

เทคนิคการแก้สัญลักษณ์ที่ถูกลบในการถอดรหัสรีด-โซโลมอน

ERASURES CORRECTION TECHNIQUES IN R-S DECODING

Dr. Somsak Choomchuay^{†*}

ABSTRACT

It is known that the Berlekamp-Massey algorithm can be modified to incorporate error-erasure locator polynomial, but neither efficient nor well-structured modification is yet exploited. This paper proposes structural and efficient techniques to initialise the error evaluator or Forney polynomial in the Berlekamp-Massey algorithm in both frequency and time domain. Such techniques enable the procedure to have the features of erasures and errors correctability. For the validation proof, proposed algorithms are verified using high-level language simulation. In addition, their hardware implementations are also suggested and discussed.

Key Words : Erasures correction, R-S Code, Error control code

1. Introduction

Error correction codes nowadays play many important roles in the areas of communication and data storage system. In many cases that signal to noise ratio becomes very low, for example deep-space communication, not only coded symbols are corrupted but some are also erased. Those known undetermined-code locations but not magnitude are termed as erasures. In such circumstances, the decoder can be made more useful if both errors and erasures can be corrected.

To compute the error locator polynomial in an efficient way, both Euclid's algorithm and Berlekamp-Massey algorithm can be used. In the mixed domain decoder, Forney polynomial is needed to evaluate errors' magnitude. This reflects the need of the evaluator polynomial. Shao and Reed [1] have proposed a modified Euclid's algorithm which can generate both

error locator and error evaluator polynomials at the same time. Their decoder also has the capability of erasures correction.

Alternatively, Berlekamp's technique can also generate both polynomials [2]. However, when erasures are involved, the procedure becomes more intricate. Blahut [3] has shown the way to obtain the error-erasure locator polynomial when erasures are considered. This is done simply by initialising the error locator polynomial with the erasure locator polynomial. He also showed that the error evaluator polynomial and the derivative of the error locator polynomial can be iterated within those $2t$ iterations [4].

The time domain decoder has been studied extensively by Choomchuay and Arambepola [5]. The decoding is carried in the same domain as the received data, i.e. without syndrome computing. This introduces a rather intensive computation since the decoder has to operate on the whole block length rather than a $2t$

[†] School of Graduate Studies, KMITL

^{*} Dept. of Electronics, Faculty of Engineering

syndrome component. Similar to the frequency domain technique, the decoder can be modified to cope with erasures [4]. Problems still persist. There is neither clear explanation nor efficient procedure used to attack erasure problem when the Berlekamp-Massey technique is employed as a warship. Neither frequency nor time domain environment is made exposed.

The ways for obtaining the right information from the code which accompanies both error and erasure are outlined in the paper. Both frequency domain and time domain technique approaches are elaborated. The clarified definitions of the time domain and the frequency domain encoding and decoding techniques can be found in [5]. Verification of algorithms is carried out using high-level language simulation. Its computational structures in both software and hardware aspects are realised.

The technique for decoding both errors and erasures in the frequency domain environment is briefed in section 2. A modified frequency domain algorithm is proposed as the algorithm \mathcal{A}_1 . Subsequently, the time domain technique is outlined in section 3 and the corresponding algorithm \mathcal{A}_2 is detailed. Architectures for both techniques are discussed in section 4 before the paper is concluded in the subsequent section.

2. The Frequency Domain Technique

Let the $R-S(N,k)$ be the code with block length $N = 2^m - 1$ and has the error correctability of $t = (N-k)/2$. Let v be the number of occurred errors and ρ be the number of prompted erasures. The decoder can give the correct information as long as $2v + \rho \leq 2t$.

Let the error locator polynomial $\Lambda(x)$ be defined as

$$\Lambda(x) = \prod_{k=1}^v (1 - x\alpha^{i_k}), \quad (1)$$

where i_k denotes the error location.

Also let $S(x)$ be the syndrome polynomial defined as

$$S(x) = \sum_{i=1}^{2t} S_i x^i. \quad (2)$$

The error evaluator polynomial, $\Omega(x)$ can be obtained from

$$\Omega(x) = \Lambda(x)\{1 + S(x)\} \bmod x^{2t}. \quad (3)$$

Roots of the error locator polynomial which define error locations can be computed using the Chien searching technique. Once the root is obtained, the corresponding error magnitude can be evaluated as

$$e_i = \frac{\Omega(\alpha^{-i})}{\alpha^{-i} \Lambda'(\alpha^{-i})}, \quad (4)$$

where α^{-i} is a root of $\Lambda(x)$. The correct coded-symbol can then be computed from

$$c_i = v_i + e_i.$$

Several methods can be used for computing the error locator polynomial from the known syndromes. The Berlekamp-Massey shift register synthesis algorithm [2] is the one that can be considered as an efficient technique. The error evaluator polynomial and the derivative of the error locator polynomial can be computed. Such a procedure is first mentioned by Blahut [3,4]. However, in the case of error correction only such an inclusion can be relatively simple since both $\Omega(x)$ and $\Lambda(x)$ can be initialised with 1. In the case that erasures are to be considered, the algorithm has to be modified.

An erasure can be viewed as a known-location error. Its polynomial can be given as

$$\psi(x) = \prod_{j=1}^{\rho} (1 - x\alpha^{i_j}), \quad (5)$$

where α^{i_j} denotes the erasure location. Obviously, roots of this polynomial define the erasure locations. The presence of erasures can affect the computed syndromes, since the syndromes now are not only the trace of errors but also the erasures. Blahut has shown that these erasure-included syndromes or modified syndromes can efficiently be used in the Berlekamp-Massey algorithm for obtaining the modified error locator polynomial which is later called the *modified error locator polynomial*, $\bar{\Lambda}(x) = \Lambda(x)\psi(x)$. Subsequently, errors and erasures magnitude can be computed using the equation (4) given above. Note that the error

evaluator polynomial can be now stated precisely as the *errors and erasures evaluator polynomial*.

In this paper, a slightly different approach compared to Blahut's [3] is detailed. The Forney polynomial is used rather than the error evaluator polynomial.

The modified Berlekamp-Massey algorithm which has been multiplied through by $\Psi(x)$ can be written as:

$$\begin{bmatrix} \Lambda^{(r)}(x)\Psi(x) \\ \mathbf{B}^{(r)}(x)\Psi(x) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1} x \\ \delta_r & (1-\delta_r)x \end{bmatrix} \begin{bmatrix} \Lambda^{(r-1)}(x)\Psi(x) \\ \mathbf{B}^{(r-1)}(x)\Psi(x) \end{bmatrix} \quad (6)$$

$\Lambda^{(r)}(x)\psi(x)$ and $\mathbf{B}^{(r)}(x)\psi(x)$ can then be redefined as $\bar{\Lambda}(x)$ and $\bar{\mathbf{B}}(x)$ respectively. With this redefined term, the discrepancy is computed as follows.

$$\begin{aligned} \Delta_r &= \sum_{j=0}^{N-1} S_j \bar{\Lambda}_{r-j}^{(r-1)} \\ &= \sum_{k=0}^{N-1} \Lambda_k^{(r-1)} S'_{r-k} \end{aligned} \quad (7)$$

where $S'(x) = S(x)\Psi(x) \bmod x^{2t}$. $S'(x)$ denotes the modified syndrome and $S(x) = \sum_{i=1}^{2t} S_i x^i$.

From equation (6), if the Berlekamp-Massey algorithm is initialised with $\Lambda^{(0)}(x) = \mathbf{B}^{(0)}(x) = \psi(x)$, then it can compute $\bar{\Lambda}(x)$ which can be defined as the error-erasure locator polynomial. The modified Forney polynomial $\bar{\Gamma}(x) = \bar{\Lambda}(x)S(x) \bmod x^{2t}$ can also be computed subsequently outside the iterative procedure. However, the inclusion of the computation of this polynomial in the Berlekamp-Massey algorithm may give the advantage that the extra polynomial multiplication block (for computing $\bar{\Gamma}(x)$) can be omitted with few extra costs. Dropping those modified polynomials notations, the algorithm for obtaining the error-erasure locator and the modified error evaluator polynomial can be written as the algorithm @1 given below.

Algorithm @1

Initialise :

$$\begin{aligned} \Lambda^{(0)}(x) &= \mathbf{B}^{(0)}(x) = \Psi(x); \\ \Gamma^{(0)}(x) &= \mathbf{M}^{(0)}(x) = S(x); \quad K_\rho = 1, \quad L_\rho = \rho. \end{aligned}$$

for $r \leq \rho$,

$$\Delta_r = \alpha^i$$

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) - \Delta_r x \mathbf{B}^{(r-1)}(x);$$

$$\mathbf{B}^{(r)}(x) = \Lambda^{(r)}(x);$$

$$\Gamma^{(r)}(x) = \Gamma^{(r-1)}(x) - \Delta_r x \mathbf{M}^{(r-1)}(x);$$

$$\mathbf{M}^{(r)}(x) = \Gamma^{(r)}(x);$$

for $r \leq 2t$,

$$\Delta_r = \sum_{i=0}^{2t} \Lambda_i^{(r-1)} S_{r-i} \quad (8a)$$

$$L_r = \delta_r (r - L_{r-1} + \rho) + (1 - \delta_r) L_{r-1}$$

$$K_r = \delta_r \Delta_r^{-1} + (1 - \delta_r) K_{r-1}$$

$$\begin{bmatrix} \Lambda^{(r)}(x) \\ \mathbf{B}^{(r)}(x) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1} x \\ \delta_r & (1 - \delta_r)x \end{bmatrix} \begin{bmatrix} \Lambda^{(r-1)}(x) \\ \mathbf{B}^{(r-1)}(x) \end{bmatrix} \quad (8b)$$

$$\begin{bmatrix} \Gamma^{(r)}(x) \\ \mathbf{M}^{(r)}(x) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1} x \\ \delta_r & (1 - \delta_r)x \end{bmatrix} \begin{bmatrix} \Gamma^{(r-1)}(x) \\ \mathbf{M}^{(r-1)}(x) \end{bmatrix} \quad (8c)$$

Here $\delta_r = 1$ if both $\Delta_r \neq 0$ and $2L_{r-1} \leq r - 1 + \rho$; otherwise $\delta_r = 0$.

Iterations are carried out for $r = \rho + 1, \rho + 2, \dots, 2t$.

The error magnitude is computed as

$$e_i = \frac{\Gamma^{(2t)}(\alpha^{-i})}{\Lambda^{(2t)}(\alpha^{-i})} \quad (8d)$$

where α^{-i} is a root of $\Lambda^{(2t)}(x)$.

Similar to the conventional case where erasure is not considered, $\Lambda(x)$ and $\mathbf{B}(x)$ are first initialised with 1, but $\Gamma(x)$ and $\mathbf{M}(x)$ are initialised with $S(x)$ instead. During the first ρ

iterations the algorithm implements $\psi(x) = (1 - x\alpha^{i_1})(1 - x\alpha^{i_2}) \cdots (1 - x\alpha^{i_\rho})$ and $\Psi(x)S(x) \bmod x^{2t}$ by setting $\Delta_r = \alpha^{i_j}$ where i_j denotes the j^{th} erasure location. When the ρ^{th} iteration is completed $\Lambda(x) = B(x) = \Psi(x)$ and $\Gamma(x) = M(x) = \Psi(x)S(x) \bmod x^{2t}$ and the register length is set to ρ . The algorithm then proceeds toward $2t$ iterations to compute $\Lambda(x)$ and $\Gamma(x)$.

It should be noted that $\Lambda(x)$ is now an error-erasure locator polynomial whose roots define the locations of errors and erasures. $\Gamma(x)$ takes the form of an error-erasure evaluator polynomial from which the error-erasure magnitudes can be computed. $B(x)$ and $M(x)$ are the auxiliary polynomials of $\Lambda(x)$ and $\Gamma(x)$ respectively. Obviously, the maximum degree of $\Lambda(x)$ or $\Gamma(x)$ is $2t$ rather than t . The synthesised filter is, hence, of length $t+\rho$ which is assumed to be at most $2t$.

3. Time Domain Techniques

The time domain Reed-Solomon decoding approaches was early proposed by Blahut [4]. More details and in-dept algorithms were later studied by Choomchuay [7]. However, those are until with the absence of erasure decoding consideration. This section is devoted to errors and erasures decoding in such an environment.

With the concept of the finite field transformation the role of frequency domain erasure can also be viewed similarly in the time domain. In the frequency domain environment, the coefficients of the syndrome polynomial are $2t$ consecutive DFT components of the received sequence starting at the frequency component t_0 . To obtain the time domain version of the syndrome, we consider syndrome as the DFT of the time domain data sequence multiplied by a relatively simple rectangular window function which comprises of 1 for $i = t_0, t_0+1, \dots, t_0+2t-1$ and zero elsewhere. The windowing operation is actually a pointwise multiplication which is mapped to the cyclic convolution between the raw received data and the time domain version of the window function.

S_1 is the coefficient of degree 0 of the syndrome polynomial. This is effectively the d.c. syndrome. However, in the DFT representation this occupies the t_0^{th} frequency location. Therefore the spectral sequence has to be left shifted by t_0 places. Each left shift corresponds to multiplying the time domain sequence by α^i for $i = 0, \dots, N-1$.

Alternatively, one can perform shift before windowing in order to have a window sequence that 1 start from the beginning of the sequence. Correspondingly, in the time domain, the t_0 place left shifting is first applied to the received sequence and followed by windowing.

Let the t_0 place left shifted of the sequence $\{v_i\}$ be defined by $\{\bar{v}_i\}$ as

$$\bar{v}_j = \alpha^{it_0} v_i, \quad \text{for } j=0 \text{ to } N-1. \quad (9)$$

And let $\{\bar{w}_i\}$ be the IDFT of the sequence $\{w_i\}$
 $= \{ \overbrace{1, 1, \dots, 1}^{2t}, 0, 0, \dots, 0 \}$ of length N which has 1 for $2t$ consecutive places.

The time domain syndrome sequence, $\{s_j\}$, is then :

$$\{s_j\} = \{\bar{v}_i\} \otimes \{\bar{w}_i\}, \quad (10)$$

where \otimes denotes convolution operation.

Thus,

$$s_j = \alpha^{jt_0} \sum_{i=0}^{N-1} v_i \bar{w}_{<j-i>_N} \quad \text{for } 0 \leq j \leq N-1. \quad (11)$$

In obtaining the time domain Forney sequence, recalled that the frequency domain version is defined such that $T(x) = S(x)\Lambda(x) \bmod x^{2t}$. Let the polynomial $S(x)$ and $\Lambda(x)$ be extended to the length N by zero padding. There are two operations involved in the Forney polynomial computation. The first is the convolution between $\{S_i\}$ and $\{\Lambda_i\}$ which corresponds to $S(x)\Lambda(x)$. The operation is the truncation of the convolution result according to modulo x^{2t} operation. The convolution operation in the frequency domain can be easily mapped into pointwise multiplications in the time domain. Let $\{g_j\}$ be the inverse transformed sequence of $S(x)\Lambda(x)$, then

$$g_j = s_j \lambda_j, \text{ for } 0 \leq k \leq N-1.$$

Applying window to the sequence $\{g_j\}$, the time domain error evaluator sequence $\{\gamma_k\}$ is then computed from :

$$\begin{aligned} \gamma_k &= \sum_{j=0}^{N-1} g_j w_{\langle k-j \rangle_N} \\ &= \sum_{j=0}^{N-1} \{ \lambda_j \alpha^{j t_0} \sum_{i=0}^{N-1} v_i w_{\langle j-i \rangle_N} \} w_{\langle k-j \rangle_N} \\ &= \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \alpha^{j t_0} \lambda_j v_i w_{\langle j-i \rangle_N} w_{\langle k-j \rangle_N}, \end{aligned} \quad (12)$$

for $0 \leq k \leq N-1$.

Note that the window sequence given in equation (12) is the same as the one given in equation (11).

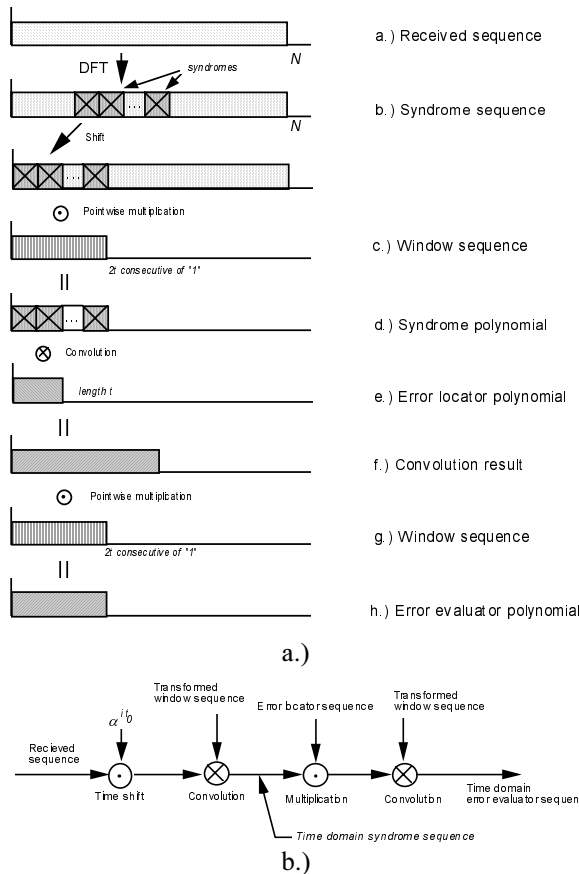


Figure 1 Operations in obtaining the error locator polynomial/sequence when erasures are concerned:

- a.) frequency domain environment
- b.) time domain environment.

The time domain operation depicted in Figure 1b. does not involve the DFT operation whilst the frequency domain operation does. However, in the frequency domain, the DFT operation is required to evaluate only $2t$ syndrome values. There are two circular convolution operations in the time domain computation. Each operation takes $O(N^2)$ multiplications. Both are the effects of the window operation in the frequency domain. The convolution in the frequency domain requires fewer multiplications since both syndrome polynomial and error locator polynomial are much shorter than N . Even though the frequency domain computation requires more operation steps, it seems to give better computing efficiency because the data are only at most of length $2t$. Moreover, a shift operation in the frequency domain does not require multiplication while that in the time domain does.

In the frequency domain, the inclusion of erasures in the Berlekamp-Massey procedure is similar to the conventional case where erasures are not considered. However, instead of 1 both the erasure-error locator and its auxiliary polynomial are initialised with the erasure polynomial. The error evaluator polynomial has to be initialised with the contents of the modified syndrome polynomial which is $S(x) \Lambda(x) \bmod x^{2t}$ or $S(x) \Psi(x) \bmod x^{2t}$ rather than with 1. Coincidentally, in the time domain, these sequences are initialised with their corresponding frequency domain versions, i.e. the erasure-error locator sequence, is initialised with a time domain erasure sequence and the error evaluator sequence is initialised with a modified time domain syndrome sequence as given in equation (11).

The effect of truncation, however, has to be refined during the iteration process. In the frequency domain, the truncation can be done automatically because the polynomial length is limited. This cannot be achieved in the time domain procedure. To achieve the correct result, the error evaluator sequence has to be refined by taking circular convolution with $\{w_i\}$ in every iteration. This might not be suitable for hardware implementation because

of the large number of non-trivial operations which leads to high computation counts. Further modification to the algorithm may thus be required.

After $2t$ iterations, the error evaluator sequence is obtained. The computation of the error magnitude is carried out subsequently. The above decoding algorithm is given below as algorithm $\mathcal{Q}2$. It should be noted that an extra computation step is needed to perform the convolution so as to refine the error evaluator sequence. The time domain erasure sequence $\{\psi_i\}$ given in the algorithm is the IDFT of its corresponding frequency domain polynomial $\{\Psi_k\}$. The derivative of the error locator sequence can be computed using the convolution method realised from the frequency domain pointwise multiplication. This is given as

$$\lambda'_i = \alpha^{i t_0} \sum_{k=0}^{N-1} h_k \lambda_{\langle i-k \rangle_N} \quad (13)$$

$$\text{where } h_k = \alpha^{-k} + \alpha^{-3k} + \alpha^{-5k} + \dots + \alpha^{-(N-3)k}$$

Algorithm $\mathcal{Q}2$

Initialisation :

$$\lambda_i^{(\rho)} = \beta_i^{(\rho)} = \psi_i; \quad L_\rho = \rho; \quad K_\rho = U_\rho = 1;$$

$$\gamma_i^{(\rho)} = \mu_i^{(\rho)} = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \psi_j \alpha^{j t_0} v_k w_{\langle j-k \rangle_N} w_{\langle i-j \rangle_N};$$

For $i=0, 1, \dots, N-1$.

$$\Delta_r = \sum_{i=0}^{N-1} v_i \lambda_i^{(r-1)} \alpha^{i(r+t_0-1)}$$

$$L_r = \delta_r (r + \rho - L_{r-1}) (1 - \delta_r) L_{r-1}$$

$$K_r = \delta_r \Delta_r^{-1} + (1 - \delta_r) K_{r-1}$$

$$U_r = \delta_r + (1 - \delta_r) (U_{r-1} + 1),$$

$$\begin{bmatrix} \lambda_i^{(r)} \\ \beta_i^{(r)} \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1} \alpha^{-i U_{r-1}} \\ \delta_r & (1 - \delta_r) \end{bmatrix} \begin{bmatrix} \lambda_i^{(r-1)} \\ \beta_i^{(r-1)} \end{bmatrix} \quad (14a)$$

$$\begin{bmatrix} \gamma_i^{(r)} \\ \mu_i^{(r)} \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1} \alpha^{-i U_{r-1}} \\ \delta_r & (1 - \delta_r) \end{bmatrix} \begin{bmatrix} \gamma_i^{(r-1)} \\ \mu_i^{(r-1)} \end{bmatrix} \quad (14b)$$

$$\{\gamma_i^{(r)}\} = \{\gamma_i^{(r)}\} \otimes \{w_i\} \quad (14c)$$

for $r = \rho+1, \rho+2, \dots, 2t$.

Here $\delta_r = 1$ if both $\Delta_r \neq 0$ and $2L_{r-1} \leq r - 1 + \rho$; otherwise $\delta_r = 0$.

The error magnitude is then computed from $e_i = \frac{\gamma_i^{(2t)}}{\lambda_i^{(2t)}}$, wherever $\lambda_i^{(2t)} \neq 0$.

The above algorithm is summarised as given in the computing steps shown in Figure 2 below.

1. Compute the modified time domain error evaluator sequence

$$\gamma_i = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \psi_j \alpha^{j t_0} v_k w_{\langle j-k \rangle_N} \quad \text{for } 0 \leq i \leq N-1$$

$$\text{where } \{w_i\} = \text{IDFT} \left\{ \underbrace{1, 1, \dots, 1, 0, 0, \dots, 0}_{N-1} \right\}$$

2. Compute the time domain erasure locator sequence

$$\{\psi_i\} = \text{IDFT} \{\Psi_k\}$$

3. Initialise the BMA algorithm as

$$r = \rho + 1, \quad L_{r-1} = \rho, \quad K_{r-1} = 1, \quad U_{r-1} = 1;$$

$$\{\lambda_i\} = \{\beta_i\} = \{\psi_i\}$$

$$\{\gamma_i\} = \{\mu_i\} = \{\alpha_i\}$$

4. Perform the BMA for the remain $2t-\rho$ iterations

5. Compute the error magnitude at the location where $\lambda_i = 0$,

$$\text{using } e_i = \frac{\gamma_i^{(2t)}}{\lambda_i^{(2t)}}$$

$$\text{where } \lambda'_i = \alpha^{i t_0} \sum_{k=0}^{N-1} h_k \lambda_{\langle i-k \rangle_N}$$

$$h_k = \alpha^{-k} + \alpha^{-3k} + \alpha^{-5k} + \dots + \alpha^{-(N-3)k}$$

Figure 2 Summarised computing steps of the algorithm $\mathcal{Q}2$

Note that algorithm $\mathcal{Q}2$ stated above can also start from zero iteration count. In such a case the initialised condition is slightly different as $\lambda_i^{(0)} = \beta_i^{(0)} = 1; \quad L_0 = 0; \quad K_\rho = U_\rho = 1;$

$\gamma_i^{(0)} = \mu_i^{(0)} = \alpha^{j t_0} \sum_{i=0}^{N-1} v_i w_{\langle j-i \rangle_N}$. Essentially, the error evaluator and its auxiliary sequences are initialised with the time domain syndrome sequence. For the first ρ iterations the algorithm implements the initial condition of algorithm $\mathcal{Q}2$ as given in equation (12). The discrepancy of each iteration is given by $\Delta_r = \alpha^{i k}$ where $i k$ denotes the k^{th} of erasure location.

Obviously, including the error evaluator sequence in the Berlekamp-Massey procedure is

possible. Such an effort increases the multiplication count extensively since each of those $2t$ iterations involves a circular convolution operation which requires $O(N^2)$ multiplications. Moreover, a number of delay registers have to be introduced because of this operation, i.e. one set of length N^2 per sequence. The algorithm \mathcal{A}_3 given below is essentially a modification of the algorithm \mathcal{A}_2 . The algorithm \mathcal{A}_3 computes only the error-erasure locator sequence. The maximum number of iterations that the algorithm has to perform is $2t$. It is sensible to include the initialisation procedure in the iterative process, say, in the first ρ iterations. When the algorithm is completed, the differentials of the error-erasure locator sequence and the error evaluator sequences are computed using equation (13) and equation (12) respectively.

Algorithm \mathcal{A}_3

Initialisation :

$$\lambda_i^{(0)} = \beta_i^{(0)} = 1; L_0 = 0; K_\rho = U_\rho = 1;$$

For $i=0, 1, \dots, N-1$.

For the first ρ iteration,

$$\Delta_r = \alpha^{i_k},$$

where i_k denotes the erasure location.

$$\begin{aligned} L_r &= L_{r-1} + 1 \\ \lambda_i^{(r)} &= \lambda_i^{(r-1)} - \Delta_r \alpha^{-i} \beta_i^{(r-1)} \\ \beta_i^{(r)} &= \lambda_i^{(r)}. \end{aligned}$$

For the next $(2t-\rho)$ iterations,

$$\Delta_r = \sum_{i=0}^{N-1} v_i \lambda_i^{(r-1)} \alpha^{i(r+t_0-1)} \quad (15a)$$

$$L_r = \delta_r (r + \rho - L_{r-1}) + (1 - \delta_r) L_{r-1}$$

$$K_r = \delta_r \Delta_r^{-1} + (1 - \delta_r) K_{r-1}$$

$$\begin{aligned} U_r &= \delta_r + (1 - \delta_r)(U_{r-1} + 1), \\ \begin{bmatrix} \lambda_i^{(r)} \\ \beta_i^{(r)} \end{bmatrix} &= \begin{bmatrix} 1 & -\Delta_r K_{r-1} \alpha^{-i U_{r-1}} \\ \delta_r & (1 - \delta_r) \end{bmatrix} \begin{bmatrix} \lambda_i^{(r-1)} \\ \beta_i^{(r-1)} \end{bmatrix} \end{aligned} \quad (15b)$$

for $r = 1, 2, \dots, 2t$.

Here $\delta_r = 1$ if both $\Delta_r \neq 0$ and $2L_{r-1} \leq r - 1 + \rho$; otherwise $\delta_r = 0$.

$$\text{Then } e_i = \frac{\gamma_i}{\lambda_i}, \text{ where } \lambda_i = 0. \quad (15c)$$

For the first ρ iterations Δ_r is set to α^{i_k} as i_k denotes the erasure location. Hence, after ρ iterations, $\{\lambda_i\} = \{\beta_i\} = \{\psi_i\}$. The algorithm proceeds toward $2t$ iterations to compute the error locator sequence. This sequence is used to compute the error magnitude consequently.

4. Architecture

Two architectures developed for erasures detection and correction are outlined in this section. the first one works under the frequency domain decoding environment according to the algorithm \mathcal{A}_1 whilst the second does in the time domain environment according to the algorithm \mathcal{A}_3 .

The frequency domain decoder comprises four main blocks, namely, syndrome computation, B-M procedure, Chien search and polynomial evaluation. According to the algorithm stated previously, the B-M computation cell can be designed similarly to the one proposed in [6]. However, when at most $2t$ erasures are considered, the syndrome path arrangement can be simpler even the computation of the discrepancy is to be performed in the t components window. This is because the main and its auxiliary register are now of length $2t+1$. Extending of registers length is not a severe drawback because in most application areas t is much smaller than N . Moreover those two main polynomials use the same scalar term in their vectors modification. This scalar term is hence common. In the polynomial initialisation stage, $\Delta^{(r)}$ is assigned to α^{i_k} where i_k is the k^{th} location of erasures. The auxiliary polynomial is taken from the

main polynomial (according to equations 8b and 8c). Hence the operation at this stage is quite simple and trivial. To obtain the error magnitude, not only the Forney polynomial but also the derivative of $\Lambda^{(2t)}(x)$ are needed. The derivative of $\Lambda^{(2t)}(x)$ can be easily achieved by sorting out $\Lambda^{(2t)}(x)$ for only the odd degree terms.

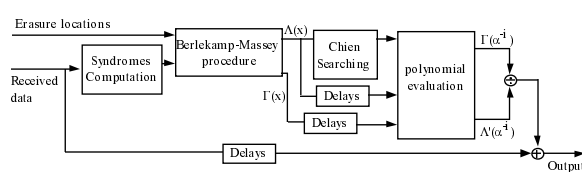


Figure 3 VLSI architecture of the frequency domain decoder (Algorithm A)

As shown in figure 3, the Berlekamp-Massey procedure can start immediately after syndrome components are available since syndrome values are needed for $\Gamma(x)$ initialisation and discrepancy computation according to equation (8a). Direct syndrome computation shown in [1, 6] can be employed. Chien searching is the efficient way to obtain roots of the error-erasure locator polynomial. This can be done in N or $2Nt$ clock cycles depending on the computing scheme. Similarly, polynomial evaluations of $\Gamma^{(2t)}(\alpha^i)$ and $\Lambda^{(2t)}(\alpha^i)$ can be carried out using the scheme shown in [1] or [5]. These circuits may consist of $2t$ multipliers and can evaluate a polynomial of degree $2t$ at a speed of one clock cycle per root. However for only one multiplier circuit, it is desirable to use the Horner's scheme which can give the speed of $2t$ clock cycles per root.

Shown in figure 4 is the time domain decoder with erasure correctivity. Although it is clearly shown that the B-M procedure can accommodate the error evaluator sequence, it may not be desirable since the computation count is rather high. Therefore, computing the error evaluator sequence by equation (12) gives better performance. Now the $2t$ -array has to compute only the error-erasure locator sequence. $2Nt$ clock cycle is needed for the computation. To obtain the time domain syndrome, the time of $2Nt$ clock cycle is available because it can be computed in parallel

with B-M procedure. In such a case, nest algorithm is suitable to attack the long length convolution problem. The algorithm breaks one dimension array to 2D arrays and uses fast computation technique for each dimension convolution. When the error-erasure locator sequence is valid, equation (11) can then be performed.

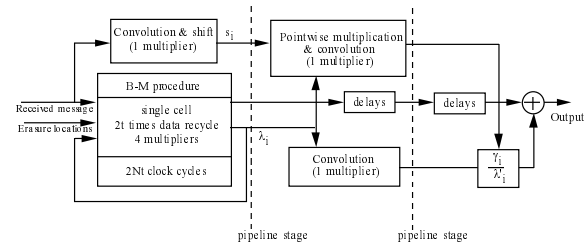


Figure 4 The architecture of a time domain decoder (algorithm B)

Consequently, the differentiation of the error-erasure locator sequence can be computed at the same time with the error evaluator sequence. These may also take $2Nt$ clock cycles. To minimise the hardware requirement, a bit serial inversion multiplication technique [8] is suitable in this case since it takes at most N clock cycles for an error magnitude. These three steps (B-M procedure, convolution and error magnitude computation) can naturally be pipelined. The decoder with data recycled in the B-M procedure is sketched in figure 4.

5. Conclusion

Errors and erasures correction capability Berlekamp-Massey algorithm has been obtained when the error evaluator or Forney polynomial has been initialised properly. Hereby $S(x)$ and $(1 + S(x))$ have been used. Using such an algorithm, $\phi(x)$ and $S(x)\phi(x) \bmod x^{2t}$ have been computed implicitly in the first ρ iterations. In contrast, using the Euclid's algorithm as proposed by Shao and Reed in [1], Λ_0 and Q_0 have to be initialised with $\phi(x)$ and $S(x)\phi(x) \bmod x^{2t}$ respectively. Therefore the erasure locator polynomial has to be computed before the iterations start. Unfortunately the proposed technique can not be easily applied to the transformed domain decoding. The time domain

technique cannot be directly derived as the case that erasures are not considered. The main reason is the effect of truncation and window operation. Convolution operation affected by the polynomial truncation in the frequency domain seems to take up major computation counts. This may not be desirable in the application where resource such as memory is limited. An alternative choice is to compute the convolution outside the iterative procedure as stated in the algorithm ③.

Two main architectures outlined in this paper are designed upon the given algorithms. Pipeline techniques are introduced to speed up the operation since tasks are performed in a broken small batch.

References

- [1] M. Shao and S. Reed, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays", IEEE Trans. on Comput., Vol. C-37, No. 10, Oct. 1988, pp. 1273-1280.
- [2] E.R. Berlekamp, "Algebraic Coding Theory", Mc Graw-Hill, 1968.
- [3] R.E. Blahut, "Theory and Practice of Error Control Code", Addison-Wesley, 1983.
- [4] R.E. Blahut, "A Universal Reed-Solomon Decoder", IBM Journal of Research & Development, Vol. 28, No. 2, March 1984, pp. 150-158.
- [5] B. Arambepola and S. Choomchuay, "Algorithm and Architectures for Reed-Solomon Codes", GEC Journal of Research, Vol. 9, No. 3. 1992, pp. 172-184.
- [6] B. Arambepola and S. Choomchuay, "Array Architectures for Reed-Solomon Decoding", Proc. IEEE-ISCAS, Singapore, June, 1991, pp. 2963-2966.
- [7] S. Choomchuay and B. Arambepola, "An algorithm and A VLSI architecture for Reed-Solomon Decoding", Proc. IEEE-ISCAS, San Diego, USA, May, 1992, pp. 2120-2123.
- [8] S. Choomchuay, "On the Implementation of Finite Field Operations", Ladkrabang Engineering Journal, Vol. 11, No. 1, June 1994, pp. 7-16.
- [9] S. Choomchuay and B. Arambepola, "Algorithms and Architectures for Time-domain Reed-Solomon Decoding", IEE Proc.-I, Vol. 140, No. 3, June 1993, pp. 189-196.