reed Solomon Codes for Reliable Communication in Internet of Things

by Ali Mahmudi

Submission date: 24-May-2023 09:15AM (UTC+0700)
Submission ID: 2100495170
File name: IOT_REED_SOLOMON_CODES_FOR_RELIABLE.pdf (1.46M)
Word count: 6658
Character count: 29283

In: Advanced Security Issues … ISBN: 978-1-53615-538-9
Editor: Vijey Thayananthan

Chapter 2

REED SOLOMON CODES FOR RELIABLE COMMUNICATION IN INTERNET OF THINGS

Ali Mahmudi
Software Engineering Department,
National Institute of Technology (ITN), Malang, Indonesia

ABSTRACT

In the networking field, Internet of Things (IoT) is the current state of the art in the nowadays Information of Technology era. The networking may be defined as an external or internal network. The backbone of the IoT is the internet connections. The IoT connects various objects to the Internet so that they can communicate and exchange billions of data and information among various devices and services. They may be remotely controlled from a distant area. As IoT systems will be open and available everywhere, a number of security issue may arise. One issue that remains open in the IoT technology is security and privacy issues. Because of this security issue, the communications among many different devices powered by IoT could not be said as a reliable technology.

Because of this, the security of the IoT systems can be enhanced by adding an error correction scheme both in the communication channel as well as the data store. By introducing the error correction scheme, the risks may be reduced to an acceptable level, and the security could be enhanced.

A Reed-Solomon (RS) code is one of many error control coding schemes that was firstly introduced by Reed and Solomon in 1960. This code has been used in various applications, such as CD-ROMs, space communications, DVD technology, digital TV and much more.

Here, the RS code is discussed in detail. It raised the issue of RS decoding scheme using the Welch Berlekamp algorithm. It presents the implementation of the Welch Berlekamp algorithm for the RS decoder in detail. The VHDL implementation VHDL for Hard Decision Decoding using the Welch Berlekamp algorithm is also presented. Without loss of generality, the Hard Decision Decoding for RS(255, 239) over GF(2^8) is developed using VHDL.

Keywords: Internet of Things, error control codes, Reed Solomon codes, VHDL

1. INTRODUCTION

Internet of Things was firstly introduced by Kevin Ashton in 1999. Fifteen years ago, there is not yet a global consensus on the definition of IoT. In general, the IoT is defined as the ability to connect various objects using internet connection as the backbone and allows them to interact with other objects, or with other intelligent computing equipment via the internet network. IoT has been used in various applications in many aspects of human life [1].

Widespread use of IoT technology in modern life turns making human life to be much more comfortable. IoT has affected human life nowadays, from the domestic home appliances up to smart cars. In business point of view, IoT has been used in decision making to increase production as well as the quality of production, to oversee the distribution of goods, to prevent counterfeiting, to shorten the waiting time of unavailability goods in retail markets, in supply chain management, and so on.

IoT has also been used in the public health sector. IoT has been used in medical applications, for example, for monitoring glucose in a diabetic patient. A doctor can receive data in real time to monitor the patient's condition and to manage the medicine of a patient from home by using IoT

application as a backbone. Thus, looking after and monitoring a patient becomes much easier to be done. IoT has also been adopted in smart home applications which makes it possible to arrange and control some home appliances remotely from a distant area [1].

IoT consists of different devices. Some technologies involved is RFID as an object and location identifier, WSN (Wireless Sensor Network), cloud computing and so on. They are connected with various data collection terminals through internet networks and other communication networks. Data about an object is taken in real time; it is then converted into the appropriate data format and then sent to the data center. The data is then processed using clouds computing and other intelligent computing technologies that can process and store large amounts of data [2].

As many technologies involved in building up IoT, a reliable data communication system is needed to protect every part of the data. Broadly speaking, there are three things that may become an issue. The first is physical security, especially the security of sensors and RFID from interference, and a signal interception. Second is that all of those elements such as sensors, transmission systems, and others, should operate normally. The system should guarantee that all those elements should work as they should be. The last one is reliable data and information exchange. The message transmission between devices in IoT should be more reliable, authentic, has integrity and confidentiality. The data and information sent and received along the internet network should not be damaged, stolen or falsified [2].

Because of this, the reliability of the IoT systems can be enhanced by adding an error correction scheme both in the communication channel as well as the data store. By introducing the error correction scheme, the risks may be reduced to an acceptable level, and the reliability could be enhanced. Reed-Solomon (RS) code is one of many error control coding schemes available. It was firstly introduced by Reed and Solomon in 1960. This code has been used in various applications, such as CD-ROMs, space communications, DVD technology, digital TV and much more.

2. Error Control Codes

The general idea behind error control coding is to put some restrictions on the characteristic of the source signals so that the transmitted signals are more immune to corruption. Therefore the effects of noise during transmission are reduced or even eliminated. The term coding refers to error detection and error correction techniques. Hence, the work of error control coding involves a technique by which some extra information is added to the message signal in such a way that allows the receiver to estimate the transmitted data from the corrupted received information. This extra information is used to shield the message signals.

The work of coding was pioneered by Shannon in 1948 [3]. He showed that the existence of error control codes would reduce the existence of data corruption in the channel if data is transmitted at a rate less than the channel capacity. Reed-Solomon (RS) codes, developed by Irving Reed and G. Solomon in 1960 [4], are a type of error control code which have proven to be of practical value in a variety of applications ranging from satellite communications to the recording of information on a storage device [6, 7, 8, 9, 10].

A typical data communication system is shown in Figure 1. The RS encoder takes a block of digital data and then adds extra 'redundant' bits. Errors may occur during transmission or storage for a number of reasons, for example, cross talk on a telephone line scratches on a CD, etc. The RS decoder processes each block and attempts to correct errors and recover the original data. The number and type of errors that can be corrected depending on the characteristics of the RS code.

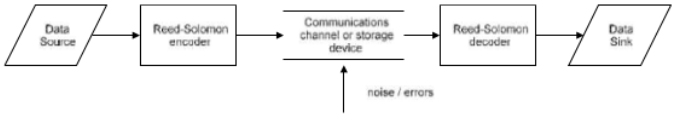

Figure 1. The Digital Communication system.

The RS code is specified as RS(n,k) over GF(2^m) with m bits per symbol. This means that

n (= $2m$-1) is the number of symbols in every codeword
k (= n – $2t$) is the number of information symbols in every codeword

If t is the maximum number of symbols that can be corrected

d (= $2t$ + 1) is the designed distance
$2t$ (= n-k) is the number of parity check symbols

then the RS code is shown in Figure 2.

Figure 2. The systematic RS code.

The RS encoder works by taking k data symbols of m bits each and adds n-k parity symbols to make an n symbol codeword. If the information symbol is left unaltered and the parity symbols are appended (as shown in Figure 2), it is called a systematic encoder. It is different from a non-systematic encoder where the information symbols have been altered in such a way that the information symbols are not obvious [11, 12]. This non-systematic encoder is not covered in this chapter. The amount of processing required to encode and decode RS codes depends on the number of symbols that can be corrected. A large value of t means that a large number of errors can be corrected, but it requires more computation and more hardware than a small value of t.

3. RS ENCODER

The RS(n, k) code of block length n, message length k, and designed distance n-k+1, is easily explained by using its special polynomial, called

generator polynomial, $g(x)$. The general form of the RS(n, k) generator polynomial is shown in equation (1) [13, 14, 15].

$$g(x) = \prod_{i=0}^{n-k-1} (x - \alpha^{b+i}) \quad (1)$$

where b is a nonnegative integer selected by the designer, without loss of generality, it is assumed that $b = 0$.

All valid RS(n, k) codewords can be regarded as being polynomials of degree (n-1) over GF(2^m) which are divisible by its generator polynomial g(x). Let

$$I(x) = i_0 + i_1 x + \ldots + i_{k-1} x^{k-1} 7 \quad (1)$$

be the information polynomial to be encoded. The systematic encoding can be considered as multiplying the information polynomial $I(x)$ by x^{n-k} and then follows by dividing the result with $g(x)$ to obtain the remainder polynomial of degree n-k-1. The encoded signal is then formed by subtracting $r(x)$ from $I(x)x^{n-k}$. It can be written as equation (2) [12, 15, 16]

$$c(x) = I(x).x^{n-k} + (I(x).x^{n-k} \bmod g(x)) \quad (2)$$

It should be noted that the remainder polynomial is, in fact, the parity check polynomial. This process ensures that a valid codeword $c(x)$ is a multiple of $g(x)$. The information symbol appears explicitly at the k highest order coefficients and the parity check symbols at the n-k lowest order coefficients. From this fact, it can be seen that the process of encoding is basically carrying out division over a finite field GF(2^m).

For example, if $g(x) = 1 + x + x^3$ and $I(x) = b_0 + b_1 x + b_2 x^2$. Multiplying $I(x)$ by x^3 will result $b_0 x^3 + b_1 x^4 + b_2 x^5$. If this result is divided by $g(x)$, then a quotient $(b_0 + b_2) + b_1 x + b_2 x^2$ and a remainder $(b_0 + b_2) + (b_0 + b_1 + b_2)x + (b_1 + b_2)x^2$. A circuit, which implements this computation, is shown in Figure 3 [4, 12].

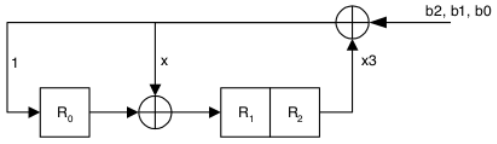

Figure 3. The structure of RS encoder over GF(2).

Shift 0).	$R_0 = 0$	$R_1 = 0$	$R_2 = 0$
Shift 1).	$R_0 = b_2$	$R_1 = b_2$	$R_2 = 0$
Shift 2).	$R_0 = b_1$	$R_1 = b_1 + b_2$	$R_2 = b_2$
Shift 3).	$R_0 = b_0 + b_2$	$R_1 = b_0 + b_1 + b_2$	$R_2 = b_1 + b_2$

The circuit in Figure 3 is very simple because it operates over GF(2) only. In general case of a code defined over GF(2^m), multiplication over GF(2^m) must also be performed. This generalized encoding scheme is shown in Figure 4 [13, 15].

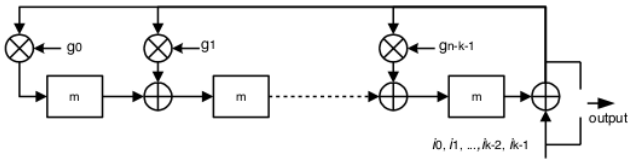

Figure 4. The structure of RS encoder over GF(2^m).

4. RS DECODER

In this section, the decoding method of RS codes is considered. The decoding process can be generally classified into two categories, hard decision decoding (HDD) algorithm, and soft decision decoding (SDD) algorithm. HDD algorithms include frequency domain decoding and time domain decoding [12, 18, 19]. Frequency domain decoding begins with the calculation of the syndromes which is the finite field equivalent of a Discrete

Fourier Transform (DFT) [17]. However, in time domain decoding there is no calculation of the syndromes.

There are quite a few algorithms to decode the RS codes, such as the Euclidean algorithm [20, 21, 22], the Fitzpatrick algorithm [23, 24], the Berlekamp Massey (BM) algorithm [25] and the Welch Berlekamp (WB) algorithm [26]. Here, the WB algorithm developed by L. Welch and E. R. Berlekamp is considered. Many researchers have commented and investigated this algorithm, as can be seen in [22, 27, 28, 29, 30, 31, 32]. This decoding method does not involve syndrome computation as a starting point. As in [30], this scheme may be used to develop the GMD decoder, one of the many approaches in applying soft decision decoding.

5. RS DECODING USING THE WB ALGORITHM

The WB algorithm performs RS decoding without explicit syndrome computation. Furthermore, the WB algorithm is known to be more amenable to SDD which offers around 2dB more coding gain than HDD [13]. Here, the WB algorithm for decoding RS code is presented [26, 30]. The algorithm of decoding RS codes can be broken down into 4 stages as shown in Figure 5:

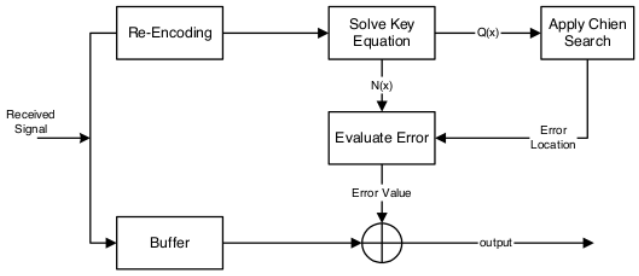

Figure 5. The RS decoder using the WB algorithm.

Stage 1: Re-encoding.
Stage 2: Solving the key equation using the Welch-Berlekamp Algorithm.
Stage 3: Applying the Chien search to solve the error locator polynomial.
Stage 4: Evaluating the error value to obtain the corrected message.

The first stage of decoding is re-encoding the received input signal $v(x)$. It corresponds to the syndrome computation in conventional RS decoding[29]. Basically, this process is quite similar to the encoding process. Re-encoding takes received information signals and then internally generates $2t$ parity check signals. If $r'(x)$ represents the internally generated parity check signals, then the re-encoding output will be the mod 2 addition of r'_i and the parity check input, v_i. This re-encoding output is called remainder polynomial $r(x)$. It can be expressed as equation (3) and (4), and the re-encoder circuitry with generator polynomial $g(x)$ as in (1) is shown in Figure 6.

$$r_i = \begin{cases} 0 : n-1 \geq i \geq 2t \\ v_i + r_i^{'} : i < 2t \end{cases} 2828 \tag{3}$$

$$r(x) = \sum_{i=0}^{2t-1} r_i x^i \; 2929 \tag{4}$$

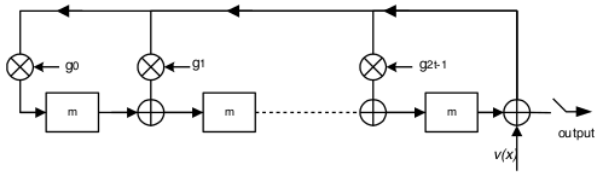

Figure 6. The re-encoder circuitry.

Let

$$g'(x) = \prod_{i=0}^{2t-2}(x - \alpha^i)\ 3030 \tag{5}$$

and then equation (6) is pre-computed constants held in memory.

$$G(x) = \sum_{i=0}^{2t-1} G_i x^i = \sum_{i=0}^{2t-1} \frac{1}{g_i} x^i\ 31 \tag{6}$$

Then, R_i is computed to get the Welch-Berlekamp input [29] as expressed in equation (7).

$$R_i = r_i G_i\ 32 \tag{7}$$

for $i = 0, 1, \ldots, 2t\text{-}1$.

Then, the Welch Berlekamp algorithm is used to solve the key equation. Let the input to WB algorithm be $2t$ pairs of (R_k, α_k) where R_k corresponds to the parity check input and α_k ($=\alpha^k$, for the case of error, only decoding) corresponds to the check location. As in [25, 26, 29], the WB algorithm is presented in this section.

For two polynomials $Q(x)$ and $N(x)$ define $L[Q(x), N(x)]$ as the length of the pair, which is the higher value between the degree of $Q(x)$ and degree of $N(x) + 1$ [26]. A similar approach applies to both $W(x)$ and $V(x)$ polynomials and its corresponding $L[W(x), V(x)]$.

At the iteration number p (where $0 < p < 2t$), there are two pairs of polynomials $[Q_p(x), N_p(x)]$ and $[W_p(x), V_p(x)]$ which already fits the already processed checks in equations (8) and (9).

$$Q^p(\alpha_{p-1})R_{p-1} = N^p(\alpha_{p-1})\ 33 \tag{8}$$

$$W^p(\alpha_{p-1})R_{p-1} = V^p(\alpha_{p-1})\ 34 \tag{9}$$

Furthermore, $L[Q(x), N(x)]$ is less than $L[W(x), V(x)]$. The algorithm needs to be extended from p to $p+1$.

At first, the pair of polynomials $[Q^p(x), N^p(x)]$ needs to be checked if it can accommodate the new input pair. The way to do it is by computing D1 as shown in equation (10)

$$D1 = Q^p(\alpha_{p+1})R_{p+1} - N^p(\alpha_{p+1})\,35 \tag{10}$$

If $D1 = 0$, it means that $[Q^p(x), N^p(x)]$ can accommodate the new input pair (R_{p+1}, α_{p+1}) and then branch A of Figure 7 is taken. However, if $D1$ is not zero, then $D2$ needs to be computed as shown in equation (11).

$$D2 = W^p(\alpha_{p+1})R_{p+1} - V^p(\alpha_{p+1})\,36 \tag{11}$$

Then, the two pairs $[Q^p(x), N^p(x)]$ and $[W^p(x), V^p(x)]$ need to be updated. The first updated pair is the $(W, V)^{p+1}$ polynomials as shown in equation (12)

$$(W,V)^{p+1} = (W,V)^p + D2(Q,N)^p / D1\,37 \tag{12}$$

and the second pair is the $(Q, N)^{p+1}$ polynomials as shown in equation (13)

$$(Q,N)^{p+1} = (Q,N)^p(x-\alpha_{p+1})\,38 \tag{13}$$

Finally, the orders of the two updated pairs need to be compared. The shorter value between equations (12) and (13) is set to be $[Q^{p+1}(x), N^{p+1}(x)]$, and the longer one is set to be $[W^{p+1}(x), V^{p+1}(x)]$. In other words, if the shorter polynomial pair is given by equation (13), then branch *B* of Figure 7 is taken. However, if the shorter polynomial pair is given by equation (12), then branch *C* is taken.

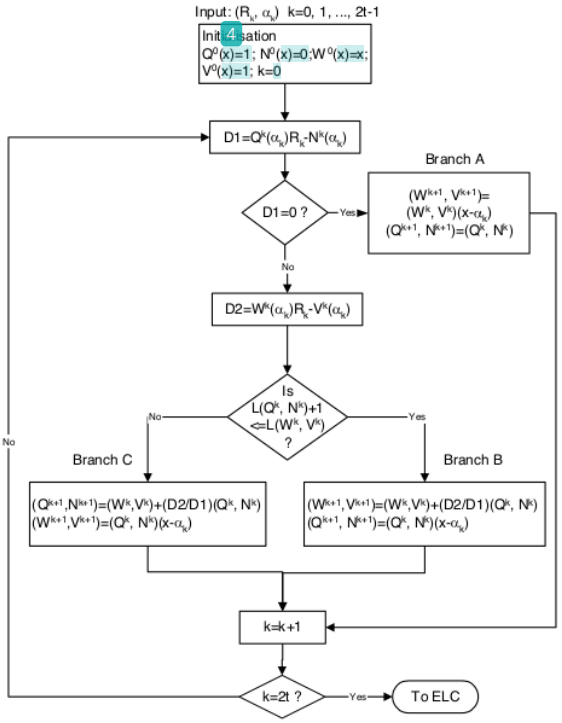

Figure 7. The WB algorithm flow chart diagram.

The WB algorithm process is carried out until all the $2t$ check pairs have been processed. Once all the $2t$ check pairs have been processed, the final value of both $Q(x)$ and $N(x)$ polynomials are taken for further process. The $Q(x)$ polynomial represents the error locator polynomial, and its roots need to be found. The Chien search method [32] is used to solve the $Q(x)$ polynomial. Let $Q(x)$ be a polynomial of degree p (where $p \leq t$) as shown in equation (14).

$$Q(x) = q_p x^p + q_{p-1} x^{p-1} + \ldots + q_0 \, 39 \tag{14}$$

If $Q_l(x)=q_lx^l$ for $0<l\leq p$, then equation (15) applies

$$Q(x) = Q_p(x) + Q_{p-1}(x) + \ldots + q_0 40 \tag{15}$$

and equation (15) can be shown as equation (16).

$$Q(\alpha^x) = Q_p(\alpha^{x+1})\alpha^{-p} + Q_{p-1}(\alpha^{x+1})\alpha^{1-p} + \ldots + q_0 41 \tag{16}$$

The implementation of equation (16) above requires p multipliers. As the $Q^{2t}(x)$ polynomial will be of the order up to t, it requires t multipliers.

In order to find the error value, more pre-computed constants are required [29]. Let

$$C = [(\alpha^0\alpha^1..\alpha^{2t-2})(\alpha^0 - \alpha^1)...(\alpha^0 - \alpha^{2t-1})]^{-1} 42 \tag{17}$$

and the C value is pre-computed. Then, the C value is multiplied by the generator polynomial $g(x)$ as shown in equation (18).

$$f(x) = Cg(x) 43 \tag{18}$$

The $f(x)$ polynomial is a pre-computed constant held in memory. This $f(x)$ value is solved using a method shown in (16). It requires $2t$ multipliers.

Once the WB algorithm has been completed, it is followed by the error correction and error evaluation. There are two possible error positions. The first case is error occurred at a data location and its value is obtained using equation (19) below. Where $Q'(x)$ polynomial is the derivative of $Q(x)$ polynomial.

$$e_i = \frac{N(\alpha^i)}{f(\alpha^i)Q'(\alpha^i)} 44 \tag{19}$$

The second case is error occurred at a check location. The error value at the check location can be found using equation (20) below.

$$e_i = \frac{N(\alpha^i)}{f(\alpha^i)Q'(\alpha^i)} + v_i \text{ , where } v_i \text{ is the received data.4} \quad (20)$$

Alternatively, it could be corrected by re-encoding the error-free information signal or not to be corrected at all.

6. RS MODIFIED WB ALGORITHM

Consider the RS(n, k) decoder with minimum distance d, error correcting capabilities t operating over GF(2^m). Let the input to the WB algorithm be 2t check pairs (R_k, α_k) where R_k and α_k represent the check value and check location respectively. The modified version of the WB algorithm is shown in Figure 8, which is a modified version of the one given in [30, 34]. This modification will be discussed in detail.

The most obvious difference is the use of J control signal instead of $L[Q(x), N(x)]$ and $L[W(x), V(x)]$. Let $L[Q(x), N(x)]$ is defined to be the higher value between the order of $Q(x)$ polynomial and the order of $N(x)$ + 1 polynomial. Similarly applies to $L[W(x), V(x)]$, which is defined as the higher value between the order $W(x)$ and order $V(x)$ + 1. The reason is explained as follows.

Let

$$J^k = L[W^k(x), V^k(x)] - L[Q^k(x), N^k(x)] \text{ 59,1} \quad (21)$$

The cycle enters branch A if $D1 = 0$ (see Figure 7). At this branch, both $W(x)$ and $V(x)$ polynomials are multiplied by (x-α_k), on the other hand, the

$Q(x)$, and $N(x)$ polynomials remain unchanged. Accordingly, the degrees of both updated $W(x)$ and $V(x)$ polynomials are incremented by one; however, the orders of the updated $Q(x)$ and $N(x)$ polynomials remain unchanged. It is then expressed in mathematical terms as (22) and (23).

$$L[W^{k+1}(x),V^{k+1}(x)] = L[W^{k}(x),V^{k}(x)]+1 \tag{22}$$

$$L[Q^{k+1}(x),N^{k+1}(x)] = L[Q^{k}(x),N^{k}(x)] \tag{23}$$

So applying equation (21) to both equations (22) and (23) yields equation (24).

$$J^{k+1} = J^{k}+1 \tag{24}$$

Branch B is taken if $D1 \neq 0$ and $L[Q^k(x), N^k(x)] + 1 \leq L[W^k(x), V^k(x)]$ (see Figure 7). In this branch, both the Q(x) and $N(x)$ polynomials are multiplied by $(x\text{-}\alpha_k)$, therefore the updated degree of $Q(x)$ (and $N(x)$, if applicable) would be incremented by one. On the other hand, the $W(x)$ and $V(x)$ polynomials are added with $(D2/D1).Q(x)$ and $(D2/D1).N(x)$, consecutively. The order of both $(W(x), V(x))$ polynomials does not change at all because of the condition $L[Q^k(x), N^k(x)] + 1 \leq L[W^k(x), V^k(x)]$. In mathematical terms, it can be expressed as equations (25) and (26)

$$L[W^{k+1}(x),V^{k+1}(x)] = L[W^{k}(x),V^{k}(x)] \tag{25}$$

$$L[Q^{k+1}(x),N^{k+1}(x)] = L[Q^{k}(x),N^{k}(x)]+1 \tag{26}$$

Then, applying equation (21) to both equations (25) and (26) yields (27)

$$J^{k+1} = J^{k}-1 \tag{27}$$

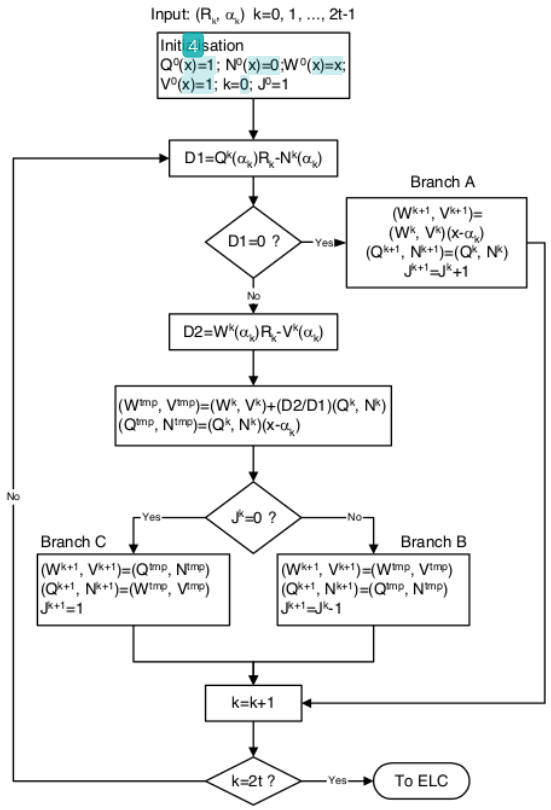

Figure 8. The modified WB algorithm flow chart diagram.

If $D1 \neq 0$ and $L[Q^k(x), N^k(x)] + 1 > L[W^k(x), V^k(x)]$, the process enters branch C (see Figure 7). The condition as in (28) can be rewritten as equation (29)

$$L[Q^k(x), N^k(x)] + 1 > L[W^k(x), V^k(x)] \tag{28}$$

$$L[W^k(x), V^k(x)] - L[Q^k(x), N^k(x)] < 1 \tag{29}$$

As $L[Q^k(x), N^k(x)]$ is never more than $L[W^k(x), V^k(x)]$ at any value of k, then equation (29) could be written as (30) or (31)

$$L[W^k(x), V^k(x)] - L[Q^k(x), N^k(x)] = 0 \tag{30}$$

$$J^k = 0 \tag{31}$$

Then, the control signal to enter either branch *B* or branch *C* is modified to checking whether $J^k = 0$ or not. The process at branch *C* generates equation (32), which can be written as equation (33), and equation (34).

$$L[Q^{k+1}(x), N^{k+1}(x)] = L[W^k(x), V^k(x)] \tag{32}$$

$$L[Q^{k+1}(x), N^{k+1}(x)] = L[Q^k(x), N^k(x)], \tag{33}$$

$$L[W^{k+1}(x), V^{k+1}(x)] = L[Q^k(x), N^k(x)] + 1. \tag{34}$$

Considering equation (21) to both equations (33) and (34), then it generates equation (35).

$$J^{k+1} = J^k + 1. \tag{35}$$

It can be written as (36)

$$J^{k+1} = 1 \tag{36}$$

The other improvement is the calculation of the intermediary stage (W^{tmp}, V^{tmp}) and (Q^{tmp}, N^{tmp}) if $D1 \neq 0$ as in Figure 8. This calculation may save hardware requirements to implement the WB algorithm and also to make the implementation easier.

7. The Implementation of the WB Algorithm

Assume that the WB algorithm operates over GF(2^8). The input check pair is (R_k, α_k) where R_k corresponds to the parity check value, and α_k corresponds to the check location. In the case of HDD, $\alpha_k = \alpha^k$. Here, four clock cycles are required to process every check pair. The first clock cycle is to compute the $D1$ value, the second cycle is to compute both $D2$ and $D2/D1$ values, the third one is to compute the ($Q^{tmp}(x)$, $N^{tmp}(x)$) polynomials and finally, the fourth cycle is to calculate the ($W^{tmp}(x)$, $V^{tmp}(x)$) polynomials and to decide which pair become the $Q^{k+1}(x)$ and $N^{k+1}(x)$ polynomials and which pair become $W^{k+1}(x)$ and $V^{k+1}(x)$ polynomials.

7.1. D1 Calculation

As the error correcting capability of the decoder is 8, the maximum degree of $Q(x)$ and $N(x)$ polynomials are 8 and 7 respectively. Assume that α_k^2, α_k^3, …, α_k^8 are already available (see section 6.5 later). To obtain the $Q^k(\alpha_k)$ value, 8 Berlekamp multipliers are required as (37).

$$Q^k(\alpha_k) = Q_8^k.(\alpha_k^8) + Q_7^k.(\alpha_k^7) + Q_6^k.(\alpha_k^6) + \ldots + Q_0^k \quad (37)$$

Similarly, the implementation of $N^k(\alpha_k)$ calculation would require 7 multipliers as shown in (38).

$$N^k(\alpha_k) = N_7^k.(\alpha_k^7) + N_6^k.(\alpha_k^6) + N_5^k.(\alpha_k^5) + \ldots + N_0^k \quad (38)$$

Once both $Q^k(\alpha_k)$ and $N^k(\alpha_k)$ values are available, $D1$ can be easily calculated by implementing equation (39)

$$D1 = Q^k(\alpha_k)R_k - N^k(\alpha_k) \quad (39)$$

and $1/D1$ can be easily obtained.

7.2. D2 Calculation

As with the $D1$ calculation, the error correcting capabilities of the decoder is 8. Hence, the maximum degree of both $W(x)$ and $V(x)$ polynomials are 8 and 7, respectively. It is assumed that α_k^2, α_k^3, …, α_k^8 are already available (see section 6.5 later). The multipliers used in $D1$ calculation in section 6.1 are re-used in this stage to implement equation (40).

$$W^k(\alpha_k) = W_8^k.(\alpha_k^8) + W_7^k.(\alpha_k^7) + W_6^k.(\alpha_k^6) + \ldots + W_0^k \tag{40}$$

Therefore 8 Berlekamp multipliers are required to compute the $W^k(\alpha_k)$ value. Similarly, the implementation of $V^k(\alpha_k)$ calculations would require 7 multipliers as in (41).

$$V^k(\alpha_k) = V_7^k.(\alpha_k^7) + V_6^k.(\alpha_k^6) + V_5^k.(\alpha_k^5) + \ldots + V_0^k\,60{,}2 \tag{41}$$

Once both $W^k(\alpha_k)$ and $V^k(\alpha_k)$ values are available, $D2$ can be found by using (42) and then D2/D1 can be calculated.

$$D2 = W^k(\alpha_k)R_k - V^k(\alpha_k)\,35 \tag{42}$$

7.3. D1 Calculation of (($W^{k+1}(x)$ & $V^{k+1}(x)$) or ($Q^{tmp}(x)$ and $N^{tmp}(x)$))

If $D1 = 0$, then equations (43) and (44) are computed.

$$\begin{aligned} W^{k+1}(x) &= \alpha_k(W_8^k x^8 + W_7^k x^7 + \ldots + W_0^k) + \\ &(W_7^k x^8 + W_6^k x^7 + \ldots + 0) \end{aligned} \tag{43}$$

$$V^{k+1}(x) = \alpha_k (V_7^k x^7 + V_6^k x^6 + ... + V_0^k) + (V_6^k x^7 + V_5^k x^6 + ... + 0) \quad (44)$$

The numbers of multipliers required to calculate ($W^{k+1}(x)$, $V^{k+1}(x)$) polynomials are 9 and 8; respectively. Here, the 8 multipliers to calculate $W^k(\alpha_k)$ in section 6.2 are re-used to implement the $W^{k+1}(x)$ circuitry with one extra multiplier is added. Similarly, the multipliers to calculate $V^k(\alpha_k)$ in section 6.2 are also re-used to implement the $V^{k+1}(x)$ circuitry with one extra multiplier is added. There is no required update on ($Q^k(x)$, $N^k(x)$) polynomials, hence ($Q^{k+1}(x)$, $N^{k+1}(x)$) polynomials are already available. J^{k+1} value needs to be updated, hence $J^{k+1} = J^k + 1$.

On the other hand, when $D1 \neq 0$, then the following equations will be computed and the hardware used will be the one on the case of $D1 = 0$.

$$Q^{tmp}(x) = \alpha_k (Q_8^k x^8 + Q_7^k x^7 + ... + Q_0^k) + (Q_7^k x^8 + Q_6^k x^7 + ... + 0) \quad (45)$$

$$N^{tmp}(x) = \alpha_k (N_7^k x^7 + N_6^k x^6 + ... + N_0^k) + (N_6^k x^7 + N_5^k x^6 + ... + 0) \quad (46)$$

7.4. Calculation of $W^{tmp}(x)$ and $V^{tmp}(x)$

At this stage, the $W^{tmp}(x)$ and $V^{tmp}(x)$ polynomials have to be found. The way to get the $W^{tmp}(x)$ polynomial is by multiplying $D2/D1$ value with $Q^k(x)$ polynomial, and then adding the $W^k(x)$ polynomial, as shown in equation (47).

$$W^{tmp}(x) = \frac{D2}{D1}(Q_8^k x^8 + Q_7^k x^7 + ... + Q_0^k) + (W_8^k x^8 + W_7^k x^7 + ... + W_0^k) \quad (47)$$

The number of multipliers required is 9, and it will re-use the multipliers to implement $W^{k+1}(x)$ or $Q^{tmp}(x)$ in section 6.3. Similar fashion is applicable to find $V^{tmp}(x)$ polynomial using equation (48).

$$V^{tmp}(x) = \frac{D2}{D1}(V_7^k x^7 + V_6^k x^6 + ... + V_0^k) + (N_7^k x^7 + N_6^k x^6 + ... + N_0^k) \tag{48}$$

The number of multipliers required is 8 which have already been used to implement $V^{k+1}(x)$ or $N^{tmp}(x)$ on section **c**. By considering the J^k value, the $W^{k+1}(x)$, $Q^{k+1}(x)$, $V^{k+1}(x)$ and $N^{k+1}(x)$ polynomials are then updated.

If $J^k = 0$ then equation (49) applies, otherwise equation (50).

$$\begin{aligned}(W^{k+1}(x), V^{k+1}(x)) &= (Q^{tmp}(x), N^{tmp}(x)) \\ (Q^{k+1}(x), N^{k+1}(x)) &= (W^{tmp}(x), V^{tmp}(x))\end{aligned} \tag{49}$$

and

$$\begin{aligned}(Q^{k+1}(x), N^{k+1}(x)) &= (Q^{tmp}(x), N^{tmp}(x)) \\ (W^{k+1}(x), V^{k+1}(x)) &= (W^{tmp}(x), V^{tmp}(x))\end{aligned} \tag{50}$$

7.5. Finite Field Exponentiation

In section 6.1 and 6.2, it has been assumed that α_k^2, α_k^3, …, α_k^8 are available. In this section, it will be explained how to obtain α_k^2, α_k^3, …, α_k^8 by performing multiplication less than 7 times.

Consider $GF(2^8)$ with $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ and take α to be a root of $p(x)$. Let

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}, \text{then it can be shown that } x^2 = \begin{pmatrix} x_0 + x_4 + x_6 + x_7 \\ x_7 \\ x_1 + x_4 + x_5 + x_6 \\ x_4 + x_6 \\ x_2 + x_4 + x_5 + x_7 \\ x_5 \\ x_3 + x_5 + x_6 \\ x_6 \end{pmatrix}$$

$$x^4 = \begin{pmatrix} x_0 + x_2 + x_3 + x_6 \\ x_6 \\ x_2 + x_3 + x_4 + x_5 + x_6 \\ x_2 + x_3 + x_4 + x_6 + x_7 \\ x_1 + x_2 + x_5 + x_7 \\ x_5 \\ x_3 + x_4 \\ x_3 + x_5 + x_6 \end{pmatrix} \text{ and } x^8 = \begin{pmatrix} x_0 + x_1 + x_3 + x_4 + x_7 \\ x_3 + x_5 + x_6 \\ x_1 + x_2 + x_3 + x_4 + x_6 + x_7 \\ x_1 + x_2 + x_3 + x_4 + x_5 + x_7 \\ x_1 + x_4 + x_7 \\ x_5 \\ x_2 + x_5 + x_6 + x_7 \\ x_3 + x_4 \end{pmatrix}$$

The combination of this method and multiplication is carried out to perform $x^{k'}$ (where $k' \neq 2, 4, 8$).

8. VHDL Decoder Using the WB Algorithm

8.1. Pin Out Connection

The HDD pins both input and output connections are as follows.

1. parity_number : integer input 8.
2. Input *clk*: clock active on the rising edge.
3. Input *reset_n*: active low asynchronous reset.

4. Input *reset_sync_n*: active low synchronous reset.
5. Input *di*: 8-bits input data to be decoded.
6. Input *di_rdy*: active high flag to indicate that *di* is valid.
7. Input *di_sync_in*: active high flag to synchronize the codeword.
8. Input *do_acpt*: active high flag to indicate that *do* is already taken.
9. Output *do*: the 8 bits decoder output.
10. Output *do_rdy*: active high flag to indicate that *do* is valid.
11. Output *di_acpt*: active high flag to indicate that *di* is already taken.
12. Output *do_sync_out*: active high flag to synchronize the output.
13. Output *do_error*: active high flag when a correction took place.
14. Output *do_parity_n*: active low flag at a parity check location.

8.2. Timing

The decoder timing diagram is shown in Figure 9. The first stage of the decoding is re-encoding which occupies 1 codeword period. Solving the key equation is carried out at the next stage using the WB algorithm. The allocated time is also 1 codeword period, even though the WB algorithm process only requires 8*t* clock cycles. There are 2*t* check pairs need to be solved, and every pair requires 4 clock cycles to proceed. After completing the algorithm process, the Chien search is carried out to solve the error locator polynomial and then the error value is computed to recover the corrupted message.

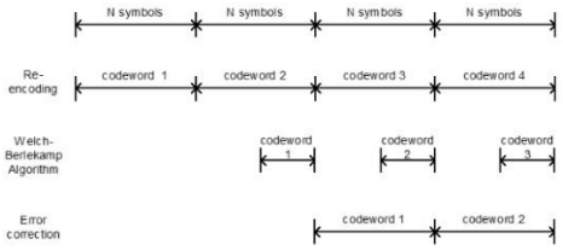

Figure 9. The timing diagram of *the RS* Decoder using the WB algorithm.

8.3. Simulation Results

Consider RS(255, 239). Let $c(x)$ be the transmitted message, and $v(x)$ be the received message.

$c(x) = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0\}$
$v(x) = \{1, 2, 4, 8, 16, 32, 64, 128, 0, 0, 0, 0, \ldots, 0\}$

The simulation result was shown in Figure 10 where all zeros became the decoder output. $v(x)$ is represented as *din_ff*, the error value is *value*, the corrected value is *corrected*, and the decoder output is *wb_do*.

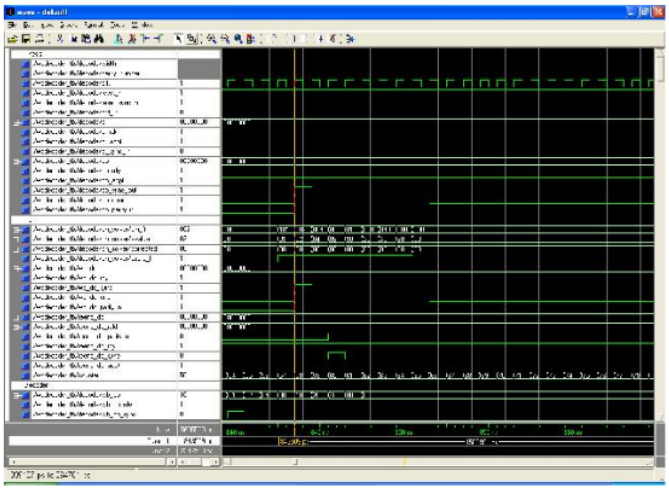

Figure 10. The simulation result for HDD RS(255, 239).

Another example, consider again RS(255, 239). Let $I(x)$ be the information message before encoding at the value of {0x0F, 0x0F, 0x0F, …, 0x0F}. Then the encoded transmitted message will be $c(x)$ = {0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, …, 0x0F, 0xA9, 0xCC, 0x91,

0x43, 0x2C, 0x7C, 0x63, 0x7B, 0x22, 0x55, 0x01, 0x87, 0xF1, 0x2B, 0xB1, 0x6A}.

Let $e(x)$ be the error value, and $v(x)$ be the received message, then

$e(x)$ = {0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0, …, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}. $v(x) = c(x) + e(x)$ = {0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0F, …, 0x0F, 0xA9, 0xCC, 0x91, 0x43, 0x2C, 0x7C, 0x63, 0x7B, 0x22, 0x55, 0x01, 0x87, 0xF1, 0x2B, 0xB1, 0x6A}.

The decoder output *do* is shown in Figure 11 and this value is valid when *do_rdy* is set to logic '1'. Figure 12 shows that the decoder output *do* is equal to the encoded, transmitted message $c(x)$.

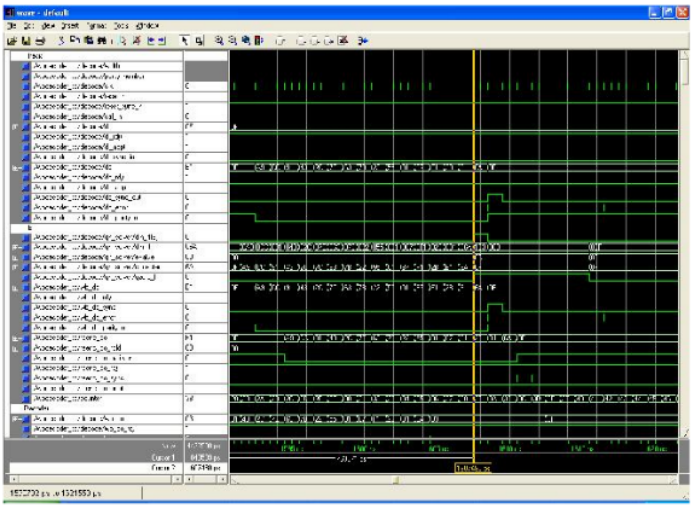

Figure 11. The simulation result for HDD RS(255, 239).

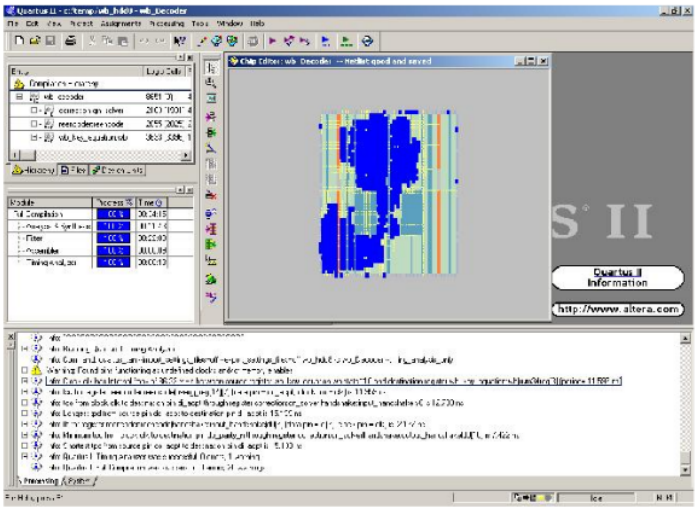

Figure 12. The synthesized result for HDD RS(255, 239).

8.4. Synthesis Results

The VHDL source code was then synthesized using *Quartus II* software. The Altera Stratix EP1S25-B672C6 was selected to be the target device with a capacity of 25660 logic element (LE). The design utilized 8651 out of 25660 (33% coverage) logic element at maximum clock frequency 86.23 MHz. A synthesize result of HDD RS(255, 239) is shown in Figure 12, where the solid black part is the utilized area.

CONCLUSION

We have presented the Reed-Solomon (RS) code as one of many error control coding schemes to enhance the security of data and information exchanges along the internet connections in the Internet of Things. Reed-

Solomon code has been discussed in details. RS decoding scheme using the Welch Berlekamp algorithm has been highlighted. The VHDL implementation of the Reed Solomon decoder is also presented. Without loss of generality, the Hard Decision Decoding for RS(255, 239) over GF(28) is developed using VHDL.

REFERENCES

[1] Meutia, Ernita Dewi. "Internet of things–keamanan dan privasi." *Semin. Nas. dan Expo Tek. Elektro* 2015 (2015): 85-89.

[2] Qiang, Chen, Guang-ri Quan, Bai Yu, and Liu Yang. "Research on security issues of the internet of things." *Issues* 371 (2016).

[3] Tiwari, A. and Maurya, H. (2017). Challenges and Ongoing Researches for IoT (Internet of Things): A Review. *International Journal of Engineering Development and Research*, vol 5, issue 2, 57-60.

[4] Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Tech Journal,* 27: 379-423.

[5] Reed, I. S., and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal Society. Industrial Applied Maths*, 8: 300-304.

[6] Chang, H. C. and Shung, C. B. (1998). A (208, 192; 8) Reed Solomon Decoder for DVD Application. *IEEE International Conference On Communications: Conf. Record*, affiliated with Supercomm, 2: 957-960.

[7] Hoeve, H., Timmermans, J., and Vries, L. B. (1982). Error correction and concealment in the compact disc system. *Phillips Tech. Rev*, 40, 6: 166-172.

[8] Im, Y. and Kwon, O. S. (1997). An Advanced VLSI Architecture of RS Decoders for Advanced TV. *IEEE International Conference On. Communications: Towards the Knowledge Millenium ICC'97*, 3: 1346-1350.

[9] Kwon, S. and Shin, H. (1997). An Area Efficient VLSI Architecture of a Reed-Solomon Decoder/Encoder for Digital VCRs. *IEEE Transactions on Consumer Electronics*, 43-4: 1019-1027.

[10] Shayan, Y. R., Le-Ngoc, T., and Bhargava, V. K. (1990). Design of RS (16, 12) Codec for North American Advanced Train Control System. *IEEE Transactions of Vehicular Tech*, vol. 39, no 4.

[11] Blahut, R. E. (1983). *Theory and Practice of Error Control Codes*. Reading MA, Addison-Wesley.

[12] Clark, G. C. and Cain, J. B. (1981). *Error-Correction Coding for Digital Communications*. Applications of Communications Theory, Plenum Press, New York.

[13] Berlekamp, E. R. (1982). Bit Serial Reed-Solomon Encoders. *IEEE Trans. Inform. Theory*, 28: 869-874.

[14] Bossert, M. (1999). *Channel Coding for Telecommunications*. John Wiley Publisher.

[15] Hsu, I. S., Reed, I. S., Truong, T. K., Wang, K., Yeh, C. S. and Deutsch, L. J. (1984). The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit Serial Multiplier algorithm. *IEEE Trans. Comp*, vol. 33.

[16] Liu, K. Y. (1984). Architecture for VLSI Design of a Reed-Solomon Decoder. *IEEE Trans. Comp*, vol. 33.

[17] Bhargava, W. *Reed-Solomon Codes and Their Applications*. IEEE press.

[18] Blahut, R. E. (1979). Transform Technique for Error Control Codes. *IBM Journal Res. Develop.*, vol 23 no 3, 299-315.

[19] Blahut, R. E. (1984). A Universal Reed-Solomon Decoder. *IBM Journal Res. Develop.*, vol 28 no 2, pp 150-158.

[20] Shao, H. M., Truong, T. K., Deutsch, L. J., Yuen, J. H. and Reed, I. S. (1985). A VLSI Design of a Pipeline Reed-Solomon Decoder. *IEEE Trans on Computers*, Vol C34, no. 5, pp. 393-403.

[21] Brent, R. P. and Kung, H. T. (1984). Systolic VLSI Arrays for Polynomial GCD Computation. *IEEE Trans. on Computers*, vol 33 no 8, pp 731-736.

[22] Dabiri, D. and Blake, I. F. (1995). Fast Parallel algorithms for Decoding Reed-Solomon Codes Based on Remainder Polynomials. *IEEE Trans. on Inform. Theory*, vol. 41 no 4, pp. 873-885.

[23] Fitzpatrick, P. (1996). On The Scalar Rational Interpolation Problem. *Mathematics of Control, Signals, and Systems*, vol. 9 part 4, pp 352-369.

[24] Blackburn, S. R. and Chambers, W. G. (1996). Some Remarks on an algorithm of Fitzpatrick. *IEEE Trans. on Inform. Theory*, vol 42 no 4, pp 1269-1271.

[25] Massey, J. L. (1969). Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122-127.

[26] Welch, L. and Berlekamp, E. R. (1986). *Error Correction for Algebraic Block Codes*. US Patent 4,633,470.

[27] Morii, M. and Kasahara, M. (1992). Generalized Key Equation of Remainder Decoding algorithm for Reed Solomon Codes. *IEEE Trans. Inform. Theory*, vol. IT 38, 1801-1807.

[28] Chambers, W. G. (1993). A solution of Welch-Berlekamp Key Equation by Euclidean algorithm. *Electronic Letters*, vol 29 no 11, pp 1031.

[29] Chambers, W. G., Peille, R. E., Tsie, K. Y. and Zein, N. (1993). Algorithm for Solving the Welch-Berlekamp Key-Equation, with a Simplified Proof. *Electronic Letters*, vol. 29 no 18, pp 1620-1621.

[30] Peille, R. E. (1994). On the Performance and Complexity of a Generalized Minimum Distance Reed-Solomon Decoding algorithm. *International Journal of Satellite Communications*, vol. 12, pp. 333-359.

[31] Jennings, S. M. (1995). Grobner Basis View of Welch-Berlekamp algorithm for Reed-Solomon Codes. *IEE Proc. Commun.*, vol. 142 no 6, pp 349-351.

[32] Sorger, U. K. (1993). A New Reed-Solomon Code Decoding algorithm Based on Newton's Interpolation. *IEEE Trans. Inform. Theory*, vol. 39, no 2, pp. 358-365.

[33] Chien, R. T. (1964), Cyclic Decoding Procedures for BCH Codes. *IEEE Trans. Inform. Theory*, vol. 10, pp. 357-363.

[34] Mahmudi, Ali, and Sentot Achmadi. "Modified Welch Berlekamp Algorithm to Decode Reed Solomon Codes." In *MATEC Web of Conferences*, vol. 164, p. 01003. EDP Sciences, 2018.

reed Solomon Codes for Reliable Communication in Internet of Things

ORIGINALITY REPORT

7%	5%	9%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	R. E. Peile. "On the performance and complexity of A generalized minimum distance reed–solomon decoding algorithm", International Journal of Satellite Communications, 07/1994 Publication	2%
2	scholarbank.nus.edu.sg Internet Source	2%
3	Ariyawan Sunardi, Fahmi Islami Suud, N Woro Agus, Indra Gunawan. "IoT Application on Aquaponics System Energy Optimization", Journal of Physics: Conference Series, 2021 Publication	2%
4	archive.org Internet Source	2%

Exclude quotes	Off	Exclude matches	< 2%
Exclude bibliography	Off		