CODING AND DECODING OF THE MATROID CODES

Ghenadie Bodean
e-mail: gbodean@mail.md
Technical University of Moldova, Faculty of Radioelectronics and Telecommunications, Department of Computer Design, MD2012, St. cel Mare,168, Kishinau, R.Moldova

Key words: Burst errors, matroid code, finite field operations

ABSTRACT
Encoder and decoder algorithm of the matroid burst errors correcting code is defined. Constant multiplications is considered, and are shown the circuits implemented in the XOR-basis that drastically reduces the encoder/decoder complexity. The features of the matroid code encoding/decoding are analysed.

I. INTRODUCTION

It is known [1] that the upper bound of error correcting capabilities of the burst errors correcting code block is equal (n-k)/2, where n is the codeword length and k is the number of information (original) symbols in the codeword. Recently in [2] has been proposed a new class of errors correcting code (ECC), named *matroid* ECC, whose efficiency is impressive and is equal to k. But unlike the classical encoding/decoding technology the matroid codes (or M-code) don't perform a correction in the sense of this word. In essence matroid coding performs a restoration of the transmitted information, i.e. data.

Take an example. Begin from the notion "matroid" that has been proposed by Hassler Whitney [3]. H.Whitney used this notion to generalize the property of linear independence in vector space. Code words of ECC also can be interpreted as the vector's coordinates. A restricted number of vectors that are linear independent, i.e. form a *basis* in the given vector spaces, can be found. For example, in the vector space of rank 2 over Galois field **GF**(2) can be distinguished the following basis:

$$B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}. \qquad (1)$$

Columns of the matrix $\boldsymbol{B}$ in (1) are the coordinates of vectors of the length r, r= 2.

The matrix

$$A_{r\times n} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad (2)$$

contains a so-called collection of sub matrix $\boldsymbol{B}$ from (1). It's said that matrix $\boldsymbol{A}$ represents the matroid M (or matroid M is represented by $\boldsymbol{A}$) and this is denoted by $M[\boldsymbol{A}]$.
Moreover, by definition the matroid of matrix (2) is a *uniform matroid* [4].

Matrix $\boldsymbol{A}$ can be used to generate the code words (vectors) of M-code:

$$\mathbf{v} = \mathbf{x}\cdot\boldsymbol{A}, \qquad (3)$$

where $\mathbf{x}= <x_1,\ldots,x_r>$ is a vector of original information symbols; $\mathbf{v}=<v_1, \ldots, v_n>$ is an output (transmitted) vector.

The right side of (3) represents a system of linear equations with *additive* and *multiplicative* operations over field **GF**(2); such subsystem of rank r= 2 represented by one of the matrix (1) is a system of *linear independent* equations.

The last remark has a very substantial significance! Coding is performed to correct the erroneous symbols. And from any of linear independent subsystems of $\boldsymbol{A}$ always can be restored the original symbols of $\mathbf{x}$. in particular, the matroid $M[\boldsymbol{A}]$, represented by matrix (2), is "capable" to correct one error.

So from (3) results:

$$\mathbf{v}=<v_1,v_2,v_3>=\mathbf{x}\times\boldsymbol{A}= <x_1,x_2>\times\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}=<x_1, x_2, x_1+ x_2>,$$

where $x_1\rightarrow v_1$, $x_2\rightarrow v_2$ and $x_1+x_2\rightarrow v_1$.

If any symbol of $\mathbf{v}$ was distorted in the transmission time, then the other remaining two symbols of $\mathbf{v}$ will be "help" to restore the failed components of the original vector $\mathbf{x}$. There are three cases (see Table 1) of restoration.

Certainly that given example is very trivial to imagine the full "beauty" of restoration.

Table I

Restorations of the original vector of the matroid code over **GF**(2)

Distorted symbol	Equations of restorations
v_1	$x_2 \leftarrow v_2$ and $x_1 \leftarrow x_2 + v_3$
v_2	$x_1 \leftarrow v_1$ and $x_2 \leftarrow x_1 + v_3$
v_3	$x_1 \leftarrow v_1$ and $x_2 \leftarrow v_2$

The cardinality of matrix ***A*** is insufficient to increase the correction capability (efficiency) of the analysed M-code! That is why it's needed to pass to the vector spaces of a greater dimension. On such a transition inevitably appears the necessity to extend the basis, i.e. to pass to an extended Galois field $\mathbf{GF}^r(2^m)$ of degree of *r* and characteristics 2^m. Degree *m* defines the number of bits (binarity) per codeword symbol.

Paper [5] introduces to theoretical fundamentals of the matroid code construction. In this work will be analysed the particularities of M-code encoding and decoding.

II. ENCODER OF THE MATROID CODE

Function of the error-correcting matroid code is determined by transformation (3): on the enter is the source codeword **x** of length *r* with *m*-binary symbols; on the exit is the codeword **v** with the twice length, i.e. equal to 2*r*, with also *m*-binary symbols; (en)coder performs the matrix multiplication of **x** on **A** over field $\mathbf{GF}^r(2^m)$, where matrix **A** represents a corresponding uniform matroid. Consider an example.

Example 1. Let *r*= 3 and *m*= 3. For predefined *m* select as field generator the polynomial $p(x)=1+x+x^3$ over **GF**(2)

Table 2

Multiplication *mod* $(1+x+x^3)$ over $\mathbf{GF}(2^3)$

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

and for code generator the polynomial $g(x)=1+3x+3x^3$ over $\mathbf{GF}^3(2^3)$. The results of multiplication *modulo* *p*(*x*) over decimal representation of the elements of $\mathbf{GF}^3(2^3)$ are presented in Table 2. It's not hard to observe that Table 2 is a Latin square.

Remind that the decimal numbers in Table 2 represent the rest from division *mod* *p*(*x*), i.e. 0 is 0, 1 is 1, 2 is *x*, 3 is *x*+1, 4 is x^2, etc., 7 is x^2+x+1. Also, the additive operations on the binary representation of the corresponding decimal numbers are performed as bit-by-bit (bit-wise) XOR.

A few matroids were found in the field $\mathbf{GF}^3(2^3)$ by a specially elaborated program. Among the found matroids was selected the uniform matroid that is represented by the following matrix:

$$\mathbf{A} = \begin{bmatrix} 7 & 4 & 7 & 7 & 2 & 5 \\ 5 & 1 & 2 & 1 & 1 & 5 \\ 4 & 6 & 2 & 4 & 6 & 2 \end{bmatrix}.$$

Matrix **A** is a constructive basis to design the matroid code encoder (or M-coder) over $\mathbf{GF}^3(2^3)$.

The functioning of M-coder is given by

$$\mathbf{v} = \mathbf{x} \cdot \mathbf{A} = < x_1, x_2, x_3 > \begin{bmatrix} 7 & 4 & 7 & 7 & 2 & 5 \\ 5 & 1 & 2 & 1 & 1 & 5 \\ 4 & 6 & 2 & 4 & 6 & 2 \end{bmatrix}$$

or

$$\begin{cases} 7x_1 + 5x_2 + 4x_3 = v_1, \\ 4x_1 + x_2 + 6x_3 = v_2, \\ 7x_1 + 2x_2 + 2x_3 = v_3, \\ 7x_1 + x_2 + 4x_3 = v_4, \\ 2x_1 + x_2 + 6x_3 = v_5, \\ 5x_1 + 5x_2 + 2x_3 = v_6. \end{cases} \qquad (4)$$

Figure 1 shows the scheme of the corresponding M-coder. Edges in Figure 1 are marked by multiplier; ⊕ is a symbol of the specified add operation. So components x_1,x_2 and x_3 of the source code **x** are multiplied by the corresponding factors (multiplier) and then are summed modulo *p*(*z*); on the exit of M-coder are the components of vector **v**.

All the operations in Figure 1 are performed asynchronously. A look-up table based multiplier circuit can be used to make a (polynomial) multiplication of *a* by *b* over $\mathbf{GF}^3(2^3)$ [6]. But in the case of one of the fixed inputs is more preferable to use a scheme of the *constant multiplier*. The expected gain is the drastically reduction of the hardware complexity.

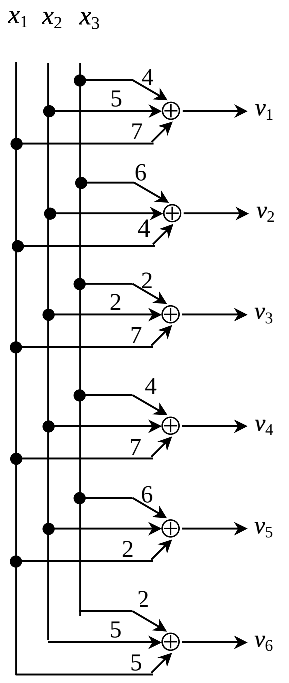


Figure 1. Block diagram of M-coder with $g(x)$=1+3x+3x^3 over $\mathbf{GF}^3(2^3)$ and $p(x)$=1+x+x^3 over **GF**(2)

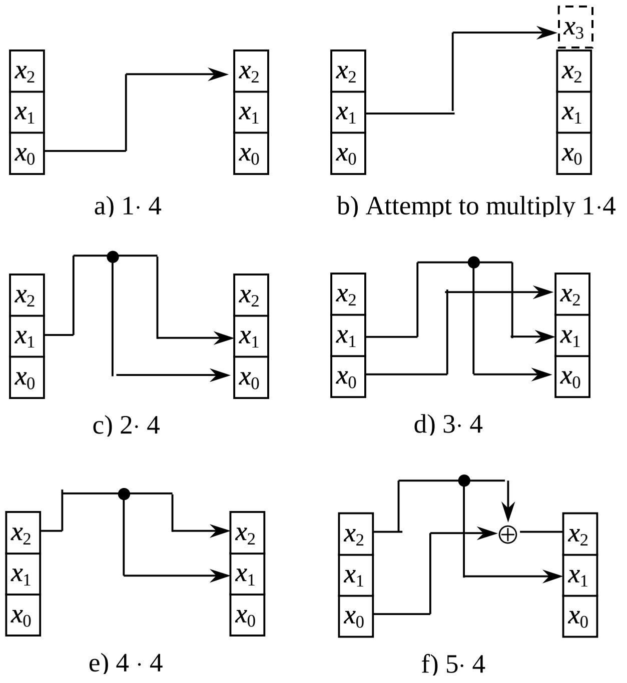


Figure 2. Step by step designing of the diagram for multiplication **x**· 4

III. XOR-BASIS CONSTANT MULTIPLIER IMPLEMENTATION

A parallel architecture of the multiplier can be derived from the standard, i.e. "school column", multiplication. In this case the total number of gates for the bit parallel multiplier is m^2 AND gates and $m(n$-1) XOR gates [7].

A technique to implement the constant multiplier, only in the basis XOR gates, can be proposed. Consider an example.

Example 2: multiplication ***x*** by a constant ***c*** over **GF** (2^3) with $p(x)$= 1+x+x^3, where ***c***=2, .., 7. Let ***x*** is multiplied by 4. Binary format of polynomial ***x*** is <x_2, x_1, x_0>. If ***x***=1 then the result is 1·4=4. This transformation can be represented by a diagram, shown in Figure 2, a. In the same way, it can be constructed a diagram for multiplication 2·4 (see Figure 2, b). But in this case the arrow will indicate to an inexistent position x_3. This inadmissible "situation" must be changed taking into account the vector-corrector for x^3, i.e. x^3= 1+x from equality $p(x)$=0. So the "right" diagram for multiplication 2·4 will be such as is shown in Figure 2, c.

Multiplication 3·4, shown in Figure 2,d, will be obtained by combining diagrams a) and c) of the Figure 2. In Figure 2, d is shown a correct diagram for multiplication 4·4; it was taken into account the vector-corrector x^4=x+x^2. Two edges will be coincide under synthesizing diagram for multiplication 5·4 from diagrams a) and e). The coinciding edges will be replaced by an XOR operation (see ⊕ in Figure 2, f). Finally, by combination of diagrams for values of ***x*** from 1 to 7 results the diagram, shown in Figure 3, c.

A matrix can represent the constant multiplier structure. Let **T**= [t_{ij}] be a matrix m×m, where

$$t_{ij} = \begin{cases} 1, \text{ } if \text{ there is an edge } i \to j; \\ \quad 0, \text{ otherwise.} \end{cases}$$

Diagram of multiplication x· 4 has a following matrix:

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

By introducing matrix **T** an algorithm for constant multiplier designing is derived (see Figure 3).

As a result of the **ConstantMutiplier** algorithm processing will generate the matrix **T** that "reflects" the constant multiplier structure in the XOR-gates basis.

Resulted circuits constant multipliers for **GF** (2^3) with $p(x)$= 1+x+x^3 are presented in Figure 4. It's easy to verify that all circuits given in Figure 4 are in accordance with the corresponding multiplication presented in decimal format in the Table 2.

So, to implement the multipliers coder, shown in Figure 1, by bit parallel architecture will be needed 18·(3^2+3(6-1))= 432 equivalent gates and by constant multipliers shown in Figure 4, will be need 27 equivalent gates.

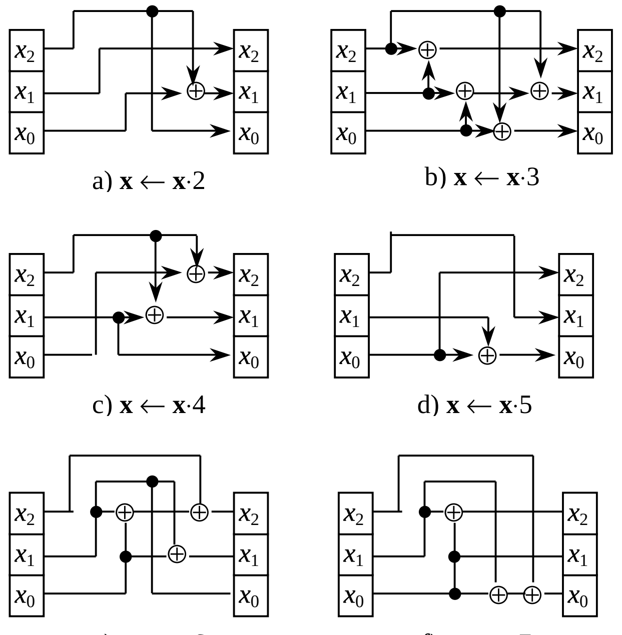
```
ConstantMultiplier ($C$, $p(x)$)
1   for $i \leftarrow m$ to $2(m-1)$ do
2       Make vector-correctors $x^i$ {$\mathbf{x}=<x_1,\ldots, x_m>$}
3   for $i \leftarrow 1$ to $m$ do
4       for $j \leftarrow 1$ to $m$ do
5           $a_{ij} \leftarrow 0$ {initialize matrix $\mathbf{T}=[t_{ij}]_{m\times m}$}
6   $t_1 \leftarrow t_1 \oplus C$ {bit-wise XOR first line $a_1$ with
                                   constant $C$ : $C=<c_1, \ldots, c_m>$}
7   for $i \leftarrow 2$ to $m$ do
8       ShiftRight($C$)
9       $D \leftarrow C$ {copy $C$ in $D$; $D= <d_1,\ldots,d_{2m-1}>$}
10      if Degree($D$) > $m$ then
11          for $j \leftarrow m+1$ to $2m-1$ do
12              if $d_j = 1$ then $D \leftarrow D \oplus x^{j-1}$
13      $t_i \leftarrow t_i \oplus D$
```

Figure 3. The algorithm of the constant multiplier structure generation.

Moreover, it can be surprisingly discovered that a universal multiplier implemented by constant multipliers has a more efficient structure (from the hardware expenditure point of view) in comparison with the known ones [7].

IV. M-CODE DECODING

As was outlined above the matroid codes are qualitatively different from other ECC types. Decoding "embraces" two process: recognition of erroneous combination of symbols in the received vector **v** and the choice of the appropriate subsystem of linear equations to restore the original transmited vector **x**.



a) $\mathbf{x} \leftarrow \mathbf{x}\cdot 2$ b) $\mathbf{x} \leftarrow \mathbf{x}\cdot 3$ c) $\mathbf{x} \leftarrow \mathbf{x}\cdot 4$ d) $\mathbf{x} \leftarrow \mathbf{x}\cdot 5$ e) $\mathbf{x} \leftarrow \mathbf{x}\cdot 6$ f) $\mathbf{x} \leftarrow \mathbf{x}\cdot 7$

Figure 4. Circuits of constant multipliers **x** by **c** over $\mathbf{GF}(2^3)$ with $p(x)=1+x+x^3$, where $\mathbf{x}=<x_2, x_1, x_0>$ and $\mathbf{c}\in\{2,\ldots,7\}$.

There are $\sum_{i=1}^{r}\binom{i}{2r}$ combinations of possible errors that can be recognised by decoder. So in the Example 1 can be $\binom{1}{6}=6$ single errors, $\binom{2}{6}=15$ double errors and $\binom{3}{6}=20$ triple errors.

To solve the recognition task complexity it is proposed a *serial-parallel decoding* procedure. This procedure is based on hierarchy solutions representations by Boolean lattice. Such representation allows to regulate the subsystem equations choice and decrease (to minimum) the number of recognition steps.

$$\begin{tikzcd}
 & \begin{bmatrix} v_1 & v_2 & v_4 \\ v_1 & v_2 & v_5 \\ v_1 & v_2 & v_6 \end{bmatrix} \arrow[r] \arrow[rdd] & \begin{bmatrix} v_2 & v_4 & v_5 \\ v_2 & v_4 & v_6 \\ v_2 & v_5 & v_6 \end{bmatrix} \arrow[r, dash] & v_2 \\
\begin{bmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \end{bmatrix} \arrow[ru] \arrow[r] \arrow[rd] & \begin{bmatrix} v_2 & v_3 & v_4 \\ v_2 & v_3 & v_5 \\ v_2 & v_3 & v_6 \end{bmatrix} \arrow[r] \arrow[ru] & \begin{bmatrix} v_3 & v_4 & v_5 \\ v_3 & v_4 & v_6 \\ v_3 & v_5 & v_6 \end{bmatrix} \arrow[r, dash] & v_3 \\
 & \begin{bmatrix} v_1 & v_3 & v_4 \\ v_1 & v_3 & v_5 \\ v_1 & v_3 & v_6 \end{bmatrix} \arrow[r] \arrow[ru] & \begin{bmatrix} v_1 & v_4 & v_5 \\ v_1 & v_4 & v_6 \\ v_1 & v_5 & v_6 \end{bmatrix} \arrow[r, dash] & v_1
\end{tikzcd}$$

Figure 5. Diagram of serial-parallel M-code decoding

İn Figure 5 is shown the serial-parallel decoding diagram for Example 1. Decoding procedure begin with searching the solution of two subsystem, symbolically represented by its free members: [v_1, v_2, v_3] and [v_4, v_5, v_6]. İf all subsystems fail, i.e. the errors occurs (see continous edge in Figure 5) then other subsystems are verified. Verification is made by comparing the obtained results.

Searching the subsystem's solution is an asyncronious procedure. For example, choose a subsystem from the set of second range of the Figure 5 diagram; let be the set

$$\begin{bmatrix} v_1 & v_2 & v_4 \\ v_1 & v_2 & v_5 \\ v_1 & v_2 & v_6 \end{bmatrix}. \qquad (5)$$

Set (5) contains a common two linear equations, symbolically marked by v_1 and v_2. From this two equations derived, for example, x_1 and x_2 that are common for all three subsystems of (5). That is fine property allowing minimizing the number of calculus. So, from (4) follows: $x_1=2v_1+v_2+5x_3$ and $x_2=3v_1+5v_2+4x_3$. Substitute the corresponding variables in the third equation of subsystems (5); results are:

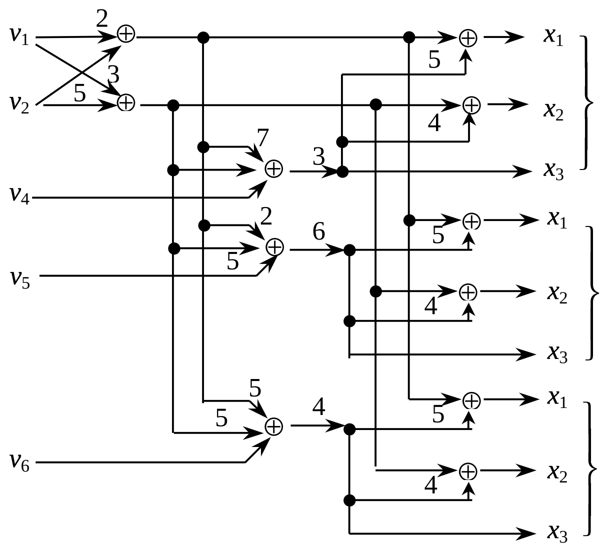


Figure 6. Diagram of the block M-decoder

$$7(2v_1+v_2+5x_3)+(3v_1+5v_2+4x_3)+4x_3=v_4;$$
$$2(2v_1+v_2+5x_3)+(3v_1+5v_2+4x_3)+6x_3=v_5;$$
$$5(2v_1+v_2+5x_3)+5(3v_1+5v_2+4x_3)+6x_3=v_6,$$

from what result:

$$x_3=(1/6)[7(2v_1+v_2)+(3v_1+5v_2)+v_4];$$
$$x_3=(1/3)[2(2v_1+v_2)+(3v_1+5v_2)+v_5];$$
$$x_3=(1/7)[5(2v_1+v_2)+5(3v_1+5v_2)+v_6].$$

In accordance with the proprieties of Galois field divide operation from equations above can be substituted by multiply on reverse (opposite) element, i.e. $a \cdot a^{-1}=1$. Thus, the diagram, shown in Figure 6, gives solution of a subsystem.

From diagram in Figure 5 results that single errors are most difficult to detect (see interrupted edges).

V. BURST ERRORS CORRECTION

By definition *burst error* is a sequence of erroneous and correct symbols "embraced" between two erroneous symbols. The main characteristic of the burst error is the length l (a variable value) and the distance between burst errors that should not be less then l.

For matroid code two cases can be analysed: $l \leq r$ and $l > r$. If $l \leq r$ then the standard (i.e. proposed) technique is applied. In the case $l > r$ the technique of *interleaving* is proposed to apply.

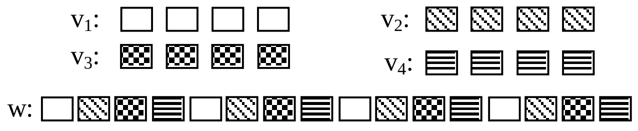


Figure 7. Codewords $\mathbf{v}_i$ and packet of vectors $\mathbf{w}$ resulted from interleaving of $\mathbf{v}_i$, $i=1,\ldots,4$.

Let $\mathbf{w}=(\mathbf{v}_1,\ldots,\mathbf{v}_k)$ is a packet of vectors, where $k=\lceil l/r \rceil$, $\lceil \cdot \rceil$ is a nearest upper integer number. Interleaving of the vectors packet $\mathbf{w}$ consists in redistribution of the vector components v_{ij}, where $v_{ij} \in \mathbf{v}_i$, $\mathbf{v}_i \in \mathbf{w}$, $i=\overline{1,k}$, $j=\overline{1,2r}$.

The easiest way to make an interleaving is to place the vectors' components with the same index j one after another. An example to explain this: let $l=7$ and $r=2$. Then $k=\lceil 7/2 \rceil=4$. In Figure 7 is shown the code words $\mathbf{v}_i$ (i=1,…,4) and the packet $\mathbf{w}$ with redistributed components of codewords; length of packet $|\mathrm{w}|=k \cdot 2r=16$.

The scheme of the interleaving on the M-encoder enter of and the *deinterleaving* scheme on enter of M-decoder does not influence on the coding-decoding algorithm. Therefore burst error can be analysed as a particular case of e-uple errors, where $e \in \{1, 2, \ldots r\}$. Herewith is admitable one burst error of length $l \leq |\mathbf{w}|/2$ on one packet of vectors!

VI. CONCLUSION

Efficient schemes of the matroid code encoding/decoding are proposed and analysed in this paper. Encoding and decoding are performed asynchronously in real time.

By interleaving symbols in the transmitted data it can be achieved a better efficiency at the same parameters (features) of the matroid code.

REFERENCES

1. S.H. Reiger, Codes for correction of "clustered" errors, IRE Transactions on Information Theory, No. 6, pp.16-21, 1960.
2. V.Borshevich, W.Oleinik, A new approach to information coding and protection based on the theory of matroids, Computer Science Journal of Moldova, Vol. 2, No. 1, pp.113-116, 1994.
3. H. Whitney, On the abstract proprieties of linear dependence, American Journal of Mathematics, Vol. 57, pp. 509-533, 1935.
4. M. Aigner, Combinatorial Theory, Springer-Verlag, Berlin, 1979.
5. G.C.Bodean, The matroid error correcting codes: conception and construction, Proceedings of the 8th International Conference on Optimisation of Electrical and Electronic Equipments (OPTIM 2002), Vol.3, May 16-17, pp.729-732, 2002.
6. M.A.Hasan, Look-up table based large finite field multiplication in memory constrained cryptosystems, IEEE Transactions on Computers, Vol. 49, No. 7, pp.749-758, 2000.
7. H.Wu, M.A.Hassan, I.F. Blake, S.Gao, Finite field multiplier using redundant representation, IEEE Transactions on Computers, Vol. 51, No. 11, pp. 1306-1316, 2002.