New Array Codes for Multiple Phased Burst Correction

Mario Blaum[*] Ron M. Roth[†]

Abstract

We present a new optimal family of array codes over $GF(q)$ for correcting multiple phased burst errors and erasures, where each phased burst corresponds to an erroneous, or erased, column in a code array. As for erasures, these array codes have an efficient decoding algorithm which avoids multiplications (or divisions) over extension fields, replacing these operations with cyclic shifts of vectors over $GF(q)$. The erasure decoding algorithm can be adapted easily to handle, in addition, single column errors as well. The new array codes are characterized geometrically by means of parity constraints along certain diagonal lines in each code array, thus generalizing a previously known construction for the special case of two erasures. Algebraically, these array codes can be interpreted as Reed Solomon codes over the ring of polynomials over $GF(q)$ modulo $1+x+\ldots+x^{p-2}+x^{p-1}$ for some prime p which is not the characteristic of $GF(q)$. When q is primitive in $GF(p)$, the resulting codes become (conventional) Reed-Solomon codes of length p over $GF(q^{p-1})$, in which case the new erasure decoding technique can be incorporated into the Berlekamp-Massey algorithm, yielding a faster way to compute the values of any prescribed number of errors.

[*] IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120.

[†] Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel. This work was done while the author was with IBM Research Division, Almaden Research Center, San Jose, CA.

1 Introduction

In this paper, we address the following coding problem: assume that information is stored in $t \times n$ bit arrays. A t-bit column in such an array is said to be *erroneous* if at least one of the bits in that column has been inverted. If the index of an erroneous column is known (say, by means of some detection device), we say that the column has been subject to an *erasure*. Such column errors and erasures, also referred to as *phased burst errors (erasures)*, usually occur in storage systems, e.g., magnetic tapes, where n is the number of tape tracks (i.e., the number of bits in each byte written across the tracks) and erasures correspond to the common case where the read/write head detects a wrongly-coded track: the information on such a track (or rather, on some corrupted portion of it) might be unreadable; however, the identity of the erroneous tracks is known (see [10][17]).

In such magnetic tape applications, the parameter t is set according to some decoding delay requirements i.e., the maximum number of look-ahead cross-track bytes required to decode a certain byte on the tape. In this paper, we adopt the block-coding approach (as opposed to convolutional-coding approach [10][14]): the tape is divided into t cross-track byte segments which are encoded and decoded independently. One of the advantages of the block-coding approach is that it avoids catastrophic propagation of errors when the error correcting capability of the code is exceeded, in which case erroneous decoding might corrupt correct information along whole tracks. Also, since in the convolutional-code case we maintain zero parity along diagonal lines (of various slopes) through the tape [10][14], these lines must be extended at the end of the data stream, thus causing overhead. These extra parity end-bits are not required in the block-code approach. The idea of maintaining zero parity along certain diagonal lines appears also in [19].

Column errors and erasures also arise in disk arrays [7][11][16]. In this scheme, the disks, each containing m memory units, are arranged in $t \times n$ arrays. We can view such disk arrays as m layers of $t \times n$ arrays over the memory-unit alphabet, where the ith layer, $i = 1, 2, \ldots,$ consists of the ith memory unit in each disk. The specific memory unit can be a bit, a byte, a sector, or the whole disk. The proposed coding scheme, presented in this paper, turns out to be as efficient on the binary layers as on any larger memory unit.

Array codes have been widely studied in recent literature. For a survey on the topic, see [8]. Array codes for single burst correction were studied in [5][9][25], and array codes for multiple burst correction were addressed in [6]. Single-track correcting array codes, also called phased burst correcting codes, are treated in [4][12][13]. The array codes presented in this paper are block-type codes capable of correcting multiple-track errors (or erasures).

An optimal solution to the problem of correcting τ column errors and ρ column erasures in $t \times n$ bit arrays can be obtained by using $[n, n-r, r+1]$ Reed-Solomon codes over $GF(2^t)$ [15, Ch. 10], where $r \geq 2\tau + \rho$. In this scheme, each element of $GF(2^t)$ is regarded as a t-bit column, thus transforming the (row-vector) codewords of the Reed-Solomon code into $t \times n$ bit arrays. The optimality of this solution is accounted for by the fact that Reed-Solomon codes are maximum distance separable (MDS) [15, Ch. 11]: for a given length n and minimum distance $r+1$, they have the maximum attainable dimension. Such a coding scheme, based on Reed-Solomon codes, is realizable whenever $t \geq \log_2 n$ (in which case Reed-Solomon codes always exist). Furthermore, any pattern of τ errors and up to $r - 2\tau$ erasures can be decoded quite efficiently using the error-erasure version of the Berlekamp-Massey decoding algorithm [1, Ch. 7], requiring $r \cdot n$ operations (i.e., additions, multiplications or divisions) over $GF(2^t)$ for syndrome calculation and $O\Big((\rho + \tau) \cdot r\Big)$ $\Big(\leq O(r^2)\Big)$ operations for finding the error and erasure values.

In practice, however, the implementation of operations over $GF(2^t)$ might turn out to be quite involved from a hardware point of view. The (asymptotically) most efficient algorithm for performing multiplication over $GF(2^t)$ requires $O(t \log t \log\log t)$ AND/XOR bit operations [20], and this is by using quite a sophisticated hardware (note that, in particular, the implementation of $GF(2^t)$-multiplication requires a number of $GF(2)$-operations which is super-linear in t).

In this paper we aim at simplifying the above encoding-decoding scheme in the multiple-erasure single-error case (i.e., $\tau \leq 1$). We introduce a new family of MDS codes which are similar to Reed-Solomon codes, except that they are defined over certain *polynomial rings*, rather than over fields. In the all-erasure case (or when, in addition, a single error has occurred), these codes provide a significant simplification of the decoding procedure, compared to Reed-Solomon codes of the same parameters. The simplified decoding scheme, which requires lower time complexity and less gates in hardware implementation, is obtained by replacing extension field multiplications with cyclic shifts of binary vectors. The new construction yields codes of $t \times n$ arrays whenever $t = p - 1$ and $n \leq p$ for some prime p (we comment later on how the range of parameters can be extended, still preserving the MDS property and the simplicity of the coding scheme).

More specifically, our codes have parity-check matrices similar to those of Reed-Solomon codes; however, the underlying field $GF(2^t)$ is substituted by the ring of binary polynomials modulo $1 + x + \cdots + x^{t-1} + x^t$. The requirement that $t + 1$ is a prime guarantees that the resulting codes are, indeed, MDS. In the case of multiple-erasure single-error decoding, the only multiplications required during encoding and decoding involve ring elements of the form x^i, $i = 0, 1, 2, \ldots$, as one of the operands. Now, using the fact that $(x - 1)(\sum_{i=0}^{t} x^i) = x^{t+1} - 1$, multiplications boil down to cyclic shifts of binary vectors of length $p = t + 1$, and the implementation of any encountered arithmetic ring operation requires only t XOR bit operations. The encoding-decoding scheme presented in [4] and [7] is, in a way, a special case of the family of codes discussed in this paper for $r = 2$ (two-erasure correction).

When 2 is primitive in $GF(p)$, our codes are equivalent to Reed-Solomon codes of length p over $GF(2^{p-1})$. In such cases, we can apply the proposed simplified procedure for syndrome calculation, then invoke the Berlekamp-Massey algorithm to find the error locations of any prescribed number of errors, and finally apply the proposed erasure decoding technique to compute efficiently the error values. It is a long-standing conjecture that there are "many" fields $GF(p)$ in which 2 is primitive.

We point out that Reed-Solomon and BCH codes over other rings, namely integer rings, were studied by several authors in [2][3][21][23][24]. A generalization of the Berlekamp-Massey algorithm for integer rings is presented in [18].

This paper is organized as follows. In Section 2 we present a geometric definition of the new codes. In Section 3, we show that these codes can be viewed as Reed-Solomon-type codes over the above-mentioned polynomial rings, thus verifying that they are MDS. In Section 4 we present a decoding procedure for the all-erasure case, and in Section 5 we show how the decoding algorithm can be adapted to handle, in addition, single errors as well. Finally, in Section 6 we address the multiple-error case; in particular, we point out the difficulties that arise while attempting to generalize the decoding algorithm for the multiple-error case when the underlying polynomial ring is *not* a field.

2 Geometric presentation of the codes

We start by a geometric description of the array codes considered in this paper. As the construction can be applied to any finite field, rather than just $GF(2)$, we describe the codes in this more general setting. Let $F = GF(q)$ and let p be a prime which is not the characteristic of F (that is, $\gcd(p, q) = 1$). For an integer $n \le p$, let $\mathcal{M}(p-1, n)$ denote the space of all $(p-1) \times n$ matrices (arrays) $\Gamma = [c_{i,j}]_{i=0,j=0}^{p-2\,n-1}$ over F. To simplify notations in the sequel, we shall assume that each array $\Gamma \in \mathcal{M}(p-1, n)$ has an extra all-zero row $[c_{p-1,0}\, c_{p-1,1}\, \ldots\, c_{p-1,n-1}]$, allowing the first index i in $c_{i,j}$ to range from 0 to $p-1$.

For an integer a, let $\langle a \rangle_p$ stand for the integer $b \in \{0, 1, \ldots, p-1\}$ such that $b \equiv a \pmod{p}$. The subscript p will sometimes be omitted if no confusion arises.

The linear array code $\mathcal{C}(p-1, n, r)$ over $F = GF(q)$ is defined as a subspace of $\mathcal{M}(p-1, n)$ consisting of all arrays $\Gamma = [c_{i,j}]_{i,j}$ which satisfy the following $p \cdot r$ linear constraints:

$$\sum_{j=0}^{n-1} c_{\langle m-jl \rangle_p\, ,\, j} = 0\ , \quad 0 \le m \le p-1\ , \quad 0 \le l \le r-1\ . \tag{1}$$

In other words, $\mathcal{C}(p-1, n, r)$ consists of all arrays in $\mathcal{M}(p-1, n)$ such that the entries along the p lines of slope l, $0 \le l \le r-1$, sum to zero. This (geometric) definition of $\mathcal{C}(p-1, n, r)$ is illustrated, for the special case of $n = p = 5$ and $r = 3$, in Figures 1–3, where we have used symbols to mark each one of the parity-check lines defined by (1). Using these figures, it is easy to verify that the array in Figure 4 is an element of the array code $\mathcal{C}(4, 5, 3)$ over $GF(2)$.

Note that $\mathcal{C}(p-1, n, 0) = \mathcal{M}(p-1, n)$ and that $\mathcal{C}(p-1, n, 1)$ is the array code consisting of all arrays in $\mathcal{M}(p-1, n)$ whose rows sum to zero. The array codes studied in [4] and [7] are identical to $\mathcal{C}(p-1, p, 2)$ over $GF(2)$.

Remark 1. It is easy to verify that the code $\mathcal{C}(p-1, n, r)$ can be obtained by taking the arrays of $\mathcal{C}(p-1, p, r)$ whose last $p-n$ columns are zero and then deleting those zero columns. In other words, $\mathcal{C}(p-1, n, r)$ can be obtained from $\mathcal{C}(p-1, p, r)$ by shortening. •

In the next section we show that the codes $\mathcal{C}(p-1, n, r)$, when regarded as $[n, n-r]$ codes over the column alphabet (of size q^{p-1}), are MDS. In particular, it will follow that any

	0	1	2	3	4
0	◇	◇	◇	◇	◇
1	△	△	△	△	△
2	♣	♣	♣	♣	♣
3	◊	◊	◊	◊	◊

Figure 1: Lines of slope 0 in $\mathcal{C}(4,5,3)$.

	0	1	2	3	4
0	◇	△	♣	◊	●
1	△	♣	◊	●	◇
2	♣	◊	●	◇	△
3	◊	●	◇	△	♣

Figure 2: Lines of slope 1 in $\mathcal{C}(4,5,3)$.

	0	1	2	3	4
0	◇	△	♣	◊	●
1	◊	●	◇	△	♣
2	△	♣	◊	●	◇
3	●	◇	△	♣	◊

Figure 3: Lines of slope 2 in $\mathcal{C}(4,5,3)$.

	0	1	2	3	4
0	1	1	0	0	0
1	1	0	0	1	0
2	0	1	1	1	1
3	1	0	0	1	0

Figure 4: An element of the array code $\mathcal{C}(4,5,3)$ over $GF(2)$.

r columns in $\Gamma \in \mathcal{C}(p-1,n,r)$ are uniquely determined by the remaining $n-r$ columns or, equivalently, any $n-r$ columns may serve as information columns (usually the information column locations are preset, say, to the first $n-r$ columns). Therefore, we can regard the encoding procedure as a special case of the erasure decoding procedure discussed in Section 4 (see also Remark 3).

Remark 2. The constraints in (1) all include additions, but no multiplications, over $GF(q)$. Therefore, the array code

$$\mathcal{C}_h(p-1,n,r) \triangleq \underbrace{\mathcal{C}(p-1,n,r) \times \mathcal{C}(p-1,n,r) \times \cdots \times \mathcal{C}(p-1,n,r)}_{h \text{ times}} ,$$

viewed as a subset of the $(p-1) \times n$ matrices over the space $\Big(GF(q)\Big)^h$, is a *linear* array code over $GF(q^h)$, consisting of $(p-1) \times n$ matrices which satisfy (1), now regarded as constraints over $GF(q^h)$. Furthermore, the code $\mathcal{C}_h(p-1,n,r)$ over its column alphabet is MDS as long as $\mathcal{C}(p-1,n,r)$ is MDS. Hence, it suffices to consider array codes over prime base fields $GF(q)$ only; moreover, the decoding of $\mathcal{C}_h(p-1,n,r)$ can be carried out independently (and in parallel) on each one of the h layers over $GF(q)$ of the code arrays of $\mathcal{C}_h(p-1,n,r)$. •

3 Algebraic presentation of the codes

In this section we obtain an equivalent definition for $\mathcal{C}(p-1,n,r)$ over the column alphabet (of size q^{p-1}). Interpreting this alphabet as a polynomial ring, we reveal the relationship between $\mathcal{C}(p-1,n,r)$ and Reed-Solomon codes.

Let $F = GF(q)$ and let p be a prime such that $\gcd(p,q) = 1$. Denote by $M_p(x)$ the polynomial $\sum_{i=0}^{p-1} x^i$ over F, and let $\mathcal{R}_p = \mathcal{R}_p(q)$ be the ring of polynomials of degree $< p-1$ over F with multiplication taken modulo $M_p(x)$. Also, let $\mathcal{R}_p^*$ denote the multiplicative group of the polynomials in $\mathcal{R}_p$ which are relatively prime to $M_p(x)$. To avoid confusion in the sequel, we shall use the indeterminate α instead of x when we refer to polynomials as elements of $\mathcal{R}_p$. This will also indicate whether operations are taken over the ring of polynomials $F[x]$ or, rather, over the ring $\mathcal{R}_p$.

First observe that $\gcd(x^l, M_p(x)) = 1$, implying $\alpha^l \in \mathcal{R}_p^*$. Also, $x^p - 1 = (x-1)M_p(x)$, implying $\alpha^p = 1$. Since $\alpha \neq 1$, we must have $\mathcal{O}(\alpha) = p$, where $\mathcal{O}(\cdot)$ stands for the multiplicative order in $\mathcal{R}_p^*$. It thus follows that $\mathcal{O}(\alpha^l) = p$ for all $l \not\equiv 0 \pmod{p}$. Note also that $\mathcal{R}_p(q)$ is isomorphic to $GF(q^{p-1})$ when $M_p(x)$ is an irreducible polynomial over F. This happens if and only if q is a primitive element in $GF(p)$ (see, for instance, [15, p. 506]).

Consider the following $r \times r$ matrix

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{r-1} \\ \alpha_0^2 & \alpha_1^2 & \ldots & \alpha_{r-1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^{r-1} & \alpha_1^{r-1} & \ldots & \alpha_{r-1}^{r-1} \end{bmatrix}$$

over $\mathcal{R}_p(q)$, where the α_i's are distinct and $\alpha_i = \alpha^{j_i}$, $0 \leq i \leq r-1$ $(\leq p-1)$. We claim that the columns of V are linearly independent over $\mathcal{R}_p(q)$. Clearly, if $\mathcal{R}_p(q) \cong GF(q^{p-1})$, then V a Vandermonde matrix which is nonsingular for distinct α_i. Referring to the general case, we first note that for $1 \leq l < p$,

$$\gcd(x^l - 1, x^p - 1) = x^{\gcd(l,p)} - 1 = x - 1 \ .$$

Since p is not the characteristic of F we have $M_p(1) \neq 0$ i.e., $\gcd(x-1, M_p(x)) = 1$. Hence, $\alpha^l - 1 \in \mathcal{R}_p^*$ and, therefore, each element of the form $\alpha^m - \alpha^l$, $m \not\equiv l \pmod{p}$, has a multiplicative inverse in $\mathcal{R}_p$. It follows that the determinant of V is an element of $\mathcal{R}_p^*$, implying that V has an inverse matrix V^{-1} over $\mathcal{R}_p$. Hence, the columns of V are linearly independent over $\mathcal{R}_p$.

For $r \leq n \leq p$, let H be the $r \times n$ matrix over $\mathcal{R}_p(q)$ defined by

$$H = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{r-1} & \alpha^{2(r-1)} & \ldots & \alpha^{(n-1)(r-1)} \end{bmatrix} ,$$

and let C be the linear code of length n over $\mathcal{R}_p(q)$ with H as a parity-check matrix i.e.,

$$C \triangleq \left\{ \mathbf{c} \in (\mathcal{R}_p)^n \mid \mathbf{c}H^T = \mathbf{0} \right\} \ . \tag{2}$$

Since every r columns in H are linearly independent over $\mathcal{R}_p$, C is a linear code of length n, dimension $n-r$ and minimum distance $r+1$ over $\mathcal{R}_p$. In particular, it is MDS.

Remark 3. When $n = p$, C is a cyclic code over $\mathcal{R}_p$ whose generator polynomial is given by $g(z) = (z-1)(z-\alpha)\cdots(z-\alpha^{r-1})$. Therefore, in general (i.e., when $n \le p$), the code C is an $[n, n-r]$ shortened cyclic code over $\mathcal{R}_p$ with the same generator polynomial $g(z)$. This gives rise to a nonsystematic encoder for C which maps an information polynomial $u(z)$ of degree $< n-r$ over $\mathcal{R}_p$ into a codeword polynomial $c(z) = \sum_{j=0}^{n-1} c_j z^j = u(z)g(z)$, where $\mathbf{c} = [c_0\, c_1\, \ldots\, c_{n-1}] \in C$ (see also Remark 8). The (common) division encoder which computes the residue $s(z)$ of $z^r u(z)$ modulo $g(z)$ to yield a codeword $c(z) = z^r u(z) - s(z)$ is equivalent to the encoder obtained by using the erasure decoding algorithm. •

In the following theorem we show that C is identical to the code obtained by regarding the columns of the arrays in $\mathcal{C}(p-1, n, r)$ as elements in $\mathcal{R}_p$.

Theorem 1. *Let C be the code over $\mathcal{R}_p(q)$ defined by (2) and let $\boldsymbol{\alpha} \triangleq [1\, \alpha\, \alpha^2\, \ldots\, \alpha^{p-2}]$ be a basis of $\mathcal{R}_p(q)$ over $F = GF(q)$. Then,*

$$C \;=\; \{\mathbf{c} = \boldsymbol{\alpha}\Gamma \mid \Gamma \in \mathcal{C}(p-1, n, r)\} \ .$$

Proof. Let $\mathbf{c} = [c_0\, c_1\, \ldots\, c_{n-1}]$ be a codeword in C, with each entry c_j being an element in $\mathcal{R}_p$. Representing each component c_j as a polynomial

$$c_j = c_j(\alpha) = \sum_{i=0}^{p-2} c_{i,j}\alpha^i \ , \quad c_{i,j} \in F \ , \quad 0 \le j \le n-1 \ ,$$

we can write $\mathbf{c} = \boldsymbol{\alpha}\Gamma$, where Γ is the following $(p-1)\times n$ matrix over F:

$$\Gamma = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & \cdots & c_{0,n-1} \\ c_{1,0} & c_{1,1} & c_{1,2} & \cdots & c_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{p-2,0} & c_{p-2,1} & c_{p-2,2} & \cdots & c_{p-2,n-1} \end{bmatrix} \ .$$

Rewriting the parity conditions $\mathbf{c}H^T = \boldsymbol{\alpha}\Gamma H^T = \mathbf{0}$, we obtain,

$$\sum_{j=0}^{n-1} \alpha^{jl} c_j(\alpha) = \sum_{j=0}^{n-1} \alpha^{jl} \sum_{i=0}^{p-2} c_{i,j}\alpha^i = 0 \ , \quad 0 \le l \le r-1 \ . \qquad (3)$$

For $l = 0$ we thus have

$$\sum_{i=0}^{p-2} \alpha^i \sum_{j=0}^{n-1} c_{i,j} = 0$$

or, since the α^i are linearly independent over F,

$$\sum_{j=0}^{n-1} c_{i,j} = 0 \; , \quad 0 \le i \le p-2 \; . \tag{4}$$

This means that each row in Γ must sum to zero. In particular,

$$\sum_{j=0}^{n-1} c_j(1) = \sum_{j=0}^{n-1} \sum_{i=0}^{p-2} c_{i,j} = 0 \; . \tag{5}$$

Writing (3) and (5) as polynomial congruences over $F[x]$, we obtain

$$\sum_{j=0}^{n-1} x^{jl} c_j(x) \equiv 0 \pmod{M_p(x)} \; , \quad 0 \le l \le r-1 \; , \tag{6}$$

and

$$\sum_{j=0}^{n-1} x^{jl} c_j(x) \equiv 0 \pmod{x-1} \; , \quad 0 \le l \le r-1 \; . \tag{7}$$

As $M_p(x)$ and $x-1$ are relatively prime, Equations (6) and (7) are equivalent to

$$\sum_{j=0}^{n-1} x^{jl} c_j(x) \equiv 0 \pmod{x^p - 1} \; , \quad 0 \le l \le r-1 \; ,$$

or

$$\sum_{j=0}^{n-1} x^{jl} \sum_{i=0}^{p-1} c_{i,j} x^i \equiv 0 \pmod{x^p - 1} \; , \quad 0 \le l \le r-1 \; , \tag{8}$$

where we have assumed an extra zero row $[c_{p-1,0}\, c_{p-1,1} \, \ldots \, c_{p-1,n-1}]$ appended to Γ. Note that (8) and (3) are *equivalent* constraints; this is due to the fact that $c_j(x)$, $j = 0, 1, \ldots, n-1$, are all polynomials of degree $\le p-2$, although the congruences in (8) are taken modulo a polynomial of degree p.

Rearranging the left-hand side of (8), we obtain,

$$\begin{aligned} \sum_{j=0}^{n-1} x^{jl} \sum_{i=0}^{p-1} c_{i,j} x^i &\equiv \sum_{j=0}^{n-1} \sum_{i=0}^{p-1} c_{i,j} x^{i+jl} \pmod{x^p - 1} \\ &\equiv \sum_{m=0}^{p-1} x^m \sum_{\substack{\langle i+jl \rangle_p = m \\ 0 \le j \le n-1}} c_{i,j} \pmod{x^p - 1} \end{aligned}$$

$$\begin{aligned} &\equiv \sum_{m=0}^{p-1} x^m \sum_{j=0}^{n-1} c_{\langle m-jl\rangle_p\,,j} \pmod{x^p-1} \qquad (9)\\ &\equiv 0 \pmod{x^p-1} \quad . \qquad (10)\end{aligned}$$

As the polynomial in (9) is of degree less than p, the congruence in (10) is, in fact, equality over $F[x]$. We thus have

$$\sum_{m=0}^{p-1} x^m \sum_{j=0}^{n-1} c_{\langle m-jl\rangle\,,j} = 0 \,, \quad 0 \le l \le r-1$$

and, therefore,

$$\sum_{j=0}^{n-1} c_{\langle m-jl\rangle\,,j} = 0 \,, \quad 0 \le m \le p-1 \,, \quad 0 \le l \le r-1 \,. \qquad (11)$$

This coincides with the constraints in (1) i.e., with the definition of $\mathcal{C}(p-1,n,r)$. □

Remark 4. The constraint set (4) (parity on rows) was required in the proof to establish the equivalence between the congruences modulo x^p-1 and those modulo $M_p(x)$ (Equations (6), (7) and (8)). To this end, the roots of the Reed-Solomon-type code C should contain the element 1. This explains the distinction between the case of $l=0$ and that of other values of l. Note, however, that we still obtain an MDS code over $\mathcal{R}_p$ even if 1 is not one of the code (consecutive) roots. The MDS property remains also if we add a "zero column" $[1\,0\,0\,\ldots\,0]^T$, or an "infinity column" $[0\,0\,\ldots\,0\,1]^T$, to the parity-check matrix H.●

Remark 5. The constraint set (1) was used also in [10] and [14] to define codes over rings of polynomials modulo x^t-1; however, such codes are usually *not* MDS. As we have just shown, the codes $\mathcal{C}(p-1,n,r)$ over $F=GF(q)$ can be regarded as constructions over the ring of polynomials over F modulo x^p-1, provided that each entry in a codeword is a polynomial of degree $\le p-2$. The fact that p is a prime (different from the characteristic of F) and that the degrees are restricted *both* account for the fact that the code $\mathcal{C}(p-1,n,r)$ is MDS over the column alphabet. It is easy to verify that our construction can be generalized to any ring $\mathcal{R}_{t+1}(q)$ of polynomials over F modulo $\sum_{i=0}^{t} x^i$ (to yield codes $\mathcal{C}(t,n,r)$ over F), provided that $\gcd(t+1,q)=1$ and that n is not greater than any divisor >1 of $t+1$. Under such conditions, we have $\gcd(x^l-1,\sum_{i=0}^{t} x^i)=1$ for any $1 \le l < n$, implying that the resulting array code (consisting of $t \times n$ arrays over F) is MDS over the column alphabet. Clearly, we can still replace the parity constraints over $\mathcal{R}_{t+1}(q)$ by congruences modulo $x^{t+1}-1$. However, since the conditions on n might be too restrictive when $t+1$ is composite, we choose to continue the discussion assuming that $t+1=p$ is a prime. •

4 All-erasure decoding

We now describe the decoding procedure of $\rho \le r$ erasures (and no errors) for the array code $\mathcal{C}(p-1,n,r)$ over $F=GF(q)$ or, rather, for the associated code C over $\mathcal{R}_p(q)$ defined by (2). Later on, in Sections 5 and 6, we show how to incorporate error correction in the proposed algorithm. Let $\mathbf{c} \in C$ be the transmitted codeword and assume that ρ erasures have occurred at the distinct locations $j_0, j_1, \ldots, j_{\rho-1}$. Let $\mathbf{y} = [y_0\, y_1\, \ldots\, y_{n-1}]$ be the vector over $\mathcal{R}_p$ which equals $\mathbf{c}$ at all coordinates except the erased ones, in which it is zero. Also, let $\alpha_i = \alpha^{j_i}$ and $e_i = -c_{j_i}$ denote the erasure locator and erasure value, respectively, for any j_i, $0 \le i \le \rho-1$. First, we calculate the syndrome values

$$S_l = (\mathbf{y}H^T)_l \;=\; \sum_{j=0}^{n-1} y_j \alpha^{jl} \;=\; \sum_{i=0}^{\rho-1} e_i \alpha_i^l \;, \quad 0 \le l \le r-1 \tag{12}$$

(all operations over $\mathcal{R}_p$). Having found the syndrome, the decoder has to solve the linear system

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{\rho-1} \\ \alpha_0^2 & \alpha_1^2 & \ldots & \alpha_{\rho-1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^{\rho-1} & \alpha_1^{\rho-1} & \ldots & \alpha_{\rho-1}^{\rho-1} \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{\rho-1} \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{\rho-1} \end{bmatrix} \tag{13}$$

(note that ρ syndrome values are sufficient for decoding ρ erasures).

When $\rho=1$, we have $e_0 = S_0$. Assume that $\rho > 1$ and define the polynomials

$$G_i(z) = \prod_{\substack{s=0 \\ s \ne i}}^{\rho-1} (1 - \alpha_s z) = \sum_{l=0}^{\rho-1} G_{i,l} z^l \;, \quad 0 \le i \le \rho-1 \;, \tag{14}$$

where $G_{i,l}$ is the lth symmetric function on $\{-\alpha_s\}_{\substack{0 \le s \le \rho-1 \\ s \ne i}}$. We easily see, from the definition of $G_i(z)$, that

$$\alpha_m^{\rho-1} G_i(\alpha_m^{-1}) = \begin{cases} \prod_{\substack{s=0 \\ s\neq i}}^{\rho-1} (\alpha_i - \alpha_s) & \text{if } m = i \\ 0 & \text{otherwise} \end{cases} \quad , \quad 0 \le m \le \rho - 1 \; . \tag{15}$$

Now,

$$\begin{aligned} \sum_{l=0}^{\rho-1} G_{i,\rho-1-l}\, S_l &= \sum_{l=0}^{\rho-1} G_{i,\rho-1-l} \sum_{m=0}^{\rho-1} e_m \alpha_m^l \\ &= \sum_{m=0}^{\rho-1} e_m \sum_{l=0}^{\rho-1} G_{i,\rho-1-l} \alpha_m^l \\ &= \sum_{m=0}^{\rho-1} e_m \alpha_m^{\rho-1} G_i(\alpha_m^{-1}) \\ &= \Big(\prod_{\substack{s=0 \\ s\neq i}}^{\rho-1} (\alpha_i - \alpha_s) \Big) e_i \; , \quad 0 \le i \le \rho - 1 \; , \end{aligned} \tag{16}$$

the last equality obtained by (15). Summarizing (16), the solution for the e_i in (13) is given by

$$\Big(\prod_{\substack{s=0 \\ s\neq i}}^{\rho-1} (\alpha_i - \alpha_s) \Big) e_i \; = \; \sum_{l=0}^{\rho-1} G_{i,\rho-1-l} S_l \; , \quad 0 \le i \le \rho - 1 \; . \tag{17}$$

Note that, up to scalar row multipliers, (17) is the same as Equation (13) with both sides multiplied by the inverse of $V \stackrel{\Delta}{=} [\, \alpha_i^l \,]_{l,i=0}^{\rho-1}$. Now, recalling that S_l, e_i and $G_{i,l}$, as elements of $\mathcal{R}_p$, can be represented by polynomials over F (i.e., $S_l = S_l(\alpha) = \sum_{m=0}^{p-2} s_{m,l} \alpha^m$, $e_i = e_i(\alpha) = \sum_{m=0}^{p-2} e_{m,i} \alpha^m$ and $G_{i,l} = G_{i,l}(\alpha) = \sum_{m=0}^{p-2} g_{m,i,l} \alpha^m$), we can rewrite (17) as the following polynomial congruence over $F[x]$:

$$\Big(\prod_{\substack{s=0 \\ s\neq i}}^{\rho-1} (x^{j_i} - x^{j_s}) \Big) e_i(x) \; \equiv \; \sum_{l=0}^{\rho-1} G_{i,\rho-1-l}(x)\, S_l(x) \pmod{M_p(x)} \; , \quad 0 \le i \le \rho - 1 \; . \tag{18}$$

We now present an efficient procedure (Steps 1–4 below) for extracting the values of e_i from (17). The basic idea is to break the multiplications modulo $M_p(x)$ into two pieces: the first step involves a multiplication modulo $x^p - 1$, whereas the second step rectifies the result modulo $M_p(x)$. The advantage of this approach stems from the fact that the only multiplications we shall need in order to calculate the right-hand side of (18) involve one operand of the form x^m; now, multiplying by x^m modulo $x^p - 1$ is simply a cyclic shift.

For example, let $b(\alpha) = \sum_{i=0}^{p-2} b_i \alpha^i \in \mathcal{R}_p$, and suppose we like to perform the multiplication $\alpha^m b(\alpha)$, resulting in an element $a(\alpha) = \sum_{i=0}^{p-2} a_i \alpha^i \in \mathcal{R}_p$. In other words, we would like to find a polynomial $a(x) \in F[x]$ of degree $< p-1$ such that

$$a(x) \equiv x^m b(x) \pmod{M_p(x)} .$$

First, we find a polynomial $\hat{a}(x) = \sum_{i=0}^{p-1} \hat{a}_i x^i \in F[x]$ of degree $< p$ such that

$$\hat{a}(x) \equiv x^m b(x) \pmod{x^p - 1} ;$$

note that, in particular, $a(x) \equiv \hat{a}(x) \pmod{M_p(x)}$. Now, finding the coefficients of $\hat{a}(x)$ is easy; these coefficients are given by

$$\hat{a}_i = b_{\langle i-m \rangle_p} , \quad 0 \le i \le p-1 ,$$

where we define $b_{p-1} \triangleq 0$. Second, we rectify $\hat{a}(x)$ to obtain $a(x)$. This is done by observing that

$$a(x) \equiv \hat{a}(x) \equiv \hat{a}(x) - \hat{a}_{p-1} M_p(x) \pmod{M_p(x)}$$

and, furthermore, that the degree of $\hat{a}(x) - \hat{a}_{p-1} M_p(x)$ is at most $p-2$; therefore, we must have $a(x) = \hat{a}(x) - \hat{a}_{p-1} M_p(x)$ i.e.,

$$a_i = \hat{a}_i - \hat{a}_{p-1} = b_{\langle i-m \rangle_p} - b_{\langle -m-1 \rangle_p} , \quad 0 \le i \le p-2 . \tag{19}$$

Observe that (19) yields simple expressions for the coefficients of $a(\alpha)$ in terms of those of $b(\alpha)$ (and, therefore, we could use these expressions directly to calculate $a(\alpha)$). However, in general, the computation might involve several multiplications. In order to get any desired result $a(\alpha) \in \mathcal{R}_p$ in such cases, we first carry out all calculations modulo $x^p - 1$, ending with a polynomial $\hat{a}(x) \in F[x]$ of degree $< p$; then, as a final step, we obtain the coefficients of $a(x)$ by $a_i \leftarrow \hat{a}_i - \hat{a}_{p-1}$, $0 \le i \le p-2$.

Returning to Equation (17), we start by calculating the right-hand side of (18) modulo $x^p - 1$; to this end, we need to calculate polynomials $\hat{S}_l(x) = \sum_{m=0}^{p-1} \hat{s}_{m,l} x^m$ such that $\hat{S}_l(x) \equiv S_l(x) \pmod{M_p(x)}$. Let $y_j = y_j(\alpha)$ $(\in \mathcal{R}_p)$, where $y_j(x) = \sum_{i=0}^{p-1} y_{i,j} x^i$ $(\in F[x])$ and $y_{p-1,j} =$

0, $0 \le j \le n-1$. Following the derivation of (11), Equation (12) yields the following simple procedure for calculating the coefficients of $\hat{S}_l(x)$:

```
/* Step 1: Syndrome calculation (global for all erasures) */
for $l \leftarrow 0$ to $r-1$ do /* run for each syndrome value $\hat{S}_l(x) = \sum_{m=0}^{p-1} \hat{s}_{m,l} x^m$ */
    for $m \leftarrow 0$ to $p-1$ do
        $\hat{s}_{m,l} \leftarrow \sum_{j=0}^{n-1} y_{\langle m-jl \rangle_p}$ .
```

Clearly, this first step of the decoding procedure requires no more than $r \cdot p \cdot n$ additions over F.

Remark 6. As mentioned earlier, in the case of all-erasure decoding, it suffices to compute only the first ρ syndrome values. Therefore, in this special case, we can assume throughout this section that r and ρ are identical. However, in order to make the forthcoming derivation more general (as to allow error correction as well), we prefer to distinguish between the redundancy r and the number of erasures ρ. •

For each i, $i = 0, 1, 2, \ldots, \rho - 1$, let $\sigma_i = \sigma_i(\alpha)$ denote the right-hand side of (17) i.e.,

$$\sigma_i(x) \;\equiv\; \sum_{l=0}^{\rho-1} G_{i,\rho-1-l}(x)\, S_l(x) \pmod{M_p(x)} , \quad 0 \le i \le \rho - 1 .$$

Let $R(z) = R(\alpha; z) = \prod_{s=0}^{\rho-1}(1 - \alpha_s z)$ be the erasure-locator polynomial in the indeterminate z over $\mathcal{R}_p$, and let $S(z) = S(\alpha; z) = \sum_{l=0}^{r-1} S_l z^l$ be the syndrome polynomial. Denote by $Q(z)$ the product of these two polynomials i.e., $Q(z) = Q(\alpha; z) = R(z)\, S(z)$. It is easy to verify that, for each i, σ_i is the coefficient of $z^{\rho-1}$ in the quotient polynomial

$$G_i(z)\, S(z) \;=\; \frac{R(z)\, S(z)}{1 - \alpha_i z} \;=\; \frac{Q(z)}{1 - \alpha_i z}$$

over $\mathcal{R}_p$. Note that the definition of $Q(z)$ is independent of the erasure index i and, therefore, it need not be computed more than once throughout the erasure decoding process.

Remark 7. There exists a known method (the Forney algorithm [1, p. 184]) for calculating the erasure values e_i out of the erasure-evaluator polynomial (which is congruent to $Q(z)$ modulo z^ρ). Obviously, the same expression for e_i is obtained by equating the left-hand side of (17) with the coefficient of $z^{\rho-1}$ in $Q(z)/(1 - \alpha_i z)$. •

In order to extract σ_i from $Q(z)/(1-\alpha_i z)$, we first transform into calculations modulo $x^p - 1$. More specifically, we compute a polynomial $\hat{Q}(x;z) = \sum_{l=0}^{r+\rho-1} \hat{Q}_l(x) z^l$ such that

$$\hat{Q}(x;z) \equiv Q(x;z) \pmod{M_p(x)} ,$$

where the congruence applies to each z-coefficient $\hat{Q}_l(x) \in F[x]$ (of degree $< p$) in $\hat{Q}(x;z)$ with the corresponding z-coefficient in $Q(x;z)$. The polynomial $\hat{Q}(x;z)$ can be computed easily using the following procedure:

```
/* Step 2: Computation of $\hat{Q}(x;z)$ (global for all erasures) */
$\hat{Q}(x;z) \leftarrow \hat{S}(x;z) \triangleq \sum_{l=0}^{r-1} \hat{S}_l(x) z^l$ (the output of Step 1) ;
for $s \leftarrow 0$ to $\rho - 1$ do
    $\hat{Q}(x;z) \leftarrow (1 - x^{j_s} z)\hat{Q}(x;z) \pmod{x^p - 1}$ .
```

Representing each z-coefficient $\hat{Q}_l(x)$ of $\hat{Q}(x;z)$ as a p-vector over F, each multiplication by $1 - x^{j_s} z$ involves a p-block shift of the coefficients of $\hat{Q}_l(x)$ over F (by virtue of the multiplier z); then a cyclic shift by j_s positions of each coefficient $\hat{Q}_l(x)$ (multiplication by x^{j_s}); and finally subtracting the result from the original copy of $\hat{Q}(x;z)$. The overall complexity of Step 2 thus totals to $\frac{1}{2}\rho \cdot (2r + \rho - 1) \cdot p$ $(\leq \frac{3}{2} r^2 p)$ additions over F, plus $\rho \cdot (2r + \rho - 1)$ shifts of vectors of length p over F. In particular, we do not need any multiplications over F. Note that, in general, the standard erasure decoding of Reed-Solomon codes over $GF(q^t)$ requires multiplications over the base-field $GF(q)$; moreover, the number of necessary base-field operations is proportional to $\rho \cdot r \cdot t \cdot f(t)$ where $\lim_{t\to\infty} f(t) = \infty$.

Next, we find the coefficient of $z^{\rho-1}$ in the quotient polynomial $\hat{Q}(x;z)/(1 - x^{j_i} z)$ to yield a polynomial $\hat{\sigma}_i(x)$ which is congruent to $\sigma_i(x)$ modulo $M_p(x)$. The following direct division procedure computes the desired coefficient of $z^{\rho-1}$ by iteratively extracting the coefficients of z^l, $l = 0, 1, \ldots, \rho - 1$, in that quotient polynomial.

```
/* Step 3: Computation of the right-hand side of (17) */
/* $\hat{Q}(x;z) = \sum_{l=0}^{r+\rho-1} \hat{Q}_l(x) z^l$ is the output of Step 2 */
for $i \leftarrow 0$ to $\rho - 1$ do begin /* run for each erasure */
    $\hat{\sigma}_i(x)$ $(= \sum_{m=0}^{p-1} \hat{\sigma}_{m,i} x^m)$ $\leftarrow \hat{Q}_0(x)$ ;
    for $l \leftarrow 1$ to $\rho - 1$ do
```

```
            $\hat{\sigma}_i(x) \leftarrow \hat{Q}_l(x) + x^{j_i}\hat{\sigma}_i(x) \pmod{x^p - 1}$ ;
        $\sigma_i(x) \leftarrow \hat{\sigma}_i(x) - \hat{\sigma}_{p-1,i}M_p(x)$
    end .
```

The inner loop of the above procedure involves a cyclic shift (by j_i) of the coefficients $\hat{\sigma}_{m,i} \in F$ of $\hat{\sigma}_i(x)$, and then adding the result to the coefficients of $\hat{Q}_l(x)$. Hence, Step 3 requires $\rho^2 \cdot p$ additions over F and ρ^2 cyclic shifts of p-vectors to find all the values of $\sigma_i(\alpha)$, $0 \leq i \leq \rho - 1$, in $\mathcal{R}_p$ (including the final rectification modulo $M_p(x)$).

Remark 8. Referring to the nonsystematic encoder mentioned in Remark 3, a procedure similar to Step 2 allows to multiply the information polynomial $u(z)$ with the generator polynomial $g(z)$ using up to $r \cdot p \cdot n$ additions over F, plus $2 \cdot r \cdot n$ (constant) shifts of vectors of length p over F. The same order of time complexity is then required at the decoder end to extract $u(z)$ out of the decoded codeword $c(z) = u(z)g(z)$, using a procedure which is basically similar to Step 3. •

Having calculated the right-hand side of (17), we now "peel off" iteratively the multipliers $(\alpha_i - \alpha_s)$ in the left-hand side of (17) to finally obtain the value of e_i for every i. Each resulting iteration is based on the following lemma.

Lemma 1. *Let $a(\alpha) = \sum_{i=0}^{p-2} a_i\alpha^i \in \mathcal{R}_p(q)$ (where $a_i \in F = GF(q)$ and $\gcd(p,q) = 1$), and define the so-called unbiased coefficients $\tilde{a}_i$ by $\tilde{a}_i \triangleq a_i - \frac{1}{p}\sum_{j=0}^{p-1} a_j$, $0 \leq i \leq p-1$, where $a_{p-1} \triangleq 0$. Then, for any $m \not\equiv 0 \pmod{p}$, the coefficients of the unique solution $b(\alpha) = \sum_{i=0}^{p-2} b_i\alpha^i$ for*

$$a(\alpha) \;=\; (\alpha^m - 1)\, b(\alpha)$$

in $\mathcal{R}_p(q)$ are given by the recursion

$$b_{\langle -km-1\rangle_p} \;=\; b_{\langle -(k-1)m-1\rangle_p} + \tilde{a}_{\langle -(k-1)m-1\rangle_p} \;, \quad 1 \leq k \leq p-1 \;,$$

with $b_{p-1} \triangleq 0$.

(Recall that $1/p$ is well-defined in F since $\gcd(p,q) = 1$.)

Proof. By (19) we have

$$a_i \;=\; b_{\langle i-m\rangle} - b_{\langle -m-1\rangle} - b_i \;, \quad 0 \le i \le p-1 \;. \tag{20}$$

Summing both sides of (20) over all i, we obtain

$$\sum_{i=0}^{p-1} a_i \;=\; -p \cdot b_{\langle -m-1\rangle}$$

i.e., $b_{\langle -m-1\rangle} \;=\; -(1/p)\sum_{i=0}^{p-1} a_i$. This proves the lemma for $k=1$. The case of larger k is obtained by substituting $i = \langle -(k-1)m-1\rangle_p$, and the value of $b_{\langle -m-1\rangle}$, into (20). □

It is easy to verify that the inverse of $\alpha - 1$ in $\mathcal{R}_p$ is given by

$$P(\alpha) \;\triangleq\; -(1/p)\sum_{i=0}^{p-2}(p-1-i)\alpha^i$$

and, therefore, the inverse of $\alpha^m - 1$, $m \not\equiv 0 \pmod{p}$, is $P(\alpha^m) = -(1/p)\sum_{i=0}^{p-2}(p-1-i)\alpha^{im}$. Lemma 1 is simply a presentation of the coefficients of $b(\alpha) = P(\alpha^m)a(\alpha)$.

Corollary 1. *Let $a(\alpha) = \sum_{i=0}^{p-2} a_i\alpha^i$ and $\tilde{a}_i$ be as in Lemma 1. Then, for any $m \not\equiv l \pmod{p}$, the coefficients of the unique solution $b(\alpha) = \sum_{i=0}^{p-2} b_i\alpha^i$ for*

$$a(\alpha) \;=\; (\alpha^m - \alpha^l)\, b(\alpha)$$

in $\mathcal{R}_p(q)$ are given by the recursion

$$b_{\langle -k(m-l)-1\rangle_p} \;=\; b_{\langle -(k-1)(m-l)-1\rangle_p} + \tilde{a}_{\langle -(k-1)(m-l)+l-1\rangle_p} \;, \quad 1 \le k \le p-1 \;,$$

with $b_{p-1} \triangleq 0$.

Proof. Transform $a(\alpha) = (\alpha^m - \alpha^l)b(\alpha)$ into $\alpha^{-l}a(\alpha) = (\alpha^{m-l} - 1)b(\alpha)$ and then apply Lemma 1. □

Example 1. The solution $b(\alpha)$ for $a(\alpha) = (\alpha^5 - \alpha^3)b(\alpha)$ in $\mathcal{R}_7(q)$ is given by

$$\begin{aligned} b_6 &= 0 \\ b_4 &= b_6 + \tilde{a}_2 \\ b_2 &= b_4 + \tilde{a}_0 \\ b_0 &= b_2 + \tilde{a}_5 \\ b_5 &= b_0 + \tilde{a}_3 \\ b_3 &= b_5 + \tilde{a}_1 \\ b_1 &= b_3 + \tilde{a}_6 \end{aligned} \quad .$$

Applying the recursion of Corollary 1 involves no more than $2p$ additions over F (excluding the constant multiplier $1/p$ which boils down to 1 in $GF(2)$), plus calculation of indices (= decimation) modulo p. We point out that these index computations can be bypassed by making use of the layout geometry, giving rise to a modular architecture which is suitable for VLSI design. We demonstrate this in Appendix A.

When $F = GF(2)$, we can further avoid calculating the unbiased coefficients $\tilde{a}_i$ by performing two recursion steps at a time, in which case the unbiasing sum $(1/p)\sum_{j=0}^{p-1} a_j$ cancels out. The resulting recursion, summarized in the next corollary, requires the same number of additions (XOR, $\oplus$) as the one in Corollary 1.

Corollary 2. *Let $a(\alpha)$, m and l be as in Corollary 1 and, in addition, assume that $F = GF(2)$. Then, the coefficients of the unique solution $b(\alpha) = \sum_{i=0}^{p-2} b_i \alpha^i$ for*

$$a(\alpha) \;=\; (\alpha^m \oplus \alpha^l)\, b(\alpha)$$

in $\mathcal{R}_p(2)$ are given by

$$\begin{aligned} b_{\langle -2s(m-l)-1\rangle_p} &= b_{\langle -(2s-2)(m-l)-1\rangle_p} \oplus a_{\langle -(2s-2)(m-l)+l-1\rangle_p} \oplus a_{\langle -(2s-1)(m-l)+l-1\rangle_p} \\ &= \bigoplus_{i=0}^{2s-1} a_{\langle -i(m-l)+l-1\rangle_p}\ , \qquad 1 \le s \le p-1\ , \end{aligned}$$

with $b_{p-1} = 0$.

Example 2. The solution $b(\alpha)$ for $a(\alpha) = (\alpha - \alpha^2)b(\alpha)$ in $\mathcal{R}_5(2)$ is given by

$$\begin{array}{lclclclclcl} b_4 & = & 0 & & & & & & & & \\ b_1 & = & b_4 \oplus a_1 \oplus a_2 & = & & a_1 & \oplus & a_2 & & & \\ b_3 & = & b_1 \oplus a_3 \oplus a_4 & = & & a_1 & \oplus & a_2 & \oplus & a_3 & \\ b_0 & = & b_3 \oplus a_0 \oplus a_1 & = & a_0 & & \oplus & a_2 & \oplus & a_3 & \\ b_2 & = & b_0 \oplus a_2 \oplus a_3 & = & a_0 & & & & & & \end{array}$$

(recall that $a_4 = 0$).

Corollaries 1 and 2 establish the iterative procedure by which the values of e_i can be obtained from (17). For each fixed i, we define the ρ terms $e_i^{(k)} \in \mathcal{R}_p$, $k = 0, 1, \ldots, \rho - 1$, as follows:

$$e_i^{(k)} = \begin{cases} \Big(\prod\limits_{\substack{s=k\\ s\neq i}}^{\rho-1} (\alpha_i - \alpha_s)\Big)\, e_i & \text{for} \quad 0 \le k < i \\ \Big(\prod\limits_{s=k+1}^{\rho-1} (\alpha_i - \alpha_s)\Big)\, e_i & \text{for} \quad i \le k \le \rho - 2 \\ \quad e_i & \text{for} \quad k = \rho - 1 \end{cases} . \tag{21}$$

The following procedure solves iteratively for $e_i^{(k)}$, $k = 1, 2, \ldots, \rho - 1$, starting with $e_i^{(0)} = \sigma_i$ (Equation (17)), and ending with $e_i = e_i^{(\rho-1)}$ (all the $e_i^{(k)}$ are computed into the same variable e_i):

```
/* Step 4: Extracting the erasure values from (17) */
for $i \leftarrow 0$ to $\rho - 1$ do begin /* run for each erasure */
    $e_i \leftarrow \sigma_i$ (the output of Step 3) ;
    for $k \leftarrow 0$ to $\rho - 2$ do
        if $k < i$ then
            $e_i \leftarrow (\alpha_i - \alpha_k)^{-1} e_i$ /* apply the recursion of Corollary 1 */
        else
            $e_i \leftarrow (\alpha_i - \alpha_{k+1})^{-1} e_i$
end .
```

Since each inner-loop iteration requires at most $2p$ additions over F, finding each e_i requires at most $2 \cdot \rho \cdot p$ additions once we have found σ_i. Totalling over all erasures, Step 4 requires $2 \cdot \rho^2 \cdot p$ additions over F, and the same order of index additions (modulo p) which, in turn, can be implemented by the scheme described in Appendix A.

The overall complexity of the erasure decoding algorithm thus sums up to $r \cdot p \cdot n$ additions over F for syndrome calculation (Step 1), and $O(\rho \cdot r)$ $\big(\le O(r^2)\big)$ operations (= additions and shifts) of p-vectors over F (Steps 2–4).

The above four-step procedure is demonstrated by way of example in Appendix B.

5 Single-error multiple-erasure decoding

The erasure decoding algorithm of Section 4 for $\mathcal{C}(p-1, n, r)$ can be adapted to cover also the case where a single column error, and up to $r-2$ column erasures, have occurred. The additional time complexity due to the single error amounts to a number of operations which is *linear* in p.

For the sake of clarity, assume first that no erasures have occurred. Therefore, the first two syndrome values are sufficient in order to correct the single column error (if any). Let $\Gamma \in C(p-1, n, r)$ over $F = GF(q)$ be the correct $(p-1) \times n$ code array, and let $Y = \Gamma + E$ be the received array (both over F), where E is an error array with at most one nonzero column, indexed j_0. Transforming into the linear code C over $\mathcal{R}_p(q)$ defined by (2), the corresponding error vector over $\mathcal{R}_p(q)$ is given by

$$\boldsymbol{\alpha} E \;=\; [\,\overset{\leftarrow\; j_0\; \rightarrow}{0\,0\,\ldots\,0}\;\, e_0\,0\,0\,\ldots\,0\,]\;,$$

where $e_0 = e_0(\alpha)$ is the error value at location j_0 and, as in Section 3, $\boldsymbol{\alpha}$ stands for the basis $[1\,\alpha\,\alpha^2\,\ldots\,\alpha^{p-2}]$ of $\mathcal{R}_p$ over F.

The two syndrome values $S_0 = S_0(\alpha)$ and $S_1 = S_1(\alpha)$ in $\mathcal{R}_p$, associated with $\mathbf{y} \stackrel{\Delta}{=} \boldsymbol{\alpha} Y$, are given by

$$[S_0\; S_1] = \mathbf{y} H^T = \boldsymbol{\alpha} E H^T = [e_0\; e_0 \alpha^{j_0}]\;.$$

That is, S_0 equals the error value e_0, whereas the error location j_0 is obtained by finding an integer k such that $S_1 = S_0 \alpha^k$. Now, as in Section 4, we transform the syndrome values into the ring modulo $x^p - 1$, resulting in the polynomials $\hat{S}_l(x) = \sum_{m=0}^{p-1} \hat{s}_{m,l} x^m$ whose coefficients (in F) are given by $\hat{s}_{m,l} = \sum_{j=0}^{n-1} y_{\langle m-jl \rangle_p}$, $0 \le m \le p-1$, $l = 0, 1$. In particular, $\hat{s}_{p-1,0} = \sum_{j=0}^{n-1} y_{p-1,j} = 0$ and, therefore, $\deg \hat{S}_0(x) < p-1$. On the other hand, since $\hat{S}_0(x) \equiv S_0(x) \pmod{M_p(x)}$, we must have $\hat{S}_0(x) = S_0(x) = e_0(x)$.

The second syndrome value $\hat{S}_1(x)$, in turn, satisfies the congruence

$$\hat{S}_1(x) \;\equiv\; x^k\, e_0(x) \;\equiv\; x^k\, \hat{S}_0(x) \pmod{x^p - 1} \tag{22}$$

for $k = j_0$; that is, the p coefficients of $\hat{S}_1(x)$ (in F) form a cyclically shifted phase of the coefficients of $\hat{S}_0(x)$. Note that $\hat{S}_0(x) = 0$ corresponds to the no-error case. Furthermore, since $\deg \hat{S}_0(x) < p-1$, these polynomials cannot equal any scalar multiple of $M_p(x)$, except the zero polynomial. We therefore conclude that, whenever a single error has occurred, there exists a *unique* cyclic shift k modulo p which translates the coefficients of $\hat{S}_0(x)$ into $\hat{S}_1(x)$. This shift, j_0, is the solution for k in (22) i.e., it points to the erroneous column.

Hence, the problem of finding the error location boils down to finding the relative cyclic shift between two phases of a sequence of period p. An algorithm due to Shiloach [22] solves this problem requiring a number of operations which is linear in p. For the sake of completeness, this algorithm is presented below. Given two sequences $a_0, a_1, \ldots, a_{N-1}$ and $b_0, b_1, \ldots, b_{N-1}$ over any ordered set, the following procedure finds two integers $l, m < N$ (if any) for which the vectors $[a_l \, a_{l+1} \, \ldots \, a_{N-1} \, a_0 \, \ldots \, a_{l-1}]$ and $[b_m \, b_{m+1} \, \ldots \, b_{N-1} \, b_0 \, \ldots \, b_{m-1}]$ are both equal (in case there are no such integers, the procedure terminates with $h = 0$; otherwise it will terminate with $h = N$).

```
$l \leftarrow 0$ ; $m \leftarrow 0$ ; $h \leftarrow 0$ ;
while $l < N$ and $m < N$ and $h < N$ do
    if $a_{\langle l+h \rangle_N} = b_{\langle m+h \rangle_N}$ then
        $h \leftarrow h + 1$
    else if $a_{\langle l+h \rangle_N} < b_{\langle m+h \rangle_N}$ then
            begin $m \leftarrow m + h + 1$ ; $h \leftarrow 0$ end
        else
            begin $l \leftarrow l + h + 1$ ; $h \leftarrow 0$ end .
```

Now, apply the above procedure to the coefficients of $\hat{S}_0(x)$ and $\hat{S}_1(x)$ with $N = p$. The desired value of j_0 is $\langle m - l \rangle_p$.

We now turn to the case where erasures have occurred as well. As we intend to apply our results also to the multiple-error case, we shall start the derivation assuming the most general setting where τ column errors and $\rho \leq r - 2\tau$ column erasures have occurred. Let $\mathbf{c} = [c_0 \, c_1 \, \ldots \, c_{n-1}] = \boldsymbol{\alpha}\Gamma$ denote the correct codeword (over $\mathcal{R}_p(q)$) and let $\mathbf{y} = [y_0 \, y_1 \, \ldots \, y_{n-1}] = \boldsymbol{\alpha}Y$ denote the received word, which is assumed to be (say) zero at the erasure locations $j_0, j_1, \ldots, j_{\rho-1}$. Also, let $j_\rho, j_{\rho+1}, \ldots, j_{\rho+\tau-1}$ denote the error locations, and for each i, $0 \leq i \leq \rho + \tau - 1$, let $e_i \triangleq y_{j_i} - c_{j_i}$ and $\alpha_i \triangleq \alpha^{j_i}$ stand for the ith error (or erasure) value and locator, respectively, both in $\mathcal{R}_p$. Without loss of generality we can assume that the j_i's are distinct for all $0 \leq i \leq \rho + \tau - 1$ and that the (proper) error values e_i, $\rho \leq i \leq \rho + \tau - 1$ are all nonzero. Recalling the definition of the syndrome as in (12), we have $S_l = \sum_{i=0}^{\rho+\tau-1} e_i \alpha_i^l$,

$0 \le l \le r-1$.

Theorem 2. *Let $S(z) = \sum_{l=0}^{r-1} S_l z^l$ be the syndrome polynomial and $R(z) = \prod_{s=0}^{\rho-1}(1 - \alpha_s z)$ be the erasure-locator polynomial (both over $\mathcal{R}_p$) associated with the received word $\mathbf{y}$. Also, let $Q(z) = \sum_{l=0}^{r+\rho-1} Q_l z^l$ denote the product $R(z)\,S(z)$ of these two polynomials. Then, for any polynomial $T(z) = \sum_{l=0}^{\tau'} T_l z^l$ of degree $\tau' \le r - \rho - \tau$ over $\mathcal{R}_p$,*

$$\textstyle\sum_{l=0}^{\tau'} T_l\, Q_{m-l} = 0 \;, \quad \rho + \tau' \le m \le r-1 \tag{23}$$

if and only if

$$e_i T(\alpha_i^{-1}) = 0 \;, \quad \rho \le i \le \rho + \tau - 1 \;. \tag{24}$$

In particular, (23) holds vacuously if $\tau = 0$.

Observe that when $2\tau + \rho \le r$, the error-locator polynomial $\Lambda(z) \overset{\Delta}{=} \sum_{l=0}^{\tau} \Lambda_l z^l = \prod_{s=\rho}^{\rho+\tau-1}(1 - \alpha_s z)$ satisfies (24) and, so,

$$\sum_{l=0}^{\tau} \Lambda_l Q_{m-l} = 0 \;, \quad \rho + \tau \le m \le r-1 \;.$$

Proof of Theorem 2. Let $P(z) = \sum_{l=0}^{\rho+\tau'} P_l z^l$ denote the product $T(z)\,R(z)$. The left-hand side of (23) is simply the coefficient of z^m in $T(z)\,Q(z) = T(z)\,R(z)\,S(z) = P(z)\,S(z)$ and, therefore,

$$\sum_{l=0}^{\tau'} T_l\, Q_{m-l} \;=\; \sum_{l=0}^{\rho+\tau'} P_l\, S_{m-l} \;, \quad \rho + \tau' \le m \le r-1 \;. \tag{25}$$

Expanding the right-hand side of (25), we obtain

$$\begin{aligned}
\sum_{l=0}^{\tau'} T_l\, Q_{m-l} &= \sum_{l=0}^{\rho+\tau'} P_l\, S_{m-l} \\
&= \sum_{l=0}^{\rho+\tau'} P_l \sum_{i=0}^{\rho+\tau-1} e_i \alpha_i^{m-l} \\
&= \sum_{i=0}^{\rho+\tau-1} e_i \alpha_i^{m} \sum_{l=0}^{\rho+\tau'} P_l\, \alpha_i^{-l} \\
&= \sum_{i=0}^{\rho+\tau-1} \alpha_i^{m} e_i P(\alpha_i^{-1}) \\
&= \sum_{i=0}^{\rho+\tau-1} \alpha_i^{m} e_i T(\alpha_i^{-1})\, R(\alpha_i^{-1}) \;, \quad \rho + \tau' \le m \le r-1 \;.
\end{aligned}$$

However, $R(\alpha_i^{-1}) = 0$ for $0 \le i \le \rho - 1$ and, so,

$$\sum_{l=0}^{\tau'} T_l \, Q_{m-l} \;=\; \sum_{i=\rho}^{\rho+\tau-1} \alpha_i^m e_i T(\alpha_i^{-1}) \, R(\alpha_i^{-1}) \;, \quad \rho + \tau' \le m \le r - 1 \;, \tag{26}$$

where the right-hand side of (26) should read zero if $\tau = 0$.

Let $V = [v_{m,i}]_{m,i}$ be the $(r - \rho - \tau') \times \tau$ matrix over $\mathcal{R}_p$ defined by $v_{m,i} \triangleq \alpha_i^m$, $\rho + \tau' \le m \le r - 1$, $\rho \le i \le \rho + \tau - 1$, and let $D = [d_{i,j}]_{i,j=\rho}^{\rho+\tau-1}$ be the $\tau \times \tau$ diagonal matrix over $\mathcal{R}_p$ whose principal diagonal is given by $d_{i,i} \triangleq R(\alpha_i^{-1})$, $\rho \le i \le \rho + \tau - 1$. Rewriting (26) in matrix form, the two column vectors

$$\mathbf{u} \;\triangleq\; \Big[\; \alpha_i^m e_i T(\alpha_i^{-1}) \; \Big]_{i=\rho}^{\rho+\tau-1}$$

and

$$\mathbf{v} \;\triangleq\; \Big[\; \sum_{l=0}^{\tau'} T_l \, Q_{m-l} \; \Big]_{m=\rho+\tau'}^{r-1}$$

are related by

$$\mathbf{v} \;=\; V D \, \mathbf{u} \, .$$

Recalling that the j_i's are distinct, the entries $d_{i,i} = R(\alpha_i^{-1})$ are all invertible in $\mathcal{R}_p$, and so D has an inverse over $\mathcal{R}_p$. In addition, the columns of V are linearly independent over $\mathcal{R}_p$ as long as $\tau \le r - \rho - \tau'$. It thus follows that, whenever $\tau' \le r - \rho - \tau$, $\mathbf{v} = \mathbf{0}$ if and only if $\mathbf{u} = \mathbf{0}$. This completes the proof of the theorem. □

We return now to the single-error multiple-erasure case i.e., $\tau \le 1$. Assuming that one error valued $e_\rho \ne 0$ has occurred at location j_ρ, the error-locator polynomial is given by $\Lambda(z) = 1 - \alpha^{j_\rho} z$. Hence, while searching for $\Lambda(z)$, we shall be looking for polynomials of the form $T_k(z) = 1 - \alpha^k z$, $0 \le k \le n-1$. Observe that, for every j, the value $T_k(\alpha^{-j})$ is invertible in $\mathcal{R}_p$ unless $j \equiv k \pmod{p}$, in which case $T_k(\alpha^{-j}) = 0$. In particular, $e_\rho T_k(\alpha^{-j_\rho}) = 0$ if and only if $k = j_\rho$. By Theorem 2 it thus follows that the unique solution k for

$$Q_{\rho+1} - \alpha^k Q_\rho \;=\; 0$$

is $k = j_\rho$. It is also easy to verify that the no-error case corresponds to $Q_\rho = Q_{\rho+1} = 0$. Note that in the no-erasure case ($\rho = 0$) discussed earlier, we have $R(z) = 1$, $Q(z) = R(z) \, S(z) = S(z)$ and, therefore, $Q_\rho = Q_0 = S_0$ and $Q_{\rho+1} = Q_1 = S_1$.

The single-error multiple-erasure decoding procedure is now easily derived. First, we compute the polynomials

$$\hat{S}(x;z) = \sum_{l=0}^{r-1} \hat{S}_l(x) z^l$$

and

$$\hat{Q}(x;z) = \sum_{l=0}^{r+\rho-1} \hat{Q}_l(x) z^l \;\equiv\; \left(\prod_{s=0}^{\rho-1} (1 - x^{j_s} z) \right) \hat{S}(x;z) \pmod{x^p - 1}$$

by the very same Step 1 and Step 2 of Section 4.

Next, we find the unique solution $k = j_\rho$ to the equation

$$\hat{Q}_{\rho+1}(x) \;\equiv\; x^k \hat{Q}_\rho(x) \pmod{M_p(x)} \; . \tag{27}$$

In fact, it can be shown that, as long as $\rho \le r-2$, the congruence in (27) holds not only modulo $M_p(x)$, but rather modulo $x^p - 1$. This has already been proved for $\rho = 0$ (Equation (22)); therefore, we can assume that $\rho > 0$. First, note that the values of the polynomials $\hat{S}_l(x) = \sum_{m=0}^{p-1} \hat{s}_{m,l} x^m$ at $x = 1$ are independent of l; this is due to the fact that $\sum_{m=0}^{p-1} \hat{s}_{m,l}$ equals the sum of the entries over F of the received array Y. Therefore,

$$\hat{S}(1;z) \left(= \hat{S}(x;z)|_{x=1} \right) \;=\; \sum_{l=0}^{r-1} \hat{S}_l(1) z^l \;=\; \hat{S}_0(1) \sum_{l=0}^{r-1} z^l \; .$$

Secondly, by the way $\hat{Q}(x;z)$ was defined, we have

$$\begin{aligned} \hat{Q}(1;z) &= \sum_{l=0}^{r+\rho-1} \hat{Q}_l(1) z^l \\ &= (1-z)^\rho \hat{S}(1;z) \\ &= \hat{S}_0(1)(1-z)^\rho \sum_{l=0}^{r-1} z^l \\ &= \hat{S}_0(1)(1-z)^{\rho-1}(z^r - 1) \; . \end{aligned}$$

Since we assume that $\rho + 1 < r$, we must have $\hat{Q}_\rho(1) = \hat{Q}_{\rho+1}(1) = 0$ which, by (27), yields the claimed congruence

$$\hat{Q}_{\rho+1}(x) \;\equiv\; x^k \hat{Q}_\rho(x) \pmod{x^p - 1} \; . \tag{28}$$

The above derivation also implies that the polynomials $\hat{Q}_\rho(x)$ and $\hat{Q}_{\rho+1}(x)$ cannot equal any scalar multiple of $M_p(x)$, except the zero polynomial (notice though the different reasoning for the no-erasure case as opposed to $\rho > 0$). Having these two polynomials equal to zero is an indication that no errors have occurred.

Now, as in the no-erasure case, we solve (28) for k using Shiloach's algorithm, yielding the location j_ρ of the single error (if any). Then, we multiply the polynomial $\hat{Q}(x; z)$ by the new-found factor $1 - x^{j_\rho}$, and continue with Step 3 and Step 4 of Section 4 as if there were $\rho + 1$ erasures.

6 The multiple-error multiple-erasure case

Finally, we turn to the multiple-error multiple-erasure case. Referring to Theorem 2, the linear constraints in (23) provide, on the one hand, *necessary* conditions for a given polynomial $T(z)$ to be an error-locator polynomial. On the other hand, by Theorem 2, these constraints are also *sufficient* for a polynomial $T(z) = 1 + \sum_{l=1}^{\tau'} T_l z^l$ of minimum degree $\tau' \le \tau \le r - \rho - \tau$ to be the error-locator polynomial *as long as the* e_i *are invertible in* $\mathcal{R}_p(q)$. Obviously, this is the case when $\mathcal{R}_p(q)$ is a field i.e., when q is primitive in $GF(p)$; and the Berlekamp-Massey algorithm provides in this case quite an efficient way to solve (23) for a polynomial $T(z) = 1 + \sum_{l\ge1} T_l z^l$ of smallest degree. Step 1 of the erasure decoding procedure of Section 4 can be used for fast syndrome calculation, whereas the rest of the steps can be used to compute the error-erasure values, once the error-locator polynomial has been found.

The following is a list of the primes $3 \le p \le 100$ (out of a total of 24) for which $\mathcal{R}_p(2)$ is a field: 3, 5, 11, 13, 19, 29, 37, 53, 59, 61, 67 and 83. Although it has not been established yet whether there exist infinitely many fields $\mathcal{R}_p(2)$, it is conjectured that the fraction of such fields among the rings $\mathcal{R}_p(2)$, $p \le N$, converges to a positive constant (approximately 0.374) when $N \to \infty$.

As for the case where $\mathcal{R}_p(q)$ is not a field, the zero divisors in $\mathcal{R}_p(q)$ do not allow a straightforward application of the Berlekamp-Massey algorithm to obtain the error-locator polynomial: in fact, we might as well have polynomials $T(z) = 1 + \sum_{l=1}^{\tau'} T_l z^l$ of degree $\tau' \le \tau \le r/2$ which satisfy (23), and yet $T(z) \ne \Lambda(z)$.

Example 3. Let $F = GF(2)$ and $p = 7$; in this case we have $M_7(\alpha) = (1 + \alpha^2 + \alpha^3)(1 +$

$\alpha+\alpha^3$). Now, assume that $r=4$ and that two errors (and no erasures) have occurred. The two error values are given by $e_0=1+\alpha^2+\alpha^3$ and $e_1=1+\alpha+\alpha^3$, whereas the error locators are $\alpha_0=1$ and $\alpha_1=\alpha^4$. It is easy to verify that the polynomial $T(z)=1+(\alpha+\alpha^3)z$ satisfies $e_0T(\alpha_0^{-1})=e_1T(\alpha_1^{-1})=0$ in $\mathcal{R}_7(2)$ and, therefore, it also satisfies (23). However, $T(z)\neq\Lambda(z)=1+(1+\alpha^4)z+\alpha^4z^2$. •

Since the e_i might be zero divisors in $\mathcal{R}_p(q)$ (as was the case in Example 3), some additional constraints should be imposed on the minimum-degree solutions $T(z)$ of (23) to yield $\Lambda(z)$. For example, we can search only for polynomials $T(z)$ whose values $T(\alpha^j)$, $0\le j\le p-1$, are all in $\mathcal{R}_p^*\cup\{0\}$ (that is, each of these values, if nonzero, is invertible in $\mathcal{R}_p$; note that $\Lambda(z)$ complies with such a requirement and that this is what we actually had in the single-error case). However, we do not know yet how to incorporate such a constraint into (23) to obtain an efficient decoding algorithm for an arbitrary number of errors.

Acknowledgment

The authors thank Henk van Tilborg for pointing out the existence of Shiloach's algorithm (and improvements thereof) used in Section 5.

Appendix A

In this appendix we suggest a VLSI architecture for implementing the recursion of Corollary 1. Figure 5 depicts a $p\times n$ array of cells, where each cell is capable of storing an element of F. In addition, each cell is equipped with a selector unit which has two data inputs (x and y), two outputs (u and v), and an input control bit s (see Figure 6). The selector has two modes of operation, according to the value of the input control s. When $s=0$, the selector is in *transparent mode*, in which case the input x is directed into u and the input y is directed into v. By setting $s=1$, the selector switches into *active mode*, where the input y is now directed into u and the sum $x+y$ (over F) is directed into v. In order to perform the recursion of Corollary 1, the selectors in columns m and l are set to active mode, whereas those in the other columns are transparent. An extra column at the far left of the array is

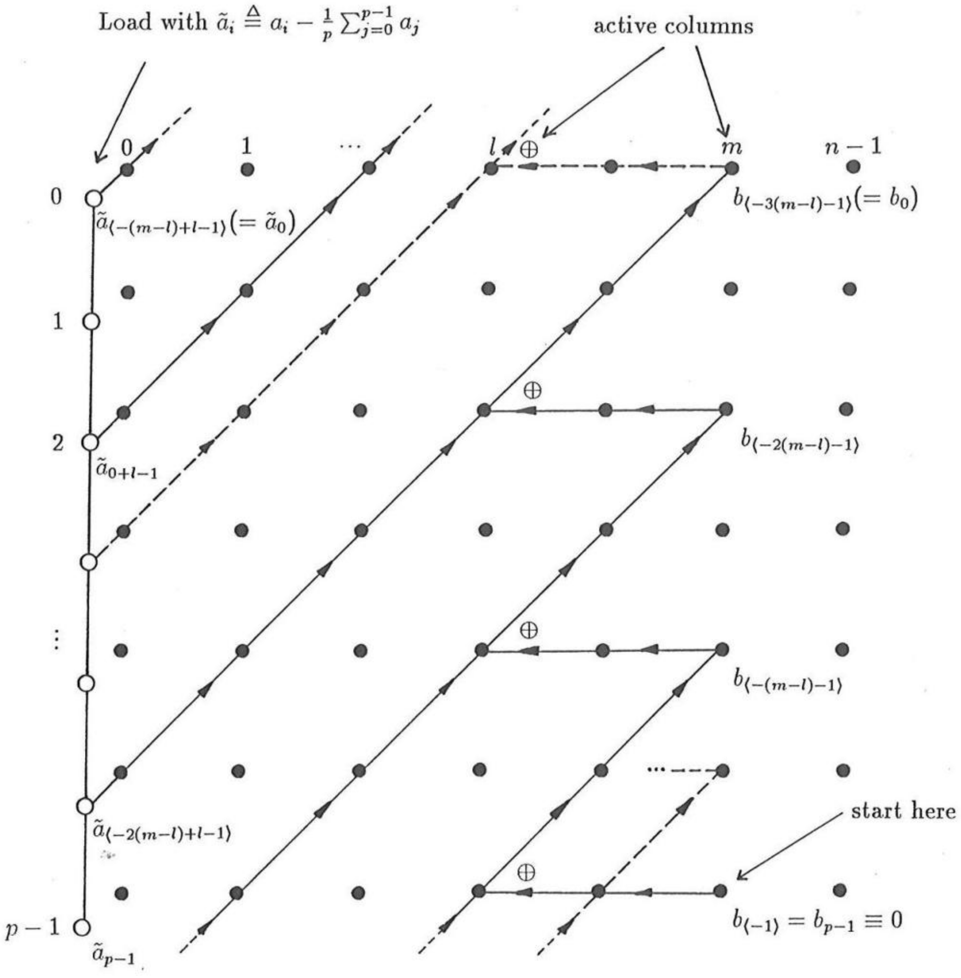
Load with $\tilde{a}_i \triangleq a_i - \frac{1}{p}\sum_{j=0}^{p-1} a_j$
active columns
0
1
$\cdots$
l
$\oplus$
m
$n-1$
0
$\tilde{a}_{\langle -(m-l)+l-1 \rangle} (= \tilde{a}_0)$
$b_{\langle -3(m-l)-1 \rangle} (= b_0)$
1
$\oplus$
2
$b_{\langle -2(m-l)-1 \rangle}$
$\tilde{a}_{0+l-1}$
$\vdots$
$\oplus$
$b_{\langle -(m-l)-1 \rangle}$
$\cdots$
$\tilde{a}_{\langle -2(m-l)+l-1 \rangle}$
start here
$\oplus$
$p-1$
$b_{\langle -1 \rangle} = b_{p-1} \equiv 0$
$\tilde{a}_{p-1}$

Figure 5: Implementation of the recursion of Corollary 1.

loaded with $\tilde{a}_i$, $i = 0, 1, \ldots, p-1$ (see Figure 5). Starting with zero at the last entry of the mth column, the correct values of b_i will eventually turn at the mth column in consecutive order.

Figure 7 depicts an implementation of the selector unit for the binary case ($F = GF(2)$).

Referring now to Step 4, observe that, while calculating the $e_i^{(k)}$, the active columns in Figure 5 correspond to column locations which contain erasures i.e., irrelevant information; furthermore, for each i, $e_i^{(k)}$ should all be computed into the same column i. Therefore, we can use the same piece of hardware both as the layout of Figure 5 and as the $(p-1) \times n$ storage memory for the received array.

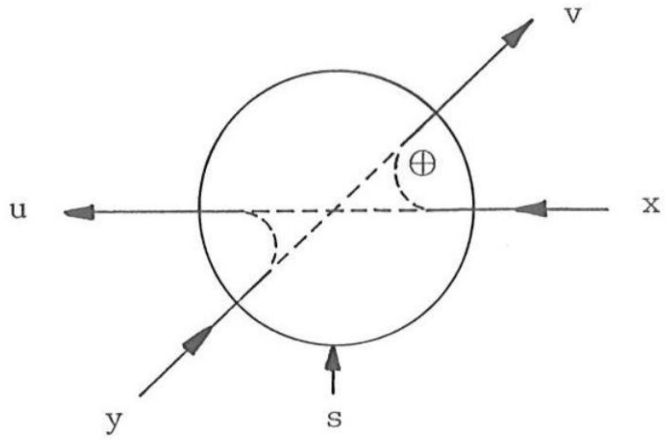

Figure 6: Selector unit input and output flow.

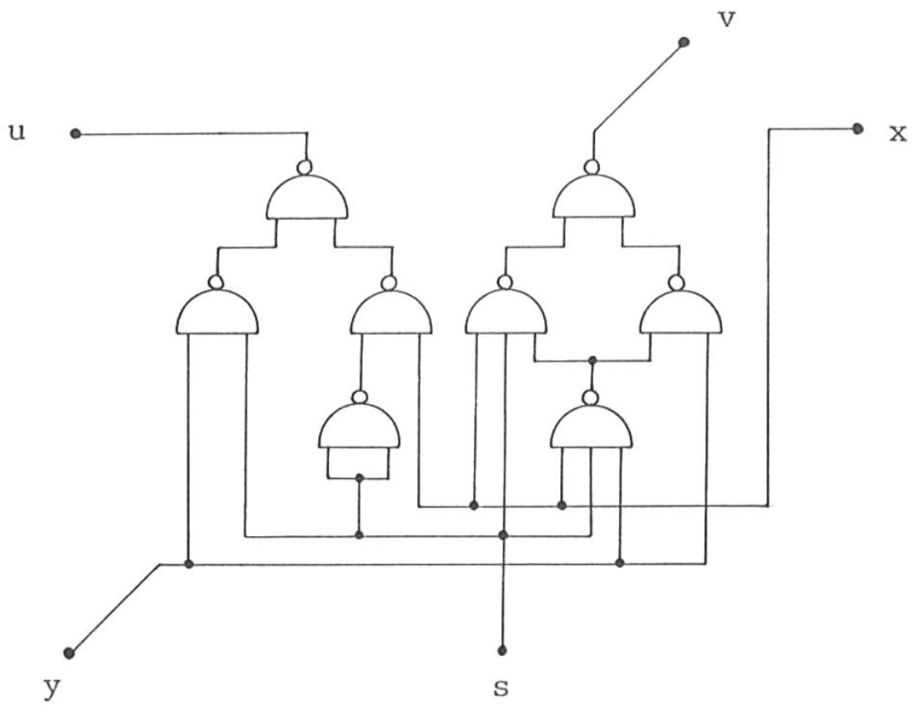

Figure 7: Implementation of a selector unit over $GF(2)$.

Appendix B

The next example illustrates the decoding procedure of Section 4 for $\mathcal{C}(4,5,3)$ over $GF(2)$. Assume that the received array is

	0	1	2	3	4
0	1	?	?	0	?
1	1	?	?	1	?
2	0	?	?	1	?
3	1	?	?	1	?

i.e., columns 1, 2, and 4, containing the values $e_0(\alpha)$, $e_1(\alpha)$, and $e_2(\alpha)$, respectively, have been erased. We reconstruct the values of $e_i(\alpha)$ by the four-step procedure of Section 4.

Step 1: Syndrome calculation. The coefficients of the polynomial $\hat{S}(x;z) = \hat{S}_0(x) + \hat{S}_1(x)z + \hat{S}_2(x)z^2$ (all in the polynomial ring $GF(2)[x]$ modulo $x^5 - 1$) are given by

$$\hat{S}_0(x) = 1 + x^2 \ , \quad \hat{S}_1(x) = x^3 + x^4 \ , \quad \text{and} \quad \hat{S}_2(x) = 1 + x + x^2 + x^4 \ ,$$

and can be calculated, respectively, following the parity lines depicted in Figures 1–3.

Step 2: Computation of $\hat{Q}(x;z)$. Noting that columns 1, 2, and 4 have been erased, the polynomial $\hat{Q}(x;z)$ is given by

$$\begin{aligned} \hat{Q}(x;z) \triangleq \sum_{l=0}^{5} \hat{Q}_l(x) z^l &= (1 - xz)(1 - x^2 z)(1 - x^4 z)\, \hat{S}(x;z) \\ &= (1 + x^2) + (x^2 + x^4)z + (1 + x + x^2 + x^3)z^2 \\ &\quad + (1 + x^4)z^3 + (x + x^2)z^4 + (x + x^2 + x^3 + x^4)z^5 \ . \end{aligned}$$

The execution of Step 2 involves three iterations, each consisting of a linear shift of the current five-bit coefficients $\hat{Q}_l(x)$ by one five-bit position, then a cyclic bit-shift of each coefficient $\hat{Q}_l(x)$, and finally XOR-ing the newly computed copies of $\hat{Q}_l(x)$ with the original ones.

Step 3: Computation of $\sigma_0, \sigma_1, \sigma_2$ (the right-hand side of (17)). Given the values of $\hat{Q}_l(x)$ of Step 2, the value of $\sigma_0 = \sigma_0(\alpha)$ is found by first computing iteratively the binary coefficients of $\hat{\sigma}_0(x)$ as follows:

$$\hat{\sigma}_0(x) \ \leftarrow \ \hat{Q}_0(x) = 1 + x^2 \ ;$$

$$\begin{aligned}\hat{\sigma}_0(x) &\leftarrow \hat{Q}_1(x) + x\,\hat{\sigma}_0(x) = (x^2+x^4) + x\,(1+x^2) = x + x^2 + x^3 + x^4\ ;\\ \hat{\sigma}_0(x) &\leftarrow \hat{Q}_2(x) + x\,\hat{\sigma}_0(x) = (1+x+x^2+x^3) + x\,(x+x^2+x^3+x^4) = x + x^4\ ;\end{aligned}$$

all computation taken modulo $x^5 - 1$. Taking now the resulting $\hat{\sigma}_0(x)$ modulo $M_5(x)$, we obtain

$$\sigma_0(\alpha) \;=\; 1 + \alpha^2 + \alpha^3\ .$$

Computing σ_1 and σ_2 in a similar manner, we end up with

$$\sigma_1(\alpha) \;=\; 1 + \alpha + \alpha^2 + \alpha^3 \quad \text{and} \quad \sigma_2(\alpha) \;=\; \alpha^2 + \alpha^3\ .$$

Step 4: Extracting the erasure values e_i from (17). It remains to solve the following equations

$$\begin{aligned}(\alpha-\alpha^2)(\alpha-\alpha^4)\,e_0(\alpha) &= \sigma_0(\alpha)\\ (\alpha^2-\alpha)(\alpha^2-\alpha^4)\,e_1(\alpha) &= \sigma_1(\alpha)\\ (\alpha^4-\alpha)(\alpha^4-\alpha^2)\,e_2(\alpha) &= \sigma_2(\alpha)\end{aligned}$$

for e_0, e_1, and e_2.

Starting with e_0, we define the intermediate values $e_0^{(k)}$, $k = 2, 1, 0$, as in (21) i.e., $e_0^{(2)} \triangleq e_0$, $e_0^{(1)} \triangleq (\alpha - \alpha^4)\,e_0^{(2)}$, and $e_0^{(0)} \triangleq (\alpha - \alpha^2)\,e_0^{(1)}$. We calculate $e_0^{(k)}$ iteratively for $k = 0, 1, 2$ using the initial condition $e_0^{(0)} = \sigma_0$, where σ_0 is as computed in Step 3. Hence, $e_0^{(1)}$ is obtained by solving

$$(\alpha-\alpha^2)\,e_0^{(1)} = 1 + \alpha^2 + \alpha^3 \quad (=\ \sigma_0)\ ,$$

which, in turn, can be computed using the recursion of Corollary 1 (or Corollary 2), resulting in

$$e_0^{(1)} \;=\; 1 + \alpha + \alpha^2$$

(see Example 2). Having computed $e_0^{(1)}$, the value of $e_0 = e_0^{(2)}$ is found by solving the recursion of Corollary 1 that corresponds to

$$(\alpha-\alpha^4)\,e_0^{(2)} = 1 + \alpha + \alpha^2 \quad (=\ e_0^{(1)})\ .$$

This yields $e_0 = 1 + \alpha^2$.

The values of e_1 and e_2 are found in a similar manner, ending up with $e_1 = e_2 = \alpha^2$. The correct code array is, therefore, the one given in Figure 4.

References

[1] R.E. Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley, Reading, Massachusetts, 1983.

[2] I.F. Blake, *Codes over certain rings, Inform. Control,* 20 (1972), 396–404.

[3] I.F. Blake, *Codes over certain residue rings, Inform. Control,* 29 (1975), 295–300.

[4] M. Blaum, *A class of byte-correcting array codes,* IBM Research Report RJ 5652 (57151), 1987.

[5] M. Blaum, P.G. Farrell, H.C.A. van Tilborg, *A class of burst error-correcting array codes, IEEE Trans. Inform. Theory,* IT-32 (1986), 836–839.

[6] M. Blaum, P.G. Farrell, H.C.A. van Tilborg, *Multiple burst-correcting array codes, IEEE Trans. Inform. Theory,* IT-34 (1988), 1061–1066.

[7] M. Blaum, H. Hao, R. Mattson, J. Menon, *A coding technique for double disk failures in disk arrays,* IBM Disclosure SA889-0443 (submitted to U.S. Patent Office).

[8] P.G. Farrell, *A survey of array error control codes,* preprint, 1990.

[9] P.G. Farrell, S.J. Hopkins, *Burst-error-correcting array codes, Radio Elec. Engr.,* 52 (1982), 188–192.

[10] T. Fuja, C. Heegard, M. Blaum, *Cross parity check convolutional codes, IEEE Trans. Inform. Theory,* IT-35 (1989), 1264–1276.

[11] G. Gibson, L. Hellerstein, R.M. Karp, R.H. Katz, D.A. Patterson, *Coding techniques for handling failures in large disk arrays,* Report No. UCB/CSD 88/477, 1988.

[12] R. Goodman, M. Sayano, *Size limits on phased burst error correcting array codes, Elec. Lett.,* 26 (1990), 55–56.

[13] R. Goodman, M. Sayano, *Phased burst correcting array codes,* preprint, 1990.

[14] G.R. Lomp, D.L. Schilling, *A new burst and random error correcting code: The projection code,* preprint, 1990.

[15] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes,* North Holland, Amsterdam, 1977.

[16] S.W. Ng, *Some design issues of disk arrays,* IBM Research Report, RJ 6590 (63550), 1988.

[17] A.M. Patel, *Adaptive cross parity code for a high density magnetic tape subsystem, IBM J. Res. Develop.,* 29 (1985), 546–562.

[18] J.A. Reeds, N.J.A. Sloane, *Shift-register synthesis (modulo m), Siam J. Comput.,* 14 (1985), 505–513.

[19] D.L. Schilling, D. Manela, *PASM and TASM forward error correction and detection code: method and apparatus,* U.S. Patent 4,849,974 (July 1989).

[20] A. Schönhage, *Schnelle Multiplikation von Polynomen über Körpern der Characteristick 2, Acta Informatica,* 7 (1977), 395–398.

[21] P. Shankar, *On BCH codes over arbitrary integer rings, IEEE Trans. Inform. Theory,* IT-25 (1979), 480–483.

[22] Y. Shiloach, *A fast equivalence-checking algorithm for circular lists, Inform. Proc. Lett.,* 8 (1979), 236–238.

[23] E. Spiegel *Codes over Z_m, Inform. Control,* 35 (1977), 48–51.

[24] E. Spiegel *Codes over Z_m, revisited Inform. Control,* 37 (1978), 100–104.

[25] W. Zhang, J.K. Wolf, *A class of binary burst error-correcting quasi-cyclic codes, IEEE Trans. Inform. Theory,* IT-34 (1988), 463–480.