Modified Time-Domain Algorithm for Decoding Reed–Solomon Codes
Yousef R. Shayan and Tho Le-Ngoc

Abstract—A technique to reduce the number of inversions in the time-domain decoding algorithm based on an algebraic decoder (Blahut’s decoder) is introduced. It is proved that the modified algorithm is equivalent to the original one. The modified algorithm can be used in the universal Reed–Solomon decoder to decrease complexity.

I. INTRODUCTION

A PRIMITIVE RS(n, k) code with symbols from GF(2^m) has codewords of length n = 2^m - 1. These codes can correct t = (n - k)/2 symbol errors, with k information symbols. Time-domain decoding algorithms are attractive candidates for designing universal hardware RS decoders [1], [2]. A universal decoder can be used to decode any RS code with any block length and symbol alphabet, up to the limits of the storage registers associated with the decoder.

The time-domain decoding algorithm based on an algebraic decoder [1] has the disadvantage of having two multiplicative inverters (divisions) in GF(2^m). Implementation of division in Galois fields is fairly complex. In this paper, the decoding algorithm is modified to eliminate one of the divisions.

II. DECODING ALGORITHMS

The time-domain decoding algorithm based on an algebraic decoder was first presented by Blahut [1]. This algorithm has one main step that evaluates the error-locator vector A, error-evaluator vector w and the vector A'. The algorithm is explained by the following set of recursive equations, where α is a primitive element of GF(2^m).

for i = 0, ···, n - 1 and r = 1, 2, ···, 2t. The initial conditions are λ_i(0) = b_i(0) = ω_i(0) = 1 for all i; λ_i'(0) = b_i'(0) = a_i(0) = 0 for all i; L = 0 and δ_i = 1 if both Δ_i ≠ 0 and 2L ≤ r - 1, and δ_i = 0 otherwise. Then, λ_i'(2t) = r if and only if the ith symbol of the received vector v is in error. Also we have X' = X(2t), and ω = ω(2t). If λ_i = 0, the error values are evaluated as

and e_i = 0, otherwise.

One multiplicative inverter is required in (5) and another for the evaluation of Δ_i^(r-1) in (3) and (4). The inverter for Δ_i^(r-1) can be omitted; hence, the structure for the decoder will need only one inverter for evaluating e_i in (5).

To omit the inverter of Δ_i^(r-1), we will show that this decoding algorithm can also be explained by the following recursive equations [2], [3]

for i = 0, ···, n - 1, and for r = 1, 2, ···, 2t. The initial conditions are λ_i(0) = b_i(0) = ω_i(0) = 1 for all i; λ_i'(0) = b_i'(0) = a_i(0) = 0 for all i; L = 0, δ_i = 1, and δ_i = 1 if both Δ_i ≠ 0 and 2L ≤ r - 1, and δ_i = 0, otherwise. Then, λ_i'(2t) = r if and only if the ith symbol of the received vector v is in error. We have X' = X(2t), and ω = ω(2t), for all i. If λ_i = 0, the error values are evaluated as

the e_i = 0, otherwise.
In this algorithm, a new variable $\theta_i$ is used. This parameter is initialized in the beginning of the algorithm and updated in each iteration.

### III. EQUIVALENCE OF TWO ALGORITHMS

To prove that the modified algorithm is equivalent to the original one, we will prove the following:

1. $\lambda_i = K \lambda_i$, for all $i$ where $K$ is a nonzero element of $\text{GF}(2^m)$.
2. $e_i = \alpha^i \left( \lambda_i / \omega_i \right)^{-1} = \alpha^i \left( \lambda_i / \omega_i \right)$ for all $i$.

To prove the preceding two conditions we present the following proposition.

**Proposition:** The variables of the original algorithm and the modified one are related by

\[
L_{x_i} = L_{x_i}^{(r)} = \theta_i b_i^{(r)}
\]

\[
\lambda_i^{(r)} = \lambda_i^{(r)} = \prod_{l=0}^{r-1} \theta_i
\]

\[
\beta_i^{(r)} = \beta_i^{(r)} = \prod_{l=0}^{r-1} \theta_i
\]

\[
\omega_i^{(r)} = \omega_i^{(r)} = \prod_{l=0}^{r-1} \theta_i
\]

where $r = 2t$.

**Proof:** To prove this proposition we use induction. The basic step of the induction is evident from the initialization of the vectors and $\prod_{l=0}^{r-1} \theta_i = \theta_0 = 1$. The assumptions of the induction are (12)-(18) when $r = r'$. Therefore, assuming (12)-(18) are true for $r = r'$, we should prove the same equations for $r = r'+1$.

Considering (1), (6) and (14) for $r = r'$, we have

\[
\Delta^{(r'+1)} = \Delta^{(r'+1)} = \prod_{l=0}^{r-1} \theta_i
\]

According to (10) and the definition of $\xi_r$, it is obvious that $\theta_i$ is always nonzero and so is $\prod_{l=0}^{r-1} \theta_i$. Hence, from (19), $\Delta^{(r'+1)}$ and $\Delta^{(r'+1)}$ are both zero or both nonzero at iteration $r'+1$. This results in $\xi^{(r'+1)} = \xi^{(r'+1)}$ and $\Delta^{(r'+1)} = \Delta^{(r'+1)}$, and the proof of (12) is complete.

Now the proof of (13)-(18) can be given by expanding the left-hand sides of these equations and showing that they are the same as the right-hand sides. This can be done by using (10), (19), (3), (4), (8), (9), and assumptions of the induction.

**Proof of (13):**

\[
\begin{align*}
\xi_{r'+1}^{(r'+1)} &= \delta_{r'+1}^{(r'+1)} + (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} b_i^{(r'+1)} \\
&= \delta_{r'+1}^{(r'+1)} \prod_{l=0}^{r'-1} \theta_i \lambda_i^{(r')} + (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} \theta_i \beta_i^{(r'+1)} \\
&= \delta_{r'+1}^{(r'+1)} \Delta^{(r'+1)} \lambda_i^{(r')} + (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} \theta_i \beta_i^{(r'+1)} \\
&= \theta_i b_i^{(r'+1)}
\end{align*}
\]

**Proof of (14):**

\[
\begin{align*}
\Delta_i^{(r'+1)} &= \theta_i \lambda_i^{(r')} + \Delta_i^{(r'+1)} \alpha^{-i} b_i^{(r')} \\
&= \prod_{l=0}^{r'} \theta_i \lambda_i^{(r')} + \Delta_i^{(r'+1)} \prod_{l=0}^{r'-1} \theta_i \alpha^{-i} \theta_i \beta_i^{(r')} \\
&= \prod_{l=0}^{r'} \theta_i \lambda_i^{(r'+1)}
\end{align*}
\]

**Proof of (15):**

\[
\begin{align*}
\beta_i^{(r'+1)} &= (1 - \delta_{r'+1}^{(r'+1)}) b_i^{(r')} + \delta_{r'+1}^{(r'+1)} \lambda_i^{(r')} \\
&= (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} b_i^{(r')} \\
&= (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} \theta_i \beta_i^{(r')} \\
&= (1 - \delta_{r'+1}^{(r'+1)}) \alpha^{-i} \theta_i \beta_i^{(r')} \\
&= \theta_i b_i^{(r'+1)}
\end{align*}
\]

**Proof of (16):**

\[
\begin{align*}
\xi_i^{(r'+1)} &= \Delta_i^{(r'+1)} \beta_i^{(r')} + \theta_i \Delta_i^{(r')} + \Delta_i^{(r'+1)} \alpha^{-i} b_i^{(r')} \\
&= \Delta_i^{(r'+1)} \prod_{l=0}^{r-1} \theta_i \beta_i^{(r')} + \theta_i \prod_{l=0}^{r'-1} \theta_i \lambda_i^{(r')} \\
&= \Delta_i^{(r'+1)} \prod_{l=0}^{r-1} \theta_i \alpha^{-i} \theta_i \beta_i^{(r')} \\
&= \prod_{l=0}^{r'} \theta_i \lambda_i^{(r'+1)}
\end{align*}
\]

The proof of (17) and (18) is the same as the proof of (13) and (14), respectively. Note that, according to (10), $\theta_i^{(r'+1)} = \theta_i^{(r')}$ for $\delta_i^{(r'+1)} = 0$ and $\theta_i^{(r'+1)} = \Delta_i^{(r'+1)}$, otherwise. This fact is used in expanding the preceding equations.

### IV. CONCLUSION

To design the universal decoder based on the original time-domain algorithm, 17 universal multipliers and two multiplicative inverters are required [2]. If the modified algorithm is used, the number of multipliers remain the same but one inverter is reduced. Reduction of one inverter decreases...
the complexity of the decoder, since one universal inverter requires a few thousand gates [2]. Note that by using the modified algorithm, an extra circuitry for evaluation of $\theta_r$ in (10) should be used. However, this circuitry needs only a few gates and has much less complexity compared with a universal multiplicative inverter.

REFERENCES

