Decoding Algorithms for Erasure-Correcting Array Codes

Rodica Stoian and Dan Alexandru Stoichescu
Faculty of Electronics and Telecommunications
Department of Applied Electronics and Information Engineering
"Politehnica" University of Bucharest
1-3 Armata Poporului Blvd.,
Bucharest
ROMANIA
E-mail: stoich@vl.elia.pub.ro

Abstract: A new class of error correcting codes, the array codes, shows large usefulness in improving the reliability of two-dimensional information. This class of codes could correct multiple burst errors and erasures represented by erroneous or erased columns in a code array. The paper presents an efficient decoding algorithm for erasures which affect two columns. It also shows a decoding that can be easily implemented based on an exclusive OR operation over GF (2).

Dr.Rodica Stoian got her Engineering Diploma in Electronics and Telecommunications in 1966 and her Doctoral Degree in 1984, both from the Polytechnical Institute of Bucharest. Currently she is Associate Professor at the Department of Applied Electronics and Information Engineering, "Politehnica" University of Bucharest. Her present main interests are coding theory, data compression and statistical process control.

Dr.Dan Alexandru Stoichescu got his Engineering Diploma in Electronics and Telecommunications and Doctoral Degree from the Polytechnical Institute of Bucharest, in 1965 and 1981 respectively. Currently he is Associate Professor at the Department of Applied Electronics and Information Engineering, "Politehnica" University of Bucharest. His present main interests are information transmission theory, control systems, and neural networks.

1. Introduction

The common error correction codes for digital recording systems, such as magnetic and optical disk drivers, use redundant storage and are based on Reed Solomon (RS) code. The RS code performs quite well and can be used for error correction and also for recovering the information lost in redundant storage area. The RS code involves operation over finite fields, e.g. GF(2^8), and is characterized by a cubic complexity [1].

In the last decade use of error correction in digital recording systems, array codes, based on exclusive OR operations over GF(2), which offer the advantages of a reduced complexity – quadratic, has been largely considered. The array codes of block type can be described as two -dimensional block codes which use for construction two types of check parity conditions, combined with diagonal read-out rule. This class of code can correct multiple burst and also presents the advantages of reduced complexity, fast encoding and decoding algorithms with simple implementation (the operation over extended finite fields is this time avoided).

2. Problem Statement

Consider the information stored on m disks and the redundancy on l disks. On each disk the information symbols are represented by disk sectors, bytes or bits. For simplicity we will consider the symbols as bits, i.e. elements from GF(2). Consequently, the $(m+l)$ disks can be represented as a column in a matrix **A** with $m+l$ columns and elements $a_{i,j} \in$GF(2). For easier notation in the encoder procedure we consider that each column contains a number of $(m-1)$ binary symbols and the number of redundant columns (disks) $l=2$.

a) Allowed codeword size

Linear correction codes can reach the Singleton bound [2] which states that to correct t errors a code A(n_2, k_2,d) must have at least $2t$ parity symbols

$2t \leq n_2 - k_2$

For a single burst error correcting array code A(n_1 x n_2 ,k,d) the codeword *size* n_1 x n_2 must satisfy the condition that for any n_1 the allowed values of n_2 include all prime numbers above n_1 [3], i.e. $n_2 \geq n_1 + 1$.

If we take $n_2 = n_1+2$, and consider a symbol as a group of n_1 bits, the redundancy of the array code is

$\rho = n_1 \times n_2 - k$:

$\rho = n_1 \cdot n_2 - n_1 \cdot (n_2 - 2) = n_1 \cdot (n_1 + 2) - n_1 \cdot n_1 = 2n_1$

A burst of dimension n_1 represents a column of bits for the array code A(n_1 x n_2 ,k,d) and a symbol for the code A(n_2, k_2,d). If A(n_2, k_2,d) can correct one error, t=1, the array code A(n_1 x n_2, k, d) can correct a single burst error represented by a column with n_1 symbols, and satisfy the Singleton bound. An erasure is an error with a known location. According to the theory of linear codes [2] two columns for

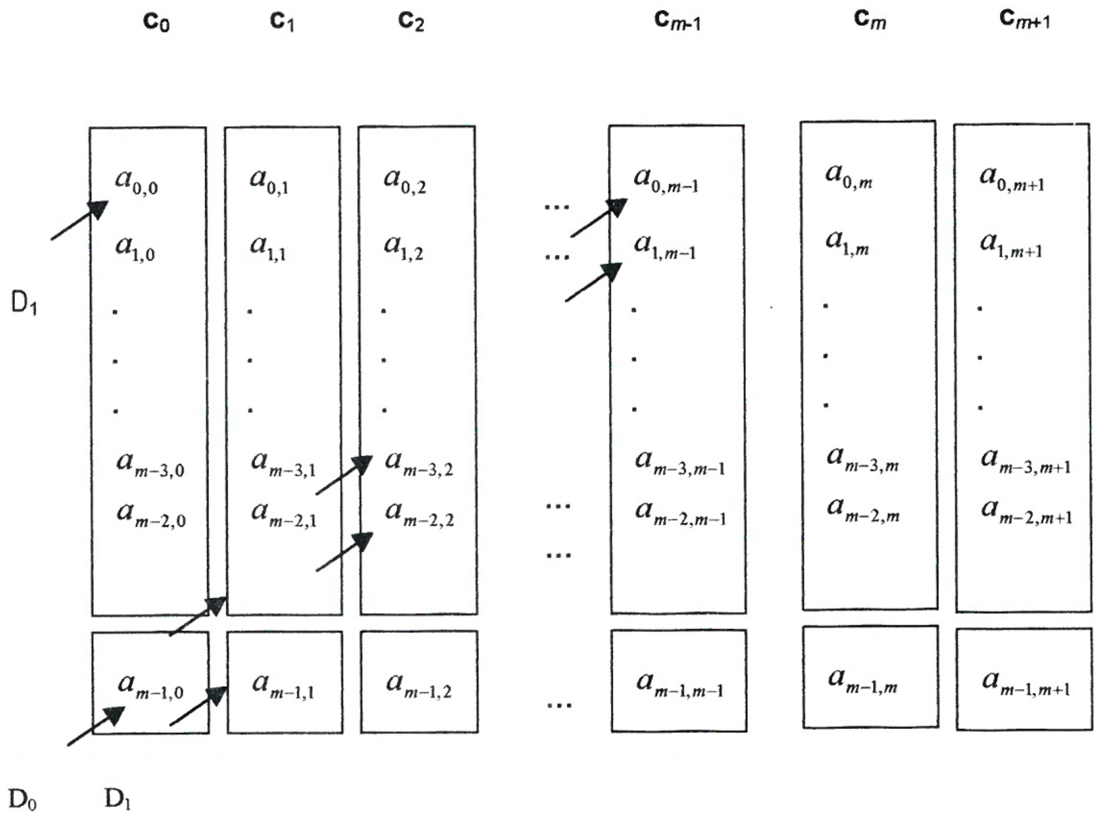

Figure 1. Array Codeword for Array Code A(m x (m+2), s, d)

parity check symbols let the array code A(n_1 x n_2, k, d) have the capability to correct any single column in error or to correct any pair of erased columns. Consequently one allowed dimension of an array codeword A(n_1 x n_2) is n_1 =m and n_2 =m+2.

Such a structure can be represented as a binary array **A**=($a_{i,j}$)$_\Omega$, $a_{i,j} \in$GF(2) in which the (m-1) row r_{m-1} is added fictitiously and contains only zero-symbols. The binary array code A(m x (m+2),s,d) where $\Omega = \{(i,j) \mid 0 \leq i \leq m-1;\ 0 \leq j \leq m+1\}$, consists of all the codewords represented by arrays **A** satisfying certain parity conditions and diagonal read-out rule.

b) Diagonal read-out rule

Gilbert codes and their generalization are the first array codes (they were developed in 1969) to have allowed burst error correction based on diagonal (helical) readout order of elements of the matrices which represent the codewords.

If the diagonal read-out is used to transmit the array codeword as a vector codeword, the array code can be used to correct one or multiple diagonal burst errors [3]. If the diagonal readout rule is used to compute one of the check parity equations, the array codes can correct bursts representing column errors.

Diagonal (helical) read-out order of elements of the matrices which represent the codeword can be used for burst errors correction.

The read-out s of the code is formally defined as follows: the first diagonal (named main diagonal) starting with element $a_{m-1,0}$ is D_0: $a_{m-1,0}\ a_{m-2,1}, \dots\ a_{0,m-1}$. For the skipping value s=1, the second diagonal D_1 starts in $a_{m-1,1}$ and is: D_1: $a_{m-1,1}\ a_{m-2,2}, \dots\ a_{1,m-1},\ a_{0,0}$ and so on.

c) Burst correcting capability

The burst correcting capability (BCC) of the array code is a function dependent on the codeword dimension m x (m+2) and on the particular diagonal read-out used. For an array code A(m x (m+2), s, d) the BCC = b satisfies the inequality:

$$b \le m-1 \quad \text{if } s = 1 \text{ and}$$

$$b \le m, \quad \text{if } 1 < s < m+2 \tag{1}$$

with the restriction that s and m+2 are relatively prime, i.e. gcd (s , m+2) = 1 [4].

For s = 1, this implies that m is necessarily a prime number. This assumption does not contradict the reality because, from a practical point of view, the addition of an arbitrary number of disks with no information may be considered, so that m, total number of disks (i.e. total number of information columns), fulfils the condition to be a prime number.

From (1) results that an array code A(m x (m+2), 1, 3) with parameter N= $m\cdot(m+2)$ redundancy $\rho = 2(m-1)$, minimum distance d = 3 and s = 1 can correct a burst of dimension $b \le m-1$ which means a column in matrix **A** or equivalently, a disk fails. If the errors affect two columns with known location, which represent two erased columns, the code has the capability to correct them with no restriction that the erased columns (disks) carry information or parity symbols [5].

3. Encoding Algorithm

The array code A(m x (m+2), 1, d) over GF(2) is represented by arrays $\mathbf{A}=(a_{i,j})_\Omega$, where $\Omega = \{(i,j) \mid 0 \le i \le m-1; 0 \le j \le m+1\}$, with binary symbols $a_{i,j} \in GF(2)$, $0 \le i \le m-2$ and $a_{m-1,j} = 0$ for all $0 \le j \le m+1$, satisfying the following two parity equations:

horizontal parity:

(which allows the computation)

$$a_{i,m} = \sum_{t=0}^{m-1} a_{i,t} \tag{2}$$

and diagonal parity:

$$a_{i,m+1} = \mathrm{D}_0 + \mathrm{D}_i \tag{3}$$

where $\mathrm{D}_0 = \sum_{t=1}^{m-1} a_{m-1-t,t}$, $\mathrm{D}_i = \sum_{t=0}^{m-1} a_{<i-t>_m, t}$

and $< l >_m = k$, represents the unique integer k, $0 \le k \le m$-1, $k = l \pmod{m}$. According to Eq (2), the horizontal redundancy contained in column $\mathbf{c}_m$ is computed based on a horizontal parity, which is always even. For the diagonal redundancy contained in column $\mathbf{c}_{m+1}$, according to Eq (3), the parity may be either even or odd depending on particular parity of the main diagonal D_0.

4. Decoding Algorithm

Provided array $\mathbf{A}_{m \times (m+2)}$ represents the correct codeword, the recorded codeword is denoted by **R=A+E**, $\mathbf{R}=(r_{i,j})_\Omega$, $r_{i,j} \in$GF(2), where $\mathbf{E}=(e_{i,j})_\Omega$ is the error array with elements $e_{i,j} \in$GF(2) and an addition sign denoting matrix addition to GF(2).

Suppose that in the recording process the codeword has been erroneously used and two columns $\mathbf{c}_u$ and $\mathbf{c}_v$ (two disks fails), $0 \le u,\ v \le m+2$ have been erased. The locations of erased columns may determine three situations:

a. errors have affected the redundant information contained in $\mathbf{c}_m$ and $\mathbf{c}_{m+1}$

b. one redundant column $\mathbf{c}_m$ (or $\mathbf{c}_{m+1}$) and a column corresponding to the information symbols, $\mathbf{c}_j$, $0 \le j \le m$ - 2 are erroneous

c. errors appear in two information columns.

For the situations (a) and (b), the decoding algorithm which allows the recovery of the erased columns, is based on the encoding algorithm represented by Eqs (2) and (3). Starting with a known situation (in a matrix **A** all the symbols on the row r_{m-1} are zero) a reconstruction of the columns corresponding to redundant information using a horizontal parity equation for $\mathbf{c}_m$, respectively, a diagonal parity equation for $\mathbf{c}_{m+1}$, takes place .

If the pair of erased columns contains $\mathbf{c}_m$ (or $\mathbf{c}_{m+1}$) and one column $\mathbf{c}_j$, $0 \le j \le m$-2, first a recovery of the column corresponding to redundant information $\mathbf{c}_m$ using a horizontal parity equation and separately the column $\mathbf{c}_j$ using a diagonal parity equation, takes place.

If the two erased columns belong to the information area, case (c), according to the encoding Eqs (2) and (3), separate use of the parities is excluded because computing the main diagonal D_0 by the encoding equation (3) is not possible. The solution for this decoding situation is presented in [4] based on the recursive technique of split syndromes.

Based on $\mathbf{c}_m$ and $\mathbf{c}_{m+1}$ known symbols, the main diagonal D_0 can be computed using Eq (4):

$$\mathrm{D}_0 = \sum_{l=0}^{m-1} a_{l,m} + \sum_{l=0}^{m-1} a_{l,m+1}$$

$$\mathrm{D}_0 = \mathbf{c}_m + \mathbf{c}_{m+1} \tag{4}$$

If we refer to each element in column $\mathbf{c}_m$ and $\mathbf{c}_{m+1}$ which represents the syndrome vectors z_H and z_D as syndrome components $z_{k,\mathrm{H}}=z_{k,m+1}$ $0 \le k \le m$-1 and $z_{l,\mathrm{D}}=z_{l,m+2}$ $0 \le l \le m$-1 it is possible that starting with the known symbols $r_{m-1,j}$, $0 \le j \le m$-2 the syndrome components are computed one by one.

Figure 2 gives an algorithm based on Eqs (2), (3) and (4), which describes a two-erasure correcting decoder for array code **A**(m x (m+2), $1,\ 3$) .

Algorithm A covers all possible situations related with locations of two erased columns.

```
read    $R = (r_{i,j})_{\substack{0 \le i \le m-2 \\ 0 \le j \le m+1}}$,    read u, v
for $j$=0 to $m+1$ do
  begin $r_{m-1,j} = 0$   end
if u $\le$ m and v = m+1
  then   begin
            for $i = 0$ to $m$-1 do
              begin $r_{i,u} = \sum_{t=0}^{m} r_{i,t}$   end
              $D_0 = \sum_{t=1}^{m-1} r_{m-1-t,t}$
              for $i = 0$ to $m$-1 do
        begin $r_{i,m+1} = D_0 + \sum_{t=0}^{m-1} r_{\langle i-t \rangle_m, t}$
        end
    end
else begin
     if u < m and (v = m or v = m+1)
       then begin
         $D_0 = r_{\langle u-1 \rangle_m, m+1} + \sum_{l=0}^{m-1} r_{\langle u-l-1 \rangle_m, l}$
            for i = 0 to m-1 do
            begin
     $r_{\langle i-u \rangle_m, u} = D_0 + r_{i,m+1} + \sum_{\substack{t=0 \\ t \ne u}}^{m-1} r_{\langle i-t \rangle_m, t}$
            end
            for i = 0 to m-1 do
            begin  $r_{i,m} = \sum_{t=0}^{m-1} r_{i,t}$
            end
          end
     else begin
         $D_0 = \sum_{t=0}^{m} r_{t,m} + t_{t,m+1}$
         for i = 0 to m-1 do
         begin
         $h_i = \sum_{\substack{t=0 \\ t \ne u,v}}^{m} r_{i,t}$
         $d_i = D_0 + \sum_{\substack{l=0 \\ l \ne u,v}}^{m-1} r_{\langle i-l \rangle_m, l}$
          end
          s = -1
          while  (s$\ne m$-1)  do
          begin
```

```
                $s \leftarrow \langle s-(v-u)\rangle_m$
                $r_{s,v} \leftarrow d_{\langle v+s\rangle_m} + r_{\langle s+(v-u)\rangle_m,u}$
                $r_{s,u} \leftarrow h_s + r_{s,v}$
            end
        end
    end
```

Figure 2. Decoding Algorithm A

5. Conclusion

Appendix is an application which illustrates encoding according to Eqs (2) and (3), and decoding following the algorithm A steps as presented in Figure 2.

The results prove the simplicity of array codes implementations over GF(2).

REFERENCES

1. MACWILLIAMS, F. J. and SLOANE, N. J. A., **The Theory of Error Correcting Codes,** NORTH-HOLLAND, Amsterdam, 1983.
2. VAN LINT, J. H. , **Introduction to Coding Theory**.
3. GOODMAN, R., MCELIECE, R. J. and SAYANO, M., **Phased Burst Error Correcting Codes**, IEEE TRANSACTIONS ON INFORMATION THEORY, IT-39, 1993, pp. 684-693.
4. BLAUM, M., FARRELL, P.G. and VAN TILBORG, H. C. A., **A Class of Burst Error-correcting Array Codes**, IEEE TRANSACTIONS ON INFORMATION THEORY, IT-32, 1986, pp. 836-839.
5. BLAUM, M. and ROTH, R. M., **New Array Codes for Multiple Phased Burst Correction**, IEEE TRANSACTIONS ON INFORMATION THEORY, IT-39, 1993, pp. 66-77.

Appendix

Let be m=5 and $\{a_{i,j}\}$, $0\ i \le 4$, $0 \le j \le 6$

$$\{a_{i,j}\} = \begin{vmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

columns $0 \le j \le 4$ contain information , columns 5 and 6 contain parity symbols and row 4 is imaginary and contains zero in all the co-ordinates.

Encoding: column 5 contained a horizontal parity and column 6 contained a diagonal parity, was computed with:

$$a_{i,m} = \sum_{t=0}^{m-1} a_{i,t} \tag{1}$$

$$a_{i,m+1} = D_0 + \sum_{t=0}^{m-1} a_{\langle i-t \rangle m,t} \tag{2}$$

where $D_0 = \sum_{t=1}^{m-1} a_{m-1-t,t}$

Horizontal parities and diagonal parity are:

$$a_{0,5} = 0, a_{1,5} = 1, a_{2,5} = 0, a_{3,5} = 1, a_{4,5} = 0$$

$$D_0 = a_{3,1} + a_{2,2} + a_{1,3} + a_{0,4} = 1$$

$$a_{0,6} = 1, a_{1,6} = 1, a_{2,6} = 0, a_{3,6} = 1, a_{4,6} = 0$$

Decoding

Let assume that received (readed) matrix is erroneous and contains two columns erased u=0 and v=2. It is:

$$\{a_{i,j}\} = \begin{vmatrix} ? & 0 & ? & 0 & 1 & 0 & 1 \\ ? & 0 & ? & 0 & 1 & 1 & 1 \\ ? & 0 & ? & 0 & 1 & 0 & 0 \\ ? & 1 & ? & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

- **step 1**

Compute new value of D_0:

$$D_0 = \sum_{t=0}^{m} (a_{t,m} + a_{t,m+1}) = 1$$

- **step 2**

Compute horizontal syndrome:

$$h_i = \sum_{\substack{t=0 \\ t \neq u,v}}^{m} a_{i,t}$$

$\underline{h}$=[1 0 1 1 0]

- **step 3**

Compute diagonal syndrome:

$$d_i = D_0 + a_{i,m+1} + \sum_{\substack{l=0 \\ l \neq u,v}}^{m-1} a_{\langle i-l \rangle m,l}$$

$\underline{d}$=[1 0 1 0 1]

- **step 4**

$$s \leftarrow \langle -(v-u) - 1 \rangle_m$$

$$a_{s,v} \leftarrow d_{\langle v+s \rangle m} + a_{\langle s+(v-u) \rangle m,u}$$

$$a_{s,u} \leftarrow h_s + a_{s,v}$$

$s \leftarrow \langle -(2-0)-1 \rangle_5 = 2$

$a_{2,2} \leftarrow d_{\langle 2+2 \rangle 5} + a_{\langle 2+(2-0) \rangle 5,0}$

$a_{2,0} \leftarrow h_2 + a_{2,2}$

$a_{2,2} = d_4 + a_{4,0} = 1$

$a_{2,0} = h_2 + a_{2,2} = 0$

- **step 5**

$s \leftarrow \langle s-(v-u) \rangle_m$

$a_{s,v} \leftarrow d_{\langle v+s \rangle m} + a_{\langle s+(v-u) \rangle m,u}$

$a_{s,u} \leftarrow h_s + a_{s,v}$

$s \leftarrow \langle 2-(2-0) \rangle_5 = 0$

$a_{0,2} \leftarrow d_{\langle 2+0 \rangle 5} + a_{\langle 0+(2-0) \rangle 5,0}$

$a_{0,0} \leftarrow h_0 + a_{0,2}$

$a_{0,2} = d_2 + a_{2,0} = 1$

$a_{0,0} = h_0 + a_{0,2} = 0$

- **step 6**

$s \leftarrow \langle s-(v-u) \rangle_m$

$a_{s,v} \leftarrow d_{\langle v+s \rangle m} + a_{\langle s+(v-u) \rangle m,u}$

$a_{s,u} \leftarrow h_s + a_{s,v}$

$s \leftarrow \langle 0-(2-0) \rangle_5 = 3$

$a_{3,2} \leftarrow d_{\langle 2+3 \rangle 5} + a_{\langle 3+(2-0) \rangle 5,0}$

$a_{3,0} \leftarrow h_3 + a_{3,2}$

$a_{3,2} = d_0 + a_{0,0} = 1$

$a_{3,0} = h_3 + a_{3,2} = 0$

- **step 7**

$s \leftarrow \langle s-(v-u) \rangle_m$

$a_{s,v} \leftarrow d_{\langle v+s \rangle m} + a_{\langle s+(v-u) \rangle m,u}$

$a_{s,u} \leftarrow h_s + a_{s,v}$

$s \leftarrow \langle 3-(2-0) \rangle_5 = 1$

$a_{1,2} \leftarrow d_{\langle 2+1 \rangle 5} + a_{\langle 1+(2-0) \rangle 5,0}$

$a_{1,0} \leftarrow h_1 + a_{1,2}$

$a_{1,2} = d_3 + a_{3,0} = 0$

$a_{1,0} = h_1 + a_{1,2} = 0$

- **step 8**

$s \leftarrow \langle s-(v-u) \rangle_m$

$s \leftarrow \langle 1-(2-0) \rangle_5 = 4 = m-1$

END