FORWARD ERROR CORRECTION FOR IMPERFECT DATA IN PACKET RADIO

W. Kinsner, VE4WK

Department of Electrical and computer Engineering
university of Manitoba
Winnipeg, Manitoba, Canada R3T-2N2
E-mail: VE4WK@VE4BBS.MB.CAN.NA

Abstract

Many current protocols employ retransmission to ensure error-free data transfers. This is necessary for perfect data where a loss of a single bit is catastrophic. For imperfect data, such as digitized speech transmitted in real time in which a loss of one bit in a 1000 may be acceptable, retransmission is not possible and forward error correction should be used for error control. This paper presents a review of suitable codes for such error control, as well as several code implementations including a modified (8,4,4) Hamming code, (15,7) Bose-Chaudhury-Hocquenghem (BCH) code, and a concatenated code capable of correcting not only random errors but also burst errors. The code has an outer (7,4,3) Hamming code, an inner self-orthogonal 1/2 convolutional code, and a code word interlace matrix.

1. INTRODUCTION

There is a need for reliability in every communication system. A major problem in digital data communication systems is the introduction of errors due to a noisy channel. Unlike other parts of the system where transmission errors can be minimized or eliminated by careful design, the channel is not under control and transmission errors inevitably occur under noisy conditions. The existing error control scheme in amateur packet radio is retransmission whenever errors are detected in the received packets or when packets are lost. The detection is often based on either checksum words or cyclic redundancy code (CRC) words. However, retransmission is inefficient since it wastes both time and energy. When the communication medium is very noisy, retransmission may fail as every packet could contain errors. Moreover, retransmission is impractical for real-time applications and uni-directional communication. But above all, we have been using retransmission even though perfect data transmission is not required. imperfect data such as non-critical text, digitized voice, and video may be acceptable with a small number or errors. Throughput of such systems could be increased considerably with a moderate error control by correcting (reconstructing) the bits in error at the receiving end, based on redundant bits inserted into the packet by the sender. This reconstruction scheme is *called forward error correction* (FEC), as opposed to ***the automatic repeat request (ARQ)*** or backward error correction (BEC) by retransmission.

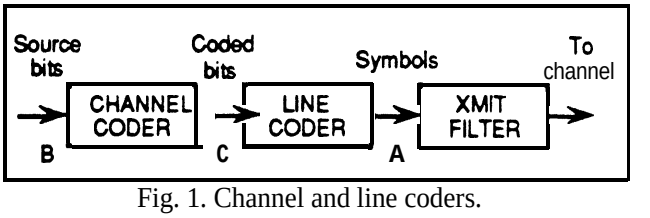

Fig. 1. Channel and line coders.

Coding refers to the translation between the source ***bits*** (user-provided message bits) and the transmitted ***data symbols (coded*** symbols) [Blah87, McEl87]. System performance can be improved by (i) fine coding to control the power spectrum of the transmitted signal and (ii) ***channel coding*** to control errors [LeMe88]. The line coding changes the statistics of the data symbols to remove undesired correlations among the source bits so as to make them more random or uniformly active (through scrambling), while the channel coding introduces controlled correlation between data symbols through redundancy to detect and correct channel errors. Figure 1 shows a channel coder translating source bits into coded bits to achieve error detection, correction , or even prevention. The line coder further improves the probability of correct transmission. The decoding process may ***be hard (the inverse*** of the process of Fig. 1, with explicit decoding of all the symbols) or ***soft*** (direct decoding of the information bits from the received signal, without the detection of any intermediate symbols) [LeMe88].

Since this paper concentrates on the channel coding to handle random and burst errors simultaneously through simple block and convolutional codes, respectively, several pertinent definitions are required. Channel coding is concerned with two general classes of codes: block and convolutional. In the ***block code class,*** a sequence of source bits is segmented into blocks of k bits and translated into n code bits, with $n>k$ and $p=(n-k)$ redundant bits. The (n,k) ***code*** depends on the current block of k source bits only ***(memoryless coder). The*** block code is said to have code rate $r=k/n$.

On the other hand, a ***convolutional coder inserts*** redundant bits without segmenting the source stream into blocks. Instead, it processes the source stream either bit-by-bit or small groups of bits at a time. The code depends not only on the current input bits, but also on a finite number of past source bits ***(coder with memory).*** Convolutional codes produce results at least as good as the best block codes due to the availability of practical soft decoding techniques. Performance of an uncoded and coded system is measured by ***coding gain*** which is the difference in signal to noise ratio (SNR) required at the detector to achieve a fixed probability of either bit or symbol error.

By combining two or more simple codes, a good low-rate ***concatenated code can be constructed*** [MiLe85]. Here, the source stream is encoded with an (n,k) outer ***code*** and then further encoded as a sequence of (N,K) inner code blocks. The function of the outer code is to decode random or burst errors that have slipped through the inner decoder. The outer code may be made more effective if error bursts following the inner decoder are spread among consecutive outer code blocks by a process called ***interleaving*** or ***interlacing*** [Hayk88]. A simple line coder in the form of an interlace matrix can be used to assist in distributing such burst errors. Although the concatenated code is less powerful than a single-stage code with the same code rate and block length, the decoder is simpler due to the partitioning of the decoding stages.

2 BLOCK CODES

2.1 The Cross-Parity Code

One of the simplest error correcting codes is the cross-parity code based on horizontal and vertical parity checks, as shown in Fig. 2. Each word in the block has a horizontal parity bit added (to make the total number of 1s even). Vertical parity bits are computed from all the successive words and added to the columns. The parity bits can be generated by taking modulo-2 addition denoted by $\oplus$ (exclusive-OR gate). When a single error occurs in the block during its transmission (e.g., bit b_2 in the 3rd word is flipped to 0), then the parity bits for that row and column computed at the receiver site will be 1s rather than 0s, thus indicating the location of the bit in error which can be corrected by flipping. Correction of a single error is, therefore, possible without retransmission.

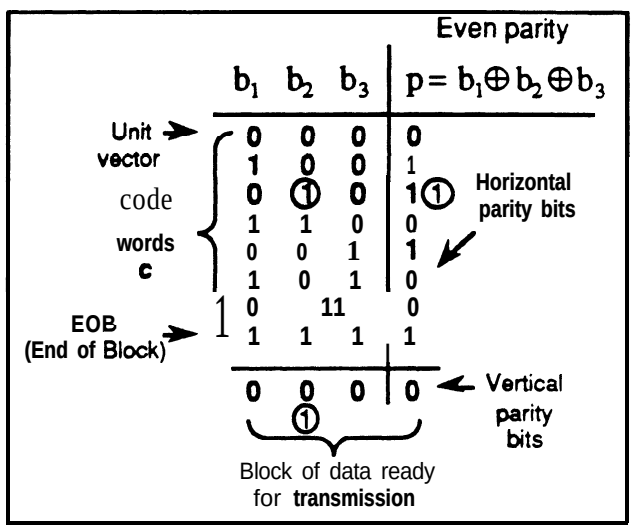

Fig. 2. Horizontal and vertical parity error correction scheme.

Notice that our numbering of bits in a word is from left-to-right, which is consistent with the majority of coding theory literature. This convention is related to matrix notation used in the early description of codes. This is contrast to the usual right-to-left number convention showing the weights of each digit in a number, as well as the modem literature that is based on polynomials, rather then matrices. The structure of the code can be described by total number of source bits

$$k = h \times v \tag{1}$$

where h and v are the numbers of rows and columns in the source block, respectively, the number of redundant parity bits

$$p = h+v+1 \tag{2}$$

and the total number of bits

$$n = k+p \tag{3}$$

giving the code rate of

$$r = k/n \tag{4}$$

For the code shown in Fig. 2, $k = 8x3 = 24$, $p = 8+3+1 = 12$, $n = 24+12 = 36$, and $r = 24/36 = 2/3$.

The code is important not only from the educational point of view, but also due to its good code rate, no limit on the size of the source block, and the simultaneous single-error correction (SEC) and double-error detection (DED) capability, also called SEC-DED. Three errors may also be detected. What properties make the code suitable for correction? *It is seen from Fig. 2 that any* two code words differ by at least two bits. This is the most important observation determining the ability of a code to detect and correct a specific number of random errors. In other words, if the distance between two words is large enough, an error produces a unique word outside the transmitted set of code words, and the original code word can be reconstructed.

The number of places where two words (also called ***row matrices*** or ***vectors,*** denoted by bold characters) u and v differ is called the ***Hamming distance*** and is computed by

$$d(u,v) = w(u \oplus v) \quad (5)$$

where $w(\bullet)$ is the ***Hamming weight*** defined as the number of nonzero elements in the resulting vector. For example, if we take the 3rd and 4th code words, then $d = w([0101] \oplus [1100]) = w([1001]) = 2$. ***The minimum distance,*** d_{min}, is the smallest distance between all the code words, and determines the number of errors that can be detected, t_d, and corrected, t_c, according to

$$d_{min} - 1 = t_d + t_c \quad \text{for } t_c \le t_d \quad (6)$$

For example, if dmin = 1, no detection or correction is possible, as any error converts the code word into another valid code word. For dmin = 2, a single error can be detected but not corrected (the h parity bit). With $d_{min} = 3$, either a single error can be detected and corrected, or if the probability of a double error is high, then such a double error can be detected but not corrected For $d_{min} = 4$, the SEC-DED scheme is possible. For dmin = 5, a double error can be detected and corrected. It is now Seen why we need the *h* and v parity bits together; since the *h* bit leads to dhmin = 2, it can detect an error in a row only, and the v bit ($d_{vmin} = 2$) is needed to locate the column where the bit in question is. Note that the cross-parity code is an example of a class of codes called ***product*** codes in which the total distance is $d_h d_v$ and the individual codes are not limited to the parity check only.

The block code of Fig. 2 has also another useful ***linear*** property because the modulo-2 sum of code words produces another code word. For example, the 3rd and 4th code words result in the 2nd code word ($[0101] \oplus [1100]) = [1001]$). This property makes encoding and decoding easy by using linear (X-OR) operations rather than random table look-up procedures.

The encoding and decoding may be further simplified because the axle is also ***systematic*** (all the parity bits are separated from the source bits). Still another useful property of the code is its ***cyclic*** form; i.e., when a code word is rotated (shifted cyclically), the resulting word is also a code word. For example, by rotating the 3rd code word on place to the right, the 6th code word results ($[0101] \rightarrow [1010]$).

The code word generation process can be expressed conveniently using matrix notation. Notice that a row code vector c is a concatenation of the row source vector b and the parity bit p, which generalizes to

$$\mathbf{c} = \mathbf{b}\mathbf{G} \quad (7)$$

where G is the ***generator matrix*** given by

$$\mathbf{G} = [\mathbf{I}_k, \mathbf{P}] \quad (8)$$

and $\mathbf{I}_k$ is the identity matrix of dimension ***kxk*** and P is the single column parity matrix. For the example of Fig. 2, the 2nd code word is generated by

$$[0011] = [001] \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (9)$$

2.2 Hamming Codes

2.2.1 Tk Nonsystematic (7,4,3) Hamming Code

In the 1950s, Hamming introduced a class of linear, cyclic, systematic and perfect correcting codes with distance of 3 and 4. A ***perfect code*** is one in which every bit in the code word may be corrected. The simple (7,3) Hamming code has $n = 7$ total bits, including $k = 3$ source bits and $p = 4$ parity bits (heavy overhead). It can be generated using LFSRs and correction can be done using majority logic. The (7,4) Hamming code improves the code rate and allows operation on 4-bit nibbles. Correction of a single error is done using a parallel syndrome vector generator which is the address of the bit in error. The simple syndrome vector is obtained by a proper coverage of the source bits by the parity bits, as shown in Fig. 3.

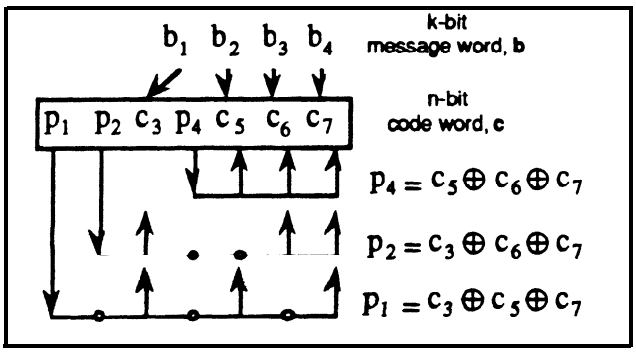

Fig. 3. Non-systematic (7,4) Hamming code.

For illustrative purposes, the source bits are interspersed with the parity bits so that the parity bits could occupy positions represented by the power of 2 ($i = 2^0=1; = 2^1=2; = 2^2=4;\ldots$). As shown by the arrows in Fig. 3, the index i of each p_i shows: (i) the group of bits to be checked for parity and (ii) the separation between the groups. This provides a binary coverage of the bits, with the most significant bit affecting all the parity bits, and the other bits affecting two parity bits (e.g., c_5 affects p_1 and p_4. For example, if

$$b = [b_1 b_2 b_3 b_4] = [c_3 c_5 c_6 c_7] = [0101] \quad (10)$$

then the parity bits are computed as

$$p_1 = c_3 \oplus c_5 \oplus c_7 = 0 \oplus 1 \oplus 1 = 0$$
$$p_2 = c_3 \oplus c_6 \oplus c_7 = 0 \oplus 0 \oplus 1 = 1$$
$$p_4 = c_5 \oplus c_6 \oplus c_7 = 1 \oplus 0 \oplus 1 = 0 \quad (11)$$

and the code word is

$$c = [p_1 p_2 c_3 p_4 c_5 c_6 c_7] = [0100101] \quad (12)$$

If we assume that c is transmitted as is (without any line coding) and that an error occurs in position c_5, then the received code word is $c^* = [0100001]$. The syndrome vector, s, is calculated as

$$s_1 = p_1 \oplus c_3 \oplus c_5 \oplus c_7 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$$
$$s_2 = p_2 \oplus c_3 \oplus c_6 \oplus c_7 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$$
$$s_4 = p_4 \oplus c_5 \oplus c_6 \oplus c_7 = 0 \oplus 0 \oplus 0 \oplus 1 = 1 \quad (13)$$

It is seen that since $s = [s_4 s_2 s_1] = [101]$ is the address of the location with error, an ordinary parallel binary decoder can correct the error.

The (7,4) code concept can be extended to more bits (n,k) with

$$p = n - k, \quad n = 2^p - 1, \quad k = n - p \quad (14)$$

For $p = 4$, $n = 16\text{-}1 = 15$ and $k = 15\text{-}4 = 11$ we have a (15,11) code with 11 source bits and a much improved code rate of $r = 1\ 1/15$.

The descriptive formulation of the (7.4) code can be presented in a more formal manner by using the following $p \times n$ *parity-check H matrix*

$$H = [h_1 h_2 h_3 h_4 h_5 h_6 h_7] = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (15)$$
$$[p_1\ p_2\ c_3\ p_4\ c_5\ c_6\ c_7]$$

Notice the binary-coded structure of the matrix from left to right. This matrix can can be used to construct any other distance-3 code. (For the distance to be at least 3, no two columns in H should be the same, or any three columns should add to 0.)

2.2.2 The Systematic (7,4,3) Hamming Code

The above analysis concentrated on a non-systematic form of the linear code which require parallel circuits for its generation and decoding. In practice, serial circuits (LFSR) are preferred since the bits are transmitted serially. To find an appropriate LFSR for the (7,4) code, the code must be cyclic and systematic. This can be achieved by changing the H matrix to a $k \times n$ ***generator matrix*** from which a ***generator polynomial*** could be derived. First, the matrix H can be changed from a non-systematic form to a systematic form by rearranging the columns for the most convenient encoding of decoding (this change preserves the SEC capability due to the uniqueness of the columns)

$$H = [h_1 h_2 h_3 h_4 h_5 h_6 h_7] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$
$$[c_5\ c_7\ c_6\ c_3\ p_4\ p_2\ p_1]$$

$$H = [P^T \mid I_p] \quad (17)$$

where P is the $p \times k$ coefficient matrix that defines how the parity bits are related to the message bits, T denotes transpose, and I_p is the $p \times p$ identity matrix. This systematic form *is* also called ***reduced echelon form*** [PeWe72] and can be obtained formally by row reduction into the canonical form [LeMe88]. It can be shown [RaFu89, MiLe85, LeMe88, Hayk88, ClCa81, Kuo81, Rode82, MiAh88] that the generator matrix is related to the paritycheck matrix through

$$G = [I_k \mid P] \quad (18)$$

where I is the $k \times k$ identity matrix. From Eq. 16, the (7,4) code has the following generator matrix

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (19)$$

and can be generated by taking

$$c = bG \quad (20)$$

Note that the generator vector g_4 in Eq. 19 is important because it specifies the generator matrix G completely. For example, the vector g_3 is obtained by shifting g_4 one place to the left. Next, vector g_2 is obtained by shifting g_3 to the left and adding (modulo-2) g_4 to it. Finally, g_1 is obtained by shifting g_2 left and adding g_4. Therefore, all the code words are generated by cyclic shifts of g_4.

The generator vector g_4

$$g_4 = [\,0\ 0\ 0\ 1\ 0\ 1\ 1\,] \quad (21)$$

is associated with a generator polynomial of the following general form

$$g(x) = x^p + \sum_{i=1}^{p-1} g_i x^i + 1, \quad g_i = \{0,1\} \quad (22)$$

where p is the number of redundant bits. For the (7,4) code, $p=3$ and g(x) becomes

$$g(x) = x^3 + x + 1 \quad (23)$$

More formally, the generator polynomial is found be factoring (x^n-1) into irreducible polynomials and selecting a primitive one of degree $p=(n-k)$.

We can now rewrite Eq. 19 in the polynomial form as

$$c(x) = b(x)g(x) \quad (24)$$

and the code is obtained by multiplying the message word with the generator polynomial using either parallel circuits or serial LFSRs. The problem here is that the multiplication almost always produces a non-systematic code. However, the message bits are not altered if they are just shifted to the left by as many places as the degree of the generator polynomial g(x). This is equivalent to the multiplication of b(x) by x^p. But to maintain c(x) unchanged, the right-hand-side of Eq. 24 must be adjusted by the remainder polynomial

$$c(x) = x^p b(x) + \text{rem}\left(\frac{x^p b(x)}{g(x)}\right) \quad (25)$$

where rem(a) denotes the remainder. The above expression constitutes the encoding algorithm for LFSRs: the message word b(x) is shifted to the left by p bits, and the p parity bits represented by the rem(•) are appended to the message word

2.23 *A (7,4,3) Hamming Encoder*

The most convenient circuit to compute the rem(e) is a LFSR whose structure is represented by the selected generator polynomial g(x). It is shown in Fig. 4.

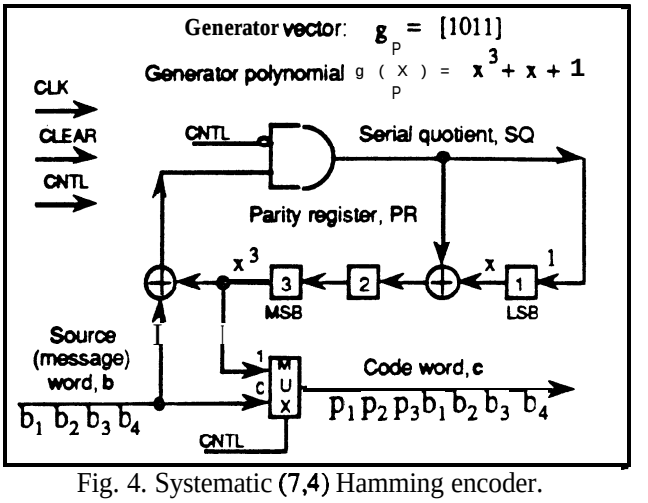

Fig. 4. Systematic (7,4) Hamming encoder.

The LFSR has $p=3$ *stages* (flip flops, FFs) in its *parity register,* PR, whose contents are first set to 0 at the beginning of each new message word, and then shifted by a clock. The stages are separated by modulo-2 adders at places determined by the g(x). The LFSR multiplies b(x) by x^p at its entry, and subtracts g(x) from its current contents whenever *the serial quotient* (SQ) is 1. The SQ is calculated from

$$SQ^{(t)} = FF_p \oplus b_{(k-t)},\ t=0,1,\ldots k-1 \quad (26)$$

where t denotes clock period, FF_p is the most-significant bit (MSB) of the FF, and $b_{(k-t)}$ are the successive message bits, starting from the MSB.

Table 1. Remainder generation in a LFSR.

CYLCE	CNTL	SQ	DATA N	REGISTER MSB LSB	OUT PUT
CLEAF	0	1	1	⊕ 0 0 ⊕ 0	1
shift 1	0	0	0	⊕ 0 1 ⊕ 1	0
2	0	0	1	⊕ 1 1 ⊕ 0	1
3	0	1	0	⊕ 1 0 ⊕ 0	0
4	1	0	x	0 1 ⊕ 1	0
5	1	0	x	1 1 0	1
6	1	0	x	1 0 0	1

Let us rewrite Eq. 9 in reverse sequence, consistent with polynomial notation

$$b = [b_4 b_3 b_2 b_1] = [1010] \quad (27)$$

and trace the events in the encoder, using Fig. 4 and Table 1. Following a CLEAR, the feedback path is closed by the application of CNTL=0, the multiplexer MUX connects the source input, b, to the coder output, and the SQ is generated as $SQ^{(0)} = FF_3 \oplus b_4 = 0 \oplus 1 = 1$. After the first shift, $FF_1 = 1$, $FF_2 = FF_1 \oplus SQ^{(0)} = 0 \oplus 1 = 1$, $FF_3 = 0$ from FF_2, and $SQ^{(1)} = FF_3 \oplus b_3 = 0 \oplus 0 = 0$. This process continues for three more clock periods, and the final remainder $p = [p_3\, p_2\, p_1] = [011]$ is appended to the message word by turning the feedback path off, switching the MUX to the PR, and applying $p=3$ extra clock periods to produce the correct code word c = [1010011].

The above operation can be confirmed using polynomial manipulations. The message vector of Eq. 27 has the following corresponding polynomial

$$b(x) = x^3 + x \quad (28)$$

and the remainder

$$p(x) = \text{rem}\left[\frac{x^3\,(x^3 + x)}{x^3 + x + 1}\right] \quad (29)$$

is computed using Euclidean long division as follows.

$$\begin{array}{r} x^3 + \quad 1 \\ x^3 + x + 1 \overline{)\, x^6 + x^4} \\ \underline{x^6 + x^4 + x^3} \\ x^3 \\ \underline{x^3 + x + 1} \\ \text{Remainder } p(x) = x + 1 \end{array} \quad (30)$$

By changing the remainder into its matrix form p = [011] and concatenating it with b, c is found as

$$c = b \mid p = [1010011] \quad (31)$$

Similarly, by using Eq. 25, we find

$$\begin{aligned} c(x) &= x^3\,(x^3 + x) + (x + 1) \\ &= x^6 + x^4 + x + 1 \end{aligned} \quad (32)$$

or

$$c = [1010011] \quad (33)$$

2.2.4 A (7,4,3) Hamming Decoder

Since syndrome generation can be considered as the inverse of code word generation, it can use the same LFSR multiplier/divider. The syndrome vector s is computed according to Eq. 25

$$\begin{aligned} s(x^p c^*(x)) &= x^p c^*(x) \bmod (g(x)) \\ &= \text{rem}\left(\frac{x^p c^*(x)}{g(x)}\right) \end{aligned} \quad (34)$$

If there is no error in c*(x), then s=0. If an error e(x) occurs, then the received signal is linear combination of c and e

$$c^*(x) = c(x) + e(x) \quad (36)$$

and the syndrome is due to the the error only

$$\begin{aligned} s(c^*(x)) &= s(c(x)) + s(e(x)) \\ &= c(x) \bmod (g(x)) + e(x) \bmod (g(x)) \\ &= 0 + e(x) \bmod (g(x)) \\ &= s(e(x)) \end{aligned} \quad (37)$$

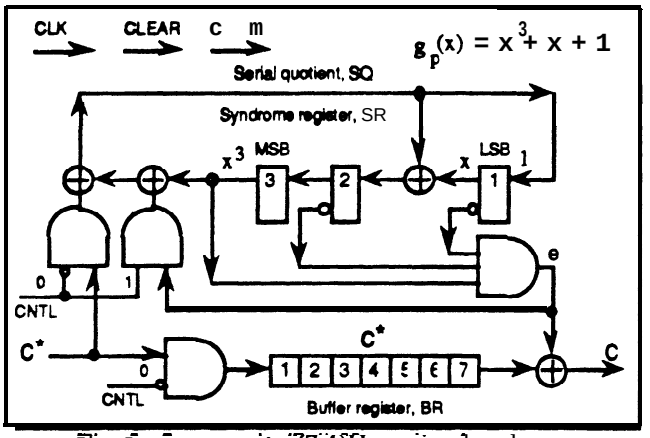

Fig. 5. Systematic (7,4) Hamming decoder.

Figure 5 shows how the received word $c^*(x)$ is fed to the syndrome generator and a buffer register, BR, while CNTL=0. After n=7 shifts, the syndrome is in the syndrome register (SR) and a copy of c^* is in BR. The control signal now becomes CNTL=1 to break the path of input c^* into the LFSR and BR, as well as to open a new feedback loop for the error correcting signal e. The error is corrected after another n clock *periods.*

Table 2. Syndrome vectors for systematic (7,4) *code.*

e(x)	s(e(x))	$s(x^3 e(x))$
1	1	x+1
x	x	x^2+x
x^2	x^2	x^2+x+1
x^3	x+1	x^2+1
x^4	x^2+x	1
x^5	x^2+x+1	x
x^6	x^2+1	x^2

Table 2 shows a polynomial form of the errors and syndromes. If the error is in the MSB (e(x)= x^6), then $s(x^3e(x))=x^2$ (or s=[100]) and the correcting AND gate generates $e=1$ which corrects the error on the 7th clock period. If the error is in the 4th position, (e(x)= x^3), then $s(x^3e(x))=x^2+1$ (or s=[101]) and the correction occurs on the 10th clock period, as shown in Table 3.

Table 3. Correction of errors in systematic (7,4) code.
Code sent is c=[101001 1] and received c*=[101101 1]

CYLCE	CNTL	SQ	c* IN	e	REGISTER MSB LSB	OUT PUT
CLEAR	0	1	1	x	⊕0 0 ⊕ 0	x
shift 1	0	0	0	x	⊕0 1 ⊕ 1	x
2	0	0	1	x	⊕1 1 ⊕ 0	x
3	0	0	0	x	⊕1 0 ⊕ 0	x
4	0	0	0	x	⊕0 0 ⊕ 0	x
5	0	1	1	x	⊕0 0 ⊕ 0	x
6	0	1	1	x	⊕0 1 ⊕ 1	x
7	1	1	0	0	⊕1 0 ⊕ 1	1
8	1	0	0	0	⊕0 0 ⊕ 1	0
9	1	0	0	0	⊕0 1 ⊕ 0	1
10	1	0	0	1	⊕1 0 ⊕ 0	0
11	1	0	0	0	⊕0 0 ⊕ 0	0
12	1	0	0	0	⊕0 0 ⊕ 0	1
13	1	0	0	0	0 0 0	1

The throughput of the decoder of Fig. 5 can be doubled by using another decoder, running with a delay of one code word.

2.2 Otber Block Codes

The (7,4) Hamming code can be modified by adding another overall parity bit to each code word. This (8,4) code [RaFu89] has distance 4 and can be used to correct single random errors and detect double errors (SEC-DED). This code was used in our experiments.

Bose, Chaudhury and Hocquenghem (BCH) class of codes is an extension to Hamming codes, and is described in nearly all the references mentioned in the previous section [e.g., RaFu89]. They were designed to correct not only multiple random errors but also burst errors. A (15,7) BCH code was used in our experiments.

There are many other block codes designed to correct multiple random errors and burst errors, including the Reed-Salomon (RS) codes [MiLe85], Goley codes, and maximal-length codes. The BCH and RS codes are used extensively due to their practical encoders and decoders (e.g., the RS code is used in the audio compact disc). However, since the Hamming codes are simple, they alone or in combination with simple convolutional codes could produce good results in packet radio.

3. CONVOLUTIONAL CODES

3.1 Systematic 1/2 Convolutional Code

Convolutional codes (tree codes or trellis codes) [ViOm79, Kuo81, MiLe85, LeMe88, ClCa81] are codes with memory, i.e., redundant bits are generated from not only the current A bits, but also from a number of past bits by modulo-2 convolutions. The theory of such codes relies heavily on the concepts discussed in the previous section, but space limitation will restrict us to discussing the principles only.

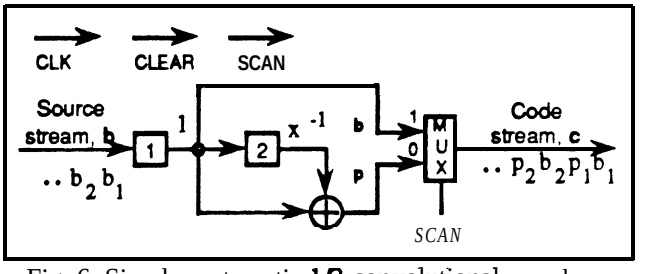

Fig. 6. Simple systematic 1/2 convolutional encoder.

Let us consider a systematic code of rate $r = k/n = 1/2$ (i.e., for each source symbol, b_i, two code symbols $c_{i+1}c_i$ are generated). One of the simplest convolutional encoders is shown in Fig. 6 [MiLe85]. It consists of a two-stage shift register (initially cleared). Following each shift, the commutator (MUX) scans both inputs, thus transferring two channel bits ($n=2$) for each source bit ($k=1$). For example, a source bit stream b = [1010...] produces a code bit stream c = [11 01 11 10 . . .]. Since the source bits appear directly in the coded stream, the code is called systematic. Since this code uses only one past bit, its memory is $m=1$, and the total number of bits, L, that can be affected by a message bit is defined as

$$L = (m+1)\,n \qquad (38)$$

and is called the code constraint length. In the above example, $L=4$ (i.e., b_1 affect p_2). The scheme can be extended to include more delay elements ($m>1$), more adders, and more source bits entering the encoder at a time.

3.2 Self Ortbogonal Codes

A subclass of convolutional codes is the ***canonical self-orthogonal code*** (CSOC) **[ClCa81]** in which **syndrome** symbols can be taken directly as estimates of the source stream, using majority logic. An optimal **1/2 CSOC** of **constraint** length $L=(6+1)\times 2 = 14$ was **designed** for digital radio at AT&T **[Mart85].** The code is optimal in the sense of the **shortest** registers used. An encoder for **the code is shown in Fig. 7.**

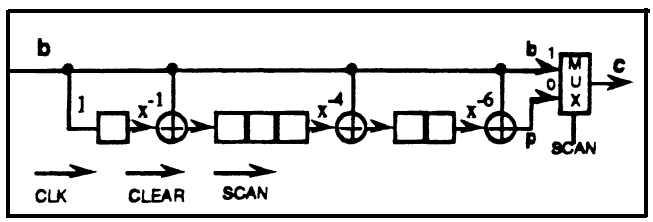

Fig. 7. Optimal systematic **1/2 CSOC** encoder.

A CSOC decoder is shown in Fig. 8. **The** received bits stream is split into message bits **b** and parity bits **p.** The **b** bits are shifted into a buffer register and into a **CSOC** parity encoder for regeneration of the parity bits $\mathbf{p^{**}}$; if the bit from the parity encoder, $\mathbf{p^{**}_i}$, and the currently received parity bit $\mathbf{p^*_i}$ are identical, no error has occurred at that bit position. The syndrome bits are then delayed through a syndrome register, SR, and five of them are passed to the majority ***circuit*** which produces a 1 whenever at **least three** inputs are 1; if $\mathbf{b^*_{i-6}}$ is incorrect and $e=1$, $\mathbf{b^*_{i-6}}$ **is corrected.** If $\mathbf{b^*_{i-6}}$ is **correct** and not more than two errors are allowed in the previous 13 $\mathbf{b^*}$ bits and 4 $\mathbf{p^*}$ bits, then $\mathbf{b^*_{i-6}}$ **will** not be **corrected falsely.** The bit error rate of the definite decoder (without feedback), $\mathbf{BER_{DD}}$, for triple errors at a received BER = $1\ 0^{-3}$ is $\mathbf{BER_{DD}} \approx 3.5\sim 10^{-7}$ **[Mart85].** If the e signal is **fed** back into the SR, then a corresponding adjustment is made to the other syndromes, leading to an improved $\mathbf{BER_{FD}}$. This encoder and decoder was used in the experiments described **next.**

4. EXPERIMENTS

4.1 Block Codes

A systematic **(8,4,4)** Hamming code encoder and decoder was built to test the transmission of ASCII **characters** and speech compressed using linear predictive coding **(LPC-10) [Swan87]** over noisy channel **[ChXu89].** The Hamming circuitry described in the previous section was built using 'ITL components, controlled by a **68000-**based laboratory system with software written in the 68000 assembly language. The important finding from this work was that a **paragraph** of English text transmitted with FEC over a channel with double and mixed errors was read by a group of testers without difficulties. Experiments with **LPC** speech indicate that the above scheme improves the quality of the bits transmitted (23 to **50 %),** but is not sufficient due to the high sensitivity of **the speech parameters to errors.**

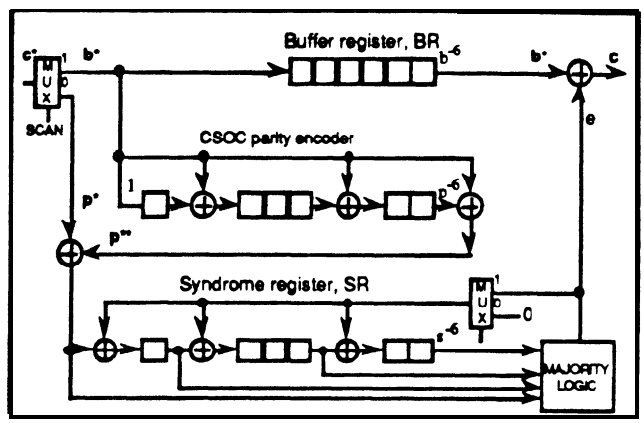

Fig. 8. **Optimal** systematic **1/2** CSOC decoder.

A (15,7) BCH encoder and decoder was also developed to study their complexity **[LaCh90].** The code can correct double random errors and **correct** burst errors of length 4. A **TTL** implementation of the code was controlled by an IBM PC through an interface and software written in Microsoft Quick C and 8086 assembly language.

4.2 Tbe Concatenated/Interlaced Code

The concatenated code **used** in our latest experiment **[KwNa90]** consisted of the **(7,4)** Hamming code, **serving as** an outer code, and the **1/2 CSOC** convolutional code **serving** as an inner code – both described in the previous section. By using an **interlace** matrix capable of holding 8 code words, burst errors were distributed over a range of bit positions so that the outer simple Hamming code could handle them better. A controller **based** on the 6802 microprocessor was used **to handle** interfacing. The encoder, decoder and interlacing circuits were realized in TTL. Most of the software was written in the 6800 assembly language. Performance of the code was **measured using probability** of character error **(PCE)** rather

than the standard BER because it is more meaningful to character transmission. A channel with additive white Gaussian noise (AWGN) was developed to test the system with the definite and feedback CSOC decoders. The PCE was measured for SNR from 3.5 to 8.7 dB, with PCE ranging between 10^{-1} to 10^{-3}. Again, reading of English text presented no problem.

5. CONCLUSIONS

A review of simple forward error correction schemes suitable for moderate correction of random and burst errors has been presented. Block codes alone can improve transmission quality of noncritical text, while convolutional or concatenated codes may be used for more sensitive source bit streams, including compress& speech.

ACKNOWLEDGEMENTS

This work was supported in part by the University of Manitoba. Critical reading of this manuscript by Ken Ferens, Christopher Love and Armain Langi is also acknowledged.

REFERENCES

[Blah87] R.E. Blahut. *Principles and Practice of Information Theory.* Reading (MA): Addison-Wesley, 458 pp., 1987.

[ClCa81] *G.C.* Clark and J.B. *Cain. Digital Communications. New* York (NY): Plenum, 422 pp., 1981.

[ChXu89] *M.* Chu and R. Xu, "Error detection and correction for packet radio," *B.Sc. Thesis*, University of Manitoba, Winnipeg, MB, Canada, 1989.

[Hayk88] S. Haykin. *Error-Correction Coding for Digital Communications. New York (NY):* Wiley, 597 pp., 1988.

[Kuo81] F.F. Kuo (ed.). *Protocols and Techniques for Data Communication Networks.* Englewood Cliffs (NJ): Prentice-Hall, 468 pp., 1981.

[KwNa90] W.Y. Kwong and E.P. Nachareun, "Error detection and correction for packet radio with imperfect *data," B.Sc. Tksis,* University of Manitoba, Winnipeg, MB, Canada, 1990.

[LaCh90] C.F. Lam and S.P. Choo, "Error correction for packet radio," *B.Sc. Thesis*, University of Manitoba, Winnipeg, MB, Canada, 1990.

[LeMe88] E.A. Lee and D.G. Messerschmitt. *Digital Communication. Boston (MA):* Kluwer Academic, 713 pp., 1988.

[Mart85] G.D. Martin, "Optimal convolutional self-orthogonal codes with an application to *digital* radio," *IEEE Intern. Communic. Conf.*, vol. 3, pp. 1249-53, 1985.

[McEl87] RJ. McEliece. *Finite Fieldsfor Computer Scientists and Engineers.* Boston (MA): Kluwer Academic, 207 pp., 1987.

[MiLe85] A.M. Michelson and A.H. Levesque. Error-*Control Techniques for Digital Communication. Nkw* York (NY): Wiley, 463 pp., 1985.

[MiAh88] MJ. Miller and S.V. *Ahamed. Digital Transmission System and Networks.* Rockville (MD): Computer Science Press, vol. I: 274 pp., vol. n: 366 pp., 1988.

[PeWe72] W.W. Peterson and EJ. Weldon, Jr. *Error Correcting Codes.* Cambridge (MA): MIT Press, 1972.

[RaFu89] T.R.N. Rao and E. Fujiwara *Error-Control Coding for Computer Systems.* Englewood Cliffs (NJ): Prentice-Hall, 524 pp., 1989.

[Rode82] M.S. Roden. *Digital and Data Communication Systems.* Englewood Cliffs (NJ): Prentice-Hall, 363 pp., 1982.

[Swan87] C. Swanson, "A study and implementation of real-time linear predictive coding of speech," *M.Sc. Thesis*, University of Manitoba, Winnipeg, MB, Canada, 1987.

[ViOm79] AJ. Viterbi and J.K. Omura. *Principles of Digital Communi*cation *and* Coding. *New* York (NY): McGraw-Hill, 560 pp., 1979.