Automatic Control and Information Sciences, 2014, Vol. 2, No. 1, 7-12
Available online at http://pubs.sciepub.com/acis/2/1/2
© Science and Education Publishing
DOI:10.12691/acis-2-1-2

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

Hung-Peng Lee*, Hsin-Chiu Chang

Department of Computer Science and Information Engineering, Fortune Institute of Technology, Kaohsiung, ROC
*Corresponding author: hpl@fotech.edu.tw

Received February 14, 2013; Revised January 17, 2014; Accepted February 09, 2014

Abstract In this paper, a more efficient syndrome-weight decoding algorithm (SWDA), called the enhanced syndrome-weight decoding algorithm (ESWDA), is presented to decode up to three possible errors for the binary systematic (23, 12, 7) and (31, 16, 7) quadratic residue (QR) codes. In decoding of the QR codes, the evaluation of the error-locator polynomial in the finite field is complicated and time-consuming. To solve such a problem, the proposed ESWDA avoids evaluating the complicated error-locator polynomial, and has no need of a look-up table to store the syndromes and their corresponding error patterns in the memory. In comparison with the SWDA developed by Lin-Chang-Lee-Truong (2010), the simulation results show that the ESWDA can serve as an efficient and high-speed decoder.

Keywords: *syndrome, error pattern, Golay code, quadratic residue code*

Cite This Article: Hung-Peng Lee, and Hsin-Chiu Chang, "Decoding of the Triple-Error-Correcting Binary Quadratic Residue Codes." *Automatic Control and Information Sciences*, vol. 2, no. 1 (2014): 7-12. doi: 10.12691/acis-2-1-2.

1. Introduction

The triple-error-correcting binary QR codes include (23, 12, 7) QR code and (31, 16, 7) QR code, respectively. The binary (23, 12, 7) QR code is also called the binary Golay code, which is a perfect code. The Golay code is first introduced by Golay in 1949 [4]. If an overall parity check is used, the rate is exactly 1/2, so that most of the known QR codes are the best-known codes. Among them, the extended (24, 12, 8) Golay code was utilized to act as an error control on the Voyager I and II spacecraft mission, providing clear remote pictures of Jupiter and Saturn [13].

Several algebraic decoding algorithms (ADAs) had been developed to decode the binary Golay code [3,12]. The key idea of decoding the Golay code by using ADA is to compute the unknown syndrome for determining the coefficients of the error-locator polynomial. One of the representative methods is inverse-free Berlekamp-Massey algorithm [10]. In [11], Reed *et al*. developed the ADA of the extended (32, 16, 8) QR code with reducible generator polynomial. However, such an algorithm is quiet complicated. Lin *et al*. [6] thus proposed a modified ADA to reduce the decoding complexity. For ADAs, once the coefficients of the error-locator polynomial are obtained, the error positions can be determined by using the Chien search algorithm [1], which is an exhaustive search over all the elements in the finite field. In the decoding procedure of ADAs, this step is the most time-consuming and need plenty of multiplication and division operations over the finite field.

Most recently, table-lookup decoding algorithms (TLDAs) [2,7,9] have played an important role in forward error correction. These types 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 SWDA proposed by Lin *et al*. [7] used the refined lookup table (RLT) to decode the triple-error-correcting binary Golay code and (31, 16, 7) QR code. For decoding the Golay code, the RLT consists of 42 syndromes and their corresponding coset leaders, and it only needs 168 bytes memory size. For decoding the (31, 16, 7) QR code, the RLT consists of 72 syndromes and their corresponding coset leaders, and it only requires 288 bytes memory size.

In this paper, the proposed ESWDA has faster decoding speed than the SWDA, and it does not need a memory size to store the lookup table. The key idea of the proposed ESWDA 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. Therefore, the error cases can be swiftly determined. The application of the syndrome weight and the syndrome difference can constitute a high-speed decoding algorithm. Moreover, no complicated computation in the finite field is required in the proposed ESWDA. Two examples demonstrate the decoding procedure of the proposed ESWDA. The proposed ESWDA is applicable to decode other cyclic codes such as the binary (15, 5, 7) BCH code; however, the decoding steps of the proposed ESWDA need to make some slight adjustments. The decoding steps of the binary (15, 5, 7) BCH code are demonstrated in a simple example. Computer simulation result shows that the decoding time of the proposed ESWDA is superior to the SWDA.

The remainder of this paper is organized as follows: The background of the binary QR codes is briefly reviewed in Section 2. The proposed ESWDA is described in Section 3. In Section 4, we use three examples to demonstrate the proposed ESWDA. Computer simulation results of the proposed ESWDA and the SWDA are given in Section 5. Finally, this paper concludes with a brief summary in Section 6.

2. Background of the Binary QR Codes

The binary QR codes are a nice family of linear cyclic codes. Let $(n, (n+1)/2, d)$ denote the binary QR codes with generator polynomial $g(x)$ over $GF(2)$. The length of this code is a prime number of the form $n = 8l \pm 1$, where l is some integer. Also, let $k = (n+1)/2$ denote the message length or information length, and d denote the minimum Hamming distance of the code. The set Q_n of quadratic residues modulo n is the set of nonzero squares modulo n; that is, $Q_n = \{j \mid j \equiv x^2 \bmod n, 1 \le x \le n-1\}$. If $n = 23$, then its quadratic residue set is $Q_{23} = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$. If $n = 31$, then its quadratic residue set is $Q_{31} = \{1, 2, 4, 5, 7, 8, 9, 10, 14, 16, 18, 20, 25, 28\}$.

Let the symbols C_{23} and C_{31} denote the binary Golay code and binary (31, 16, 7) QR code, respectively. Let α be a root of primitive polynomial $p_{r23}(x) = x^{11} + x^2 + 1$. If $\alpha = 2$, then the 11 roots of $p_{r23}(x)$ are $\alpha = 2$, $\alpha^2 = 4$, $\alpha^{2^2} = 16$, $\alpha^{2^3} = 256$, $\alpha^{2^4} = 160$, $\alpha^{2^5} = 1064$, $\alpha^{2^6} = 1605$, $\alpha^{2^7} = 158$, $\alpha^{2^8} = 380$, $\alpha^{2^9} = 1530$, $\alpha^{2^{10}} = 1987$. Let the element $\beta = \alpha^u$ be a primitive 23rd root of unity in $GF(2^{11})$, where $u = (2^{11} - 1) / 23 = 89$. Therefore, $\beta = \alpha^{89} = 322$, where 322 is the decimal representation of β. The total 23 roots are listed in Table 1.

Table 1. The total 23 roots of $x^{23} - 1$ (in decimal number)

$\beta^0 = 1$	$\beta^1 = 322$	$\beta^2 = 174$	$\beta^3 = 1164$	$\beta^4 = 1148$
$\beta^5 = 481$	$\beta^6 = 637$	$\beta^7 = 1942$	$\beta^8 = 1887$	$\beta^9 = 1518$
$\beta^{10} = 1155$	$\beta^{11} = 167$	$\beta^{12} = 2011$	$\beta^{13} = 418$	$\beta^{14} = 281$
$\beta^{15} = 1525$	$\beta^{16} = 378$	$\beta^{17} = 1728$	$\beta^{18} = 1747$	$\beta^{19} = 319$
$\beta^{20} = 552$	$\beta^{21} = 1876$	$\beta^{22} = 1085$		

The generator polynomial of C_{23} is defined by

$$\begin{aligned} g_{23}(x) &= \prod_{i \in Q_{23}} (x - \beta^i) \\ &= x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1. \end{aligned} \tag{1}$$

Now let α be a generator of the multiplicative group of all nonzero elements in $GF(2^5)$. Then, the element $\beta = \alpha^u$, where $u = (2^5 - 1)/31 = 1$. The $x^{31}-1$ can be factored into seven primitive minimal irreducible polynomials; that is, $x^{31}-1 = (x^5 + x^2 + 1)(x^5 + x^4 + x^2 + x + 1)(x^5 + x^3 + x^2 + x + 1)(x^5 + x^4 + x^3 + x^2 + 1)(x^5 + x^3 + 1)(x^5 + x^4 + x^3 + x + 1)(x + 1)$. Then, the generator polynomial of C_{31} is reducible, and is defined by

$$\begin{aligned} g_{31}(x) &= \prod_{i \in Q_{31}} (x - \beta^i) = g_1(x) g_5(x) g_7(x) \\ &= (x^5 + x^2 + 1)(x^5 + x^4 + x^2 + x + 1) \\ &\quad (x^5 + x^3 + x^2 + x + 1) \\ &= x^{15} + x^{14} + x^{13} + x^9 + x^8 + x^3 + 1, \end{aligned} \tag{2}$$

where $g_1(x)$, $g_5(x)$, and $g_7(x)$ are the minimal polynomials of $x^{31}-1$. That is, $g_r(x) = (x - ß^r)(x - ß^{2r}) \cdots (x - ß^{2^4 r})$ and $ß^{2^i r} \in GF(2^5)$ for $0 \le i \le 4$ are the roots of $g_r(x)$, where $r = 1$, 5, and 7. If $\alpha = 2$, then the total 31 roots of $x^{31} - 1$ are listed in Table 2.

Table 2. The total 31 roots of $x^{31} - 1$ (in decimal number)

$\beta^0 = 1$	$\beta^1 = 2$	$\beta^2 = 4$	$\beta^3 = 8$	$\beta^4 = 16$
$\beta^5 = 5$	$\beta^6 = 10$	$\beta^7 = 20$	$\beta^8 = 13$	$\beta^9 = 26$
$\beta^{10} = 17$	$\beta^{11} = 7$	$\beta^{12} = 14$	$\beta^{13} = 28$	$\beta^{14} = 29$
$\beta^{15} = 31$	$\beta^{16} = 27$	$\beta^{17} = 19$	$\beta^{18} = 3$	$\beta^{19} = 6$
$\beta^{20} = 12$	$\beta^{21} = 24$	$\beta^{22} = 21$	$\beta^{23} = 15$	$\beta^{24} = 30$
$\beta^{25} = 5$	$\beta^{26} = 23$	$\beta^{27} = 11$	$\beta^{28} = 22$	$\beta^{29} = 9$
$\beta^{30} = 18$				

Because the minimum Hamming distance of C_{23} and C_{31} is $d = 7$, the inequality $2v + 1 \le 7$ is valid, where v is the actual number of errors to be corrected. Hence, the error-correcting capability is $t = \lfloor (d-1)/2 \rfloor = 3$, where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. The codeword is a multiple of generator polynomial $g(x)$; that is, $C(x) = \sum_{i=0}^{n-1} C_i x^i = m(x)g(x)$, where $C_i \in GF(2)$ for $0 \le i \le n-1$, and $m(x) = \sum_{i=0}^{k-1} m_i x^i$ denotes the message polynomial, where $m_i \in GF(2)$ for $0 \le i \le k-1$. Let $p(x) = \sum_{i=0}^{n-k-1} p_i x^i$ be the parity-check polynomial, where $p_i \in GF(2)$ for $0 \le i \le n-k-1$. Let $p(x) = \sum_{i=0}^{n-k-1} p_i x^i$ be the parity-check polynomial, where $p_i \in GF(2)$ for $0 \le i \le n-k-1$. Let $p(x) \equiv m(x)x^{n-k} \bmod g(x)$, then one obtains $m(x)x^{n-k} = q(x)g(x) + p(x)$. The term $(p(x) + m(x)x^{n-k})$, which is a multiple of $g(x)$, is a codeword polynomial given by

$$c(x) = \sum_{i=0}^{n-1} c_i x^i = p(x) + m(x)x^{n-k}, \tag{3}$$

where $c_i \in GF(2)$ for $0 \le i \le n-1$. In this paper, the systematic encoding method is utilized. Now, let a codeword be transmitted through an additive white Gaussian noise (AWGN) channel to obtain a received word with the form $r(x) = c(x) + e(x)$, where $e(x)$ is the polynomial of the received error pattern expressed as $e(x) = \sum_{i=0}^{n-1} e_i x^i$, where $e_i \in GF(2)$ for $0 \le i \le n-1$. The syndromes polynomial is expressed as $s(x) = \sum_{i=0}^{n-k-1} s_i x^i$.

To simplify the polynomial expressions mentioned above, let the message, codeword, error pattern, received word, and syndrome polynomials be expressed as the binary vector forms $\boldsymbol{m} = (m_0\, m_1 \ldots m_{k-1})$, $\boldsymbol{c} = (c_0\, c_1 \ldots c_{n-1})$, $\boldsymbol{e} = (e_0\, e_1 \ldots e_{n-1})$, $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e} = (r_0\, r_1 \ldots r_{n-1})$, and $\boldsymbol{s} = (s_0\, s_1 \ldots s_{n-k-1})$, respectively. The systematic codeword of the vector form is given by

$$c = mG \tag{4}$$

where $\mathbf{G}$ is called the systematic generator matrix. Let $\mathbf{P}$ be a $k \times (n-k)$ matrix and $\mathbf{I}_k$ be a $k \times k$ identity matrix, and $\mathbf{G}$ can be expressed as

$$\mathbf{G} = \left[\mathbf{P}_{k \times (n-k)} \middle| \mathbf{I}_k \right]_{k \times n} \tag{5}$$

The parity-check matrix $\mathbf{H}$ can be expressed as $\boldsymbol{H} = \left[\boldsymbol{I}_{n-k} \middle| \boldsymbol{P}^T_{(n-k) \times k} \right]_{(n-k) \times n}$, where $\mathbf{P}^T$ denotes the $(n-k) \times k$ transpose matrix of $\mathbf{P}$. The vector form of the syndrome is defined by

$$s = \mathbf{r}\mathbf{H}^T, \tag{6}$$

where $\mathbf{H}^T$ denotes the $n\times(n–k)$ transpose matrix of $\mathbf{H}$; that is, $\mathbf{H}^T$ can be expressed as

$$\mathbf{H}^T = \left[\begin{array}{c} \mathbf{I}_{n-k} \\ \hline \mathbf{P}_{k\times(n-k)} \end{array}\right]_{n\times(n-k)} = \begin{bmatrix} \mathbf{h}_0 \\ \vdots \\ \mathbf{h}_{n-k} \\ \vdots \\ \mathbf{h}_{n-1} \end{bmatrix}. \tag{7}$$

For C_{23}, $\mathbf{H}^T$ has the following form:

$$\mathbf{H}^T = \left[\begin{array}{c} \mathbf{I}_{11} \\ \hline \begin{array}{cccccccccccc} 1&0&1&0&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&1&0&0&1 \\ 1&1&0&1&0&0&1&0&1&0&1 \\ 1&1&0&0&0&1&1&1&0&1&1 \\ 1&1&0&0&1&1&0&1&1&0&0 \\ 0&1&1&0&0&1&1&0&1&1&0 \\ 0&0&1&1&0&0&1&1&0&1&1 \\ 1&0&1&1&0&1&1&1&1&0&0 \\ 0&1&0&1&1&0&1&1&1&1&0 \\ 0&0&1&0&1&1&0&1&1&1&1 \\ 1&0&1&1&1&0&0&0&1&1&0 \\ 0&1&0&1&1&1&0&0&0&1&1 \end{array} \end{array}\right]. \tag{8}$$

3. Decoding Algorithm and Theorems

In this section, the proposed ESWDA is used to decode the C_{23} and C_{31}. For the development of the proposed ESWDA, the following definition, theorem and corollary given in Lin *et al*. [8] are needed.

Definition 1: 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 $d(\mathbf{a}, \mathbf{b}) = w(\mathbf{a} + \mathbf{b})$.

Theorem 1: Let $\mathbf{a} = (a_0 \cdots a_1\ a_{n-1})$ and $\mathbf{b} = (b_0 \cdots b_1\ b_{n-1})$ be two binary vectors, then

$$w(\mathbf{a}+\mathbf{b}) = w(\mathbf{a}) + w(\mathbf{b}) - 2\sum_{i=1}^{n} a_i b_i. \tag{9}$$

Corollary 1: If $a_i b_i = 0$ for $1 \le i \le n$, then

$$w(\mathbf{a}+\mathbf{b}) = w(\mathbf{a}) + w(\mathbf{b}). \tag{10}$$

The following Theorem 2 is useful to compute the syndrome of the received word when the received word shifts one bit to the left. For more detailed proof of this theorem, see [13, p118].

Theorem 2: 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 left. Then the remainder obtained when dividing $xs(x)$ by $g(x)$ is the syndrome $s^{(1)}(x)$ corresponding to $r^{(1)}(x)$.

However, for each cyclical shift of the received word, we have to divide $xs(x)$ by $g(x)$. If the syndrome cyclically shifts many times, then the syndrome computation is rather time-consuming for dividing $xs(x)$ by $g_n(x)$ many times. The following theorem provides an efficient method to compute $\mathbf{s}^{(i)}$ for $0 \le i \le n–1$, and it can save a lot of computational time.

Theorem 3: For the binary QR codes, let r_j be an element of **r** and $\mathbf{h}_j$ be the jth row vector of $\mathbf{H}^T$ for $0 \le j \le n–1$. Then the syndrome $\mathbf{s}^{(i)}$ of $\mathbf{r}^{(i)}$ for $0 \le i \le n–1$ has the form

$$\mathbf{s}^{(i)} = \sum_{j=0}^{n-1} r_j \mathbf{h}_{[i+j]}, \tag{11}$$

where the suffix $[x]$ of $\mathbf{h}$ denotes x mod n.

Proof: Let $\mathbf{r} = (r_0,\ldots,r_{n-1})$ and $\mathbf{r}^{(i)} = (r_i,\ldots, r_{n-1}, r_0,\ldots, r_{i-1})$ for $0 \le i \le n–1$. By (7), we have $\mathbf{s}^{(i)} = \mathbf{r}^{(i)}\mathbf{H}^T = \sum_{j=0}^{n-1} r_{[j-i]}\mathbf{h}_j = \sum_{j=0}^{n-1} r_j \mathbf{h}_{[i+j]}$. The proof is thus completed.

Theorem 3: reveals that the syndrome of $\boldsymbol{r}^{(i)}$ can be fast computed by the vector addition. Theorem 4 also provides an efficient method to simplify the decoding step by using the syndrome weight. For a detailed proof, see [8].

Theorem 3 reveals that the syndrome of $\boldsymbol{r}^{(i)}$ can be fast computed by the vector addition. Theorem 4 also provides an efficient method to simplify the decoding step by using the syndrome weight. For a detailed proof, see [8].

Theorem 4: For the binary QR codes, it is assumed that there are v errors in the received word, where $1 \le v \le t$. All v errors are in the parity-check bits if and only if the weight of syndrome $w(\mathbf{s}) = v$.

By using Theorem 4, we can develop the following useful theorem.

Theorem 5 For the binary QR codes, if v errors are in the information bits of the received word, where $1 \le v \le t$ and $t = \lfloor (d-1)/2 \rfloor$, then the weight of the corresponding syndrome polynomial or syndrome vector satisfies

$$w\big(s(x)\big)^3 \ge (d-v) \text{ or } w(s)^3 \ge (d-v). \tag{12}$$

Proof: Let error polynomial $e(x)$ present the v errors in the information bits; that is, $w(e(x)) = v$. Since $s(x) \equiv r(x) \equiv e(x) \pmod{g(x)}$, we have $e(x) + s(x) \equiv 0 \pmod{g(x)}$. This implies that $e(x) + s(x)$ is a codeword and hence the codeword must satisfy $w(e(x) + s(x)) \ge d$. By Corollary 1, $w(e(x) + s(x)) = w(e(x)) + w(s(x)) \ge d$ and then $w(s(x)) \ge d – w(e(x))$. Thus, the weight of the syndrome polynomial satisfies $w(s(x)) \ge d–v$ or $w(\mathbf{s}) \ge (d–v)$. The proof is thus completed.

Given a received word **r**, the syndrome of $\mathbf{r}^{(i)}$ can be fast computed by Theorem 3. According to Theorem 4, if $1 \le w(\mathbf{s}) \le 3$, then the error positions are in the parity-check bits of **r**. If $1 \le w(\mathbf{s}^{(n-k)}) \le 3$, then the error positions are in the information bits of **r**. Let $\mathbf{h}_i$ denote the ith row vector of $\mathbf{H}^T$, where $0 \le i \le n–1$. Also let $\mathbf{sd}_w$ denote the syndrome difference between the syndromes of **r** and $\mathbf{h}_i$ in each decoding step w. By using the weight of $\mathbf{sd}_w$, the error cases can be quickly determined. Let $\mathbf{u}_0 = (1, 0,\ldots, 0)$ be a k-tuples unit vector and $\mathbf{u}_j$ has only one nonzero component at the jth position, where $0 \le j \le k–1$. By using these properties, the proposed ESWDA can be constructed.

Let upper case P, C, and H denote the error position in the parity-check bits, center bit, and information bits of $\boldsymbol{r}$, respectively. For C_{23} or C_{31}, there are 15 error cases (P, PP, PPP, H, HH, HHH, C, PC, PH, PPC, PPH, PHH, CH, CHH, and PCH), which cover all $\sum_{i=1}^{3}\binom{23}{i} = 2047$ and $\sum_{i=1}^{3}\binom{31}{i} = 4991$ error patterns. If $w(\mathbf{s}) = 0$, then **r** has no error. If $1 \le w(\mathbf{s}) \le 3$ or $1 \le w(\mathbf{s}^{(n-k)}) \le 3$, then there are 6 error cases (P, PP, PPP, H, HH, and HHH). Let the syndrome difference $\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_i)$ for $n–k \le i \le n–1$. If $0 \le w(\mathbf{sd}_3) \le 2$, then there are 5 error cases (C, PC, PH, PPC, and PPH). Let the syndrome difference $\mathbf{sd}_4 = (\mathbf{s}^{(n-k)}–\mathbf{h}_i)$ for $n–k \le i \le n–1$. If $n–k \le i \le n–2$ and $w(\mathbf{sd}_4) = 2$, then there is only 1 error case (PHH). If $i = n–1$ and $1 \le w(\mathbf{sd}_4) \le 2$,

then there are 2 error cases (CH and CHH). Let the syndrome difference $\mathbf{sd}_5 = ((\mathbf{s}–\mathbf{h}_{n-k})–\mathbf{h}_i)$ for $k \le i \le n–1$. If $w(\mathbf{sd}_5) = 1$, then there is only 1 error case (PCH). The decoding steps of the proposed ESWDA work as follows:

1). (No error, P, PP, and PPP cases.) By Theorem 3, compute $\mathbf{s}$ and $w(\mathbf{s})$. If $0 \le w(\mathbf{s}) \le 3$, then the information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1})$. Go to step 6.

2). (H, HH, and HHH cases.) By Theorem 3, compute $\mathbf{s}^{(n-k)}$ and $w(\mathbf{s}^{(n-k)})$. If $1 \le w(\mathbf{s}^{(n-k)}) \le 3$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + (\mathbf{s}^{(n-k)} >>1)$, where ">>" denotes the logical right shift operator in programming or the extension by zeros on the right in mathematics. Go to step 6.

3). (C, PC, PH, PPC, and PPH cases.) Compute the syndrome difference $\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_i)$ for $n–k \le i \le n–1$ and $w(\mathbf{sd}_3)$. If $0 \le w(\mathbf{sd}_3) \le 2$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + \mathbf{u}_{i-(n-k)}$. Go to step 6.

4). (PHH, CH, and CHH cases) Compute the syndrome difference $\mathbf{sd}_4 = (\mathbf{s}^{(n-k)}–\mathbf{h}_i)$ for $n–k \le i \le n–1$ and $w(\mathbf{sd}_4)$. If $n–k \le i \le n$-2 and $w(\mathbf{sd}_4) = 2$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + (\mathbf{sd}_4 >>1)$. If $i = n–1$ and $1 \le w(\mathbf{sd}_4) \le 2$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + \mathbf{u}_0 + (\mathbf{sd}_4 >> 1)$. Go to step 6.

5). (PCH case.) Compute the syndrome difference $\mathbf{sd}_5 = ((\mathbf{s}–\mathbf{h}_{n-k})–\mathbf{h}_i)$ for $k \le i \le n–1$ and $w(\mathbf{sd}_5)$. If $w(\mathbf{sd}_5) = 1$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + \mathbf{u}_0 + \mathbf{u}_{i-(n-k)}$. Go to step 6.

6). Stop.

Table 3 lists all the 15 error cases and the number of error patterns in each decoding steps of the proposed ESWDA. The flowchart of the proposed ESWDA is shown in Figure 1.

Table 3. The number of error patterns in each decoding step of C_{23} and C_{31}

Steps	Cases	Number of error patterns	
		C_{23}	C_{31}
1	P	$\binom{11}{1} = 11$	$\binom{15}{1} = 15$
	PP	$\binom{11}{2} = 55$	$\binom{15}{2} = 105$
	PPP	$\binom{11}{3} = 165$	$\binom{15}{3} = 455$
2	H	$\binom{11}{1} = 11$	$\binom{15}{1} = 15$
	HH	$\binom{11}{2} = 55$	$\binom{15}{2} = 105$
	HHH	$\binom{11}{3} = 165$	$\binom{15}{3} = 455$
3	C	1	1
	PC	$\binom{11}{1} = 11$	$\binom{15}{1} = 15$
	PPC	$\binom{11}{2} = 55$	$\binom{15}{2} = 105$
	PH	$\binom{11}{1}\binom{11}{1} = 121$	$\binom{15}{1}\binom{15}{1} = 225$
	PPH	$\binom{11}{2}\binom{11}{1} = 605$	$\binom{15}{2}\binom{15}{1} = 1575$
4	PHH	$\binom{11}{1}\binom{11}{2} = 605$	$\binom{15}{2}\binom{15}{1} = 1575$
	CH	$\binom{11}{1} = 11$	$\binom{15}{1} = 15$
	CHH	$\binom{11}{2} = 55$	$\binom{15}{2} = 105$
5	PCH	$\binom{11}{1}\binom{11}{1} = 121$	$\binom{15}{1}\binom{15}{1} = 225$

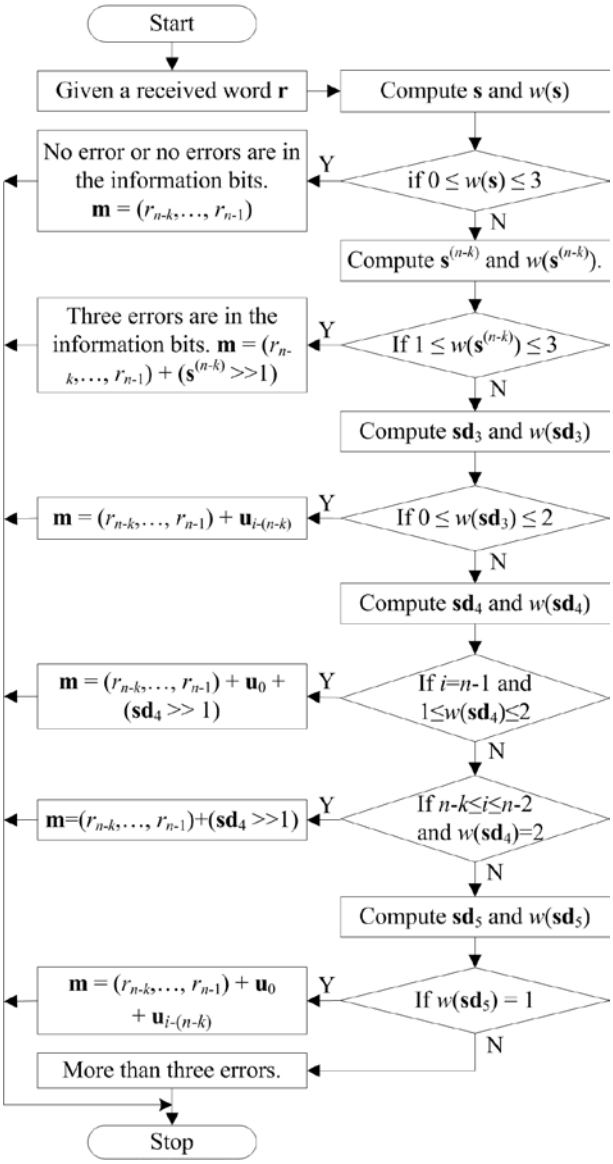

Figure 1. Flowchart of the proposed ESWDA

4. Examples

In this section, three examples are presented to illustrate the proposed ESWDA. Example 1 and 2 show the decoding steps for the binary systematic Golay code. Example 3 shows the decoding steps for the binary systematic (15, 5, 7) BCH code, denoted by C_{15}; however, the decoding steps of the proposed ESWDA have to make some slight adjustments.

Example 1: Let a message $\mathbf{m}$ = (000110101010) be encoded into a C_{23} codeword $\mathbf{c}$ = (11011010100000110101010). If the received word $\mathbf{r}$ = (11011010100010111001010), then the error pattern $\mathbf{e}$ = (00000000000010001100000), which means a HHH error case. The decoding steps are shown below.

1). Compute $\mathbf{s}$ = (10101100100) and $w(\mathbf{s})$ = 5. Since $w(\mathbf{s}) > 3$, go to step 2.

2). Compute $\mathbf{s}^{(11)}$ = (10001100000) and $w(\mathbf{s}^{(11)}) = 3 \le 3$. The corrected information word $\mathbf{m}$ = (010111001010) + (010001100000) = (000110101010). Go to stop.

Example 2: This example demonstrates the worst decoding case. Let a message $\mathbf{m}$ = (000110101010) be encoded into a C_{23} codeword $\mathbf{c}$ =

(11011010100000110101010). If the received word **r** = (01011010100100110101011), then the error pattern **e** = (10000000000100000000001), which means a PCH error case. The decoding steps are shown below.

1). Compute $\boldsymbol{s}$ = (11001001111) and $w(\boldsymbol{s})$ = 7. Since $w(\mathbf{s}) > 3$, go to step 2.

2). Compute $\boldsymbol{s}^{(11)}$ = (01001001110) and $w(\boldsymbol{s}^{(11)})$ = 5. Since $w(\mathbf{s}^{(11)}) > 3$, go to step 3.

3). Compute $\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_i)$ for $11 \le i \le 22$ and $w(\mathbf{sd}_3)$.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{11})$ = (11001001111) – (11000111010) = (00001110101). $w(\mathbf{sd}_3)$ = 5.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{12})$ = (11001001111) – (01100011101) = (10101010010). $w(\mathbf{sd}_3)$ = 5.

…

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{21})$ = (11001001111) – (10010011111) = (01011010000). $w(\mathbf{sd}_3)$ = 4.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{22})$ = (11001001111) – (10001110101) = (01000111010). $w(\mathbf{sd}_3)$ = 5.

Since every $w(\mathbf{sd}_3) > 2$, go to step 4.

4). Compute $\mathbf{sd}_4 = (\mathbf{s}^{(11)}–\mathbf{h}_i)$ for $11 \le i \le 22$ and $w(\mathbf{sd}_4)$.

$\mathbf{sd}_4 = (\mathbf{s}^{(11)}–\mathbf{h}_{11})$ = (01001001110) – (11000111010) = (10001110100). $w(\mathbf{sd}_4)$ = 5.

$\mathbf{sd}_4 = (\mathbf{s}^{(11)}–\mathbf{h}_{12})$ = (01001001110) – (01100011101) = (00101010011). $w(\mathbf{sd}_4)$ = 5.

…

$\mathbf{sd}_4 = (\mathbf{s}^{(11)}–\mathbf{h}_{21})$ = (01001001110) – (10010011111) = (11011010001). $w(\mathbf{sd}_4)$ = 6.

$\mathbf{sd}_4 = (\mathbf{s}^{(11)}–\mathbf{h}_{22})$ = (01001001110) – (10001110101) = (11000111011). $w(\mathbf{sd}_4)$ = 7.

Since every $w(\mathbf{sd}_4) > 2$, go to step 5.

5). Compute $\mathbf{sd}_5 = ((\mathbf{s}–\mathbf{h}_{11})–\mathbf{h}_i) = (00001110101) – \mathbf{h}_i$ for $12 \le i \le 22$ and $w(\mathbf{sd}_5)$.

$\mathbf{sd}_5 = (\mathbf{s}–\mathbf{h}_{11})–\mathbf{h}_{12}$ = (00001110101) – (11000111010) = (11001001111). $w(\mathbf{sd}_5)$ = 7.

$\mathbf{sd}_5 = (\mathbf{s}–\mathbf{h}_{11})–\mathbf{h}_{13}$ = (00001110101) – (11110110100) = (11111000001). $w(\mathbf{sd}_5)$ = 6.

…

$\mathbf{sd}_5 = (\mathbf{s}–\mathbf{h}_{11})–\mathbf{h}_{21}$ = (00001110101) – (10010011111) = (10011101010). $w(\mathbf{sd}_5)$ = 6.

$\mathbf{sd}_5 = (\mathbf{s}–\mathbf{h}_{11})–\mathbf{h}_{22}$ = (00001110101) – (10001110101) = (10000000000). $w(\mathbf{sd}_5)$ = 1.

Since $w(\mathbf{sd}_5)$ = 1, the corrected information word $\mathbf{m} = (r_{11},\ldots, r_{22}) + \mathbf{u}_0 + \mathbf{u}_{22\text{-}11}$ = (100110101011) + (100000000000) + (000000000001) = (000110101010). Go to stop.

PCH case is the worst case; however, only 121 error patterns, namely 5.91%, will enter step 5.

Example 3: Let a message **m** = (00011) be encoded into a codeword of C_{15} with $g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1$, and obtain the codeword **c** = (101100101000011). If the received word **r** = (101110101011011), then the error pattern **e** = (000010000011000), which means a PHH error case. First, the decoding steps of the proposed ESWDA mentioned in Section 3 can be slightly adjusted as follows:

1). (No error, P, PP, and PPP cases.) By Theorem 3, compute **s** and $w(\mathbf{s})$. If $0 \le w(\mathbf{s}) \le 3$, then the information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1})$, and go to step 5.

2). (H, HH, HHH, PH, PPH, and PHH cases.) By Theorem 3, compute $\mathbf{s}^{(n-k)}$ and $w(\mathbf{s}^{(n-k)})$. If $1 \le w(\mathbf{s}^{(n-k)}) \le 3$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + (\mathbf{s}^{(n-k)}$ & (11111)), where the notation "&" denotes the bitwise AND operator, and go to step 5.

3). (PH and PPH cases.) Compute the syndrome difference $\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_i)$ for $n–k \le i \le n–1$ and $w(\mathbf{sd}_3)$. If $0 \le w(\mathbf{sd}_3) \le 2$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + \mathbf{u}_{i-(n-k)}$, and go to step 5.

4). (PHH case) Compute the syndrome difference $\mathbf{sd}_4 = (\mathbf{s}^{(n-k)}–\mathbf{h}_i)$ for $n–k \le i \le n–1$ and $w(\mathbf{sd}_4)$. If $w(\mathbf{sd}_4) = 2$, then the corrected information vector is $\mathbf{m} = (r_{n-k},\ldots, r_{n-1}) + (\mathbf{sd}_4$ & (11111)), and go to step 5.

5). Stop.

For this code, there are 9 error cases. Table 4 lists all the 9 error cases and the number of error patterns in each decoding steps.

Table 4. The number of error patterns in each decoding step of C_{15}

Steps	Cases	Number of error patterns
		C_{23}
1	P	$\binom{10}{1} = 10$
	PP	$\binom{10}{2} = 45$
	PPP	$\binom{10}{3} = 120$
2	H	$\binom{5}{1} = 5$
	HH	$\binom{5}{2} = 10$
	HHH	$\binom{5}{3} = 10$
	PH	$\binom{5}{1}\binom{5}{1} = 25$
	PPH	$\binom{5}{2}\binom{5}{1} = 50$
	PHH	$\binom{5}{1}\binom{5}{2} = 50$
3	PH	$\binom{10}{1}\binom{5}{1} - \binom{5}{1}\binom{5}{1} = 25$
	PPH	$\binom{10}{2}\binom{5}{1} - \binom{5}{2}\binom{5}{1} = 175$
4	PHH	$\binom{10}{1}\binom{5}{2} - \binom{5}{1}\binom{5}{2} = 50$

The decoding steps are shown below.

1). Compute **s** = (1001001011) and $w(\mathbf{s})$ = 5. Since $w(\mathbf{s}) > 3$, go to step 2.

2). Compute $\mathbf{s}^{(5)}$ = (1101111101) and $w(\mathbf{s}^{(5)})$ = 8. Since $w(\mathbf{s}^{(5)}) > 3$, go to step 3.

3). Compute $\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_i)$ for $10 \le i \le 14$ and $w(\mathbf{sd}_3)$.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{11})$ = (1001001011) – (1101100101) = (0100101110). $w(\mathbf{sd}_3)$ = 5.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{12})$ = (1001001011) – (0110101111) = (1111100100). $w(\mathbf{sd}_3)$ = 6.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{13})$ = (1001001011) – (1101011110) = (0100010101). $w(\mathbf{sd}_3)$ = 4.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{14})$ = (1001001011) – (0111011001) = (1110010010). $w(\mathbf{sd}_3)$ = 5.

$\mathbf{sd}_3 = (\mathbf{s}–\mathbf{h}_{15})$ = (1001001011) – (1110110010) = (0111111001). $w(\mathbf{sd}_3)$ = 7.

Since every $w(\mathbf{sd}_3) > 2$, go to step 4.

4). Compute $\mathbf{sd}_4 = (\mathbf{s}^{(5)}–\mathbf{h}_i)$ for $10 \le i \le 14$ and $w(\mathbf{sd}_4)$.

$\mathbf{sd}_4 = (\mathbf{s}^{(5)}–\mathbf{h}_{11})$ = (1101111101) – (1101100101) = (0000011000). $w(\mathbf{sd}_4)$ = 2.

Since $w(\mathbf{sd}_4)$ = 2, the corrected information word $\mathbf{m} = (r_{10},\ldots, r_{14}) + (\mathbf{sd}_4$ & (11111)) = (11011) + ((0000011000) & (11111)) = (00011). Go to stop.

5. Simulation Results

The proposed ESWDA has been programmed in C++ language. On an Intel Q6600 PC with XP operating system, all 2^k codewords with all $\sum_{i=1}^{3}\binom{n}{i}$ error patterns were created to check every possible error pattern of C_{15}, C_{23}, and C_{31}, respectively. In other words, the error patterns of C_{15}, C_{23}, and C_{31} are $\sum_{i=1}^{3}\binom{15}{i}=575$, $\sum_{i=1}^{3}\binom{23}{i}=2047$, and $\sum_{i=1}^{3}\binom{31}{i}=4991$, respectively. The decoding times of the proposed ESWDA and the SWDA are shown in the Table 5, Table 6, and Table 7, respectively. For $v = 1$, it means that one error of that code input to the decoder, and for the average decoding time, it means that all the error patterns of that code input to the decoder. In these three tables, the average decoding time of the proposed ESWDA is about 10.6 times, 19.6 times, and 4 times faster than the SWDA, respectively. The memory requirements of the two algorithms are also shown in Table 5, Table 6, and Table 7, respectively. It is obvious that the proposed ESWDA significantly reduces decoding time with the increase of the code length.

Table 5. Comparison of the decoding time (in μs) and memory requirement (in bytes) for the Golay code between two algorithms

Algorithms	Number of errors				Memory size
	$v = 1$	$v = 2$	$v = 3$	Average	
ESWDA	0.232	0.372	0.661	0.612	0
SWDA	4.026	5.311	6.728	6.514	168

Table 6. Comparison of the decoding time (in μs) and memory requirement (in bytes) for the (31, 16, 7) QR code between two algorithms

Algorithms	Number of errors				Memory size
	$v = 1$	$v = 2$	$v = 3$	Average	
ESWDA	0.316	0.548	0.997	0.956	0
SWDA	11.23	14.89	19.07	18.73	288

Table 7. Comparison of the decoding time (in μs) and memory requirement (in bytes) for the (15, 5, 7) BCH code between two algorithms

Algorithms	Number of errors				Memory size
	$v = 1$	$v = 2$	$v = 3$	Average	
ESWDA	0.196	0.236	0.321	0.289	0
SWDA	0.097	0.933	1.151	1.115	36

6. Conclusions

Binary QR codes are well known for their good features. A high-speed and efficient ESWDA is developed to decode C_{15}, C_{23}, and C_{31}. The proposed ESWDA neither stores large lookup table in the memory nor computes complicated algebraic computations. By using Theorem 3, Theorem 4, Theorem 5, and the weight of $\mathbf{sd}_w$, the error cases can be quickly identified and corrected. Therefore, the proposed ESWDA is a very efficient and low-cost decoder for decoding the triple-error-correcting QR codes. The proposed ESWDA can be extended to decode other QR codes or BCH codes; however, the decoding steps of the proposed ESWDA need to make some slight adjustments.

References

[1] Chien, R.T., "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inform. Theory*, 10(4). 357-363. Oct. 1964.

[2] Chen, Y.H., Chien, C.H., Huang, C.H., Truong, T.K., and Jing, M.H., "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.

[3] Elia, M., "Algebraic decoding of the (23, 12, 7) Golay codes," *IEEE Trans. Inform. Theory*, 33(1). 150-151. Jan. 1987.

[4] Golay, M.J.E., "Notes on digital coding," *Proc. IRE*, 37, 657. 1949.

[5] Lee, C.D., "Weak general error locator polynomials for triple-error-correcting binary Golay code," *IEEE Comm. Letters*, 15(8). 857-859. Aug. 2011.

[6] Lin, T.C., Chang, H.C., Lee, H.P., Chu, S.I, and Truong, T.K., "Decoding of the (31, 16, 7) quadratic residue code," *J. Chinese Institute of Engineers*, 33(4). 573-580. June 2010.

[7] Lin, T.C., Chang, H.C., Lee, H.P., and Truong, T.K., "On the decoding of the (24, 12, 8) Golay code," *Inform. Sci.*, 180(23). 4729-4736. Dec. 2010.

[8] Lin, T.C., Lee, H.P., Chang, H.C., Chu, S.I, and Truong, T.K., "High speed decoding of the binary (47, 24, 11) quadratic residue code," *Inform. Sci.*, 180(20). 4060-4068. Oct. 2010.

[9] Lin, T.C., Lee, H.P., Chang, H.C., and Truong, T.K., "A cyclic weight algorithm of decoding the (47, 24, 11) quadratic residue code," *Inform. Sci.*, 197. 215-222. Aug. 2012.

[10] Reed, I.S., Shih, M.T., and Truong, T.K., "VLSI design of inverse-free Berlekamp-Massey algorithm," *IEE Proc. Comput. Digit. Tech.*, 138(5). 295-298. Sept. 1991.

[11] Reed, I.S., Yin, X., and Truong, T.K., "Algebraic decoding of the (32, 16, 8) quadratic residue code," *IEEE Trans. Inform. Theory*, 36 (4). 876-880. July 1990.

[12] Reed, I.S., Yin, X., Truong, T.K., and Holmes, J.K., "Decoding the (24, 12, 8) Golay code," *IEE Proc. Comput. Digit. Tech.*, 137(3). 202-206. May 1990.

[13] Wicker, S.B. *Error control systems for digital communication and storage*, Prentice Hall, New Jersey, 1995.