

國立交通大學

電信工程學系碩士班

碩士論文

低密度同位檢查碼技術及其在數位電視廣播系統應用之研究

A Study on LDPC Coding Technique and Its Application to Digital Video Broadcasting Systems

研究生：林永哲
指導教授：黃家齊 博士

中華民國九十四年八月

低密度同位檢查碼技術及其在數位
電視廣播系統應用之研究
A Study on LDPC Coding Technique and Its
Application to Digital Video Broadcasting Systems

研究生：林永哲

Student：Yong-Jhe Lin

指導教授：黃家齊 博士

Advisor：Dr. Chia-Chi Huang

國立交通大學
電信工程學系碩士班
碩士論文



Submitted to Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Communication Engineering
June 2005
Hsinchu, Taiwan, Republic of China

中華民國九十四年八月

低密度同位檢查碼技術及其在數位 電視廣播系統應用之研究

研究生：林永哲

指導教授：黃家齊 博士

國立交通大學
電信工程學系



錯誤更正碼的種類很多，其中有一種名為低密度同位檢查碼的錯誤更正碼在最近受到重視，因為其編碼增益直逼雪儂極限，解碼器簡單且硬體成本低。在本論文中，我們將低密度同位檢查碼的編碼與解碼方式應用於數位電視廣播系統中。傳統的數位電視廣播系統的編碼方式是由兩層編碼所組成，分別為外層編碼(里德所羅門碼)及內層編碼(迴旋碼)，而且這兩層的編碼都有各自的交錯器。為了簡化數位電視廣播系統的編碼，我們使用一個低密度同位檢查碼來取代原本的里德所羅門碼及迴旋碼，及其所對應的交錯器。我們藉著計算機模擬來比較低密度同位檢查碼與傳統的兩層編碼在數位電視廣播系統中錯誤率的表現。

A Study on LDPC Coding Technique and Its Application to Digital Video Broadcasting Systems

Student: Yong-Jhe Lin

Advisor: Dr. Chia-Chi Huang

Institute of Communication Engineering
National Chiao Tung University

The logo of National Chiao Tung University, featuring a blue circular emblem with a gear-like border and a central design. The word "Abstract" is overlaid on the logo.

Abstract

This thesis proposes a new encoding and decoding method which can be applied to Digital Video Broadcasting (DVB) Systems. There are many kinds of error correction codes. Low Density Parity Check (LDPC) Code gets a lot of attention recently because its coding gain is near the Shannon limit. Moreover, the decoder is easy to implement and the hardware cost is low. Conventional Digital Video Broadcasting System uses two concatenated codes, an outer code (Reed-Solomon code) and an inner code (Convolutional code), respectively. These two codes have different interleavers. In order to simplify the coding scheme of Digital Video Broadcasting Systems, here we use a single Low Density Parity Check code to substitute for the original Reed-Solomon code and the Convolutional code, together with their interleavers. We use computer simulation to compare the bit error rate performance of the Low Density Parity Check Code and the original concatenated codes in Digital Video Broadcasting Systems.

誌謝

兩年研究生的日子，在師長諄諄教誨與同學的相互扶持中，歡樂的劃下完美的句點。這之間學到很多也得到很多，期望自己能將所學貢獻給社會，也不枉父母與從小到大的老師們對我的栽培與愛戴。能運用所學為這個社會盡一份棉力，並實踐夢想，身為畢業生的我，感到無比光榮與鼓舞。

能順利的完成學業，要感謝論文指導教授黃家齊先生，在教授細心教導下，所有課業上的問題都迎刃而解；在人生觀上也給予寶貴的意見，謝謝您！老師。

其次，感謝實驗室的學長、學姊和學弟妹們對我生活上的關心和課業上的協助，共同度過這一段歡樂時光。一起寫論文的子豪、清凱、冠樺，是患難相助的好同學們，無論是課業討論、論文撰寫、工作站維護等等，感激這些同學的鼎力幫忙。寢室裡一起生活的室友，柏良、凱元與景程，兩年來就像家人般給予我全力的支持，是我一生中難忘的朋友。

另外，好朋友佳璘、佳穗、育維、憲明、聲政、弘恩、一夫和適任等等，更帶給我許多歡笑也與分享予我寶貴經驗，讓我學習到許多新知識，增廣了見聞，大家相處在「北國精靈」更是笑聲不斷、熱情不減，透過成員們的知識交流與心得分享，這個溫馨的社團必定可以永續經營下去。

最後衷心謝謝我的親愛的家人們，爸爸、媽媽、姊姊和弟弟，是我最大的精神支柱，讓我無後顧之憂地完成碩士學位，這份榮耀是屬於全家人的。

林永哲 謹誌

中華民國九十四年八月

目錄

中文摘要.....	i
英文摘要.....	ii
誌謝.....	iii
目錄.....	iv
圖目錄.....	v
表目錄.....	vi
第一章.....	1
簡介.....	1
第二章.....	2
2.1 里德所羅門碼介紹.....	2
2.2 編碼器.....	3
2.3 解碼器.....	7
第三章.....	14
3.1 低密度同位檢查碼介紹.....	14
3.2 編碼器.....	15
3.3 解碼器.....	18
3.3.1 Tanner Graph.....	18
3.3.2 解碼的機率演示圖.....	20
3.3.3 Bit node 到 Check node 的機率算法.....	21
3.3.4 Check node 到 Bit node 的機率算法.....	23
3.3.5 LDPC 解碼時的 Update Equation.....	29
第四章.....	38
4.1 LDPC block length 求得.....	38
4.2 模擬結果.....	44
第五章.....	49
結論與未來發展的方向.....	49
參考文獻.....	50

圖目錄

圖 2.1 Linear Feedback Shift Register 架構圖.....	12
圖 3.1 低密度同位檢查碼編碼方塊圖.....	17
圖 3.2 Tanner Graph.....	18
圖 3.3 $HV^T = 0$ 方塊圖.....	19
圖 3.4 解碼器的架構圖.....	20
圖 3.5 Bit node 的 message-passing.....	21
圖 3.6 Bit node 的 message-passing.....	23
圖 3.7 LDPC decoder.....	29
圖 3.8 bit node 傳送到 check node 的機率資訊.....	30
圖 3.9 check node 傳送到 bit node 的機率資訊.....	30
圖 3.10 $\Psi(x)$ 的函數圖.....	35
圖 4.1 DVB-T 系統發射機的功能方塊圖.....	39
圖 4.2 DVB-T 系統接收機的功能方塊圖.....	39
圖 4.3 LDPC 取代 DVB-T 兩層編碼的發射機功能方塊圖.....	40
圖 4.4 LDPC 取代 DVB-T 兩層編碼的接收機功能方塊圖.....	40
圖 4.5 里德所羅門碼 RS(204,188,t=8)錯誤保護封包.....	41
圖 4.6 外層編碼之交錯分佈後的資料結構分佈, I=12 bytes.....	41
圖 4.7 外層交錯器和反交錯器結構圖.....	42
圖 4.8 交錯器的輸入與輸出關係圖.....	43
圖 4.9 DVB-T 在 AWGN 通道境下模擬結果.....	45
圖 4.10 DVB-T 在 fading 通道境下 30 公里車速模擬結果.....	45
圖 4.11 DVB-T 在 fading 通道境下 60 公里車速模擬結果.....	46
圖 4.12 LDPC 在 AWGN 通道境下模擬結果.....	46
圖 4.13 LDPC 在 fading 通道境下 30 公里車速模擬結果.....	47
圖 4.14 LDPC 在 fading 通道境下 60 公里車速模擬結果.....	47
圖 4.15 LDPC 與 RS+convolution 在 fading 通道境下模擬結果.....	48

表目錄

表格 2.1 由本質多項式 $p(x) = x^4 + x + 1$ 建構 $GF(16)$	4
表格 4.1 系統的模擬參數.....	44



第一章

簡介

在歐規的數位影像廣播系統之地面廣播系統(DVB-T System)的架構中由兩層編碼所組成，且各自有其交錯器(Interleaver)，外層編碼器(Outer Coder)為 Reed-Solomon code、內層編碼器(Inner Coder)為 Convolutional code。

因為 LDPC(Low Density Parity Check code)的效能可以比 Turbo code 還好，所以在論文中研究使得 LDPC 的 block length(編碼後的長度)等於 DVB-T 系統中兩層交錯器總共的深度，根據這個深度來決定 LDPC 的碼字(code word)長度，以模擬結果來分析 LDPC 在同樣 DVB-T 系統環境下的效能。在論文中詳細針對 Reed-Solomon code 及 LDPC code 的編碼、解碼及所用到的演算法方法做一個介紹。

第二章

2.1 里德所羅門碼介紹

里德所羅門碼(Reed-Solomon Codes)，簡稱 RS Code，里德所羅門碼是在西元 1960 年由 I. Reed 以及 G. Solomon 在他們麻省理工學院(M.I.T.)實驗室所共同發明。自從里德所羅門碼被發明之後，這麼多年來它一直被廣泛的應用於現今工業界各種方面，例如家庭裡的光碟機、硬體儲存設備及外太空的太空船通訊。

里德所羅門碼的效能非常好，但其缺點為需要用大的有限場(Finite field) $GF(q)$ 來建立長的里德所羅門碼，有限場(Finite field)亦可稱為 Galois Field。且里德所羅門碼的算術複雜度隨符號域大小 $GF(q)$ 成指數成長，因此里德所羅門碼的編解碼複雜度將隨其長度增加而變得非常高。里德所羅門碼的解碼過程一般分為四個步驟：計算病徵函數(Syndromes)、找出錯誤位置多項式(Error Locator Polynomials)、解出錯誤位置(Error Locator)與計算錯誤值(Error Value)。

2.2 編碼器

假設一個有限場(Finite field)裡有 q 個元素(elements)，通常以符號表示為 $GF(q)$ ，其中 Finite field 裡面的元素個數 q 一定是以 p^m 的形式個數來表示， p 在此表示為一個質數，而 m 表示為一個正整數。在 Finite field 裡的元素 α 其次方(order)為最小的正整數 m ，且滿足 $\alpha^m=1$ ，則 α 稱為本質元素(primitive elements)，在 $GF(q)$ 中的元素都可以用 α 的次方的形式來表示，也使得 $GF(q)$ 的乘法運算變得更為簡單，而連續 $q-1$ 個 α 的次方所表示出的數值是相異的，所以 $GF(q)$ 總共有 $q-1$ 個非零的元素，分別表示成 $\{1, \alpha, \alpha^2, \alpha^3, \dots, \alpha^{q-2}\}$ 。

假設在 $GF(q)$ 中的本質元素 α 是一個本質多項式(primitive polynomial) $p(x)$ 的根，所以我們可以利用這個本質多項式 $p(x)$ 來建構一個 $GF(q)$ 。在原本的本質多項式是以多項式型式(polynomial representations)的表示方法，而根據本質元素是多項式的一個根的關係，我們可以把他代換成指數形式(exponential representations)的表示方法，這時是利用把多項式型式的表示方法取 *modulo* 轉換成指數形式(exponential representations)的表示方法。

以下我們舉一個例子來說明 $GF(q)$ 的建立，首先假設在一個 $GF(16)$ 中，有一個本質多項式 $p(x) = x^4 + x + 1$ ，本質元素 α 是本質多項式 $p(x)$ 其中的一個根，所以我們可以得到 $\alpha^4 + \alpha + 1 = 0$ ，在二進位表示法的算術運算下， $\alpha^4 + \alpha + 1 = 0$ 相當於 $\alpha^4 = \alpha + 1$ 。下列表格 2.1 為 $GF(16)$ 的建立。

指數形式	多項式型式
0	0
α^0	1
α^1	x^1
α^2	x^2
α^3	x^3
α^4	$x^4 = x + 1$
α^5	$x^5 = x + x^2$
α^6	$x^6 = x^2 + x^3$
α^7	$x^7 = x^3 + x + 1$
α^8	$x^8 = x^2 + 1$
α^9	$x^9 = x^3 + x$
α^{10}	$x^{10} = x^2 + x + 1$
α^{11}	$x^{11} = x^3 + x^2 + x$
α^{12}	$x^{12} = x^3 + x^2 + x + 1$
α^{13}	$x^{13} = x^3 + x^2 + 1$
α^{14}	$x^{14} = x^3 + 1$

表格 2.1 由本質多項式 $p(x) = x^4 + x + 1$ 建構 $GF(16)$

里德所羅門碼其編碼方式如下，假設有一連串 k 個的消息符號 (information symbol) m ，可以表示成 $\{m_0, m_1, m_2, \dots, m_{k-2}, m_{k-1}\}$ ，經過編碼後的里德所羅門碼的碼字 (code word) c ，可以表示成 $\{c_0, c_1, c_2, \dots, c_{q-1}\}$ ，這是因為 $GF(q)$ 裡總共有 q 個元素，而碼字是這 q 個元素中的組合，我們先利用 k 個的消息符號來組成一個消息符號多項式 $m(x)$ ， $m(x) = m_0 + m_1x + \dots + m_{k-2}x^{k-2} + m_{k-1}x^{k-1}$ ，因為消息符號 (information symbol) 是 $GF(q)$ 裡面的元素，所以有 q 種不同的可能性，因此里

德所羅門碼的碼字 c 總共有 q^k 種可能性，碼字 c 可以表示為 $c = (c_0, c_1, \dots, c_{q-1}) = [m(0), m(\alpha), m(\alpha^2), \dots, m(\alpha^{q-1})]$ 。

公式(2.1)以方程式的形態表示里德所羅門碼的碼字 c ，若以矩陣的型態來表示的話，則如公式(2.2)所示。

$$\begin{aligned}
 m(0) &= m_0 \\
 m(\alpha) &= m_0 + m_1\alpha + m_1\alpha^2 + \dots + m_{k-1}\alpha^{k-1} \\
 m(\alpha^2) &= m_0 + m_1\alpha^2 + m_1\alpha^4 + \dots + m_{k-1}\alpha^{2(k-1)} \\
 &\vdots \\
 m(\alpha^{q-2}) &= m_0 + m_1\alpha^{q-2} + m_1\alpha^{2(q-2)} + \dots + m_{k-1}\alpha^{(q-2)(k-1)} \\
 m(\alpha^{q-1}) &= m_0 + m_1\alpha^{q-1} + m_1\alpha^{2(q-1)} + \dots + m_{k-1}\alpha^{(q-1)(k-1)}
 \end{aligned} \tag{2.1}$$

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & \alpha & \alpha^2 & \dots & \alpha^{(k-1)} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{(q-1)} & \alpha^{2(q-1)} & \dots & \alpha^{(q-1)(k-1)} \end{bmatrix} \cdot \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{k-1} \end{bmatrix} = \begin{bmatrix} m(0) \\ m(\alpha) \\ m(\alpha^2) \\ \vdots \\ m(\alpha^{q-1}) \end{bmatrix} \tag{2.2}$$

以上的里德所羅門碼表示法可以寫成 (n, k) RS Code， n 代表經過編碼後的消息符號長度，也就是跟上面 q 個元素是一樣的 ($n = q$)，而其中 k 代表輸入碼字的消息符號長度。以 $\frac{k}{n}$ 來表示里德所羅門碼的編碼率 (Code Rate)，其中同位元檢查碼的長度 (parity-check symbols) 為 $n - k$ 個。

假設里德所羅門碼的產生多項式 (generator polynomial) 為 $g(x)$ ， $g(x) = \prod_{i=1}^{n-k} (x - \alpha^i) \triangleq g_0 + g_1x + g_2x^2 + \dots + g_{n-k}x^{n-k}$ 。同位元檢查碼表示成 p ，總共有 $n - k$ 個同位元檢查碼， $p = [p_0, p_1, p_2, \dots, p_{n-k-1}]$ ，所以碼字表示為 C ，為消息符號 m 後面加上同位元檢查碼 p 所產生的。且碼字 $c(x) = m(x)g(x)$ ，也就是說消息符號 $m(x)$ 乘上產生多項式 $g(x)$ 。

$$C = [c_0, c_1, c_2, \dots, c_{n-1}] = [p_0, p_1, p_2, \dots, p_{2t-1}, m_0, m_1, m_2, \dots, m_{k-1}] \tag{2.3}$$

$$\begin{aligned}
c(x) &= m(x)g(x) \\
&= (m_0 + m_1x + \dots + m_{k-2}x^{k-2} + m_{k-1}x^{k-1}) \cdot (g_0 + g_1x + g_2x^2 + \dots + g_{n-k}x^{n-k}) \\
&= c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1} \\
&= \sum_{i=0}^{n-k-1} p_i x^i + x^{n-k} m(x) \\
&= p(x) + x^{n-k} m(x)
\end{aligned} \tag{2.4}$$

其中同位元檢查碼 $p(x)$ 的選擇是由產生多項式 $g(x)$ 可以整除碼字多項式 $c(x)$ 決定，在二進位表示法的算術運算下，可知同位元檢查碼 $p(x) = R_{g(x)}[x^{n-k}m(x)]$ ，而符號 $R_{g(x)}[\cdot]$ 代表被產生多項式 $g(x)$ 除之後所剩下的餘式，即 $p(x)$ 是 $x^{n-k}m(x)$ 被 $g(x)$ 除之後所剩下的餘式。

綜合以上編碼的方式，我們可以整理出里德所羅門碼的三個編碼步驟：

第一步：由消息符號可以算出消息符號多項式

$$m(x) = m_0 + m_1x + \dots + m_{k-2}x^{k-2} + m_{k-1}x^{k-1}, \text{ 進而求出 } x^{n-k}m(x)。$$

第二步：求出同位元檢查碼 $p(x) = R_{g(x)}[x^{n-k}m(x)]$ 。

第三步：由第一步驟所求出的 $x^{n-k}m(x)$ ，再加上第二步驟中運算出的 $p(x)$ ，即可求出碼字多項式 $c(x)$ 。

2.3 解碼器

里德所羅門碼其解碼方式如下，假設錯誤向量(error vector)為 $e = [e_0, e_1, \dots, e_{n-1}]$ ，因此可以寫出錯誤多項式為 $e(x) = e_0 + e_1x + \dots + e_{n-1}x^{n-1}$ ，所以訊號經過通道之後所接收到的訊號 $v(x)$ 為碼字多項式 $c(x)$ 加上錯誤多項式 $e(x)$ ， $v(x) = c(x) + e(x) = v_0 + v_1x + v_2x^2 + \dots + v_{n-1}x^{n-1}$ ， $v(x)$ 可以由產生多項式 $g(x)$ 的根

$\{\alpha, \alpha^2, \dots, \alpha^{n-k}\}$ 算出， $g(x) = \prod_{i=1}^{n-k} (x - \alpha^i) \triangleq g_0 + g_1x + g_2x^2 + \dots + g_{n-k}x^{n-k}$ 。因為碼字

多項式 (code word polynomial) $c(x)$ 可以被產生多項式 $g(x)$ 整除，而且 $g(\alpha^j) = 0$

for $j = 1, 2, 3, \dots, 2t$ ，其中 $t = \frac{n-k}{2}$ 代表里德所羅門碼可以更正的錯誤數量。利用

這些關係，我們可以導出接收到的訊號 $v(x)$ 有著下列的式子：

$$\begin{aligned}
 v(\alpha^j) &= c(\alpha^j) + e(\alpha^j) \\
 &= m(\alpha^j) \cdot g(\alpha^j) + e(\alpha^j) \\
 &= e(\alpha^j) \\
 &= e_0 + e_1\alpha^j + e_2\alpha^{2j} + \dots + e_{n-1}\alpha^{(n-1)j} \\
 &= \sum_{i=0}^{n-1} e_i\alpha^{ij}, \quad j = 1, 2, \dots, 2t
 \end{aligned} \tag{2.5}$$

所以由式子(2.5)中的 $2t$ 個方程式可以算出病徵函數(syndromes)多項式 S_j ，只要把錯誤(error pattern) e_i 從 $i = 0, 1, 2, \dots, n-1$ 全部求出就可以得到 S_j 了。

$$S_j \triangleq v(\alpha^j) = \sum_{i=0}^{n-1} e_i\alpha^{ij}, \quad j = 1, 2, \dots, 2t \tag{2.6}$$

假設傳送的訊號經過通道(channel)後有發生 v 個錯誤，且這 v 個錯誤在里德所羅門碼可以更正的錯誤數量 t 範圍內， $0 \leq v \leq t$ ，一開始我們不知道這些 v 個錯誤發生在哪些位置上，我們先假設這 v 個位置分別是 $i_1, i_2, i_3, \dots, i_v$ ，因此我們可以知道錯誤的多項式(error polynomial)為下式：

$$e(x) = e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + e_{i_3}x^{i_3} + \dots + e_{i_v}x^{i_v} \tag{2.7}$$

其中 e_{i_l} 代表錯誤的多項式中第 l 個位置錯誤的值，因為一開始我們都不知道 $i_1, i_2, i_3, \dots, i_v$ 錯誤的位置發生在哪裡，及 $e_{i_1}, e_{i_2}, e_{i_3}, \dots, e_{i_v}$ 錯誤的值是多少，於是接下來我們要求出錯誤的位置和其值。從式子 (2.5) 及 (2.6) 我們可以求出 $S_1, S_2, S_3, \dots, S_v$ 。

$$\begin{aligned} S_1 &= v(\alpha) \\ &= e(\alpha) \\ &= e_{i_1} \alpha^{i_1} + e_{i_2} \alpha^{i_2} + e_{i_3} \alpha^{i_3} + \dots + e_{i_v} \alpha^{i_v} \end{aligned} \quad (2.8)$$

根據公式 (2.8) 我們在這邊做一個代數變換的動作，首先定義錯誤的值 (error value) 為 Y_l ， $Y_l = e_{i_l}$ ， $l = 1, 2, 3, \dots, v$ ，另外定義錯誤位置 (error location number) 為 X_l ， $X_l = \alpha^{i_l}$ ， $l = 1, 2, 3, \dots, v$ ， i_l 表示第 l 個錯誤的真實位置。所以把這兩個重新定義後變數代入到公式 (2.8)，我們可以得到下式：

$$\begin{aligned} S_1 &= v(\alpha) = e(\alpha) = e_{i_1} \alpha^{i_1} + e_{i_2} \alpha^{i_2} + e_{i_3} \alpha^{i_3} + \dots + e_{i_v} \alpha^{i_v} \\ &= Y_1 X_1 + Y_2 X_2 + Y_3 X_3 + \dots + Y_v X_v \end{aligned} \quad (2.9)$$

另外我們也可以根據新的定義把公式 (2.8) 代換成下式：

$$S_j = Y_1 X_1^j + Y_2 X_2^j + Y_3 X_3^j + \dots + Y_v X_v^j, \quad j = 1, 2, \dots, 2t \quad (2.10)$$

以上說明里德所羅門碼整個解碼過程有下列兩個步驟：

第一步：由公式 (2.6) 計算特徵多項式

$$S_j \triangleq v(\alpha^j) = \sum_{i=0}^{n-1} e_i \alpha^{ij} = \sum_{l=1}^v e_{i_l} X_l^j, \quad j = 1, 2, \dots, 2t$$

第二步：由公式 (2.10) $S_j = Y_1 X_1^j + Y_2 X_2^j + Y_3 X_3^j + \dots + Y_v X_v^j$ ， $j = 1, 2, \dots, 2t$ ，我們可以從總共 $2t$ 個非線性多項式中分別解出錯誤的位置 $X_l = \alpha^{i_l}$ ， $l = 1, 2, 3, \dots, v$ ，還有錯誤的 $Y_l = e_{i_l}$ ， $l = 1, 2, 3, \dots, v$ 。

因為上面所提到的解碼方法要解 $2t$ 個方程式，若 t 很大時，則解方程式變的比較複雜，為了有效率來解出以上介紹里德所羅門碼錯誤的位置和錯誤的值，後人發明出一種演算法來解碼，稱為 Forney's algorithm，Forney's algorithm 中定義

一些參數：病徵多項式 (syndrome polynomial) $S(x) = \sum_{i=1}^{2t} S_i x^i$ ，錯誤位置多項式 (error

locator polynomial) 為 $\Lambda(x)$ ，且定義 $\Lambda(x)$ 多項式的根是錯誤位置(error location number) 為 $X_l = \alpha^l$ ， $l=1,2,3,\dots,v$ 的倒數。

$$\begin{aligned}\Lambda(x) &\triangleq 1 + \Lambda_1 x + \Lambda_2 x^2 + \dots + \Lambda_v x^v \\ &\triangleq \prod_{l=1}^v (1 - xX_l) \quad , \quad X_l = \alpha^l\end{aligned}\tag{2.11}$$

把上式(2.11)中的 $\Lambda(x)$ 一次微分得到 $\Lambda'(x)$ 。

$$\begin{aligned}\Lambda'(x) &= \left[\prod_{l=1}^v (1 - xX_l) \right]' \\ &= - \sum_{l=1}^v \left[X_l \prod_{j \neq l} (1 - xX_j) \right]\end{aligned}\tag{2.12}$$

接下來我們把 X_k^{-1} 代入到 $\Lambda'(x)$ ，得到式子(2.13)。

$$\begin{aligned}\Lambda'(X_k^{-1}) &= - \sum_{l=1}^v \left[X_l \prod_{j \neq l} (1 - X_k^{-1} X_j) \right] \\ &= - X_k \prod_{j \neq k} (1 - X_k^{-1} X_j)\end{aligned}\tag{2.13}$$

再來定義錯誤值多項式(error evaluator polynomial) 為 $\Omega(x)$

$$\begin{aligned}\Omega(x) &= \Lambda(x)[1 + S(x)] \quad \text{mod } x^{2t+1} \\ &= \left[\prod_{l=1}^v (1 - xX_l) \right] \left[1 + \sum_{j=1}^{\infty} S_j x^j \right] \\ &= \left[\prod_{l=1}^v (1 - xX_l) \right] \left[1 + \sum_{j=1}^{\infty} \left(\sum_{l=1}^v e_l X_l^j \right) x^j \right] \\ &= \left[\prod_{l=1}^v (1 - xX_l) \right] \left[1 + \sum_{l=1}^v e_l \sum_{j=1}^{\infty} (X_l x)^j \right] \\ &= \left[\prod_{l=1}^v (1 - xX_l) \right] \left[1 + \sum_{l=1}^v e_l \left(\frac{X_l x}{1 - X_l x} \right) \right] \\ &= \Lambda(x) + \sum_{l=1}^v \left[e_l X_l x \prod_{j \neq l} (1 - X_l x) \right]\end{aligned}\tag{2.14}$$

接下來我們把 X_k^{-1} 代入到 $\Omega(x)$ ，得到式子(2.15)。

$$\begin{aligned}\Omega(X_k^{-1}) &= \Lambda(X_k^{-1}) + \sum_{l=1}^v \left[e_{ji} X_l X_k^{-1} \prod_{j \neq l} (1 - X_j X_k^{-1}) \right] \\ &= e_{ik} \prod_{j \neq k} (1 - X_j X_k^{-1})\end{aligned}\quad (2.15)$$

由式子(2.13)及(2.15)我們經過運算之後可以求出 e_{ik} 。

$$\begin{aligned}e_{ik} &= \frac{\Omega(X_k^{-1})}{\prod_{j \neq k} (1 - X_j X_k^{-1})} \\ &= -\frac{\Omega(X_k^{-1})}{\frac{\Lambda'(X_k^{-1})}{X_k}} \\ &= -\frac{X_k \Omega(X_k^{-1})}{\Lambda'(X_k^{-1})}\end{aligned}\quad (2.16)$$

以上我們已經算出錯誤的值(error value)為 e_{ik} ，接下來要算出錯誤位置多項式(error locator polynomial)的係數 $X_l = \alpha^i$ ， $l=1,2,3,\dots,v$ 則需要下列的過程：

$$\begin{aligned}\Lambda(x) &= 1 + \Lambda_1 x + \Lambda_2 x^2 + \dots + \Lambda_v x^v \\ \Rightarrow \Lambda(X_l^{-1}) &= 1 + \Lambda_1 X_l^{-1} + \Lambda_2 (X_l^{-1})^2 + \dots + \Lambda_v (X_l^{-1})^v = 0 \\ \Rightarrow 0 &= [Y_l X_l^{j+v}] \left[1 + \Lambda_1 X_l^{-1} + \Lambda_2 (X_l^{-1})^2 + \dots + \Lambda_v (X_l^{-1})^v \right] \\ \Rightarrow Y_l (X_l^{j+v} + \Lambda_1 X_l^{j+v-1} + \Lambda_2 X_l^{j+v-2} + \dots + \Lambda_v X_l^j) &= 0, \quad l=1,2,3,\dots,v\end{aligned}\quad (2.17)$$

針對公式(2.17)的 $l=1,2,3,\dots,v$ ，我們把這 v 個錯誤全部代入到公式之後，則公式(2.17)可以改寫成下式：

$$\begin{aligned}Y_l (X_l^{j+v} + \Lambda_1 X_l^{j+v-1} + \Lambda_2 X_l^{j+v-2} + \dots + \Lambda_v X_l^j) &= 0, \quad l=1,2,3,\dots,v \\ \Rightarrow \sum_{l=1}^v Y_l X_l^{j+v} + \Lambda_1 \sum_{l=1}^v Y_l X_l^{j+v-1} + \Lambda_2 \sum_{l=1}^v Y_l X_l^{j+v-2} + \dots + \Lambda_v \sum_{l=1}^v Y_l X_l^j &= 0 \\ \Rightarrow \Lambda_1 S_{j+v-1} + \Lambda_2 S_{j+v-2} + \Lambda_3 S_{j+v-3} + \dots + \Lambda_v S_j &= -S_{j+v}, \quad j=1,2,3,\dots,v \\ (\because S_j = Y_1 X_1^j + Y_2 X_2^j + Y_3 X_3^j + \dots + Y_v X_v^j, \quad j=1,2,3,\dots,2t)\end{aligned}\quad (2.18)$$

根據公式(2.18)，我們可以把它用寫成用矩陣的方法來表示：

$$\begin{bmatrix} S_1 & S_2 & \cdots & \cdots & S_v \\ S_2 & S_3 & \cdots & \cdots & S_{v+1} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ S_v & S_{v+1} & \cdots & \cdots & S_{2v-1} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \vdots \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ \vdots \\ \vdots \\ -S_{2v} \end{bmatrix} \quad (2.19)$$

因為一開始解碼的過程中，我們並不知道有幾個錯誤 v ，因此首先假設錯誤的個數剛好等於里德所羅門碼可以更正的錯誤數量($v=t$)，進而計算

$$\begin{bmatrix} S_1 & S_2 & \cdots & \cdots & S_v \\ S_2 & S_3 & \cdots & \cdots & S_{v+1} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ S_v & S_{v+1} & \cdots & \cdots & S_{2v-1} \end{bmatrix} \text{ 的行列式，若行列式} \neq 0 \text{，則方程式有解，表示} v \text{ 為里}$$

德所羅門碼可以更正的錯誤數量 t ，若行列式 $=0$ ，則就逐次把 v 減1，直到行列式 $\neq 0$ 才停止，此時 v 就是實際錯誤的個數了，而得到正確實際錯誤的個數 v 後，我們利用公式(2.19)就可得到錯誤位置多項式(error locator polynomial)，所以整個里德所羅門碼解碼過程就結束了。

由於前述的里德所羅門碼解碼演算法運算過程過於複雜，必須要運用到反矩陣的運算，所以比較沒效率，速度也比較減慢，於是這裡另外介紹一個疊代演算法可以改進此問題，稱為 Berlekamp-Massey Algorithm，以下為此演算法的詳細介紹。

根據公式(2.18)我們做一下代數變換，令 $j+v=j'$ 代入公式(2.18)之後可以寫出下式：

$$\begin{aligned} & \Lambda_1 S_{j+v-1} + \Lambda_2 S_{j+v-2} + \Lambda_3 S_{j+v-3} + \cdots + \Lambda_v S_j = -S_{j+v}, \quad j=1, 2, 3, \dots, v \\ \Rightarrow & \Lambda_1 S_{j'-1} + \Lambda_2 S_{j'-2} + \Lambda_3 S_{j'-3} + \cdots + \Lambda_v S_{j'-v} = -S_{j'}, \quad j'=v+1, v+2, v+3, \dots, 2v \\ \Rightarrow & S_{j'} = -\sum_{i=1}^v \Lambda_i S_{j'-i}, \quad j'=v+1, v+2, v+3, \dots, 2v \\ \Rightarrow & S_j = -\sum_{i=1}^v \Lambda_i S_{j-i}, \quad j=v+1, v+2, v+3, \dots, 2v \end{aligned} \quad (2.20)$$

因為由上式可以知道 $S_j = -\sum_{i=1}^v \Lambda_i S_{j-i}$, $j = v+1, v+2, v+3, \dots, 2v$ 是屬於疊代形式，所以在電路實現可以表示成線性迴授移位暫存器 (Linear Feedback Shift Register)，如圖 2.1 所示。

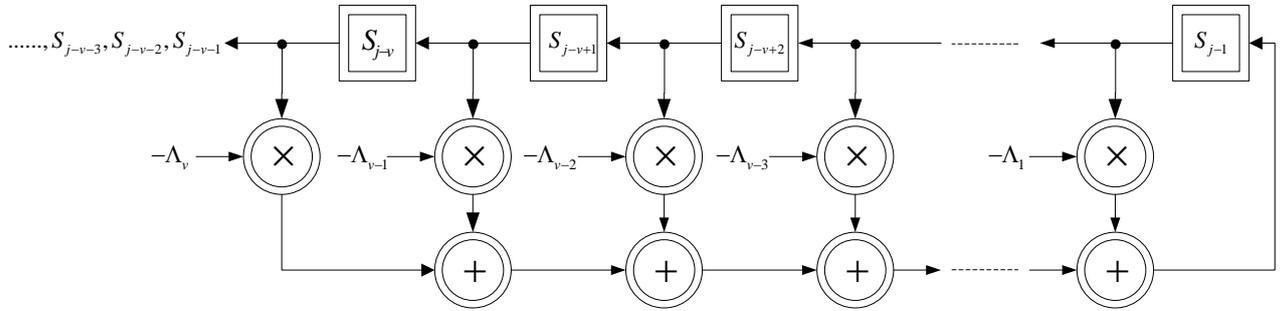


圖 2.1 Linear Feedback Shift Register 架構圖

所以原先需要求錯誤位置多項式之係數的問題，我們可以將其轉變成找出線性迴授移位暫存器的最短長度，因而使其輸出的前 $2t$ 個線性迴授移位暫存器的輸出就是病徵函數 $S_1, S_2, S_3, \dots, S_{2t}$ ，此時乘法器所乘上的值 (tap) 就是我們所要的錯誤位置多項式 $\Lambda(x)$ 。

在Berlekamp-Massey 演算法有五個參數定義，分別為連接多項式(connection polynomial) $\Lambda^{(k)}(x)$ 、更正多項式(correction polynomial) $T(x)$ 、差異值(discrepancy) $\Delta^{(k)}$ 、線性迴授移位暫存器的長度 L 、指標變數(indexing variable) k 。

Berlekamp-Massey 演算法的解碼方式就是使得 $\Lambda^{(k)}(x) = \Lambda_k x^k + \Lambda_{k-1} x^{k-1} + \Lambda_{k-2} x^{k-2} + \dots + \Lambda_1 x + 1$ 成為一個長度為 k 的的連接多項式，其係數為 k 個線性迴授移位暫存器的值(tap)。

我們開始要找第一個 $\Lambda^{(1)}(x)$ ，則在線性迴授移位暫存器相對應的輸出即為第一個病徵函數 S_1 。再來線性迴授移位暫存器的第二個輸出拿來和第二個病徵函數 S_2 做比較，如果這兩者有差異值 $\Delta^{(1)}$ ，就用這個差異值來建立一個更新的連接多項式；相反的，如果這兩者中間沒有差異，則連接多項式不用更新，且繼續拿此連接多項式去產生線性迴授移位暫存器的第三個輸出值，再與第三個病徵函數

S_3 做比較。如此重複上述步驟運算，直到線性迴授移位暫存器的輸出已經產生 $2t$ 個病徵函數 $S_1, S_2, S_3, \dots, S_{2t}$ 。因此 Berlekamp-Massey 演算法運算過 $2t$ 次後就可以得到錯誤位置多項式 $\Lambda(x)$ 了，用 Berlekamp-Massey 演算法可以利用遞迴的方法使得解碼更有效率。

以下為詳細的 Berlekamp-Massey 演算法過程：

第 1 步：從接收到的訊號算出病徵函數 $S_1, S_2, S_3, \dots, S_{2t}$ 。

第 2 步：初始化演算法中的各項變數， $k=0$ ， $\Lambda^{(0)}(x)=1$ ， $L=0$ ， $T(x)=x$ 。

第 3 步： $k=k+1$ ，由前面 $k-1$ 的連接多項式 $\Lambda^{(k-1)}(x)$ 來計算差異值 $\Delta^{(k)}$ 。

$$\Delta^{(k)} = S_k - \sum_{i=1}^L \Lambda_i^{(k-1)} S_{k-i}$$

第 4 步：若 $\Delta^{(k)}=0$ ，則表示不用更新連接多項式，直接進行第 8 步。

第 5 步：更新連接多項式 $\Lambda^{(k)}(x) = \Lambda^{(k-1)}(x) - \Delta^{(k)}T(x)$

第 6 步：若 $k \leq 2L$ ，則，直接進行第 8 步。

第 7 步： $L = k - L$ ， $T(x) = \frac{\Lambda^{(k-1)}(x)}{\Delta^{(k)}}$

第 8 步： $T(x) = x \cdot T(x)$

第 9 步：若 $k < 2t$ ，則直接進行第 3 步。

第 10 步：求出 $\Lambda(x) = \Lambda^{(2t)}(x)$ 的根，這樣就可以得到錯誤的位置及錯誤的大小了。

第三章

3.1 低密度同位檢查碼介紹

低密度同位檢查碼 LDPC code(Low Density Parity Check code)原本年由 Robert Gallager 在 1962 年所發明的，這是一種使用大段長的線性段碼，由於當時電腦能力不足以處理複雜性計算，而且由於 VLSI 技術尚未成熟，因此使人們淡忘許久，直到 1995 年由 Mackay 與 Neal 重新發展出 Tanner Graph 的解碼方式，使得解碼時使用兩個狀態的同位檢查格子(parity check trellis)，所以解碼器容易實現，且 VLSI 技術的快速發展使得 LDPC code 又逐漸的被人們所廣為討論。



3.2 編碼器

低密度同位檢查碼的編碼方式為 G (產生矩陣) 乘上一個 U (訊息向量) 而產生 V (碼字向量)。由於低密度同位檢查碼須符合 V (編碼向量) 乘上 H (同位檢查矩陣) 後等於零的規則，即是 $HV^T = 0$ ，所以可由下列方法得到 V (碼字向量)。假設低密度同位檢查碼以 (N, K) 的線性區塊碼來表示，其中有 $N-K$ 個檢查位元， K 個位元與 U (訊息向量) 相同。 H (同位檢查矩陣) 的大小為 $M \times M$ 。

$$HV^T = 0 \quad (3.1)$$

$$H = [A | B] \quad (3.2)$$

$$V = [C | U] = UG \quad (3.3)$$

其中 A 表示為 $M \times M$ 的矩陣大小， B 表示為 $M \times (N-M)$ 的矩陣大小。 C 表示為 $1 \times (N-K)$ 的矩陣大小， U 表示為 $1 \times K$ 的矩陣大小。由(3.2)式及(3.3)式代入到(3.1)式。

即可得到


$$\begin{aligned} AC^T + BU^T &= 0 \\ \Rightarrow C^T &= A^{-1}BU^T \end{aligned} \quad (3.4)$$

$$\text{所以編碼向量 } V = [UB^T(A^{-1})^T | U] = UG \quad (3.5)$$

又因為 $UG = V = [UB^T(A^{-1})^T | U]$ 的關係式，可以求得產生矩陣 G 。

$$\text{所以產生矩陣 } G = [B^T(A^{-1})^T | I] \quad (3.6)$$

H (同位檢查矩陣) 的維度為 $M \times N$

G (產生矩陣) 的維度為 $K \times N$

U (訊息向量) 的維度為 $1 \times K$

其中低密度同位檢查碼的編碼率為 $\frac{K}{N} = \frac{N-M}{N}$ 。

以下舉一個例子可以更方便了解低密度同位檢查碼的編碼過程：

$$H = \left[\begin{array}{cccccc|cccc} 1 & 1 & 0 & 1 & 0 & 1 & \vdots & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & \vdots & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & \vdots & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & \vdots & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \vdots & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & \vdots & 1 & 1 & 1 & 1 \end{array} \right]$$

$\underbrace{\hspace{10em}}$
A

$\underbrace{\hspace{10em}}$
B

$$A^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad B^T = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$\Rightarrow B^T(A^{-1})^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

所以產生矩陣 $G = [B^T(A^{-1})^T | I] =$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

以上介紹低密度同位檢查碼的編碼方式，而其步驟可用下圖來表示：

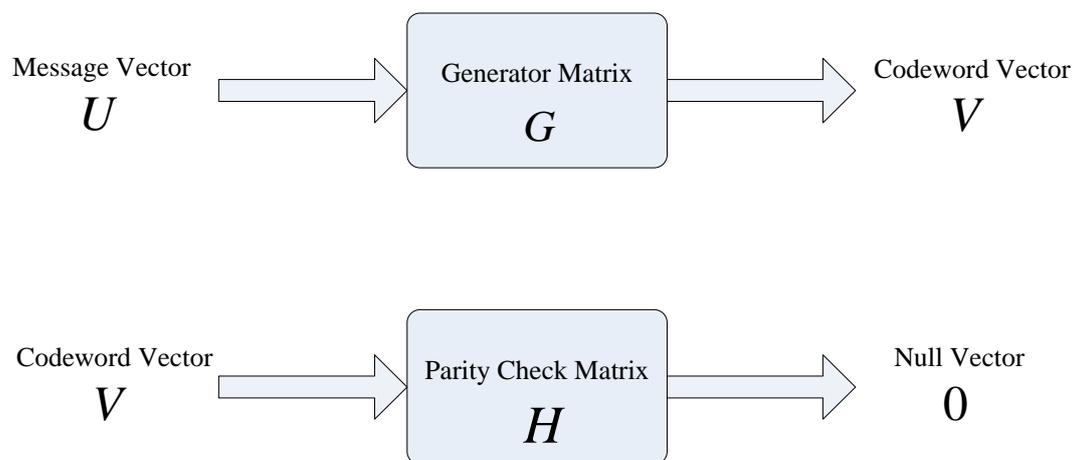


圖 3.1 低密度同位檢查碼編碼方塊圖



3.3 解碼器

3.3.1 Tanner Graph

在低密度同位檢查碼中，我們可以把 H (同位檢查矩陣) 分成兩個部份來看，其維度為 $M \times N$ 。這兩部份分別包含 check node 和 bit node。第一個部份是先看 H (同位檢查矩陣) 的列，把所有的列可以看成 check node，每一列看成一個 check node，也就是說總共有 M 個 check node，再來就是把行看成 bit node，也就是碼字(Codeword)的 bit 數，相當於有 N 個 bit node。

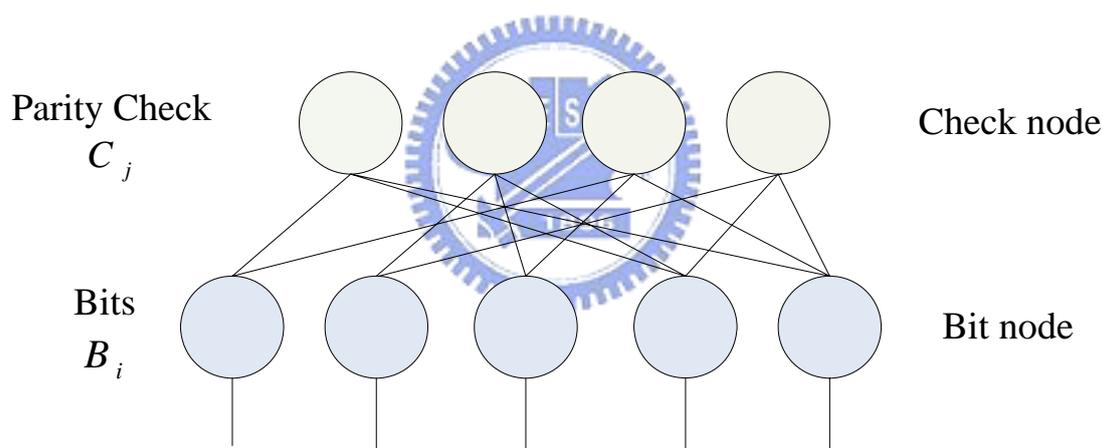


圖 3.2 Tanner Graph

我們以 3.2 節的 H (同位檢查矩陣) 為簡單的說明例子，因為低密度同位檢查碼解碼的過程要符合 $HV^T = 0$ ，所以每一個 check node 連到的 bit node 均要滿足 $HV^T = 0$ 的關係式， $V = (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10})$ 。

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \text{經由 } HV^T = 0 \text{ 的式子成立，也就是說 bit}$$

node 只有連到 check node 的元素是“1”的時候，因此可以用下面的方塊圖來簡化表示 $HV^T = 0$ 的關係。

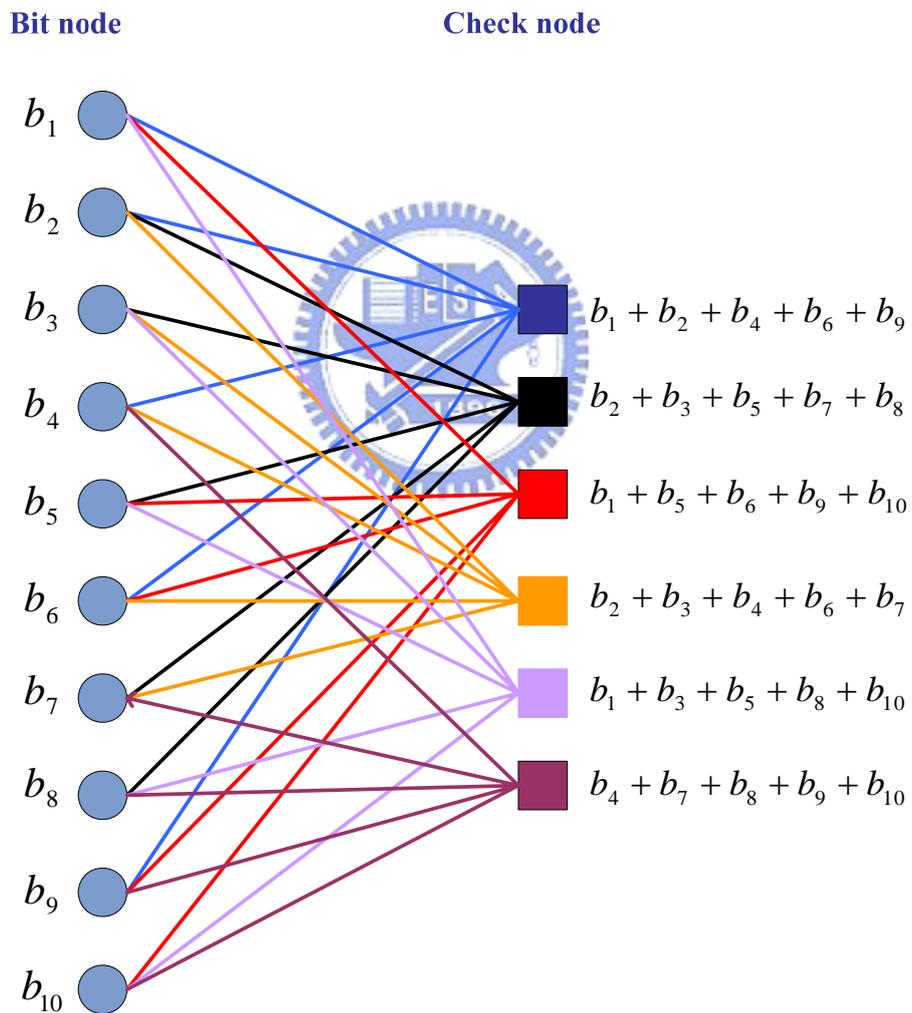


圖 3.3 $HV^T = 0$ 方塊圖

3.3.2 解碼的機率演示圖

由上一節的 Tanner Graph 可以知道低密度同位檢查碼的解碼過程是經由 bit node 及 check node 這兩端互相算出機率再丟給對方，應用到 Message Passing 的概念，本節將討論這些 Message 機率的算法。

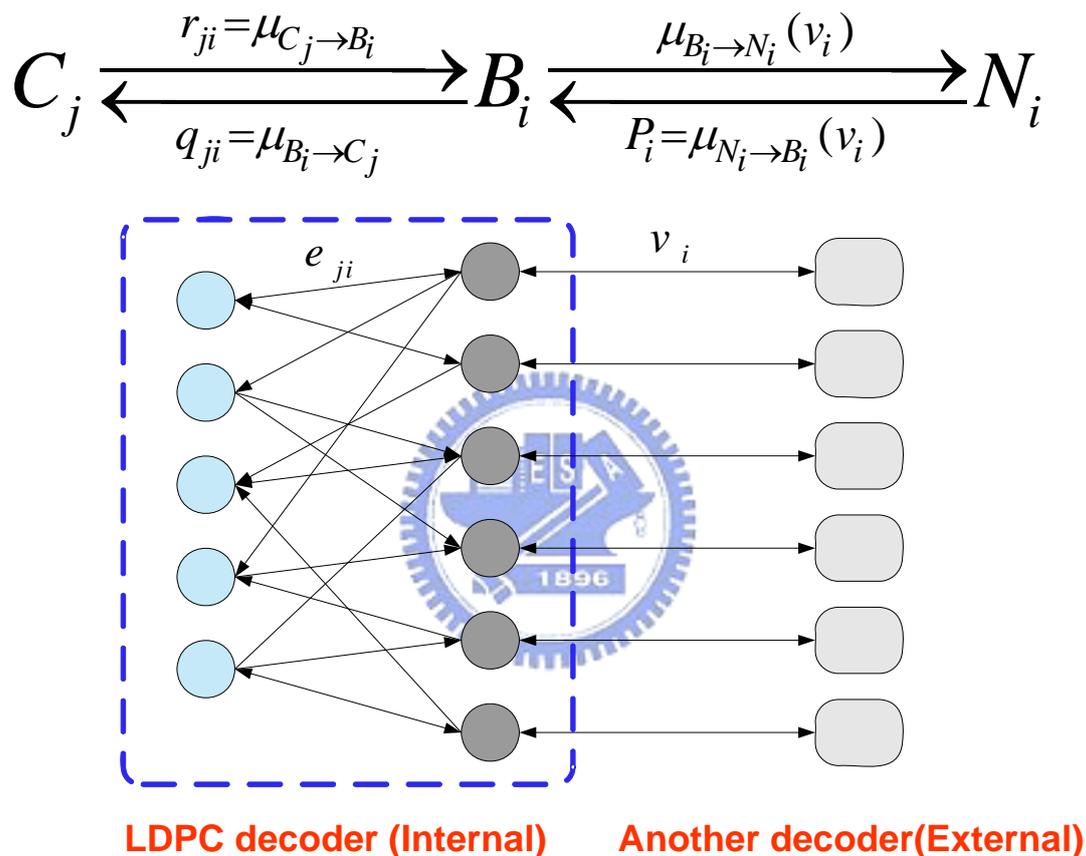


圖 3.4 解碼器的架構圖

在圖 3.4 中所表示，其中的 C_j 代表 check node(C)、 B_i 代表 bit node(B)， N_i 代表 another node(N)，這是另外一個 decoder 連接到 LDPC decoder 的 node。

3.3.3 Bit node 到 Check node 的機率算法

由圖 3.5 可知，假設 bit node(B) 與 $K+1$ 個 stage(check node)相連，且彼此是獨立(independent)，這些 $K+1$ 個 stage 分別表示成 $c_0, c_1, c_2, \dots, c_K$ ，而且 $c_0, c_1, c_2, \dots, c_K$ 屬於同一個 alphabet A ，根據上面的特性 bit node 的 constraint set 可以寫成下列的關係式：

$$S_B = \{(c_0, c_1, c_2, \dots, c_K) \mid c_0 = c_1 = c_2 = \dots = c_K\}$$

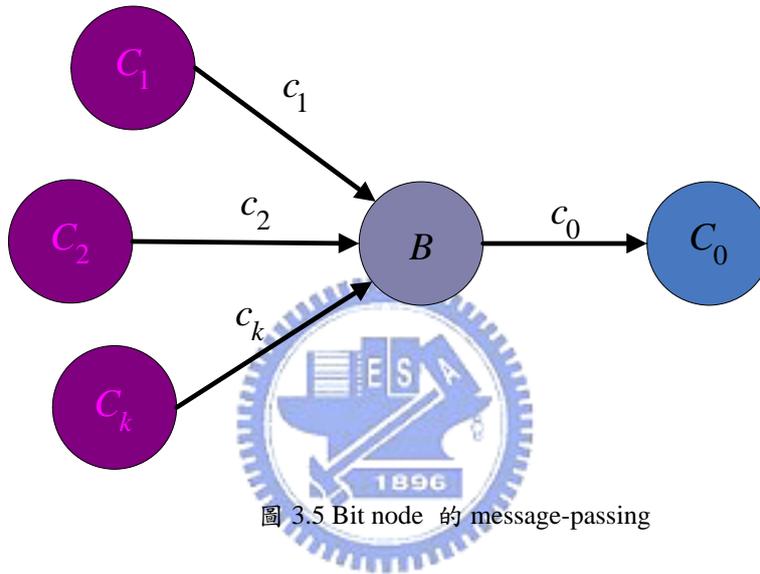


圖 3.5 Bit node 的 message-passing

假設從外部的 check node ($C_1, C_2, C_3, \dots, C_K$) 進到 bit node (B) 的機率為 $\mu_{C_i \rightarrow B}(c_i)$ for $i=1, 2, 3, \dots, K$ ，則把輸入到 bit node (B) 的所有 check node ($C_1, C_2, C_3, \dots, C_K$) 機率資訊運算之後可以得到一個輸出機率，以 $\mu_{B \rightarrow C_0}(c_0 = \xi)$ 來表示 bit node (B) 輸出到 check node (C_0) 的機率，所以整個關係可以寫成下式：

$$\begin{aligned} \mu_{B \rightarrow C_0}(c_0 = \xi) &= m_{c_0} \sum_{(c_0, c_1, \dots, c_K) \in S_B \sim \{c_0\}} \prod_{l=1}^K \mu_{C_l \rightarrow B}(c_l) \\ &= m_{c_0} \prod_{l=1}^K \mu_{C_l \rightarrow B}(c_l = \xi) \end{aligned} \quad (3.7)$$

其中 r_{c_0} 是一個正規化常數(normalization factor)

$$r_{c_0} = \left(\sum_{\zeta \in A} \prod_{l=1}^K \mu_{C_l \rightarrow B}(c_l = \zeta) \right)^{-1}$$

以上為 bit node (B) 到 check node (C_0) 的機率算法，接下來根據(3.7)式子，我們

可以用另外一種方法來求得機率，此方法為 Log-Likelihood ratio(LLR)，

$LLR = \log \frac{\text{"1"的機率}}{\text{"0"的機率}}$ ，所以(3.7)可以改寫成下式：

$$\begin{aligned} LLR_{B \rightarrow C_0(c_0)} &= \log \frac{\mu_{B \rightarrow C_0}(c_0 = 1)}{\mu_{B \rightarrow C_0}(c_0 = 0)} \\ &= \log \frac{\prod_{i=1}^K \mu_{C_i \rightarrow B}(c_i = 1)}{\prod_{i=1}^K \mu_{C_i \rightarrow B}(c_i = 0)} \\ &= \sum_{i=1}^K LLR_{C_i \rightarrow B}(c_i) \end{aligned}$$

如在 3.2 節所提到的，假設低密度同位檢查碼以(N,K)的線性區塊碼來表示，則 H (同位檢查矩陣)的維度為 $M \times N$ ， $K = N - M$ ，所以 bit node 的維度為 $1 \times N$ ，以 B_i 來表示， $i = 1, 2, 3, \dots, N$ 。check node 的維度為 $1 \times M$ ，以 C_j 來表示， $j = 1, 2, 3, \dots, M$ 。而 bit node (B_i) 跟 check node (C_j) 之間的 edge 以變數 e_{ji} 來表示，bit node (B_i) 跟 another decoder (N_i) 間的 edge 以變數 v_i 來表示，如圖 3.4 的架構圖所示。

LDPC decoder 的本質機率(intrinsic probability) 為 another decoder (N_i) 傳送到 bit node (B_i) 的機率資訊，以 $P_{LDPC}^{int}(v_i) = \mu_{N_i \rightarrow B_i}(v_i)$ 來表示。Message-passing 演算法由本質機率(intrinsic probability) $P_{LDPC}^{int}(v_i) = \mu_{N_i \rightarrow B_i}(v_i)$ 開始運算的，利用式子(3.7)我們可以求得 bit node (B_i) 到 check node (C_j) 的機率，如下列所示：

$$\text{"0"的機率} : \mu_{B_i \rightarrow C_j}(e_{ji} = 0) = m_{ji} \mu_{N_i \rightarrow B_i}(v_i = 0) \prod_{j \in M(i) \setminus \{j\}} \mu_{C_j \rightarrow B_i}(e_{ji} = 0) \quad (3.8)$$

$$\text{"1"的機率} : \mu_{B_i \rightarrow C_j}(e_{ji} = 1) = m_{ji} \mu_{N_i \rightarrow B_i}(v_i = 1) \prod_{j \in M(i) \setminus \{j\}} \mu_{C_j \rightarrow B_i}(e_{ji} = 1) \quad (3.9)$$

其中 $M(i)$ 表示在 H (同位檢查矩陣)第 i 行(column) 裡面那些是"1"的列(row) 的位置的集合，而 $M(i) \setminus \{j\}$ 表示在 $M(i)$ 的集合裡把第 j 個元素扣除， m_{ji} 是指一個正規化的常數。

3.3.4 Check node 到 Bit node 的機率算法

由圖 4.6 可知，假設 check node(C) 與 $K+1$ 個 stage(bit node)相連，且彼此是獨立(independent)，這些 $K+1$ 個 stage 分別表示成 $b_0, b_1, b_2, \dots, b_K$ ，而且 $b_0, b_1, b_2, \dots, b_K$ 屬於同一個 alphabet A，則 check node 到 bit node 的機率資訊公式推導如下：

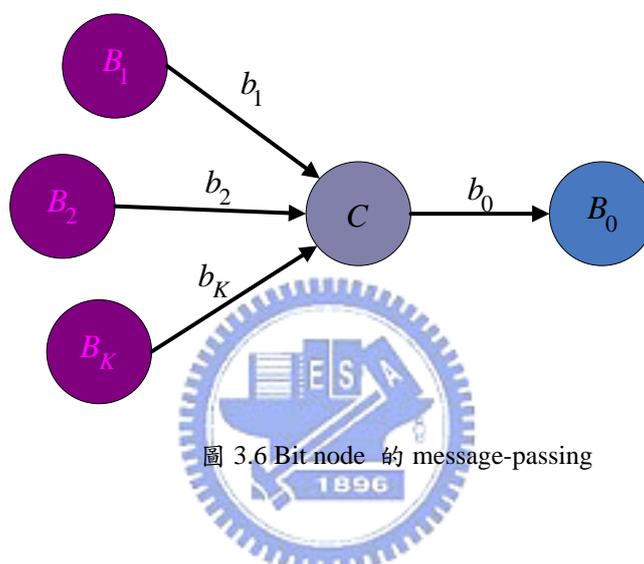


圖 3.6 Bit node 的 message-passing

在圖 3.6 中，輸入到 check node(C)的 K 個機率資訊中，假設為本質機率 (intrinsic probability)，以 $b_i, i=0,1,2,\dots,K$ 用來表示 bit node 和 check node 相連接的 "0" 或 "1" 機率，由 bit node 連接到 check node 的機率稱為本質機率 (intrinsic probability)，以 $\mu_{B_i \rightarrow C}(b_i)$ 來表示。

$$\text{本質機率"0"的機率：} \mu_{B_i \rightarrow C}(b_i = 0) = 1 - p_i \quad (3.10)$$

$$\text{本質機率"1"的機率：} \mu_{B_i \rightarrow C}(b_i = 1) = p_i \quad (3.11)$$

而從 check node 傳送到 bit node 的機率資訊稱為外質機率 (extrinsic probability)，以 $\mu_{C \rightarrow B_0}(b_0)$ 來表示。

因為從 check node 傳送至 bit node 的機率資訊 ($\mu_{C \rightarrow B_0}(b_0)$) 是由 K 個機率資訊 ($\mu_{B_i \rightarrow C}(b_i), i=1,2,3,\dots,K$) 來決定，所以傳送至 bit node 的機率資訊 $\mu_{C \rightarrow B_0}(b_0)$ 是 "0" 或是 "1" 的機率為這 K 個 ($\mu_{B_i \rightarrow C}(b_i), i=1,2,3,\dots,K$) 的 XOR 組成，若 ($\mu_{B_i \rightarrow C}(b_i), i=1,2,3,\dots,K$) 全部 XOR 起來是 "0" 才是傳送到 check node 的

“0”的機率。若 XOR 起來是“1”的情況，則是傳送到 check node 的“1”的機率，有這下列的關係式。

$$\text{外質機率"0"的機率：}\mu_{C \rightarrow B_0}(b_0 = 0) = P(b_1 \oplus b_2 \oplus b_3 \cdots \oplus b_K = 0) \quad (3.12)$$

$$\text{外質機率"1"的機率：}\mu_{C \rightarrow B_0}(b_0 = 1) = P(b_1 \oplus b_2 \oplus b_3 \cdots \oplus b_K = 1) \quad (3.13)$$

為了要證明上面這兩個式子(3.12)及(3.13)的成立，我們需要下列的假設與推導過程，因為我們推導的式子從 bit node 傳送到 check node 的機率資訊有 K 個，我們先假設只有 2 個 stage(B_1 和 B_2)時的情況，首先證明只有 2 個 stage(B_1 和 B_2)時的式子會成立，再由數學歸納法證明有 K 個 stage 時仍然會成立。

若只考慮 2 個 stage(B_1 和 B_2)時，則從 bit node 傳送至 check node 的機率資訊($\mu_{C \rightarrow B_0}(b_0)$) 只是由 b_1 和 b_2 的 XOR(\oplus) 來決定，若 $b_1 \oplus b_2 = 0$ 則表示是“0”的機率，若 $b_1 \oplus b_2 = 1$ 則表示是“1”的機率，因為要 XOR 起來是“0”則 b_1 和 b_2 均要為“0”或均為“1”，若 XOR 起來是“1”則 $b_1 = 0, b_2 = 1$ 或 $b_1 = 1, b_2 = 0$ ，要如下列兩個式子所表示：

$$\mu_{C \rightarrow B_0}(b_0 = 0) = P(b_1 \oplus b_2 = 0) = p_1 p_2 + (1 - p_1)(1 - p_2) \quad (3.14)$$

$$\mu_{C \rightarrow B_0}(b_0 = 1) = P(b_1 \oplus b_2 = 1) = p_1(1 - p_2) + p_2(1 - p_1) \quad (3.15)$$

其中 p_1 表示第一個 stage 傳送到 check node 是“1”機率資訊 ($\mu_{B_1 \rightarrow C}(b_1 = 1) = p_1$)，而 p_2 表示第二個 stage 傳送到 check node 是“1”機率資訊 ($\mu_{B_2 \rightarrow C}(b_2 = 1) = p_2$)。 $1 - p_1$ 代表第一個 stage 傳送到 check node 是“0”機率資訊， $1 - p_2$ 代表第二個 stage 傳送到 check node 是“0”機率資訊。

我們針對(3.14)的式子來化簡，首先把(3.14)式子乘以 2 再減去 1，即可化簡得到下列的式子，假設 2 個 stage(B_1 和 B_2)時的式子已經成立了。

$$\mu_{C \rightarrow B_0}(b_0 = 0) = 2P(b_1 \oplus b_2 = 0) - 1 = (1 - 2p_1)(1 - 2p_2) \quad (3.16)$$

再來我們要把一開始假設只有 2 個 stage(B_1 和 B_2)推廣到有 K 個 stage($\mu_{B_i \rightarrow C}(b_i)$, $i = 1, 2, 3, \dots, K$)的情況，如(3.18)式子所示。首先假設 $K - 1$ 個 stage 成立，如(3.17)所示，再由數學歸納法可以知道 K 個 stage ($\mu_{B_i \rightarrow C}(b_i)$ ，

$i=1,2,3,\dots,K$)的公式也是成立的。

$$M_{K-1} = P(b_1 \oplus b_2 \oplus b_3 \oplus \dots \oplus b_{K-1} = 0) \quad (3.17)$$

$$M_K = P(b_1 \oplus b_2 \oplus b_3 \oplus \dots \oplus b_K = 0) \quad (3.18)$$

$$p_K = P^{int}(b_K = 1) \quad (3.19)$$

其中式子(3.19)表示由第 K 個 stage(B_K)傳送到 check node 的機率資訊 ($\mu_{B_K \rightarrow C}(b_K) = p_K = P^{int}(b_K)$)。我們由式子(3.16)可以把他推廣到假設 $K-1$ 個 stage 時會成立，如下列式子所示：

$$2M_{K-1} - 1 = \prod_{i=1}^{K-1} (1 - 2p_i) \quad (3.20)$$

因為考慮 K 個 stage 時，是由前面總共 $K-1$ 個 stage 和第 K 個 stage 的機率資訊來推導，所以如 (3.18) 式子所示， M_K 是由 XOR 起來的機率是 "0" ($b_1 \oplus b_2 \oplus b_3 \oplus \dots \oplus b_K = 0$) 來決定，所以有兩種情況，第一種情形是， $b_1 \oplus b_2 \oplus b_3 \oplus \dots \oplus b_{K-1} = 0$ 且 $b_K = 0$ 。第二種情形是 $b_1 \oplus b_2 \oplus b_3 \oplus \dots \oplus b_{K-1} = 1$ 且 $b_K = 1$ 。所以把這兩種情況寫成一個式子時，可以得到下列的機率資訊：

$$\begin{aligned} \Rightarrow M_K &= (1 - p_K)M_{K-1} + p_K(1 - M_{K-1}) \\ &= (1 - 2p_K)M_{K-1} + p_K \\ \Rightarrow 2M_K - 1 &= 2(1 - 2p_K)M_{K-1} + 2p_K - 1 \\ &= 2M_{K-1} - 4p_K M_{K-1} + 2p_K - 1 \\ &= (1 - 2p_K)(2M_{K-1} - 1) \\ &= (1 - 2p_K) \prod_{i=1}^{K-1} (1 - 2p_i) \\ &= \prod_{i=1}^K (1 - 2p_i) \end{aligned} \quad (3.21)$$

以上證明由數學歸納法得證，有 K 個 stage 的機率資訊公式 仍然適用，所以由式子(3.21)可以移項得到下式：

$$M_K = \frac{1 + \prod_{i=1}^K (1 - 2p_i)}{2}$$

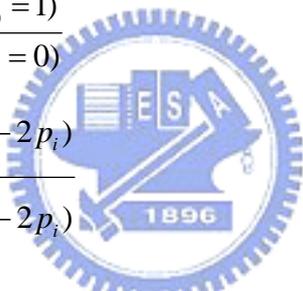
所以 "0" 和 "1" 的機率資訊分別為：

$$\text{"0"的機率} : \mu_{C \rightarrow B_0}(b_0 = 0) = P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 0) = \frac{1 + \prod_{i=1}^K (1 - 2p_i)}{2} \quad (3.22)$$

$$\text{"1"的機率} : \mu_{C \rightarrow B_0}(b_0 = 1) = P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 1) = \frac{1 - \prod_{i=1}^K (1 - 2p_i)}{2} \quad (3.23)$$

以上的公式推導為 check node(C) 到 bit node (B_0)的機率算法，接下來根據 (3.22)及(3.23)式子可以用另外一種方法來求得機率，此方法為 Log-Likelihood

ratio(LLR)， $LLR = \log \frac{\text{"1"的機率}}{\text{"0"的機率}}$ ，所以可以寫出下列的式子：

$$\begin{aligned} \Rightarrow LLR_{C \rightarrow B_0}(b_0) &= \log \frac{\mu_{C \rightarrow B_0}(b_0 = 1)}{\mu_{C \rightarrow B_0}(b_0 = 0)} \\ &= \log \frac{1 - \prod_{i=1}^K (1 - 2p_i)}{1 + \prod_{i=1}^K (1 - 2p_i)} \end{aligned} \quad (3.24)$$


再來運用一些代數的特性，可以把式子(3.24)推導下去，直到式子(3.24)和 bit node 傳送到 check node 的機率資訊($\mu_{B_i \rightarrow C}(b_i)$)有關係。

首先我們由前面的介紹已經知道其中 p_i 表示第 i 個 stage 傳送到 check node 是 "1" 機率資訊 ($\mu_{B_i \rightarrow C}(b_i = 1) = p_i$)，因此第 i 個 stage 傳送到 check node 是 "0" 機率資訊 ($\mu_{B_i \rightarrow C}(b_i = 0) = 1 - p_i$)，所以 Log-Likelihood ratio(LLR) 寫成下式：

$$\begin{aligned} \Rightarrow LLR(p_i) &= \log \frac{p_i}{1 - p_i} \\ \Rightarrow \frac{p_i}{1 - p_i} &= e^{LLR(p_i)} \end{aligned} \quad (3.25)$$

$$\Rightarrow 1 - 2p_i = -\tanh\left(\frac{1}{2} LLR(p_i)\right), \quad \because \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{\frac{1}{2}(e^x - e^{-x})}{\frac{1}{2}(e^x + e^{-x})}$$

$$\begin{aligned} \because y &= \tanh\left(\frac{x}{2}\right) = \frac{e^x - 1}{e^x + 1} \\ \Rightarrow 2 \tanh^{-1}(y) &= \log \frac{1+y}{1-y} \end{aligned} \quad (3.26)$$

所以我們可以利用式子(3.25)及式子(3.26)代入到式子(3.24)即可以導出下列公式：

$$\begin{aligned} LLR_{C \rightarrow B_0}(b_0) &= \log \frac{1 - \prod_{i=1}^K (1 - 2p_i)}{1 + \prod_{i=1}^K (1 - 2p_i)} \\ &= \log \left(\frac{1 - (-1)^K \prod_{i=1}^K \tanh\left(\frac{1}{2} LLR(p_i)\right)}{1 + (-1)^K \prod_{i=1}^K \tanh\left(\frac{1}{2} LLR(p_i)\right)} \right) \\ &= -2 \tanh^{-1} \left((-1)^K \prod_{i=1}^K \tanh\left(\frac{1}{2} LLR(p_i)\right) \right) \\ &= 2(-1)^{K+1} \tanh^{-1} \left(\prod_{i=1}^K \tanh\left(\frac{1}{2} LLR(p_i)\right) \right) \\ &= 2(-1)^{K+1} \tanh^{-1} \left(\prod_{i=1}^K \tanh\left(\frac{1}{2} LLR_{B_i \rightarrow C}(b_i)\right) \right) \end{aligned} \quad (3.27)$$

所以我們由式子(3.22)及式子(3.23)可以知道由 check node 傳送到 bit node 的機率資訊為：

$$\text{"0"的機率} : \mu_{C \rightarrow B_0}(b_0 = 0) = P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 0) = \frac{1 + \prod_{i=1}^K (1 - 2p_i)}{2} \quad (3.28)$$

$$\text{"1"的機率} : \mu_{C \rightarrow B_0}(b_0 = 1) = P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 1) = \frac{1 - \prod_{i=1}^K (1 - 2p_i)}{2} \quad (3.29)$$

於是我們把這兩個分別表示傳送“0”或“1”的公式套用到 LDPC decoder 中，則 check node 傳送到 bit node 的機率資訊為：

$$\mu_{C_j \rightarrow B_i}(b_0 = 0) = \frac{1}{2} \left(1 + \prod_{i \in L(j) \setminus \{i\}} (1 - 2\mu_{B_i \rightarrow C_j}(x_i = 1)) \right) \quad (3.30)$$

$$\mu_{C_j \rightarrow B_i}(b_0 = 1) = \frac{1}{2} \left(1 - \prod_{i \in L(j) \setminus \{i\}} (1 - 2\mu_{B_i \rightarrow C_j}(x_i = 1)) \right) \quad (3.31)$$

其中在式子(3.30)和(3.31)中 $L(j)$ 表示在 H (同位檢查矩陣) 第 j 列(row) 裡面那些是“1”的行(column)的位置的集合，而 $L(j) \setminus \{i\}$ 表示在 $L(j)$ 的集合裡把第 i 個元素扣除。



3.3.5 LDPC 解碼時的 Update Equation

以上介紹過 LDPC 解碼過程中機率訊息的傳遞，接下來的部份要推導 LDPC 解碼時的流程步驟，首先我們可以把圖 3.7 的方塊圖拆解成兩個部份，分別算出 check node 傳送到 bit node 的機率資訊，及 bit node 傳送到 check node 的機率資訊。首先定義本質機率(intrinsic probability) $\mu_{N_i \rightarrow B_i}$ 為 another decoder 的 node 傳送到 bit node 的機率資訊。後置機率(posterior probability) $\mu_{B_i \rightarrow C_j}$ 為 bit node 傳送到 check node 的機率資訊。而 $\mu_{C_j \rightarrow B_i}$ 為 check node 傳送到 bit node 的機率資訊。

$$p_i^b = \mu_{N_i \rightarrow B_i}(v_i = b) \quad (3.32)$$

$$q_{ji}^b = \mu_{B_i \rightarrow C_j}(e_{ji} = b) \quad (3.33)$$

$$r_{ji}^b = \mu_{C_j \rightarrow B_i}(e_{ji} = b) \quad (3.34)$$

其中 b 表示為 "0" 或者為 "1"

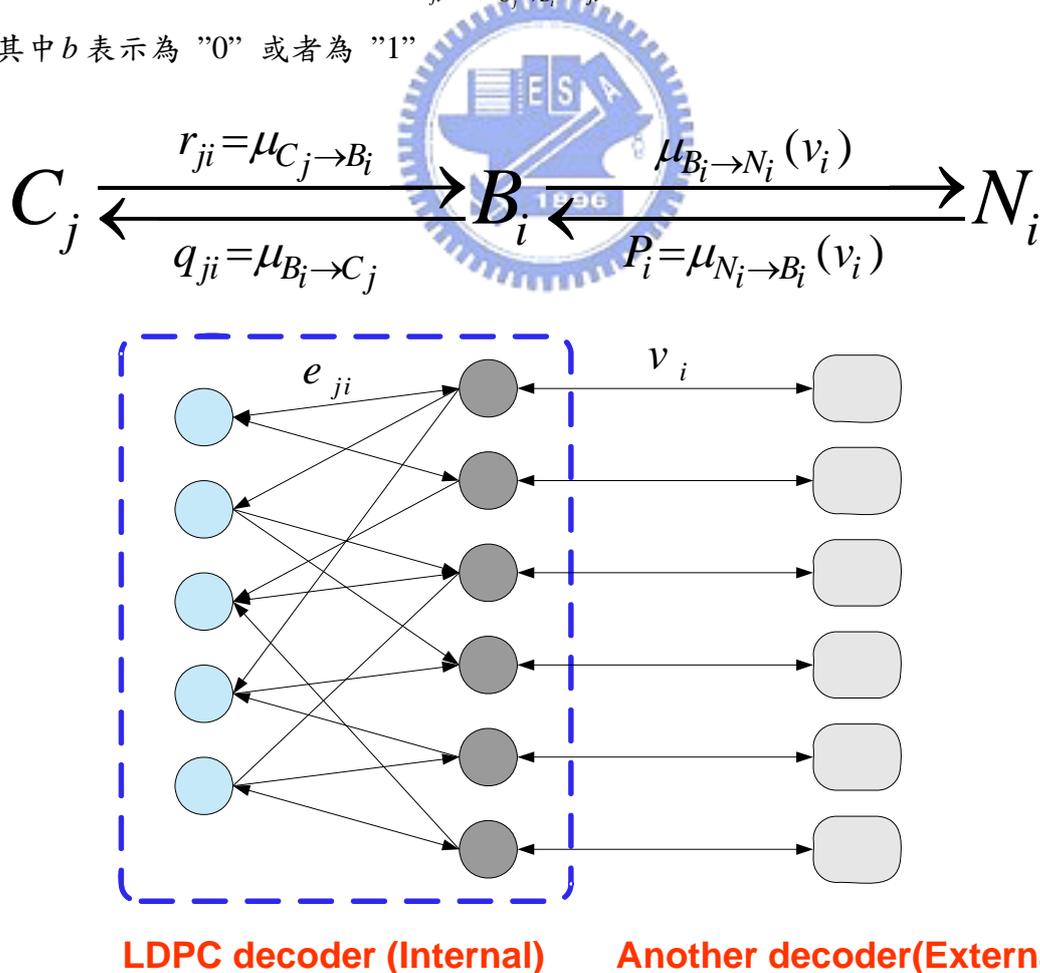


圖 3.7 LDPC decoder

由圖 3.7 的方塊圖所分解成的兩個部份，分別以圖 3.8 及圖 3.9 來表示，其中圖 3.8 代表由 bit node 傳送到 check node 的機率資訊，而圖 3.9 代表由 check node 傳送到 bit node 的機率資訊。

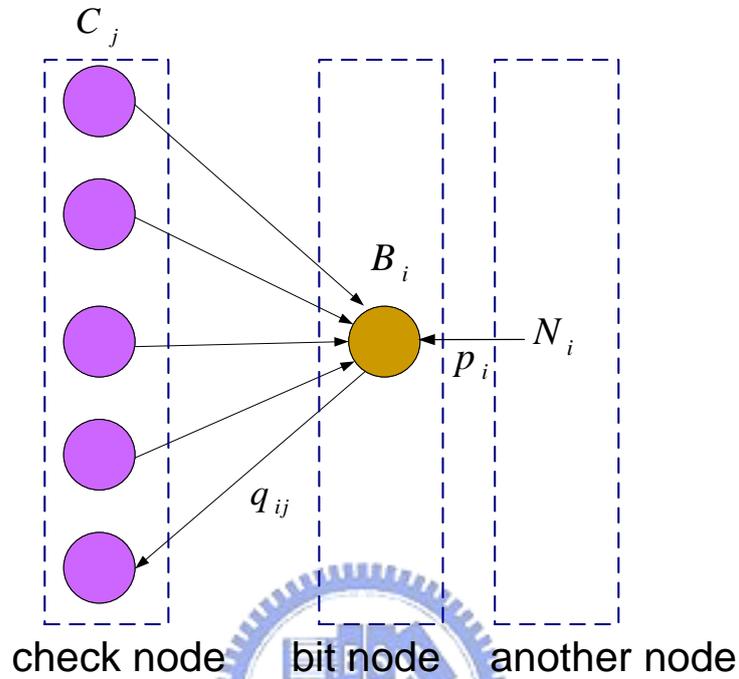


圖 3.8 bit node 傳送到 check node 的機率資訊

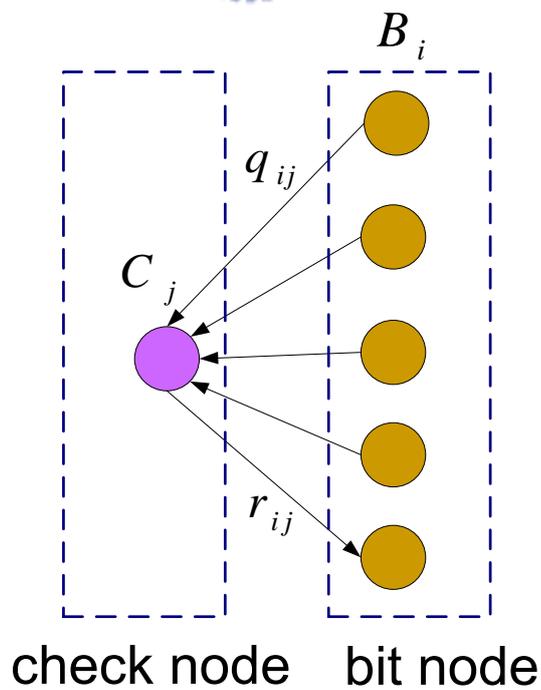


圖 3.9 check node 傳送到 bit node 的機率資訊

我們把式子(3.32)、(3.33)、(3.34) 的代數變換代入 3.3.3 節中所導出的式子(3.8)和(3.9)，即可得到 bit node 傳送到 check node 的機率資訊，如下列兩個式子所表示：

$$\begin{aligned}
 \text{"0"的機率：} \quad \mu_{B_i \rightarrow C_j}(e_{ji} = 0) &= m_{ji} \mu_{N_i \rightarrow B_i}(v_i = 0) \prod_{j' \in M(i) \setminus \{j\}} \mu_{C_j' \rightarrow B_i}(e_{ji'} = 0) \\
 \Rightarrow q_{ji}^0 &= m_{ji} \cdot p_i^0 \prod_{j' \in M(i) \setminus \{j\}} r_{ji'}^0
 \end{aligned} \tag{3.35}$$

$$\begin{aligned}
 \text{"1"的機率：} \quad \mu_{B_i \rightarrow C_j}(e_{ji} = 1) &= m_{ji} \mu_{N_i \rightarrow B_i}(v_i = 1) \prod_{j' \in M(i) \setminus \{j\}} \mu_{C_j' \rightarrow B_i}(e_{ji'} = 1) \\
 \Rightarrow q_{ji}^1 &= m_{ji} \cdot p_i^1 \prod_{j' \in M(i) \setminus \{j\}} r_{ji'}^1
 \end{aligned} \tag{3.36}$$

接下來如同上面的步驟，我們把式子(3.32)、(3.33)、(3.34) 的代數變換代入 3.3.4 節中所導出的式子(3.22)和(3.23)，即可得到 check node 傳送到 bit node 的機率資訊，如下列兩個式子所表示：

$$\begin{aligned}
 \text{"0"的機率：} \quad \mu_{C \rightarrow B_0}(b_0 = 0) &= P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 0) = \frac{1 + \prod_{i=1}^K (1 - 2p_i)}{2} \\
 \Rightarrow r_{ji}^0 &= \frac{1}{2} \left(1 + \prod_{i' \in L(j) \setminus \{i\}} \delta_{q_{ji'}} \right)
 \end{aligned} \tag{3.37}$$

$$\begin{aligned}
 \text{"1"的機率：} \quad \mu_{C \rightarrow B_0}(b_0 = 1) &= P(b_1 \oplus b_2 \oplus \dots \oplus b_K = 1) = \frac{1 - \prod_{i=1}^K (1 - 2p_i)}{2} \\
 \Rightarrow r_{ji}^1 &= \frac{1}{2} \left(1 - \prod_{i' \in L(j) \setminus \{i\}} \delta_{q_{ji'}} \right)
 \end{aligned} \tag{3.38}$$

上列兩式中的 $\delta_{q_{ji}}$ 表示為 q_{ji} 為 "0" 的機率減去 q_{ji} 為 "1" 的機率，
 $\delta_{q_{ji}} = q_{ji}^0 - q_{ji}^1 = 1 - 2q_{ji}^1 \quad \because q_{ji}^0 + q_{ji}^1 = 1$

有了 bit node 傳送到 check node 的機率資訊及 check node 傳送到 bit node 的機率資訊後，因為這兩個機率資訊有著互相相連接的關係(互為另外一個的變數)，所以我們可以藉由這兩個資訊一直更新彼此的機率，如式子(3.39)和(3.40)，用來增加解碼的可靠度，這也是 LDPC decoder 的解碼過程，稍後我們會詳細介紹。

$$\begin{cases} q_{ji}^0 = m_{ji} \cdot p_i^0 \prod_{j \in M(i) \setminus \{j\}} r_{ji}^0 \\ q_{ji}^1 = m_{ji} \cdot p_i^1 \prod_{j \in M(i) \setminus \{j\}} r_{ji}^1 \end{cases} \quad (3.39)$$

$$\begin{cases} r_{ji}^0 = \frac{1}{2} \left(1 + \prod_{i \in L(j) \setminus \{i\}} \delta_{q_{ji}} \right) \\ r_{ji}^1 = \frac{1}{2} \left(1 - \prod_{i \in L(j) \setminus \{i\}} \delta_{q_{ji}} \right) \end{cases} \quad (3.40)$$

以上所介紹的均是機率(probability)的算法，接下來要推導 Log-Likelihood ratio(LLR) 的方法，之前我們有定義本質機率 $p_i^b = \mu_{N_i \rightarrow B_i}(v_i = b)$ 為 another decoder 的 node 傳送到 bit node 的機率資訊。後置機率(posterior probability) $q_{ji}^b = \mu_{B_i \rightarrow C_j}(e_{ji} = b)$ 為 bit node 傳送到 check node 的機率資訊。其中 b 表示為 "0" 或者為 "1"。

在 3.3.3 節及 3.3.4 節中，我們有分別導出 bit node 傳送到 check node 的機率資訊如下列式子所示：

$$\begin{aligned} \mu_{B_i \rightarrow C_j}(e_{ji} = b) &= m_{ji} \mu_{N_i \rightarrow B_i}(v_i = b) \prod_{j \in M(i) \setminus \{j\}} \mu_{C_j \rightarrow B_i}(e_{ji} = b) \\ \Rightarrow q_{ji}^b &= m_{ji} \cdot p_i^b \prod_{j \in M(i) \setminus \{j\}} r_{ji}^b \end{aligned} \quad (3.41)$$

所以此時的 bit node 傳送到 check node 的機率資訊表示成 Log-Likelihood ratio(LLR) 為 $LLR(q_{ji}) = \log \frac{\text{"1"的機率}}{\text{"0"的機率}}$

$$LLR(q_{ji}) = \sum_{j \in M(i) \setminus \{j\}} LLR(r_{ji}) + LLR(p_i) \quad (3.42)$$

而 check node 傳送到 bit node 的機率資訊由式子(3.27)所導出來

$$LLR_{C \rightarrow B_0}(b_0) = 2(-1)^{K+1} \tanh^{-1} \left(\prod_{i=1}^K \tanh \left(\frac{1}{2} LLR_{B_i \rightarrow C}(b_i) \right) \right)$$

所以此時的 check node 傳送到 bit node 的機率資訊表示成 Log-Likelihood

ratio(LLR) 為 $LLR(r_{ji}) = \log \frac{\text{"1"的機率}}{\text{"0"的機率}}$

$$LLR(r_{ji}) = 2(-1)^{|L(j)|} \tanh^{-1} \left(\prod_{i \in L(j) \setminus \{i\}} \tanh\left(\frac{1}{2} LLR(q_{ji'})\right) \right) \quad (3.43)$$

因為公式(3.43)含有 \tanh 及 \tanh^{-1} 的函數，所以在硬體電路的實作方面，要實現這兩個函數的硬體比較複雜，而且 \tanh 的函數在公式(3.43)中還是連乘的形式，更增加硬體的實現的困難度，於是我們以下用一個方法來逼近公式(3.43)，把連乘的運算符號簡化成連加的運算式子，以降低硬體的複雜度，方便電路實現。

首先我們引進兩個式子，假設 a_i 是一個實數時，則總共有 i 個實數相乘可以寫成如下式(3.44)的表示方法。另外我們定義一個 $\Psi(x)$ 函數，其方程式如(3.45)所表示。

$$\prod_i a_i = \left(\prod_i \text{sgn}(a_i) \right) \exp \left(\sum_i \log(|a_i|) \right) \quad (3.44)$$

$$\Psi(x) = -\log \left(\tanh\left(\frac{x}{2}\right) \right) = \log \frac{1 + \exp(-x)}{1 - \exp(-x)} \quad (3.45)$$

把公式(3.44)、(3.45)代入到公式(3.43)裡面，則可以化簡為如下式所表示：

$$\begin{aligned} LLR(r_{ji}) &= 2(-1)^{|L(j)|} \tanh^{-1} \left(\prod_{i \in L(j) \setminus \{i\}} \tanh\left(\frac{1}{2} LLR(q_{ji'})\right) \right) \\ \Rightarrow LLR(r_{ji}) &= 2(-1)^{|L(j)|} \tanh^{-1} \left(\left(\prod_{i \in L(j) \setminus \{i\}} \text{sgn}\left(\tanh\left(\frac{1}{2} LLR(q_{ji'})\right)\right) \right) \exp \left(\sum_{i \in L(j) \setminus \{i\}} \log\left(|\tanh\left(\frac{1}{2} LLR(q_{ji'})\right)|\right) \right) \right) \\ \Rightarrow LLR(r_{ji}) &= (-1)^{|L(j)|} \cdot \prod_{i \in L(j) \setminus \{i\}} \text{sgn}(LLR(q_{ji'})) \cdot \Psi \left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji'})|) \right) \\ \Rightarrow LLR(r_{ji}) &= (-1)^{|L(j)|} \cdot s_{ji} \cdot \Psi \left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji'})|) \right) \end{aligned} \quad (3.46)$$

其中 s_{ji} 表示成 $s_{ji} = \prod_{i \in L(j) \setminus \{i\}} \text{sgn}(LLR(q_{ji'}))$

因為函數 $\Psi(x) = -\log\left(\tanh\left(\frac{x}{2}\right)\right) = \log\frac{1+\exp(-x)}{1-\exp(-x)}$ 有一個特性，那就是 x 的

越小時，則所對應的 $\Psi(x)$ 值會很大，如圖 3.10 所呈現的曲線圖，而且 $\Psi(x)$ 的反

函數是他自己本身， $\Psi(\Psi(x)) = x$ ，所以藉由 $\Psi(x)$ 的這些特性，我們可以知道在

公式(3.46)中的 $\Psi\left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|)\right)$ 中，因為在 x 的值越小時則 $\Psi(x)$ 值會很

大，所以整個 \sum 起來時，只看最小的 $|LLR(q_{ji})|$ 即可，因為最小的 $|LLR(q_{ji})|$ 會

決定出最大的 $\Psi(|LLR(q_{ji})|)$ ，用來近似 $\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|)$ ，因此

決定一切。

$$\Psi\left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|)\right) \approx \Psi\left(\max_{i \in L(j) \setminus \{i\}} (|LLR(q_{ji})|)\right) \approx \min_{i \in L(j) \setminus \{i\}} (|LLR(q_{ji})|)$$

因此公式(3.46)可以近似成下式，把原本的複雜的連乘 \tanh 函數簡化掉了，這樣可以以利電路硬體的實現，降低複雜度。

$$LLR(r_{ji}) = (-1)^{|L(j)|} \cdot s_{ji} \cdot \Psi\left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|)\right) \quad (3.47)$$

$$\Rightarrow LLR(r_{ji}) \approx (-1)^{|L(j)|} \cdot s_{ji} \cdot \min_{i \in L(j) \setminus \{i\}} (|LLR(q_{ji})|)$$

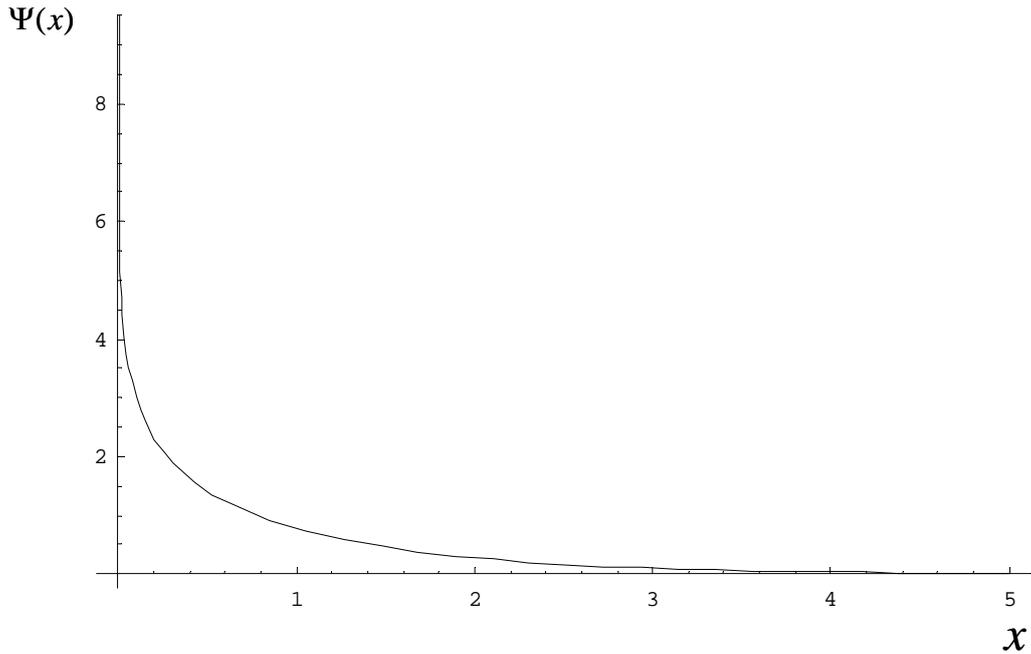


圖 3.10 $\Psi(x)$ 的函數圖

經過以上的推導出 bit node 傳送到 check node 的機率資訊表示成 Log-Likelihood ratio(LLR)為 $LLR(q_{ji}) = \sum_{j \in M(i) \setminus \{j\}} LLR(r_{ji}) + LLR(p_i)$ 。而 check node 傳送到 bit node 的機率資訊表示成 Log-Likelihood ratio(LLR)為 $LLR(r_{ji}) = 2(-1)^{|L(j)|} \tanh^{-1} \left(\prod_{i \in L(j) \setminus \{i\}} \tanh\left(\frac{1}{2} LLR(q_{ji})\right) \right)$ 。及把 $LLR(r_{ji})$ 裡的 \tanh 函數經過簡化，則整個 LDPC decoder 過程變的比較簡單而且容易實現了，下面列出五個 LDPC 解碼的解碼過程。

第一步：初始化(Initialize)

首先令一開始的 check node 傳送到 bit node 的 "0" 或 "1" 機率資訊 $LLR(r_{ji})$ 是相等的， $(r_{ji}^0, r_{ji}^1) = (\frac{1}{2}, \frac{1}{2})$ ， $LLR^{(0)}(r_{ji}) = \log \frac{r_{ji}^1}{r_{ji}^0} = \log\left(\frac{0.5}{0.5}\right) = 0$ 。再來接著計算 another node 傳送到 bit node 的本質機率(intrinsic probability) $LLR(p_i)$

$$LLR(p_i) = LLR_{LDPC}^{\text{int}}(v_i) = \log \frac{R_{LDPC}^{\text{int}}(v_i = 1)}{R_{LDPC}^{\text{int}}(v_i = 0)} = \log \frac{\mu_{N_i \rightarrow B_i}(v_i = 1)}{\mu_{N_i \rightarrow B_i}(v_i = 0)}$$

第二步：計算 bit node 傳送到 check node 的機率資訊 (Bit-to-check messages)

bit node 傳送到 check node 的後置機率資訊表示成 Log-Likelihood ratio (LLR) 為 $LLR(q_{ji})$ ，如同公式(3.42)所推導出的一樣，只是這邊加上一個 iteration 的次數而已。

$$LLR^{(K)}(q_{ji}) = \sum_{j \in M(i) \setminus \{j\}} LLR^{(K-1)}(r_{ji}) + LLR(p_i) \quad (3.48)$$

此處的 K 表示 LDPC 解碼的時候最大 iteration 的次數， K 從 1 開始增加。

第三步：計算 check node 傳送到 bit node 的機率資訊 (Check -to-bit messages)

在公式(3.46)中，我們推導出 check node 傳送到 bit node 的機率資訊為

$$LLR(r_{ji}) = (-1)^{|L(j)|} \cdot \prod_{i \in L(j) \setminus \{i\}} \text{sgn}(LLR(q_{ji})) \cdot \Psi \left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|) \right)$$

$\Psi(x) = -\log \left(\tanh \left(\frac{x}{2} \right) \right) = \log \frac{1 + \exp(-x)}{1 - \exp(-x)}$ 函數的特性，可以把 $LLR(r_{ji})$ 簡化成用

$|LLR(q_{ji})|$ 的最小值來近似，如下列式子，此處 K 和第二步裡的 K 一樣表示 LDPC 解碼的時候最大 iteration 的次數， K 從 1 開始增加。

$$LLR^{(K)}(r_{ji}) = (-1)^{|L(j)|} \times \left(\prod_{i \in L(j) \setminus \{i\}} \text{sgn}(LLR^{(K)}(q_{ji})) \right) \times \min_{i \in L(j) \setminus \{i\}} (|LLR^{(K)}(q_{ji})|) \quad (3.49)$$

第四步：經過 iteration 後計算後置機率資訊 (Compute output)

一開始我們由第一步先初始化本質機率 (intrinsic probability) $LLR(p_i)$ 及

$$LLR^{(0)}(r_{ji}) = \log \frac{r_{ji}^1}{r_{ji}^0} = \log \left(\frac{0.5}{0.5} \right) = 0$$

把這兩個機率值代入第二步計算，進而我們可以得到後置機率 (posterior probability) $LLR^{(K)}(q_{ji})$ ，得到後置機率後我們利用硬性判斷 (hard decision) 來決定解出來的 bit 是 "1" 還是 "0"，若 $LLR^{(k)}(q_i) > 0$ ，則表

示經過解碼後認為傳送端是傳 "1"，反之若 $LLR^{(k)}(q_i) < 0$ ，則表示經過解碼後認為傳送端是傳 "0"， $\hat{v}_i^{(K)}$ 代表解碼後的碼字 (code word)。

$$\hat{v}_i^{(K)} = \begin{cases} 1 & \text{if } LLR^{(K)}(q_i) > 0 \\ 0 & \text{if } LLR^{(K)}(q_i) < 0 \end{cases} \quad (3.50)$$

第五步：解碼過程一直 iteration，直到解出正確的碼字(Repeat until done)

由第四步驟我們經過解碼後得到的碼字，我們要看他有沒有符合 $H\hat{V}^T = 0$ 的關係式，若沒有符合時則代表我們解碼出來的碼字還不是正確的，因此要繼續進行下一個 iteration，也就是把第三步驟中 check node 傳送到 bit node 的機率資訊

$$LLR(r_{ji}) = (-1)^{|L(j)|} \cdot \prod_{i \in L(j) \setminus \{i\}} \text{sgn}(LLR(q_{ji})) \cdot \Psi \left(\sum_{i \in L(j) \setminus \{i\}} \Psi(|LLR(q_{ji})|) \right)$$

，代入到第二步驟中去計算出後置機率 (posterior probability) $LLR^{(K)}(q_{ji})$ ，然後再把後置機率 $LLR^{(K)}(q_{ji})$ 代入到第三步驟中，因此我們得到更新過後的 check node 傳送到 bit node 的機率資訊 $LLR^{(K)}(r_{ji})$ ，然後經過第四步驟的硬性判斷 (hard decision) 決定出解碼過後的碼字 $\hat{v}_i^{(K)}$ ，若條件 $H\hat{V}^T = 0$ 不成立的話，則一直重複步驟二、三、四。但是若一直沒有解碼出正確的碼字時，則也不可能無限制次數的一直 iteration 下去，所以這邊定義一個 K_{\max} ，若解碼的次數 K 每經過一次 iteration 則加 1，變成 $K+1$ ，直到達到 K_{\max} 時，還沒有解出正確的碼字，則整個整個 LDPC decoder 過程要停止，而輸出的資訊以最後一次 K_{\max} 的 iteration 為準。

第四章

4.1 LDPC block length 求得

在歐規的數位影像廣播系統之地面廣播系統(DVB-T System)的架構中有分成兩組編碼方式，編碼方式包含有外層編碼器(Outer Coder)、外層交錯器(Outer Interleaver)、內層編碼器(Inner Coder)以及內層交錯器(Inner Interleaver)等技術。圖 4.1 為 DVB-T 系統發射機的功能方塊圖，由此圖我們可以知道 DVB-T 系統的編碼方式在整個發射機的位置，資料依序先經過外層編碼器、外層交錯器、內層編碼器以及最後經過內層交錯器，以下將依次介紹這個系統的編碼方式。

反之接收機的架構依序為先經過內層反交錯器(Inner De-interleaver)、內層解碼器(Inner Decoder)、外層反交錯器(Outer De-interleaver)、外層解碼器(Outer Decoder)，而解出原本傳送端的資料流。如圖 4.2 所示。

而本論文所使用的方式為使用 LDPC code 來取代原本的里德所羅門碼及迴旋碼，及其所對應的交錯器。其傳送端與接收機的架構如圖 4.3 及圖 4.4 所示。

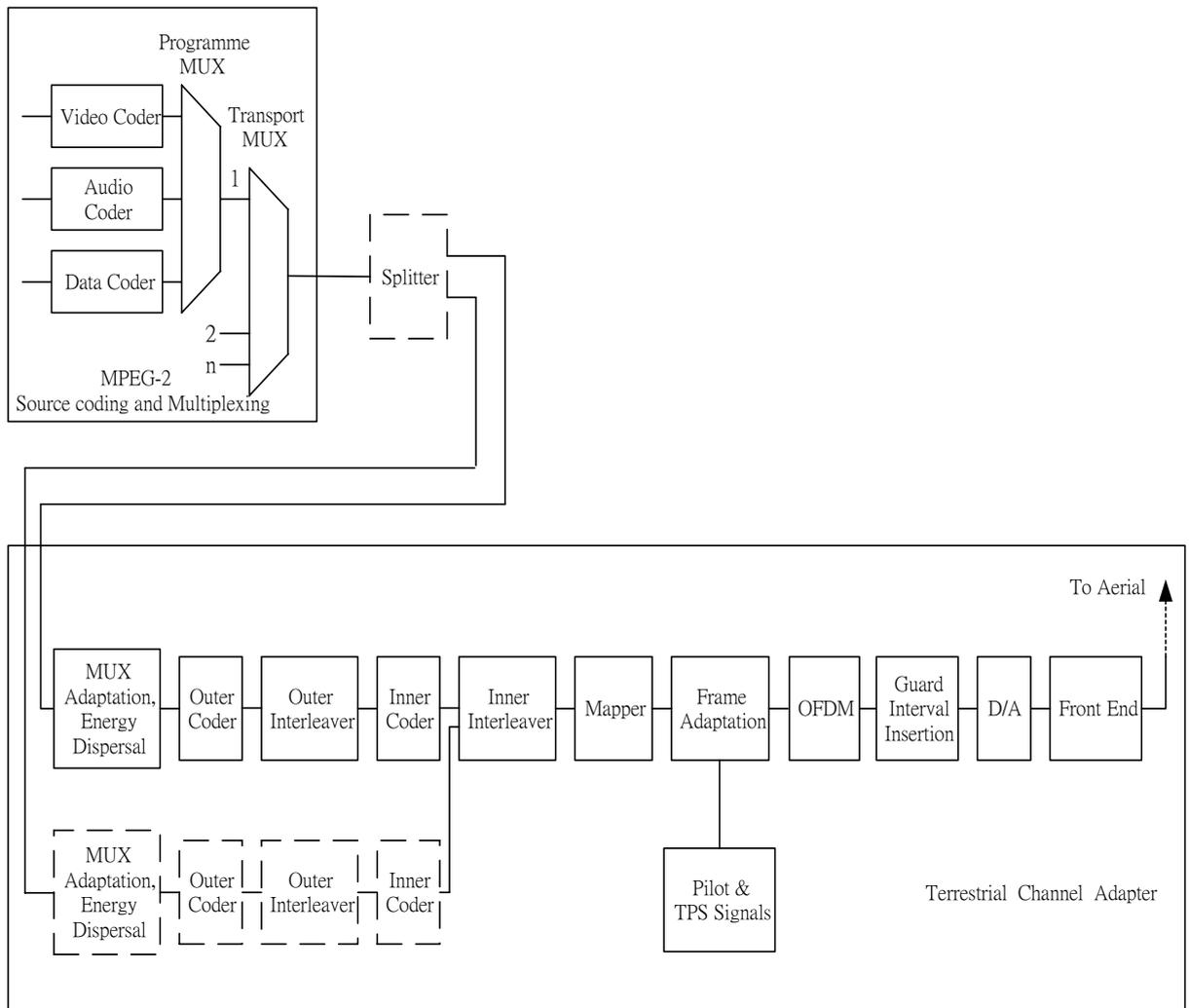


圖 4.1 DVB-T 系統發射機的功能方塊圖

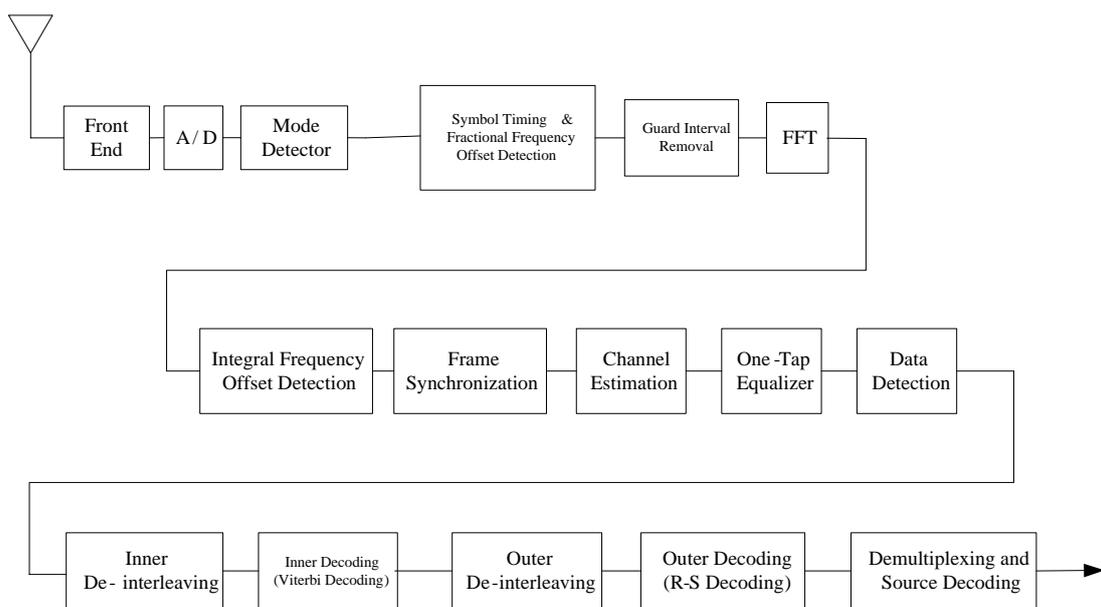


圖 4.2 DVB-T 系統接收機的功能方塊圖

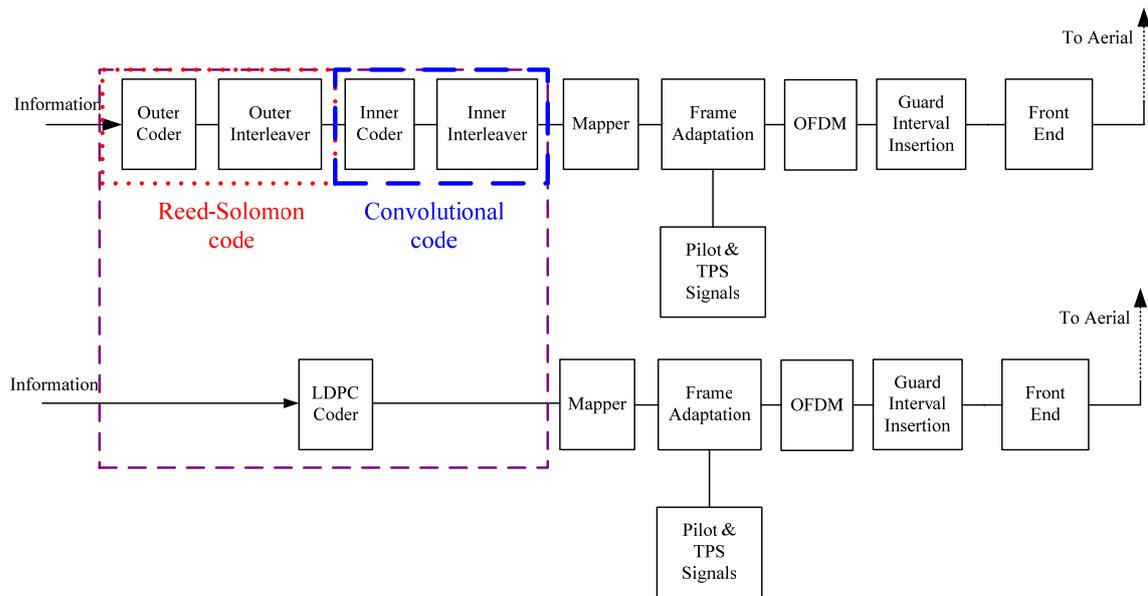


圖 4.3 LDPC 取代 DVB-T 兩層編碼的發射機功能方塊圖

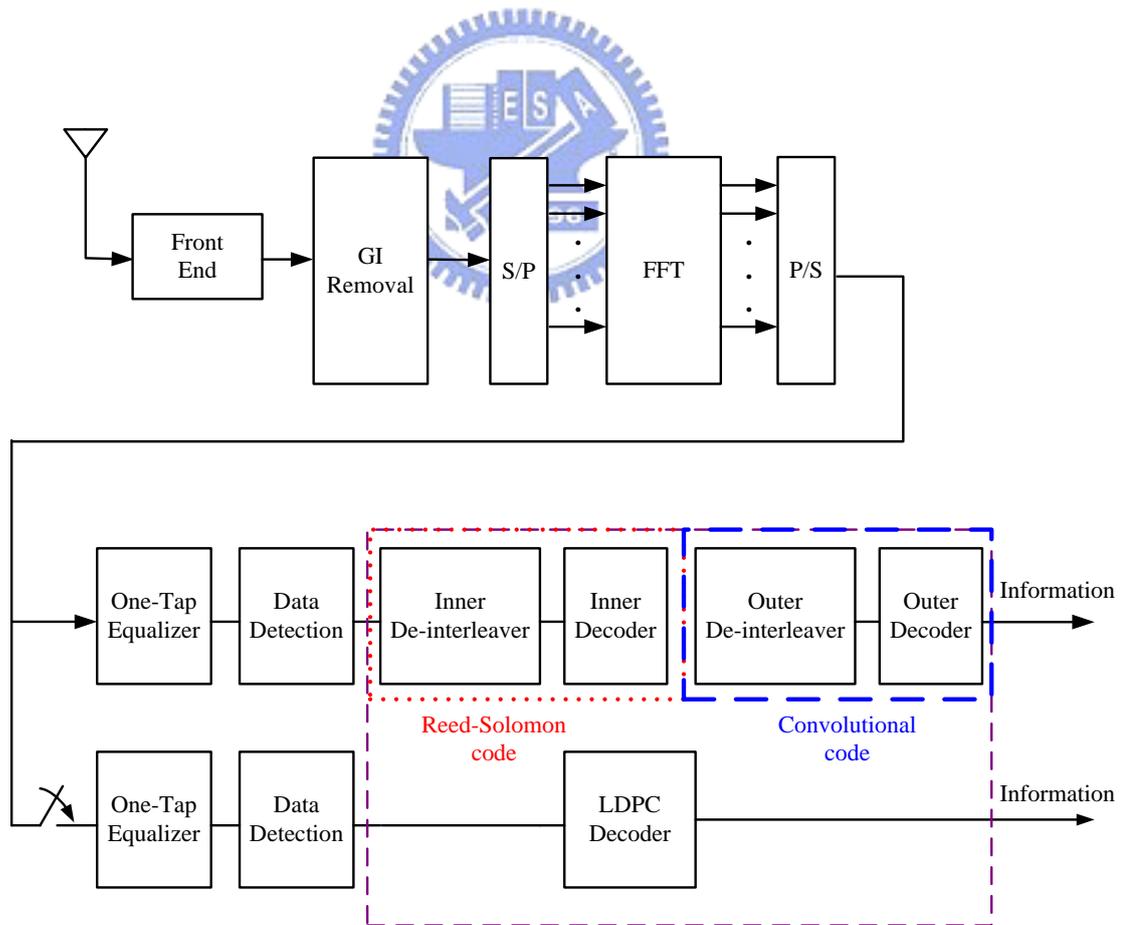


圖 4.4 LDPC 取代 DVB-T 兩層編碼的接收機功能方塊圖

數位影像廣播之地面廣播系統的外層編碼器(Outer Coder)是使用里德所羅門短碼(Reed-Solomon Shorten Code, RS(204,188,t=8)),使每個封包由 188 個位元組經過編碼後變成為 204 個位元組,並且可以更正最長為 8 個位元組的連續錯誤。

此里德所羅門短碼 RS(204,188,t=8)可以由正規的 RS(255,239,t=8)導出來。首先在原本的資料位元組之前加入 51 個“zero”位元組,這 51 個位元組都設為零,所以總共有 51+188=239 個資料位元組,再把這 239 個位元組經過正規的 RS(255,239,t=8)編碼器,等到資料流經過編碼程序後,會產生 255 個位元組,再把前面 51 個“Null”位元組去掉,所以剩下 204 個位元組 (255-51=204),就成為 DVB-T 外層編碼器要求的里德所羅門短碼 RS(204,188,t=8)。

在里德所羅門碼錯誤保護封包中末端加上 16 個保護位元組(Parity bytes),而變成 188+16=204 個位元組,如圖 4.5 所示。

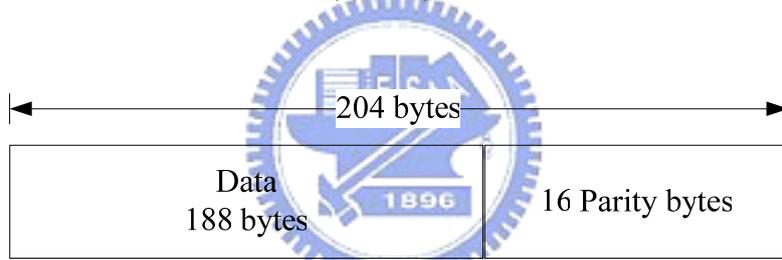


圖 4.5 里德所羅門碼 RS(204,188,t=8)錯誤保護封包

在經過外層交錯器後其傳輸封包的結構如圖 4.6 所示,共有 204 個位元組。

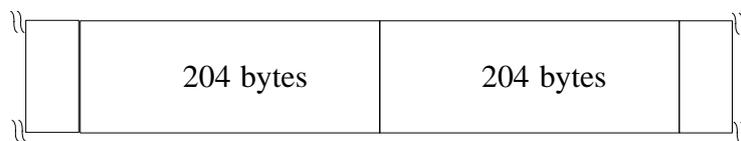


圖 4.6 外層編碼之交錯分佈後的資料結構分佈, I=12 bytes

外層交錯器是一個以位元組為單位的迴旋交錯器(Convolutional Interleaver),用迴旋交錯器的優點是可以節省記憶體,這個交錯器有 I=12 個分支結構連接於輸入的資料流,每個分支是一個延遲長度以 M=17 個位元組為單位成線性比例增加的先輸入先輸出(First-In,First-Out, FIFO)平移暫存器(Shift Register),且深度為 $17 \times j$ 個位元組時間(j 為交錯器結構的分支編號,

$j=0,1,2,\dots,11$)，所有暫存器的深度總和即為延遲的時間長度 ($17 \times 11 \times 12 / 2 = 1122$ 個位元組)。交錯器和反交錯器的結構剛好相反，可以在反交錯器端還原正確的接收資料流 (每個分支的總延遲時間相同)。為了達到訊號同步的需求，同步位元組一定在編號為“0”的分支傳送。如圖 4.7 所示。

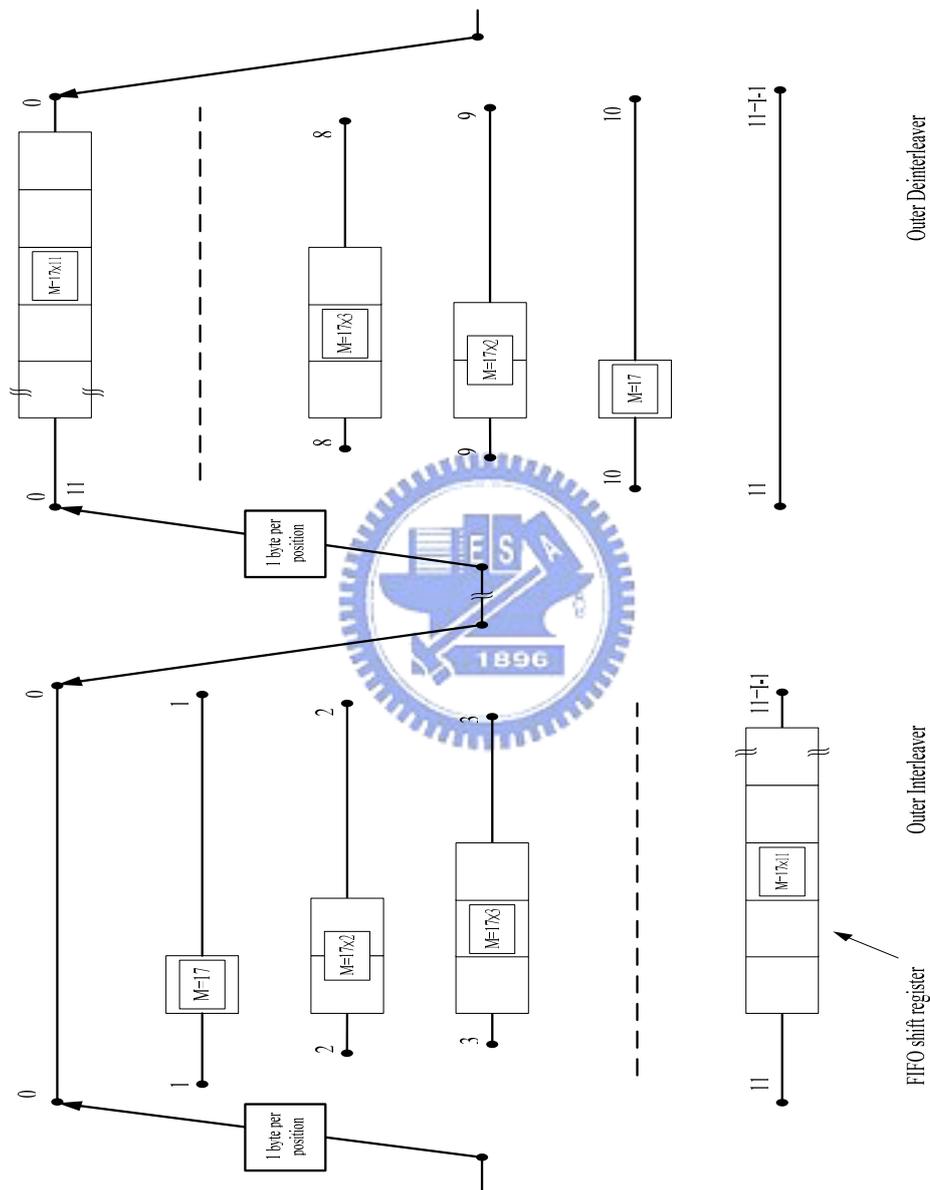


圖 4.7 外層交錯器和反交錯器結構圖

在交錯器的結構中，原本在資料流中兩相鄰的位元組經過交錯器後會相隔 $17 \times 12 + 1 = 205$ 個位元組。也就是說一個里德所羅門碼 RS(204,188,t=8)錯誤保護封包中的每一個位元組經過交錯器後只會分散到隔壁位置的錯誤保護封包而已。如圖 4.8 所示。

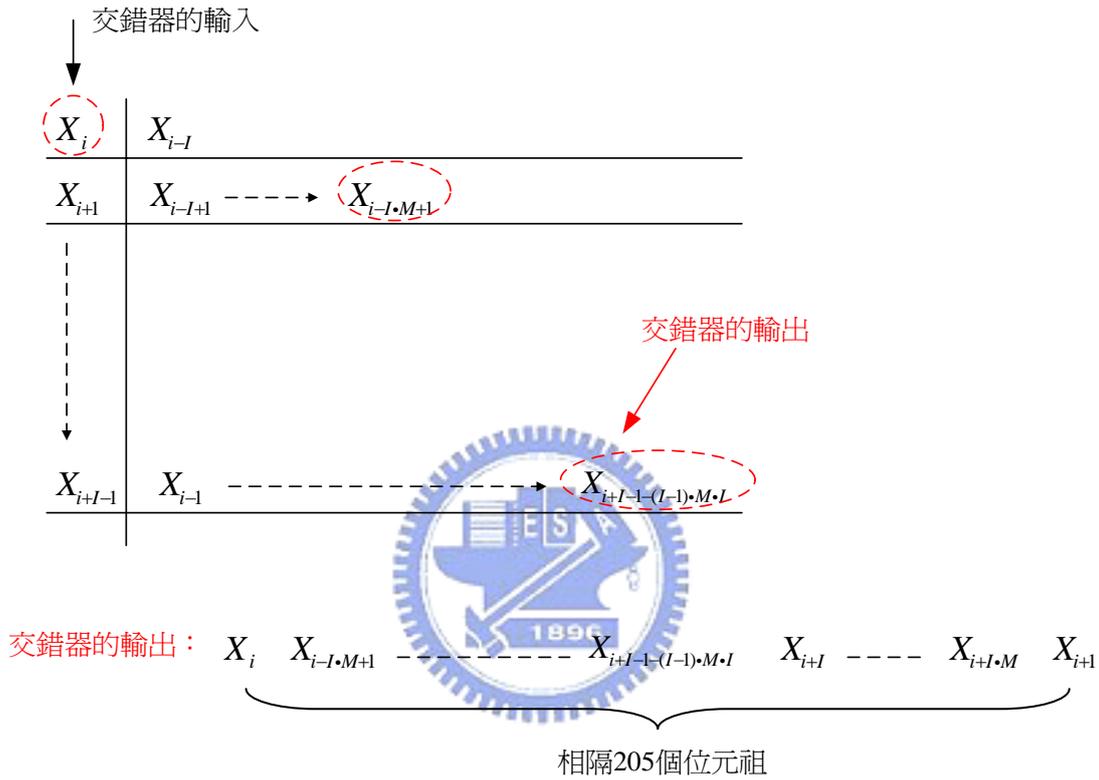


圖 4.8 交錯器的輸入與輸出關係圖

以上的介紹為外層編碼器的交錯器，而在 DVB-T 系統中，還有一個內層編碼器的交錯器，也就是 convolutional code 的交錯器，這部份是用查表的方式來求得，所以整個 DVB-T 系統中外層交錯器和內層交錯器的總深度(depth)很難去計算，因此靠模擬的方法來求得總深度為 3768 個 bit，所以 LDPC 的 block length 就設為 3600 個 bit 來模擬，編碼率為 $\frac{1}{2}$ ，iteration 數目為 100 次。

4.2 模擬結果

模擬參數：

DVB-T 模式(mode)	2K mode
調變(modulation)	QPSK
載波頻率(carrier frequency)	600 MHz
頻寬(total bandwidth)	7.61MHz
次載波個數(number of subcarriers)	2048
有效符元時間(useful symbol time)	$T_U = 224 \mu \text{ sec}$
護衛間隔(guard interval)	$1/4 T_U = 56 \mu \text{ sec}$
車速(vehicle speed)	30,60Km/hr
路徑個數(path number)	2

表格 4.1 系統的模擬參數



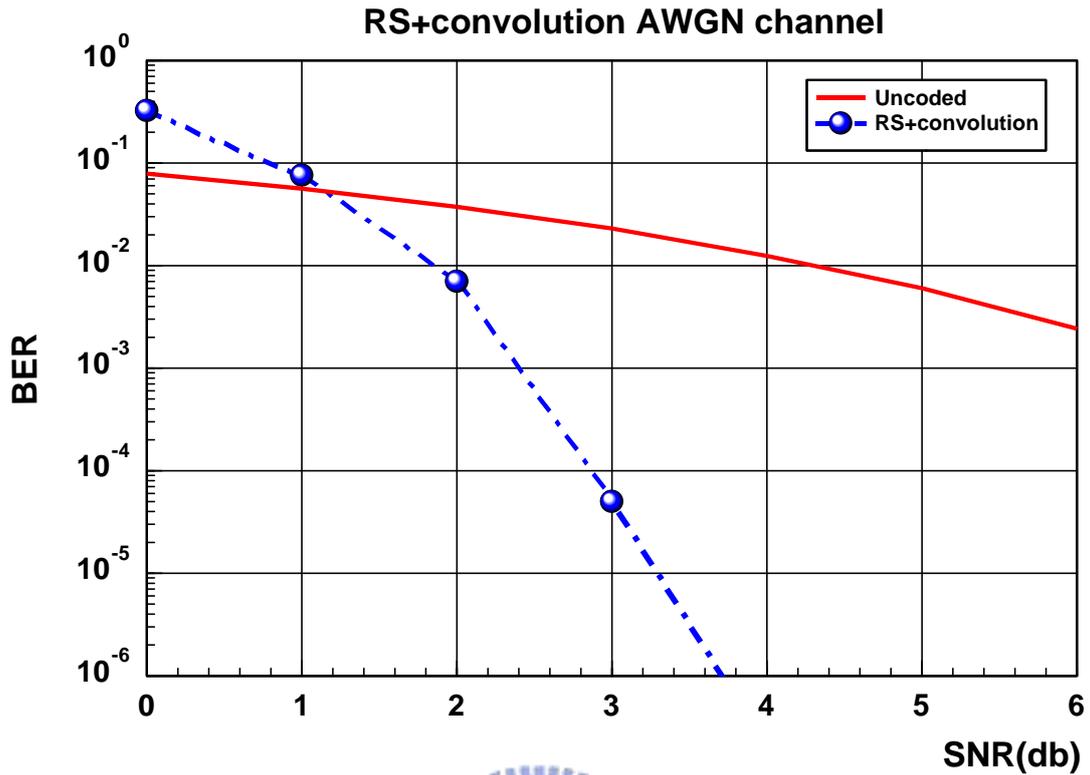


圖 4.9 DVB-T 在 AWGN 通道境下模擬結果

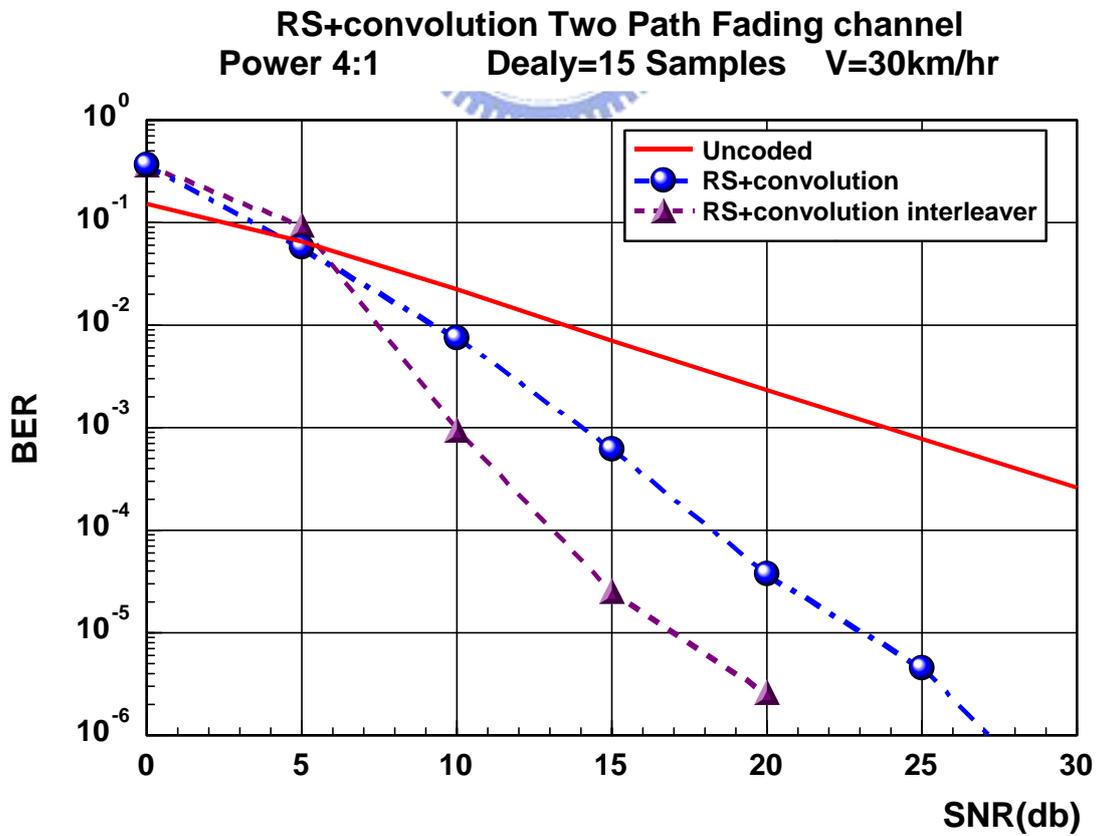


圖 4.10 DVB-T 在 fading 通道境下 30 公里車速模擬結果

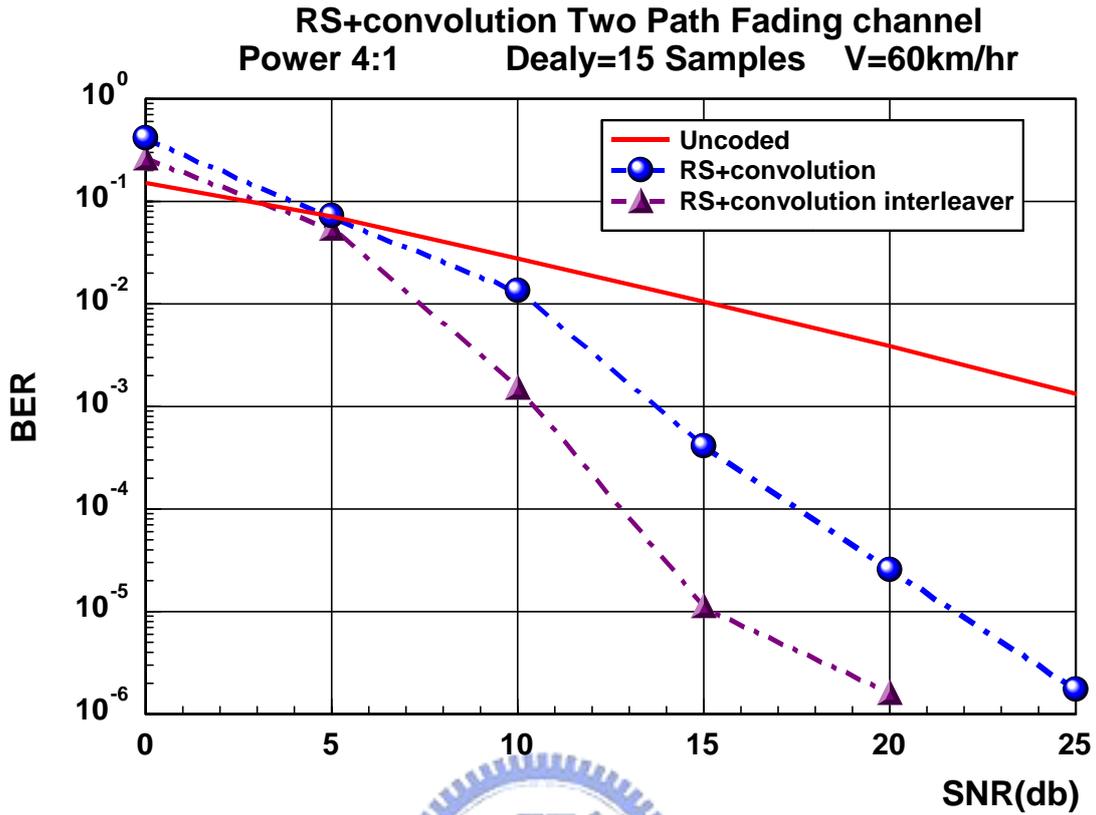


圖 4.11 DVB-T 在 fading 通道境下 60 公里車速模擬結果

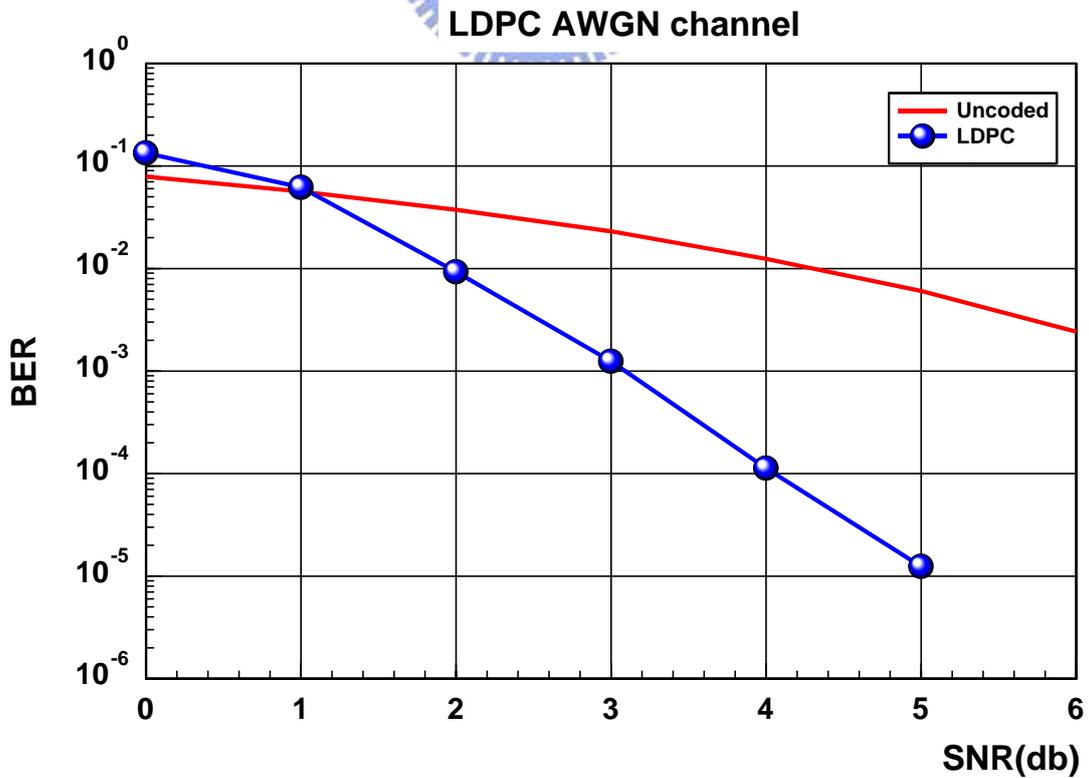


圖 4.12 LDPC 在 AWGN 通道境下模擬結果

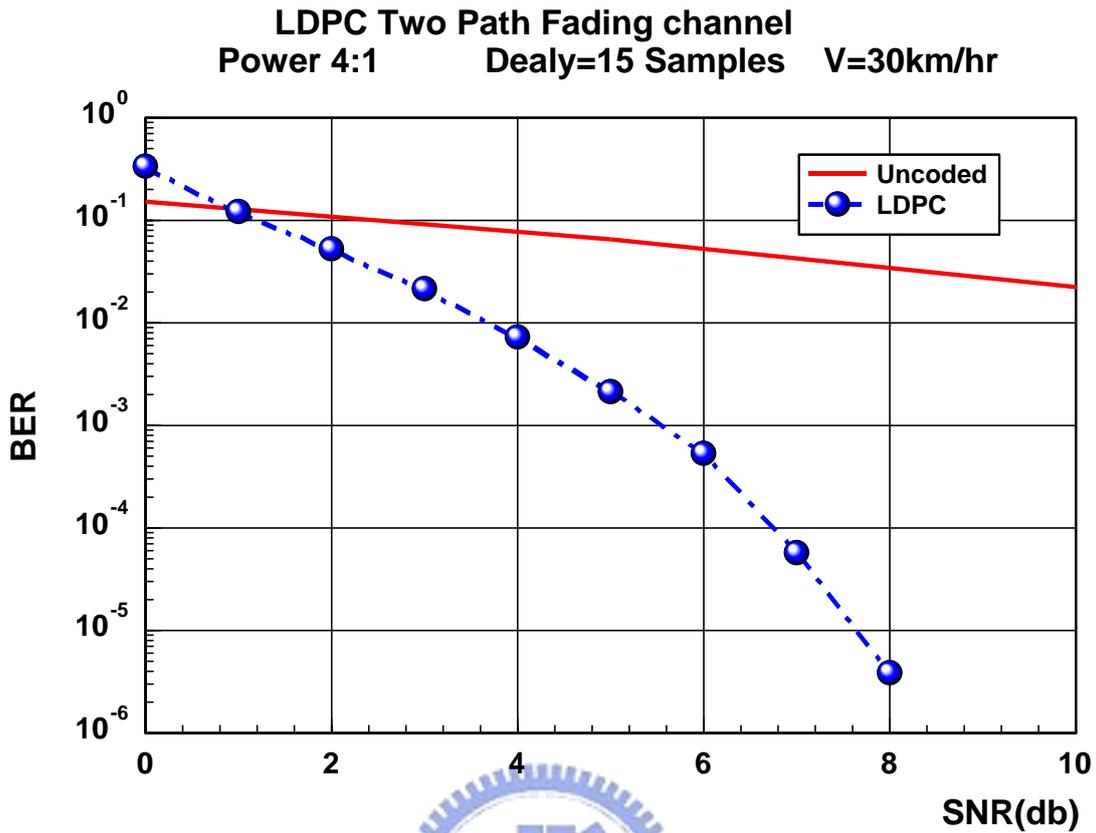


圖 4.13 LDPC 在 fading 通道境下 30 公里車速模擬結果

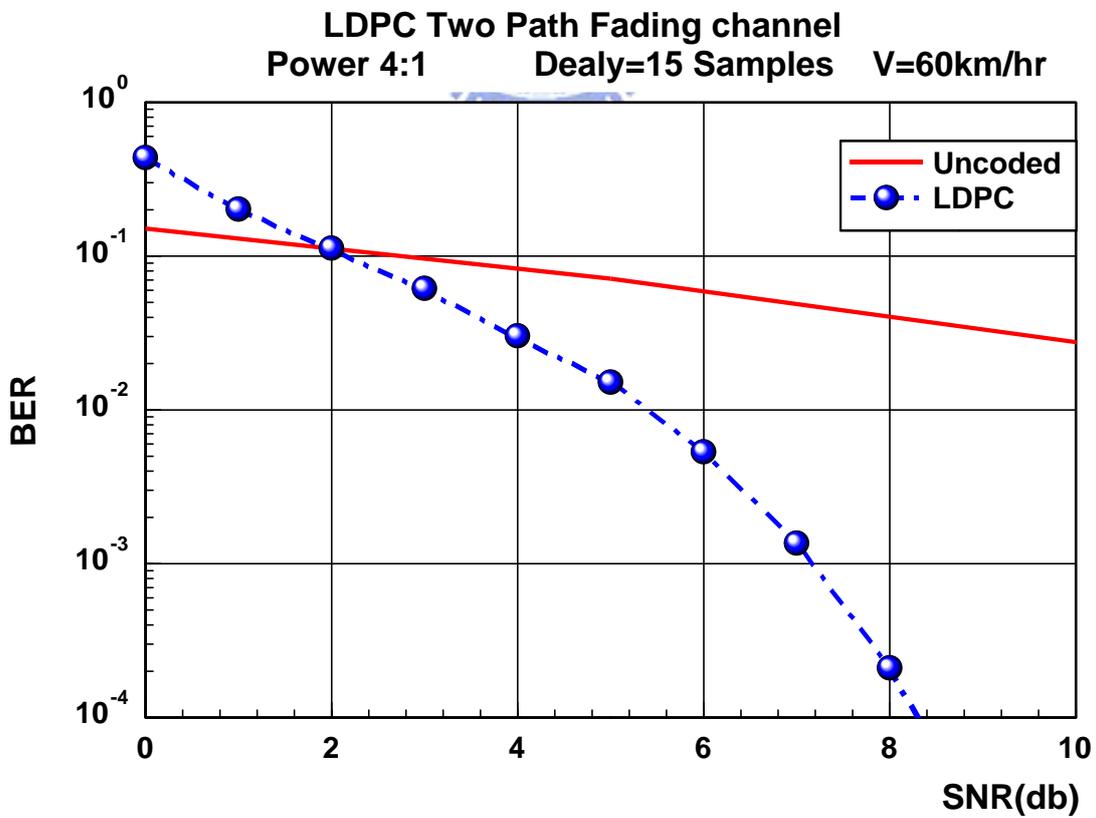


圖 4.14 LDPC 在 fading 通道境下 60 公里車速模擬結果

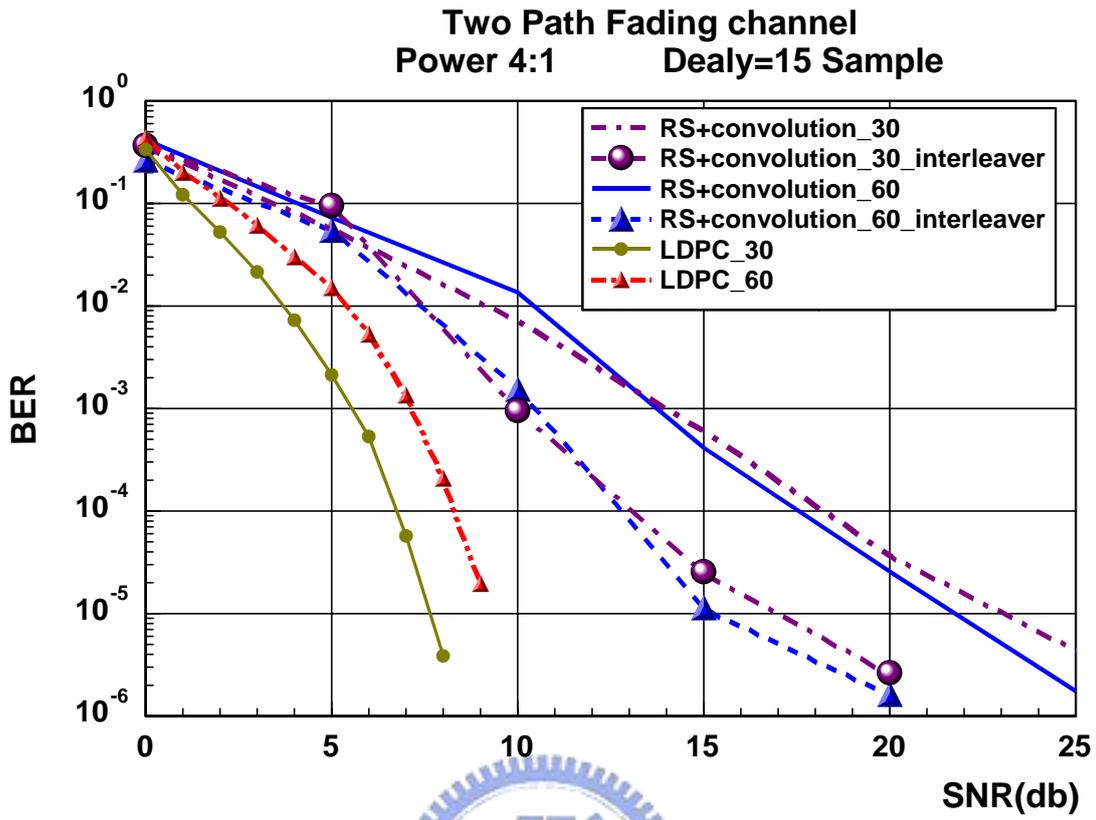


圖 4.15 LDPC 與 RS+convolution 在 fading 通道境下模擬結果

第五章

結論與未來發展的方向

在第四章的模擬中我們總共模擬兩種車速，分別為 30 公里/小時及 60 公里/小時，模擬結果圖 4.15 知道在和 DVB-T 的交錯器相同深度的 LDPC 可以有著較好的效能，因此 DVB-T 系統的編碼方式可以用 LDPC 來實現。

因為這個模擬中並沒有針對 LDPC 的 parity matrix(H)找出最佳化的解，模擬的過程中發現 parity matrix(H)對 performance 的影響很大，因此未來可以針對如何找出好的 parity matrix(H)來進行研究。

LDPC 的解碼演算法，是平行且疊代的架構，適合硬體的實現。又因為 LDPC 的 parity matrix(H)有著稀疏特性，使得其在編碼的時候不需要交錯器，因為 bit 的資訊會相隔比較遠，有著同樣交錯器的效果。

參考文獻

- [1] John L. Fan , Constrained Coding and Soft Iterative Decoding.
- [2] Stephen B. Wicker; Vijay K. Bhargava, Reed-Solomon Codes and Their Applications.
- [3] Stephen B. Wicker, Error Control Systems for Digital Communication and Storage.
- [4] William E. Ryan, “A Turbo Code Tutorial.”
- [5] Simon Haykin, Communication System, 4th ed. John Wiley & Sons, 2001.
- [6] J. G. Proakis, Digital Communications, New York: McGraw-Hill,1989.
- [7] William E. Ryan, “A Turbo Code Tutorial.”
- [8] Therodore S. Rappaport, Wireless Communications Principles and Practice, Prentice Hall PTR, 1996.
- [9] A. J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,” *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, April, 1967.
- [10] R. G. Gallager, “Low Density Parity Check Codes”, *IEEE Trans. Inform. Theory*, IT:8:21-28, Jan. 1962.
- [11] T. J. Richardson, R. E. Urbanke, “Efficient encoding of low-density parity-check codes”, *IEEE Trans. Inform. Theory*, vol. 47, pp. 638-656, Feb. 2001.
- [12] D. J. C. MacKay, “Near Shannon limit performance of low density parity check codes,” *Electron. Lett.*, vol. 33, pp. 457–458, Mar. 1997.
- [13] Futaki, H.; Ohtsuki, T., “Low-density parity-check (LDPC) coded OFDM systems”, Page(s): 82 -86 vol.1, Vehicular Technology Conference, 2001. VTC 2001 Fall. *IEEE VTS 54th* , Volume: 1 , 2001
- [14] D. Burshtein, G. Miller, “Bounds on the performance of belief propagation decoding”, *IEEE Trans. Inform. Theory*, vol. 48, pp.112-122, Jan.. 2002
- [15] Digital Video Broadcasting System *European Standard*