A Memory Improvement on Decoding of the (41, 21, 9) Quadratic Residue Code

Hung-Peng Lee and Hsin-Chiu Chang

***Abstract*—In this paper, an effective table lookup decoding algorithm (TLDA), called the syndrome and syndrome difference decoding algorithm (SSDDA), is developed to decode the binary systematic (41, 21, 9) quadratic residue (QR) code. It is based on the property of the weight of syndrome and the weight of syndrome difference to reduce the memory size of the lookup table. The proposed algorithm requires a lookup table, called the compact lookup table (CLT), which only consists of 231 syndromes and their corresponding error patterns. The advantage of the SSDDA over the previous TLDAs is that the memory size of the proposed CLT is only about 82.2% and 2% of the lookup table needed in the decoding algorithms of Lin *et al*. (2010) and Chen *et al*. (2010), respectively.**

***Index Terms*—Quadratic residue code, lookup table, error pattern, syndrome.**

I. Introduction

The well-known QR codes, introduced by Prange [1] in 1957, are cyclic BCH codes with code rates greater than or equal to one-half. In addition, the codes generally have large minimum distances, so that most of the known QR codes are the best-known codes. The code augmented by a parity bit, for example, the (24, 12, 8) QR code was utilized to provide error control on the Voyager deep-space mission [2].

In the past decades, several decoding techniques have been developed to decode the QR codes. The algebraic decoding algorithms (ADAs) given in [3]–[15] used the error-locator polynomial to find the different error positions. Recently, some table lookup decoding algorithms (TLDAs) given in [16]–[19] play an important role in error-correcting decoding, and they have faster decoding speed than the ADAs. However, the TLDAs require a memory space in the decoder chip and increase the decoding cost rapidly when the code length is large. Although, the ADAs do not need any lookup table; however, they require a large number of multiplication and division operations in finite field. These complicated computations will lead to a decoding delay when the code length is large. Besides, according to the appendixes of [7], the correct codewords cannot be obtained from the unknown syndrome polynomials $f(S_5)$ and $g(S_5)$ of degree 11 in Case 5, where S_5 is the unknown syndrome of the (47, 24, 11) QR code. Lin *et al.* [18] corrects all coefficients of $f(S_5)$ and $g(S_5)$ and can obtain the correct codewords from the improved error-locator polynomial $L_5(z)$. Therefore, the case of five errors given in [7], [10], [11], and [13] can be modified by the new case of five errors given in [18].

The error-correcting capability of the (41, 21, 9) QR code is $t = \lfloor (d-1)/2 \rfloor = \lfloor (9-1)/2 \rfloor = 4$, where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x, and $d = 9$ is the minimum Hamming distance of the code. The decoding of the binary (41, 21, 9) QR code was first proposed by Reed *et al.* [6]. In this paper, we can further reduce the size of the condensed lookup table given in [18] and lookup table given in [17] by 11.8% and 98%, respectively; besides, simulation results show that the average decoding time of the proposed SSDDA is faster than that of those two decoding algorithms.

The structure of the remainder of the paper is organized as follows: The background of binary systematic (41, 21, 9) QR codes is simply introduced in Section II. The generation of the CLT is shown in Section III. The proposed SSDDA is described in Section IV. Simulation results are shown in Section V. Finally, this paper concludes with a brief summary in Section VI.

II. Background of the Binary Systematic (41, 21, 9) QR Code

A binary QR code (n, k, d) or (n, (n+1)/2, d) is defined algebraically as a multiple of its generator polynomial $g(x)$ over $GF(2)$, where n is the code length, k is the message length, and d is the minimum Hamming distance of the code. Let n be a prime number of the form $n = 8l \pm 1$, where l is a positive integer and m be the smallest positive integer such that n divides $2^m - 1$. The set Q of quadratic residues modulo n is the set of nonzero squares modulo n; that is, $Q_n = \{j \mid j \equiv x^2 \bmod n, 1 \le x \le (n-1)/2\}$. For the binary (41, 21, 9) QR code over $GF(2^{20})$, the quadratic residue set is

$$Q_{41} = \{1, 2, 4, 5, 8, 9, 10, 16, 18, 20, 21, 23, 25, 31, 32, 33, 36, 37, 39, 40\} \tag{1}$$

Let α be a root of primitive polynomial $p(x) = x^{20} + x^3 + 1$; that is, $p(\alpha) = 0$. Thus, the element $\beta = \alpha^u$, where $u = (2^m - 1)/n = (2^{20} - 1)/41 = 25575$, is a primitive 41-th root of unity in $GF(2^{20})$. The generator polynomial $g(x)$ is defined by

$$\begin{aligned} g(x) &= \textstyle\prod_{i \in Q_{41}} (x - \beta^i) \\ &= x^{20} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{11} + \\ &\quad x^{10} + x^9 + x^6 + x^5 + x^4 + x^3 + x^2 + 1 \end{aligned} \tag{2}$$

where the degree of $g(x)$ is 20, which is the multiplicative order of the integer 2 modulo the code length 41; that is, $2^{20} \equiv 1 \bmod 41$.

If $\alpha = 2$, then $\beta = \alpha^{25575} = 396789$ and $g(\beta) = 0$. By

Manuscript received June 5, 2012; revised July 5, 2012.

The authors are with the Department of Computer Science and Information Engineering, Fortune Institute of Technology, Kaohsiung City 83160, Taiwan, ROC (e-mail: hpl@center.fotech.edu.tw, newballch@gmail.com).

cyclically shifting β once to the left and mod $g(x)$, then we obtain the 41 roots of $x^{41}-1$. The total 41 roots are listed in Table I, and the value of the root is expressed in hexadecimal digits.

TABLE I: THE 41 ROOTS OF $x^{41}-1$ (IN HEXADECIMAL).

β^0 = 1	β^1 = 60DF5	β^2 = 6216F	β^3 = 4000	β^4 = 20
β^5 = F6846	β^6 = AFB06	β^7 = 80000	β^8 = 100	β^9 = 2
β^{10} = C1BEA	β^{11} = C42DE	β^{12} = 8000	β^{13} = 40	β^{14} = 91EF1
β^{15} = 23871	β^{16} = 7CE7D	β^{17} = 800	β^{18} = 4	β^{19} = FF9A9
β^{20} = F4BC1	β^{21} = 10000	β^{22} = 80	β^{23} = 57E6F	β^{24} = 470E2
β^{25} = F9CFA	β^{26} = 1000	β^{27} = 8	β^{28} = 83D2F	β^{29} = 959FF
β^{30} = 20000	β^{31} = 100	β^{32} = BE73E	β^{33} = 8E1C4	β^{34} = 8F789
β^{35} = 1FEC	β^{36} = 10	β^{37} = 7B420	β^{38} = 57D83	β^{39} = 40000
β^{40} = 200				

A codeword of the binary systematic (41, 21, 9) QR code is a polynomial $c(x) = c_{40}x^{40} + \cdots + c_1x + c_0$ such that it is a multiple of $g(x)$. If the codeword $c(x)$ is transmitted through a noisy channel, then the received polynomial $r(x) = r_{40}x^{40} + \cdots + r_1x + r_0$ can be expressed as the sum of the codeword polynomial $c(x)$ and the error polynomial $e(x) = e_{40}x^{40} + \cdots + e_1x + e_0$. For simplicity, let the message or information, codeword, error pattern, received word, and syndrome be expressed as the binary vector forms $m = (m_{k-1},\ldots, m_1, m_0)$, $c = (c_{n-1},\ldots, c_1, c_0)$, $e = (e_{n-1},\ldots, e_1, e_0)$, $r = (r_{n-1},\ldots, r_1, r_0)$, and $s = (s_{n-k-1},\ldots, s_1, s_0)$, respectively. For the binary systematic (41, 21, 9) QR code, it follows from [11, p85] that the systematic $k \times n = 21 \times 41$ generator matrix $\mathbf{G}$ can be expressed as follows:

$$\mathbf{G} = \left[\mathbf{P}_{21\times 20} \mid \mathbf{I}_{21}\right]_{21\times 41} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,19} & 1 & 0 & \ldots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,19} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{20,0} & p_{20,1} & \cdots & p_{20,19} & 0 & 0 & \ldots & 1 \end{bmatrix}_{21\times 41} \tag{3}$$

where I_{21} is an 21×21 identity matrix and $P_{21\times 20}$ is an 21×20 matrix given by

$$\mathbf{P} = \begin{bmatrix} 10111111001110111110 \\ 01011111001110011111 \\ 10010001111011110001 \\ 11110110100001000110 \\ 01111011010000100011 \\ 10000011110100101111 \\ 11111111100110101001 \\ 11000001101111101010 \\ 01100000110111110101 \\ 10001110000111000100 \\ 01000110000011100010 \\ 00100011000001110001 \\ 10101111101100000110 \\ 01010111110110000011 \\ 10010101100111111111 \\ 11110100101111000001 \\ 11000100001011011110 \\ 01100010000101101111 \\ 10001111011110001001 \\ 11111001110011111010 \\ 01111100111001111101 \end{bmatrix}_{21\times 20} \tag{4}$$

The codeword of systematic form can be obtained in matrix form by

$$\begin{aligned} \mathbf{c} = \mathbf{mG} &= (m_{20} \ \ldots m_1 \ m_0)\left[\mathbf{P}_{21\times 20} \middle| \mathbf{I}_{21}\right] \\ &= (p_{19} \ \ldots p_1 \ p_0 \mid m_{20} \ \ldots m_1 \ m_0) \end{aligned} \tag{5}$$

The $(p_{19},\ldots, p_1, p_0)$ is the parity-check part of codeword $\mathbf{c}$ and the $(m_{20},\ldots, m_1, m_0)$ is the message part of codeword $\mathbf{c}$. The systematic parity check matrix H of size $(n-k) \times n = 20 \times 41$ can be expressed as follows:

$$\mathbf{H} = \left[\mathbf{I}_{20} \mid \mathbf{P}^T\right]_{20\times 41} = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,0} & p_{1,0} & \cdots & p_{20,0} \\ 0 & 1 & \cdots & 0 & p_{0,1} & p_{1,1} & \cdots & p_{20,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{0,19} & p_{1,19} & \cdots & p_{20,19} \end{bmatrix}_{20\times 41} \tag{6}$$

where P^T is the transpose matrix of P and I_{20} is an 20×20 identity matrix. The vector form of the syndrome can be defined by

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = \mathbf{r}\begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,0} & p_{1,0} & \cdots & p_{20,0} \\ 0 & 1 & \cdots & 0 & p_{0,1} & p_{1,1} & \cdots & p_{20,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{0,19} & p_{1,19} & \cdots & p_{20,19} \end{bmatrix}^T \tag{7}$$

where H^T denotes an 41×20 transpose matrix of H; that is,

$$\mathbf{H}^T = \left[\frac{\mathbf{I}_{20}}{\mathbf{P}_{21\times 20}}\right]_{41\times 20}.$$

The proposed algorithm only needs to compute the syndrome of r for every received word and the complicated computation of the error-locator polynomial in $GF(2^{20})$ can be completely avoided. This is the reason why the proposed algorithm significantly reduces the computational complexity. If r has no error occurred, namely $\mathbf{e} = \mathbf{0}$, then the syndrome $\boldsymbol{s} = rH^T = (\boldsymbol{c} + \mathbf{0})H^T = cH^T = \mathbf{0}$, where $\mathbf{0}$ denotes a zero vector. Otherwise, $\boldsymbol{s} = rH^T = (\boldsymbol{c} + \boldsymbol{e})H^T = \mathbf{0} + eH^T = eH^T$.

III. GENERATION OF THE CLT

The full size of the set of syndromes corresponding to error patterns in a lookup table is $\sum_{i=1}^{t}\binom{n}{i}$. Therefore, for the (41, 21, 9) QR code, the full size of the syndrome lookup table is $\sum_{i=1}^{4}\binom{41}{i} = \binom{41}{1} + \binom{41}{2} + \binom{41}{3} + \binom{41}{4} = 112{,}791$, and thus this table is what is called the full lookup table (FLT). A syndrome needs 3 bytes to store in memory and an error pattern needs 6 bytes to store in memory. In other words, the total memory size of the FLT needs 112,791 × (3 bytes + 6 bytes) = 1,015,119 bytes ≈ 991.33 Kbytes. However, searching a syndrome in this large table is very time-consuming and impractical.

The fast lookup table decoding algorithm (FLTDA) developed in [17], a type of shift-search method [4], needs $\sum_{i=1}^{3}\binom{41}{i} = \binom{41}{1} + \binom{41}{2} + \binom{41}{3} = 11{,}521$ syndromes and their corresponding error patterns; that is, this lookup table needs

11,521 × (3 bytes + 6 bytes) = 103,689 bytes ≈ 101.26 Kbytes memory size to store the lookup table. The shift-search method deletes all $v = t$, where $v \le t$ is the actual number of errors occurred, patterns and keeps $1 \le v \le t - 1$ patterns in lookup table. However, the size of this lookup table is still very large.

The syndrome decoding algorithm (SDA) with a reduce-size lookup table (RSLT) [2, p119] is a very efficient method of decoding linear cyclic codes over a noisy channel for moderate code lengths. In essence, the SDA is a minimum distance decoding using the syndromes corresponding to their error patterns in the lookup table. Due to the property of cyclic code, the size of the RSLT can be greatly reduced up to 1/41 of the size of the FLT. In this case, it uses a coset leader to represent a coset with the same syndrome. So the size of the RSLT only needs 112,791/41 = 2,751 syndromes and their corresponding error patterns; that is, the RSLT needs 2,751 × (3 bytes + 6 bytes) = 24,759 bytes ≈ 24.18 Kbytes memory size. However, this memory size of the RSLT is still large that we need to further reduce the memory size of the RSLT.

Recently, Lin *et al*. [18] proposed a TLDA combined with the condensed lookup table (COLT), a type of shift-search method [4], to further reduce the memory size of the RSLT. The COLT requires $(\sum_{i=1}^{3}\binom{41}{i})/41$ = 11,521/41 = 281 syndromes and their corresponding error patterns, and it only needs 281 × (3 bytes + 6 bytes) = 2,529 bytes ≈ 2.47 Kbytes memory size. The memory size of the COLT is only 2.44% of the lookup table given in [17].

In fact, the size of the COLT is very small and can fit in any DSP or embedded system software implementation. However, we can again further reduce the size of the COLT and obtain a faster average decoding time compared with other exist TLDAs. Now, we develop a SSDDA with CLT to further reduce the size of the COLT. First, we give the following definition for the proposed CLT.

Definition 1: The compact lookup table (CLT) is a table of all error patterns of weight v occurred only in the message part together with their corresponding syndromes, where $1 \le v \le 2$.

For decoding the (41, 21, 9) QR code, from Definition 1 we know that the numbers of the syndromes and their corresponding error patterns in the CLT are the sum of $C_1^{21} = 21$ and $C_2^{21} = 210$. So the total memory size of the CLT is only 231 × (3 bytes + 6 bytes) / 1024 = 2,079 bytes ≈ 2.03 Kbytes, which is only 82.2% of the memory size of the COLT. In such a very small table, the proposed SSDDA can decode the total 112,791 error patterns of the binary (41, 21, 9) QR code. The relationship of the syndromes and their corresponding error patterns in the CLT is shown in Table II. Therefore, the proposed CLT is the least lookup table over all previous proposed lookup tables for decoding the binary (41, 21, 9) QR code.

TABLE II: THE SYNDROMES AND THEIR CORRESPONDING ERROR PATTERNS IN THE CLT (IN HEXDECIMAL).

Syndromes	Error patterns
$v = 1$	
$\mathbf{s}_1$ = (7CE7D)	$\mathbf{e}_1$ = (00000000001)
$\mathbf{s}_2$ = (F9CFA)	$\mathbf{e}_1$ = (00000000001)
…	…
$\mathbf{s}_{20}$ = (5F39F)	$\mathbf{e}_{20}$ = (00000080000)
$\mathbf{s}_{21}$ = (BE73E)	$\mathbf{e}_{21}$ = (00000100000)
$v = 2$	
$\mathbf{s}_{22}$ = (85287)	$\mathbf{e}_{22}$ = (00000000003)
$\mathbf{s}_{23}$ = (F39F4)	$\mathbf{e}_{23}$ = (00000000005)
…	…
$\mathbf{s}_{230}$ = (2F9CF)	$\mathbf{e}_{230}$ = (00000140000)
$\mathbf{s}_{231}$ = (E14A1)	$\mathbf{e}_{231}$ = (00000180000)

In order to reduce the searching time in the CLT, it is well-known that the famous binary search algorithm is utilized instead of performing an exhaustive search. However, the syndromes need to be arranged in ascending order. The relationship of the syndromes in ascending order and their corresponding error patterns in the CLT is shown in Table III. Let N = 231 be the number of all syndromes in the CLT. Then, the time complexity of finding a syndrome in the CLT is at most $O(\log_2 N) = O(\log_2(231)) = 7.85$ times. If a linear search algorithm is used to find the syndrome in the CLT, then the time complexity is at most O(231) = 231 times. By using the binary search algorithm compared to the linear search algorithm, the average searching time can be saved up to 29.42 times. Therefore, the CLT is naturally suitable for both software and hardware implementation.

TABLE III: SYNDROMES IN ASCENDING ORDER AND THEIR CORRESPONDING ERROR PATTERNS IN THE CLT (IN HEXDECIMAL).

Syndromes	Error patterns
$\mathbf{s}_1$ = (0164D)	$\mathbf{e}_1$ = (00000000804)
$\mathbf{s}_2$ = (02387)	$\mathbf{e}_2$ = (00000020020)
$\mathbf{s}_3$ = (02C9A)	$\mathbf{e}_3$ = (00000001008)
$\mathbf{s}_4$ = (0470E)	$\mathbf{e}_4$ = (00000040040)
$\mathbf{s}_5$ = (05934)	$\mathbf{e}_5$ = (00000002010)
…	…
$\mathbf{s}_{227}$ = (F890C)	$\mathbf{e}_{227}$ = (00000018000)
$\mathbf{s}_{228}$ = (F97DC)	$\mathbf{e}_{228}$ = (00000100400)
$\mathbf{s}_{229}$ = (F9CFA)	$\mathbf{e}_{229}$ = (00000000002)
$\mathbf{s}_{230}$ = (FF352)	$\mathbf{e}_{230}$ = (00000008001)
$\mathbf{s}_{231}$ = (FF9A9)	$\mathbf{e}_{231}$ = (00000004000)

IV. PROPOSED DECODING ALGORITHM

The novel SSDDA with the CLT for decoding the binary systematic (41, 21, 9) QR code is presented to further reduce the memory size of the lookup table given in [18].

Given a received word $r = c + e$. Then, the syndrome $\mathbf{s}$ is computed and the weight of this syndrome $w(\mathbf{s})$ is then computed. If $w(\mathbf{s}) = 0$, it denotes that there are no errors in $\mathbf{r}$, namely $r = c$. If $w(\mathbf{s}) \le 4$, then the $v \le 4$ errors are all in the parity check part of r. Now we shift the syndrome left by 21 bits to form a 41-bit length word, and the corrected codeword can be obtained by subtracting (modulo 2) this 41-bit length word from the received word r. If $w(s) > 4$, it means that at least one error is in the message part of r. Next, using the well-known binary search algorithm, we search the CLT whether the syndrome is in the CLT or not. If the syndrome is found in the CLT, namely, $s = s_j$ for some $1 \le j \le 231$, at most $\lceil t/2 \rceil = 2$ errors are in the message part of r, and then subtract the error pattern of the syndrome from the received word r to obtain the corrected codeword. Otherwise, set the counter j to be zero. Subtract the syndrome $\mathbf{s}_j$ in the CLT to obtain the syndrome difference $\mathbf{s}_{dj}$, and compute the weight of this syndrome difference $w(s_{dj})$. If $w(s_{dj}) \le 3$, then we shift this syndrome difference left by 21 bits to form a 41-bit length word. Finally, the vector r subtracts this 41-bit length word and subtracts the corresponding error pattern to correct the r. If $w(s_{dj}) \ge 4$ and $j \le 231$, then set $j = j + 1$ and continue this step. If $w(s_{dj}) \ge 4$ and $j > 231$, then go to next step. Let the vector r be cyclically shifted left by 20 bits and compute its weight $w(s)$. If $w(s) = 3$ or $w(s) = 4$, then $r = r - (s << 21)$ and cyclically shift r left by 21 bits to obtain the corrected $\mathbf{c}$. Next, Compute the syndrome difference $s_{dj} = s - s_j$ for $1 \le j \le 231$ and its weight $w(s_{dj})$. If $w(s_{dj}) \le 3$, then $r = r - (s_{dj} << 21) - e_j$ and cyclically shift r left by 21 bits to obtain the corrected c. Otherwise, declare a decoding failure. The decoding steps for the proposed SSDDA are stated explicitly as follows:

1) Given a received word $r = c + e$.
2) By (7), compute the syndrome of r and its weight $w(s)$.
3) Step 3. If $w(s) = 0$, then no error has occurred. Go to stop.
4) If $w(s) \le 4$, then $c = r - (s << 21)$. Go to step 14.
5) Search whether $\mathbf{s}$ is in the CLT. If $\mathbf{s}$ is in the CLT, that is $s = s_j$ for some $1 \le j \le 231$ corresponding to some error pattern e_j , then $c = r - e_j$. Go to step 14.
6) Compute the syndrome difference $s_{dj} = s - s_j$ for $1 \le j \le 231$ and its weight $w(\mathbf{s}_{dj})$.
7) If $w(s_{dj}) \le 3$, then $c = r - (\mathbf{s}_{dj} << 21) - e_j$. Go to step 14. Otherwise, go to step 8.
8) Set $j = j + 1$. If $j \le 231$, then go to step 6, else go to step 9.
9) Cyclically shift r left by 20 bits. By (7), compute the syndrome of this new $\mathbf{r}$ and its weight $w(\mathbf{s})$.
10) If $3 \le w(s) \le 4$, then $r = r - (s << 21)$ and cyclically shift r left by 21 bits to obtain the corrected $\mathbf{c}$. Go to step 14.
11) Compute the syndrome difference $s_{dj} = s - s_j$ for $1 \le j \le 231$ and its weight $w(\mathbf{s}_{dj})$.
12) If $w(s_{dj}) \le 3$, then $r = r - (s_{dj} << 21) - e_j$ and cyclically shift r left by 21 bits to obtain the corrected $\mathbf{c}$. Go to step 14.
13) $w(\mathrm{e}) > 4$. Declare a decoding failure.
14) Stop or go to step 1 to correct next received word.

V. Simulation Results

The proposed decoder written in C++ program is implemented on an Intel Q6600 PC with Windows XP operating system. All 112,791 error patterns are created and are inputted to the proposed decoder. The SSDDA can perfectly correct all error patterns with an average decoding speed of 11.082μs per error pattern. In comparison, the average decoding time of the decoding algorithms given in [17] and [18] are about 49.355μs and 30.397μs per error pattern, respectively. Obviously, the average decoding time of the SSDDA is the fastest among the three decoders. Simulation results of the three decoding algorithms are shown in Table IV.

TABLE IV: The Time to Decode the (41, 21, 9) QR Code (in μs).

Number of errors	1	2	3	4	Average
Proposed SSDDA	0.3805	1.2768	7.4024	11.209	11.082
FLTDA given in [17]	0.5734	0.6001	0.6141	54.528	49.355
TLDA given in [18]	1.8066	2.0232	2.0611	32.547	30.397

VI. Conclusions

In this paper, a memory-efficient SSDDA with CLT is developed to correct up to four errors for the binary systematic (41, 21, 9) QR code. The main idea of the SDWDA is based on the fact that it exploits the weight of syndrome and the weight of syndrome difference to reduce the memory size of the lookup table. The memory size of the CLT is only 2.03 Kbytes, which can be embeded in any software and decoder chip. Moreover, the decoding time of the proposed SSDDA is faster than the TLDAs given in [17] and [18] in software. It is expected that the memory size and the decoding time can be further reduced in the future while maintaining the same decoding capability of the code.

References

[1] E. Prange, “Cyclic error-correcting codes in two symbols,” Technical report AFCRC-TN-57-103, *Air Force Cambridge Research Center*, Cambridge, Mass. September 1957.

[2] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*; Englewood Cliffs NJ: Prentice-Hall, 1995.

[3] M. Elia, “Algebraic decoding of the (23, 12, 7) Golay codes,” *IEEE Trans. Information Theory*, vol. 33, no. 1, pp. 150–151, Jan. 1987.

[4] I. S. Reed, X. Yin, T. K. Truong, and J. K. Holmes, “Decoding the (24, 12, 8) Golay code,” *IEE Proc. - Computers and Digital Techniques*, vol. 137, no. 3, pp. 202–206, May 1990.

[5] I. S. Reed, X. Yin, and T.K. Truong, “Algebraic decoding of the (32, 16, 8) quadratic residue code,” *IEEE Trans.on Information Theory*, vol. 36, no. 4, pp. 876–880, July 1990.

[6] I. S. Reed, T. K. Truong, X. Chen, and X. Yin, “The algebraic decoding of the (41, 21, 9) quadratic residue code,” *IEEE Trans. on Information Theory*, vol. 38, no. 3, pp. 974–986, May 1992.

[7] R. He, I. S. Reed, T. K. Truong, and X. Chen, “Decoding the (47, 24, 11) quadratic residue code,” *IEEE Trans. on Information Theory*, vol. 47, no. 3, pp. 1181–1186, March 2001.

[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 Communications*, vol. 51, no. 9, pp. 1463–1473, Sept. 2003.

[9] T. K. Truong, Y. Chang, Y. H. Chen, and C. D. Lee, “Algebraic decoding of (103, 52, 19) and (113, 57, 15) quadratic residue code,” *IEEE Trans. on Communications*, vol. 53, no. 5, pp. 749–754, May 2005.

[10] Y. H. Chen, T. K. Truong, Y. Chang, C. D. Lee, and S. H. Chen, “Algebraic decoding of quadratic residue codes using Berlekamp-Massey algorithm,” *Journal of Information Science and Engineering*, vol. 23, no. 1, pp. 127–145, Jan. 2007.

[11] W. K. Su, P. Y. Shih, T. C. Lin, and T. K. Truong, “Decoding of the (48, 24, 12) extended quadratic residue code up to six errors,” in *2008 International Conference on Communication, Circuits and Systems*, Xiamen, China, 2008, pp. 01–05.

[12] T. K. Truong, P. Y. Shih, W. K. Su, C. D. Lee, and Y. Chang, “Algebraic decoding of The (89, 45, 17) quadratic residue code,” *IEEE Trans. on Information Theory*, vol. 54, no. 11, pp. 5005–5011, Nov. 2008.

[13] G. Dubney, I. S. Reed, T. K. Truong, and J. Yang, "Decoding the (47, 24, 11) quadratic residue code using bit-error probability estimates," *IEEE Trans. on Communications*, vol. 57, no. 7, pp. 1986–1993, July 2009.

[14] T. C. Lin, T. K. Truong, H. P. Lee, and H. C. Chang, "Algebraic decoding of the (41, 21, 9) quadratic residue code," *Information Sciences*, vol. 179, no. 19, pp. 3451–3459, Sept. 2009.

[15] T. C. Lin, H. C. Chang, H. P. Lee, S. I. Chu, and T. K. Truong, "Decoding of the (31, 16, 7) quadratic residue code," *Journal of the Chinese Institute of Engineers*, vol. 33, no. 4, pp. 573–580, June 2010.

[16] Y. H. Chen, T. K. Truong, C. H. Huang, and C. H. Chien, "A lookup table decoding of systematic (47, 24, 11) quadratic residue code," *Information Sciences*, vol. 179, no. 14, pp. 2470–2477, June 2009.

[17] 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," *Journal of Information Science and Engineering*, vol. 26, no. 5, pp. 1831–1843, Sept. 2010.

[18] T. C. Lin, H. P. Lee, H. C. Chang, S. I. Chu, and T. K. Truong, "High speed decoding of the binary (47, 24, 11) quadratic residue code," *Information Sciences* vol. 180, no. 20, pp. 4060–4068, Oct. 2010.

[19] T. C. Lin, H. C. Chang, H. P. Lee, and T. K. Truong, "On the decoding of the (24, 12, 8) Golay code," *Information Sciences*, vol. 180, no. 23, pp. 4729–4736, Dec. 2010.

Hung-Peng Lee was born in Taiwan in 1958. He received the B.S. degree in Electronic Engineering from Feng Chia University, Taichung, Taiwan, in 1987, the M.S. degree in Electrical Engineering from the University of Memphis, Memphis, Tennessee, in 1989, and the Ph.D. degree in Information Engineering from the I-Shou University, Kaohsiung, Taiwan, in 2010. The areas of research include error control coding, digital circuit design, and embedded system.

He was a lecture in the Department of Computer Science and Information Engineering and in the Department of Electronic Engineering, Fortune Institute of Technology, Taiwan, from 1981 to 2010. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, Fortune Institute of Technology.

Hsin-Chiu Chang was born in Taiwan in 1954. He received the B.S. degree in Applied Mathematics from National Chung Hsing University, Taichung, Taiwan, in 1976, received his M.S. degrees from National Sun Yat-sen University, Kaohsiung City, Taiwan, in 2005, and the Ph.D. degree in Information Engineering from the I-Shou University, Kaohsiung, Taiwan, in 2010. The research interests of Dr. Chang include statistics, probability, mathmatics, error control coding, and information theory.

He retired in July 2005 from the Kaohsiung City Government, Taiwan. He is currently a part time Assistant Professor in the Department of Computer Science and Information Engineering, Fortune Institute of Technology.