# Chien's Search Algorithm with Recursive Structure for the Study of BCH Hard-Decoding

# 遞迴秦式搜尋演算法於 BCH 硬式解碼之研究

Wei-Wen Hung Yi-Nan Lin Jhih-Jhen Hwang

洪偉文 林義楠 黃植振

## 摘要

通道編碼中使用線性區塊碼之代數解碼過程中，對於解出錯誤多項式後，須進一步地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，以進一步地找出發生錯誤位元的位置來。本文，乃是針對解出多項式根的秦式演算法，利用 Horner 法則來重新無損地組織其原解碼計算式，並提出一個修正型的遞迴解碼架構。此一修正型的演算架構，主要具有使核心計算之電路單元，較精簡化、及較規則化，並可遞迴式計算，完成錯誤位元更正的好處。此一架構，將有利於運用在日益便利之行動裝置上其電路的實現。

關鍵詞：線性區塊錯誤更正碼、秦式搜尋演算式、遞迴式架構、Horner 法則

## ABSTRACT

A modified Chien search[1] circuit architecture for binary Bose-Chaudhuri-Hocquenghem(BCH) codes decoding is studied using the Horner's rules to substitute the definition of error-location polynomial. Thus, we get a regular, recursive circuit for this procedure. Our simulation results demonstrate that incorporating the recursive Chien search (RCS) scheme into the BCH codes decoding process in bit error rate (BER) is no degradation, and it makes the circuit saving-area about 1/error-correcting-bit(t) than conventional Chien search (CCS) scheme. The proposed scheme can be easily implemented on VLSI circuit. Furthermore, this structure can be applied to the nonbinary BCH codes decoding.

Keywords： linear block error code, Chien search algorithm, recursive structure, Horner's rule

### 1: INTRODUCTION

The original applications of BCH 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 the electronic storage devices to the wireless mobile, 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) and Chien's searching algorithm.

In this paper we shall reorganize the error-location polynomial to find the roots based on Chien search algorithm and present a 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. The subsequent section then describes the conventional Chien search (CCS) scheme and explains the proposed the recursive Chien search (RCS) scheme. Simulation results over AWGN channels and comparisons are illustrated and discussed in 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

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.

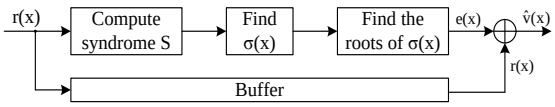


Fig. 1 The BCH codes decoding scheme.

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

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

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

where $S_i = r(\alpha^i)$ $1 \le i \le 2t$, 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).

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

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

Step3. Determine the error-location numbers $\beta_1$, $\beta_2$, …, $\beta_v$ 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 $x^{j_1}, x^{j_2} ..., x^{j_\upsilon}$, e(x) = $x^{j_1} + x^{j_2} + ... + x^{j_\upsilon}$, and $S_i$ = r($\alpha^i$) = v($\alpha^i$) + e($\alpha^i$) = e($\alpha^i$), we know the

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

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

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

Define an Error Location Polynomial σ(x)

$$
\begin{aligned}
\sigma(x) &= (1+\beta_1 x)(1+\beta_2 x)\cdots(1+\beta_v x) \\
&= \sigma_0+\sigma_1 x+\sigma_2 x^2+\cdots+\sigma_v x^v
\end{aligned}
$$

Then, we can get

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

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

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

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:

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

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.

$$
\begin{aligned}
\sigma_1 &= \alpha \\
\sigma_2 &= \frac{\alpha^2\alpha^8+1}{\alpha^3+\alpha^8} = \alpha^7 \\
\sigma_3 &= (\alpha^3+\alpha^8)+\alpha\alpha^7 = \alpha^3
\end{aligned}
$$

Thus, we get error-location polynomial σ(x)

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

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

$$\sigma(1)=1+\alpha*1+\alpha^{7}*1^{2}+\alpha^{3}*1^{3}=0$$
$$\sigma(\alpha^{3})=1+\alpha*\alpha^{3}+\alpha^{7}*(\alpha^{3})^{2}+\alpha^{3}*(\alpha^{3})^{3}=0$$
$$\sigma(\alpha^{9})=1+\alpha*\alpha^{9}+\alpha^{7}*(\alpha^{9})^{2}+\alpha^{3}*(\alpha^{9})^{3}=0$$

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

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

, 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 $\alpha^{j}$ the root of σ(x). Further, we find $(\alpha^{j})^{-1}$ that is error location, and we use it to correct error bits. Fig. 2[13] shows the classical Chien search circuit. It is implemented by Equation (3-1).

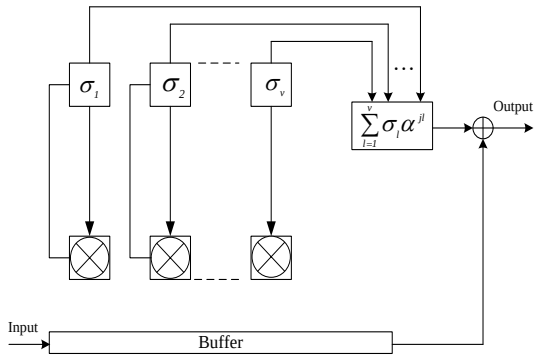


Fig. 2 The classic Chien search circuit.

### 3.1: RECURSIVE CHIEN SEARCH ALGORITHM

In this paper, we proposed a modified method for Chien search algorithm, it applied 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.

$$\sigma(x)=\prod_{l=1}^{v}(1+\alpha^{j_l}x)=1+\sigma_1 x+\sigma_2 x^{2}+\cdots+\sigma_v x^{v}$$
$$=(\cdots(\sigma_v x+\sigma_{v-1})x+\cdots+\sigma_1)x$$

(3-2).

Using this recursion form makes the circuit it can save area about 1/t for the implementation of hardware than the CCS circuit. By this way we can also calculate the root of error-location polynomial and more regularly. Now we use the Equation (3-2) to realize a decoding algorithm:

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

Step2, we multiple $\sigma_v$ and x, where x belongs to a set of $\{\alpha^{0},\alpha^{1}\cdots\alpha^{2^m-2}\}$, then add $\sigma_{v-1}$. Recursively, until add 1 if results is 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. Using this structure to solve the roots of σ(x), we can save more area of VLSI circuit than classical Chien search circuit.

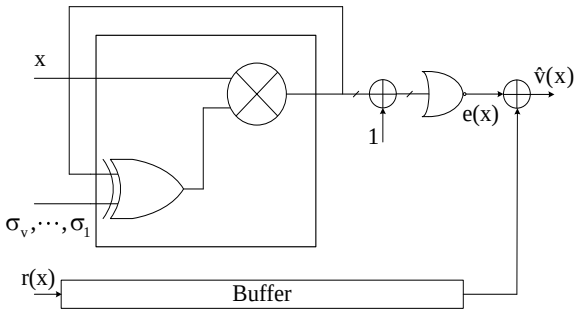


Fig. 3 The recursive Chien search circuit.

## 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 triple 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 are determined by the proposed recursive Chien's search circuit scheme.

The results of the simulation are shown in Fig. 4. Comparing the results CCS and RCS are no coding gain different. It proved our proposed RCS scheme is effective.

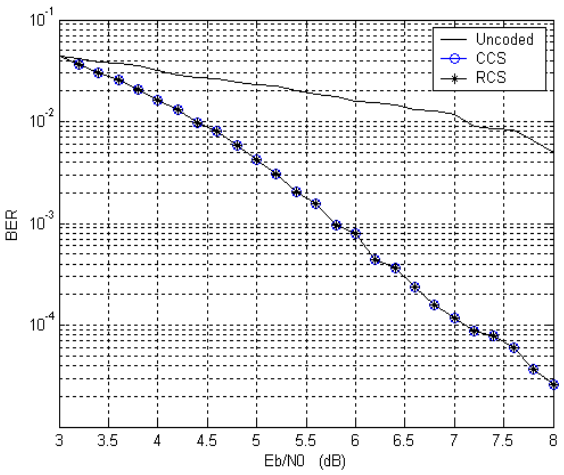


Fig. 4 Comparison of BER performances for the "Uncoded" case, the "CCS" case and the "RCS" case.

## 5: CONCLUSION

In this paper, we proposed a recursive Chein search circuit to decode the binary BCH codes. The idea of our approach is the application of Horner's rules. It has been shown by simulations that the CCS and RCS have the same performance in the coding gain, i.e., they are the same BER (Bit Error Rate). But, our proposed RCS circuit is more regular and saving-area 1/t than the CCS circuit. Thus the RCS circuit will be more benefit than CCS in the VLSI implementations. Furthermore, this structure can be applied to nonbinary 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] Shu Lin and Danial J. Costello. Jr., “Error Control Coding 2nd edition,” Prentice Hall, 2004.