公告本

申請日期：90.11.29	案號：90129718
類別：H03M13/00	

（以上各欄由本局填註）

發明專利說明書 566008

一、發明名稱	中文	解碼錯誤訂正碼時用以解答鍵方程式多項式之方法及其裝置
	英文	Apparatus for Solving Key Equation Polynomials in Decoding Error Correction Codes
二、發明人	姓名（中文）	1.李鎮宜 2.張錫嘉
	姓名（英文）	1.Chen-Yi Lee 2.Hsie-Chia Chang
	國籍	1.中華民國 2.中華民國
	住、居所	1.新竹市博愛街75之1號 2.基隆市忠三路73號
三、申請人	姓名（名稱）（中文）	1.國立交通大學
	姓名（名稱）（英文）	1.National Chiau-Tung University
	國籍	1.中華民國
	住、居所（事務所）	1.新竹市大學路1001號
	代表人姓名（中文）	1.張俊彥
	代表人姓名（英文）	1.C.Y.Chang

本案已向

國(地區)申請專利	申請日期	案號	主張優先權
無			

有關微生物已寄存於

寄存日期	寄存號碼
無	

四、中文發明摘要　（發明之名稱：解碼錯誤訂正碼時用以解答鍵方程式多項式之方法及其裝置）

在解碼時，通常為了錯誤訂正目的而將接收字碼經編碼處理。據此，一種在鍵方程式解答步驟中，用以計算錯誤定位器多項式和錯誤求值器多項式之一種方法，在本發明加以揭露。藉由本發明，多項式可經由多個中間步驟而產生，而上述步驟可由最少量之硬體線路來加以實施。中間步驟之數目需要一相對應之運算時間週期（cycle）數目，以完成多項式之計算。依所選擇之（N,K）碼而定，計算多項式所須之運算時間週期數目將是介於上游資料計算所須時間之內。特別的是，只須小數量之暫存器及有限場乘法器（FFM），且不須有限場反轉器（FFI）之有效安排配置也加以揭露。使用這些新方法，僅使用4t+2 ρ+4個暫存器，3個FFM且不須FFI之有效節省面積的架構也已揭

英文發明摘要　（發明之名稱：Apparatus for Solving Key Equation Polynomials in Decoding Error Correction Codes）

It is an object of the present invention to provide a method and apparatus for solving key equation polynomials in the decoding of codewords. Based upon the Euclidean algorithm, it can be implemented with minimal hardware circuitry and provide a method and apparatus for solving key equation within a t-step iterative decoding procedure while the prior art architectures require at most 2t iterations. It is yet another object of the present invention to provide a method and

四、中文發明摘要　(發明之名稱：解碼錯誤訂正碼時用以解答鍵方程式多項式之方法及其裝置)

示，用以實施由無反轉Euclidean演算法推導而得之方法。此外，所提出之架構也可用來求出Forney徵兆多項式。本發明之方法及其裝置可廣泛應用於各種具有適當編碼長度之RS碼和BCH碼，包括錯誤更正以及抹除更正之情況。

英文發明摘要　(發明之名稱：Apparatus for Solving Key Equation Polynomials in Decoding Error Correction Codes)

apparatus for solving key equation polynomials without decreasing the overall decoding speed of the decoder. Briefly, in a presently invention, a method for computing error locator polynomial and error evaluator polynomial in the key equation solving step of the error correction code decoding process is presented whereby the polynomials are generated through at most t intermediate iterations that can be implemented with minimal amount of hardware circuitry. However, depending on the

四、中文發明摘要　(發明之名稱：解碼錯誤訂正碼時用以解答鍵方程式多項式之方法及其裝置)

英文發明摘要　(發明之名稱：Apparatus for Solving Key Equation Polynomials in Decoding Error Correction Codes)

selected (N,K) code, the number of cycles required for the calculation of the polynomials would be within the time required for the calculation of upstream data. Additionally, a presently invention for computing the error locator polynomial and the error value polynomial employs an efficient scheduling of a small number of registers and finite-field multipliers (FFMs) without the need of finite-field inverters (FFIs) is illustrated. Using these new methods, a new area-efficient

四、中文發明摘要　（發明之名稱：解碼錯誤訂正碼時用以解答鍵方程式多項式之方法及其裝置）

英文發明摘要　（發明之名稱：Apparatus for Solving Key Equation Polynomials in Decoding Error Correction Codes）

architecture that uses only $4t+2\rho+4$ registers and three FFMs and no FFIs is presented to implement the inversionless Euclidean algorithm. This method and architecture can be applied to a wide variety of RS and BCH codes with suitable code sizes.

五、發明說明（1）

發明背景：

經過多種不同媒介，由發信位置至目地位置之資料傳輸，由於傳送路徑、及（或）媒介本身所造成之雜訊，會造成傳輸資料之錯誤。因此，傳輸之資料不會與所接收到之資料相同。為了判定接收資料之錯誤與否，已發展出各種方法及技術，用以偵測和訂正接收資料。方法之一為產生包括訊息部份（所傳送之資料）和奇偶性部份（parity part，據以實施錯誤訂正之信息）的字碼（codeword）。在本文中，字碼係針對原始資料施行編碼操作而得。字碼具有同一形式，為包括N個符號之信息，其中前K個符號係為訊息符號，而後N-K個符號係為奇偶性符號。

知名之錯誤訂正碼中，BCH碼（Bose-Chaudhuri-Hocquenghen Codes）以及RS碼（Reed-Solomon Codes）是在通訊領域和貯存器系統應用中，最廣為使用之區塊碼（Block Codes）。BCH碼以及RS碼之數學理論基礎在「E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968」以及「S. Lin and D.J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1983」兩本著作均有詳細之解釋說明。

一個（N,K）BCH或RS碼具有K個訊息符號和N個編碼符

五、發明說明 (2)

號，其中BCH碼之符號屬於GF（q）集合而RS碼之符號屬於GF（q^m）集合。在$N=2^m-1$及$N-K \leqq mt$之情形下，一個二進位（N,K）BCH碼能訂正t個錯誤符號。而在$t=\ \llcorner(N-K-\rho)/2\lrcorner$之情形下，一個（N,K）RS碼可以訂正t個錯誤符號及ρ個抹除（erasure）符號。對二進位BCH碼而言，藉由發現錯誤符號之位置，可很簡單地訂正一個錯誤符號。對RS碼而言，則必須藉由發現錯誤符號之位置及其錯誤值，來訂正一個錯誤符號。此外，抹除符號在RS碼中之定義為一已知錯誤位置上之錯誤；換句話說，只要發現錯誤值就可以訂正一個抹除符號。

使用普遍之RS解碼器架構，倘若只需訂正錯誤符號，其方法步驟可摘要為以下四個步驟：（1）由所接收之字碼計算出徵兆（syndrome），（2）運算出錯誤定位器多項式及錯誤求值器多項式，（3）發現錯誤所在位置，以及（4）運算出錯誤值。假設需被訂正的是錯誤以及抹除符號，則四個步驟修正為如下：（1）由所接收之字碼及抹除位置計算出Forney徵兆，（2）計算錯誤及抹除定位器多項式以及錯誤及抹除求值器多項式，（3）發現錯誤所在位置，以及（4）計算錯誤及抹除之訂正值。

請參照顯示一般習知的解碼步驟之第1A圖。所接收之資料R(x)輸入徵兆計算器10以產生徵兆多項式S(x)，其代表字碼之錯誤型式，可藉以訂正錯誤。徵兆係僅依錯誤型

五、發明說明 (3)

式而定，而非依傳輸之字碼而定。接著，將徵兆輸入一鍵方程式解答器（Key Equation Solver）12，產生一錯誤定位器多項式$\sigma(x)$、及一錯誤求值器多項式$\Omega(x)$。錯誤定位器多項式指示發生錯誤之位置，而錯誤求值器多項式可用以求出錯誤之值。下一步驟，錯誤定位器多項式傳至Chien搜尋器14以解出方程式之根，其表示錯誤符號之位置。錯誤求值器16接收根以及錯誤求值器多項式$\Omega(x)$，產生相對應於根之錯誤值。

在實施鍵方程式解答器（上述第2步驟）時，本要係解出如下之鍵方程式：

$$S(x)\ \sigma(x) = \Omega(x) \bmod x^{N-K} \qquad (1)$$

其中，$S(x)$為徵兆多項式，$\sigma(x)$為錯誤定位器多項式，以及$\Omega(x)$為錯誤求值器多項式。當同時訂正錯誤及抹除時，$\sigma(x)$即為錯誤及抹除定位器多項式而和$\Omega(x)$為錯誤及抹除求值器多項式；此時$\sigma(x)=\lambda(x)\Lambda(x)$，$\lambda(x)$和$\Lambda(x)$分別為錯誤定位器多項式以及抹除定位器多項式。參照第1B圖，其顯示用於訂正錯誤及抹除符號之一般處理步驟。徵兆計算器20除接收$R(x)$，也接收抹除資料，藉以產生Forney徵兆多項式$T(x)$。鍵方程式解答器22處理$T(x)$並產生錯誤及抹除求值器多項式$\Omega(x)$、及錯誤及抹除定位器多項式$\sigma(x)$。錯誤及抹除定位器多項式輸入Chien搜尋器24以判定錯誤符號發生之位置，而錯誤及抹除求值器多項式和錯誤及抹除位置兩者均輸入一錯誤及抹除值求值

五、發明說明 (4)

器，用以產生錯誤及抹除值。

經常用以解出鍵方程式之技術包括Berlekamp-Massey演算法，Euclidean演算法。這些演算法之衍生作法用來同時解出錯誤符號及抹除符號，詳細的介紹可以參照先前所提，Blahut著作裡頭的說明。在這裡我們提出了所謂無反轉分解（inversionless decomposed）Euclidean架構，在維持相同的解碼速度下，大幅地減少硬體複雜度。

習知技術係應用傳統Euclidean演算法，以計算錯誤定位器多項式和錯誤求值器多項式，以及作為設計電路之基礎。然而，每一種演算法均需要很大數量之暫存器，有限場乘法器（finite-field multiplier，FFM），以及或許需要有限場反轉器（finite-field inverter，FFI）。每一暫存器，FFM和FFI都會轉換成硬體線路並實作於積體電路中；因此，我們想要導出一有效率之多項式解法，並且可減少實作演算法時硬體線路之大小（複雜度）。暫存器和FFM之數目基本上是為變數t跟ρ之函數，其中t=「(N-K-ρ)/2」，變數t及ρ分別表示可以解回之錯誤符號及抹除符號數目的最大值。表一僅就錯誤訂正（而非錯誤及抹除訂正），顯示各種不同演算法及暫存器、FFM和FFI所相對應之個數。

請參照第1表所示，實作傳統之Euclidean演算法，

五、發明說明 (5)

Reed在「VISI Implementation of A Pipeline Reed-Solomon Decoder, IEEE Transaction on Computers, vol. C-34, pp. 393-403, May 1985」中提出一種不需要FFI，但需要8t個暫存器，8t個FFM的架構。而在「An Efficient Architecture for Implementing the Modified Euclidean Algorithm, the 9th NASA Symposium on VLSI Design, Nov. 2000」一文中，Song揭露一種不需要FFI，但是需要6t+4個暫存器以及6t+2個FFM的實現架構。

另一方面，Wu所揭示之演算法中需要7t+5個暫存器並可大幅減少FFM數目至t+「(t+1)/2┐，然而仍須具有相當複雜度之FFI。上述之演算法揭露於「An Area-efficient Versatile Reed-Solomon Decoder for ADSL, IEEE International Symposium on Circuits and Systems, May 1999」。以上架構若應用於錯誤及抹除訂正時，所須之暫存器以及FFM數目將會更高。

因此，在實作演算法時，一種無須使用FFI並可使暫存器及FFM之數目降至最小且電路複雜度不受錯誤或錯誤及抹除更正之影響的無反轉方法及其裝置是眾所期望。

本發明之發明目的與概述：

本發明之最主要的目的為提供一種在字碼解碼時，用

五、發明說明（6）

以解答鍵方程式多項式之方法及其裝置。在Euclidean演算法中，本裝置可以用最少之硬體線路來加以實作。本發明之另一目的為提供一種解答鍵方程式多項式之方法及其裝置，最多僅需要t個重複的步驟就可以完成整個解碼程序。本發明之又一目的為提供一種解答鍵方程式多項式之方法及其裝置，其不會降低解碼器之整體解碼速度。

簡而言之，在較佳實施例中，揭示一種在錯誤訂正碼解碼過程之鍵方程式解答步驟中，用以計算錯誤定位器多項式和錯誤求值器多項式之方法，經由上述方法之t個中間步驟而產生多項式，並且可以用最小數目之硬體線路來加以實作。然而，依所選擇之（N,K）碼而定，計算多項式所需之運算時間週期數目會在計算上游資料（upstream data）所需時間之範圍內。此外，在較佳實施例中，有效率地規劃小數量之暫存器及有限場乘法器（FFM）而無須使用有限場反轉器（FFI），用以計算錯誤定位器多項式和錯誤求值器多項式之較佳方法也加以揭示。使用這些新方法，實作由無反轉Euclidean演算法所導出之方法，一種僅使用$4t+2\rho+4$個暫存器以及3個FFM而無須FFI的具面積效率（節省線路面積）之分解架構亦加以揭示。這方法及架構可廣泛地應用於各種具有適當編碼長度之RS碼和BCH碼。特別是，在較佳實施例中，解答鍵方程式多項式之方法及其裝置也可以用來計算先前所介紹過的Forney徵兆多項式T(x)。這方法及架構可同時應用於更正錯誤字碼

五、發明說明 (7)

或錯誤及抹除字碼。

本發明之一優點為提供一種方法及裝置，用以在字碼解碼時，解答鍵方程式多項式。在Euclidean演算法中，可以最少數量之硬體線路來加以實施。本發明之另一優點為提供一種方法及裝置，最多僅需要t個中間步驟就可以解出鍵方程式多項式，並且不會降低解碼器之整體解碼速度。本發明之又一優點為提供一種相同的方法及裝置，除了可以解答鍵方程式多項式，也可以用來計算Forney徵兆多項式。

茲為使貴審查委員，對本發明之結構特徵優點及達成之功效有更進一步之了解與認識，謹佐以較佳之實施例，並配合所附圖式做詳細之說明如後。

發明之詳細說明：

在本發明之詳細說明中，首先將顯示本發明中較佳實施例：首先，我們將介紹僅需要使用t個反覆步驟之改良後（modified）解碼程序，之後提出新的無反轉Euclidean演算法，其所產生之改良後錯誤及抹除定位器多項式$\underline{\sigma}(x)$和改良後錯誤及抹除求值器多項式$\underline{\Omega}(x)$可以跟原始Euclidean演算法之錯誤及抹除定位器多項式$\sigma(x)$和錯誤及抹除求值器多項式$\Omega(x)$解出相同的錯誤所在

五、發明說明 (8)

位置跟錯誤及抹除訂正值。參照在此所使用之符號，沒有"_"之符號如$\Omega(x)$和$\sigma(x)$係引用原始Euclidean演算法（具有反轉），而有"_"之符號如 $\underline{\Omega}(x)$和$\underline{\sigma}(x)$係引用改良後無反轉Euclidean演算法。

此外，我們藉分解所提出之無反轉Euclidean演算法來減少硬體複雜度到僅需$4t+2\rho+4$個暫存器以及3個FFM，並用所提出之架構求解Forney徵兆多項式。最後，列出條件式以表示可適用此架構之N、K值。

Euclidean解碼程序：

為了說明Euclidean演算法，我們將鍵方程式(1)重寫為：

$$\Omega(x) = x^{N-K}Q(x) + T(x)\lambda(x) \qquad (2)$$

其中$Q(x)$可當成被除式$T(x)\lambda(x)$以及除式x^{N-K}之商式；$T(x)=S(x)\Lambda(x)$為Forney徵兆多項式；$\sigma(x)=\Lambda(x)\lambda(x)$為錯誤及抹除定位器多項式，也就是抹除定位器多項式$\Lambda(x)$與錯誤定位器多項式$\lambda(x)$之乘積。因此，可經由一個類似Euclidean演算法求出x^{N-K}以及$T(x)$之最大公因式的程序來解出錯誤及抹除求值器多項式$\Omega(x)$，該解碼程序如下所示：

$\Omega^{(-1)}(x)=x^{N-K}$

五、發明說明 (9)

$$\Omega^{(0)}(x)=T(x)$$
$$\Omega^{(1)}(x)=\Omega^{(-1)}(x)-\Omega^{(0)}(x)Q^{(1)}(x)$$
$$\cdots$$
$$\Omega^{(i)}(x)=\Omega^{(i-2)}(x)-\Omega^{(i-1)}(x)Q^{(i)}(x) \qquad (3)$$
$$\cdots$$
$$\Omega^{(n)}(x)=\Omega^{(n-2)}(x)-\Omega^{(n-1)}(x)Q^{(n)}(x)$$

其中每一個重複步驟（iteration）相當於一次除法運算，由第（3）式可知，第i個除法運算之被除式與除式分別為第i-2個除法運算之餘式$\Omega^{(i-1)}(x)$與第i-1個除法運算之餘式$\Omega^{(i)}(x)$；而在求出商式$Q^{(i)}(x)$及餘式$\Omega^{(i)}(x)$之後再重覆進行下一個除法運算。n個除法運算過後，假設所得到第n個除法運算之餘式$\Omega^{(n)}(x)$即為所求之錯誤及抹除求值器多項式，$\Omega(x)$。此外，在「Error-Control Coding for Data Networks, Kluwer Academic, 1999」一書中所介紹的Euclidean演算法之延伸形式，可用一個除了起始條件，其餘皆相當類似上述解碼程序之方法來求出錯誤及抹除定位器多項式，$\sigma(x)$，該解碼程序如下所示：

$$\sigma^{(0)}(x)=\Lambda(x)$$
$$\sigma^{(1)}(x)=\sigma^{(0)}(x)Q^{(1)}(x)$$
$$\sigma^{(2)}(x)=\sigma^{(1)}(x)Q^{(2)}(x)+\sigma^{(0)}(x)$$
$$\cdots$$

五、發明說明 (10)

$$\sigma^{(i)}(x)=\sigma^{(i-1)}(x)Q^{(i)}(x)+\sigma^{(i-2)}(x) \qquad (4)$$

$$\cdots$$

$$\sigma^{(n)}(x)=\sigma^{(n-1)}(x)Q^{(n)}(x)+\sigma^{(n-2)}(x)$$

其中$\Lambda(x)=\Pi(1+\chi_j x)$表示抹除定位器多項式，s為真實發生之抹除符號個數，χ_j表示第j個抹除定位值；所有步驟之$Q^{(i)}(x)$就是上述Euclidean演算法中除法運算之商式$Q^{(i)}(x)$。n個運算步驟過後，所算出之$\sigma^{(n)}(x)$即為所求之錯誤及抹除定位器多項式，$\sigma(x)$。而由第（3）式跟第（4）式可以證明出，多項式$\Omega^{(i)}(x)$與$\sigma^{(i+1)}(x)$次方和會等於常數，N-K+s。

改良後解碼程序：

藉由事先限制並計算每個次方為一的商式，所提出之改良後解碼程序如下所示：

起始條件

$$A^{(0)}(x)=x^{N-K},\ M^{(0)}(x)=\Omega^{(0)}(x)=T(x)$$

$$a^{(0)}(x)=0,\ m^{(0)}(x)=\sigma^{(0)}(x)=\Lambda(x)$$

For(i=0 to t)

$$\delta=\deg(A^{(i)}(x)),\ \Delta=\deg(M^{(i)}(x))$$

if(deg($\sigma^{(i)}(x)$) $\leqq \Delta$)

五、發明說明 (11)

$$q_1^{(i)} = A_\delta^{(i)} / M_\Delta^{(i)}$$

$$q_0^{(i)} = 0 \quad \text{for} \quad \delta = \Delta$$

$$q_0^{(i)} = [(M_\Delta^{(i)} A_{\delta-1}^{(i)} + M_{\Delta-1}^{(i)} A_\delta^{(i)})] / M_\Delta^{(i)} M_\Delta^{(i)} \quad \text{for} \quad \delta \neq \Delta$$

$$\Omega^{(i+1)}(x) = A^{(i)}(x) + x^{\delta-\Delta-1} M^{(i)}(x) q^{(i)}(x) \qquad (5)$$

$$\sigma^{(i+1)}(x) = a^{(i)}(x) + x^{\delta-\Delta-1} m^{(i)}(x) q^{(i)}(x) \qquad (6)$$

if(deg($\Omega^{(i+1)}(x)$) < Δ)

$A^{(i+1)}(x) = \Omega^{(i)}(x)$, $M^{(i+1)}(x) = \Omega^{(i+1)}(x)$

$a^{(i+1)}(x) = \sigma^{(i)}(x)$, $m^{(i+1)}(x) = \sigma^{(i+1)}(x)$

else

$A^{(i+1)}(x) = \Omega^{(i+1)}(x)$, $M^{(i+1)}(x) = M^{(i)}(x)$

$a^{(i+1)}(x) = \sigma^{(i+1)}(x)$, $m^{(i+1)}(x) = m^{(i)}(x)$

else

$\Omega(x) = \Omega^{(i)}(x)$, $\sigma(x) = \sigma^{(i)}(x)$ 結束

其中 $q^{(i)}(x) = q_0^{(i)} + q_1^{(i)} x$ 表示第i個運算步驟之商式，$A_\delta^{(i)}$ 及 $M_\Delta^{(i)}$ 分別為 $A^{(i)}(x)$ 以及 $M^{(i)}(x)$ 之領頭係數。$A^{(i)}(x)$ 與 $M_{(i)}(x)$ 表示為了求出第i個步驟中，第i個錯誤及抹除求值器多項式 $\Omega^{(i)}(x)$ 的輔助運算多項式；同理，$a^{(i)}(x)$ 與 $m^{(i)}(x)$ 表示為了求出第i個步驟中，第i個錯誤及抹除定位器多項式 $\sigma^{(i)}(x)$ 的輔助運算多項式。值得注意的是，倘若只考慮更正錯誤符號（沒有抹除符號發生），此時抹除定

五、發明說明 (12)

位器多項式Λ(x)等於1，且Forney徵兆多項式T(x)等於徵兆多項式S(x)。

與第（3）式相較，倘若將$A^{(i)}(x)$與$M^{(i)}(x)$視為第i-2個除法運算之餘式$\Omega^{(i-1)}(x)$與第i-1個除法運算之餘式$\Omega^{(i)}(x)$；也就是將$A^{(i)}(x)$與$M^{(i)}(x)$當成第i個除法運算之被除式與除式。藉由所提出之改良後解碼程序，$\Omega^{(i-1)}(x)$與$\Omega^{(i)}(x)$的次方差等於$A^{(i)}(x)$與$M^{(i)}(x)$的次方差$\delta^{-}\Delta$，僅需要「$(\delta-\Delta+1)/2$」個運算步驟就可以求出第（3）式中之餘式Ω(x)；換句話說，倘若被除式$\Omega^{(i-1)}(x)$與除式$\Omega^{(i)}(x)$的次方差為1，僅僅需要改良後解碼程序的一個運算步驟就可以完成第（3）式中的第i個除法運算。

當$\Omega^{(i)}(x)$之次方（degree）小於$\sigma^{(i)}(x)$之次方時，所提出之改良後解碼程序將結束運算；此時$\sigma^{(i)}(x)$即為次方$s+\nu$的錯誤及抹除定位器多項式σ(x)。其中s與ν分別表示真實發生之錯誤符號與抹除符號之個數。由於$\sigma^{(0)}(x)$的次方等於抹除定位器多項式Λ(x)的次方s，所以在整個解碼程序中，$\sigma^{(i)}(x)$的次方將從s到$s+\nu$。由第（4）式可知在解碼過程中$\sigma^{(i)}(x)$的次方等於$\sigma^{(i-1)}(x)$與$Q^{(i)}(x)$的次方和，而商式$Q^{(i)}(x)$的次方至少為1，因此所提出之改良後解碼程序在最糟的情況下（所有$Q^{(i)}(x)$的次方都為1，即$\delta-\Delta=1$，完成每個除法運算只需要一個運算步驟）需要ν個運算步驟。由於真實發生之錯誤符號個數ν必

五、發明說明 (13)

須小於t，所提出之改良後解碼程序最多需要t個運算步驟來解答鍵方程式多項式。

無反轉解碼程序：

針對所提出之改良後解碼程序，消去其中的反轉（inverse）運算後，一種新穎之無反轉解碼程序如下所示：

起始條件

$$\underline{A}^{(0)}(x)=x^{N-K},\ \underline{M}^{(0)}(x)=\underline{\Omega}^{(0)}(x)=T(x)$$
$$\underline{a}^{(0)}(x)=0,\ \underline{m}^{(0)}(x)=\underline{\sigma}^{(0)}(x)=\Lambda(x)$$

For (i=0 to t)

$$\delta=\deg(\underline{A}^{(i)}(x)) \qquad \Delta=\deg(\underline{M}^{(i)}(x))$$

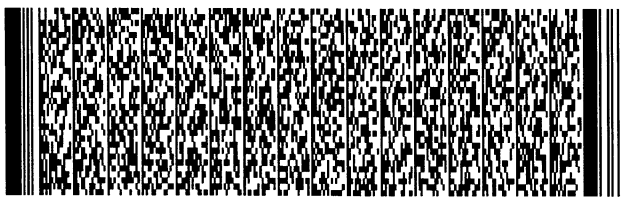
if($\deg(\sigma^{(i)}(x)) \leq \Delta$)

$$\underline{q}_1^{(i)}(x)=\underline{A}_{\delta}^{(i)}\underline{M}_{\Delta}^{(i)}$$

$$\underline{q}_0^{(i)}(x)=0 \quad \text{for } \delta \neq \Delta$$

$$\underline{q}_0^{(i)}(x)=M_{\Delta}^{(i)}\underline{A}_{\delta-1}^{(i-1)}+\underline{M}_{\Delta-1}^{(i)}\underline{A}_{\delta}^{(i)} \quad \text{for } \delta=\Delta$$

$$\underline{\Omega}^{(i+1)}(x)=\underline{M}_{\Delta}^{(i)}\underline{M}_{\Delta}^{(i)}\underline{A}^{(i)}(x)+x^{\delta-\Delta-1}\underline{M}^{(i)}(x)\underline{q}^{(i)}(x) \qquad (7)$$



五、發明說明 (14)

$$\underline{\sigma}^{(i+1)}(x)=\underline{M}_{\Delta}^{(i)}\underline{M}_{\Delta}^{(i)}\underline{a}^{(i)}(x)+x^{\delta-\Delta-1}\ \underline{m}^{(i)}(x)\underline{q}^{(i)}(x) \qquad (8)$$

if(deg $\underline{\Omega}^{(i+1)}(x) < \Delta$)

$$\underline{A}^{(i+1)}(x)=M^{(i)}(x) \qquad \underline{M}^{(i+1)}(x)=\underline{\Omega}^{(i+1)}(x)$$
$$\underline{a}^{(i+1)}(x)=m^{(i)}(x) \qquad \underline{m}^{(i+1)}(x)=\underline{\sigma}^{(i+1)}(x)$$

else

$$\underline{A}^{(i+1)}(x)=\underline{\Omega}^{(i+1)}(x) \qquad \underline{M}^{(i+1)}(x)=\underline{M}^{(i)}(x)$$
$$\underline{a}^{(i+1)}(x)=\underline{\sigma}^{(i+1)}(x) \qquad \underline{m}^{(i+1)}(x)=\underline{m}^{(i)}(x)$$

else

$\underline{\Omega}(x)=\underline{\Omega}^{(i)}(x)$ $\underline{\sigma}(x)=\underline{\sigma}^{(i)}(x)$ 結束

其中$\underline{\Omega}(x)$以及$\underline{\sigma}(x)$分別表示改良後錯誤及抹除求值器多項式以及改良後錯誤及抹除定位器多項式。我們可以證明出 $\underline{\Omega}(x)$和$\underline{\sigma}(x)$ 可以跟原來演算法之錯誤及抹除定位器多項式$\sigma(x)$和錯誤及抹除求值器多項式$\Omega(x)$產生相同的錯誤及抹除所在位置跟錯誤及抹除訂正值。與其它架構相比，所提出之無反轉Euclidean演算法不止去除了具相當複雜度的反轉運算，也提供了一個最多僅需要t個步驟就可以完成的解碼程序。

分解架構：

五、發明說明 (15)

針對所提出之無反轉Euclidean演算法，我們以個別的係數運算來代替整體的多項式運算，提出一種分解架構。如前所述，假設將$A^{(i)}(x)$與$M^{(i)}(x)$分別視為第（3）式中第i個除法運算之被除式 $\underline{\Omega}^{(i-1)}(x)$以及除式$\underline{\Omega}^{(i)}(x)$，則上述之第（7）式與第（8）式可以重寫成：

$$\underline{\Omega}^{(i+1)}(x)=\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}^{(i-1)}(x)+x^{\delta-\Delta-1}\,\underline{\Omega}^{(i)}(x)\underline{q}^{(i)}(x) \quad (9)$$

$$\underline{\sigma}^{(i+1)}(x)=\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}_{\Delta}^{(i)}\,\underline{\sigma}^{(i-1)}(x)+x^{\delta-\Delta-1}\,\underline{\sigma}^{(i)}(x)\underline{q}^{(i)}(x) \quad (10)$$

其中δ與Δ分別表示被除式 $\underline{\Omega}^{(i-1)}(x)$以及除式$\underline{\Omega}^{(i)}(x)$之次方，且第i個運算步驟之商式$\underline{q}^{(i)}(x)=\underline{q}_0^{(i)}+\underline{q}_1^{(i)}x$ 意味著餘式 $\underline{\Omega}^{(i+1)}(x)$之次方至少為$\delta-2$。因此在不失去一般性的情況下，令$\delta-\Delta=1$並分解上述兩個方程式如下：

$$\underline{\Omega}_j^{(i+1)}=\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}_j^{(i-1)}+\underline{\Omega}_j^{(i)}\underline{q}_0^{(i)}+\underline{\Omega}_{j-1}^{(i)}\underline{q}_1^{(i)}$$
$$0\leqq j\leqq \delta-2 \quad (11)$$

$$\underline{\sigma}_{\lambda}^{(i+1)}=\underline{\Omega}_{\Delta}^{(i)}\,\underline{\Omega}_{\Delta}^{(i)}\,\underline{\sigma}_{\lambda}^{(i-1)}+\underline{\sigma}_{\lambda}^{(i)}\underline{q}_0^{(i)}+\underline{\sigma}_{\lambda-1}^{(i)}\underline{q}_1^{(i)}$$
$$0\leqq \lambda\leqq \phi+1 \quad (12)$$

五、發明說明 (16)

其中 $\underline{\Omega}_j^{(i+1)}$ 以及 $\underline{\sigma}_\lambda^{(i+1)}$ 分別表示第i個運算步驟之 $\underline{\Omega}^{(i+1)}(x)$ 第j個係數以及 $\underline{\sigma}^{(i+1)}(x)$ 第λ個係數；而

$\underline{\Omega}^{(i+1)}(x) = \underline{\Omega}_0 + \underline{\Omega}_1 x + \ldots\ldots + \underline{\Omega}_{\delta-2} x^{\delta-2}$

，δ 表示 $\underline{\Omega}^{(i-1)}(x)$ 多項式之最高次，

$\underline{\sigma}^{(i+1)}(x) = \underline{\sigma}_0 + \underline{\sigma}_1 x + \ldots\ldots\ldots + \underline{\sigma}_{\phi+1} x^{\phi+1}$

，ϕ 表示 $\underline{\sigma}^{(i)}(x)$ 多項式之最高次。由第（11）式及第（12）式可知，倘若 $\underline{\Omega}_\Delta^{(i)}$ $\underline{\Omega}_\Delta^{(i)}$ 、$\underline{q}_0^{(i)}$ 以及 $\underline{q}_1^{(i)}$ 可以被事先算出，則在每個運算時間週期中，只需要三個有限場乘法器就可以求出 $\underline{\Omega}_j^{(i+1)}$ 以及 $\underline{\sigma}_\lambda^{(i+1)}$。請參考第2表所提出之無反轉分解架構，在第i個運算步驟之各時間週期的詳細運算：

如第2表中所示，於運算時間週期j=0計算 $\underline{\Omega}_0^{(i+1)}$ 所需要的 $\Omega_\Delta^{(i)}$ $\Omega_\Delta^{(i)}$ 、$q_0^{(i)}$ 之值，已於起始運算（initialization）時算出。同理，於運算時間週期j≧1，計算 $\Omega_j^{(i+1)}$ 所需要的$q_1^{(i)}$之值，也於運算時間週期j=0算出。此外，每一個運算時間週期僅需要三個有限場乘法器，並且 $\underline{\sigma}^{(i+1)}(x)$的計算過程與 $\underline{\Omega}^{(i+1)}(x)$ 相當類似。

使用上述之無反轉分解Euclidean演算法，使得以一實施作為鍵方程式解答器之3-FFM架構成為可能，且如第2圖所示；其中第2圖上之標示表示在計算 $\underline{\Omega}^{(i+1)}(x)$ 時，某一特定時間之對應值。與表2相比較，第2A圖表示起始運算之狀態，第2B圖表示運算時間週期j=0時，計算出$q_1^{(i)}$

五、發明說明 (17)

以及 $\underline{\Omega}_0^{(i+1)}$ 之狀態。 $\underline{\Omega}^{(i+1)}(x)$ 之其他係數的計算過程如第2C圖所示。由於 $\underline{\sigma}^{(i+1)}(x)$ 的計算過程與 $\underline{\Omega}^{(i+1)}(x)$ 的計算過程非常類似，我們可以將不同值灌入此3-FFM架構來算出 $\underline{\sigma}^{(i+1)}(x)$ ，如第2D圖所示。

這一架構可使用於錯誤訂正或是錯誤及抹除訂正。相較於先前提出之架構需6t+2至8t個FFM，本發明之較佳實施例可以大大地減少硬體複雜度至僅需3個FFM。然而，為了完成第i運算步驟，較佳實施例之架構需要 $\delta+\Psi+1$ 個運算時間週期，但是習知技術之架構僅需要2至3個運算時間週期。

然而，使用本發明之架構，用以產生資料所須增加之額外時間並不會減慢系統整體之處理速度。這是由於系統整體之處理速度主要受到其上游步驟，如第1圖之徵兆產生器所限制，而非鍵方程式解答器。因此，為了使任何資料被接收和處理，皆必須等待上游步驟所得之結果；也就是說所提出之架構雖然需要較多之運算時間，只要在計算上游資料所需時間之內（N個運算時間週期），仍不會影響整體之解碼速度。

此外，本發明之方法及其裝置，也將所需要的暫存器數目降至最低。如前所述，$\underline{\sigma}^{(i)}(x)$ 與 $\underline{\Omega}^{(i-1)}(x)$ 這兩個多項式之次方和為 $\delta+\phi=N-K+s\leqq 2t+\rho$ ，其中t與 ρ 分別表

五、發明說明 (18)

示在字碼解碼時可以解回之錯誤符號與抹除符號的最大值。因此，本發明之較佳實施例需要$2t+\rho+2$個暫存器來儲存$\underline{\Omega}^{(i)}(x)$與$\underline{\sigma}^{(i+1)}(x)$與的所有係數，並且需要另外的$2t+\rho+2$個暫存器來儲存$\underline{\Omega}_{(i-1)}(x)$與$\underline{\sigma}^{(i)}(x)$的所有係數。這表示當使用所提出之解碼程序來計算$\underline{\Omega}^{(i+1)}(x)$以及$\sigma^{(i+2)}(x)$，總共需要$4t+2\rho+4$個暫存器；倘若只需更正錯誤符號，只需使用$4t+4$個暫存器，而表一所列之其他架構則需要$6t+4$至$8t$個暫存器。

再者，本發明之方法及其裝置，也可以用來實施Forney徵兆多項式$T(x)$之計算，$T(x)$之定義為：

$$T(x)=S(x)\,\Lambda(x) \mod x^{N-K} \qquad (13)$$

其中$\Lambda(x)=\Pi(1+\chi_j x)$，表示抹除定位器多項式，s為真實發生之抹除符號個數而χ_j為第j個抹除定位值。由下列之計算程序可求出$T(x)$：

起始條件

$T^{(0)}(x)=S(x)$

For(i=0 to t)

if(2i < s)

$$\Lambda^{(i)}(x)=(1+\chi_{2i+1}x)(1+\chi_{2i+2}x) \qquad (14)$$

五、發明說明 (19)

$$T^{(i+1)}(x)=T^{(i)}(x)\ \Lambda^{(i)}(x) \bmod X^{N-K} \qquad (15)$$

else

$T(x)=T^{(i)}(x)$ 結束

其中 $\Lambda^{(i)}(x)$ 為計算第i個中間步驟之Forney徵兆多項式 $T^{(i+1)}(x)$ 的輔助多項式，並且可以展開為 $1+\Lambda_1^{(i)}x+\Lambda_2^{(i)}x^2$，因此可分解第i個中間步驟之Forney徵兆多項式 $T^{(i+1)}(x)$ 為：

$$T_\tau^{(i+1)}=T_\tau^{(i)}+T_{\tau-1}^{(i)}\ \Lambda_1^{(i)}+T_{\tau-2}^{(i+1)}\ \Lambda_2^{(i)}$$

$$0 \leqq \tau \leqq N-K-1 \qquad (16)$$

其中 $T_\tau^{(i+1)}$ 表示第i個Forney徵兆多項式 $T^{(i+1)}(x)$ 的第 τ 個係數，其計算過程與第（11）式之 $\underline{\Omega}_j^{(i+1)}$ 或第（12）式之 $\underline{\sigma}_\lambda^{(i+1)}$ 非常類似；因此，所提出之方法及其裝置也可以用來求解Forney徵兆多項式T(x)，如第2E圖所示。

應用條件：

在使用本實施例之3-FFM架構，計算 $\underline{\sigma}(x)$ 和 $\underline{\Omega}(x)$ 所須之運算時間週期總數，以及其對系統整體效能表現之

五、發明說明 (20)

潛在影響是我們所關心的。從所提出的反覆演算法中，第(11)式中$0 \leqq j \leqq \delta-2$ 表示計算$\underline{\Omega}^{(i+1)}(x)$ 所需要之運算時間週期數目為$\delta-1$；第(12)式中$0 \leqq \lambda \leqq \phi+1$表示計算$\sigma^{(i+1)}(x)$ 所需要之運算時間週期數目為$\phi+2$；因此再加上計算$g_1^{(i)}$ 及$g_0^{(i)}$ 所需要的一個運算時間週期後，所提出之解碼程序針對每一個的運算步驟，總共需要$\delta+\phi+2$個運算時間週期，其中$\delta+\phi=N-K+s \leqq 2t+\rho$。因此，在一個可以更正t個錯誤符號以及$\rho$個抹除符號之(N,K)RS碼中，使用t個運算步驟之無反轉分解架構，所須運算時間週期之總數將小於$2t^2+\rho t+2t$。

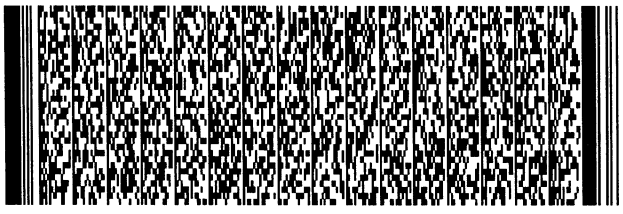
各種不同之(N,K) RS碼，其N-K之編碼長度範圍介於4至16者，運算時間週期所須之總數(最大值)已計算出並列於表三。若是N大於所須之運算時間週期數目，則本發明之方法及其裝置可用以減少硬體複雜度，而仍保持整體之解碼速度。

在通訊和貯存系統中，BCH和RS碼有多種之應用，均可以由本發明之方法及其裝置獲益。例如，數位影音光碟(digital versatile disks；DVDs)係使用RS相乘碼，其在列方向為(182,172)，而在欄方向為(208,192)；數位電視廣播使用(204,188)之RS碼；CD-ROM使用兩組較小之RS碼，包括(32,28)及(28,24)RS碼；在光纖之海底纜線通訊系統中，(255,239)RS碼已被使用並且當

五、發明說明（21）

成抵抗猛暴性（burst）錯誤之標準；在無線通訊中，AMPS蜂巢式行動電話系統使用（40,28）及（48,36）之二進位BCH碼，其均為（63,51）BCH碼之縮減碼。可訂正2個錯誤（N-K=12，m=6）之（63,51）碼，需小於12運算時間週期（t=2，表三之第一列）。所有諸如此類和其他方面之應用，均可從本發明之方法及其裝置有所獲益。

雖然本發明已以較佳實施例揭露如上，然其並非用以限定本發明，任何熟悉本項技藝者，在不脫離本發明之精神和範圍內所提出之修改和潤飾，均係涵括在本發明之保護範圍內，而本發明之保護範圍視後附之申請專利範圍所界定者為準。



圖式簡單說明

圖式之簡單說明：

第1A圖 顯示解碼具錯誤訂正功能字碼時之處理方塊圖。

第1B圖 顯示解碼具錯誤及抹除訂正功能字碼時之處理方塊圖

第2A圖 顯示解碼程序中第i個步驟中起始運算之狀態

第2B圖 顯示解碼程序中第i個步驟中運算時間週期j=0之狀態

第2C圖 顯示用以求出解碼程序中第i個步驟之第j個錯誤及抹除求值器多項式係數之狀態

第2D圖 顯示用以求出解碼程序中第i個步驟之第λ個錯誤定位器多項式係數之狀態

第2E圖 顯示相同的方法及裝置，用以求出Forney徵兆多項式時第i個步驟之第τ個係數之狀態。

符號說明：

10 徵兆計算器

12 鍵方程式解答器

14 Chien搜尋器

16 錯誤求值器

20 徵兆計算器

圖式簡單說明

22 鍵方程式解答器

24 Chien搜尋器

26 錯誤及抹除求值器

30 有限場乘法器

32 有限場乘法器

34 有限場乘法器

36 有限場加法器

40 有限場乘法器

42 有限場乘法器

44 有限場乘法器

46 有限場加法器

50 有限場乘法器

51 暫存器

52 有限場乘法器

53 暫存器

54 有限場乘法器

56 有限場加法器

60 有限場乘法器

61 暫存器

62 有限場乘法器

63 暫存器

64 有限場乘法器

66 有限場加法器

70 有限場乘法器

71 暫存器

圖式簡單說明

72 有限場乘法器

74 有限場加法器

六、申請專利範圍

1. 一種在解碼錯誤訂正碼時用以解答鍵方程式多項式之裝置，尤其是一種可以應用在BCH以及Reed- Solomon (RS)解碼器的所謂無反轉打散架構裝置，其主要之組成架構係包括下列幾個部分:

(a)徵兆計算器，接收字元碼並輸出徵兆多項式到鍵方程式解答器；

(b)鍵方程式解答器，接收徵兆多項式以產生錯誤定位多項式及錯誤計算多項式；

(c)Chein Search，接收錯誤定位多項式，並將結果輸入到錯誤值計算器並輸出錯誤區域；

(d)錯誤值計算器，接收由鍵方程式解答器及Chein Search的訊號並輸出錯誤值。

2. 如申請專利範圍第1項所述之裝置，其中所述之架構是使用在BCH和Reed-Solomon(RS)解碼器。

3. 如申請專利範圍第1項所述之裝置，其中所述之架構是以無反轉打散架構之方法應用於BCH和Reed-Solomon解碼器。

4. 如申請專利範圍第1項所述之裝置，其中所述之架構除了可以除去錯誤的訊號之外也可用於抹除符號。

5. 如申請專利範圍第1項所述之裝置，其中所述之架構是

六、申請專利範圍

用於無反轉Euclidean演算法。

6. 如申請專利範圍第1項所述之裝置，其中所述之架構是以移除會拖累整體有限場反轉器來完成。

7. 如申請專利範圍第1項所述之裝置，其中所述之架構可將原來2t個步驟更改為只需t個步驟就能完成。

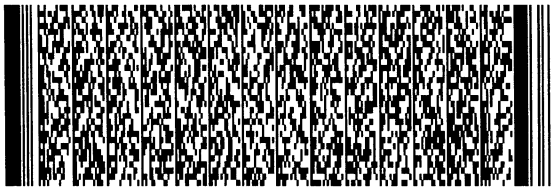
8. 如申請專利範圍第1項所述之裝置，其中所述之架構可使用打散的技巧大幅的將有限場乘法器由4t-6t個減少為僅僅3個。

9. 如申請專利範圍第1項所述之裝置，其中所述之架構可使用打散的技巧，其中僅僅使用4t+2 ρ+4個暫存器。

10. 如申請專利範圍第1項所述之裝置，其中所述之架構可使用打散的技巧，無須使用FFI。

11. 如申請專利範圍第1項所述之裝置，其中所述之架構可使用以求Forney徵兆多項式。

12. 如申請專利範圍第1項所述之裝置，其中所述之架構是使用於通訊及儲存系統上。



六、申請專利範圍

13. 一種在解碼錯誤訂正碼時用以解答鍵方程式多項式之方法，尤其是一種可以應用在BCH以及Reed- Solomon (RS)解碼器的所謂無反轉打散架構之方法，其主要包括下列幾個步驟:

(a)接收字元碼並計算徵兆；

(b)產生錯誤定位多項式及錯誤計算多項式；

(c)尋找錯誤區域；

(d)計算錯誤值。

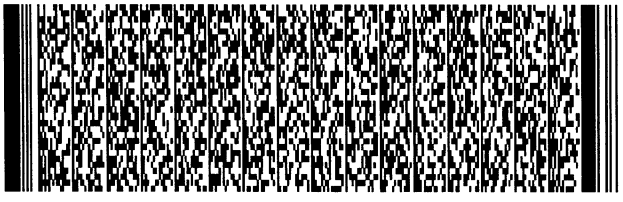
14. 如申請專利範圍第13項所述之方法，其中所述之架構是使用在BCH和Reed-Solomon解碼器。

15. 如申請專利範圍第13項所述之方法，其中所述之架構是以無反轉打散架構之方法使用於BCH和Reed-Solomon解碼器。

16. 如申請專利範圍第13項所述之方法，其中所述之架構除了可以除去錯誤的訊號之外也可用於抹除符號。

17. 如申請專利範圍第13項所述之方法，其中所述之架構適用於無反轉Euclidean演算法。

18. 如申請專利範圍第13項所述之方法，其中所述之架構是以移除會拖累整體有限場反轉器來完成。



六、申請專利範圍

19. 如申請專利範圍第13項所述之方法，其中所述之架構可將原來2t個步驟更改為只需t個步驟就能完成。

20. 如申請專利範圍第13項所述之方法，其中所述之架構可使用打散的技巧大幅的將有限場乘法器由4t-6t個減少為僅僅3個。

21. 如申請專利範圍第13項所述之方法，其中所述之架構可使用打散的技巧，其中僅僅使用4t+2 ρ+4個暫存器。

22. 如申請專利範圍第13項所述之方法，其中所述之架構可使用打散的技巧，無須使用FFI。

23. 如申請專利範圍第13項所述之方法，其中所述之架構可使用以求Forney徵兆多項式。

24. 如申請專利範圍第13項所述之方法，其中所述之架構是使用於通訊及儲存系統上。

25. 一種在解碼錯誤訂正碼時用以解答鍵方程式多項式之方法，尤其是一種可以應用在BCH以及Reed- Solomon (RS)解碼器的所謂無反轉打散架構之方法，其過程包括下列幾個改善之步驟：

六、申請專利範圍

(a)改善Euclidean演算法的速度；

(b)修飾解碼步驟以減少半數之解碼結果；

(c)將錯誤定位多項式和錯誤評價多項式之計算結合。

26.如申請專利範圍第25項所述之方法，其中所述之Euclidean演算法是和時間共享一個有限場乘法器(FFMs)。

27.如申請專利範圍第25項所述之方法，其中所述之方法可降低硬體使用面積。

28.如申請專利範圍第25項所述之方法，其中所述之改善Euclidean演算法的速度是以移去有限場反轉器之限制的無反轉之Euclidean演算法來完成。

29.如申請專利範圍第25項所述之方法，其中所述之無反轉Euclidean演算法是使角度重疊的數目降低到t來完成。

30.如申請專利範圍第28項所述之方法，其中所述之無反轉Euclidean演算法是使錯誤定位多項式角度由$\rho+1$增加為$\rho+t$及錯誤計算多項式。

31.如申請專利範圍第28項所述之方法，其中所述之無反

六、申請專利範圍

轉Educlidean演算法其每次之重複小於t。

32. 一種在解碼錯誤訂正碼時用以解答鍵方程式多項式之方法，尤其是一種可以應用在BCH以及Reed- Solomon (RS)解碼器的所謂無反轉打散架構之方法，其方法包含下列：

(a)每一個分割是被限制到至少一度；

(b)將錯誤定位多項式和錯誤評價多項式計算使用之硬體結合;

(c)將FFMs之數目降低為3個。

33. 如申請專利範圍第32項所述之方法，將使Euclidean Algorithm變慢，但是不會衝擊整體之解碼速度。

34. 如申請專利範圍第32項所述之方法，其中所述BCH以及Reed- Solomon (RS)解碼器，其合成碼在影音光碟中(DVD)使用RS合成碼在行方向(182,172)及列方向(208,192)。

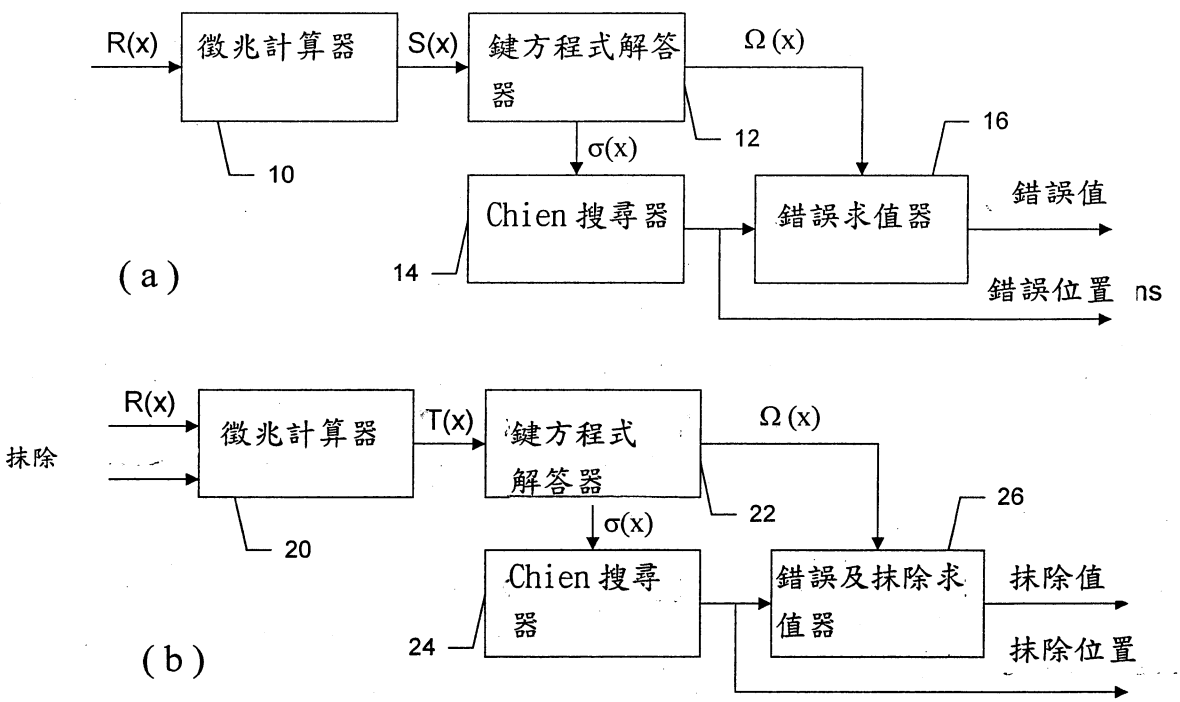
35. 如申請專利範圍第32項所述之方法，其中所述BCH以及Reed- Solomon (RS)解碼器，其合成碼在數位電視廣播系統使用(204,188)RS碼。

36. 如申請專利範圍第32項所述之方法，其中所述BCH以及

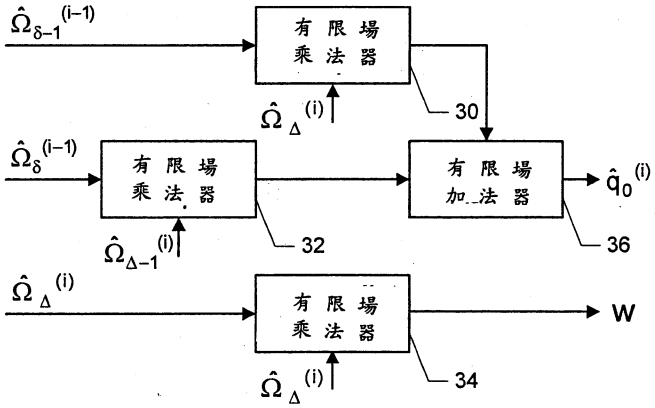
六、申請專利範圍

Reed- Solomon (RS)解碼器，其合成碼在一般CD-ROM使用小數目的RS碼包括(32,28)(28,24)。

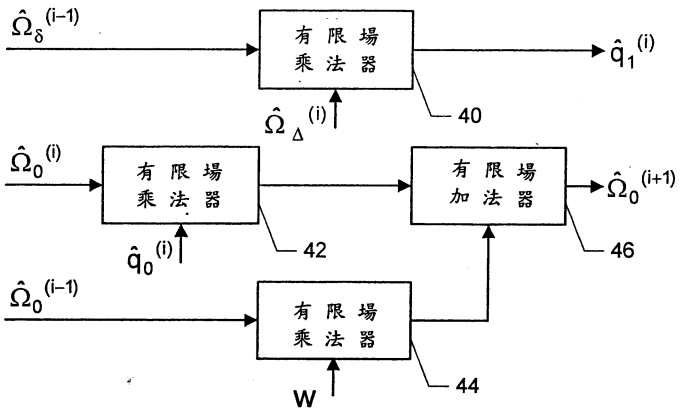
37. 如申請專利範圍第32項所述之方法，其中所述BCH以及Reed- Solomon (RS)解碼器，其合成碼在無線通訊方面，尤其AMPS電話系統使用(40,28)和(48,36)雙位元的BCH碼，其為(63,51)的短碼。



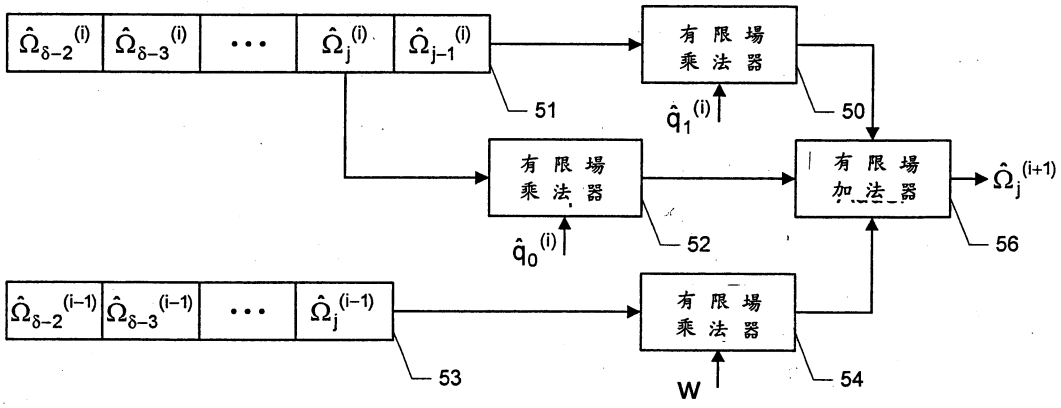
第 1 圖



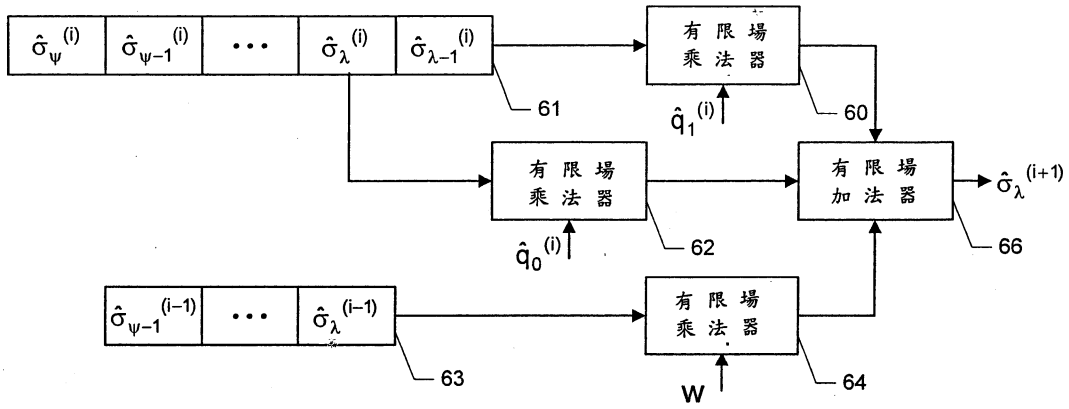
第 2A 圖


$\hat{\Omega}_\delta^{(i-1)}$
有限場乘法器
$\hat{q}_1^{(i)}$
$\hat{\Omega}_\Delta^{(i)}$
40
$\hat{\Omega}_0^{(i)}$
有限場乘法器
有限場加法器
$\hat{\Omega}_0^{(i+1)}$
$\hat{q}_0^{(i)}$
42
46
$\hat{\Omega}_0^{(i-1)}$
有限場乘法器
44
W

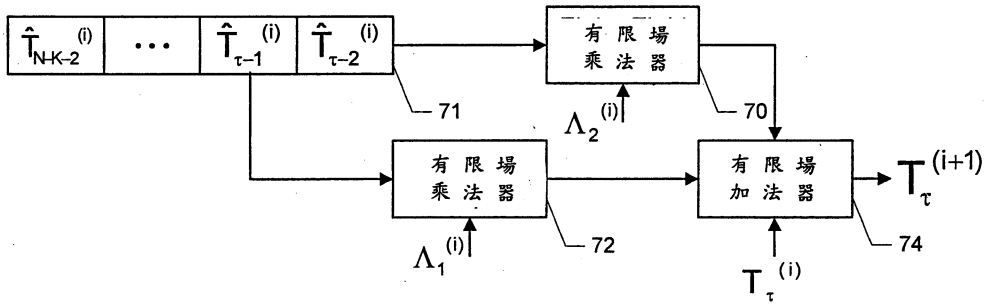
第 2B 圖


$\hat{\Omega}_{\delta-2}^{(i)}$ $\hat{\Omega}_{\delta-3}^{(i)}$ ... $\hat{\Omega}_j^{(i)}$ $\hat{\Omega}_{j-1}^{(i)}$
51
有限場乘法器
$\hat{q}_1^{(i)}$
50
有限場乘法器
有限場加法器
$\hat{\Omega}_j^{(i+1)}$
$\hat{q}_0^{(i)}$
52
56
$\hat{\Omega}_{\delta-2}^{(i-1)}$ $\hat{\Omega}_{\delta-3}^{(i-1)}$ ... $\hat{\Omega}_j^{(i-1)}$
有限場乘法器
53
W
54

第 2C 圖


$\hat{\sigma}_{\psi}^{(i)}$
$\hat{\sigma}_{\psi-1}^{(i)}$
• • •
$\hat{\sigma}_{\lambda}^{(i)}$
$\hat{\sigma}_{\lambda-1}^{(i)}$
61
有限場乘法器
$\hat{q}_1^{(i)}$
60
有限場乘法器
有限場加法器
$\hat{\sigma}_{\lambda}^{(i+1)}$
$\hat{q}_0^{(i)}$
62
66
$\hat{\sigma}_{\psi-1}^{(i-1)}$
• • •
$\hat{\sigma}_{\lambda}^{(i-1)}$
63
有限場乘法器
W
64

第 2D 圖


$\hat{T}_{N-K-2}^{(i)}$
• • •
$\hat{T}_{\tau-1}^{(i)}$
$\hat{T}_{\tau-2}^{(i)}$
71
有限場乘法器
$\Lambda_2^{(i)}$
70
有限場乘法器
有限場加法器
$T_{\tau}^{(i+1)}$
$\Lambda_1^{(i)}$
72
74
$T_{\tau}^{(i)}$

第 2E 圖