International Journal of
Operations Research

International Journal of Operations Research Vol. 13, No. 4, 165−174 (2016)

Developing Efficient Algorithms of Decoding the Systematic Quadratic Residue Code with Lookup Tables

Chih-Hua Chien*

Department of Information Engineering, I-Shou University

Received July 2016; Revised November 2016; Accepted December 2016

Abstract: The lookup table methods for decoding binary systematic Quadratic Residue (QR) code are presented in this paper. The key ideas behind this decoding technique are based on one to one corresponding mapping between the syndromes and the correctable error patterns. Such algorithms determine the error locations directly by lookup tables without the operations of addition and multiplication over a finite field. Moreover, the methods to dramatically reduce the memory requirement by shift-search decoding are utilized. Two new algorithm have been verified through a software simulation in C language. The new approach is modular, regular and naturally suitable for System on Chip (SOC) software implementation.
Keyword — QR code, Lookup Table method.

1. INTRODUCTION

In this paper, two efficient algorithms called the direct method algorithm and the Lookup Table Decoding (LTD) algorithm to decode the binary systematic QR codes are proposed. The advantage of two algorithms is that they make use of the lookup tables not only to determine the error patterns directly but also to avoid executing the operations of addition and multiplication over a finite field. As a result, the decoding speed of both algorithms are considerably faster than those methods before. Also, the use of a systematic encoder leads to a substantial reduction in the memory required for lookup tables. Furthermore, the LTD method to dramatically reduce the memory requirement only stores the lookup tables for all error patterns of weight less than or equal to t-1, where t is the number of correctable errors. To decode the t-th error case, we utilize the shift-search algorithm. In the direct method algorithm, we construct all error patterns in the table and reduce the memory requirement by systematic encoding. However, the memory requirement is still tremendous larger than LTD algorithm. The proposed algorithms have been verified through a software implementation using C language. Finally, a comparison to decode QR codes with code word length 23, 41 and 47 between the direct method algorithm and the LTD algorithm is given in Table 4.2.

2. BACKGROUND OF QR CODES

In general, QR codes tend to be very good block codes with rates approximating 1/2. We examined the properties of polynomial over Galois field in some detail. Let n be a prime number of the form $n = 8l \pm 1$, where l is a positive integer. The set Q_n of quadratic residues modulo n is the set of nonzero squares modulo n; that is,

(2.1)

Let m be the smallest positive integer such that n divides $2^m - 1$ and let α be a generator of the multiplicative group of all nonzero elements in $GF(2^m)$. Then the element $\beta = \alpha^u$, where $u = (2^m - 1)/n$, is a primitive n-th root of unity in $GF(2^m)$. A binary (n, k, d) QR code is a cyclic code with the generator polynomial $g(x)$ of the form,

*Corresponding author's e-mail: ghostmarty@yahoo.com.tw

$$g(x) = \prod_{i \in Q_n} (x - \beta^i) \tag{2.2}$$

A codeword of the (*n*, *k*, *d*) QR code is a binary vector $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ so that its associated polynomial $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$ is a multiple of $g(x)$. If the codeword **c** is transmitted through a noisy channel, and if the vector $r = (r_0, r_1, \cdots, r_{n-1})$ is received, then the polynomial $r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1}$ corresponding to **r** can be expressed as a sum of the code polynomial $c(x)$ and the error polynomial $e(x) = e_0 + e_1 x + \cdots + e_{n-1} x^{n-1}$. The set of known syndromes is obtained by evaluating $r(x)$ at the roots of $g(x)$, i.e.,

$$S_i = r(\beta^i) = c(\beta^i) + e(\beta^i) = e(\beta^i) = e_0 + e_1(\beta^i) + \cdots + e_{n-1}(\beta^i)^{n-1},\ i \in Q_n \tag{2.3}$$

If, during the data transmission, *v* errors occur in the received vector **r**, then it is said that it is said that the error polynomial has *v* nonzero terms, namely, $e(x) = x^{l_1} + \cdots + x^{l_v}$, where $0 \le l_1 < \cdots < l_v \le n-1$. And, for $i \in Q_n$ the syndrome S_i, written as $S_i = Z_1^i + \cdots + Z_v^i$, where $Z_j = \beta^{l_j}$ for $1 \le j \le v$, are called the error locators.

For any binary QR codes, there is an obvious relation among syndromes, namely, $S_{2i} = S_i^2$, with sub-index modulo *n* if necessary.

Assuminging that *v* errors occurred, we define the error-locator polynomial $\sigma(z)$ to be the polynomial of degree *v*

$$\sigma(z) = \prod_{j=1}^{v} (1 + Z_j z) = 1 + \sum_{j=1}^{v} \sigma_j z^j \tag{2.4}$$

where the coefficients $\sigma_1 = Z_1 + \cdots + Z_v$, $\sigma_2 = Z_1 Z_2 + Z_1 Z_3 + \cdots + Z_{v-1} Z_v, \cdots$, and $\sigma_v = Z_1 \cdots Z_v$. In the past, the most of techniques mainly focus on the Newton identities with either Sylvester resultants or Gröbner bases to calculate the error locator polynomial.

3. CONSTRUCTION OF LOOKUP TABLES AND BINARY-SEARCH ALGORITHM

3.1 Generate the Decoding Table

According to (2.4), we compute the syndromes corresponding to the received error patterns. The superscript *v* represents the *v* errors case of syndrome "S_1", namely "$S_1^{(v)}$". The set of all syndromes corresponding to correctable error patterns that mapped into $S_1^{(v)}$, are defined as follows:

Case 0) For 0 error occurred, the syndrome is $S_1^{(0)} = 0$.

Case 1) For 1 error occurred, the set of syndromes is: $S_1^{(1)} = \{s \mid s = \beta^{i_1} \text{ for } 0 \le i_1 < n\}$.

Case 2) For 2 errors occurred, the set of syndromes is: $S_1^{(2)} = \{s \mid s = \beta^{i_1} + \beta^{i_2} \text{ for } 0 \le i_1 < i_2 < n\}$.

⋮

Case t) For *t* errors occurred, the set of syndromes is: $S_1^{(t)} = \{s \mid s = \beta^{i_1} + \beta^{i_2} + \cdots + \beta^{i_t} \text{ for } 0 \le i_1 < i_2 < \cdots < i_t < n\}$.

To avoid memory fragment, we replace these sets into two lookup tables, called Table A and Table B, to store syndrome values and correctable error patterns, respectively. By these two tables, the index of these syndromes in

the memory can be a consecutive list. Once the consecutive list is available, the binary-search is utilized to search the syndrome more quickly. However, the required memory size is too huge to be implemented in the SOC software. Therefore some methods to reduce the memory usage are necessary.

With the systematic encoding, we could also reduce the memory requirement. According to systematic encoding, we do not need to store all correctable error patterns. Only those error patterns that contain at least one error appeared in the message part are necessary to be stored in the memory.

Following the idea mentioned above, all correctable error pattern sets mapped into $\overline{S}_1^{(v)}$ are defined as follows.

Case 0)For 0 error occurred, the syndrome is $\overline{S}_1^{(0)} = 0$.

Case 1)For 1 error occurred, the set of syndrome is $\overline{S}_1^{(1)} = \{s \mid s = \beta^{i_1} \text{ for } k-1 \le i_1 < n\}$.

Case 2)For 2 errors occurred, the set of syndrome is $\overline{S}_1^{(2)} = \{s \mid s = \beta^{i_1} + \beta^{i_2} \text{ for } k \le i_1 < n, i_1 > i_2 > 0\}$.

⋮

Case t) For t errors occurred, the syndrome of $\overline{S}_1^{(t)} = \{s \mid s = \beta^{i_1} + \beta^{i_2} + \cdots + \beta^{i_t} \text{ for } k \le i_1 < n$, $0 < i_t < i_{t-1} < \ldots < i_1\}$

By the definition above, the necessary error patterns reduces as: $\left|\overline{S}_1^{(1)}\right| + \left|\overline{S}_1^{(2)}\right| + \cdots + \left|\overline{S}_1^{(t)}\right| = \sum_{i=0}^{t}\binom{n}{i} - \sum_{i=0}^{t}\binom{k-1}{i}$.

To reduce more memory requirement, the LTD algorithm by utilizing the systematic encoding property and constructing only 1 to *t*-1 error patterns in the table is proposed. As for the *t*-th error, we apply the shift-search algorithm which is introduced in Section 4. By this idea, the necessary memory sizes are reduced dramatically. The total number of error patterns is thus reduced as $\left|\overline{S}_1^{(1)}\right| + \left|\overline{S}_1^{(2)}\right| + \cdots + \left|\overline{S}_1^{(t-1)}\right| = \sum_{i=0}^{t-1}\binom{n}{i} - \sum_{i=0}^{t-1}\binom{k-1}{i}$.

To describe the construction of tables, let $\mathbf{e} = (\mathbf{e}_d, \mathbf{e}_m) = (e_0, e_1, \ldots, e_{k-2}, e_{k-1}, e_k, \ldots, e_{n-1})$, $\mathbf{e}_d = (e_0, e_1, \ldots, e_{k-2})$ and $\mathbf{e}_m = (e_{k-1}, \ldots, e_{n-1})$, be the error vectors and $\overline{\mathbf{e}}_i$, where $0 \le i \le n-1$, be a binary *n*-tuples in which only the *i*-th coordinate contains a nonzero value. For example, $\overline{\mathbf{e}}_1 = (0,1,0,\ldots,0)$. The pseudo code of generating tables is described as follows.

```
Step 0:   Initial index ← 0.
Step 1:   For 1 Error pattern
          for i1 ← k-1 until n do
          begin
          De_tableA[index] ← record of β^i1;
          e ← ē_i1;
          De_tableB[index] ← e_m;
          index ← index +1;
          end;
Step 2:   For 2 Error patterns
          for i1 ← k-1 until n do
          for i2 ← i1-1 until 0 do
          begin
          De_tableA[index] ← record of β^i1 + β^i2;
          e ← ē_i1 + ē_i2;
          De_tableB[index] ← e_m;
          index ← index +1;
          end;
          ⋮
Step t:   For t Error patterns
          for i1 ← k-1until n do
          for i2 ← i1 until 1 do
```

⋮
for $i_{t-1} \leftarrow i_{t-2}$ until t-2 do
begin
De_tableA[index] $\leftarrow$ record of $\beta^{i_1} + \beta^{i_2} + \cdots + \beta^{i_{t-1}}$;
$\mathbf{e} \leftarrow \overline{\mathbf{e}}_{i_1} + \overline{\mathbf{e}}_{i_2} + \cdots + \overline{\mathbf{e}}_{i-1_t}$;
De_tableB[index] $\leftarrow \mathbf{e}_m$;
index $\leftarrow$ index +1;
end;

Step t+1:De_tableB[] is sorted according to the sorting pattern of the syndrome array of De_tableA[].

Both arrays of De_tableA[.] and De_tableB[.] are called the lookup Table A and the lookup Table B, respectively.

3.2 An Example of Binary-Search Algorithm

To find the syndrome in the lookup Table A efficiently, the binary-search algorithm is necessary and is introduced below. The C-program example of (47, 24, 11) QR code of binary-search algorithm is shown as follows:

```
int Bin_Search(unsigned int syndrome, unsigned int De_tableA[]){
    int high =184806, low = -1, middle;
    while (high - low > 1){
            middle = (low + high) >> 1;
            if (De_tableA[middle] > syndrome)
                high = middle;
            else
                low = middle;
        }
        if (low == -1 || De_tableA[low] != syndrome)
            return -1;
        else
            return low;
}
```

By using of binary-search algorithm, the complexity is O(log n). Although in linear- search algorithm, the complexity is O(n). In other words, the worst case of finding the target element takes 14 and 184,806 comparisons, respectively.

4. THE SHIFT-SEARCH ALGORITHM AND THE DECODING ALGORITHMS

4.1 Shift-Search Algorithm

The idea of the shift-search decoding algorithm is described as follows:
For 0 to t-1 error cases, it is easy to decode the new decoding algorithm directly. Suppose more than t-1 errors, say t errors, are received. In order to decode the code for the t errors case by received $r(x)$, we first change one bit of the received $r(x)$, i.e., let $r'(x) = r(x) + x^p$, $0 \le p \le n-1$. Then, each syndrome is described as follows:

$$S_1 = r'(\beta) = r(\beta) + \beta^p = S_1^{(t)} + \beta^p, \text{ for } 0 \le p \le n-1. \tag{4.1}$$

According to (4.1), $S_1^{(t)}$ becomes $S_1^{(t)} = r(\beta) = e(\beta) = \beta^{i_1} + \beta^{i_2} + \cdots + \beta^{i_t}$, where $0 \le i_1 < i_2 < \cdots < i_t < n-1$. For β^p in (4.1), there are only p's that can match an error position and we need to search at most $n-t-1$ times. If $S_1^{(t)} + \beta^p = S_1^{(t-1)}$, it becomes the syndrome of a $t-1$ errors cases that can be found in the lookup Table A. Otherwise we obtain the syndrome of a $t+1$ errors case (i.e., $S_1^{(t)} + \beta^p = S_1^{(t+1)}$).
For 0 to $t-1$ error cases, it is easy to decode by new decoding algorithm directly. Suppose more than t-1 errors are received, to decode the code for the t-th error case by received $r(x)$, one first changes one bit of the received $r(x)$, i.e.,

let $r'(x) = r(x) + x^p$, $0 \le p \le n-1$. Then, for each syndrome as follows:

$$S_1 = r'(\beta) = r(\beta) + \beta^i = S_1^{(v)} + \beta^p, \text{ for } 0 \le p \le n-1 \tag{4.2}$$

According to (4.2), $S_1^{(t)}$ becomes $S_1^{(t)} = r(\beta) = e(\beta) = \sum_{j=1}^{t} \beta^{i_j}$, where $0 \le i_j < n$. For β^p in (4.2), there are only t p's that can match an error position and one needs to search at most $0 \le p \le n-t$ times. If $S_1^{(t)} + \beta^p = S_1^{(t-1)}$, it becomes the syndrome of a v-1 error case that can be found in the lookup Table A. Otherwise one obtains the syndrome of a v+1 error case (i.e., $S_1^{(v)} + \beta^p = S_1^{(v+1)}$). However, if syndrome $S_1^{(v+1)}$ equals syndrome of $S_1^{(v)}, 1 \le v \le t-1$, the LTD algorithm fails the decoding. To avoid this situation, the necessary condition of correcting exactly v error cases is,

$$\overline{S}_1^{(v)} \cap \overline{S}_1^{(v+1)} = \varphi, \ 1 \le v \le t-1 \tag{4.3}$$

A syndrome is to be satisfied with (4.3) and the following Theorem is necessary.

Theorem 1.
Let $g(x)$ divide $e(x)$, where $g(x)$ is generator polynomial and the weight of $e(x)$ is equal to d, i.e. $wt(e(x)) = d$. Then syndrome $S_1^{(t)} = S_1^{(d-t)}$, where t is the number of errors.

Proof. Since $g(\beta) = 0$ and $g(x)$ divide $e(x)$, one has $e(\beta) = 0$. Let $S_1^{(t)} = S_1^{(t')}$, where t and t' are different numbers of the error pattern case, then $S_1^{(t)} + S_1^{(t')} = e(\beta) = 0$. Denote the weight function by wt(.). Since $wt(e(x)) = d$, an error pattern could be as follows

$S_1 = e(\beta) = \beta^{l_1} + \beta^{l_2} + \cdots + \beta^{l_d}$, where $0 \le l_1 < \cdots < l_d \le n-1$.

The syndrome S_1 can be written as

$$e(\beta) = \underbrace{(\beta^{l_1} + \beta^{l_2} + \cdots + \beta^{l_t})}_{t} + \underbrace{(\beta^{l_{t+1}} + \beta^{l_{t-2}} + \cdots + \beta^{l_d})}_{d-t} = S_1^{(t)} + S_1^{(d-t)} = 0 \tag{4.4}$$

Therefore, (4.4) holds for $S_1^{(t)} = S_1^{(d-t)}$ and the theorem is proved. □

For example, if the minimum distance of this QR code is $d = 11$, we obtain $S_1^{(1)} = S_1^{(10)}$, $S_1^{(2)} = S_1^{(9)}$, $S_1^{(3)} = S_1^{(8)}$, $S_1^{(4)} = S_1^{(7)}$ and $S_1^{(5)} = S_1^{(6)}$. Although some syndromes of six errors that could be equal to the syndromes of 5 errors i.e. $S_1^{(5)} = S_1^{(6)}$, the LTD algorithm would not fail to decode because the syndrome of 5 error case does not exist in the lookup table.

Note that if all 1 to t-1 error patterns appear in the remainder block $\mathbf{r}_d$, it would be regarded as a t error case because the syndrome of these error patterns does not exist in the lookup table. Moreover, if the p-th position also occurs in the remainder block $\mathbf{r}_d$, this syndrome cannot be found in the lookup table by the LTD algorithm. On the other hand, if the p-th position occurs in the message block $\mathbf{r}_m$, the syndrome will be found in the lookup table, i.e., $S_1^{(v+1)} = S_1^{(v)} + \beta^p$, where $(n-1)/2 \le p \le n-1$ and $1 \le v \le t-2$. Therefore, $S_1^{(v+1)}$ could be found in the table and we obtain the error patterns $\mathbf{e}_m$, According to the LTD algorithm, the receive vector could be

corrected as $\mathbf{r}_m^{'} = \mathbf{r}_m + \mathbf{e}_m + \overset{=}{\mathbf{e}}_p = \mathbf{r}_m$, where $\overset{=}{\mathbf{e}}_p$ is set of binary $(n+1)/2$ -tuples in which only the *p*-th coordinate contains a nonzero value at *p*-th position.

The flowchart of the new decoding algorithm is depicted in Figure II. The LTD algorithm, shown in Figure II, has also been programmed in C language.

4.2 The direct method algorithm and the LTD Algorithm

As long as the table is constructed, the algorithm of algorithm of direct method is thus applied to decode the QR codes which can be summarized by the following ten steps:

The direct method algorithm

1) Initialized by letting $p = k-1$.
2) Compute the syndromes S_1 for a received vector **r**.
3) If S_1=0 go to stop.
4) If syndrome is in *T*, go to step 9.
5) Check $S_1 + \beta^p$ value in *T*. If syndrome is in *T,* go to step 8.
6) Compute $p \leftarrow p+1$.
7) If *p*>*n*-1, go to stop. Otherwise return to step 5.
8) If the index corresponding to the syndrome in the lookup table A**,** then the error pattern $\mathbf{e}_m^{'}$ is obtained by index to find in lookup table B. Addition $\mathbf{e}_m^{'}$ by one bit at *p*-th position, obtaining error pattern $\mathbf{e}_m$ (i.e., $\mathbf{e}_m = \mathbf{e}_m^{'} + \overset{=}{\mathbf{e}}_p$ where $\overset{=}{\mathbf{e}}_p$ is a binary *k*-tuples in which contains only a nonzero value at *p*-th position), go to step 10.
9) Look for the index corresponding to the syndrome in the lookup table A, then error pattern $\mathbf{e}_m$ is obtained by index to be found in the lookup table B.
10) Subtract the error pattern $\mathbf{e}_m$ from received vector $\mathbf{r}_m$, obtaining the message block $\mathbf{c}_m$**,** go to stop.

The flowchart of the direct method algorithm is shown in Fig. 1.

To decode all the (23, 12, 7), (41, 21, 9) and (47, 24, 11) QR codes, we construct sorted syndrome values in the memory, namely set *T*. The set *T* is designated as $T = \bigcup_v \overline{S_1}^{(v)}$. This algorithm is summarized by the following ten steps:

The LTD algorithm

1) Initialized by letting $p = 0$.
2) Compute the syndromes S_1 for a received vector **r**.
3) If $S_1 = 0$ go to stop.
4) If syndrome is in *T*, go to step 9.
5) Check $S_1 + \beta^p$ value in *T*. If syndrome is in *T,* go to step 8.
6) Compute $p \leftarrow p+1$.
7) If $p \geq n-t$, go to stop. Otherwise return to step 5.
8) If $p \geq k-1$,look for the index corresponding to the syndrome in the lookup table A**,** then the error pattern $\mathbf{e}_m^{'}$ is obtained by index to find in lookup table B. Addition $\mathbf{e}_m^{'}$ by one bit at *p*-th position, obtaining error pattern $\mathbf{e}_m$ (i.e., $\mathbf{e}_m = \mathbf{e}_m^{'} + \overset{=}{\mathbf{e}}_p$ where $\overset{=}{\mathbf{e}}_p$ is a binary *k*-tuples in which the *p*-th coordinate contains a nonzero value only at *p*-th position), go to step 10.
9) Look for the index corresponding to the syndrome in the lookup table A, then error pattern $\mathbf{e}_m$ is obtained by index to be found in the lookup table B.
10) Subtract the error pattern $\mathbf{e}_m$ from received vector $\mathbf{r}_m$, obtaining the message block $\mathbf{c}_m$**,** go to stop.

The above new decoding schemes have been verified exhaustively for v errors, where $0 \le v \le t$, by a computer simulation. The computer simulation comparisons among errors of QR codes with different methods are shown in the Table 4.1. In computer simulation, we test 1,000,000 codewords for each error case. According to the simulation results, all codewords can be decoded within 4*10-6 second.

The new decoding schemes are suitable for both software and hardware realizations. However, in searching the syndrome of t errors case, we could not avoid that all 1 to *t*-1 error patterns occurred at remainder *d*(*x*). In this situation, the error would be found after executing the shift-search algorithm. Therefore, some computer simulation time would be increased.

5. COLCLUSION

The result in Table 4.1 is quite practical that can be utilized in the SOC software. Moreover, the memory requirement of the direct method algorithm is about ten times larger than of the LTD algorithm, but the CPU time of the direct method algorithm is half of the LTD algorithm. Finally, the flowchart for each algorithm is given in Fig.1 and Fig.2.

Table 4.1 Comparisons among different methods for each QR code

QR Code	*v* errors	Direct method		LTD algorithm	
(23, 12, 7)	1	0.27s	10K Bytes	0.75s	1.03K Bytes
	2	0.42s		0.92s	
	3	0.58s		1.34s	
(41, 21, 9)	1	0.49s	991K Bytes	2.36s	89.4K Bytes
	2	0.95s		1.89s	
	3	1.04s		1.93s	
	4	1.10s		2.92s	
(47, 24, 11)	1	0.516 s	9640 K Bytes	3.765 s	1050 K Bytes
	2	1.125 s		2.968 s	
	3	1.515 s		1.984 s	
	4	2.063 s		1.782 s	
	5	2.094 s		3.884 s	

REFERENCES

1. Assmus, E. F., Jr. and Mattson, H. F., New 5-designs, Theory, vol. 6, pp. 22-151, 1969.
2. Berlekamp, E. R., Algebraic Decoding Theory. New York, McGraw-Hill, 1968.
3. MacWilliams, F. J. and Sloan, N. J. A., The theory of error-correcting codes. Amsterdam, The Netherlands: North-Holland, 1977.
4. Reed, I. S., Yin, X., Truong, T. K., and Holmes, J. K., Decoding the (24, 12, 8) Golay code, Proc. IEE Inst. Elec. Eng., vol. 137, no. 3, 1990, 202–206.
5. Wolfmann, J., A permutation decoding of the (24, 12, 8) Golay code, IEEE Trans. Inform. Theory, vol. IT-29, no. 5, 1983, 748–750.
6. Massey, J. L., Shift-register synthesis and BCH decoding, IEEE Trans. On Inform. Theory, vol. IT-15, no. 1, 1969, 122–127.
7. Blaum, M. and Bruck, J., Decoding the Golay code with Venn diagrams, IEEE Trans. Inform. Theory, vol. 36, no. 4, 1990, 906–910.
8. Elia, M., Algebraic decoding of the (23, 12, 7) Golay code, IEEE Trans. Inform. Theory, vol. 33, no. 1, 1987, 150–151.
9. He, R., Reed, I. S., Truong, T. K., and Chen, X., Decoding the (47, 24, 11) quadratic residue code, IEEE Trans. Inform. Theory, vol. 47, no. 3, 2001, 1181–1186.
10. Reed, S. and Chen, X., Error-Control Coding for Data Networks, Massachusetts: KAP, 1999
11. Wicker, S. B., Error Control Sytems for Digital Comunication and Storage, Canada: PHI. 1985
12. Reed, S., Yin, X., and Truong, T. K., Algebraic decoding of the (32, 16, 8) quadratic residue code, IEEE Trans. Inform. Theory, vol. 36, no. 4, 1990, 876–880.
13. Reed, S., Truong, T. K., Chen, X., and Yin, X., The algebraic decoding of the (41, 21, 9) quadratic residue code,

IEEE Trans. Inform. Theory, vol. 38, no. 3, 1992, 974–985.
14. Reed, S., Shih, M. T., and Truong, T. K., VLSI design of inverse-free Berlekamp-Massey algorithm, Proc. IEE, vol. 138, no. 5, 1991, 295–298.
15. Truong, T. K., Jeng, J. H., and Reed, I. S., "Fast algorithm for computing the roots of the error locator polynomials up to degree 11 in Reed-Solomon decoders, " IEEE Trans. on Commun., vol. 49, no. 5, pp. 779–783, May 2001.
16. Kasami, T., A decoding procedure for multiple-error-correcting cyclic codes, IEEE Trans. Inform. Theory, vol. IT-10, 1964, 134–138.
17. Pless, V., Decoding the Golay codes, IEEE Trans. Inform. Theory, vol. IT-32, no. 4, 1986, 561–567.
18. Chen, X., Reed, I. S., and Truong, T. K., A performance comparison of the binary quadratic residue codes with the 1/2-rate convolutional codes, IEEE Trans. on Inform. Theory, vol. 40, 1994, 126–136.
19. Chen, X., Reed, I. S., and Truong, T. K., Decoding the (73, 37, 13) quadratic residue code, Proc. IEE Comput. Digit. Tech., vol. 141, no. 5, 1994, 253–258.
20. Chen, X., Reed, I. S., Helleseth, T., and Truong, T. K., Use of Gröbner bases to decode binary cyclic codes up to the true minimum distance, IEEE Trans. Inform. Theory, vol. 40, no. 5, 1994, 1654–1661.
21. Chen, X., Reed, I. S., and Truong, T. K., "Decoding the (73, 37, 13) quadratic residue code," Proc. IEE Comput. Digit. Tech., vol. 141, no. 5, pp. 253–258, Sept. 1994.
22. Chang, Y., Truong, T. K., Reed, I. S., Cheng, H. Y., and Lee, C. D., Algebraic decoding of the (71, 36, 11), (79, 40, 15), and (97, 49, 15) quadratic residue codes, IEEE Trans. Commun., vol. 51, no. 9, 2003, 1463–1473.
23. Chen, Y. H., Chien, C. H., Huang, C. H., Truong, T. K., Jing, M. H., "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, 2010.09
24. Chen, Y. H., Truong, T. K., Huang, C. H., Chien, C. H., "A Lookup Table Decoding of Systematic (47, 24, 11) Quadratic Residue Code", Information Sciences, vol. 179, no. 14, pp. 2470-2477, 2009.06

Chih-Hua Chien: *Algorithms of Decoding the Systematic Quadratic Residue Code with Lookup Tables*
IJOR Vol. 13, No. 4, 165−174 (2016)

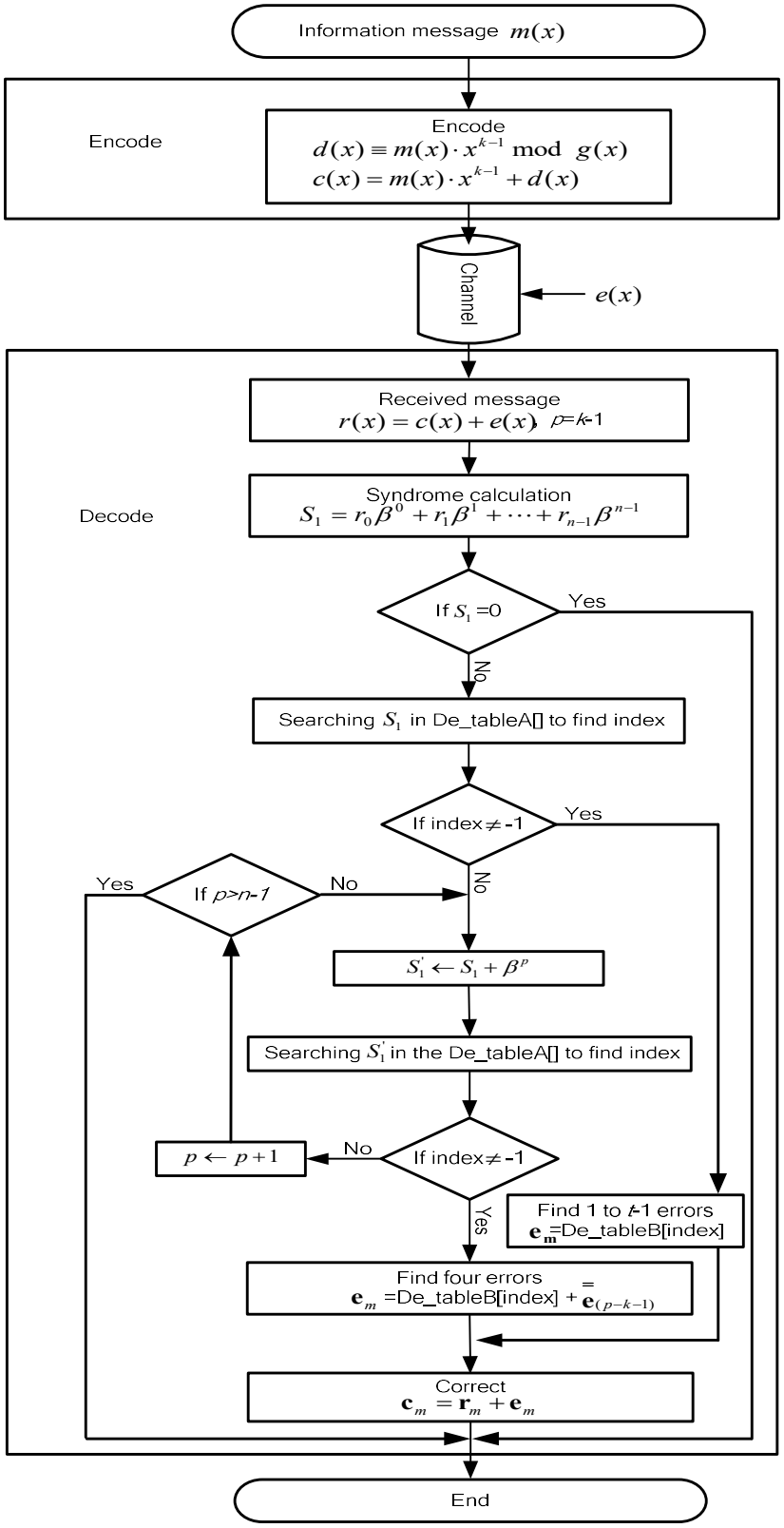

Fig.1 The flowchart of direct method algorithm

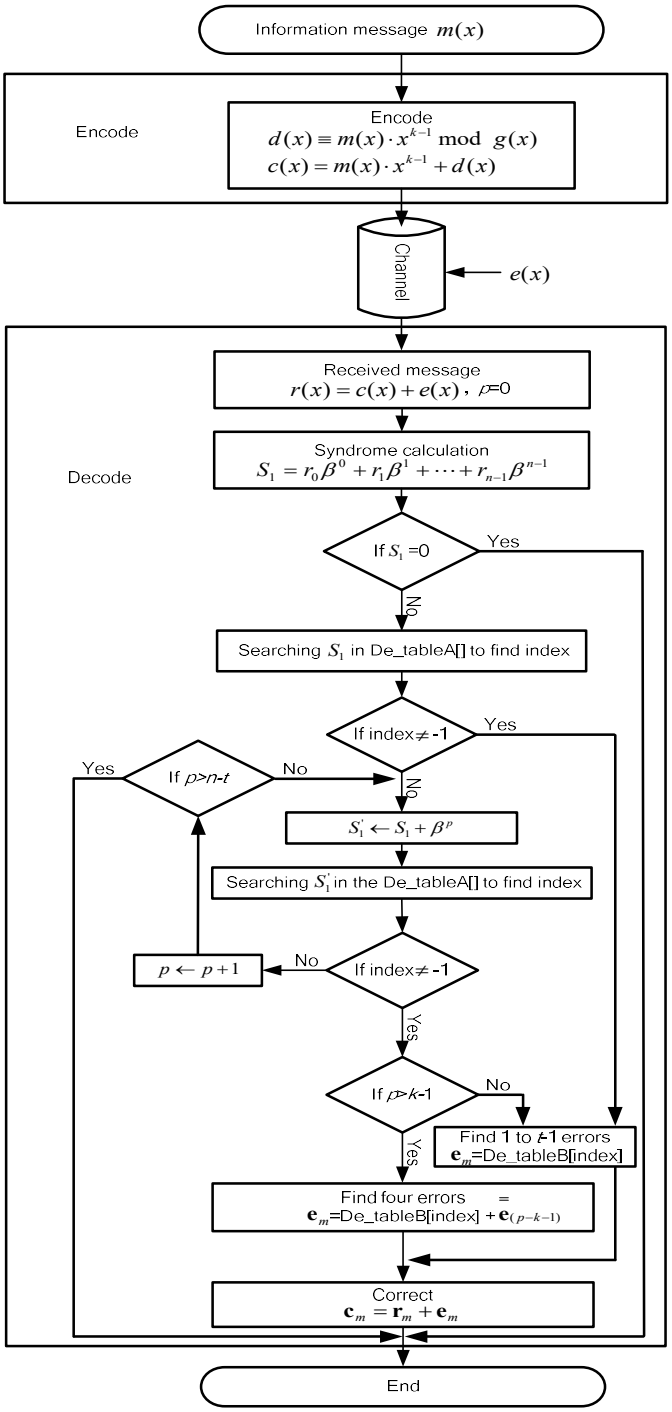

Fig. 2. LTD algorithm Flowchart of the (*n*, *k*, *d*) QR encoder and decoder.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and indebtedness to my advisors Professor T. K. Truong and Professor Yan-Haw Chen, for their valuable guidance in the engineering knowledge and mathematical background.