APPENDIX – A

COMPLEXITY ANALYSIS OF RS DECODER USING BERLEKAMP MESSAY ALGORITHM:

For an \((n, k)\) Reed Solomon code, the rate \(R\) is given by \(R = \frac{k}{n}\). Because of the Maximum Distance Separable property, the error correcting capability of the code is given by the relation \(2t = n - k\). This means in terms of rate \(R\) and the code length \(n\) the error correcting capability of the code is given as \(t = \frac{1-R}{2}\).

The syndrome based decoding of Reed Solomon code involves the following 3 steps.

1. Syndrome Computation
2. Key Equation Solver
3. Evaluation of error magnitude, say by Forney’s Formula

The computational complexity of each of these 3 steps need to be evaluated.

**Step1: Syndrome Computation**

The first step in decoding process is to divide the received polynomial by each of the factors of the generator polynomial. The reminders resulting from these divisions are known as the syndromes and

\[
S_i = Q_i(x) \cdot (x + \alpha^i) + R(x) \tag{A.1}
\]

so that when \(x = \alpha^i\) reduces to \(S_i = R(\alpha^i)\) for \(i = 1\) to \(2t\).
This means that each of the syndrome values can also be obtained by substituting \( x = \alpha^i \) in the received polynomial as an alternative to the division of \( R(x) \) by \( (x + \alpha^i) \) to form the reminder. From the above equation, the there are \((2t-1)\) additions of syndromes coefficients to be added and to get the individual terms of the syndrome, there are \((2t-1)\) multiplications to find the syndrome terms.

**Step2: Key Equation Solver**

\[
\Omega(x) = \Xi(x).S(x) \pmod{x^{2t}}
\]

(A.2)

Where \( \Omega(x) \) or \( V(x) = \) Connection polynomial
\( \Xi(x) \) or \( \Omega(x) = \) Error locator polynomial

In order to understand the above equation and its complexity, The Inversion less Berlekamp-Massey algorithm is briefly described as:

For the input syndrome \( S = [S_0 \ldots S_{2t-1}] \). With the initialization of \( \Lambda^{(r)}(x) = 1 \) (Error locator polynomial) and \( B^{(r)}(x) = 1 \) (being the Galois polynomials), \( \Delta \) being the discrepancy value and \( \Xi = 1 \), \( L = 0 \) (integers),

For the loop index \( r = 0,1,2,\ldots,2t-1 \), do:

Compute the discrepancy value \( \Delta^{(r)} = \sum_{i=0}^{L} \Lambda^{(i)} S_{r-i} \)

Update the \( \Delta^{(r+1)}(x) = \Xi^{(r)}.\Lambda^{(r)}(x) - \Delta^{(r)} \cdot xB^{(r)}(x) \)

(Thsi is Ex-OR summation, subtraction is same as sum)

If \( \Delta^{(r)} \neq 0 \) and \( 2L \leq r \), then

Assign \( B^{(r+1)}(x) \leftarrow \Delta^{(r)}(x) \) and \( L \leftarrow r+1-L \)
Assign $\mathbf{y}^{(r+1)} \leftarrow \Lambda^{(r)}$

Else

Assign $B^{(r+1)}(x) \leftarrow x.B^{(r)}(x)$

Assign $\gamma^{(r+1)} \leftarrow \gamma^{(r)}$

Endif

End of for loop

Outputs are $\mathbb{H} = [\Lambda_0^{(2\mathbb{H})}, \Lambda_1^{(2\mathbb{H})}, \Lambda_2^{(2\mathbb{H})}, \ldots \Lambda_{2\mathbb{H}}^{(2\mathbb{H})}]$, $\mathbb{H} = \gamma^{(2\mathbb{H})}$, $L, B = [B_0^{(2\mathbb{H})}, B_1^{(2\mathbb{H})}, \ldots, B_{(2\mathbb{H})}^{(2\mathbb{H})}]$

The superscript (r) indicates the rth iteration and subscript (i) indicates the ith coefficient.

The diagram shown below illustrates the rIBMM system. The syndrome ($S_0$ to $S_{2t-1}$) is the input while $\mathbb{H}, \mathbb{H}, L, B$ and $z$ are output.
Fig. A.1 Syndrome Computation

The discrepancy computation $\Delta^{[*]}$ from the block 200, as seen, uses $t+1$ multipliers, $t$ adders, $2t$ registers, neglecting miscellaneous circuitry. The error locator update block 202 uses $2t+1$ multipliers, $t+1$ adders, $2t+1$ latches ($t+1$ latches for $\Lambda(x)$ and $t$ latches for $B(x)$), $t$ multiplexers. $2t+1$ multipliers are used since the update of $\Lambda_0$ uses one multiplier while the update of each of the remaining $t$ terms $\Lambda_1, \Lambda_2, \ldots, \Lambda_t$, uses two multipliers. The critical path of the error locator update block contains one multiplier and one adder. MC stands for
Boolean operation $\Delta^{(r)} \neq 0$ and $2L \leq r$, corresponding to the $\mathcal{E}_i$ values and is a scratch polynomial. Hence, the IRBM system shown contains a total of $3t+2$ multipliers, $2t+1$ adders, $2t+1$ latches and $t$ multiplexers. The figure shown below illustrates the computations required in generating the discrepancy values. Similarly, for parallel inversion less Berlekamp–Massey method:

![Diagram](image)

**Fig.A.2 Number Multiplications in IBKM algorithm**

The complexity can be reduced by using similar techniques as shown in the table below. But what is important is the order of growth of the complexity which is order of square of time for multiplications (therefore, it is non-linear decoding as further shown for the decoding of error magnitude).

**Step 3: Error Magnitude Evaluator:**

When the error locator polynomial $\tilde{\mathcal{E}}(x)$ is obtained, its roots (indicating the error locations) can be obtained, by a Chien search.
The corresponding error magnitudes may be obtained from Forney formula:

\[ Y_i = \frac{\Omega(x_i^{-1})}{x_i^{-1}A(x_i^{-1})} \]  \hspace{1cm} (A.3)

where \( i = 1, 2, \ldots, e \) (error locations), \( \Lambda \) is the even or odd polynomial.

This approach takes \( t \) additional cycles after Chien search has found the error locations and may not be desirable. By the following equation

\[ Y_i = \frac{\Omega_{0b}B(x_i^{-1})}{x_i^{-1}A(x_i^{-1})} \]  \hspace{1cm} where \( i = 1, 2, \ldots, e \) (error locations) \( \Lambda \) is the even or odd polynomial.

If \( t \) is even, \( MC \) denotes clock cycles and indicates the index of a root (error location). The evaluation of \( \Lambda(x), \Omega(x), B(x) \) use half multipliers (i.e., one input end of the multiplier is fixed) whose complexity may be roughly one third that of a full multiplier. By setting \( B \) to the initial value of the loop logic and multiplying \( \Lambda_x \) in the first iteration (which otherwise is a null operation), the computation of \( \Omega_{0b} \Lambda_x 2^t \) is effectively achieved with one full-multiplier and one multiplexer, whereas other implementations may use three full-multipliers. In total, the revised Parallel CSEE system uses \( (2t-1) \) half multipliers, 3 full multipliers, 1 inverter, \( 2t-2 \) adders, and \( 2t+3 \) latches/registers. The error evaluator polynomial \( \Omega(x) \) has the same degree bound as \( B(x) \).
Fig. A.3 Error Magnitude Evaluator

The odd polynomial is

\[ \Lambda'(x) = \Lambda_1 + 2\Lambda_2x + \cdots + \Lambda_{t-1}x^{t-2} \]

Hence, the computation of error magnitudes can be simplified further to be

\[ Y_i = \frac{m_B(X_i^{-1})}{B(X_i^{-1})\Lambda_{\text{odd}}(X_i^{-1})}, \quad i = 1, 2, 3, \ldots, e \]

Where \( \Lambda_{\text{odd}}(X_i^{-1}) \) can be obtained during Chien search

The algorithm involves ‘n(t-1)’ Galois field multiplications and ‘nt’ Galois field additions. Knowing \( \alpha^i \) s, Forney’s formula is evaluated
with ‘2t^2’ Galois field multiplications, ‘2t-1’ Galois field additions and ‘t’ inversions.

Steps 1, 2 and 3 involve ‘3nt+10t^2-n+6t’ Galois field multiplications and ‘3nt+t^2-t’ Galois field additions

Substituting $t = \frac{(1-R)}{2}$ in the expressions for total number of Galois Field multiplications and Galois field additions we get the syndrome based decoding complexity for a given rate R as:

$$\left(\frac{5R^2-13R+8}{2}n^2 + (2-3R)n\right) + \left(\frac{6-9R+3R^2}{2}n^2 + \frac{1-R}{2}n\right) = O(n^2)$$

Thus for a given rate R complexity of decoding grows exponentially with code length n (i.e., it is non linear)
APPENDIX B

AUTHOR(S) PUBLICATIONS IN SUPPORT OF THE THESIS

INTERNATIONAL JOURNALS:


NATIONAL CONFERENCE:
