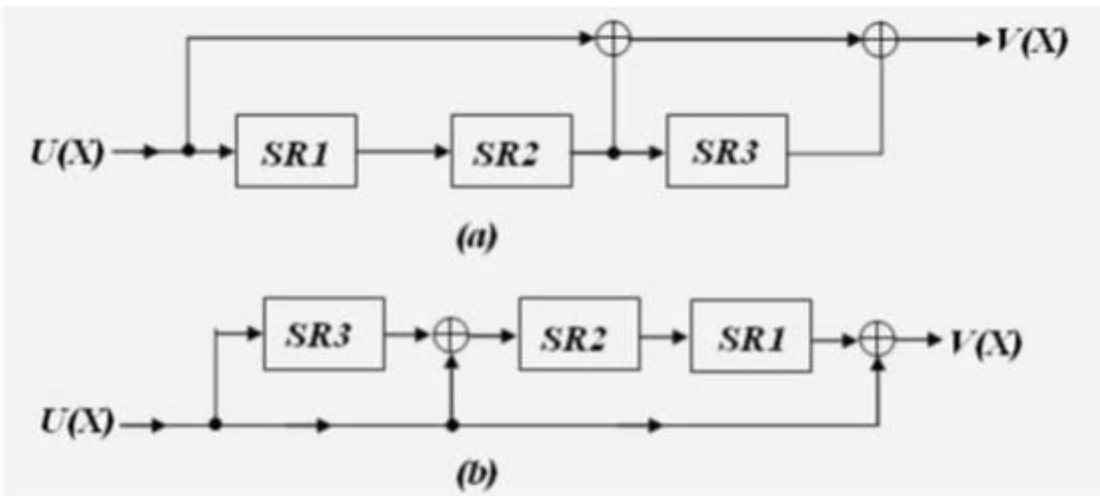
(a) A circuit for multiplying A(X) by B(X)
(b) An alternative circuit for multiplication
(c) Two input multiplier to perform $C(X) = A_1(X)\,B_1(X) + A_2(X)\,B_2(X)$

Multiplication circuits

Example : Consider the generation of a (**7, 4**) cyclic code. Here**(n- k)**= (**7-4**) =**3**and we have tofind agenerator polynomial of degree **3** which is a factor of **Xn+ 1 = X7+ 1**.To find the factors of" degree **3**, divide **X7+1** by **X3+aX2+bX+1**, where '**a**' and '**b**' are binarynumbers, to get the remainder as **abX2+ (1 +a +b) X+ (a+b+ab+1).** Only condition for the remainderto be zero is **a +b=1** which means either **a = 1, b = 0** or **a = 0, b = 1**. Thus we have two possiblepolynomials of degree 3, namely**g1 (X) =** $X^3 + X^2 + 1$ and g^2 **(X) =** X^3+X+1In fact, X^7+1 can be factored as:$(X^7+1) = (X+1)(X^3+X^2+1)(X^3+X+1)$Thus selection of a 'good' generator polynomial seems to be a major problem in the design of cycliccodes. No clear-cut procedures are available. Usually computer search procedures are followed.

Let us choose **g (X) =** $X^3 + X + 1$ as the generator polynomial. The encoding circuits are shown in

Fig below (a) and (b).



Generation of Non-systematic Cyclic codes V(X) = U(X).g(X)

To understand the operation, Let us consider **u = (10 1 1)** i.e.

U (X) = $1 + X^2 + X^3$.

We have **V (X) =** $(1 + X^2 + X^3)(1 + X + X^3)$.

$= 1 + X^2 + X^3 + X + X^3 + X^4 + X^3 + X^5 + X^6$

$= 1 + X + X^2 + X^3 + X^4 + X^5 + X^6$

because $(X^3 + X^3 = 0)$

=> v = (1 1 1 1 1 1 1)

theproduct polynomial is:

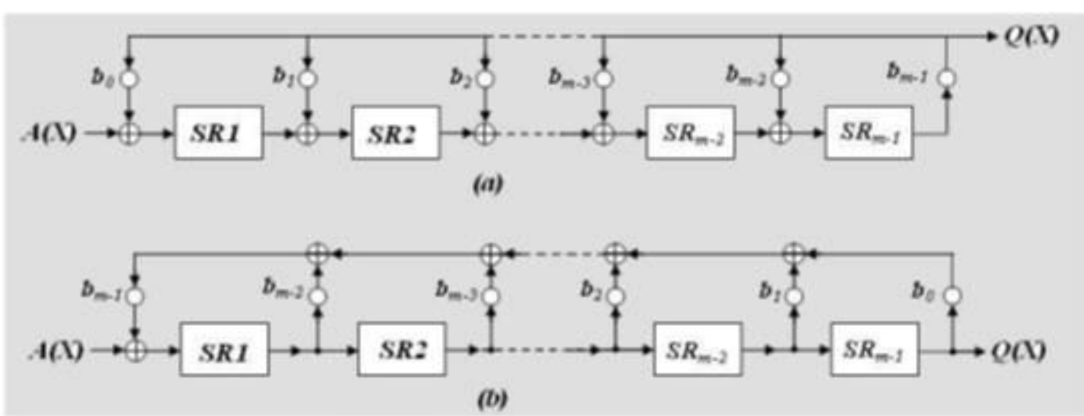
V (X) = $1 + X + X^2 + X^3 + X^4 + X^5 + X^6$

and hence out put code vector is **v = (1 1 1 1 1 1 1),** as obtained by direct multiplication. The readercan verify the operation of the circuit in the same manner. Thus the multiplicationcircuits of can be used for generation of non-systematic cyclic codes.

Shift Number	Input Queue	Bit shifted IN	Contents of shift registers.			Out put	Remarks
			SRI	SR2	SR3		
0	0001011	-	0	0	0	-	Circuit In reset mode
1	000101	1	1	0	0	1	Co-efficient of X^6
2	00010	1	1	1	0	1	Co-efficient of X^5
3	0001	0	0	1	1	1	X^4 co-efficient
*4	000	1	1	0	1	1	X^3 co-efficient
5	00	0	0	1	0	1	X^2 co-efficient
6	0	0	0	0	1	1	X^1 co-efficient
7	-	0	0	0	0	1	X^0 co-efficient

Dividing Circuits:

As in the case of multipliers, the division of **A (X)** by **B (X)** can be accomplished by using shiftregisters and Mod-2 adders, as shown in Fig. below In a division circuit, the first co-efficient of thequotient is **(an-1/(bm -1) = q1**, and **q1.B(X)** is subtracted from **A (X)**. This subtraction is carried out bythe feed back connections shown. This process will continue for the second and subsequent terms.However, remember that these coefficients are binary coefficients. After **(n-1)** shifts, the entirequotient will appear at the output and the remainder is stored in the shift registers.



Dividing Circuits

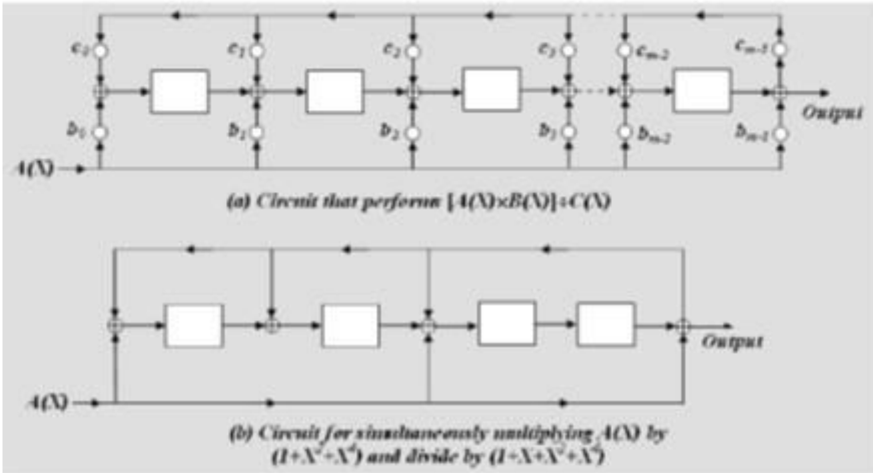
It is possible to combine a divider circuit with a multiplier circuit to build a “composite multiplier-divider circuit” which is useful in various encoding circuits.

Example:

Let $A(X) = X^3+X^5+X^6$, $\rightarrow A = (0001011)$, $B(X) = 1 + X + X^3$.

We want to find the quotient and

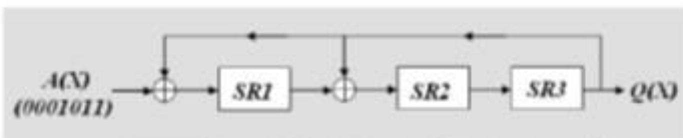
remainder after dividing **A(X)** by **B(X).** The circuit to perform this division is shown in Fig below. The operation of the divider circuit is listed in the table



(a) Circuit that perform $[A(X)\times B(X)]\div C(X)$

(b) Circuit for simultaneously multiplying $A(X)$ by $(1+X^2+X^4)$ and divide by $(1+X+X^2+X^3)$

Circuits for Simultaneous Multiplication and Division



Circuit for dividing $A(X)$ by $(1+X+X^3)$

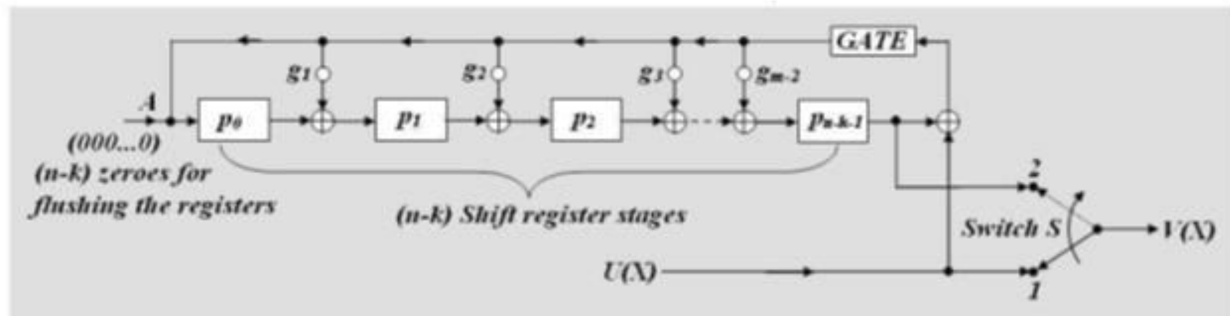
Table Showing the Sequence of Operations of the Dividing circuit

Shift Number	Input Queue	Bit shifted IN	Contents of shift Registers.			Out put	Remarks
			SR1	SR2	SR3		
0	0001011	-	0	0	0	-	Circuit in reset mode
1	000101	1	1	0	0	0	Co-efficient of X^6
2	00010	1	1	1	0	0	Co-efficient of X^5
3	0001	0	0	1	1	0	X^4 co-efficient
4	000	1	0	1	1	1	X^3 co-efficient
5	00	0	1	1	1	1	X^2 co-efficient
6	0	0	1	0	1	1	X^1 co-efficient
7	-	0	1	0	0	1	X^0 co-efficient

The quotient co-efficients will be available only after the fourth shift as the first three shiftsresult in entering the first 3-bits to the shift registers and in each shift out put of the last register, **SR3**,iszero.The quotient co-efficient serially presented at the out put are seen to be **(1111)** and hence thequotient polynomial is**$Q(X) = 1 + X + X^2 + X^3$**
The remainder co-efficients are **(1 0 0)** and theremainder polynomial is **$R(X) = 1$**. after the **$(n\text{-}k)^{th}$**shift register the result is the division of **X^{n-k} A (X)** by **B (X).**Accordingly, we have the following scheme to generate systematic cyclic codes. The generatorpolynomial is written as:

$$g(X) = 1 + g_1X + g_2X^2 + g_3X^3 + \ldots + g_{n-k-1}X^{n-k-1} + X^{n-k} \qquad \ldots\ldots\ldots\ldots\ldots$$

The circuit below does the job of dividing **X^{n-k} U (X)** by **g(X)**. The following steps describe the encoding operation.



Systematic encoding of cyclic codes with (n-k) shift Register stages

1. The switch **S** is in position **1** to allow transmission of the message bits directly to an out put shift register during the first **k**-shifts.
2. At the same time the '**GATE**' is '**ON**' to allow transmission of the message bits into the **(n-k)** stage encoding shift register
3. After transmission of the **k**thmessage bit the **GATE** is turned **OFF** and the switch **S** is moved to position **2**.
4. **(n-k)** zeroes introduced at "**A**" after step **3**, clear the encoding register by moving theparity bits to the output register
5. The total number of shifts is equal to **n** and the contents of the output register is the code word polynomial **V (X) =P (X) +** X^{n-k}**U (X).**
6. After step-**4**, the encoder is ready to take up encoding of the next message input

Clearly, the encoder is very much simpler than the encoder of an **(n, k)** linear block code and the

memory requirements are reduced. The following example illustrates the procedure.

Example:

Let **u = (1 0 1 1)** and we want a **(7, 4)** cyclic code in the systematic form. The generator polynomialchosen is **g (X) = 1 + X + X3**

For the given message, $U(X) = 1 + X^2 + X^3$

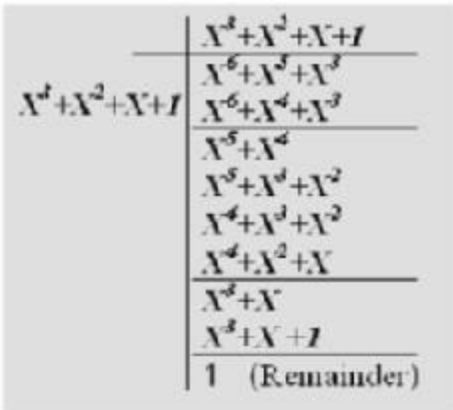
$$X^{n-k}\, U(X) = X^3 U(X) = X^3 + X^5 + X^6$$

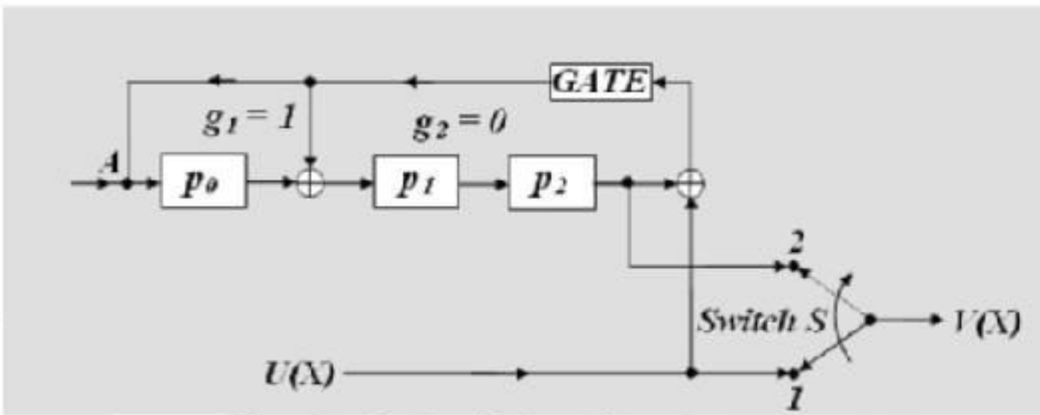
We perform direct division $X^{n-k} U(X)$ *by* $g(X)$ as shown below.

From direct division observe that

p0=1, p1=p2=0. Hence the code word in systematic format is:

v = (p0, p1, p2; u0, u1, u2, u3) = (1, 0, 0, 1, 0, 1, 1)





Encoder for the (7,4) cyclic code

The encoder circuit for the problem on hand is shown operational steps are as follows:

Shift Number	*Input Queue*	*Bit shifted IN*	*Register contents*	*Output*
0	1011	-	000	-
1	101	1	110	1
2	10	1	101	1
3	1	0	100	0
4	-	1	100	1

After the Fourth shift **GATE** Turned **OFF**, switch **S** moved to position **2,** and the parity bits
contained in the register are shifted to the output. The out put code vector is **v = (100 1011)** whichagrees with the direct hand calculation.

Syndrome Calculation - Error Detection and Error Correction:
Suppose the code vector **v= (v0, v1, v2…vn-1)** is transmitted over a noisy channel. Hence the
received vector may be a corrupted version of the transmitted code vector. Let the received code
vector be **r = (r0, r1, r2…rn-1)**. The received vector may not be anyone of the **2**^{k}valid code vectors.The function of the decoder is to determine the transmitted code vector

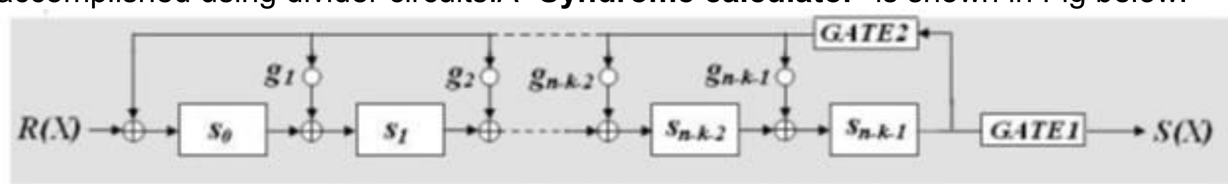
based on the received vector.The decoder, as in the case of linear block codes, first computes the syndrome to check whether ornot the received code vector is a valid code vector. In the case of cyclic codes, if the syndrome iszero, then the received code word polynomial must be divisible by the generator polynomial. If thesyndrome is non-zero, the received word contains transmission errors and needs error correction. Letthe received code vector be represented by the polynomial

$$R(X) = r_0+r_1X+r_2X^2+\ldots+r_{n-1}X^{n-1}$$

Let **A(X)** be the quotient and **S(X)** be the remainder polynomials resulting from the division of
R(X) by **g(X)** i.e.

$$\frac{R(X)}{g(X)} = A(X) + \frac{S(X)}{g(X)}$$

The remainder **S(X)** is a polynomial of degree **(n-k-1)** or less. It is called the "Syndrome polynomial".
If **E(X)** is the polynomial representing the error pattern caused by the channel, then we have:
R(X) =V(X) + E(X)
And it follows as **V(X) = U(X) g(X)**, that:
E(X) = [A(X) + U(X)] g(X) +S(X)
That is, the syndrome of R(X) is equal to the remainder resulting from dividing the error pattern bythe generator polynomial; and the syndrome contains information about the error pattern, which canbe used for error correction. Hence syndrome calculation can be accomplished using divider circuits.A" **Syndrome calculator**" is shown in Fig below.



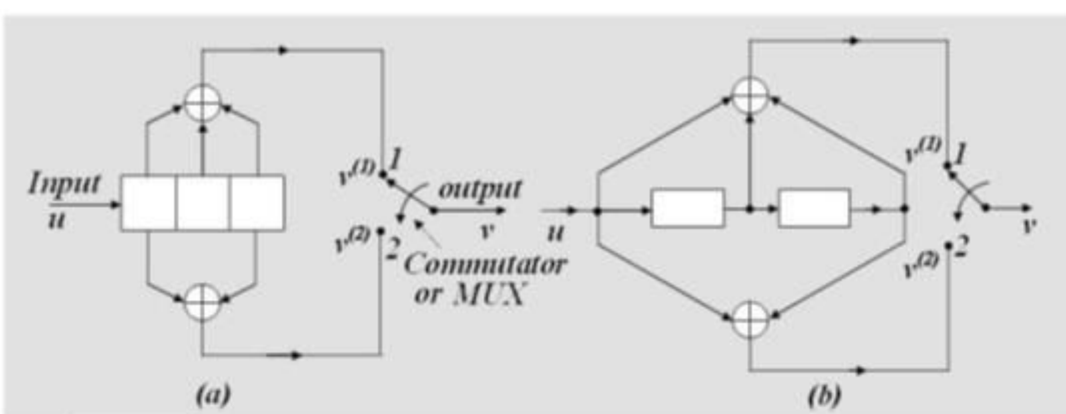
Syndrome calculator using (n-k) Shift registers

The syndrome calculations are carried out as below:
1 The register is first initialized. With **GATE 2 -ON** and **GATE1- OFF**, the received vector isentered into the register
2 After the entire received vector is shifted into the register, the contents of the register will bethe syndrome, which can be shifted out of the register by turning **GATE-1 ON** and **GATE-2OFF**. The circuit is ready for processing next received vector.
Theerror correction procedure consists of the following steps:
Step1. Received data is shifted into the buffer register and syndrome registers with switches
SINclosed and **SOU**Topen and error correction is performed with **SIN**open and **SOUT** closed.
Step2. After the syndrome for the received code word is calculated and placed in thesyndrome
register, the contents are read into the error detector. The detector is a combinatorialcircuit designed to output a „ **1**" if and only if the syndrome corresponds to

a correctableerror pattern with an error at the highest order position X^{n-l}. That is, if the detector outputis a '**1**' then the received digit at the right most stage of the buffer register is assumed to bein error and will be corrected. If the detector output is '**0**' then the received digit at the rightmost stage of the buffer is assumed to be correct. Thus the detector output is the estimateerror value for the digit coming out of the buffer register.
Step3. In the third step, the first received digit in the syndrome register is shifted right once. Ifthefirst received digit is in error, the detector output will be '**1**' which is used for errorcorrection. The output of the detector is also fed to the syndrome register to modify thesyndrome. This results in a new syndrome corresponding to the „ **altered** „received codeword shifted to the right by one place.
Step4. The new syndrome is now used to check and correct the second received digit, whichisThe multiplication operation, performed by the circuit, is listed in the Table below step by step. In shift number 4, „ **000"** is introduced to flush the registers. As seen from the tabulationnow at the right most position, is an erroneous digit. If so, it is corrected, a new syndromeis calculated as in step-3 and the procedure is repeated.
Step5. The decoder operates on the received data digit by digit until the entirereceived code wordis shifted out of the buffer.

CONVOLUTIONAL CODES
In block codes, a block of **n**-digits generated by the encoder depends only on the block of **k**datadigits in a particular time unit. These codes can be generated by combinatorial logic circuits. In aconvolutional code the block of **n**-digits generated by the encoder in a time unit depends on not onlyon the block of **k**-data digits with in that time unit, but also on the preceding „ **m**" input blocks. An (**n,k, m**) convolutional code can be implemented with **k**-input, **n**-output sequential circuit withinputmemory **m**. Generally, **k** and **n** are small integers with **k < n** but the memory order **m** must be madelarge to achieve low error probabilities. In the important special case when **k = 1**, the informationsequence is not divided into blocks but can be processed continuously.

Connection Pictorial Representation:
The encoder for a (rate **1/2**, **K** = **3**) or (**2, 1, 2**) convolutional code is shown in. Bothsketches shown are one and the same.



(2, 1, 2) Encoder (a) Representation using 3-bit shift register.
(b)Equivalent representation requires only two shift register stages.

Fig.Con1

At each input bit time one bit is shifted into the left most stage and the bits that were present in theregisters shifted to the right by one position. Output switch (commutator /MUX) samples the outputof each X-OR gate and forms the code symbol pairs for the bits introduced. The final code is obtainedafter flushing the encoder with "**m**" zero's where '**m**'- is the memory order (In Fig.con.1, **m = 2**). Thesequence of operations performed by the encoder of Fig.con.1 for an input sequence **u = (101)** areillustrated diagrammatically in Fig con.2.

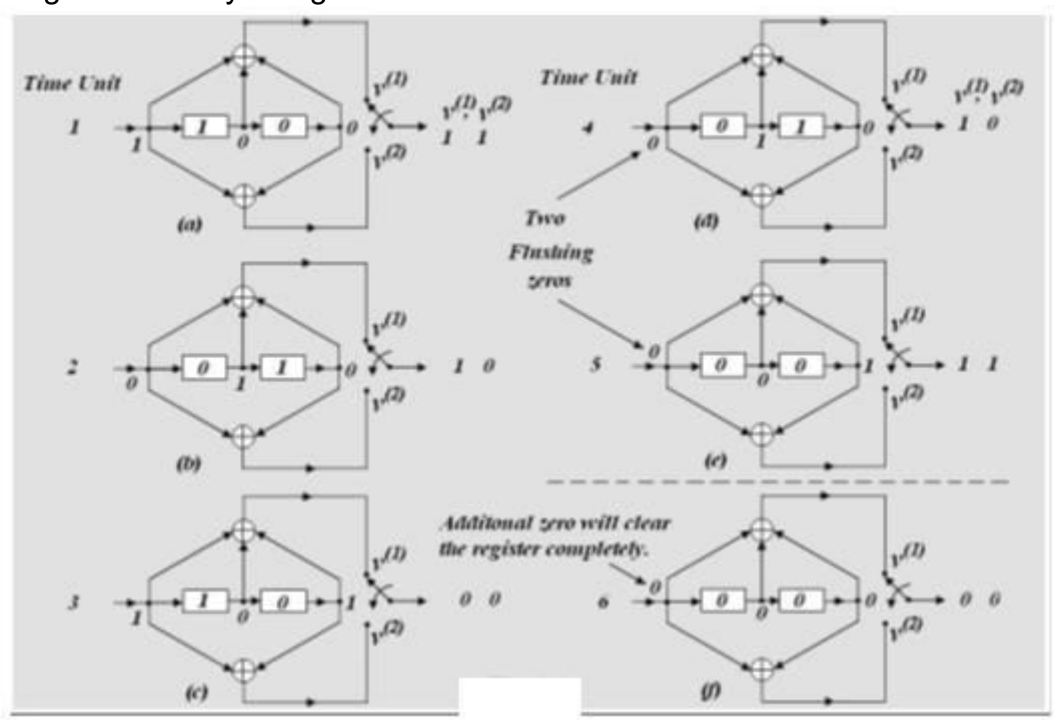
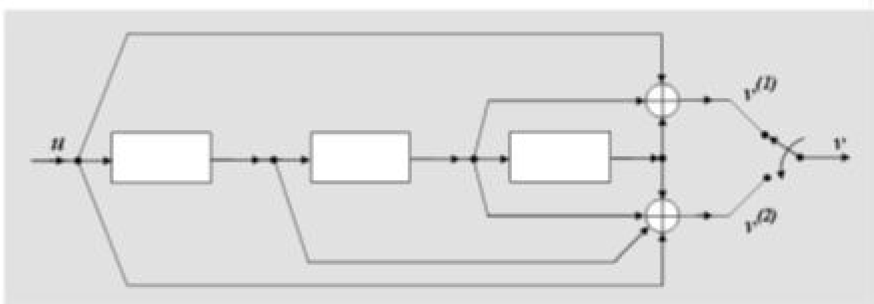


Fig Con2
From Fig con.2, gives the encoding procedure.

.

Convolutional Encoding – Time domain approach:
The encoder for a (**2, 1, 3**) code is shown in Fig. con3. Here the encoder consists of **m=3** stageshift register, **n=2** modulo-2 adders (X-OR gates) and a multiplexer for serializing the encoderoutputs. Notice that module-2 addition is a linear operation and it follows that all convolutionencoders can be implemented using a " **linear feed forward shift register circuit**".
The "information sequence" **u = (u1, u2, u3…….)** enters the encoder one bit at a time starting from**u1**. As the name implies, a convolutional encoder operates by performing convolutions on theinformation sequence. Specifically, the encoder output sequences, in this case$\mathbf{v^{(1)}=\{v1^{(1)}, v2^{(1)}, v3^{(1)}\ldots\}}$and $\mathbf{v^{(2)}= \{v1^{(2)},v2^{(2)},v3^{(2)}\ldots\}}$are obtained by the discrete convolution of the information sequencewith the encoder "impulse responses'. The impulse responses are obtained by determining the outputsequences of the encoder produced by the input sequence **u= (1, 0, 0, 0…)** .The impulse responses sodefined are called **'generator sequences**' of the code. Since the encoder has a **m-** time unit memory theimpulse responses can last at most **(m+ 1)** time units (That is a

total of **(m+ 1)** shifts are necessary fora message bit to enter the shift register and finally come out) and are written as:$\mathbf{g^{(i)}= \{g1^{(i)}, g2^{(i)},g3^{(i)}\ldots g\ m+1^{(i)}\}}$.



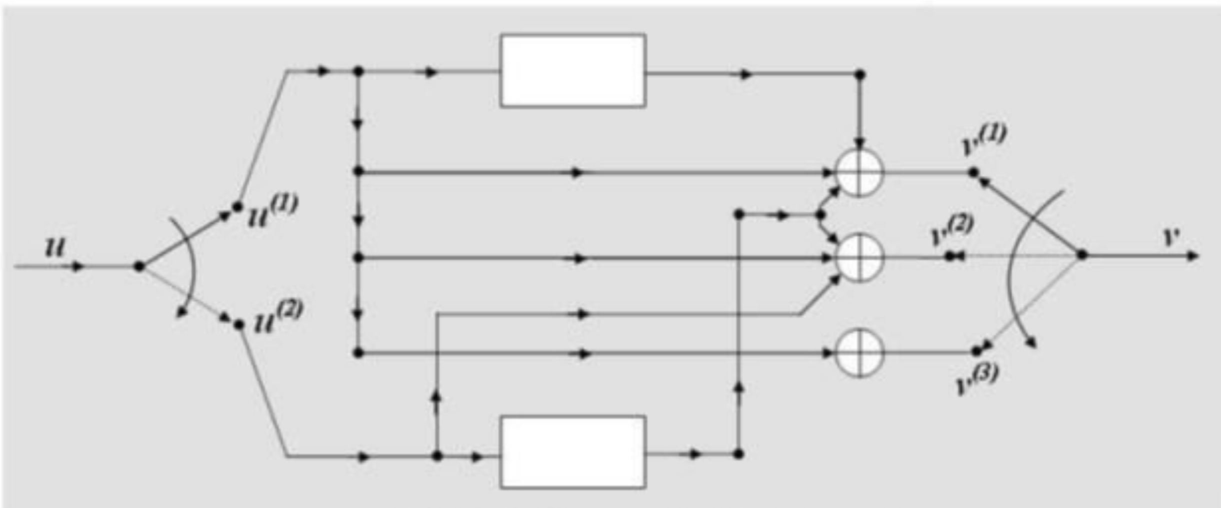
(2, 1, 3) binary encoder.

Fig Con3
For the encoder of Fig.con.3, we require the two impulse responses,
$\mathbf{g^{(1)}= \{g1^{(1)}, g2^{(1)}, g3^{(1)}, g4^{(1)}\}}$and $\mathbf{g^{(2)}= \{g1^{(2)}, g2^{(2)}, g3^{(2)}, g4^{(2)}\}}$
By inspection, these can be written as: $\mathbf{g^{(1)}= \{1, 0, 1, 1\}}$ and $\mathbf{g^{(2)}= \{1, 1, 1, 1\}}$
Observe that the generator sequences represented here is simply the 'connection vectors' of theencoder. In the sequences a '**1**' indicates a connection and a '**0**' indicates no connection to thecorresponding X - OR gate. If we group the elements of the generator sequences so found in to pairs,we get the overall impulse response of the encoder, Thus for the encoder of Fig con.3, the „over-allimpulse response" will be:
v = (11, 01, 11, 11)
The encoder outputs are defined by the convolution sums:
v (1) = u * g (1) …………………. (eqn con.1 a)
v (2) = u * g (2) …………………. (eqn con1.b)
Where * denotes the „discrete convolution", which i mplies:

$$v_l^{(j)} = \sum_{i=0}^{m} u_{l-i} g_{i+1}^{(j)} = u_l g_1^{(j)} + u_{l-1}\ g_2^{(j)} + u_{l-2}\ g_3^{(j)} + \ldots + u_{l-m}\ g_{m+1}^{(j)}$$

(eqn con2)

for**j = 1, 2** and where **ul-i= 0** for all **l<i** and all operations are modulo - **2**. Hence for the encoder ofFig (con3), we have:
$\mathbf{vl^{(1)}}= \mathbf{u_l + u_{l-2} + u_{l-3}}$
$\mathbf{vl^{(2)}}= \mathbf{u_l + u_{l-1} + u_{l-2} + u_{l-3}}$
This can be easily verified by direct inspection of the encoding circuit. After encoding, the
two output sequences are multiplexed into a single sequence, called the "**code word**" for transmissionover the channel. The code word is given by:
$\mathbf{v = \{v1^{(1)}v1^{(2)}, v2^{(1)}v2^{(2)}, v3^{(1)}v3^{(2)}\ldots\}}$
Fig.con.4. Here, as **k =2**, the encoder consists of two **m = 1** stage shift registers together with **n = 3**modulo -2 adders and two multiplexers. The information sequence enters the encoder **k = 2** bits at atime and can be written as $\mathbf{u= \{u1\ (1)\ u1^{(2)}, u2\ (1)\ u2^{(2)}, u3\ (1)\ u3\ (2) \ldots\}}$ or as two separate inputsequences:
$\mathbf{u^{(1)}= \{u1^{(1)}, u2^{(1)}, u3\ (1) \ldots\}}$ and$\mathbf{u^{(2)}= \{u1^{(2)}, u2^{(2)}, u3\ (2) \ldots\}}$.



A (3, 2, 1) convolutional encoder

Fig con 4

There are three generator sequences corresponding toeach input sequence. Letting

$\mathbf{gi^{(j)} = \{gi,1^{(j)}, gi,2^{(j)}, gi,3^{(j)} \ldots g\ i,m+1^{(j)}\}}$

input**i** and output **j**. The generator sequences for the encoder are:

$\mathbf{g1^{(1)} = (1, 1), g1^{(2)} = (1, 0), g1^{(3)} = (1, 0), g2^{(1)} = (0, 1), g2^{(2)} = (1, 1), g2^{(3)} = (0, 0)}$

The encoding equations can be written as:

$\mathbf{v\ (1) = u\ (1) * g1\ (1) + u^{(2)} * g2\ (1)}$……………………. (eqn con.5 a)

$\mathbf{v\ (2) = u^{(1)} * g1\ (2) + u\ (2) * g2\ (2)}$ ……………………. (eqn con5b)

$\mathbf{v\ (3) = u\ (1) * g1\ (3) + u\ (2) * g2\ (3)}$ …………………… (eqn con.5 c)

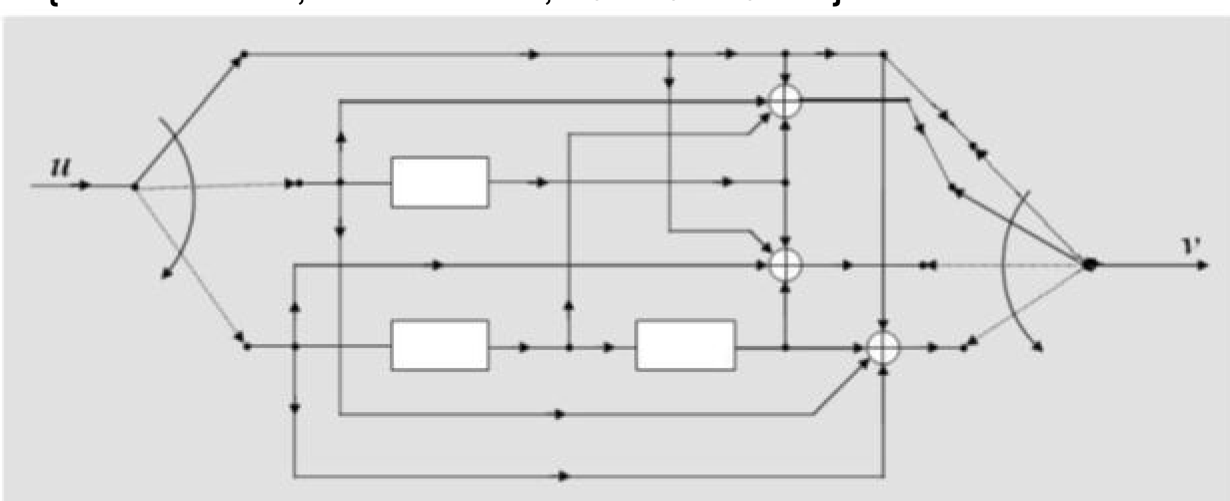
The convolution operation implies that:

$\mathbf{v\ l^{(1)} = u\ l^{(1)} + u\ l\text{-}1^{(1)} + u\ l\text{-}1^{(2)} v\ l^{(2)} = u\ l^{(1)} + u\ l^{(2)} + u\ l\text{-}1^{(2)} v\ l^{(3)} = u\ l^{(1)}}$

as can be seen from the encoding circuit.

After multiplexing, the code word is given by:

$\mathbf{v = \{ v\ 1^{(1)} v\ 1^{(2)} v\ 1^{(3)}, v\ 2^{(1)} v\ 2^{(2)} v\ 2^{(3)}, v\ 3^{(1)} v\ 3^{(2)} v\ 3^{(3)} \ldots \}}$



A (4, 3, 2) binary convolutional encoder

Fig con 5

Since each information bit remains in the encoder up to (**m + 1**) time units and during eachtime unit it can affect any of the **n**-encoder outputs (which depends on the shift register connections)it follows that "**the maximum number of encoder outputs that can be affected by a singleinformation bit**" is

$$n_A \triangleq n(m+1)$$ (eqn con8)

„ **nA**" is called the 'constraint length" of the code output isusually denoted as **K.** For the encoders of Fig con.3, con.4 and con.5 have values of **K** = **4, 2** and **3**respectively. The encoder in Fig con.3 will be accordingly labeled as a „ **rate 1/2, K = 4**" convolutionalencoder. The term **K** also signifies the number of branch words in the encoder"s impulse response.

$$\frac{k/n - kL/n(L+m)}{k/n} = \frac{m}{L+m}$$

and is called "**fractional rate loss**". Therefore, in order to keep the fractional rate loss at a minimum(near zero), „ **L**" is always assumed to be much larger than „ **m**". For the information 'sequence of Example we have **L = 5**, **m =3** and **fractional rate loss = 3/8 = 37.5%**. If **L** is made 1000, thefractional rate loss is only **3/1003≈0.3%**.

Encoding of Convolutional Codes; Transform Domain Approach:

In any linear system, we know that the time domain operation involving the convolution integral can be replaced by the more convenient transform domain operation, involving polynomialmultiplication. Since a convolutional encoder can be viewed as a 'linear time invariant finite statemachine, we may simplify computation of the adder outputs by applying appropriate transformation.

As is done in cyclic codes, each 'sequence in the encoding equations can' be replaced by acorresponding polynomial and the convolution operation replaced by polynomial multiplication. Forexample, for a (**2, 1, m)** code, the encoding equations become:

$\mathbf{v^{(2)}(X) = v1^{(2)} + v2^{(2)}X + v3^{(2)}X^2 + ...}$are the encoded polynomials.

$\mathbf{g^{(1)}(X) = g1^{(1)} + g2^{(1)}X + g3(1)X^2 + ...}$, and$\mathbf{g^{(2)}(X) = g1^{(2)} + g2^{(2)}X + g3^{(2)}X^2 + ...}$

are the "generator polynomials" of' the code; and all operations are modulo-2. After multiplexing, thecode word becomes:

$\mathbf{v(X) = v^{(1)}(X^2) + X\, v^{(2)}(X^2)}$

The indeterminate '**X**' can be regarded as a "**unit-delay operator**", the power of **X** defining thenumber of time units by which the associated bit is delayed with respect to the initial bit in thesequence.

Example:

For the (**2, 1, 3**) encoder of Fig con.3, the impulse responses were: $\mathbf{g^{(1)} = (1,0,1,1)}$, and $\mathbf{g^{(2)} = (1,1,1,1)}$

The generator polynomials are: $\mathbf{g^{(l)}(X) = 1 + X^2 + X^3}$, and $\mathbf{g^{(2)}(X) = 1 + X + X^2 + X^3}$

For the information sequence **u = (1, 0, 1, 1, 1);** the information polynomial is:

$\mathbf{u(X) = 1+X^2+X^3+X^4}$

The two code polynomials are then:

$\mathbf{v^{(1)}(X) = u(X)\, g^{(l)}(X) = (1 + X^2 + X^3 + X^4)(1 + X^2 + X^3) = 1 + X^7 v^{(2)}(X) = u(X)\, g^{(2)}(X)}$

$\mathbf{= (1 + X^2 + X + X^4)(1 + X + X^2 + X^3) = 1 + X + X^3 + X^4 + X^5 + X^7}$

From the polynomials so obtained we can immediately write:
$\mathbf{v^{(1)}= (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1)}$, and $\mathbf{v^{(2)}= (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1)}$
Pairing the components we then get the code word **v = (11, 01, 00, 01, 01, 01, 00, 11).**
We may use the multiplexing technique of the last code word equation and write:
$\mathbf{v(1)(X^2) = 1 + X^{14}}$
and$\mathbf{v^{(2)}(X^2) = 1+X^2+X^6+X^8+X^{10}+X^{14}}$; $\mathbf{Xv^{(2)}(X^2) = X + X^3+ X^7+ X^9+ X^{11}+ X^{15}}$;
and the code polynomial is: $\mathbf{v(X) = v^{(1)}(X^2) + X\ v^{(2)}(X^2) = 1 + X + X^3+ X^7+ X^9+ X^{11}+ X^{14}+ X^{15}}$
Hence the code word is: **v = (1 1, 0 1, 0 0, 0 1, 0 1, 0 1, 0 0, 1 1);** this is exactly the same as obtainedearlier.

Shannon – Fano Binary Encoding Method:

Shannon – Fano procedure is the simplest available. Code obtained will be optimum if and only if $p_k = r^{-l_k}$.The procedure is as follows:

1. List the source symbols in the order of decreasing probabilities.
2. Partition this ensemble into almost two equi- probable groups.

Assign a „ ***0***" to one group and a „ ***1***" to the other group. These form the starting code symbols of the codes.

3. Repeat steps ***2*** and ***3*** on each of the subgroups until the subgroups contain only one source symbol, to determine the succeeding code symbols of the code words.
4. For convenience, a code tree may be constructed and codes read off directly.

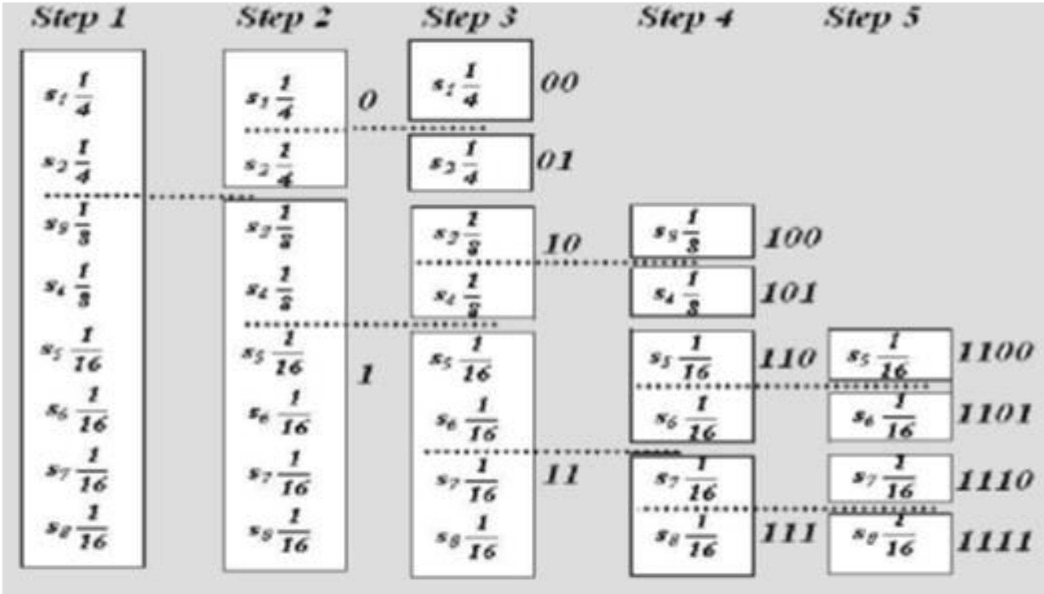
Example

Consider the message ensemble ***S = {s1, s2, s3, s4, s5, s6, s7, s8}*** with
P = ¼, ¼, 1/8 ,1/8 ,1/16, 1/16, 1/16, 1/16 X = { 0 ,1 }
The procedure is clearly indicated in the Box diagram shown below. The Tree diagram for the steps followed is also shown in Fig below. The codes obtained are also clearly shown. For this example,

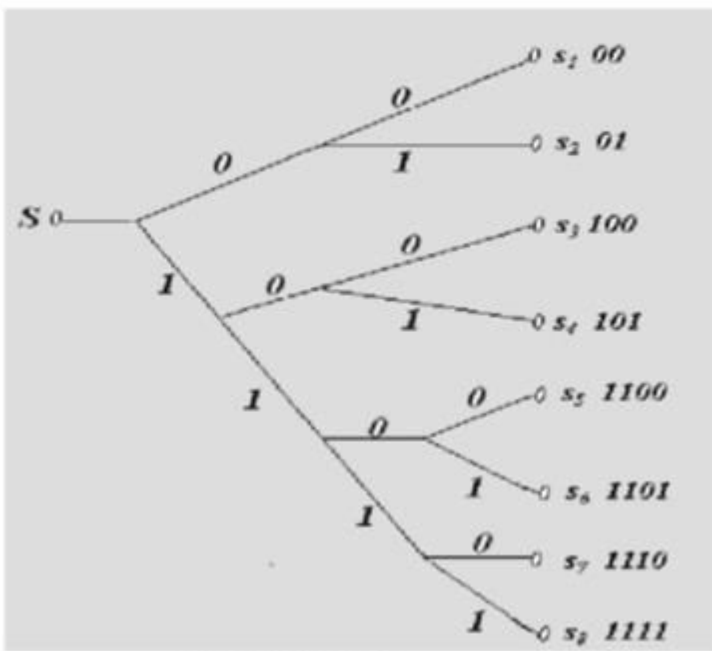
$L = 2\times 1/4 + 2\times 1/4 + 3\times 1/8 + 3\times 1/8 + 4\times 1/16 + 4\times 1/16 + 4\times 1/16 + 4\times 1/16 = 2.75$ ***binits / symbol***

$$H(S) = 2\times\frac{1}{4}\log 4 + 2\times\frac{1}{8}\log 8 + 4\times\frac{1}{16}\log 16 = 2.75 \text{ bits/symbol.}$$

And as $\log r = \log 2 = 1$, we have $\eta_c = \dfrac{H(S)}{L\log r} = 100\%$ and $E_c = 0\%$



Box Diagram.



Tree diagram.

Incidentally, notice from tree diagram that the **codes originate from the same source and divergeintodifferent tree branches** and hence it is clear that no complete code can be a prefix of any othercode word.

Thus the Shannon- Fano algorithm provides us a means for constructing optimum, instantaneous codes.

In making the partitions, remember that the symbol with highest probability should be made
to correspond to a code with shortest word length. Consider the binary encoding of the following

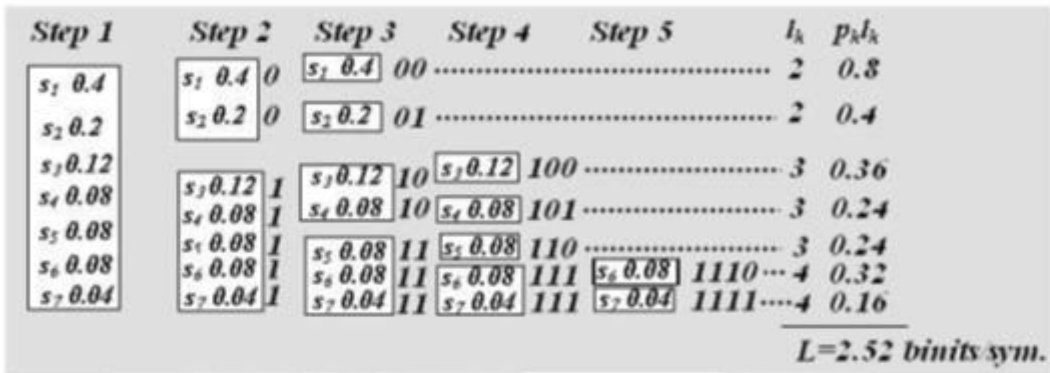
message ensemble.

Example:

S = {s1, s2, s3, s4, s5, s6, s7, s8}

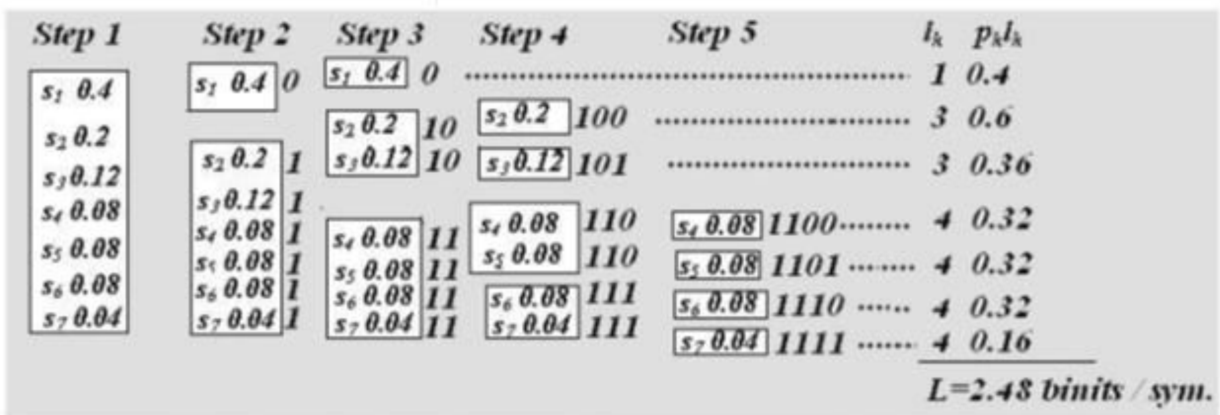
P = $\left\{ \boldsymbol{0.4 \text{ , } 0.2 \text{ , } 0.12 \text{ , } 0.08 \text{ , } 0.08 \text{ , } 0.08 \text{ , } 0.04} \right\}$

Method - I



Partition diagram for Method- I

Method – II



Partition diagram for Method-II

For the partitions adopted, we find ***L=2.52 binits / sym***. for the ***Method – I***
L=2.48 binits/symfor the ***Method – II***

For this example, ***H(S) =2.420504 bits/sym***and
For the first method, η_{c1}=***96.052%***
For the second method, η_{c2}=***97.6***

This example clearly illustrates the logical reasoning required while making partitions. TheShannon – Fano algorithm just says that the message ensemble should be partitioned into two almostequi-probable groups. While making such partitions care should be taken to make sure that thesymbol with highest probability of occurrence will get a code word of minimum possible length. Inthe example illustrated, notice that even though both methods are dividing the message ensemble intotwo almost equi-probable groups, the Method – II as signs a code word of smallest possible length tothe symbol **s1**.

Compact code: Huffman"s Minimum Redundancy code:
for an optimum coding we require:

1) Longer code word should correspond to a message with lowest probability.
1) Longer code word should correspond to a message with lowest probability.
2) 2) $l_k \geq l_{k-1} \forall\ k = 1,2, \ldots\ldots q-r+1$
3) (3) $l_{p-r} = l_{q-r-1} = l_{q-r-2} = \ldots.. = l$(4) The codes must satisfy the prefix property.

Huffman has suggested a simple method that guarantees an optimal code. The procedure consists of step- by- step reduction of the original source followed by acode construction, starting with the final reduced source and working backwards to the originalsource. The procedure requires α steps, where

$$q = r + \alpha\ (r-1)$$

The procedure is as follows:

1. List the source symbols in the decreasing order of probabilities.

2. Check if $q = r + \alpha(r-1)$ is satisfied and find the integer „ α" . Otherwise add suitable number of dummy symbols of zero probability of occurrence to satisfy the equation. ***This step is not required if we are to determine binary codes.***

3. Club the last ***r*** symbols into a single composite symbol whose probability of occurrence is equal to the sum of the probabilities of occurrence of the last ***r*** – symbols involved in the step.

4. Repeat steps ***1*** and ***3*** respectively on the resulting set of symbols until in the final step exactly ***r***- symbols are left.

5. Assign codes freely to the last ***r***-composite symbols and work backwards to the original source to arrive at the optimum code.

6. Alternatively, following the steps carefully a tree diagram can be constructed starting from the final step and codes read off directly.

7. Discard the codes of the dummy symbols.

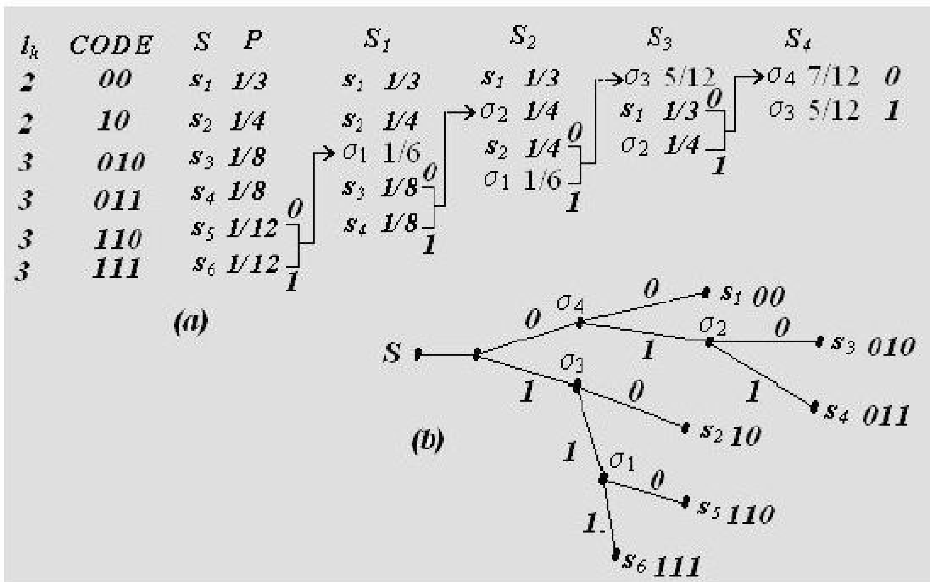
Example 6.12: (*Binary Encoding*)

S = {s1, s2, s3, s4, s5, s6}, X = {0, 1};

$$P = \frac{1}{3}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{12}, \frac{1}{12}$$

there can be as many as ***2. (2.2+2.2) = 16*** possible instantaneous code patterns. For example we can take thecompliments of First column, Second column, or Third column and combinations there of asillustrated below.

	Code	I	II	III
s_1	00	10	11	11
s_2	10	00	01	01
s_3	010	110	100	101
s_4	011	111	101	100
s_5	110	010	000	001
s_6	111	011	001	000



Code I is obtained by taking complement of the first column of the original code. ***Code II*** is obtained by taking complements of second column of ***Code I***. ***Code III*** is obtained by taking complements of third column
Code II. However, notice that, ***lk***, the word length of the code word for ***sk***is a constant for all possible codes.

For the binary code generated, we have:

$$L = \sum_{k=1}^{6} p_k l_k = \frac{1}{3}\times 2 + \frac{1}{4}\times 2 + \frac{1}{8}\times 3 + \frac{1}{8}\times 3 + \frac{1}{12}\times 3 + \frac{1}{12}\times 3 = \frac{29}{12} \text{ binits/sym} = 2.4167 \text{ binits/sym}$$

$$H(S) = \frac{1}{3}\log 3 + \frac{1}{4}\log 4 + 2\times\frac{1}{8}\log 8 + 2\times\frac{1}{12}\log 12$$

$$= \frac{1}{12}(6\log 3 + 19) \text{ bits/sym} = 2.3758 \text{ bits/ sym}$$

$$\therefore \eta_c = \frac{6\log 3 + 19}{29} = 98.31\%; E_c = 1.69\%$$