(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201413727 A
(43)公開日： 中華民國 103 (2014) 年 04 月 01 日

(21)申請案號：101135990 (22)申請日： 中華民國 101 (2012) 年 09 月 28 日
(51)Int. Cl.： *G11C29/42 (2006.01)* *H03M13/05 (2006.01)*
(71)申請人：國立交通大學 (中華民國) NATIONAL CHIAO TUNG UNIVERSITY (TW)
新竹市大學路 1001 號
(72)發明人：朱家慶 CHU, CHIACHING (TW) ；林義閔 LIN, YIMIN (TW) ；楊其衡 YANG, CHIHENG (TW) ；張錫嘉 CHANG, HSIECHIA (TW)
(74)代理人：蔡坤財；李世章
申請實體審查：有 申請專利範圍項數：15 項 圖式數：6 共 36 頁

(54)名稱

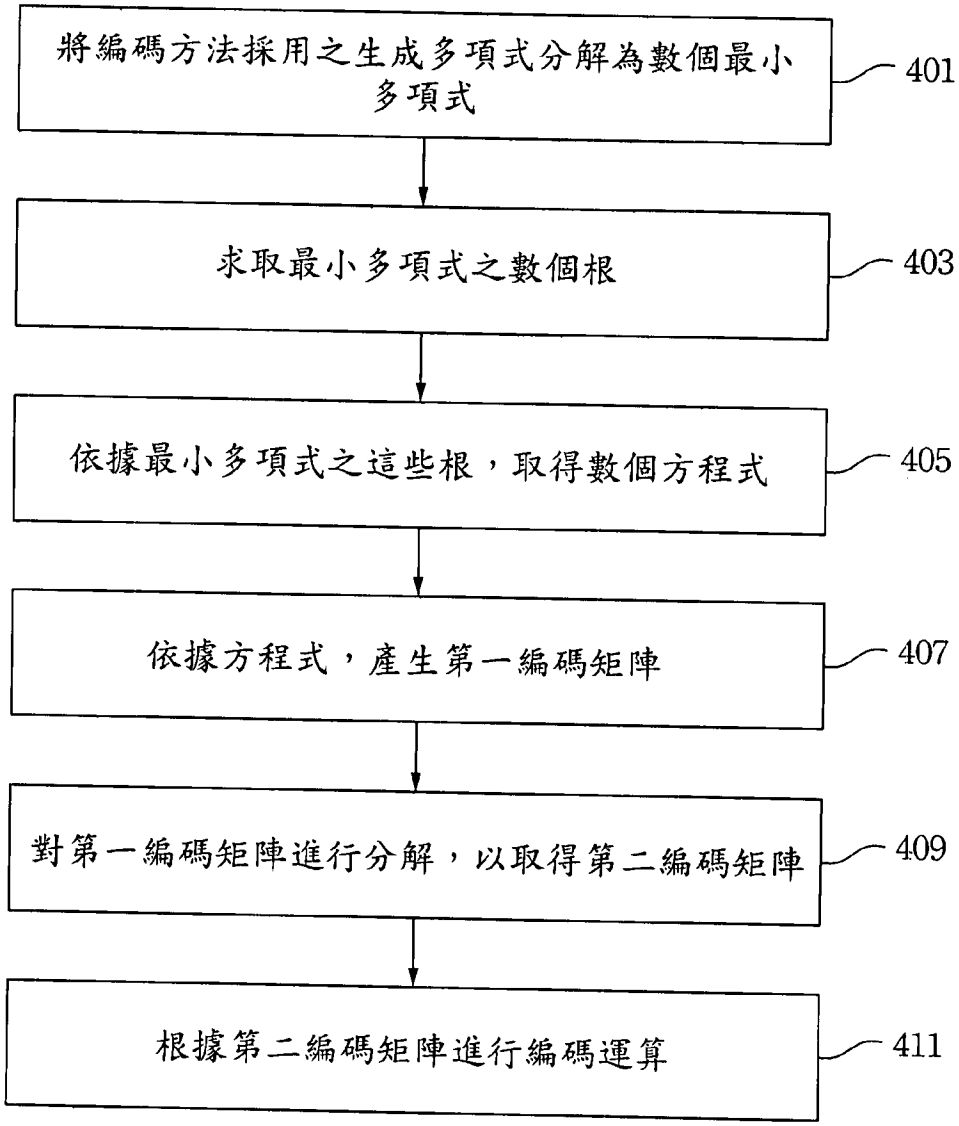
記憶體系統之全套平行編碼方法與全套平行解碼方法

FULLY PARALLEL ENCODING METHOD AND FULLY PARALLEL DECODING METHOD OF MEMORY SYSTEM

(57)摘要

一種記憶體系統之全套平行編解碼方法以及記憶體系統，編碼方法利用組成生成多項式的各最小多項式，求取最小多項式之複數根並利用其產生第一編碼矩陣，其中，解碼器之校驗矩陣與第一編碼矩陣分解後所得到的第二編碼矩陣具有共同元素，使編解碼器具有良好硬體共用效益。解碼方法則新定義一錯誤位置多項式，並採用平方器之矩陣運算方式對方程式進行分項合併，減少全套平行所需龐大硬體需求。

401~411：步驟



第 4 圖

201413727

發明專利說明書

（本說明書格式、順序，請勿任意更動，※記號部分請勿填寫）

※申請案號：101135990

※申請日：101.9.28　※IPC 分類：G11C 29/42 (2006.01)
H03M 13/25 (2006.01)

一、發明名稱：（中文/英文）

(中文)記憶體系統之全套平行編碼方法與全套平行解碼方法

(英文) Fully Parallel Encoding Method and Fully Parallel Decoding Method of Memory System

二、中文發明摘要：

一種記憶體系統之全套平行編解碼方法以及記憶體系統，編碼方法利用組成生成多項式的各最小多項式，求取最小多項式之複數根並利用其產生第一編碼矩陣，其中，解碼器之校驗矩陣與第一編碼矩陣分解後所得到的第二編碼矩陣具有共同元素，使編解碼器具有良好硬體共用效益。解碼方法則新定義一錯誤位置多項式，並採用平方器之矩陣運算方式對方程式進行分項合併，減少全套平行所需龐大硬體需求。

三、英文發明摘要：

A memory system, a fully parallel encoding method, and a fully parallel decoding method are disclosed. The encoding method utilizes a plurality of minimal polynomials that constitute a generator polynomial and derives a plurality

of roots from the minimal polynomials. A first encoding matrix derived according to the roots of the minimal polynomials is subsequently decomposed to derive a second encoding matrix, in which partial elements of the second encoding matrix are the same with those of a parity check matrix of the decoder, such that the encoder and the decoder can efficiently share the same hardware. In addition, the decoding method defines a new error locator polynomial and utilizes a cubic matrix operation to respectively combine the equations, which reduces the hardware required by the fully parallel architecture.

四、指定代表圖：

(一)本案指定代表圖為：第（4）圖。

(二)本代表圖之元件符號簡單說明：

401~411：步驟

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

無。

六、發明說明：

【發明所屬之技術領域】

本發明是有關於一種記憶體，且特別是有關於一種記憶體之編解碼器。

【先前技術】

由於近年消費性電子等應用對於反或閘型(NOR)快閃記憶體性能之需求逐漸提昇與製程微縮對 NOR 快閃記憶體可靠度造成的嚴苛挑戰，傳統可更正單一位元之錯誤更正碼逐漸不敷使用。NOR 快閃記憶體之主要功能在於儲存系統的控制程式，其應用的涵蓋領域除 PC 產品外，在廣大的消費性電子產品市場中更是處處充斥著它的身影，例如：數位相機、DVD 錄放影機，MP3 音樂播放機、印表機、機上盒(DVB-S,-T)及車用電子等，不勝枚舉。

NOR 快閃記憶體的特點在於晶片內執行(eXecute In Place;XIP)，這使應用程序可以直接在快閃記憶體內運行而不必把代碼讀到系統隨機存取記憶體(Random Access Memory; RAM)中，因此 NOR 快閃記憶體的傳輸效率很高，不需要冗長的開機等待時間。

早期 NOR 快閃記憶體儲存量較小、寫入速度慢，但因隨機存取速度快，主要鎖定在手機及數位相機(Digital Still Camera; DSC)、手持式錄放影機(DV)等消費性電子產品的嵌入式記憶體應用。近年來車用電子、消費性電子、家用遊戲主機(如：任天堂 Wii、XBOX、Play Station 3 等)、智慧電錶等需求當道，NOR 快閃記憶體在科技產業中所扮

演的角色越來越重要，特別是應用於車用市場如數位儀表板、衛星導航系統、車載娛樂系統、數位電視、機上盒等。

記憶體元件的位元錯誤率(Bit Error Rate; BER)預估在45奈米製程下將大於10^{-6}，但如果要將NOR快閃記憶體應用在產品上，其位元錯誤率則必須要小於10^{-12}才能提供足夠的可靠度。所以必須要應用錯誤更正碼(Error Correcting Codes; ECC)使快閃記憶體元件符合產品所需的可靠度水準。以往單位元錯誤更正碼(Single-Error-Correcting Code; SEC)，例如漢明碼(Hamming code)，可以偵測並更正一位元錯誤。

然而，隨著奈米製程的進步發展，漢明碼的錯誤更正能力已經不敷使用，漢明碼的編解碼速度、電路複雜度都已經無法符合現代的需要；即便捨棄漢明碼改採可更正兩位元錯誤的Bose-Chaudhuri-Hochquenghem (BCH)碼，其電路結構仍然過於複雜，解碼速度也依然趕不上現今電子裝置的需求。

因此需要一種新的全套平行編解碼方法，能夠加快編解碼速度，同時降低電路的複雜度。

【發明內容】

因此，本發明一方面提供一種記憶體系統之全套平行編碼方法，能夠加快編解碼速度，同時減少所需要的硬體電路，降低電路複雜度。

依據本發明一實施例，記憶體系統之全套平行編碼方法先將全套平行編碼方法採用之生成多項式分解為複數個

最小多項式，求取最小多項式之複數個根；然後依據最小多項式之根，取得複數個方程式，並依據方程式，產生一第一編碼矩陣；接著則對第一編碼矩陣進行分解，以取得一第二編碼矩陣，其中第二編碼矩陣為一解碼器之一校驗矩陣之一部分矩陣；之後再依據第二編碼矩陣進行編碼運算。

本發明之另一方面提供一種記憶體系統之全套平行解碼方法，相較於傳統方法，能夠以使用較小面積之電路達到相同的錯誤更正能力。

依據本發明另一實施例，記憶體系統之全套平行解碼方法先依據一校驗矩陣，計算得出複數個徵兆值，其中此校驗矩陣與一編碼器之一第二編碼矩陣具有部分共同元素；再依據這些徵兆值，解出一錯誤位置多項式，並將複數個基本元素帶入此錯誤位置多項式，當一碼字(Codeword)發生錯誤，則取得此錯誤位置多項式之根，進而得到相對應的錯誤位置。

本發明之又一態樣提供一種記憶體系統，能夠在維持相同的更正能力之下，加快編解碼速度，同時降低電路的面積。

根據本發明之又一實施例，記憶體系統含有一編碼器、一記憶體元件、一寫入電路、一讀取電路，以及一解碼器。編碼器利用一第二編碼矩陣對一輸入資料進行編碼，以產生至少一組碼字，此碼字(Codeword)內含校驗碼(Parity bits)；記憶體元件依據碼字以及校驗碼，進行存儲運作；寫入電路電性連接編碼器，以將碼字以及校驗碼傳

送至記憶體元件；讀取電路自記憶體元件讀出資料；解碼器利用一校驗矩陣解碼自記憶體讀出資料，其中，第二編碼矩陣為校驗矩陣之一部分矩陣。

以上實施例的記憶體系統以及其全套平行(Fully-Parallel)編解碼方法，採用了全套平行式的設計以提升編碼速度，降低解碼延遲，因此具備高速編解碼能力，能夠加速記憶體的讀寫速度；另一方面，利用改良的演算法，將編碼和解碼的硬體做有效率的共用，省去編碼部分的硬體，減少解碼的最長延遲路徑，降低記憶體系統整體面積，有效減少硬體花費。

【實施方式】

隨著記憶體技術的進步以及普羅大眾對於視聽娛樂的需求，電視機上盒(Set Top Box；STB)中的電子節目指南(Electronic Programming Guides；EPG)對於 NOR 快閃記憶體之要求也逐漸攀升。因為 EPG 的功能越來越多元化與複雜化，如果 EPG 內容存於動態隨機存取記憶體(Dynamic Random Access Memory; DRAM)等揮發性記憶體中，當發生電力中斷時，需花費數小時才能完全恢復，若存在 NOR 快閃記憶體則不同，其非揮發性之特性可使內容完整保存不受電力存在與否影響，因此 EPG 的內容需要藉由高密度 NOR 快閃記憶體來儲存。

目前許多汽車製造商在新一代電子儀表板和資訊娛樂系統也將採用高效能記憶體來減少開機時間，NOR 快閃記憶體晶片正符合此需求。它能夠提供快速的開機時間和及

時回饋並具備高度可靠性，這些都是汽車製造商為保護駕駛人而打造安全駕駛環境所需要之不可或缺要素，並且內建於汽車和消費性電子產品內的語音辨識系統等都需要大量相關技術的記憶體作為後盾，其目標都在於必須快速且準確地處理相關指令以縮短搜尋及反應時間和使語音搜尋的準確度提升。而這些高達 256Mb、512Mb 甚至 1Gb 的高容量 NOR 快閃記憶體晶片在這些領域上顯得格外重要。

在引進雙位元錯誤更正碼(Double-Error-Correcting code; DEC)，例如 Bose-Chaudhuri-Hocquenghem (BCH)碼後，位元錯誤率降低至大約 10^{-13}。由此可見，引進 DEC BCH 碼的 NOR 快閃記憶體能達到應用於產品上的可靠度要求。

在碼字長度及更正能力皆相同的條件之下，本發明的一實施例針對全套平行架構的 BCH 編解碼器，化簡硬體複雜度。此外，本發明的一實施例採用新定義的錯誤位置多項式(Error Locator Polynomial)，相對於 Peterson's algorithm 推導出來的錯誤位置多項式有更短的解碼時間。相較於傳統的解碼器，本發明一實施例當中的編解碼器在面積及成本上有非常大的優勢。

1. 有限場的基本運算

◆常數乘法器

在有限場 $GF(2^m)$ 下，每個元素都可以表示成 $\lambda = \lambda_0 + \lambda_1\alpha + \cdots + \lambda_{m-1}\alpha^{m-1}$，其中，$\alpha$ 是 $GF(2^m)$ 下的基本元素(Primitive Element)，而 λ_i 是二位元座標，$\{\alpha^0, \alpha^1, \cdots, \alpha^{m-1}\}$ 可視為 m 個座標軸。因此，當 λ 與一個已知常數相乘時可以表示成

$$\begin{aligned}\alpha^j \times \lambda &= \alpha^j \times (\lambda_0 + \lambda_1\alpha + \cdots + \lambda_{m-1}\alpha^{m-1}) \\ &= \lambda_0\alpha^j + \lambda_1\alpha^{j+1} + \cdots + \lambda_{m-1}\alpha^{j+m-1} \\ &= \begin{bmatrix} \alpha_0^j & \alpha_0^{j+1} & \cdots & \alpha_0^{j+m-1} \\ \alpha_1^j & \alpha_1^{j+1} & \cdots & \alpha_1^{j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^j & \alpha_{m-1}^j & \cdots & \alpha_{m-1}^{j+m-1} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{bmatrix} \\ &= \mathbf{C}^j\lambda\end{aligned}$$

其中，α^p的第l個座標以二位元的元素α_l^p來表示，而矩陣$\mathbf{C}^j$則被稱為是一個被乘數是α^j的常數乘法器。

◆平方器

在有限場$GF(2^m)$下，每個元素λ都可以使用多項式來表示。此多項式的係數是二位元，所以計算任意一個元素的平方可以表示成

$$\begin{aligned}\lambda^2 &= (\lambda_0 + \lambda_1\alpha + \cdots + \lambda_{m-1}\alpha^{m-1})^2 \\ &= \lambda_0 + \lambda_1\alpha^2 + \cdots + \lambda_{m-1}\alpha^{2(m-1)} \\ &= (\lambda_0 + \lambda_1 x + \cdots + \lambda_{m-1}x^{m-1})\Big|_{x=\alpha^2} \\ &= \begin{bmatrix} \alpha_0^0 & \alpha_0^2 & \cdots & \alpha_0^{2(m-1)} \\ \alpha_1^0 & \alpha_1^2 & \cdots & \alpha_1^{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^0 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{2(m-1)} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{bmatrix} \\ &= \mathbf{Q}\lambda\end{aligned}$$

其中，平方器就是依據矩陣Q來實現。由平方器的性質，可以延伸討論出在有限場$GF(2^m)$下，任意一個多項式$f(x)$，只要其係數皆為二位元，便有$f(x^2) = f^2(x)$的性質。

◆立方器

傳統上，立方器 101 是由一個平方器 103 和一個乘法器 105 組成，如第 1 圖所示。這種架構的最長延遲路徑(critical path)是$(\log_2 m + \log_2(2m-2)+1)\cdot\tau_{XOR} + \tau_{AND}$。而$\tau_{XOR}$及$\tau_{AND}$分別是

XOR 閘和 AND 閘的延遲時間。為了加速立方器 101 的處理時間，將任意一個 $GF(2^m)$ 的元素的三次方，λ^3，以下面的方式展開：

$$\begin{aligned}\lambda^3 &= (\lambda_0+\lambda_1\alpha+\cdots+\lambda_{m-1}\alpha^{m-1})^3 \\ &= \lambda_0+\lambda_0\lambda_1\alpha+(\lambda_0\lambda_1+\lambda_0\lambda_2)\alpha^2+\cdots+\lambda_{m-1}\alpha^{3(m-1)} \\ &= \begin{bmatrix} \alpha_0^0 & \alpha_0^1 & \alpha_0^2 & \cdots & \alpha_0^{3(m-1)} \\ \alpha_1^0 & \alpha_1^1 & \alpha_1^2 & \cdots & \alpha_1^{3(m-1)} \\ \alpha_2^0 & \alpha_2^1 & \alpha_2^2 & \cdots & \alpha_2^{3(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^0 & \alpha_{m-1}^1 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{3(m-1)} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_0\lambda_1 \\ \lambda_0\lambda_1+\lambda_0\lambda_2 \\ \vdots \\ \lambda_{m-1} \end{bmatrix}\end{aligned}$$

依據上面的矩陣來實現立方器，可以使得最長延遲路徑降至 $(\log_2(3m-3)+1)\cdot\tau_{XOR}+\tau_{AND}$。以 $m=9$ 為例，本發明一實施例的立方器 101 可以減少三個 XOR 閘的延遲時間，本發明一實施例的有限場的立方器示意圖則繪示於第 1 圖當中。

2. 編解碼器共享硬體資源

本發明一實施例之編碼器與解碼器係應用於 Bose-Chaudhuri-Hocquenghem (BCH)碼，編碼器係採用全套平行架構，編碼器與解碼器的徵兆值計算器可以共享一套硬體資源，此編碼器使用系統化編碼的方式(Systematic Encoding)，$u(x)x^{n-k}=q(x)g(x)+p(x)$，其中，$u(x)$ 是訊息多項式(Message Polynomial)，$p(x)$ 是校驗多項式(Parity Check Polynomial)，而生成多項式(Generator polynomial)，$g(x)$，可以表示成

$$\begin{aligned}g(x) &= LCM\{M_1(x),M_2(x),\cdots,M_{2t}(x)\} \\ &= M_1(x)\times M_3(x)\times\cdots\times M_{2t-1}(x) \\ &= g_{n-k}x^{n-k}+\cdots+g_2x^2+g_1x+g_0\end{aligned}$$

其中，$M_i(x)$ 是第 i 最小多項式(i-th minimal polynomial)，從定義上，可以得知 α^i 及其共軛根是第 i 最小多項式的根。所以將生成多項式的 $n-k$ 個根代入系統化的編碼等式，可以得到 $n-k$ 個方程式，

$P(X)=U(X)X^{n-k}|_{x=\alpha^i\text{及其共軛根}}$，其中，$i=1, 3, \ldots 2t-1$。將這 $n-k$ 個方程式用矩陣的方式表示，

$$\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{n-k-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2\times(n-k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{(2t-1)\times 2^{l_{2t-1}-1}} & \cdots & \alpha^{(2t-1)\times 2^{l_{2t-1}-1}\times(n-k-1)} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-k-1} \end{bmatrix} =$$

$$\begin{bmatrix} \alpha^{n-k} & 0 & \cdots & 0 \\ 0 & \alpha^{2(n-k)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha^{(2t-1)\times 2^{l_{2t-1}-1}\times(n-k)} \end{bmatrix} \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2\times(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{(2t-1)\times 2^{l_{2t-1}-1}} & \cdots & \alpha^{(2t-1)\times 2^{l_{2t-1}-1}\times(k-1)} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{k-1} \end{bmatrix}$$

其中，l_i 定義為 α^i 共軛根的個數。為了撰寫上的方便，將上面矩陣的方程式改寫成 $\mathbf{BP}=\mathbf{AH}_{EN}\mathbf{U}$。

因為目的是要讓編碼器可以和徵兆值計算器共享一套硬體，也就是必須找出一種編碼方法，可以讓編碼矩陣與校驗矩陣有共通性。根據定義，$g(x)=M_1(x)\times M_3(x)\times\cdots\times M_{2t-1}(x)$，可以先根據最小多項式將生成多項式的根分成 t 個部分，接著利用本章節的第一部分所提到的平方器觀念，

$$\begin{aligned} \lambda^{2^j} &= (((\lambda_0+\lambda_1\alpha+\cdots+\lambda_{m-1}\alpha^{m-1})^2)^2\cdots)^2 \\ &= \lambda_0+\lambda_1\alpha^{2^j}+\cdots+\lambda_{m-1}\alpha^{2^j(m-1)} \\ &= (\lambda_0+\lambda_1 x+\cdots+\lambda_{m-1}x^{m-1})\big|_{x=\alpha^{2^j}} \\ &= \begin{bmatrix} \alpha_0^0 & \alpha_0^{2^j} & \cdots & \alpha_0^{2^j(m-1)} \\ \alpha_1^0 & \alpha_1^{2^j} & \cdots & \alpha_1^{2^j(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^0 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{2^j(m-1)} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{bmatrix} \\ &= \mathbf{Q}(\cdots(\mathbf{Q}(\mathbf{Q}\lambda)) \\ &= \mathbf{Q}^j\lambda \end{aligned}$$

將矩陣 $\mathbf{H}_{EN}$ 作分解，$\mathbf{H}_{EN} = \mathbf{Q}_{EN} \cdot \tilde{\mathbf{H}}_{EN}$，其中，$\mathbf{Q}_{EN}$ 和 $\tilde{\mathbf{H}}_{EN}$ 分別是

$$\mathbf{Q}_{EN} = \begin{bmatrix} \mathbf{Q}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{Q}_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{Q}_{2t-1} \end{bmatrix}, \quad \mathbf{Q}_i = \begin{bmatrix} \mathbf{Q}^0 \\ \mathbf{Q}^1 \\ \vdots \\ \mathbf{Q}^{l_i} \end{bmatrix}, \quad \mathbf{Q}^0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$\tilde{\mathbf{H}}_{EN} = \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t-1} & \cdots & \alpha^{(2t-1)\times(k-1)} \end{bmatrix}$$

從上面的式子，可以很輕易地發現矩陣 $\tilde{\mathbf{H}}_{EN}$ 是校驗矩陣 $\tilde{\mathbf{H}}_{SYN}$ 的部分矩陣(Sub-Matrix)，所以在用硬體實現編碼器時，$\tilde{\mathbf{H}}_{EN}$ 這部分就不需要額外的硬體資源，只需要借用解碼器中的徵兆值計算器即可。

因此，$\mathbf{BP} = \mathbf{AH}_{EN}\mathbf{U}$ 可以改寫成 $\mathbf{BP} = \mathbf{AQ}_{EN}\tilde{\mathbf{H}}_{EN}\mathbf{U}$，又因為矩陣 B 是非奇異性(Non-Singular)的矩陣，會存在有反矩陣，所以校驗位元便可以從 $\mathbf{P} = \mathbf{B}^{-1}\mathbf{AQ}_{EN}\tilde{\mathbf{H}}_{EN}\mathbf{U} = \mathbf{E}\tilde{\mathbf{H}}_{EN}\mathbf{U}$ 得到。其中，矩陣 E 定義為 $\mathbf{B}^{-1}\mathbf{AQ}_{EN}$ 的矩陣相乘。值得一提的是當決定訊息長度(Message Length)和錯誤更正能力(Error Correcting Capability)後，矩陣 E 與矩陣 $\tilde{\mathbf{H}}_{EN}$ 便可以從 C/C++或是 Matlab 等電腦程式事先計算求得，不需要為此再付出額外的硬體資源去做運算。

矩陣的運算就是一連串的相乘與相加。從本章節的第一部分所提到的常數乘法器觀念，當乘上矩陣 E 每一個元素時，相當於是乘上一個 $m \times m$ 的常數乘法矩陣後得到一個 $GF(2^m)$ 下 m-bit 的元素。接著在與相對應的相乘結果相加後得到校驗位元。但校驗位元是一個位元的，所以可以把矩陣 E 的常數乘法矩陣簡化為

$$\alpha^j \times \lambda = \alpha^j \times (\lambda_0 + \lambda_1\alpha + \cdots + \lambda_{m-1}\alpha^{m-1})$$

$$= \begin{bmatrix} \alpha_0^j & \alpha_0^{j+1} & \cdots & \alpha_0^{j+m-1} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{bmatrix}$$

$$= \mathbf{C}_E^j \lambda$$

其中，C_E^j是一個存在於矩陣 E 中，被乘數是α^j的常數乘法器。

原來的編碼矩陣拆解成兩個矩陣。其中一個是可以與徵兆值計算器共用硬體資源的。固有的徵兆值計算器，只需要在額外多一點點硬體便可以完成編碼。下面的表一呈現的是估計的 XOR gate 數目，並與傳統架構作比較，其中 n 代表碼字(Codeword)長度，k 代表訊息(Message)長度，t 代表錯誤更正能力，mt 則為編碼矩陣當中的參數。表二是使用 UMC 90nm 1P9M CMOS 製程，編碼器與徵兆值計算器共用硬體資源後與傳統架構的實驗數據與比較結果。由表上數據可以發現，本發明一實施例提出之硬體架構應用在訊息位元為 256 位元或 512 位元；更錯能力為 2 位元或 3 位元的情況下，皆可以比傳統作法省下至少 40%之硬體複雜度。這代表本發明一實施例提出的硬體架構，除了能夠滿足目前世代之 NOR 快閃記憶體對於可靠度與硬體複雜度的需求之外，在未來即使可更正 2 位元之錯誤更正碼逐漸不敷使用，而需要可更正 3 位元或 3 位元以上之錯誤更正碼的時候，本發明一實施例的方法依然是一個具有優勢的低複雜度硬體架構。

表一

(N, k; t)	本發明實施例	傳統架構	面積節省(%)
	mtxmt + mtxn	kxmt + mtxn	
(274,256;2)	$324\times\rho_{Eavg}+4932\times\rho_{Savg}$	$4608\times\rho_{Eavg}+4932\times\rho_{Savg}$	45%
(283,256;3)	$729\times\rho_{Eavg}+7398\times\rho_{Savg}$	$6912\times\rho_{Eavg}+7398\times\rho_{Savg}$	43%
(532,512;2)	$400\times\rho_{Eavg}+10640\times\rho_{Savg}$	$10240\times\rho_{Eavg}+10640\times\rho_{Savg}$	47%
(542,512;3)	$900\times\rho_{Eavg}+16260\times\rho_{Savg}$	$15360\times\rho_{Eavg}+16260\times\rho_{Savg}$	46%

表二

(n, k; t)	本發明實施例	傳統	省下的面積
(274, 256; 2)	2671	4621	42%
(283, 256; 3)	4100	6904	41%
(532, 512; 2)	5105	9234	45%
(542, 512; 3)	7745	13707	43%

上表一係以矩陣大小去估計硬體所需要的 XOR 閘數，ρ_{avg}是矩陣內"1"的密度，其中ρ_{Savg}是與徵兆值計算相關的矩陣平均密度，ρ_{Eavg}則是與編碼相關的矩陣平均密度，上表二則顯示編碼器與徵兆值計算器邏輯閘數(Gate count)比較表。以矩陣化簡的低複雜度 BCH 解碼器，計算

徵兆值$S_1,S_2,\cdots,S_{2t}$是 BCH 解碼器的第一個步驟。因為偶數項徵兆值是奇數項徵兆值的線性組合，所以只需計算 t 個徵兆值$S_1,S_3,\cdots,S_{2t-1}$。以更正能力為 2(Double Error Correcting)的 BCH 解碼器為例，其徵兆值計算如下:

$$\begin{bmatrix} S_1 \\ S_3 \end{bmatrix} = \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(k-1)} \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{k-1} \end{bmatrix} = \tilde{\mathbf{H}}_{SYN} \cdot \mathbf{R}$$

一旦得當徵兆值S_1,S_3，採用 Reversed Error Locator Polynomial 來進行後續處理。因為它相對於 Peterson's algorithm 推導出的錯誤位置多項式有較短的解碼時間。可以得到經過簡化、更正能力為 2 的 BCH 錯誤位置多項式為：

$$\sigma(x)=S_1X^2+S_1^2x+(S_1^3+S_3)=\sigma^{(1)}(x)+(S_1^3+S_3)$$

當錯誤位置多項式決定後，利用 Chien search 的方法，將$\alpha^0,\alpha^1,\cdots,\alpha^{n-1}$代入錯誤位置多項式去判斷是否為其根。如果$\alpha^i$是錯誤位置多項的根表示第 i 個位置是有錯的。從上面的式子可以看出，當在計算 Chien search 時，$S_1x^2+S_1^2x$是造成硬體複雜度高的主因。因此針對$S_1x^2+S_1^2x$的計算提出了兩個方法去減少複雜度。

實施例一

因為S_1是一個$GF(2^m)$下的元素，如果假設S_1是α^j，$S_1x^2+S_1^2x$可以改寫成$\alpha^jx^2+\alpha^{2j}x$。接著，利用常數乘法器及平方器的概念將$\alpha^jx^2+\alpha^{2j}x$作以下的推導：

$$\alpha^j x^2 + \alpha^{2j} x$$

$$= \begin{bmatrix} \alpha_0^j & \alpha_0^{j+1} & \cdots & \alpha_0^{j+m-1} \\ \alpha_1^j & \alpha_1^{j+1} & \cdots & \alpha_1^{j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^j & \alpha_{m-1}^{j+1} & \cdots & \alpha_{m-1}^{j+m-1} \end{bmatrix} \begin{bmatrix} x_0^2 \\ x_1^2 \\ \vdots \\ x_{m-1}^2 \end{bmatrix} + \begin{bmatrix} \alpha_0^{2j} & \alpha_0^{2j+1} & \cdots & \alpha_0^{2j+m-1} \\ \alpha_1^{2j} & \alpha_1^{2j+1} & \cdots & \alpha_1^{2j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^{2j} & \alpha_{m-1}^{2j+1} & \cdots & \alpha_{m-1}^{2j+m-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_0^j & \alpha_0^{j+1} & \cdots & \alpha_0^{j+m-1} \\ \alpha_1^j & \alpha_1^{j+1} & \cdots & \alpha_1^{j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^j & \alpha_{m-1}^{j+1} & \cdots & \alpha_{m-1}^{j+m-1} \end{bmatrix} \begin{bmatrix} 1 & \alpha_0^2 & \cdots & \alpha_0^{2(m-1)} \\ 0 & \alpha_1^2 & \cdots & \alpha_1^{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{2(m-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix} + \begin{bmatrix} \alpha_0^{2j} & \alpha_0^{2j+1} & \cdots & \alpha_0^{2j+m-1} \\ \alpha_1^{2j} & \alpha_1^{2j+1} & \cdots & \alpha_1^{2j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^{2j} & \alpha_{m-1}^{2j+1} & \cdots & \alpha_{m-1}^{2j+m-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_0^j & \alpha_0^{j+2} & \cdots & \alpha_0^{j+2(m-1)} \\ \alpha_1^j & \alpha_1^{j+2} & \cdots & \alpha_1^{j+2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^j & \alpha_{m-1}^{j+2} & \cdots & \alpha_{m-1}^{j+2(m-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix} + \begin{bmatrix} \alpha_0^{2j} & \alpha_0^{2j+1} & \cdots & \alpha_0^{2j+m-1} \\ \alpha_1^{2j} & \alpha_1^{2j+1} & \cdots & \alpha_1^{2j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^{2j} & \alpha_{m-1}^{2j+1} & \cdots & \alpha_{m-1}^{2j+m-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_0^j + \alpha_0^{2j} & \alpha_0^{j+2} + \alpha_0^{2j+1} & \cdots & \alpha_0^{j+2(m-1)} + \alpha_0^{2j+m-1} \\ \alpha_1^j + \alpha_1^{2j} & \alpha_1^{j+2} + \alpha_1^{2j+1} & \cdots & \alpha_1^{j+2(m-1)} + \alpha_1^{2j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^j + \alpha_{m-1}^{2j} & \alpha_{m-1}^{j+2} + \alpha_{m-1}^{2j+1} & \cdots & \alpha_{m-1}^{j+2(m-1)} + \alpha_{m-1}^{2j+m-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{C}^0\alpha^j + \mathbf{Q}\alpha^j & \mathbf{C}^2\alpha^j + \mathbf{C}^1\mathbf{Q}\alpha^j & \cdots & \mathbf{C}^{2(m-1)}\alpha^j + \mathbf{C}^{m-1}\mathbf{Q}\alpha^j \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{C}^0 + \mathbf{Q})\alpha^j & (\mathbf{C}^2 + \mathbf{C}^1\mathbf{Q})\alpha^j & \cdots & (\mathbf{C}^{2(m-1)} + \mathbf{C}^{m-1}\mathbf{Q})\alpha^j \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{T}_0 S_1 & \mathbf{T}_1 S_1 & \cdots & \mathbf{T}_{m-1} S_1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}$$

$$= \sigma^{(1)}(x)$$

錯誤位置多項式可以表示成 $\sigma(x) = \sigma^{(1)}(x) + S_1^3 + S_3$。當 m 被決定後，可以透過軟體事先運算，例如 C/C++或是 Matlab，來找到矩陣 $\mathbf{T}_0, \mathbf{T}_1, \cdots \mathbf{T}_{m-1}$，此外因為矩陣 $\mathbf{T}_0, \mathbf{T}_1, \cdots \mathbf{T}_{m-1}$ 的維度與常數乘法器一樣。因此在做硬體估計時，將它們視為一個常數

乘法器。所以傳統計算$\sigma^{(1)}(x)$時，需要2n個常數乘法器降到n+m個，其中，n是碼字的長度，有n個位元，m是GF參數$GF(2^m)$，一般來說$m \geqq \log_2 n$。

第2圖則繪示本發明此一實施例採用常數乘法器以及平方器的解碼器硬體架構示意圖。在此第2圖當中，徵兆值計算器201依據解碼器所接收到的碼字r_0、$r_{1...}r_{n-1}$，計算出徵兆值S_1、S_3，立方器203與加法器207對徵兆值S_1、S_3進行運算，以得到$S_3+S_1^3$。接著，電路205、乘法器209以及加法器207對徵兆值S_1以及矩陣T_0、$T_{1...}T_{m-1}$進行運算，得到輸出Output[0]、Output[1]...Output[n-1]，再據以得到錯誤位置多項式$\sigma(x)$。

實施例二

這個方法的主要概念是產生n個m×m矩陣來取代本來作Chien search所需要的2n個常數乘法器。當矩陣的輸入都相同時，可以在對錯誤位置多項式在做更進一步的最佳化。一樣先假設S_i是α^j，當Chien search時，α^i被代入$\sigma(x)$，可以寫成$\sigma(\alpha^i)=\sigma^{(1)}(\alpha^i)+S_1^3+S_3$。首先先考慮$\sigma^{(1)}(\alpha^i)$:

$$\begin{aligned}\sigma^{(1)}(\alpha^i) &= S_1\alpha^{2i} + S_1^2\alpha^i \\ &= \alpha^j\alpha^{2i} + \alpha^{2j}\alpha^i \\ &= \begin{bmatrix} \alpha_0^{2i} & \alpha_0^{2i+1} & \cdots & \alpha_0^{2i+m-1} \\ \alpha_1^{2i} & \alpha_1^{2i+1} & \cdots & \alpha_1^{2i+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^{2i} & \alpha_{m-1}^{2i+1} & \cdots & \alpha_{m-1}^{2i+m-1} \end{bmatrix} \begin{bmatrix} \alpha_0^j \\ \alpha_1^j \\ \vdots \\ \alpha_{m-1}^j \end{bmatrix} + \\ &\begin{bmatrix} \alpha_0^i & \alpha_0^{i+1} & \cdots & \alpha_0^{i+m-1} \\ \alpha_1^i & \alpha_1^{i+1} & \cdots & \alpha_1^{i+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^i & \alpha_{m-1}^{i+1} & \cdots & \alpha_{m-1}^{i+m-1} \end{bmatrix} \begin{bmatrix} 1 & \alpha_0^2 & \cdots & \alpha_0^{2(m-1)} \\ 0 & \alpha_1^2 & \cdots & \alpha_1^{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{2(m-1)} \end{bmatrix} \begin{bmatrix} \alpha_0^j \\ \alpha_1^j \\ \vdots \\ \alpha_{m-1}^j \end{bmatrix} \\ &= \mathbf{L}_i \cdot S_1\end{aligned}$$

其中，$\mathbf{L}_i$可以透過軟體事前運算求得，例如 C/C++或是 Matlab 得到。

$$\mathbf{L}_i = \begin{bmatrix} \alpha_0^{2i} & \alpha_0^{2i+1} & \cdots & \alpha_0^{2i+m-1} \\ \alpha_1^{2i} & \alpha_1^{2i+1} & \cdots & \alpha_1^{2i+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^{2i} & \alpha_{m-1}^{2i+1} & \cdots & \alpha_{m-1}^{2i+m-1} \end{bmatrix} + \begin{bmatrix} \alpha_0^i & \alpha_0^{i+1} & \cdots & \alpha_0^{i+m-1} \\ \alpha_1^i & \alpha_1^{i+1} & \cdots & \alpha_1^{i+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1}^i & \alpha_{m-1}^{i+1} & \cdots & \alpha_{m-1}^{i+m-1} \end{bmatrix} \begin{bmatrix} 1 & \alpha_0^2 & \cdots & \alpha_0^{2(m-1)} \\ 0 & \alpha_1^2 & \cdots & \alpha_1^{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \alpha_{m-1}^2 & \cdots & \alpha_{m-1}^{2(m-1)} \end{bmatrix}$$

第 3 圖係繪示本發明一實施例採用 n 個 m×m 矩陣解碼器的硬體架構示意圖。徵兆值計算器 301 依據解碼器所接收到的碼字 r_0、$r_{1...}r_{n-1}$，計算出徵兆值 S_1、S_3，立方器 303 與加法器 307 對徵兆值 S_1、S_3 進行運算，以得到 $S_3+S_1^3$。接著，乘法器 309 以及加法器 307 對徵兆值 S_1 以及矩陣 L_0、$L_1...L_{n-1}$ 進行運算，得到錯誤位置多項式 $\sigma(x)$。

表三

	本發明實施例	先前技術
元件	編碼器加上解碼器	解碼器
製程	90nm	180nm
運算所需時脈週期	1	1
時脈長度	2.5ns	4.51ns
資料大小	256	256
輸出率 (Throughput)	102.4Gb/s	56.76Gb/s
面積(μm^2)	41,705 或 35,205	283,512
邏輯閘數目	14.789 或 12484	N/A

表三則是整個(274, 256; 2)BCH 編解碼器與先前技術的比較結果，其中先前技術指的是 X. Wang, D. Wu, C. Hu, L. Pan, and R. Zhou, "Embedded high-speed BCH decoder for new-generation NOR flash memories," in IEEE Custom Integrated Circuits Conference, pp.195-198, Sep. 2009。

由表列數據可見，本發明一實施例除了具有解碼時間較短的優點之外，其所需的硬體複雜度亦明顯低於先前技術。由此可見本發明一實施例提出之方法與國際研討會上之相關研究成果相比亦毫不遜色，也顯示本發明一實施例提出之方法是具備雄厚競爭力且非常適合用於實際產品的架構。

以下的實施例依序介紹編碼器、解碼器，以及記憶體

系統，編碼器以及解碼器採用全套平行(Fully-Parallel)架構，以提升編碼速度，降低解碼延遲，達到應用上的需求。

請參照第 4 圖，其係繪示本發明一實施例記憶體系統之全套平行編碼方法流程圖。此一實施例當中的編碼方式是利用根，建立編碼矩陣後再拆解。

記憶體系統之全套平行編碼方法，首先將編碼方法採用之生成多項式分解為數個最小多項式(步驟 401)，也就是 $g(x)=LCM\{M_1(x),M_2(x),...,M_{2t}(x)\}$，其中，

g(x)為生成多項式，$M_1(x)$、$M_2(x)...M_{2t}(x)$則為最小多項式；然後求取這些最小多項式之數個根(步驟 403)，也就是 α^i 與其共軛根，並依據這些最小多項式的根，取得數個方程式(步驟 405)，也就是 $p(x)=u(x)x^{n-k}|x=\alpha^i$ 與其共軛根。

接著，依據這些方程式，產生第一編碼矩陣 H_{EN} (步驟 407)，也就是將這些方程式用矩陣的方式表示，$BP=AH_{EN}U$，第一編碼矩陣 H_{EN} 為：

$$\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2\times(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{(2t-1)\times 2^{\lceil 2t-1\rceil-1}} & \cdots & \alpha^{(2t-1)\times 2^{\lceil 2t-1\rceil-1}(k-1)} \end{bmatrix},$$

然後再對第一編碼矩陣 H_{EN} 進行分解(步驟 409)，也就是使 $H_{EN}=Q_{EN}\cdot\tilde{H}_{EN}$，以取得第二編碼矩陣 $\tilde{H}_{EN}$，其中第二編碼矩陣 $\tilde{H}_{EN}=\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(k-1)} \\ \cdots & \cdots & \cdots & .. \\ 1 & \alpha^{2t-1} & \cdots & \alpha^{(2t-1)\times(k-1)} \end{bmatrix}$，第二編碼矩陣係由電腦系統執行電腦軟體求得，例如 C/C++，或是 Matlab。

如前所述，第二編碼矩陣為一解碼器之校驗矩陣 $\tilde{H}_{SYN}$ 之部分矩陣，這個校驗矩陣 $\tilde{H}_{SYN}$ 等於：

$$\tilde{H}_{SYN} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \dots & (\alpha^3)^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{2t-1} & (\alpha^{2t-1})^2 & (\alpha^{2t-1})^3 & \dots & (\alpha^{2t-1})^{n-1} \end{bmatrix},$$

最後再根據第二編碼矩陣 $\tilde{H}_{EN}$ 進行編碼運算(步驟 411)，記憶體所需要的校驗位元 P 可依據第二編碼矩陣 $\tilde{H}_{EN}$ 以及非奇異矩陣 B 求得，也就是 $P=B^{-1}AQ_{EN}\tilde{H}_{EN}U$。

請參照第 5 圖，其係繪示本發明一實施例記憶體系統之全套平行解碼方法流程圖。在此一實施例的解碼方法當中，定義了一個新的錯誤位置方程式，並利用平方器整理方程式。

記憶體系統之全套平行解碼方法先依據校驗矩陣，計算得出數個徵兆值(步驟 501)，其中校驗矩陣 $\tilde{H}_{SYN}$ 與編碼器之第二編碼矩陣 $\tilde{H}_{EN}$ 具有部分共同元素，校驗矩陣 $\tilde{H}_{SYN}$ 與第二編碼矩陣 $\tilde{H}_{EN}$ 如第 4 圖的實施例所記載。

接著依據徵兆值，解出一錯誤位置多項式(步驟 503)，此錯誤位置多項式 $\sigma(x)=S_1x^2+S_1^2x+S_1^3+S_3$

$$=[T_0S_1\ T_1S_1\ \dots\ T_{m-1}S_1]\begin{bmatrix} X_0 \\ X_1 \\ \dots \\ X_{m-1} \end{bmatrix}+S_1^3+S_3$$，其中，

$\sigma(x)$為錯誤位置多項式，T_0、T_1、T_{m-1}為矩陣，S_1、S_3為徵兆值；矩陣 T_0、T_1 可由一電腦系統執行一電腦軟體求得，例如 C/C++或是 Matlab。

然後將數個基本元素α^0、$\alpha^1 \ldots \alpha^{n-1}$帶入錯誤位置多項式(步驟 505)，當碼字(Codeword)發生錯誤，則取得此錯誤位置多項式之根，(步驟 507)，再根據錯誤位置多項式之根，找出錯誤位置(步驟 509)，其中，如果α^i是錯誤位置多項式的根，表示第 i 個位置是有錯的。

請參見第 6 圖，其係繪示本發明一實施例記憶體系統之方塊圖。記憶體系統 600 主要含有編碼器 601、寫入電路 603、記憶體元件 605、讀取電路 607，以及解碼器 609。編碼器 601 利用前述第二編碼矩陣對輸入資料進行編碼，以產生至少一組碼字 (Codeword) c(x)，此碼字內含校驗碼(Parity Bits)。寫入電路 603 電性連接編碼器 601，以將碼字傳送至記憶體元件 605。記憶體元件 605 依據碼字進行存儲運作，此記憶體元件 605 為反或閘型快閃記憶體。

讀取電路 607 自記憶體元件 605 讀出資料。解碼器 609 則利用校驗矩陣解碼自記憶體元件 605 所讀出之資料，此解碼器 609 含有徵兆值計算器(Syndrome Calculator)609a、方程式解出器(Key Equation Solver)609b，以及錯誤位置搜尋器(Chien Search)609c。徵兆值計算器 609a 用來計算數個徵兆值。方程式解出器 609b 電性連接徵兆值計算器 609a，以解出錯誤位置多項式。錯誤位置搜尋器 609c 電性連接方程式解出器 609b，以根據錯誤位置多項式找出出錯的位置，在此一實施例當中，徵兆值計算器與編碼器為全套平行架構。

在記憶體系統 600 當中，解碼器 609 與編碼器 601 為組合電路，兩者分別佔用一個時脈週期的運算時間。編碼

器 601 所利用的第二編碼矩陣為解碼器 609 採用的校驗矩陣之一部分矩陣，由於兩者用來運算的矩陣具有共同元素，再加上解碼器 609 與編碼器 601 不會同時被驅動，而是在不同的時段獨自運作，因而編碼器 601 與解碼器 609 可以共用相應於此共同矩陣的電路，不會產生衝突，這樣一來，就減少了所需要的硬體電路。

以上實施例的記憶體系統及其全套平行編解碼方法，採用全平行化的架構，能在極短的時間內完成編碼或解碼之運算；由於編碼器、解碼器兩者所採用的矩陣具有共同元素，編解碼器的硬體資源得以高度共用，再加上新定義的錯誤位置多項式之應用，能夠使得運算之矩陣維度減少，降低所需之硬體複雜度，加快運算速度。

雖然本發明已以實施例揭露如上，然其並非用以限定本發明，任何在本發明所屬技術領域當中具有通常知識者，在不脫離本發明之精神和範圍內，當可作各種之更動與潤飾，因此本發明之保護範圍當視後附之申請專利範圍所界定者為準。

【圖式簡單說明】

為讓本發明之上述和其他目的、特徵、優點與實施例能更明顯易懂，所附圖式之說明如下：

第 1 圖係繪示本發明一實施例的有限場的立方器示意圖。

第 2 圖係繪示本發明一實施例採用常數乘法器以及平方器的解碼器硬體架構示意圖。

第 3 圖係繪示本發明一實施例採用 n 個 m×m 矩陣解碼器的硬體架構示意圖。

第 4 圖係繪示本發明一實施例記憶體系統之全套平行編碼方法流程圖。

第 5 圖係繪示本發明一實施例記憶體系統之全套平行解碼方法流程圖。

第 6 圖係繪示本發明一實施例記憶體系統之方塊圖。

【主要元件符號說明】

101：有限場立方器　　103：有限場平方器

105：有限場乘法器　　201：徵兆值計算器

203：立方器　　205：候選者產生器

207：加法器　　209：乘法器

301：徵兆值計算器　　303：立方器

307：加法器　　309：乘法器

401~411：步驟　　501~509：步驟

600：記憶體系統　　601：編碼器

603：寫入電路　　605：記憶體元件

607：讀取電路　　609：解碼器

609a：徵兆值計算器　　609b：方程式解出器

609c：錯誤位置搜尋器

七、申請專利範圍：

1. 一種記憶體系統之全套平行編碼方法，包含：

將編碼方法採用之一生成多項式分解為複數個最小多項式；

求取該些最小多項式之複數個根；以及

依據該些根產生一編碼矩陣或分解為複數個分解矩陣。

2. 如請求項 1 所述之記憶體系統之全套平行編碼方法，其中該編碼矩陣 H_{EN} 為 $\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2\times(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{(2t-1)\times 2^{t}2^{t-1}-1} & \cdots & \alpha^{(2t-1)\times 2^{t}2^{t-1}-1(k-1)} \end{bmatrix}$，該編碼矩陣 H_{EN} 第二欄(Column) $\begin{bmatrix} \alpha \\ \alpha^2 \\ \cdots \\ \alpha^{(2t-1)\times 2^{t}2^{t-1}-1} \end{bmatrix}$ 中，各個元素為該最小多項式的根，t、k 為正整數，且 k 代表訊息長度，t 代表錯誤更正能力。

3. 如請求項 1 所述之記憶體系統之全套平行編碼方法，其中該些分解矩陣可為一解碼器之一校驗矩陣之一部分矩陣。

4. 如請求項 3 所述之記憶體系統之全套平行編碼方

法，其中該校驗矩陣之該部分矩陣 $\tilde{H}_{EN}$ 為 $\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{2t-1} & \cdots & \alpha^{(2t-1)\times(k-1)} \end{bmatrix}$。

5. 一種記憶體系統之全套平行解碼方法，包含：

(a)依據一校驗矩陣，計算得出複數個徵兆值，其中該校驗矩陣與一編碼器的一分解矩陣具有部分共同元素；

(b)依據該些徵兆值，解出一錯誤位置多項式；

(c)將複數個基本元素代入該錯誤位置多項式，以取得該錯誤位置多項式之根；以及

(d)根據該錯誤位置多項式之該些根找出錯誤位置。

6. 如請求項 5 所述之記憶體系統之全套平行解碼方法，其中在步驟(C)當中，減少該錯誤位置多項式需要運算的冪次。

7. 如請求項 6 所述之記憶體系統之全套平行解碼方法，其中減少該錯誤位置多項式需要運算的冪次步驟包含：

利用一平方器之矩陣運算，將該錯誤位置多項式之奇數冪次和偶數冪次合併運算。

8. 一種記憶體系統，包含：

一編碼器，以利用一編碼矩陣對一輸入資料進行編

碼，產生一組碼字；

一記憶體元件，以依據該碼字進行存儲運作；

一寫入電路，電性連接該編碼器，以將該碼字傳送至該記憶體元件；

一讀取電路，以自該記憶體元件讀出資料：以及

一解碼器，以利用一校驗矩陣，解碼自該記憶體元件所讀出資料。

9. 如請求項 8 所述之記憶體系統，其中該編碼器將一生成多項式分解為複數個最小多項式，求取該些最小多項式之複數個根，並依據該些根產生一編碼矩陣，或依據該些根分解為複數個分解矩陣。

10. 如請求項 9 所述之記憶體系統，其中該編碼器所產生之該些分解矩陣為該解碼器之該校驗矩陣之一部分矩陣。

11. 如請求項 10 所述之記憶體系統，其中該校驗矩陣之該部分矩陣為 $\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{2t-1} & \cdots & \alpha^{(2t-1)\times(k-1)} \end{bmatrix}$。

12. 如請求項 8 所述之記憶體系統，其中該編碼器所

使用之該編碼矩陣 H_{EN} 為 $\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2\times(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{(2t-1)\times 2^{l}2t-1-1} & \cdots & \alpha^{(2t-1)\times 2^{l}2t-1-1(k-1)} \end{bmatrix}$。

13. 如請求項 8 所述之記憶體系統，其中該解碼器包含：

一徵兆值計算器，以依據該校驗矩陣，計算得出複數個徵兆值，其中該校驗矩陣與該編碼器之一分解矩陣具有部分共同元素；

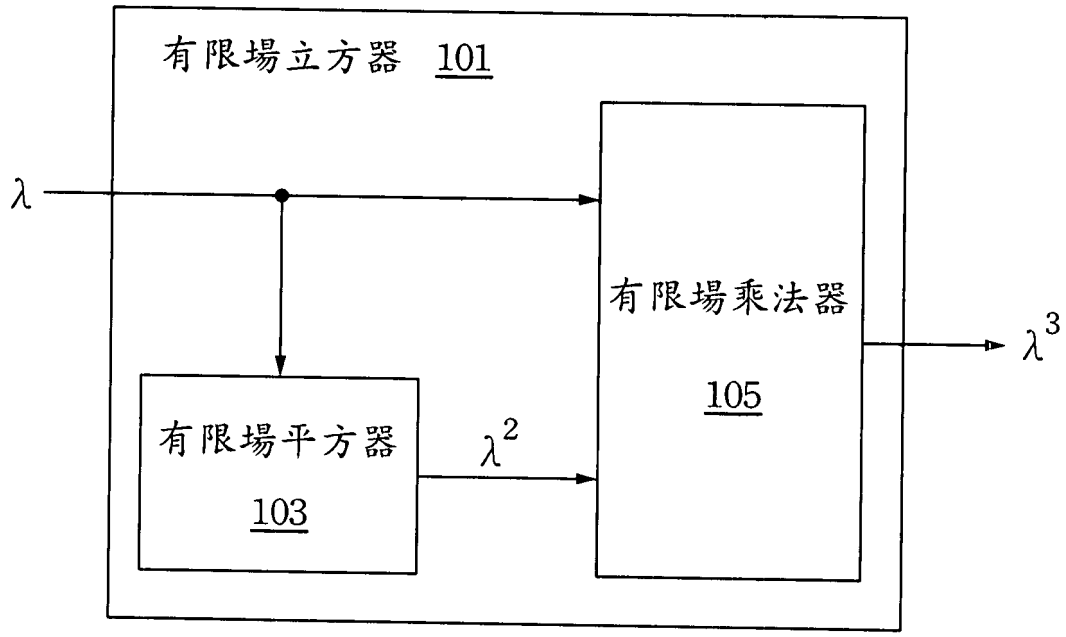
一錯誤方程式解出器，電性連接該徵兆值計算器，以解出一錯誤位置多項式；以及

一錯誤位置搜尋器，電性連接該錯誤方程式解出器，以根據該錯誤位置多項式，取得該錯誤位置多項式之一根，並根據該錯誤位置多項式之該根找出錯誤位置。

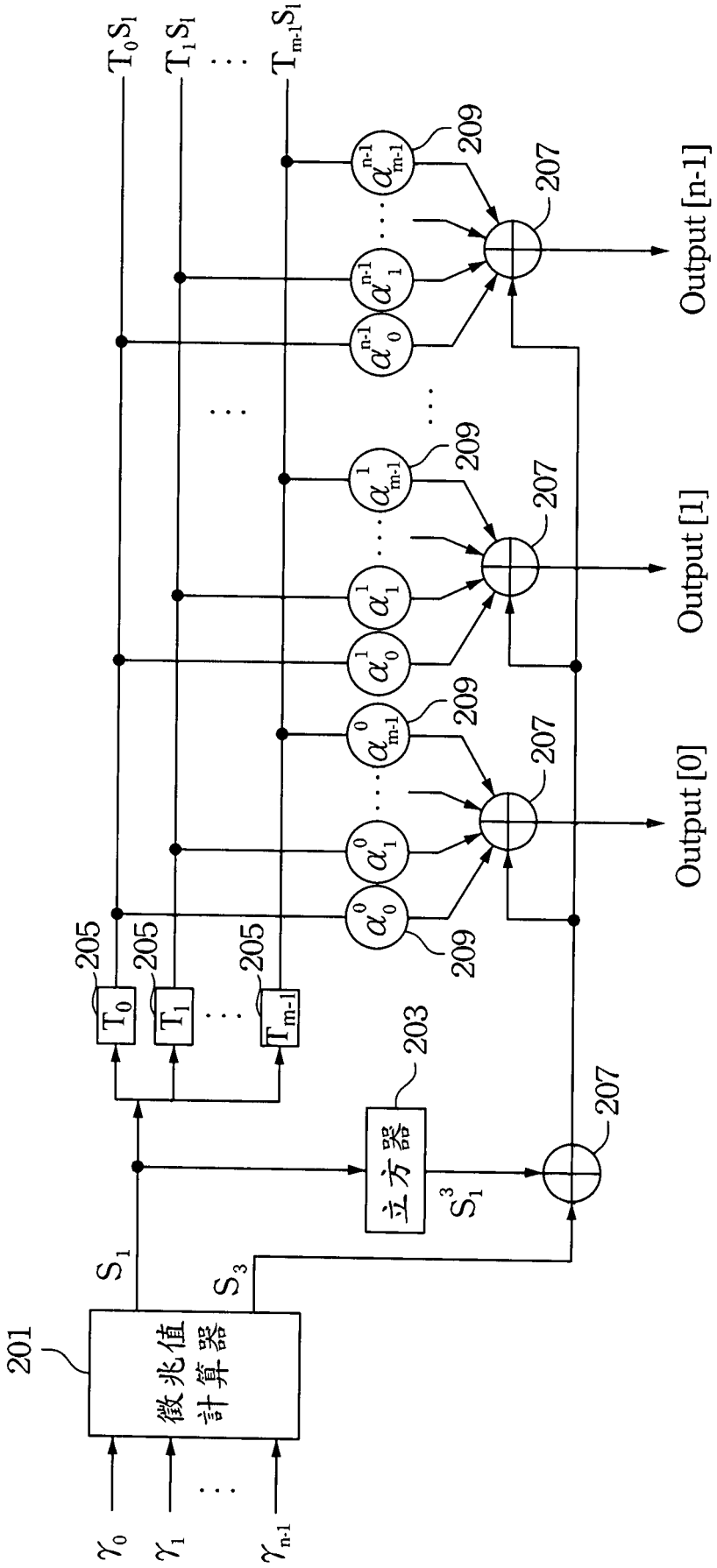
14. 如請求項 13 所述之記憶體系統，其中該解碼器將複數個基本元素帶入該錯誤位置多項式，以取得該錯誤位置多項式之該根，藉以減少該錯誤位置多項式需要運算的冪次。

15. 如請求項 14 所述之記憶體系統，其中該解碼器係利用一平方器之矩陣運算，將該錯誤位置多項式之奇數冪次和偶數冪次合併運算，藉以減少該錯誤位置多項式需要運算的冪次方。

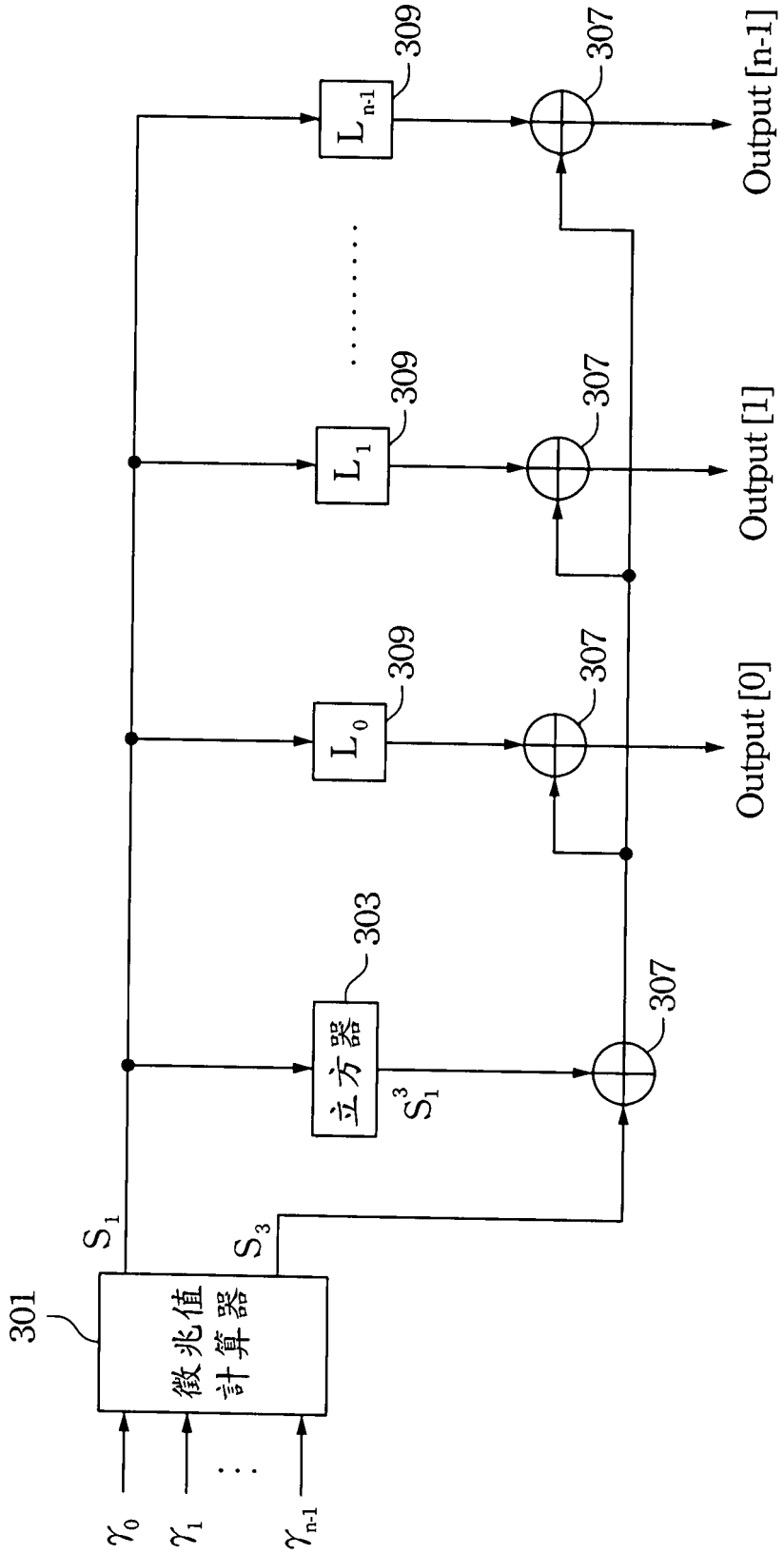
八、圖式：



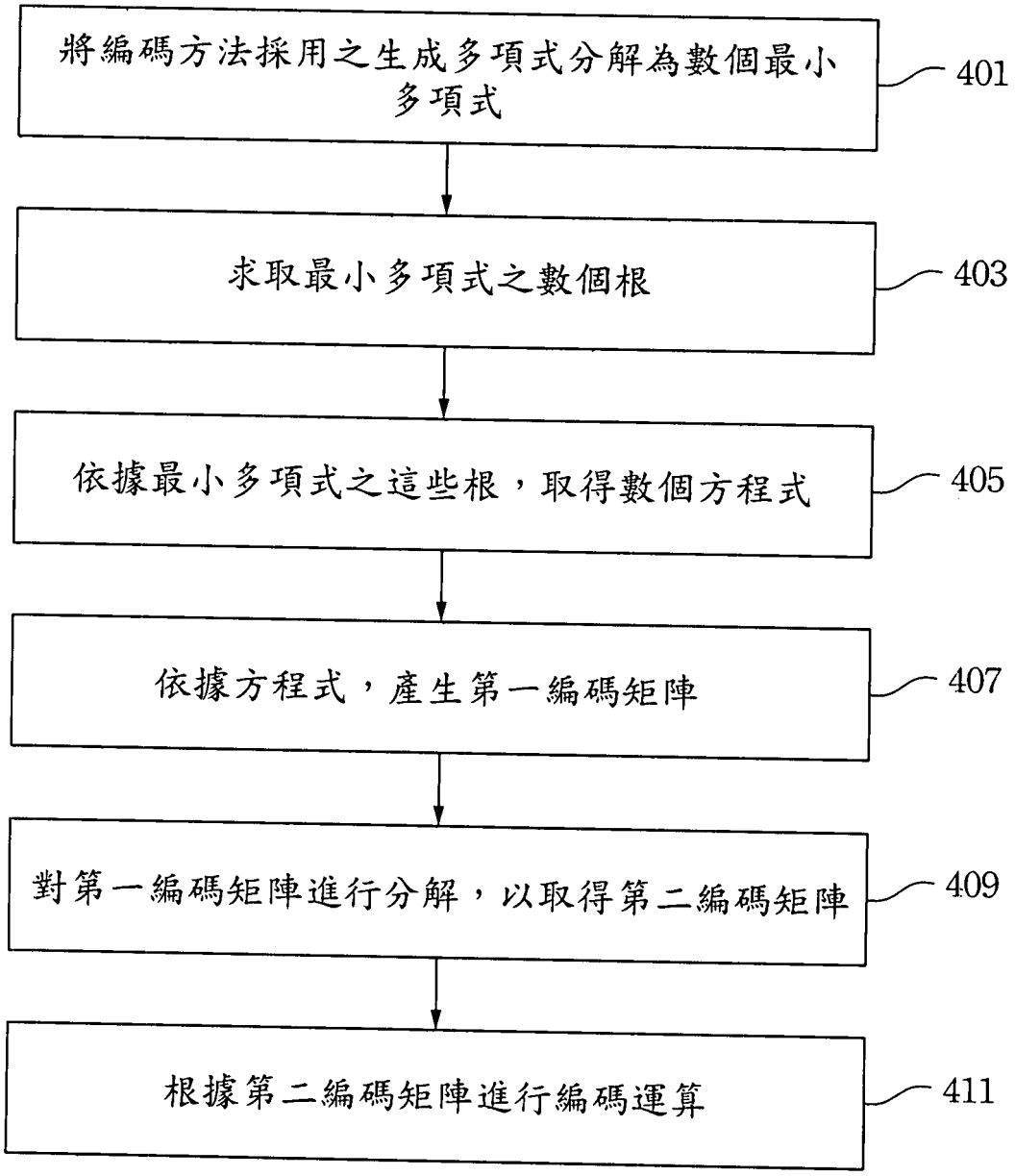
第 1 圖


201
微兆值計算器
γ_0
γ_1
γ_{n-1}
S_1
S_3
立方器
S_1^3
203
205
T_0
T_1
T_{m-1}
T_0S_1
T_1S_1
$T_{m-1}S_1$
209
207
Output [0]
Output [1]
Output [n-1]

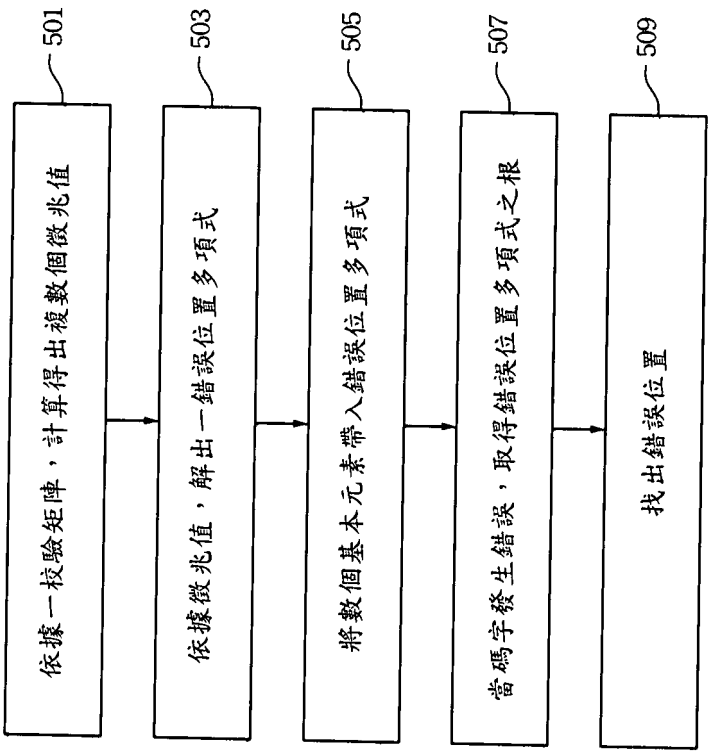
第 2 圖


301
微光值計算器
γ_0
γ_1
γ_{n-1}
S_1
S_3
303
立方器
S_1^3
307
309
L_0
L_1
L_{n-1}
Output [0]
Output [1]
Output [n-1]

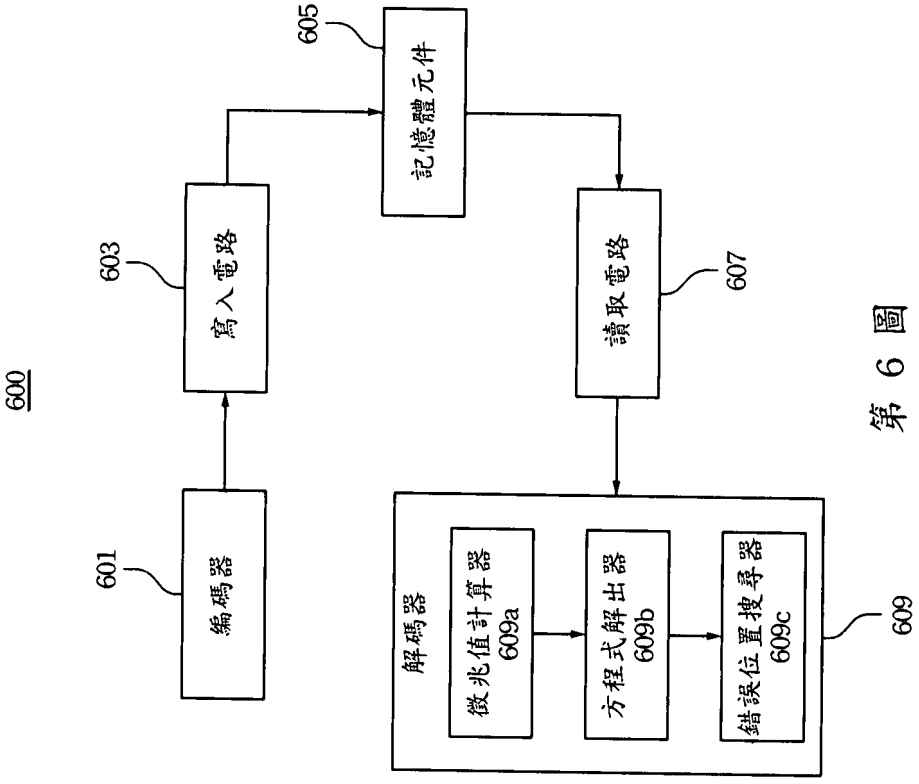
第 3 圖


將編碼方法採用之生成多項式分解為數個最小多項式
401
求取最小多項式之數個根
403
依據最小多項式之這些根，取得數個方程式
405
依據方程式，產生第一編碼矩陣
407
對第一編碼矩陣進行分解，以取得第二編碼矩陣
409
根據第二編碼矩陣進行編碼運算
411

第 4 圖


501
依據一校驗矩陣，計算得出複數個微兆值
503
依據微兆值，解出一錯誤位置多項式
505
將數個基本元素帶入錯誤位置多項式
507
當碼字發生錯誤，取得錯誤位置多項式之根
509
找出錯誤位置

第 5 圖


600
601
編碼器
603
寫入電路
605
記憶體元件
607
讀取電路
609
解碼器
微光值計算器
609a
方程式解出器
609b
錯誤位置搜尋器
609c

第 6 圖