Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 13, Number 9 (2017), pp. 5083–5099

http://www.ripublication.com/gjpam.htm

Decoding of the Seven-Error-Correcting Binary Quadratic Residue Codes

Renuka Sahu and B.P. Tripathi
Department of Mathematics,
Govt. N.P.G.College of Science, Raipur (C.G.), India.

Abstract

In this paper, fast syndrome-weight decoding algorithm (FSWDA) is proposed to decode up to seven possible errors in a binary systematic quadratic residue (QR) codes (79, 40, 15) and (97, 49, 15). The main conception of FSWDA is predicated on the property of cyclic codes together with the weight of syndrome difference. In decoding of the QR codes, the evaluation of the error-locator polynomial in the finite field is complicated and time-consuming. To deal with such an issue, our scheme FSWDA keeps away from evaluating the error locator polynomial and has no need to generate the table which store the syndromes and their corresponding patterns of error in the memory. Also, our scheme serve as an efficient and high speed decoder.

AMS subject classification: 94A60.
Keywords: Quadratic Residue Codes, Cyclic Codes, Decoding.

1. Introduction

The seven-error-correcting binary QR codes include (79, 40, 15) and (97, 49, 15) QR code, respectively. The ADA's for this codes were initially presented by Chang et al. [8] and they asserted that an exhaustive computer simulation was executed successfully. He proposed a ADAs, in which the coefficients of the error-locator polynomial are acquired, the error positions can be determined by using the Chien search algorithm [3], which is an comprehensive inquiry over every one of the components in the limited field. In the decoding strategy of ADAs, this step is the most time-consuming and need a lot of multiplication and division operations over the finite field.

Most recently, table-lookup decoding algorithms (TLDAs) [9], [5], [6] have assumed an imperative part in forward error correction. These sorts of decoders are efficient with minimum decoding delay; however, the TLDAs require a memory space in the decoder chip and increase the decoding cost rapidly when the code length is large. The Efficient syndrome weight decoding algorithm (ESWDAs) proposed by Chang et al. [1] is displayed to iterpret up-to triple possible errors for the binary Golay code and (31, 16, 7) QR code. At that point, a new efficient syndrome-weight decoding algorithm (NESWDA) [10] is presented to decode up to five possible errors in a binary systematic (47, 24, 11) QR code.

In this paper, we will proposed a novel storage-efficient FSWDA to decode the long binary systematic (79, 40, 15) and (97, 49, 15) QR code. As shown in this paper, the proposed FSWDA does not need a memory size to store the look-up table. The prime idea of the proposed FSWDA is based on the weight of syndrome difference between the syndrome of the received word and the row vector of the transpose of the parity-check matrix. Moreover, no complicated computation in the finite field is required in the proposed FSWDA and it also can be extended to decode all seven-error-correcting binary QR codes.

The remainder of this paper is organized as follows: The preliminary of the binary QR codes is briefly given in Section 2. The proposed FSWDA is described in Section 3 with one example to demonstrate the proposed FSWDA. Finally, this paper concludes with a brief summary in Section 4.

2. Preliminary

2.1. Binary Code

The information to be sent is transmitted by a sequence of zeros and ones called Binary Codes, which means that the code is defined on the field 0,1.

2.2. Binary Quadratic Residue Code [8]

A binary QR code (n, k, d) or $(n, (n+1)/2, d)$ with minimum distance d is defined algebraically as a multiple of its generator polynomial $g(x)$ over $GF(2^m)$, where $k = (n+1)/2$ is the message length and n is the code length. Let n be a prime number of the form $n = 8m \pm 1$, where m is a positive integer and m be the smallest positive integer such that $2^m \equiv 1(\bmod n)$. The set Q_n of quadratic residues modulo n is the set of nonzero squares modulo n; that is, $Q_n = \{j \mid j \equiv x^2 \bmod n, 1 \leq x \leq (n-1)/2\}$.

Let the symbol C_{79} and C_{97} denote the binary (79, 40, 15) and (97, 49, 15) QR code respectively.

Let α be a root of primitive irreducible polynomial $p_{79}(x) = x^{39} + x^4 + 1$ and $p_{97}(x) = x^{48} + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$ such that α is a generator of the multiplicative group of all nonzero elements in $GF(2^{39})$ and $GF(2^{48})$ respectively. Then, the element $\beta = \alpha^u$.

The set of quadratic residues modulo are

$$Q_{79} = \{1, 2, 4, 5, 8, 9, 10, 11, 13, 16, 18, 19, 20, 21, 22, 23, 25, 26, 31, 32, 36, 38, 40, 42, 44, 45, 46, 49, 50, 51, 55, 62, 64, 65, 67, 72, 73, 76\} \quad (1)$$

$$Q_{97} = \{1, 2, 3, 4, 6, 8, 9, 11, 12, 16, 18, 22, 24, 25, 27, 31, 32, 33, 35, 36, 43, 44, 47, 48, 50, 53, 54, 61, 62, 64, 65, 66, 70, 72, 73, 75, 79, 81, 85, 86, 88, 89, 91, 93, 94, 95, 96\} \quad (2)$$

he generator polynomial of long binary systematic (79, 40, 15) and (97, 49, 15) QR code are

$$g_{79}(x) = \prod_{i \in Q_{79}} (x - \beta_i) = x^{39} + x^{36} + x^{35} + x^{31} + x^{30} + x^{29} + x^{27} + x^{26} + x^{25} + x^{24} + x^{21} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{13} + x^{11} + x^5 + x^4 + x^2 + x + 1 \quad (3)$$

$$g_{97}(x) = \prod_{i \in Q_{97}} (x - \beta_i) = x^{48} + x^{47} + x^{46} + x^{45} + x^{44} + x^{41} + x^{36} + x^{35} + x^{33} + x^{32} + x^{30} + x^{29} + x^{25} + x^{24} + x^{23} + x^{19} + x^{18} + x^{16} + x^{15} + x^{13} + x^{12} + x^7 + x^4 + x^3 + x^2 + x + 1 \quad (4)$$

For the binary (79, 40, 15) and (97, 49, 15) QR codes, the error-correcting capability $t = \lfloor (d-1)/2 \rfloor = \lfloor (15-1)/2 \rfloor = 7$ errors, where x denotes the greatest integer less than or equal to x, and $d = 15$ is the minimum Hamming distance of the code.

2.3. Matrix Representation of Binary QR Codes [2]

A codeword of binary QR code is a polynomial $c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$ such that it is a multiple of the generator polynomial $g(x)$. If the codeword $c(x)$ is transmitted through a noisy channel, then the received polynomial $r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1} x^{n-1}$ can be expressed as the sum of the codeword polynomial $c(x)$ and the error polynomial $e(x) = e_0 + e_1 x + e_2 x^2 + \cdots + e_{n-1} x^{n-1}$. For simplicity, let the message or information, codeword, error pattern, received word, and syndrome be expressed as the binary vector forms $m = (m_0, m_1, \ldots, m_{n-1})$, $c = (c_0, c_1, \ldots, c_{n-1})$, $e = (e_0, e_1, \ldots, e_{n-1})$, respectively. The systematic codeword of the vector form is given by $c = mG$, where G is called the systematic generator matrix.

For the binary systematic (79, 40, 15) QR code, the systematic $k \times n = 79 \times 40$

generator matrix G can be expressed as follows:

$$\mathbf{G} = \lfloor \mathbf{A}_{k\times(n-k)} | \mathbf{I}_k \rfloor_{k\times n} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,38} & 1 & 0 & \cdots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,38} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{39,0} & p_{39,1} & \cdots & p_{39,38} & 0 & 0 & \cdots & 1 \end{bmatrix}_{40\times 39} \tag{5}$$

where $\mathbf{I}_k$ is the $k \times k = 40 \times 40$ identity matrix and $\mathbf{A}_{k\times(n-k)}$ is the matrix of size $k \times (n-k) = 40 \times 39$. The codeword of systematic form can be obtained in matrix form by

$$\mathbf{c} = \mathbf{mG} = (m_{39} \ldots m_1 m_0)\mathbf{G} = (p_{38} \ldots p_1 p_0 | m_{35} \ldots m_1 m_0), \tag{6}$$

where the $(p_{38} \ldots p_1 p_0)$ is the parity-check part of $\mathbf{c}$ and the $(m_{39} \ldots m_1 m_0)$ is the message part of $\mathbf{c}$. The systematic parity check matrix $\mathbf{H}$ of size $(n-k) \times n = 39 \times 79$ can be expressed as follows:

$$\mathbf{H} = \lfloor \mathbf{I}_{n-k} | \mathbf{A}^T_{k\times(n-k)} \rfloor_{(n-k)\times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,0} & p_{1,0} & \cdots & p_{39,0} \\ 0 & 1 & \cdots & 0 & p_{0,1} & p_{1,1} & \cdots & p_{39,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{0,38} & p_{1,38} & \cdots & p_{39,38} \end{bmatrix}_{39\times 79} \tag{7}$$

where $\mathbf{A}^T_{k\times(n-k)}$ is the transpose matrix of $\mathbf{A}_{k\times(n-k)}$ and $\mathbf{I}_{n-k}$ is the $(n-k) \times (n-k) = 39 \times 39$ identity matrix.

For C_{79}, matrix $\mathbf{A}_{k\times(n-k)}$ has the following form:

$$A = \begin{bmatrix}
1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0\\
0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0\\
0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1\\
1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1\\
1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0\\
0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0\\
0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0\\
0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1\\
1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1\\
1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1\\
1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0\\
0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1\\
1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1\\
1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1\\
1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1\\
1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0\\
0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0\\
0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1\\
1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1\\
1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1\\
1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1\\
1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0\\
0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1\\
1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1&0\\
0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1&1\\
1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0&1\\
1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1&0\\
0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&1\\
1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0&0\\
0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0&0\\
0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0&0\\
0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0&0\\
0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1&0\\
0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1&1\\
1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0&1\\
1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1&0\\
0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1&1\\
1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1&1\\
1&1&0&1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&1&1&1&0&0&1&1&1&1&0&1&1&1&0&0&0&1&1&0&0&1\\
0&0&1&1&0&1&0&0&0&1&0&1&1&0&1&1&1&0&1&1&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0
\end{bmatrix}$$

The vector form of the syndrome can be defined by

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = \mathbf{r}\begin{bmatrix}
1&0&\cdots&0&p_{0,0}&p_{1,0}&\cdots&p_{39,0}\\
0&1&\cdots&0&p_{0,1}&p_{1,1}&\cdots&p_{39,1}\\
\vdots&\vdots&\ddots&\vdots&\vdots&\vdots&\ddots&\vdots\\
0&0&\cdots&1&p_{0,38}&p_{1,38}&\cdots&p_{39,38}
\end{bmatrix}^T, \tag{8}$$

where $\mathbf{H}^T$ denotes the $n \times (n-k) = 79 \times 39$ transpose matrix of $\mathbf{H}$; that is, $\mathbf{H}^T = [\mathbf{I}_{39}|\mathbf{A}_{40\times 39}]_{79\times 39}$.

The proposed algorithm only needs to compute the syndrome of **r** for every received word and the complicated computation of the error-locator polynomial given in the ADA can be completely avoided. If **r** occurs with no error, then the syndrome $\mathbf{s} = \mathbf{r}\mathbf{H}^T = (\mathbf{c} + 0)\mathbf{H}^T = \mathbf{c}\mathbf{H}^T = 0$, where 0 denotes a zero vector. Otherwise, $\mathbf{s} = \mathbf{r}\mathbf{H}^T = (\mathbf{c} + e)\mathbf{H}^T = 0 + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$.

2.4. Hamming weight

[7] The Hamming Weight of a binary vector **a** is denoted by $w(\mathbf{a})$, and the Hamming distance between **a** and **b** is denoted by $d(\mathbf{a}, \mathbf{b}) = w(\mathbf{a} + \mathbf{b})$.

3. Existing Results

Following theorems are the results proofed by various authors, which are useful in our proposed algorithm.

Theorem 3.1. [7] Let $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$ be two binary vectors, then

$$w(\mathbf{a} + \mathbf{b}) = w(\mathbf{a}) + w(\mathbf{b}) - 2\sum_{i=0}^{n-1} a_i b_i$$

if $a_i b_i = 0$ for $0 \leq i \leq n-1$, then

Corollary 3.2.

$$w(\mathbf{a} + \mathbf{b}) = w(\mathbf{a}) + w(\mathbf{b}).$$

The following Theorem is useful to compute the syndrome of the received word when the received word shifts one bit to the right.

Theorem 3.3. [4] Let $s(x)$ be the syndrome polynomial corresponding to a received polynomial $r(x)$. Also, let $r^{(1)}(x)$ be the polynomial obtained by cyclically shifting the coefficients of $r(x)$ one bit to the right. Then the remainder obtained when dividing $xs(x)$ by $g(x)$ is the syndrome $s^{(1)}(x)$ corresponding to $r^{(1)}(x)$. However, if the syndrome cyclically shifts many times, then the syndrome computation is quite time-consuming for dividing $xs(x)$ by $g(x)$ many times. The following theorem provides an efficient method to compute $s^{(i)}$ for $0 \leq i \leq n-1$, and it can save a lot of computational time.

Theorem 3.4. [1] For the binary QR codes, let r_j be an element of r and h_j be the $j-th$ row vector of H^T for $0 \leq j \leq n-1$. Then the syndrome $s^{(i)}$ of $r^{(i)}$ for $0 \in i \in n-1$ has the form

$$\mathbf{s}^{(i)} = \sum_{j=0}^{n-1} r_j h_{[i+j]},$$

where the suffix [x] of h denotes x mod n.

Theorem 3.3. reveals that the syndrome of $\mathbf{r}^{(i)}$ can be fast computed by the vector addition. Theorem 3.4 also provides an efficient method to simplify the decoding step by using the syndrome weight.

Theorem 3.5. [10] For the binary QR codes, it is assumed that there are v errors in the received word, where $1 \leq v \leq t$ and $t = \lfloor (d-1)/2 \rfloor$. All v errors are in the parity-check bits if and only if the weight of syndrome $w(s) = v$.

Theorem 3.6. [6] For the binary QR codes, if v errors are in the information bits of the received word, where $1 \leq v \leq t$ and $t = \lfloor (d-1)/2 \rfloor$, then the weight of the corresponding syndrome vector satisfies

$$w(s(x)) \geq d - v \text{ or } w(\mathbf{s}) \geq (d - v)$$

theorem 3.7. [10] For the binary QR codes, let $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ be an error pattern and $\mathbf{e}_m = (e_0, \ldots, e_{k-1})$, $\mathbf{e}_p = (e_k, \ldots, e_{n-1})$ be respectively its message section and parity check section. Assume that $w(e_m) \geq 1$, $w(e_p) \geq 1$ and $w(e) \geq t$, where $t = \lfloor (d-1)/2 \rfloor$, then the weight of the corresponding syndrome vector satisfies

$$w(\mathbf{s}) \geq t + 1$$

Given a received word $\mathbf{r}$, the syndrome $\mathbf{s}^{(i)}$ of $\mathbf{r}^{(i)}$ can be fast computed by theorem 3.3. According to theorem 3.4, if $1 \leq w(\mathbf{s}) \leq 7$ then error positions are in the parity-check bits of $\mathbf{r}$.

If $1 \leq w(\mathbf{s}^{(k)}) \leq 7$, then the error positions are in the information bits of $\mathbf{r}$. Let $\mathbf{h}_j$ denote the $j-th$ row vector of $\mathbf{H}^T$, where $0 \leq j \leq n-1$. Also let sd_φ denote the syndrome difference between the syndromes of $\mathbf{r}$ and $\mathbf{h}_j$ in each decoding step φ. By using the weight of sd_φ, the error cases can be quickly determined.

4. Proposed fast syndrome weight decoding algorithm (FSWDA) algorithm

Let $u_0 = (1, 0, \cdots, 0)$ be a k-tuples unit vector and u_i has only one nonzero component at the $i-th$ position, where $0 \leq i \leq k-1$. By using these properties the proposed FSWDA can be constructed.Let case **H**, **C**, **P** denote the error position in the information bits, center bit, and parity-check bits of $\mathbf{r}$, respectively.

The decoding steps of the proposed FSWDA work as follows:

1. (No error, **P**, **PP**, **PPP**, **PPPP**, **PPPPP**, **PPPPPP** and **PPPPPPP**cases)
   First compute $\mathbf{s}$ and $w(\mathbf{s})$ by the use of **theorem 2.3**
   If $0 \leq w(\mathbf{s}) \leq 7$, then the information vector is
   $\mathbf{m} = (r_0, r_1, \ldots, r_{k-1})$. Go to step (10).

2. (**H**, **HH**, **HHH**, **HHHH**, **HHHHH**, **HHHHHH** and **HHHHHHH** cases)
   Compute $\mathbf{s}^k$ and $w(\mathbf{s}^{(k)})$ by the use of **theorem 2.3**

Steps	Cases	Number of Error patterns		Steps	Cases	Number of Error patterns	
		C_79	C_97			C_79	C_97
1	P	39	97	5	HHPP	549081	21678336
	PP	741	4656		HHPPP	6771999	686480640
	PPP	9139	147440		HHPPPP	60947991	16132295040
	PPPP	82251	3464840		HHPPPPP	426635937	3.00006E+12
	PPPPP	575757	64446024		HCP	1521	9409
	PPPPPP	3262623	988172368		HCPP	28899	451632
	PPPPPPP	15380937	12846240784		HCPPP	356421	14301680
	TOTAL	19311487	13902476209		HCPPPP	3207789	336089480
2	H	39	97		HCPPPPP	22454523	6251264328
	HH	741	4656		TOTAL	520954161	3.0235E+12
	HHH	9139	147440	6	HHCP	28899	451632
	HHHH	82251	3464840		HHHCP	356421	14301680
	HHHHH	575757	64446024		HHHHCP	3207789	336089480
	HHHHHH	3262623	988172368		HHHHHCP	22454523	6251264328
	HHHHHHH	15380937	12846240784		HHHPP	6771999	686480640
	TOTAL	19311487	13902476209		HHHHPP	60947991	16132295040
3	HP	1521	9409		HHHHHPP	426635937	3.00006E+12
	HPP	28899	451632		TOTAL	520403559	3.02348E+12
	HPPPP	356421	14301680	7	HHCPP	549081	21678336
	HPPPPP	3207789	336089480		HHCPPP	6771999	686480640
	HPPPPPP	22454523	6251264328		HHCPPPP	60947991	16132295040
	HPPPPPPP	127242297	95852719696		TOTAL	68269071	16840454016
	C	1	1	8	HHHCPP	6771999	686480640
	CP	39	97		HHHHCPP	60947991	16132295040
	CPP	741	4656		TOTAL	67719990	16818775680
	CPPP	9139	147440	9	HHHCPPP	83521321	21738553600
	CPPPP	82251	3464840		SUM	3200838455	
	CPPPPP	575757	64446024				
	CPPPPPP	3262623	988172368				
	TOTAL	157222001	1.03511E+11				
4	HC	39	97				
	HHC	741	4656				
	HHHC	9139	147440				
	HHHHC	82251	3464840				
	HHHHHC	575757	64446024				
	HHHHHHC	3262623	988172368				
	HHP	28899	451632				
	HHHP	356421	14301680				
	HHHHP	3207789	336089480				
	HHHHHP	22454523	6251264328				
	HHHHHHP	127242297	95852719696				
	TOTAL	157220479	1.03511E+11				

Figure 1: List all the 61 error cases and the number of error patterns in each decoding steps of the proposed FSWDA.

If $1 \leq w(\mathbf{s}^{(k)}) \leq 7$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \cdots, r_{k-1}) + (s^{(k)}, 0)$. Go to step (10).

3. (**HP**, **HPP**, **HPPP**, **HPPPP**, **HPPPPP**, **HPPPPPP**, **C**, **CP**, **CPP**, **CPPP**, **CPPPP**, **CPPPPP** and **CPPPPPP** cases)
Compute the syndrome difference $\mathbf{sd}_3 = \mathbf{s} - \mathbf{h}$ for $0 \leq i \leq k-1$ and $w(\mathbf{sd}_3)$.
If $w(\mathbf{sd}_3) \leq 6$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_i$. Go to step (10).

4. (**HC**, **HHC**, **HHHC**, **HHHHC**, **HHHHHC**, **HHHHHHC**, **HP**, **HHP**, **HHHP**, **HHHHP**, **HHHHHP** and **HHHHHHP**)
Compute the syndromes $\mathbf{sd}_4 = \mathbf{s}^k - \mathbf{h}_i$ for $0 \leq i \leq k-1$ and $w(\mathbf{sd}_4)$.
If $w(\mathbf{sd}_4) \leq 6$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_{k-1} + (\mathbf{sd}_4, 0)$ and
if $w(\mathbf{sd}_4) \leq 6$ and $0 \leq i \leq k-1$,then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + (\mathbf{sd}_4, 0)$. Go to step (10).

5. (**HHPP**, **HHPPP**, **HHPPPP**, **HHPPPPP**, **HCP**, **HCPP**, **HCPPP HCPPPP** and **HCPPPPP** cases)

Compute the syndromes difference $\mathbf{sd}_5 = \mathbf{s} - (\mathbf{h}_i + \mathbf{h}_j)$ for $0 \leq i < j \leq k-1$ and $w(\mathbf{sd}_5)$.
If $w(\mathbf{sd}_5) \leq 5$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_i + u_j$. Go to step (10).

6. (**HHCP**, **HHHCP**, **HHHHCP**, **HHHHHCP**, **HHHPP**, **HHHHPP** and **HHHHHHPP** cases)
Compute the syndromes difference $\mathbf{sd}_6 = \mathbf{s}^k - (\mathbf{h}_i + \mathbf{h}_j)$ for $0 \leq i < j \leq k-1$ and $w(\mathbf{sd}_6)$.
If $w(\mathbf{sd}_6) \leq 5$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_{k-1} + (\mathbf{sd}_6, 0)$ and
if $w(\mathbf{sd}_6) \leq 6$ and $i > 0$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + (\mathbf{sd}_6, 0)$. Go to step (10).

7. (**HHCPP**, **HHCPPP** and **HHCPPPP** cases)
Compute the syndromes difference $\mathbf{sd}_7 = \mathbf{s} - (\mathbf{h}_{k-1} + \mathbf{h}_i + \mathbf{h}_j)$ for $0 \leq i < j \leq k-2$ and $w(\mathbf{sd}_7)$.
If $w(\mathbf{sd}_7) \leq 4$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_{k-1} + u_i + u_j$. Go to step (10).

8. (**HHHCPP** and **HHHHCPP** cases)
Compute the syndromes difference $\mathbf{sd}_8 = \mathbf{s}^k - (\mathbf{h}_{k-1} + \mathbf{h}_i + \mathbf{h}_j)$ for $0 \leq i < j \leq k-2$ and $w(\mathbf{sd}_8)$.
If $w(\mathbf{sd}_8) \leq 4$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_{k-2} + (\mathbf{sd}_8, 0)$ and

if $w(\mathbf{sd}_8) \leq 4$ and $i > 0$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + (\mathbf{sd}_8, 0)$. Go to step (10).

9. (**HHHCPPP** case)
Compute the syndromes difference $\mathbf{sd}_9 = \mathbf{s} - (+\mathbf{h}_{k-2} + \mathbf{h}_{k-1} + \mathbf{h}_i + \mathbf{h}_j)$ for $0 \leq i < j \leq k-3$ and $w(\mathbf{sd}_9)$.
If $w(\mathbf{sd}_9) \leq 3$, then the corrected information vector is
$\mathbf{m} = (r_0, r_1, \ldots, r_{k-1}) + u_{k-2} + u_{k-1} + u_i + u_j$. Go to step (10)

10. stop.

4.1. Example

Let a information m = (0100000000000000000000000000000000000000) be encoded into a C_{79} codeword

c = (0100100000000000000000000000000000000000001110000000000001100000101010110001010 0).

If the received word
r = (1111000000000000000000000000000000000000001110000000000001100000101010110001010 0),

then the error pattern
e = (1011100000000000000000000000000000000000000000000000000000000000000000000000000000),

which means a **HHHCPP** case. The decoding procedure is composed of the following steps:

1. Compute s = 0100000000000000110000010101011000101 00 and then $w(s) = 10 > 7$, go to step 2.
2. Compute s^{40} = 0010011011001000110101100111111001110 01 and $w(s^{40}) = 21 > 7$, go to step 3.
3. Compute $sd_3 = s - h_i$ for $0 \leq i \leq 39$ and $w(sd_3)$.

$sd_3 = s - h_0$
$= (0100000000000000110000010101011000101 00) - (1110110000010110101111001111011100011 00)$
$= (1010110000010110011111011010000100110 00).$

$$w(sd_3) = 18.$$

$sd_3 = s - h_1$
$= (0100000000000000110000010101011000101 00) - (0111011000001011010111100111101110001 10)$
$= (0011011000001011100111110010110110100 10).$

$$w(sd_3) = 20.$$

$$\ldots$$

$sd_3 = s - h_{12}$
$= (0100000000000000110000010101011000101 00) - (1011100011001110110000010110101111001 11)$
$= (1111100011001110000000000011110111100 11).$

$$w(sd_3) = 20.$$

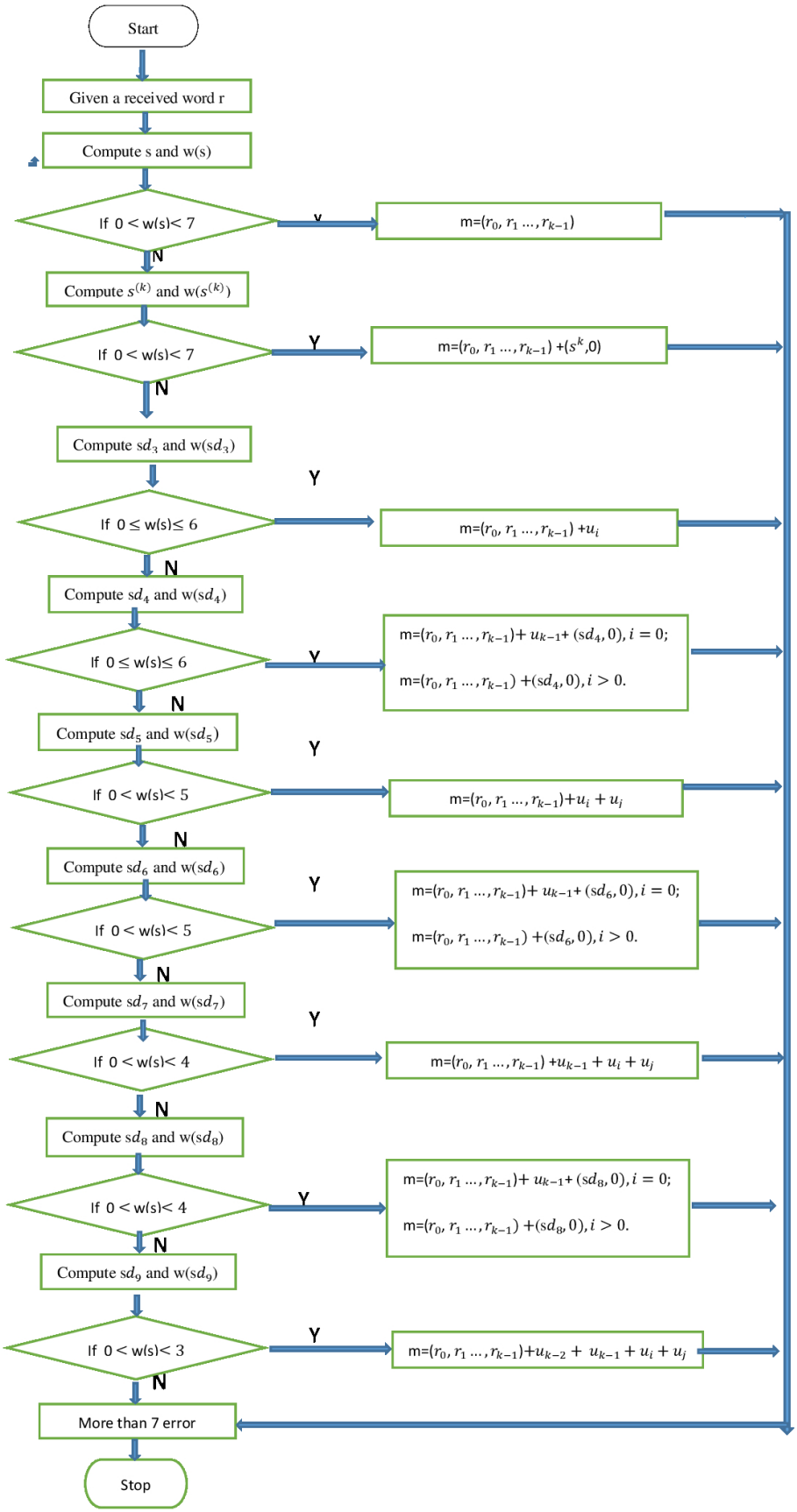

Figure 2: The flowchart of the proposed FSWDA

...

$$\begin{aligned} sd_3 &= s - h_{24} \\ &= (0100000000000000011000001010101100010100) - (0101111001111011100011001111011000001011) \\ &= (0001111001111011010011011011101000011111). \end{aligned}$$

$$w(sd_3) = 24.$$

...

$$\begin{aligned} sd_3 &= s - h_{35} \\ &= (0100000000000000011000001010101100010100) - (1100000101101011110011110111000110011 10) \\ &= (1000000101101011000011100010011110110 10). \end{aligned}$$

$$w(sd_3) = 18$$

...

$$\begin{aligned} sd_3 &= s - h_{39} \\ &= (0100000000000000011000001010101100010100) - (0001001010010011011001100011011011111011) \\ &= (0110101010010011101001110110000011011 11). \end{aligned}$$

$$w(sd_3) = 21.$$

since every $w(sd_3) > 7$, go to step 4.

4. Compute $(sd_4) = s^{40} - h_0$ for $0 \leq i \leq 6$ and $w(sd_4)$.

$$\begin{aligned} sd_4 &= s^{40} - h_0 \\ &= (0010011011001000110101100111111001110 01) - (1011000000000000000000000000000000000000) \\ &= (1100101011011110011010101000100101101 01). \end{aligned}$$

$$s(sd_4) = 21.$$

$$\begin{aligned} sd_4 &= s^{40} - h_1 \\ &= (0010011011001000110101100111111001110 01) - (0111011000001011010111100111101110000 10) \\ &= (0101000011000011100010000000010111111 11). \end{aligned}$$

$$w(sd_4) = 17.$$

...

$$\begin{aligned} sd_4 &= s^{40} - h_{15} \\ &= (0010011011001000110101100111111001110 01) - (1111011100011001110110000010110101111 00) \\ &= (1101000111010001000011100101001100000 101). \end{aligned}$$

$$w(sd_4) = 17.$$

...

$$\begin{aligned} sd_4 &= s^{40} - h_{25} \\ &= (0010011011001000110101100111111100111001) - (1010111100111101110001100111011000000101) \\ &= (1000100111110101000100000000010000111100). \end{aligned}$$

$$w(sd_4) = 15.$$

...

$$\begin{aligned} sd_4 &= s^{40} - h_{39} \\ &= (0010011011001000110101100111111100111001) - (0001001010010011011001100011011011111011) \\ &= (0011010001011011101100000100100010000010). \end{aligned}$$

$$w(sd_4) = 15.$$

Since every $s(sd_4) > 6$,go to step 5.

5. Compute $sd_5 = s - (h_i + h_j)$ for $0 \leq i < j \leq 39$ and $w(sd_5)$.

$$\begin{aligned} sd_5 &= s - (h_0 + h_1) \\ &= (0100000000000000011000001010101100010100) - ((1110110000010110101111001111011100011100) \\ &+ (0111011000001011010111100111101110001100)) \\ &= (1101101000011100001000111101101010111110). \end{aligned}$$

$$w(sd_5) = 21.$$

$$\begin{aligned} sd_5 &= s - (h_0 + h_2) \\ &= (0100000000000000011000001010101100010100) - ((0011101100000101101011110011111011100011) \\ &+ (0011101100000101101011110011111011100011)) \\ &= (0100000000000000011000001010101100010100). \end{aligned}$$

$$w(sd_5) = 10.$$

...

$$\begin{aligned} sd_5 &= s - (h_1 + h_2) \\ &= (0100000000000000011000001010101100010100) - ((0111011000001011010111100111101110001100) \\ &+ (0011101100000101101011110011111011100011)) \\ &= (0000011010000111000110000000100000110001). \end{aligned}$$

$$w(sd_5) = 12.$$

...

$$sd_5 = s - (h_1 + h_{39})$$
$$= (0100000000000000011000001010101100010100) - ((0111011000001011010111100111101110001100)$$
$$+ (0001001010010011011001100011011011111011))$$
$$= (0010010010011000111110010001101101010010).$$

$$w(sd_5) = 18.$$

...

$$sd_5 = s - (h_{38} + h_{39})$$
$$= (0100000000000000011000001010101100010100) - ((1101100000101101011110011110111000110011)$$
$$+ (0001001010010011011001100011011011111011)$$
$$= (1000101010111110110111101000111011101100).$$

$$w(sd_5) = 24.$$

since every $w(sd_5) > 5$, go to step 6.

6. Compute $sd_6 = s^{40} - (h_i + h_j)$ for $0 \leq i < j \leq 39$ and $w(sd_6)$.

$$sd_6 = s^{40} - (h_0 + h_1)$$
$$= (0010011011001000110101100111111100111001) - ((1110110000010110101111001111011100011100)$$
$$+ (0111011000001011010111100111101110001100)$$
$$= (1011110011010101001101001111001011100100).$$

$$w(sd_6) = 22.$$

$$sd_6 = s^{40} - (h_0 + h_2)$$
$$= (0010011011001000110101100111111100111001) - ((1110110000010110101111001111011100011100)$$
$$+ (0011101100000101101011110011110111000011))$$
$$= (1111000111011011110001011011010010101100).$$

$$w(sd_6) = 23.$$

...

$$sd_6 = s^{40} - (h_1 + h_2)$$
$$= (0010011011001000110101100111111100111001) - ((0111011000001011010111100111101110001100)$$
$$+ (0011101100000101101011110011110111000011))$$
$$= (0110101111000110001001110011100000011100).$$

$$w(sd_6) = 19.$$

$$sd_6 = s^{40} - (h_1 + h_2)$$
$$= (0010011011001000110101100111111100111001) - ((0111011000001011010111100111101110001100)$$
$$+ (0011101100000101101011110011110111000011))$$
$$= (0110101111000110001001110011100000011100).$$

$$w(sd_6) = 19.$$

...

$$\begin{aligned} sd_6 &= s^{40} - (h_1 + h_{39}) \\ &= (0010011011001000110101100111111100111001) - ((0111011000001011010111100111101110001 10) \\ &+ (0001001010010011011001100011011011111011)) \\ &= (0100001001010000111011100011001100000100). \end{aligned}$$

$$w(sd_6) = 15.$$

...

$$\begin{aligned} sd_6 &= s^{40} - (h_{38} + h_{39}) \\ &= (0010011011001000110101100111111100111001) - ((0111011000001011010111100111101110001 10) \\ &+ (0001001010010011011001100011011011111011)) \\ &= (0100001001010000111011100011001100000101). \end{aligned}$$

$$w(sd_6) = 15.$$

Since every $w(sd_6) > 5$, go to step 7.

7. Compute $sd_7 = s - (h_{k-1} + h_i + h_j)$ for $0 \leq i < j \leq 38$ and $w(sd_7)$.

$$\begin{aligned} sd_7 &= s - (h_{39} + h_0 + h_1) \\ &= (0100000000000000011000001010101100010100) - ((1110110000010110101111001111011100011 00) \\ &+ (0111011000001011010111100111101110001 10) + (0001001010010011011001100011011011111011)) \\ &= (1100100010001110010001011110110001001 01). \end{aligned}$$

$$w(sd_7) = 18.$$

$$\begin{aligned} sd_7 &= s - (h_{39} + h_0 + h_2) \\ &= (0100000000000000011000001010101100010100) - ((1110110000010110101111001111011100011 00) \\ &+ (0011101100000101101011110011110111000 11) + (0001001010010011011001100011011011111011)) \\ &= (1000010110000000010110100101010100000000). \end{aligned}$$

$$w(sd_7) = 12$$

...

$$\begin{aligned} sd_7 &= s - (h_{39} + h_5 + h_{10}) \\ &= (0100000000000000011000001010101100010100) - ((0110011101100000101101011110011111011 00) \\ &+ (1110001100111011000001011010111100111 10) + (0001001010010011011001100011011011111011)) \\ &= (1101011011001000000101110010100001011 01). \end{aligned}$$

$$w(sd_7) = 18.$$

...

$$sd_7 = s - (h_{39} + h_{37} + h_{38})$$
$$= (0100000000000000011000001010101100010100) - ((0001001010010011011001100011011011111011)$$
$$+ (1011000001011010111100111101110001100 11) + (1101100000101101111110011110111000110 01))$$
$$= (0011101011100100001011010101001010 00101).$$

$$w(sd_7) = 18.$$

Since every $w(sd_7) > 4$, go to step 8.

8. Compute $(sd_8) = s^{40} - (h_{k-1} + h_i + h_j)$ for $0 \leq i < j \leq k-2$

$$sd_8 = s - (h_{39} + h_0 + h_1)$$
$$= (0010011011001000110101100111111001110 01) - ((0001001010010011011001100011011011111011)$$
$$+ (1110110000010110101111001111011100011 00) + (0111011000001011110111100111101110000110))$$
$$= (1010111001000110010100101100010000010 00).$$

$$w(sd_8) = 15.$$

$$sd_8 = s - (h_{39} + h_5 + h_{10})$$
$$= (0010011011001000110101100111111001110 01) - ((0001001010010011011001100011011011111011)$$
$$+ (0110011101100000101101011110011110111 00) + (1110001100111011000001011010111100111 10))$$
$$= (1011000000000000000000000000000000000000).$$

Since $w(sd_8) = 3 \leq 4$, and $i > 0$.

The corrected information vector will be $m = (r_0 + r_1 + \cdots + r_{39}) + (sd_8, 0)$.

$$m = (1111000000000000000000000000000000000000) + (1011000000000000000000000000000000000000)$$
$$= (0100000000000000000000000000000000000000).$$

Go to stop.

5. Conclusion

A fast syndrome weight decoding algorithm (FSWDA) is developed to correct binary systematic (79, 40, 15) and (97, 49, 15) quadratic residue (QR) code upto seven errors. Our proposed scheme neither compute complicated algebraic computation nor stores large lookup table in the memory. The main concept behind our proposed FSWDA is based on the property of cyclic code and the weight of syndrome difference. By using **Theorem 2.3, Theorem 2.4, Theorem 2.5** and the weight of $\mathbf{sd}_\varphi$, the error cases can be fastly identified and corrected. The proposed FSWDA can be extended to decode other QR codes.

References

[1] H.P. Lee, and H.C. Chang, Decoding of the Triple-Error-Correcting Binary Quadratic Residue Codes. Automatic Control and Information Sciences 2.1 (2014): 7–12, doi:10.12691/acis-2-1-2.

[2] Pless, Vera (1998), Introduction to the Theory of Error-Correcting Codes (3rd ed.), Wiley Interscience, ISBN 0-471-19047-0

[3] R.T.Chien, "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," IEEE Trans. Inform. Theory, 10(4). 357–363. Oct. 1964, doi>10.1109/TIT.1964.1053699.

[4] S. B. Wicker, "Error Control Systems for Digital Communication and Storage," Prentice Hall, 1995.

[5] T.C. Lin, H.C. Chang, H.P. Lee, and T.K. Truong, "On the decoding of the (24, 12, 8) Golay code," Inform. Sci., 180(23). 472 9–4736. Dec. 2010, DOI: 10.1016/j.ins.2010.08.015.

[6] T.C. Lin, H.P. Lee, H.C. Chang, and T.K. Truong, "A cyclic weight algorithm of decoding the (47, 24, 11) quadratic residue code," Inform. Sci., 197. 215–222. Aug. 2012, DOI: 10.1016/j.ins.2012.02.020.

[7] T.C. Lin, H.P. Lee, H.C. Chang, T.K. Truong, and S.I. Chu, "High speed decoding of the binary (47, 24, 11) quadratic residue code," Inform. Sci., 180(20). 4060–4068. Oct. 2010, doi>10.1016/j.ins.2010.06.022.

[8] Y. Chang, T. K. Truong, I. S. Reed, H. Y. Cheng, and C. D. Lee, "Algebraic Decoding of (71, 36, 11), (79, 40, 15), and (97, 49, 15) Quadratic Residue Codes," IEEE Trans. on Comm., vol. 51, no. 9, pp. 1463–1473, September (2003), DOI: 10.1109/TCOMM.2003.816994.

[9] Y.H. Chen, C.H. Chien, C.H. Huang, T.K.Truong, and M.H. Jing, "Efficient decoding of systematic (23, 12, 7) and (41, 21, 9) quadratic residue codes," J. Inform. Sci. and Eng., 26(5). 1831–1843. Sept. 2010.

[10] Yani Zhang, Xiaomin Bao, Zhihua Yuan, Xusheng Wu. "Decoding of the Five-Error-Correcting Binary Quadratic Residue Codes." American Journal of Mathematical and Computer Modelling. Vol. 2, No. 1, 2017, https://doi: 10.11648/j.ajmcm.20170201.12.