Scalable Recursive Chien Search Architecture for Block Codes Hard-Decoding

擴展性遞迴秦式搜尋電路於區塊碼解碼之研究

Wei-Wen Hung, Yi-Nan Lin, J.-J. Huang

洪偉文　林義楠　黃植振

摘要

線性區塊碼通道編碼中使用代數解碼過程時對於解出錯誤多項式後，利用秦式搜尋演算法，將場域內之所有元素代入多項式中，以解出滿足此多項式解之所有的根，並找出發生錯誤位元的位置。本文，乃是針對欲解出多項式根的秦式演算重組電路演算使其具遞迴性因而可使計算的核心電路模組達到最小化，同時加入擴展模組的機制，提出一個具遞迴解碼及擴展性的秦式搜尋計算架構。本文提出之架構有利於運用在日益便利之行動嵌入式裝置上電路的實現。

關鍵詞：線性區塊碼、秦式搜尋演算式、遞迴式、可擴展性

ABSTRACT

The Chien search is a crucial arithmetic operation for the error correction codes and the number of roots involved in the block codes decoding. Several efficient modified Chien search algorithm architectures have been proposed. However, there require large circuit space, thus increasing the cost of the decoder. This would be a drawback when designing circuits in systems requiring low cost and low power consumption. However, some cost saving can be attained by compromised speed, as in portable devices and many embedded systems. This paper proposes a scalable and recursive $\sigma_i \cdot x^2 + \sigma_j$, which is the core circuit module of Chien search operation. A scalable and recursive Chien search can thus be obtained based on the proposed architecture. Embedded system engineers may specify a target Chien search operation with appropriate scaling circuits.

Keywords： Linear block codes, Chien search architecture, recursive, scalable

1: INTRODUCTION

The original applications of BCH (Bose − Chaudhuri - Hocquenghem) codes [2, 3] were restricted to binary codes of length 2^m-1 for some integer m. These were extended later by Gorenstein and Zieler (1961) [4] to the nonbinary codes with symbols from Galois field GF(q). BCH codes have found a wide range of applications mainly in modern communication systems, ranging from digital electronic storage devices to the wireless mobiles, such as: Bluetooth [5], Advanced Mobile Phone System (AMPS) [6], etc. The decoding procedure is an important topic for BCH codes. The first decoding algorithm for binary BCH codes was devised by Peterson in 1960 [3]. Since then, Peterson's algorithm has been refined by Berlekamp [7, 8], Massey [9, 10], Chien [1], Forney[11], and many others. The most efficient decoding algorithm and error correction algorithm for binary BCH codes was Berlekamp-Massey algorithm (BMA) that finds the error locator polynomial and Chien search algorithm that used for error correction codes and the number of roots involved.

In this paper we reorganize the error-location polynomial to find the roots based on Chien search algorithm and present a scalable and recursive Chien search circuit scheme. This paper is structured as follows: we first give a brief review of the BCH codes decoding by an example in the next section.

洪偉文　明志科技大學電子工程系教授
林義楠　明志科技大學電子工程系講師
黃植振　明志科技大學電子工程系助理教授

The subsequent section then describes the conventional Chien search (CCS) scheme and explains the proposed the scalable and recursive Chien search (SRCS) scheme. Simulation results over AWGN channels and comparisons are illustrated and discussed in Section 4. Finally, conclusions are made in Section 5.

2: BCH DECODING

Now, we concentrate on the BCH decoding. Since 1960, the first decoding algorithm for binary BCH codes was devised. Then, more and more research for its decoding was explored. This decoding scheme [12] is shown in Fig. 1.

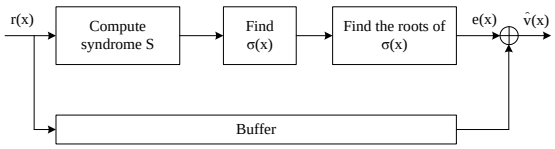

Fig. 1 The binary BCH codes decoding scheme.

In the scheme, S represents the syndrome corresponding to receive polynomial r(x), σ(x) is a error-location polynomial, e(x) is a error polynomial, and $\hat{v}(x)$ is a decoding codeword. Finally, we describe the decoding scheme[13] as following:

Step1: Compute the syndrome $\boldsymbol{S}$ from the received $r(x)$,

$$\boldsymbol{S} = (S_1, S_2, ..., S_{2t}) \quad (2\text{-}1)$$

where $S_i = r(\alpha^i) \quad 1 \le i \le 2t$, and t = $\frac{\lfloor d_{\min} - 1 \rfloor}{2}$ is the error-correcting capability, α is a primitive element of GF(2^m), and $d_{\min}$ is a minimum distance of a code.

Step2: Determine the error-location polynomial $\sigma(x)$ from the syndrome components $S_1, S_2, ..., S_{2t}$. We usually compute $\sigma(x)$ by following three algorithms: (1). Berlekamp-Massey algorithm (BMA), (2). Euclidean algorithm(EA), or (3). Peterson-Gorenstein-Zieler algorithm(PGZ).

$$\sigma(x) \equiv (1+\beta_1 x)(1+\beta_2 x)\cdots(1+\beta_v x)$$
$$= \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_v x^v, \quad v : \quad (2\text{-}2)$$

where $\beta_l = \alpha^{j_l}, 1 \le l \le v,\ v : error\ numbers$

Step3. Determine the error-location numbers β_1, β_2, …, β_v by finding the roots of $\sigma(x)$, and correct errors in $r(x)$. The algorithm of finding the roots of $\sigma(x)$ is called Chien's searching algorithm. $\hat{v}(x) = r(x) + e(x)$, where + is a XOR operation in GF(2).

Next, we describe an instance to explain the BCH codes decoding procedure. Considering the BCH(15, 5, 7) code has triple-error-correcting (t = 3) capability over GF(2^4). Its corresponding the generator polynomial $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$. Assume the message polynomial $u(x) = x + x^2 + x^4$, and code polynomial $v(x) = x + x^2 + x^3 + x^4 + x^8 + x^{11} + x^{12} + x^{14}$, if we received a polynomial $r(x) = 1 + x + x^2 + x^3 + x^4 + x^6 + x^8 + x^{11} + x^{14}$, then comparing the $v(x)$ and $r(x)$ found out three errors on 1、x^6 and x^{12} positions.

Step1: Compute the syndrome S from the received $r(x)$.

$\boldsymbol{S} = (S_1, S_2, ..., S_6)$, where

$$S_1 = r(\alpha) = 1 + \alpha^6 + \alpha^{12} = \alpha$$
$$S_2 = S_1^2 = \alpha^2$$
$$S_3 = r(\alpha^3) = 1 + \alpha^3 + \alpha^6 = \alpha^8$$
$$S_4 = S_2^2 = \alpha^4$$
$$S_5 = r(\alpha^5) = 1 + 1 + 1 = 1$$
$$S_6 = S_3^2 = \alpha$$

Step2: Determine the error-location polynomial σ(x), we use the PGZ algorithm to calculate it. Assume the error pattern polynomial e(x) have ν errors and occurs on $x^{j_1}, x^{j_2} ..., x^{j_v}$, $e(x) = x^{j_1} + x^{j_2} + ... + x^{j_v}$, and

$S_i = r(\alpha^i) = v(\alpha^i) + e(\alpha^i) = e(\alpha^i)$, we know

$$\begin{aligned}
S_1 &= \alpha^{j_1} + \alpha^{j_2} + \ldots + \alpha^{j_\upsilon} \\
S_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \ldots + (\alpha^{j_\upsilon})^2 \\
S_3 &= (\alpha^{j_1})^3 + (\alpha^{j_2})^3 + \ldots + (\alpha^{j_\upsilon})^3 \\
&\vdots \\
S_{2t} &= (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + \ldots + (\alpha^{j_\upsilon})^{2t}
\end{aligned} \tag{2-3}$$

Let $\beta_l = \alpha^{j_l}$, we can find

$$\begin{aligned}
S_1 &= \beta_1 + \beta_2 + \ldots + \beta_\upsilon \\
S_2 &= (\beta_1)^2 + (\beta_2)^2 + \ldots + (\beta_\upsilon)^2 \\
S_3 &= (\beta_1)^3 + (\beta_2)^3 + \ldots + (\beta_\upsilon)^3 \\
&\vdots \\
S_{2t} &= (\beta_1)^{2t} + (\beta_2)^{2t} + \ldots + (\beta_\upsilon)^{2t}
\end{aligned} \tag{2-4}$$

Define an error location polynomial $\sigma(x)$

$$\sigma(x) \equiv (1+\beta_1 x)(1+\beta_2 x)\cdots(1+\beta_v x)$$
$$\Rightarrow \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_v x^v$$

Then, we can get

$$\begin{aligned}
\sigma_0 &= 1 \\
\sigma_1 &= \sum_{i=1}^{\upsilon} \beta_i = \beta_1 + \beta_2 + \ldots + \beta_\upsilon \\
\sigma_2 &= \sum_{i<j} \beta_i \beta_j = \beta_1\beta_2 + \beta_1\beta_3 + \ldots + \beta_{\upsilon-1}\beta_\upsilon \\
&\vdots \\
\sigma_\upsilon &= \prod_{i=1}^{\upsilon} \beta_i = \beta_1\beta_2\ldots\beta_\upsilon
\end{aligned} \tag{2-5}$$

From the equations (2-4) and (2-5), Newton Identity can be obtained as follow:

$$\begin{aligned}
&S_1 + \sigma_1 = 0 \\
&S_2 + \sigma_1 S_1 + 2\sigma_2 = 0 \\
&S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 = 0 \\
&\qquad\vdots \\
&S_\upsilon + \sigma_1 S_{\upsilon-1} + \ldots + \sigma_{\upsilon-1} S_1 + \upsilon\sigma_\upsilon = 0 \\
&S_{\upsilon+1} + \sigma_1 S_\upsilon + \sigma_2 S_{\upsilon-1} + \ldots + \sigma_\upsilon S_1 = 0 \\
&\qquad\vdots \\
&S_{2t} + \sigma_1 S_{2t-1} + \sigma_2 S_{2t-2} + \ldots + \sigma_\upsilon S_{t-\upsilon} = 0
\end{aligned}$$

Because that $i\sigma_i = \begin{cases} 0 & i : even \\ \sigma_i & i : odd \end{cases}$, and $S_{2j} = S_j^{\,2}$.

Newton Identity can be reduced as follow:

$$\begin{aligned}
&S_1 + \sigma_1 = 0 \\
&S_3 + \sigma_1 S_2 + \sigma_2 S_1 + \sigma_3 = 0 \\
&S_5 + \sigma_1 S_4 + \sigma_2 S_3 + \sigma_3 S_2 + \sigma_4 S_1 + \sigma_5 = 0 \\
&\qquad\vdots \\
&S_{2t-1} + \sigma_1 S_{2t-2} + \sigma_2 S_{2t-3} + \ldots + \sigma_t S_{t-1} = 0
\end{aligned}$$

We applied the PGZ method to solve Newton Identity.

$$A\sigma = \begin{bmatrix}
1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
S_2 & S_1 & 1 & 0 & \cdots & 0 & 0 \\
S_4 & S_3 & S_2 & S_1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\
S_{2t-4} & S_{2t-5} & S_{2t-6} & S_{2t-7} & \cdots & S_{t-2} & S_{t-3} \\
S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \cdots & S_t & S_{t-1}
\end{bmatrix}
\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \vdots \\ \sigma_{t-1} \\ \sigma_t \end{bmatrix}
= \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ S_7 \\ \vdots \\ S_{2t-3} \\ S_{2t-1} \end{bmatrix}$$

When the fewer error occurred, the answer can be obtained easily.

One error occurred:

$$\sigma_1 = S_1$$

Two errors occurred:

$$\begin{aligned}
\sigma_1 &= S_1 \\
\sigma_2 &= \frac{S_3 + S_1^{\,3}}{S_1}
\end{aligned}$$

Three errors occurred:

$$\begin{aligned}
\sigma_1 &= S_1 \\
\sigma_2 &= \frac{S_1^{\,2}S_3 + S_5}{S_1^{\,3} + S_3} \\
\sigma_3 &= (S_1^{\,3} + S_3) + S_1\sigma_2
\end{aligned}$$

Four errors occurred:

$$\begin{aligned}
\sigma_1 &= S_1 \\
\sigma_2 &= \frac{S_1(S_1^{\,7} + S_7) + S_3(S_1^{\,5} + S_5)}{S_1(S_1^{\,5} + S_5) + S_3(S_1^{\,2} + S_3)} \\
\sigma_3 &= (S_1^{\,3} + S_3) + S_1\sigma_2 \\
\sigma_4 &= \frac{(S_1^{\,2}S_3 + S_5) + (S_1^{\,3} + S_3)\sigma_2}{S_1}
\end{aligned}$$

Finally, we can find there are three errors occurred in received polynomial r(x). So, we use the formula to obtain the error locations.

$$\sigma_1 = \alpha$$

$$\sigma_2 = \frac{\alpha^2\alpha^8 + 1}{\alpha^3 + \alpha^8} = \alpha^7$$

$$\sigma_3 = (\alpha^3 + \alpha^8) + \alpha\alpha^7 = \alpha^3$$

Thus, we get error-location polynomial $\sigma(x)$

$$\sigma(x) = 1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3$$

Step3: Use Chien search algorithm to find root of $\sigma(x)$. We find that

$$\sigma(1) = 1 + \alpha \times 1 + \alpha^7 \times 1^2 + \alpha^3 \times 1^3 = 0$$

$$\sigma(\alpha^3) = 1 + \alpha \times \alpha^3 + \alpha^7 \times (\alpha^3)^2 + \alpha^3 \times (\alpha^3)^3 = 0$$

$$\sigma(\alpha^9) = 1 + \alpha \times \alpha^9 + \alpha^7 \times (\alpha^9)^2 + \alpha^3 \times (\alpha^9)^3 = 0$$

roots of the σ(x) are $1,\ \alpha^3 = \alpha^{-12},\ \alpha^9 = \alpha^{-6}$.

Then, the decoded codeword is

$$\hat{v}(x) = e(x) + r(x)$$
$$= x + x^2 + x^3 + x^4 + x^8 + x^{11} + x^{12} + x^{14}$$

3: CHIEN SEARCH ALGORITHM

It's a way to find the root of error-location polynomial, if we get the error-location polynomial from decoding algorithm i.e., PGZ, EA, or BMA. Then we can use the Chien's search algorithm to find all roots of this polynomial, Equation (3-1) shows the definition of error-location polynomial:

$$\sigma(x) \equiv \prod_{l=1}^{v}(1 + \beta_l x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_v x^v \quad (3\text{-}1)$$

and because all nonzero element β in GF(2^m) can be expressed in $1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}$,so we will put the elements $1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}$ into the Equation (3-1), if $\sigma(\alpha^j) = 0,\ \text{for}\ j = 0, \cdots, 2^m - 2$ then α^j the root of $\sigma(x)$. Further, we find $(\alpha^j)^{-1}$ that is error location, and we use it to correct error bits. figure. 2[13] shows the classical Chien search circuit. It is implemented by equation (3-1).

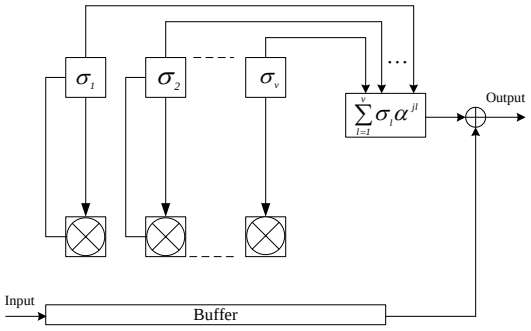

Fig. 2 The classic Chien search circuit.

3.1: SCALABLE RECURSIVE CHIEN SEARCH

In this paper, we proposed a scalable and recursive Chien search architecture for computing the error correction cods and the number of roots involved. Because by the conventional Chien search circuit, it will increase it's area in hardware and compute complexity depends on m. So we proposed a new scheme to improve the conventional Chien search, we define this circuit name as "scalable recursive chien search" (SRCS), it can use a regular circuit to find all root of the error-location polynomial $\sigma(x)$. First we divide the Equation (3-1) into two terms: even-term and odd-term. There apply the Horner's rules to substitute the error-location polynomial. Then, we can reorganize the original Chien search circuit to form a recursive structure as following:

$$\sigma(x) \equiv \prod_{l=1}^{v}(1 + \alpha^{jl} x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_v x^v$$
$$= (\cdots(\sigma_v x^2 + \sigma_{v-2})x^2 \cdots \sigma_1)x +$$
$$(\cdots(\sigma_{v-1} x^2 + \sigma_{v-3})x^2 \cdots \sigma_2)x^2 + 1$$
$$= (\text{odd-term}) + (\text{even-term}) + 1 \quad (3.2)$$

So according the Equation (3-2), we design a core circuit module that can attain a recursive unit: $\sigma_i \cdot x^2 + \sigma_j$, defines as a process element (P.E.) as shown in figure 3.

Step1: we get $\sigma_v, \cdots, \sigma_1$ from finding $\sigma(x)$ decoding algorithm.

Step2: we multiple σ_v and x, where x belongs to a set of $\{\alpha^0, \alpha^1 \cdots \alpha^{2^m-2}\}$, then add σ_{v-1}. Recursively, until add 1 if results are equal to zero, then x is the root of error-location polynomial. Otherwise, it is not a root.

Fig. 3 is the proposed recursive Chien search circuit. In this figure we only use one XOR gate and one polynomial multiple in a kernel of the computing process unit (P.E.). Using this structure to solve the roots of $\sigma(x)$, we can save more area of VLSI circuit than classical Chien search circuit.

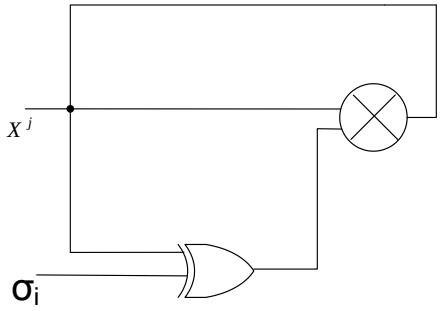

Fig. 3 A recursive core unit: P.E.

The overall Chien search can be implemented to a scalable and recursive architecture by using three P.E. units and shown in Fig. 4. If using this circuit we only have to use $2/(2^m - 2)$ area less than conventional Chien search and it can compute all roots of this $\sigma(x)$ by recursive form.

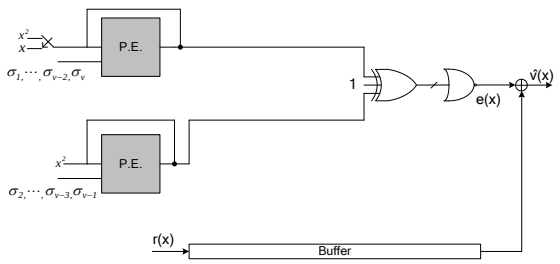

Fig. 4 Scalable recursive Chien search circuit.

In figure 4, upper of parts is odd-term operations and lower of parts is even-term operations, they are independent, so they can increase twice times process speed than only using one structure, but only increase one P.E. module circuit in whole structure. Following the scheme we can get an algorithm for SRCS architecture.

Step1: we divide the $\sigma(x)$ into become odd and even term.

Step2: odd-term: $(\cdots(\sigma_v x^2 + \sigma_{v-2})x^2 \cdots \sigma_1)x$

even-term: $(\cdots(\sigma_{v-1} x^2 + \sigma_{v-3})x^2 \cdots \sigma_2)x^2$,

where, variables $\sigma_v, \cdots, \sigma_1$ from finding $\sigma(x)$ decoding algorithm, and $x \in \{\alpha^0, \alpha^1 \cdots \alpha^{2^m-2}\}$ in GF(2^m).

Step3: we add the two terms by XOR.

4: SIMULATION RESULTS

In this section, we conduct a series of experiments to evaluate the effectiveness of the recursive Chien search algorithm we proposed for BCH codes decoding. In this simulation, a data codeword of 15 bits is considered and 100,000 blocks are transmitted. The BCH coding parameters (n, k, d_{min}) is equal to (15, 5, 7), i.e., the code has tripe error-correcting capability. The overall code rate is 1/3. The coded bits are modulated using binary phase shift keying (BPSK) and white Gaussian noise with a double-sided power spectral density of $No/2$ is added to the modulated signal. Decoding used the PGZ algorithm to find out the error-location polynomial. Error locations and correcting error bits are determined by the classical Chien search and the proposed scalable and recursive Chien search circuit scheme.

The results of the simulation are shown in Fig. 5. Comparing the results between CCS and SRCS is no coding gain sacrificing. The effectiveness of the proposed SRCS scheme is guaranteed.

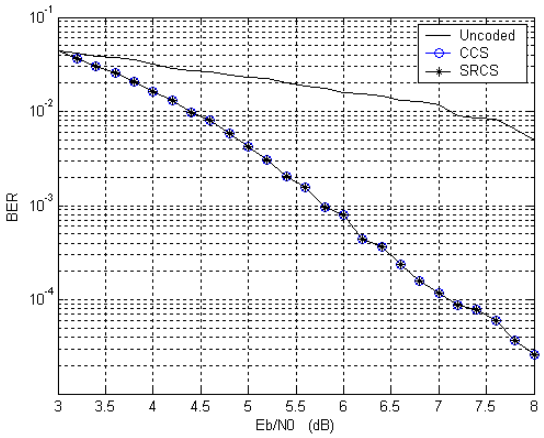

Fig. 5 Comparison of BER performances for the "Uncoded" case, the "CCS" case and the "SRCS" case.

5: CONCLUSION

In this paper, a scalable recursive Chien search is presented based on the proposed architecture which is a scalable and recursive $\sigma_i \cdot x^2 + \sigma_j$ as a core circuit module of Chien search operation. It has been shown by simulations that the CCS and SRCS have the same performance coding gain in BER. But, the proposed SRCS circuit is more regular and saving-area $2/t$ than the CCS circuit. Embedded system engineers may specify a target Chien search operation with appropriate scaling-t circuits. Furthermore, this structure can be applied to a non-binary BCH decoding.

REFERENCES

1. R. T. Chien, "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," IEEE Trans. Inform. Theory, IT-10, pp. 357-363, October 1964.
2. A. Hocquenghem, "Codes corecteurs d'erreurs," Chiffres, No. 2, pp. 147-156, 1959.
3. R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," Inform. Control, No. 3, pp. 68-79, March 1960.
4. D. Gorenstein and N. Zierler, "A class of cyclic linear error-correcting codes in p^m symbols," J. Soc. Ind. Appl. Math., No. 9, pp. 107-214, June 1961.
5. Specification of the Bluetooth System, V1.0A, July 1999.
6. D. J. Goodman, Wireless Personal Communications Systems, Addison-Wesley Longman, 1997.
7. E. R. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem Codes," IEEE Trans. Inform. Theory, IT-11, pp. 577-580, October 1965.
8. E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.
9. J. L. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," IEEE Trans. Inform. Theory, IT-11, pp. 580-85, October 1965.
10. J. L. Massey, "Shift-register synthesis and BCH decoding," IEEE Trans. Inform. Theory, IT-15, pp. 122-127, January 1969.
11. G. D. Forney, "On decoding BCH codes," IEEE Trans. Inform. Theory, IT-11, pp. 549-557, October 19.
12. Robert H. Morelos-Zaragoza, "The Art of Error Correcting Coding," Wiley, 2002
13. S. Lin and D. J. Costello Jr., "Error Control Coding: Fundamental and Applications, NJ: Prentice Hall, 1988