CODING/DECODING CONCEPTS AND BLOCK CODING

ERROR DETECTION CORRECTION

EDC can be implemented in the five following ways:

Increase signal power

Decrease signal power

Reduce Diversity

Retransmission

Forward Error Correction

- 1. Block Codes
- 2. Convolutional Codes – CONTINIOUS CODES (deal with certain no. Of bits continuosly)

3. Interleaving – mitigating properties for fading channels.

BLOCK CODES

Block codes operate on a block of bits.

Using a preset algorithm.

Add a code to a group of bits (enlarge block).

This block is checked at the receiver.

The receiver validates sequence.



CONVOLUTIONAL CODES

Aka continuous codes - they operate on a certain number of bits continuously.

INTERLEAVING

Interleaving has mitigating properties good for fading channels and works well in conjunction with Block Codes and Convolutional Codes.

EDC APPLICATIONS

All 3 techniques are used together in many EDC suites such as Digital Video, Broadcast, satellite communications, radio and cell phones and baseband systems such as PCs and CD players.

BLOCK CODES
IN DETAIL

Block codes are referred to as (n, k) codes → n bits

Hamming distance concept

Sequence	Integer	Sequence	Integer
0000	0	0001	1
0010	2	0011	3
0100	4	0101	5
0110	6	0111	7
1000	8	1001	9
1010	Unassigned	1011	Unassigned
1100	Unassigned	1101	Unassigned
1110	Unassigned	1111	Unassigned

Codeword space

Hamming Weight: The Hamming weight of this code scheme is the largest number of 1's in a valid codeword. This number is 3 among the 10 codewords we have chosen.

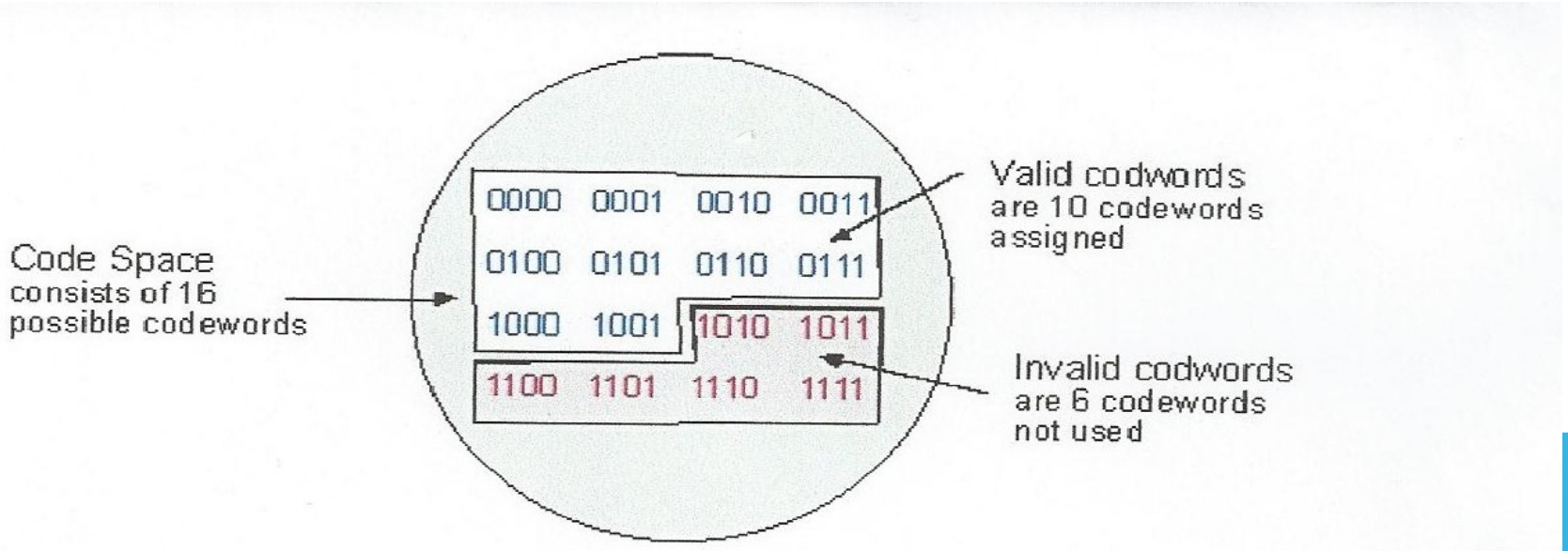


Figure 1 – Codeword space is a list of all words possible form a particular alphabet size such as 4-bit

CONCEPT OF HAMMING DISTANCE

Hamming distance is used to measure distances between two binary words

The Hamming distance between sequences 001 and 101 is = 1; Whereas the

001	0011001
101	1010100
------	---------------
100 1+0+0=1	1001101 (1+1+1+1 = 4)

Hamming distance between sequences 0011001 and 1010100 is = 4

The knowledge of Hamming distance is used to determine the capability of a code to detect and correct errors.

Hamming weight In coding theory, is the number of nonzero digits in a word. ie. in our examples number of 1s in a word. Ie. 1010 = 2

The knowledge of Hamming distance is used to determine the capability of a code to detect and correct errors.

HOW TO CALCULATE HAMMING DISTANCE

Ensure the two strings are of equal length. The Hamming distance can only be calculated between two strings of equal length.
String 1: "1001 0010 1101"
String 2: "1010 0010 0010"

Compare the first two bits in each string. If they are the same, record a "0" for that bit. If they are different, record a "1" for that bit. In this case, the first bit of both strings is "1," so record a "0" for the first bit.

Compare each bit in succession and record either "1" or "0" as appropriate.
String 1: "1001 0010 1101"
String 2: "1010 0010 0010"
Record: "0011 0000 1111"

Add all the ones and zeros in the record together to obtain the Hamming distance.
Hamming distance = 0+0+1+1+0+0+0+0+1+1+1+1 = 6

Table II – Codespace vs. valid codewords, extending the sequence by adding a parity bit

Integer	4 bit Seq.	Extended by 1 bit	Extended again by 1 bit
0	0000	00000	000000
1	0001	00011	000110
2	0010	00101	001010
3	0011	00110	001100
4	0100	01001	010010
5	0101	01010	010100
6	0110	01100	011000
7	0111	01111	011110
8	1001	10010	100100
9	1010	10100	101000
	6 more not used	22 more not used	52 more not used

codespace is equal to 2 to the power of N, When N=4 codespece is 16

We can generalize this to say that the maximum number of error bits that can be detected is

$$t = d_{min} - 1$$

Where d_{min} is Hamming distance of the codewords. For a code with $d_{min} = 3$, we can both detect 1 and 2 bit errors.

NUMBER OF ERRORS WE CAN CORRECT

Detecting is one thing but correcting is another. For a sequence of 6 bits, there are 6 different ways to get a 1 bit error and there are 15 different ways we can get 2-bit errors.

$$\text{No. of 2-bit combinations} = \frac{6!}{2!\ \ 4!} = 15$$

If the transition probability p is small (<<1), the probability of getting three errors is cube of the channel error rate,

$$P_E(N) = p^N$$

The number of errors that we can correct is given by

$$t(\text{int}) = \frac{d_{\min} - 1}{2}$$

For a code with $d_{min} = 3$, the correction capability is just 1 bit.

CREATING BLOCK CODES

- The block codes are specified by (n.k). The code takes k information bits and computes (n-k) parity bits from a code generator matrix.
- Most block codes are systematic in that the information bits remain unchanged with parity bits attached either to the front or to the back of the information sequence.

HAMMING CODE, A SIMPLE LINEAR BLOCK CODE

Table III- 16 possible sequences from a 4 bit word

Seq No.	i0	i1	i2	i3
1 0	0	0	1	
2 0	0	1	0	
3 0	0	1	1	
4 0	1	0	0	
5 0	1	0	1	
6 0	1	1	0	
7 0	1	1	1	
8 0	0	0	0	
9 1	0	0	1	
101	0	1	0	
111	0	1	1	
121	0	0	0	
131	1	0	1	
141	1	1	0	
151	1	1	1	
160	0	0	0	

Hamming codes are most widely used linear block codes. A Hamming code is generally specified as (2^n-1, 2^n-n-1). The size of the block is equal to 2^n-1. The number of information bits in the block is equal to 2^n-n-1 and the number of overhead bits is equal to n. All Hamming codes are able to detect three errors and correct one. Common sizes of Hamming codes are (7, 4), (15,11) and (31, 26). All of these have the same Hamming distance.

CREATING PARITY MATRIX

Table IV – Creating a parity matrix

Sequence			
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Following are just two ways we can order the rows of H, each of these will result in a different code.

$$H = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

A different ordering of

$$H = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{array}\right]$$

Now we take the upper part of this matrix (under the line) and create a new matrix called G

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Now all we have to do is multiply the information vector d with matrix G to get a codeword c.

$c = d \cdot G$

For information sequence [0, 1, 1, 0], we get the following transmitted codeword of length 7.

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

CODE ARCHITECTURE

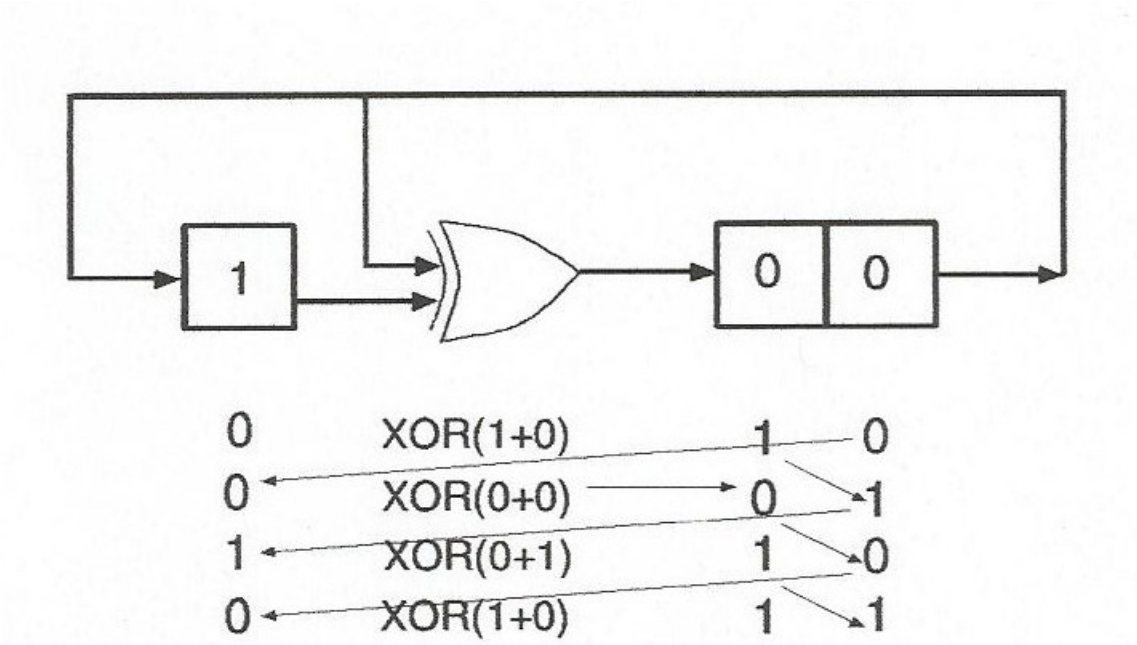
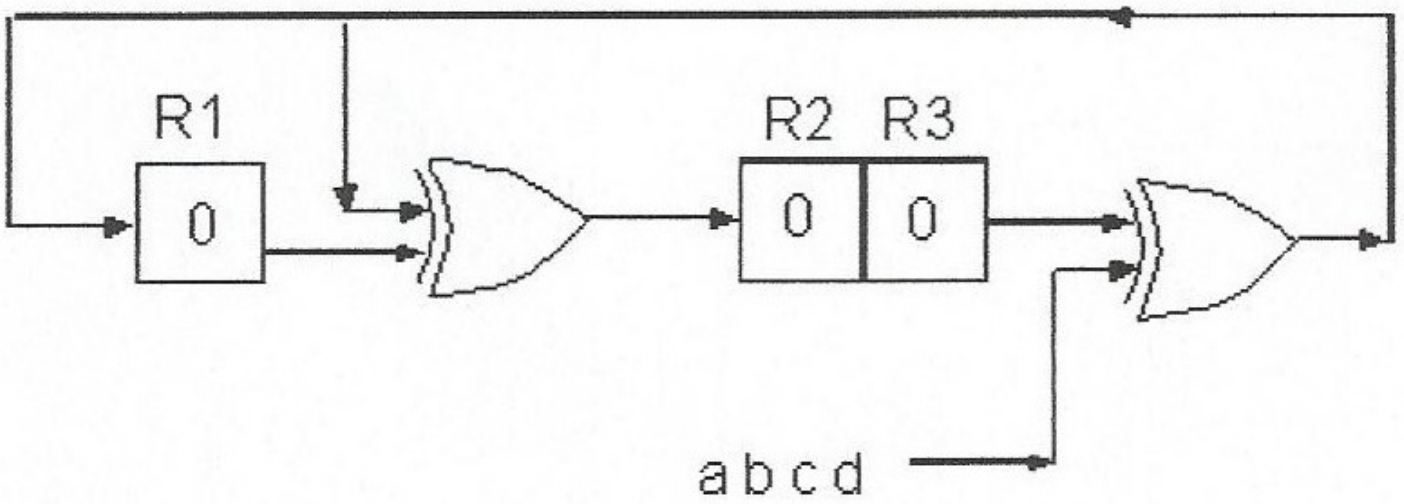
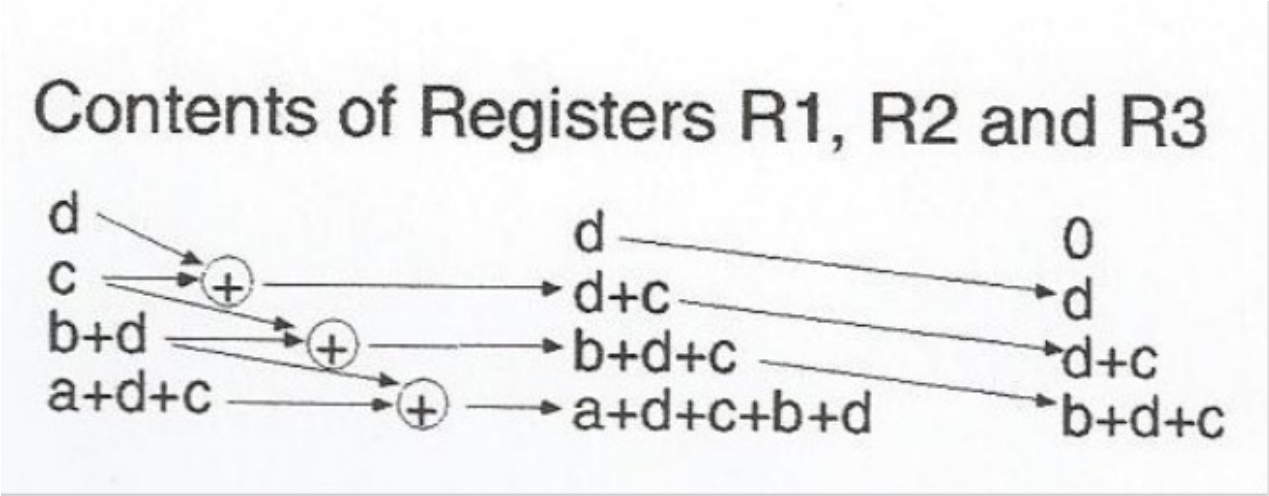


Fig 2 – Using an LSFR to create the H matrix

CODE ARCHITECTURE – CONT. BENJAMIN ARAZI (REF. 1) PROPOSAL



THE ENCODER


Contents of Registers R1, R2 and R3
d
c
b+d
a+d+c
+
+
+
d
d+c
b+d+c
a+d+c+b+d
0
d
d+c
b+d+c

We can also then say that the equation for the parity bits is

$P1 = a + d + c$
$P2 = a + d + c + b + d$
$P3 = b + d + c$

For sequence of 0 1 1 0 , we get

$P1 = 0 + 0 + 1 = 1$
$P2 = 0 + 0 + 1 + 1 + 0 = 0$
$P3 = 1 + 0 + 1 = 0$

THE CODEWORD IS 100011

DECODING

$$s = H^T \cdot c$$

Let's multiply the received code vector [0 1 1 0 1 1 0] with the H^T matrix, to see if we get all zeros since we know that this is a valid codeword.

$$s = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0]$$

$s = [\,0\,0\,0\,]$

The results is indeed [0 0 0], an all zero syndrome.

Let's see what syndrome we get when an error is made. The received vector is now [**1** 1 1 0 1 1 0], the left most bit is received inverted. Now we compute the syndrome again and get

$$s = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

s = 1 1 1

Table V – Syndrome to error location mapping

Error pattern	Syndrome
0 0 0 0 0 0 0	0 0 0
1 0 0 0 0 0 0	1 1 1
0 1 0 0 0 0 0	0 1 1
0 0 1 0 0 0 0	1 0 1
0 0 0 1 0 0 0	1 1 0
0 0 0 0 1 0 0	1 0 0
0 0 0 0 0 1 0	0 1 0
0 0 0 0 0 0 1	0 0 1

$$s = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$s = 1\ 1\ 0$$

REFERENCES

1. **A common sense approach to the theory of error correcting codes, by Benjamin Arazi, MIT press, 1988**
2. **Digital Communications, I. A. Glober and P M Grant, Prentice Hall**
3. **Modulation Detection and Coding, Tommy Oberg**