國　立　交　通　大　學

電信工程學系碩士班

碩士論文

IEEE 802.11n 無線區域網路系統之偵測與 Viterbi 解碼:設計及實現

Detection and Viterbi Decoding for IEEE 802.11n Wireless LAN System – Design and Implementation

研 究 生：蔡耀毅

指導教授：吳 文 榕　博士

中華民國九十四年十月

IEEE 802.11n 無線區域網路系統之偵測與 Viterbi 解碼:設計及實現

Detection and Viterbi Decoding for IEEE 802.11n Wireless LAN System – Design and Implementation

研 究 生：蔡耀毅　　Student：Yao-Yi Tsai

指導教授：吳文榕 博士　　Advisor：Dr. Wen-Rong Wu

國 立 交 通 大 學

電信工程學系碩士班

碩 士 論 文

A Thesis

Submitted to Department 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

October 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年十月

IEEE 802.11n 無線區域網路系統之偵測與 Viterbi 解碼:設計及實現

研究生：蔡耀毅　　指導教授：吳文榕 教授

國立交通大學電信工程學系碩士班

中文摘要

在本論文中，我們考慮 IEEE 802.11n 基頻接收機之設計與實現。明確地，我們著重在信號偵測與 Viterbi 解碼。802.11n 系統為一個 bit-interleaved coded modulation (BICM)，多輸入多輸出（Multi-input Multi-output，MIMO）以及正交分頻多工技術（Orthogonal Frequency Division Multiplexing, OFDM）的系統。雖然有許多偵測與解碼演算法是眾所周知的，有效的 BICM MIMO-OFDM 系統之實現仍舊是一大挑戰。這是因為使用 MIMO 的結構顯著地增加系統的吞吐量以及在位元串流(bit streams)間引進干擾。模擬顯示 Viterbi 解碼器前面串接 ZF 演算法以及其前面串接最小均方差(minimum mean square，MMSE)演算法有著相近的效能。而 ZF 演算法的計算複雜度較低。因此我們提出使用 zero-forcing (ZF)演算法來做信號偵測，而使用 Radix-4 之 trace-back 的 Viterbi 解碼器來做解碼。我們亦提出可以避免 ZF 演算法以及 channel state information (CSI)計算中的除法運算之演算法。使用 IEEE 預先定義好的通道模型，我們引導出一些模擬結果來評估系統的效能。最後，我們使用 FPGA 設計流程來實現這個基頻接收機。

Detection and Viterbi Decoding for IEEE 802.11n Wireless LAN System – Design and Implementation

Student: Yao-Yi Tsai Advisor: Dr. Wen-Rong Wu

Institute of Communication Engineering

National Chiao-Tung University

Abstract

In this thesis, we consider design and implementation of an IEEE 802.11n baseband receiver. Specifically, we focus on signal detection and Viterbi decoding. The 802.11n system is known to be a bit-interleaved coded modulation (BICM), multi-input multi-output (MIMO), and frequency division multiplexing (OFDM) system. Although many detection and decoding algorithms are well-known, efficient implementation for the BICM MIMO-OFDM system remains challenging. This is due to the use of the MIMO configuration significantly increasing the system throughput and introducing interference between bit streams. It is shown that the Viterbi decoder pre-cascaded with the ZF algorithm has similar performance with that pre-cascaded with the minimum mean square (MMSE) algorithm. However, the computational complexity of the ZF algorithm is much lower. We thus propose to use the zero-forcing (ZF) algorithm for signal detection and a radix-4 trace-back Viterbi decoder for decoding. We also propose algorithms that can avoid division operations required in the ZF algorithm and channel state information (CSI) calculation. Using

IEEE pre-defined channels, we conduct simulations to evaluate the performance of the system. Finally, we implement the baseband receiver with the FPGA design flow.

誌謝

首先我要感謝指導老師吳文榕教授，在研究所求學期間對於論文研究詳盡的指導，使我在論文研究中研究方向都不會偏離正軌。而老師細心、嚴謹的求學態度更使我受益匪淺。同時感謝口試委員陳紹基教授與張大中助理教授，對本篇論文提出寶貴意見與建議，使得論文內容更加充實、完備。

其次，我要感謝林壽煦學長、李俊芳學長、楊華龍學長、李彥文學長和許兆元學長他們在研究及課業學習上不吝指導及鼓勵，且同時感謝寬頻傳輸與訊號處理實驗室所有同學與學弟妹們的幫忙。再來感謝我老婆多年來不斷地鼓勵與支持。最後致上我最深的感謝給我父母，他們給予我精神和經濟上的支持，使我無後顧之憂順利完成研究所的碩士學位。

內容目錄

表目錄

圖目錄

第1章簡介

隨著科技與無線通訊技術快速成長，使用者對於通訊傳輸的速率、可靠度與頻譜使用效率的要求也越來越高。正交分頻多工技術（Orthogonal Frequency Division Multiplexing, OFDM）是一種多載波調變技術，它具有高速率傳輸的能力以及高頻譜使用效率，並可克服多重路徑通道（multi-path channel）造成符元間的干擾（Inter Symbol Interference, ISI）與頻率選擇衰落（frequency-selective fading）的問題；此外，為了更進一步提升通道傳輸能力與性能，更結合了多輸入多輸出（Multi-input Multi-output，MIMO）的架構成為多輸入多輸出正交分頻多工（MIMO-OFDM）系統，這樣的系統架構已被制訂中的 IEEE 802.11n 等的傳輸標準採用。WLAN 近年來在市場上取得很大的成功，從 IEEE 802.11b 到 11g 和 11a，在傳輸率的提升也讓人相當肯定，進一步穩固了 802.11 系列在市場上的地位。而目前 IEEE 進行制定的下一代標準 802.11n 也如火如荼的進行當中，其目標是將傳輸率由目前最快 54Mbps 一舉提高至 100Mbps 以上。

MIMO 系統，該技術最早是由 Marconi 於 1908 年提出的，它利用多天線來抑制通道衰落。以收發兩端天線數量而言，MIMO 還可以包括 Single-Input Multiple-Output (SIMO)系統和 Multiple-Input Single-Output (MISO)系統。原則上，通道容量(capacity)隨著天線數量的增大而線性增大。也就是說可以利用 MIMO 系統倍數地提高無線通道容量，在不增加頻寬和天線發送功率的情況下，頻譜利用率可以倍數地提高。

MIMO 的核心概念為利用多根發射天線與多根接收天線所提供之空間自由度提升傳輸速率與改善通訊品質；它主要有兩種功能形式：一為空間多工(spatial multiplex)，另一為空間分集(spatial diversity)。前者是在發射端利用多根天線傳送不同資料序列，並在接收端利用多根天線的空間自由度將該組資料序列分別解出。經由此一程序，在發射端與接收端之間彷彿形成一組虛擬的平行空間通道，可在同一時間、同一頻段，以同一功率傳送多個資料序列。如此一來，整體系統

的有效資料傳輸率便可以在不增加任何通訊資源的前題下提升數倍。而後者是利用發射或接收端的多根天線所提供的多重傳輸途徑來對抗通道衰落(fading)的影響；所謂分集意即多重選擇性，它可由多個獨立的傳輸途徑中選擇或組合出衰落現象較輕微的接收訊號，以維持穩定的鏈路品質。空間多工接收機的演算法主要有貝爾實驗室的 BLAST 演算法、ZF 演算法、MMSE 演算法、ML 演算法等[1]-[6]。而空間分集主要代表便是空時區塊編碼(Space-Time Block Coding, STBC)[7]，它於發射端將待傳送之資料符元(data symbol)在空間與時間上作預前編碼，產生適當的冗餘(redundancy)，並在接收端經由簡易的處理將此冗餘轉化為「分集增益」(diversity gain)。

OFDM 的優點歸納如下：1.相較於單載波調變，頻帶上不需額外的保護區間(guard band)，擁有較佳的頻譜使用效率 2.能有效對抗多路徑通道，降低接收端通道等化器之複雜度。3.硬體上可採用制式化之 FFT 與 IFFT 模組實現調變與解調變，縮短開發時程與降低硬體成本。

雖然 OFDM 有著諸多優點，但其調變系統的複雜度相較於單載波系統仍顯得複雜許多，且系統對於時間與頻率同步誤差非常敏感，一旦存在同步誤差，則接收機將遭遇嚴重之載波間干擾(Inter-Carrier Interference, ICI)，影響系統效能甚鉅。此外，OFDM 發射訊號之峰均功率比(Peak-to-Average Power Ratio, PAPR)較單載波訊號大，不利於功率放大器之運作，也會使得接收訊號失真。

近年來，bit-interleaved coded modulation (BICM)已成為一個有效率的技術而廣泛地應用在無線區域網路系統上。BICM 原先是由 Zehavi 所提出來的[15]，它透過由位元上的交錯(bit-level interleaver)而將編碼和調變分開。事實上在瑞雷快速型衰落(Rayleigh fast fading)通道中，BICM 比起將編碼和調變一起做的傳統系統有更好的碼分集(code diversity)。當通道是頻率選擇衰落時，BICM 將會和 OFDM 結合而使得訊號可以在頻域上獲得分集而將通道轉為平衰落(flat fading)。例如 IEEE802.11a 無線區域網路便是在 5-GHz 的頻帶上採用 BICM-OFDM 的系統。MIMO-OFDM 則是藉由多傳送接收天線及頻率選擇衰落

通道而獲得在空間及頻率上的分集。

在本篇論文中，吾人模擬比較現存 MIMO 偵測(detection)方法並針對一個結合 MIMO 偵測及 Viterbi 解碼的 BICM MIMO-OFDM 系統做實現。現存的系統中大多以 MMSE 及 VBLAST 為 MIMO 偵測的方法，再經過解碼器後輸出，吾人模擬發現以 ZF 為 MIMO 偵測的方法，再經過解碼器輸出後，其效能和 MMSE 偵測非常接近，故在硬體設計中，吾人採用 ZF 為 MIMO 偵測器(Detector)，並做了 ZF 之等化矩陣為奇異矩陣時之處理，及將 ZF 之等化矩陣運算時所產生之除法和 channel state information (CSI)所產生之除法轉換成乘法併到其它硬體方塊(Block)內以消除除法運算；針對 Viterbi 解碼器，吾人使用基數-4 之相加-比較-選擇單元(Radix-4 ACSU)搭配 Radix-4 之分支計量值單元(BMU)及 Radix-4 之預處理-存活記憶單元(Pre-SMU)和 Radix-16 之存活記憶單元(SMU)，以達到降低時脈速度或提昇資料的傳輸速度之目的。最後再根據在西元 2005 年 3 月由 TGn Sync 所提出的 802.11n 的草案[8]來設計並以 Verilog 設計出低複雜度的接收機。值得注意的是，本篇論文中所使用的 Radix-4 ACSU 在[17]也有類似的作法。

本篇論文的結構如下: 第二章將簡介 IEEE 802.11n 系統，包括一般性之概論、此系統的數學模型建立、主要的 BICM 的架構和在 TGn Sync 所提出的 11n 草案中的編碼過程。第三章則說明吾人所提出的系統主要的 MIMO 偵測演算法(ZF、MMSE 及 VBLAST)及 Viterbi 解碼(decoding)和其系統模擬比較。第四章介紹吾人所實現的低複雜度之 802.11n 接收機，主要分成二部分，一部分為 MIMO 偵測，另一部分則為 Viterbi 解碼器。最後總結在第五章中。

第2章 IEEE 802.11n 系統介紹

2.1 IEEE 802.11n 系統概論

802.11 是國際電機電子協會(IEEE)裡面負責規劃無線區域網路(WLAN)規格的組織，主要目的是要訂出符合大眾需求並提供一個規範準則給各家生產設備廠商作為標準。在 802.11 家族裡面，目前已經規範完成的標準包含 a,b,e,g,以及 i。而截至目前，正在規劃且尚未完成的 802.11n 是一個訴求高速傳輸的下一代無線網路傳輸技術，比目前最快的 a,g(54Mbps)系列高出兩倍以上的傳輸率，最快甚至可達 600Mbps。而之所以可以提升如此高的傳輸速度主要就是加入了多天線(MIMO)的架構。在規格訂定過程中，現在有兩個主要的大團體在競爭規格，一個是 TGn Sync[8] ，另一個則是 WWiSE。在本論文中，我們以 TGn Sync 的架構為基礎，在本節將介紹 TGn Sync 所提出的系統架構，而遵循的規格是以 2005 年 3 月所提出的架構為主。

Feature	Mandatory	Optional
Number of Spatial Streams	1 and 2	3 and 4
Number of Transmit Antennas	2	Greater than 2
Channelization bandwidth	20MHz	40MHz
Number of Occupied Subcarriers	56 in 20MHz	114 in 40MHz
Number of Data Subcarriers	52	108
Number of Pilot Subcarriers	4	6
Modulation Order	BPSK, QPSK, 16-QAM, 64-QAM	256-QAM
Code Rate	1/2, 2/3, 3/4, 5/6	
Guard Interval	800ns	400ns
Convolutional Coding	R=1/2, K=7, ($g_1=133_8$, $g_2=171_8$)	
LDPC		TX and RX Optional
TX Beamforming		TX and RX Optional

表 2-1 802.11n 實體層(PHY)需求

在 TGn Sync 所提出的 802.11n 草案中，其特色主要有下列幾點：

- 使用較大的頻寬(40MHz)
- 以 MIMO-OFDM 為主
- 空間多工(SDM)或時空區塊編碼(STBC)
- 空間延展(Spatial Spreading)與傳輸端的波束成型(Beamforming)
- 進階編碼(LDPC)
- 延伸的調變與編碼方式(Extended Modulation and Coding Scheme)

觀察表 2-1，TGn Sync 主要以兩根天線且 20MHz 頻寬為必備模式，與早期的單天線傳輸使用相同的頻寬。操作的頻帶在 2.4GHz 以及 5GHz，其傳輸端架構如下圖：

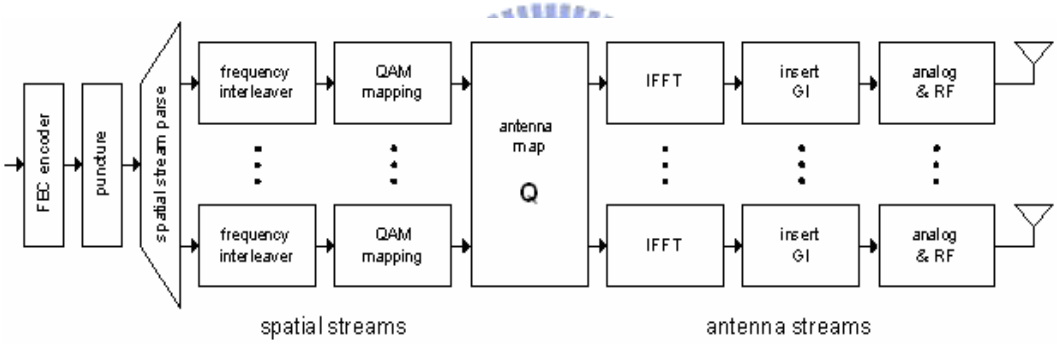


圖 2-1 TGN Sync 傳送端方塊圖

由圖可知，其錯誤更正碼架構是採用單一編碼器。對傳統的迴旋碼(Convolution Code)而言，在這邊採用的主要編碼率為 1/2。而這些編碼過後的位元資料可以經由壓縮(puncture)而產生 2/3、3/4 或 5/6 共四種編碼率。而對 LDPC 碼來說，則不應用其它的編碼率。被壓縮後的位元透過空間資料流分離器(spatial stream parser)將資料分成 N_{SS} 個空間資料流。對每個資料流都會成塊地做交錯(interleaving)而會對映到一個 OFDM symbol 的大小。這樣的交錯是將不同子載波間的位元交錯，所以又稱為頻域交錯器(Frequency Interleaver)，可避免 Burst Error 對接收端解碼時的影響。這頻率交錯器和在 802.11a[9]中是十分相似的。結合空

間資料流分離和頻率交錯便成了空時(space-time)交錯。交錯後的位元將會對映到 QAM 的星狀圖上，這對映可能是 BPSK、QPSK、16QAM、64QAM 或是 256QAM。下一步是天線對應轉換(antenna map transformation)。這轉換是將 N_{SS} 個空間資料流對應到 N_{Tx} 個傳送天線資料流。注意，在 MIMO 的傳送下必須要 $N_{SS} \le \min\{N_{Tx}, N_{Rx}\}$。特別地，我們總是讓 $N_{SS} \le N_{Tx}$。將 N_{SS} 資料對應到 N_{Tx} 根天線的動作可以由一個矩陣 Q 表示。Q 轉換模式總共有三種，以下將個別介紹。本論文所模擬實現的為直接對映(direct map)轉換架構，且本章主要將介紹在本論文中所用到的元件，這包括了編碼、壓縮、空間分離及交錯。

2.1.1 直接對映(direct map) 的 MIMO 架構

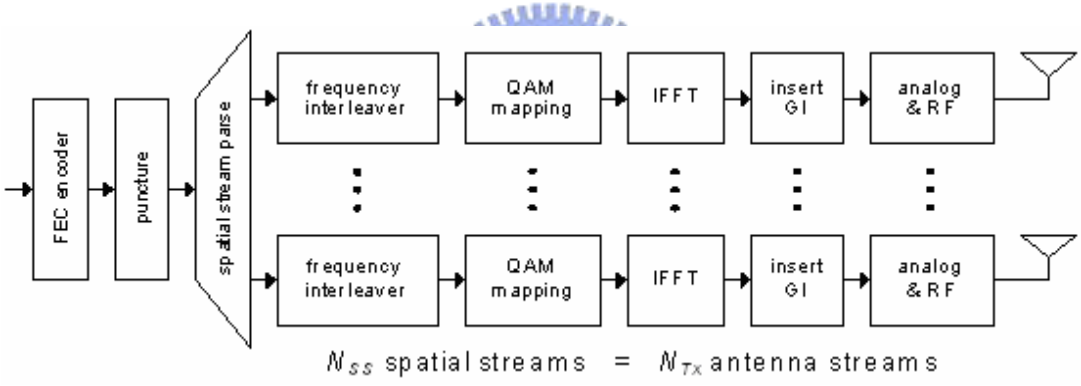


圖 2-2 直接對映轉換的 MIMO 傳送方塊圖

此架構是最簡單的對應模式。在此架構中，轉換矩陣 **Q** 只是一個單位矩陣的對應，也就是 $\mathbf{Q} = \left[\mathbf{I}_{N_{Tx} \times N_{Tx}}\right]_{N_{SS}}$。每個空間資料串會對應到每根天線上，為一對一的對應。由於 N_{ss} 小於 N_{Tx}，所以只有 **Q** 的前 N_{ss} 個列會被使用到，而剩下的 N_{Tx}-N_{ss} 個列則都設為零。

2.1.2 空間延展(Spatial Spreading)

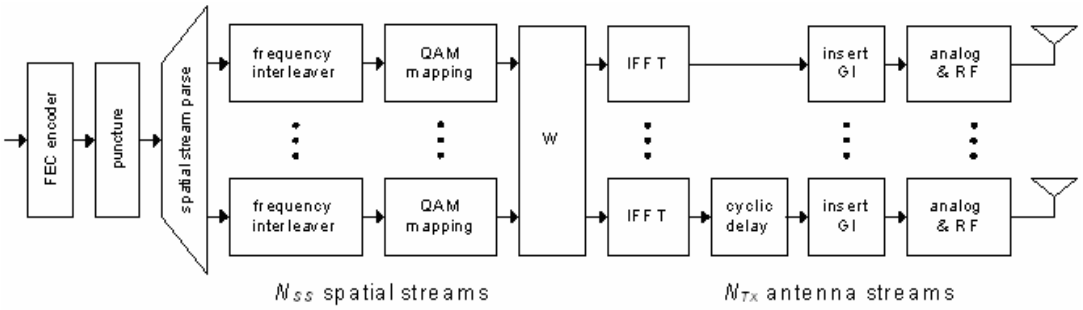


圖 2-3 空間延展搭配循環延遲傳送方塊圖

基本的空間延展概念是使用一正交矩陣 **W** 的前 N_{SS} 列，而此矩陣在每個子載波間都是保持不變的。通常在傳輸天線數目為 2 或 4 的時候，我們會使用 Walsh 矩陣，若傳輸天線數目為 3 的時候，則使用傅立葉矩陣。

在 N_{Tx} 為 2 或 4，轉換矩陣如下：

$$\mathbf{W}_{2\times 2}=\frac{1}{\sqrt{2}}\begin{bmatrix}+1 & +1\\ +1 & -1\end{bmatrix}\qquad \mathbf{W}_{4\times 4}=\frac{1}{2}\begin{bmatrix}+1 & +1 & +1 & +1\\ +1 & -1 & +1 & -1\\ +1 & +1 & -1 & -1\\ +1 & -1 & -1 & +1\end{bmatrix} \tag{2-1}$$

在 N_{Tx} 為 3，轉換矩陣如下：

$$\mathbf{W}_{3\times 3}=\frac{1}{\sqrt{3}}\begin{bmatrix}+1 & +1 & +1\\ +1 & e^{j2\pi/3} & e^{-j2\pi/3}\\ +1 & e^{-j2\pi/3} & e^{j2\pi/3}\end{bmatrix} \tag{2-2}$$

使用 Wlash-Hadamard 與傅立葉矩陣的用意在於其本身的正交性，也就是 $\mathbf{W}^{T}\times\mathbf{W}=\mathbf{I}$。假設傳送端送出向量 $\mathbf{x}_{2\times 1}$，經由轉換矩陣分配到四根天線上，所以 **W** 的大小為 4×4。根據定義，我們只會用到 **W** 的前兩列，標示為 $\mathbf{W}_{r}$，所以送出的訊號為 $\mathbf{y}_{4\times 1}=\mathbf{W}_{r4\times 2}\mathbf{x}_{2\times 1}$。在接收端，利用其正交性，只要再乘上 $\mathbf{W}_{r}$ 的轉置矩陣便可以將原始訊號求出，也就是 $\hat{\mathbf{x}}=\mathbf{W}_r^T\mathbf{y}=\mathbf{W}_r^T\mathbf{W}_r\mathbf{x}=\mathbf{x}$。然而在使用基本的空間延

展時，傳輸端會產生一個波束成型(Beamforming)的效應，此效應對系統而言是我們不想要的。為了減輕此效應的影響，在傳送端採用循環延遲(Cyclic Delay)，也就是每根天線傳送訊號的時間會不同。第一根天線是沒有延遲的，而剩下的天線依照順序，其延遲時間會有一個線性關係存在。這種延遲的方式不止可以減輕波束成型的效應，也可以增加系統的效能(Transmit Diversity)。圖 2-3 可看成是圖 2-1 的一種變形，但是在數學上可以使他們相等。我們知道在時域的延遲等於在頻域乘上一個相位的平移，因此只要讓 $\mathbf{Q}=\mathbf{\Phi W}_r$，則圖 2-3 就可以表示成如圖 2-1 之結構，其中 $\mathbf{\Phi}$ 是一個 $N_{\mathrm{Tx}}\times N_{\mathrm{Tx}}$ 的對角矩陣，如(2-3)所示。在頻域乘以 $\exp(-j2\pi k\Delta_F D)$，等效在時域做一個 D 時間的循環延遲。所以第 i 根天線會有 $(i\text{-}1)D$ 的循環延遲時間。

$$\mathbf{\Phi}^{(k)} = \mathrm{diag}\left(1, \exp(-j2\pi k\Delta_F D), \ldots, \exp(-j2\pi k(N_{Tx}-1)\Delta_F D)\right) \tag{2-3}$$

2.1.3 傳送端波束成型(Beamforming)

傳送端波束成型的目的是調整天線的對應，使得接收端的每個空間資料串可以接收到最大的能量。要產生波束成型矩陣的方式很多，在此我們只簡要介紹基於通道 Singular value decomposition(SVD)下的一個最佳方式。在此架構下，轉換矩陣 **Q** 會根據目前 MIMO 通道狀況做設計。最佳的轉換矩陣就是通道在 SVD 下的右奇異向量(Singular Vectors)。假設通道 **H** 代表訊號在頻域遭遇到的通道，是一個 $N_{\mathrm{Rx}}\times N_{\mathrm{Tx}}$ 的矩陣。則通道的 SVD 分解如下所示：

$$\mathbf{H}_{N_{Rx}\times N_{Tx}} = \mathbf{U}_{N_{Rx}\times N_{Rx}}\mathbf{D}_{N_{Rx}\times N_{Tx}}\mathbf{V}^{H}_{N_{Tx}\times N_{Tx}} \tag{2-4}$$

其中 $\mathbf{D}=\mathrm{diag}\left(\sigma_0,\ldots,\sigma_{N-1}\right)$，是一個奇異數的對角矩陣。而 **U** 與 **V** 皆為正交矩陣。假設傳送端送了一個向量 **x**，經過轉換矩陣可以得到 **Vx**。於是在接收端可以收到：

$$\begin{aligned}\mathbf{y} &= \mathbf{HVx} + \mathbf{n} \\ &= \mathbf{UDV}^H\mathbf{Vx} + \mathbf{n} \\ &= \mathbf{UDx} + \mathbf{n}\end{aligned}$$

(2-5)

使用 SVD 的好處是可以藉由適應性調變(Adaptive Modulation)增加傳送效率，也就是不同的資料串可以根據通道狀況做調整，使傳輸效率提高。在接收端可以乘上 $\mathbf{U}^H$ 以取得我們想要的資料。

$$\begin{aligned}\mathbf{y} &= \mathbf{UDx} + \mathbf{n} \\ \mathbf{U}^H\mathbf{y} &= \mathbf{Dx} + \mathbf{U}^H\mathbf{n}\end{aligned}$$

(2-6)

其中奇異數矩陣 **D** 代表目前通道的狀況，我們可以在 SNR 高的通道傳送較高的資料量，而 SNR 低的通道則傳送比較少的資料量。

MIMO Modes	**Characteristic**	**Extent of TX adaptation**
Basic MIMO (mandatory)	• Antenna mapping: direct map or spatial spreading	• No adaptation is performed in the spatial domain • MCS selection (adaptation) is based upon either: (1) binary feedback (i.e. ACK/noACK), or (2) recommendation from the RX, or (3) availability of CSI at the TX
TX Beamforming (Optional)	• All spatial streams are spatially shaped via a beamforming matrix that may be different for each subcarrier. • The basic MCS set and extended MCS set is used, which would support different data rates across spatial streams.	• CSI is required at the TX • Sounding packet from the recipient to the initiator is required to estimate the CSI at the transmitter • RF calibration at transmitter side is required since reciprocity is used to perform TX beamforming • It is not mandated at devices which don't support Tx BF that they should be involved in calibration sequence.

表 2-2 MIMO 傳輸模式

表 2-2 是針對上述幾種 MIMO 的傳輸模式做一個整理，主要分為兩種模式。一個是基本的 MIMO 傳輸，採用直接對應或是空間延展的傳輸架構，並且可藉由接收端的回饋(Feedback)訊號或是由傳送端自己估測得到的通道狀態資訊(CSI)來做調變的選擇；另一種模式是波束成型，採用延伸的調變模式，可以使不同的空間資料流採用不同的傳輸率。

2.1.4 不同頻寬模式下之 Tone 配置介紹

在 20MHz 的頻寬模式下，每個 OFDM 符元包含 56 個 tones，其中 52 個是 data tones，4 個是 pilot tones，其配置方式大致與 IEEE 802.11a[9]相同，只是在兩邊各多加了兩個 data tones。在 40MHz 頻寬下，比傳統單天線的頻寬多了一倍，所以 tone 的配置也會不同。如下所示：

$$Nulled\ Tones = \{-64 \ldots -59, -1, 0, +1, +59 \ldots +63\}$$

$$Populated\ Tones = \{-58 \ldots -2, +2 \ldots +58\}$$

$$Pilot\ Tones = \{-53, -25, -11, +11, +25, +53\}$$

$$Data\ Tones = \{Populated\ Tones\} - \{Pilot\ Tones\}$$

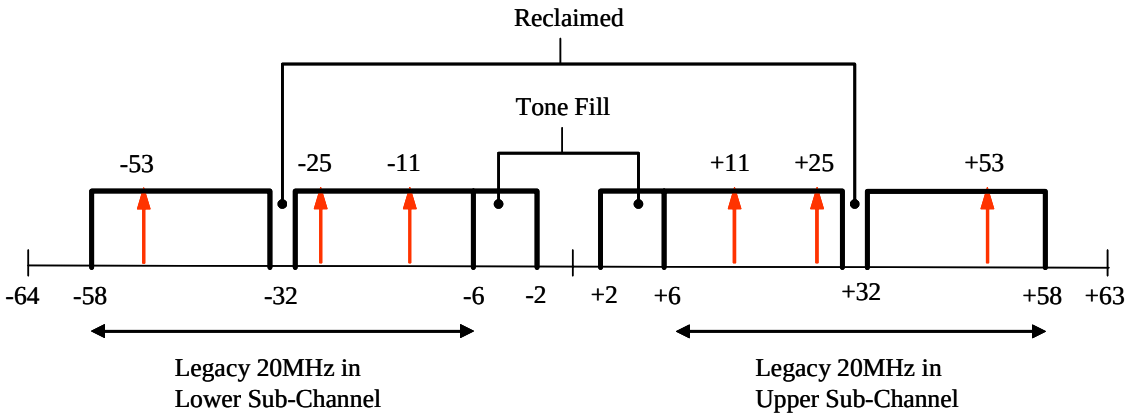


圖 2-4 40MHz 頻寬的 Tone 配置

圖 2-4 是採用 40MHz 的頻寬時，Data Tones 與 Pilot Tones 的所在位置。基

本上主要是由兩個舊的 20MHz 頻寬所組成，但是 Pilot Tone 的位置與數目有些許調整，例如舊的 20MHz 的 Tone 是由-58 到-6 以及+6 到+58，現在則變成-58 到-2 以及+2 到+58，因此多了四個 Data Tones；另外，之前+39 以及-39 的位置是擺放 Pilot Tones，現在則換成 Data Tones，如此一來增加了資料的傳輸量。

2.1.5 IEEE 802.11n Preamble 格式介紹

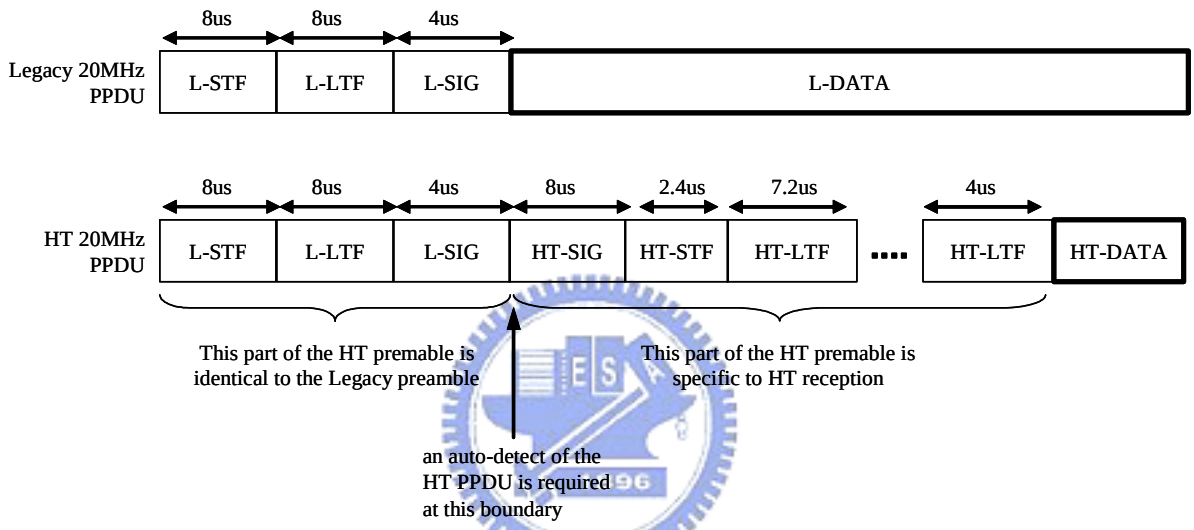


圖 2-5 20MHz 的 Preamble 格式

Preamble 的功能包含以下：

1. 封包偵測(Start-of-Packet detection)
2. 自動增益控制(AGC)
3. 粗略頻率偏移估計(Coarse Frequency Offset Estimation)
4. 粗略時序估計(Coarse Timing Offset Estimation)
5. 精準時序估計(Fine Timing Offset Estimation)
6. 精準頻率偏移估計(Fine Frequency Offset Estimation)
7. 通道估計(Channel Estimation)

如圖 2-5 所示，為了使 802.11n 的實體層封包可以相容於傳統的 802.11a 與 802.11g，因此在封包的一開頭會加入舊式的 preamble(L-STF, L-LTF, L-SIG)。如

此一來，當傳統單天線的接收機收到 11n 的封包時，也可以順利的解開 Legacy Signal field (L-SIG)並作適當的處理。

相對於 802.11a 與 802.11g，以下我們定義 11n 的傳送接收為高吞吐量(High Throughput)。通常高吞吐量接收機並無法事先預知其接收到的封包是傳統單天線的封包或是 11n 的封包。因此，高吞吐量接收機必須要能判別在 L-SIG 之後是 L-DATA 或是 High Throughput Signal field (HT-SIG)。在此，Preamble 的設計提供了兩個方案使得高吞吐量接收機可以判別出 L-DATA 或是 HT-SIG：(1)將 HT-SIG 以 BPSK 星狀圖傳送，且其對應的軸為正交軸(Quadrature Axis)，如圖 2-6。(2)從 L-SIG 到 HT-SIG 會反轉 Pilot 的極性。如此一來，高吞吐量接收機便可以輕易的分辨出接收到的封包是單天線系統的封包或是高吞吐量傳輸的封包。

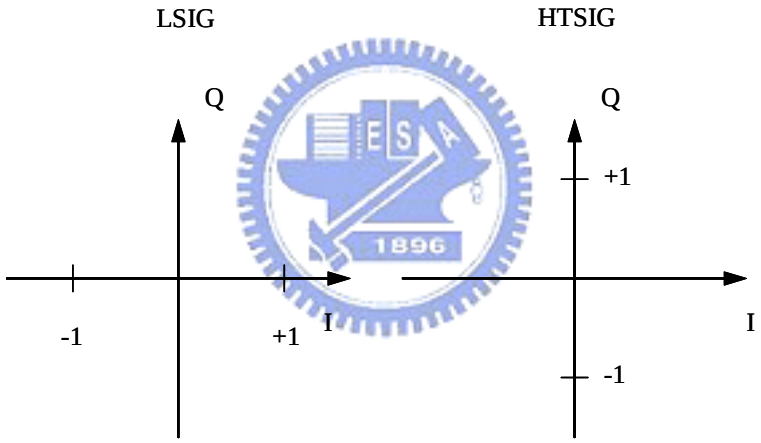


圖 2-6 L-SIG 與 HT-SIG 的星狀圖對應

一個高吞吐量接收機必須要能完成前頁所述之功能 1 到 7，才能成功的解出封包。功能 1 到 4(或 5)是由 L-STF 完成，而功能 5 到 7 是由 L-LTF 完成。由於 L-STF 無法提供足夠且正確的資訊來操作 MIMO AGC，因此藉由 HT-STF 彌補這個缺點。換句話說，HT-STF 只是單純為了做 MIMO AGC 所設計。L-STF 與 L-LTF 在以前的單天線系統已經是大家所熟知的，可以在[9]找到相關資訊，其功能與細節便不再贅述。以下我們將從 HT-SIG，HT-STF 與 HT-LTF 開始討論。

- HT-SIG

由圖 2-7 可知，此欄位包含許多 HT 實體層的資訊，包含調變資訊(Modulation Coding Scheme, MCS)，進階編碼以及頻寬選擇等。對於實體層的訊號處裡而言，都是非常重要的，詳細功用可參考[8]。

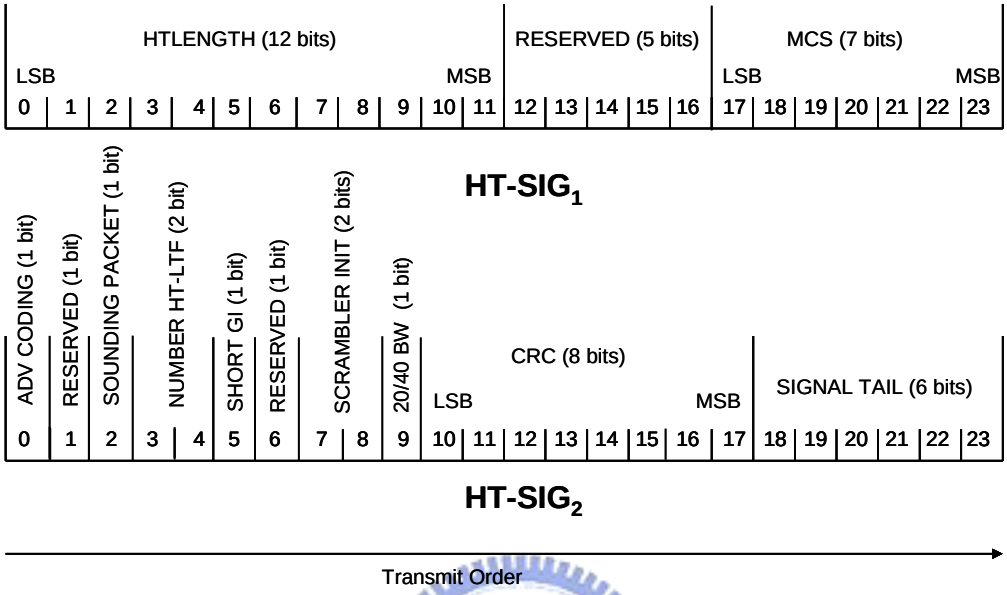


圖 2-7 HT-SIG 格式與各個欄位代表的意義

- HT-STF

高吞吐量短訓練符元，此區塊純粹是為了做自動增益控制(AGC)的微調。藉由 HT-STF 估計接收訊號的強弱，提供資訊給 AGC。而 AGC 電路會根據此訊息自動增加或衰減增益，並透過基頻濾波器來避免 A/D 的飽和，維持適當的基頻信號準位再送給基頻處裡器。為了使估計出來的功率夠準確，且功率的波動(Fluctuation)最小化，在這邊採用頻率交錯(Tone-Interleaving)的方式。以圖 2-9 的頻率交錯為例子，較粗的線代表有放 Pilot 的 Tones，所以不同天線間，放資料

$$HTS_{-26,26} = \frac{1}{\sqrt{2}} \begin{cases} \{0, 0, 1+j, 0, -1-j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, \\ 0, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 0\} & N_{Tx}=1 \\ \{0, 0, -1-j, 0, 1+j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, \\ 0, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, 1+j, 0, -1-j, 0, \} & N_{Tx}=2 \\ \{0, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, \\ 0, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, 1+j, 0, -1-j, 0, 0\} & N_{Tx}=3 \\ \{0, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, -1-j, 0, -1-j, 0, \\ 0, 0, -1-j, 0, -1-j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, 1+j, 0, 1+j, 0, -1-j, 0, -1-j, 0, 0\} & N_{Tx}=4 \end{cases}$$

圖 2-8 在 20MHz 不同傳送天線所採用的 STF

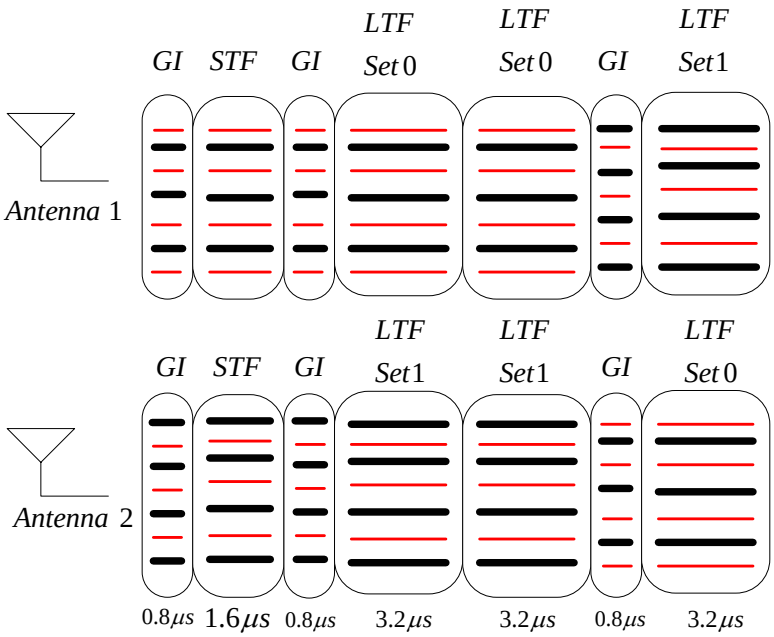


圖 2-9 兩根天線的頻率交錯示意圖

的頻率會互相交錯。因此，在接收端經過 FFT 後，不同天線的 Pilot 並不會互相干擾。下表是搭配圖 2-9，說明在 Set0 到 Set1 放置 LTF 的頻率交錯位置。

Nss	Set 0	Set 1	Set 2	Set 3
1	[-28:1:-1] [1:1:+28]			
2	[-28:2:-2] [2:2:28]	[-27:2:-1] [1:2:27]		
3	[-28:3:-1] [2:3:26]	[-27:3:-3] [3:3:27]	[-26:3:-2] [1:3:28]	
4	[-28:4:-4] [1:4:25]	[-27:4:-3] [2:4:26]	[-26:4:-2] [3:4:27]	[-25:4:-2] [4:4:28]

表 2-3 頻率交錯以分組表示

- HT-LTF

高吞吐量長訓練符元，主要用途是作通道估測(Channel Estimation)，也可以用來做頻率偏移的補償。在通道估測時，傳送端也是採用頻率交錯的方式，避免

天線之間的互相干擾。以兩根天線，20MHz 的模式為例子，如圖 2-9，其中第一根天線的 Set0 重覆了兩次，此目的是為了在接收端能夠對兩次估到的相同通道作平均。但是為了避免 LTF 浪費傳輸率，藉由模擬結果發現，在 Set1 之後剩下一個 OFDM 符元的這種模式依然可以維持我們想達到的封包錯誤率(PER)，並且估計出來的通道也不會誤差太多。隨著天線數目的增加，LTF 的數目也要隨著增加，並且在不同天線所傳送的頻率會互相錯開，也就是可以在不受其他天線干擾的情形下估出完整的通道。表 2-4 是不同天線間採用的 LTF 格式，之所以有不同的序列是為了要使其時域的鋒均值比(Peak-to-Average Power Ratio, PAPR)最小。

N_{Tx}=1	$HTL_{-28:28}$=[-1, -1, -1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1, 1, -1, 1, 0, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, 1, -1, -1, -1]
N_{Tx}=2	$HTL_{-28:28}$=[-1, -1, 1, 1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, 1, -1, -1, 1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 1, 0, -1, -1, -1, 1, -1, 1, -1, -1, 1, 1, -1, 1, -1, -1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1]
N_{Tx}=3	$HTL_{-28:28}$=[1, -1, -1, -1, 1, 1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, -1, -1, -1, 0, 1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, -1, 1, -1, -1, 1, -1, -1, -1, -1, 1, -1, 1, 1, -1, -1, 1]
N_{Tx}=4	$HTL_{-28:28}$=[1, 1, 1, 1, -1, 1, -1, 1, -1, 1, 1, 1, 1, 1, -1, 1, -1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 0, -1, 1, 1, -1, -1, -1, 1, 1, 1, -1, -1, -1, 1, 1, -1, 1, 1, -1, -1, -1, 1, 1, -1, -1, 1, -1, -1, 1]

表 2-4 在 20MHz 不同傳送天線所採用的 LTF

2.2 BICM MIMO-OFDM 系統

2.2.1 BICM MIMO-OFDM 系統架構

802.11n 規範一個 BICM MIMO-OFDM[22]的系統，如圖 2-10 所示，其中包括了 N_t 的傳送天線及 N_r 的接收天線。傳送端包括了迴旋編碼器(FEC encoder)，位元交錯器(π)，將位元對映到 2^M 個 QAM 訊號的對映(Mapping)，其中 M 表示一次對映所需的傳送位元數，以及 OFDM 系統中 IFFT(size N_C)及循環前序(cyclic prefix)的插入。接收端則主要由 OFDM 系統的 FFT 及 CP 去除、符元反對映(demapping)計算位元計量值(bit metrics)、反位元交錯器(π^{-1})及 Viterbi 解碼器所構成。

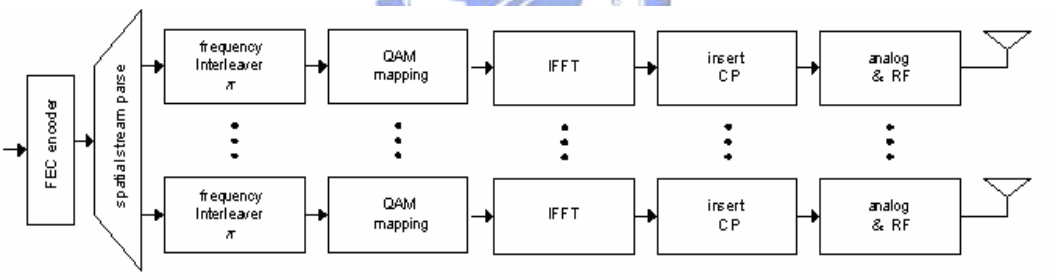


圖 2-10 BICM-MIMO-OFDM 傳送端方塊圖

資料位元一開始經過一個編碼率(coding rate)為 R_C 的迴旋編碼器。這些位元再通過一個位元上的(bitwise)交錯器產生新的位元。這些經過編碼且交錯器的位元資料接著會被以大小為 $N_t \times M \times N_c$ 分塊，成為多天線的 OFDM symbol。在這塊的資料中又以大小為 $M \times N_c$ 分群，對應到 $1 \times N_c$ 複數訊號的向量，令此向量為 $S(k,n)$，$k = 1,2,...N_c$。之後對應至相同 OFDM symbol 的訊號便通過 IFFT 及加

CP 方塊。每根多天線的 OFDM symbol 的資料率為 $R_c \times N_t \times M \times N_c$。

我們假設 MIMO 通道為頻率選擇衰落通道(frequency selective channel)，並有適當長度的 CP 及完美的同步。在頻域上，在第 n 個 OFDM symbol 的第 k 個子載波(subcarrier)中，假設傳送訊號為 $X(k,n)$ ，我們將會收到 $1\times N_c$ 的複數訊號 $Y(k,n)$ ， $k=1,2,...N_c$ 。

$$Y(k,n)=H(k,n)S(k,n)+N(k,n) \tag{2-7}$$

這裡 $N(k,n)$ 是指 Additional White Gaussian Noise (AWGN)而 $H(k,n)$ 則是指在第 n 個 OFDM symbol 中的第 k 個子載波中的通道。$H(k,n)$ 的第 (i,j) 項則為從第 i 根傳送天線到第 j 個接收天線的增益(gain)。另外在理想的交錯(interleaving)下，我們可假設 $H(k,n)$ 對不同的 k 和 n 而言是獨立且相等的分佈(iid)的。而為了表示方便，在本論文之後的數學模式中都將 n 省略，也就是(2-7)可表示為(2-8):

$$Y(k)=H(k)S(k)+N(k) \tag{2-8}$$

2.2.2 802.11n 編碼器及交錯器（Interleaver）

2.2.2.1 編碼(Encoding)

在 11n 所用的是使用限定長度(constrain length)為 7 的迴旋碼，編碼多項式分別為 $g_0=133_8$ 及 $g_1=171_8$ ，如圖 2-11 所示。

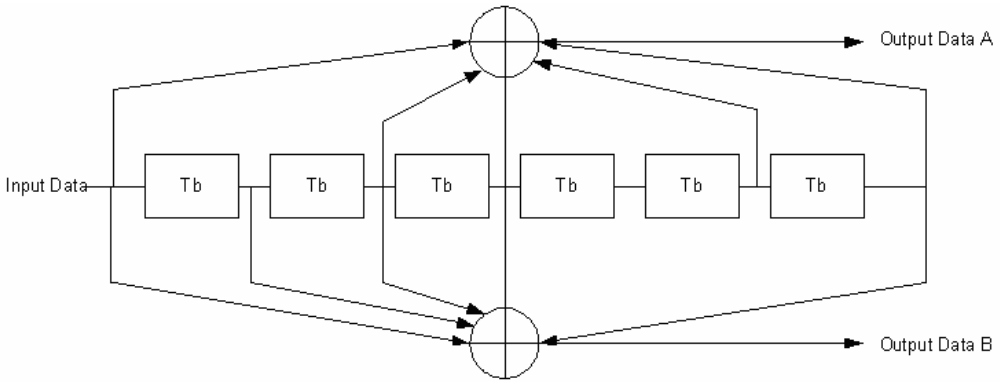


圖 2-11 迴旋編碼器

2.2.2.2 壓縮(Puncture)

資料位元在經過編碼後的編碼率是 1/2，如果想要有更高的編碼率則藉由在將固定順序的編碼後的資料位元打掉不送，而在解碼器時便在相同位置補上沒有意義的位元而使得編碼率變高。在 TGn Sync 之提案中，共有 1/2、2/3、3/4、5/6 四種速率可以選擇，其中除了 5/6 是 11n 所新增的，其它的速率都和 11a 相同。圖 2-12 壓縮流程圖為編碼率為 2/3、3/4 及 5/6 的例子。

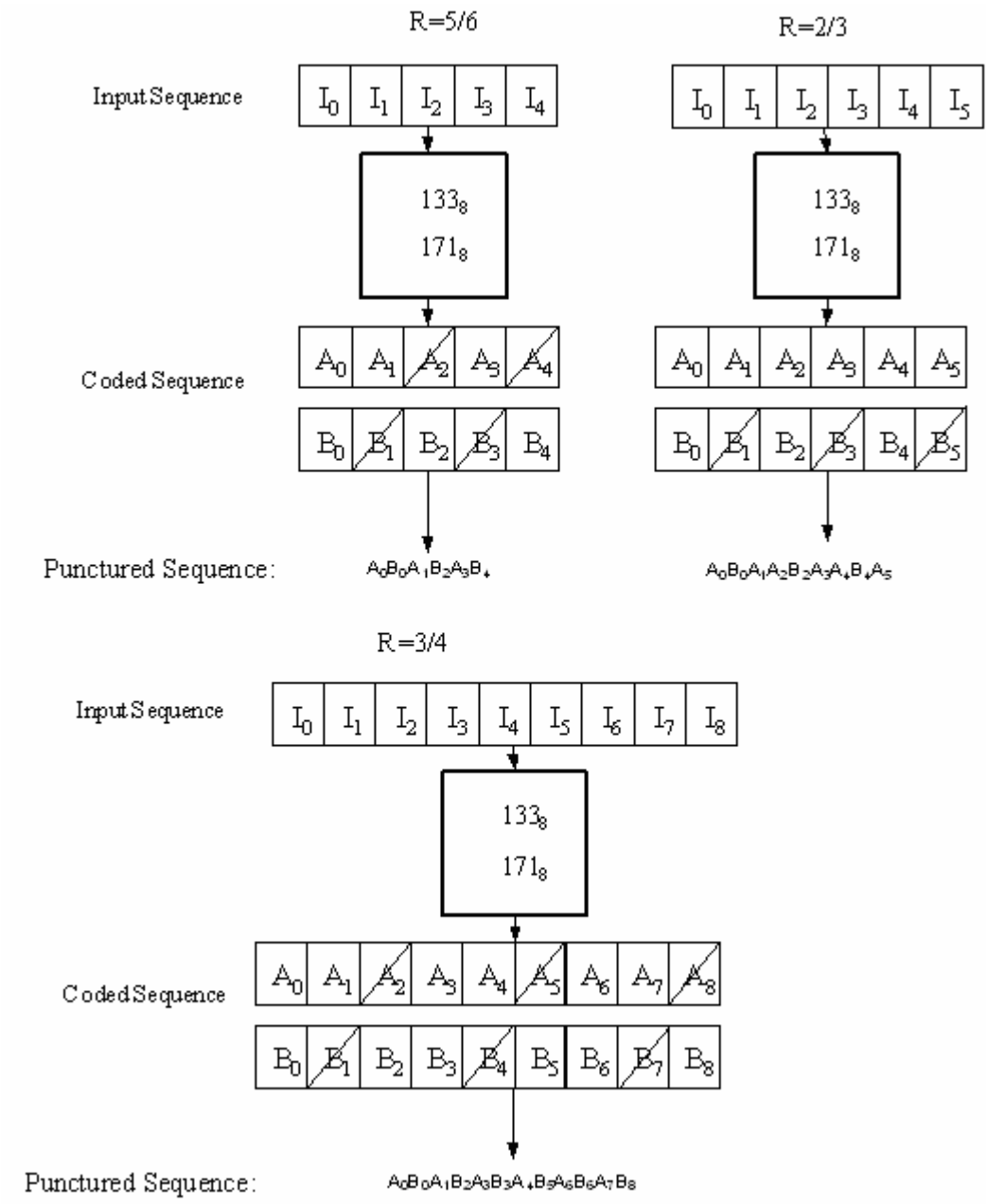


圖 2-12 壓縮流程圖

2.2.2.3 交錯器(Interleaver)

資料在經過編碼及壓縮後，會交錯地送到各個天線及子載波上。而此交錯器共有二個步驟:

1. 空間分離(space parsing):

$$s = \max\{N_{BPSC} / 2, 1\} \tag{2-9}$$

N_{BPSC} 是指在每個子載波上的位元數(number of bits per subcarrier)也就是使用的 QAM 的等級(order)，例如當 BPSK 時，N_{BPSC}=1，而 QPSK 時是 2。在壓縮後的資料會以大小為 s 的方塊，以 round-robin 的方式從第一根天線開始地放到各天線上去。

2. 頻率交錯器(frequency interleaver)

所有被分到各個天線上的資料位元都會被獨立的交錯到不同的子載波上去，而每個天線都有各自己的交錯器，大小為一個 OFDM symbol 的長度，N_{CBPS}。交錯器是根據 802.11a 中所定義的交錯器而加以修改成適用於多天線及 40MHZ 的傳輸。如圖 2-13 所示。

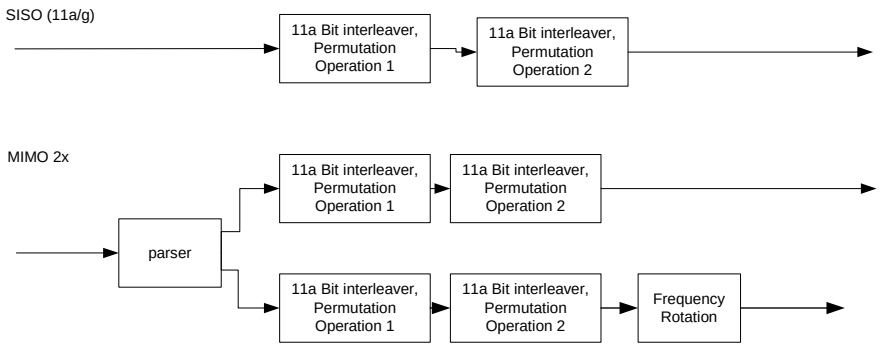


圖 2-13 頻率交錯器

此交錯器分為三步驟。第一次對調(permutation)為保證相隣的位元能對映到不相隣的子載波上。第二次對調則是能讓資料位元輪流地對映到星狀點上的 MSB 及 LSB 上。第三次對調保證讓資料位元透過頻率旋轉(frequency rotation)在所有的天線之間達到較好的頻率-空間分集。

基本的交錯器陣列有 N_{row} 行及 N_{column} 列，且頻率旋轉的基底則表示為 N_{rot} 。這些參數如表 2-5 及表 2-6 所示。

	N_{row}	N_{column}	N_{rot}
20MHZ	13	$4N_{BPSC}$	11
40MHZ	18	$6N_{BPSC}$	29

表 2-5 頻率交錯參數表

不同的天線則有頻率旋轉數，列表如下:

Frequency Rotation	Total Number of Streams			
	1	2	3	4
1st stream	0	0	0	0
2nd stream		$2N_{rot}$	$2N_{rot}$	$2N_{rot}$
3rd stream			N_{rot}	N_{rot}
4th stream				$3N_{rot}$

表 2-6 頻率旋轉參數表

假設在第一次對調之前資料的索引(index)為 k ，而在第一次對調之後第二次對調之前的索引為 i ， j 則為第二次對調及第三次對調之間的索引，r 是第三次對調之後 QAM 對映之前的索引。則我們可以將這三次對調用數學表示:

第一次對調是根據以下規則所定義:

$$i = N_{row} \times \left(k \bmod N_{column}\right) + \text{floor}\left(k / N_{column}\right) \quad k = 0,1,\ldots,N_{CBPS} - 1 \tag{2-10}$$

第二次對調是根據以下規則所定義:

$$j = s \times \text{floor}\left(i / s\right) + \left(i + N_{CBPS} - \text{floor}\left(N_{column} \times i / N_{CBPS}\right)\right) \bmod s \quad i = 0,1,\ldots,N_{CBPS} - 1 \tag{2-11}$$

其中 $s = \max\{N_{BPSC} / 2, 1\}$。

第三次對調是根據以下規則所定義:

$$r = \left(j - \left(\left(2 \times i_{ss}\right) \bmod 3 + 3 \times \text{floor}(i_{ss} / 3)\right) \times N_{rot} \times N_{BPSC}\right) \bmod N_{CBPS} \quad j = 0,1,\ldots,N_{CBPS} - 1 \tag{2-12}$$

$i_{SS} = 0,1,\ldots,N_{SS} - 1$，代表天線的索引，第一根天線 i_{SS} =0。

反交錯器(deinterleaver) 則相對應地分成三次對調來定義:

假設在第一次對調之前資料的索引(index)為 r，而在第一次對調之後第二次對調之前的索引為 j，i 則為第二次對調及第三次對調之間的索引，k 是第三次對調之後 QAM 的反對映之前的索引。則我們可以將這三次對調用數學表示:

第一次對調是根據以下規則所定義:

$$j = \left(r + \left(\left(2 \times i_{ss}\right) \bmod 3 + 3 \times \text{floor}(i_{ss} / 3)\right) \times N_{rot} \times N_{BPSC}\right) \bmod N_{CBPS}, \qquad r = 0,1,\ldots,N_{CBPS} - 1 \tag{2-13}$$

第二次對調是根據以下規則所定義:

$$i = s \times \text{floor}\left(j / s\right) + \left(j + \text{floor}\left(N_{column} \times j / N_{CBPS}\right)\right) \bmod s, \qquad j = 0,1,\ldots,N_{CBPS} - 1 \tag{2-14}$$

第三次對調是根據以下規則所定義:

$$k = N_{column} \times i - \left(N_{CBPS} - 1\right) \text{floor}\left(i / N_{row}\right), \qquad i = 0,1,\ldots,N_{CBPS} - 1 \tag{2-15}$$

(2-15)、(2-14)、(2-13)分別是(2-10)、(2-11)、(2-12)的反對調。

第3章偵測(Detection)和解碼(Decoding)

現存的系統中大多以 MMSE 及 VBLAST 為 MIMO 偵測的方法，再經過解碼器後輸出。本論文中由模擬發現 ZF 為 MIMO 偵測的方法，再經過解碼器後輸出，亦有和 MMSE 偵測非常接近之效能(Performance)，而 VBLAST 之計算較複雜，且效能並無顯著的提昇，故在硬體設計中，吾人採用 ZF 為 MIMO 偵測器(Detector)，並做了 ZF 之等化矩陣為奇異矩陣時之處理，及將 ZF 之等化矩陣運算時所產生之除法和 CSI 所產生之除法轉換成乘法併到其它硬體方塊內以消除除法運算。如此一來則可達到降低計算的複雜度，進而達到節省面積降低成本的目的。針對 Viterbi 解碼器電路，吾人使用 Radix-4 之 Viterbi 解碼器架構，如 4.2.6.4 所述，其可在固定的資料傳輸速度下，降低時脈速度以達到節省消耗功率之目的。在 3.1 中，吾人將介紹傳統的偵測方式，包含 ZF、MMSE 以及 VBLAST。在 3.2 中，吾人將針對 Viterbi 解碼器的演算法(algorithm)部份加以闡述。

3.1 ZF、MMSE 及 V-BLAST 偵測

在上一章中，我們將 BICM MIMO OFDM 系統如(2-8)所示，傳送訊號 $S(k)$ 經過通道 $H(k)$ 後，加上高斯雜訊 $N(k)$ 得到接收訊號 $Y(k)$。在空間多工的應用中，MIMO 偵測(detection)就是要在接收端將其它根天線的訊號消除，從 $Y(k)$ 去推算出 $S(k)$ ，在現行的方法中，大致可以分成三大類的偵測方法，一為最大可能性解碼為基礎的 Sphere decoding[13]-[14]，二為以 Bell lab 所提出來 MMSE 加 SIC 的 VBLAST[4]-[6]，三則是以 ZF 及 MMSE 做 QR 分解為主的方法[10]-[12]。底下則介紹在本論文中所採用 ZF、MMSE 及 V-BLAST[4]演算法。

3.1.1 Zero Forcing 估測(ZF)

在收到接收訊號的條件下:$Y(k) = H(k)S(k) + N(k)$ ，我們可將等式兩邊各乘上

$G = H^{-1}(k)$，可得如下式：

$$GY(k) = S(k) + GN(k) \tag{3-1}$$

如(3-1)，我們再定義一個錯誤向量$e(k) = GY(k) - S(k)$，由此可得$e(k) = GN(k)$；也就是說，傳送的訊號與接收訊號經過 ZF 壓抑矩陣G後的誤差，將只剩下等效的雜訊項$GN(k)$，而沒有了來自其它天線上的干擾，其 CSI 項也會相對於它種偵測演算法較為簡單。當然，ZF 的壓抑矩陣$G = H^{-1}(k)$，必須假設在傳送天線數N_t等於接收天線數N_r的條件下才會成立。如果資料流大於接收天線數時，則 ZF 的壓抑矩陣G如下式所示：

$$G = H^H(k)[H(k)H^H(k)]^{-1} = [H^H(k)H(k)]^{-1}H^H(k) \tag{3-2}$$

(3-2)式的推導，將於下節做較為詳細的探討。本論文所實現的硬體中含一2×2的 ZF 偵測器，故可用$G = H^{-1}(k)$為 ZF 的壓抑矩陣。另外，$H^{-1}(k)$的行列式有可能為零，所以在硬體的設計上，須針對此一情況加以處理。

3.1.2 最小均方差估測(MMSE)

在收到接收訊號的條件下:$Y(k) = H(k)S(k) + N(k)$，我們定義一個錯誤向量$e(k)$，此向量代表傳送的訊號與接收訊號乘上 MMSE 壓抑矩陣後的誤差，也就是$e(k) = S(k) - G^H Y(k)$，其中G是 MMSE 壓抑矩陣。另外，定義一個成本函數(Cost Function) J，如下式：

$$J = E\{e^H(k)e(k)\} = tr[E\{e(k)e(k)^H\}] \tag{3-3}$$

為了使成本最小，我們對J微分，使其等於零，並找出此時的 MMSE 壓抑矩陣的值，於是可以推導出 Wiener-Hopf equation：

$$G^H R_{YY} = R_{SY}$$

(3-4)

其中

$$R_{YY} = E\{Y(k)Y^H(k)\} \quad R_{SY} = E\{S(k)Y^H(k)\}$$

(3-5)

前者為接收訊號向量的協方差矩陣(Covariance Matrix)，後者為傳輸和接收向量的交互相關矩陣(Cross-Correlation Matrix)。然後我們假設:

$$E[SS^H] = \frac{1}{\alpha} I_{Nt},\ E[NN^H] = I_{Nr},\ E[SN^H] = 0$$

(3-6)

這裡$\alpha = \frac{N_t}{\sigma^2}$。換句話說，訊號$S$及外加雜訊$N$在空間上是白色(white)及沒有相關的隨機向量，而變異數分別為$\frac{1}{\alpha} I_{Nt}$及I_{Nr}(我們可以將$\frac{1}{\alpha}$想像成在每根天線上SNR)，便可以推導出 MMSE 壓抑矩陣G如下：

$$G = [HH^H + \alpha I_{N_r \times N_r}]^{-1} H$$

(3-7)

經由一些簡單的運算，上式可等效於式(3-8)

$$G = H[H^H H + \alpha I_{N_t \times N_t}]^{-1}$$

(3-8)

由於$N_{ss} \leqq N_r$，在運算反矩陣的時候，式(3-8)的運算量會小於或等於式(3-7)，因此在實際使用的效率上，我們比較偏好使用式(3-8)。如果我們令$\alpha = 0$，可得如下式所示：

$$G = [HH^H]^{-1} H = H[H^H H]^{-1} \text{ 或 } G^H = H^H [HH^H]^{-1} = [H^H H]^{-1} H^H$$

(3-9)

此時，MMSE 偵測器之G^H，即為 ZF 的壓抑矩陣G，如式(3-2)所示。

3.1.3 V-BLAST

在 V-BLAST 的過程中，偵測的順序對於系統整體的效能影響很大，若第一個解出來的訊號是錯的，在接下來的遞迴過程中，錯誤會延續到第二個訊號上，也就是所謂的錯誤延展(Error Propagation)。因此，有效的排序將可以對系統效能有很大的提升。因此，我們底下討論如何對訊號排序。首先，我們知道 MMSE 壓抑矩陣可表示成 $G = HQ$，且 $Q = [H^H H + \alpha I_{N_t \times N_t}]^{-1}$。當我們計算誤差向量 $e(k)$ 的協方差矩陣時，可得下式：

$$\begin{aligned} R_{ee} &= E\{e(k)e^H(k)\} \\ &= \sigma_n^2 Q \end{aligned} \tag{3-10}$$

觀察(3-10)，擁有 SNR 最高的訊號就是有最小的誤差變異數，所以要解的第一個訊號可以從(3-11)獲得，其中 q_{mm} 代表矩陣 Q 之第 m 個對角線的值。

$$p_1 = \arg\min_m q_{mm} \tag{3-11}$$

而 V-BLAST 的流程主要分為三大步驟，依序如下：

I. 使用(3-8)MMSE 壓抑矩陣，我們可以使用(3-12)計算 $Z(k)$:

$$Z(k) = G^H Y(k) \tag{3-12}$$

II. 利用(3-11)找出 $Z(k)$ 裡面 SNR 最高的子資料符元，並將此符元偵測出來。假設找出在 Z(k) 裡面的第 p_1 個子資料符元擁有最高的 SNR，於是可對第 p_1 個符元作判斷(slice)：$\hat{S}_{p_1}(k) = Q[Z_{p_1}(k)]$ 其中，$Q[.]$ 是根據訊號的星狀圖(Constellation)對符元作判斷。

III. 假設 $\hat{S}_{p_1}(k) = S_{p_1}(k)$，我們就可以把 $S_{p_1}(k)$ 從接收訊號向量 $Y(k)$ 中移除，並獲得一組新的接收向量 $Y_2(k)$，如下所示：

$$
\begin{aligned}
Y_2(k) &= Y(k) - \hat{S}_{p_1}(k) h_{:p_1} \\
&= \sum_{m \neq p_1} h_{:m} S_m + N(k) \\
&= H_{N_t-1}(k) S_{N_t-1}(k) + N(k)
\end{aligned}
\tag{3-13}
$$

其中 $h_{:p_1}$ 是 $H(k)$ 中的第 p_1 列，而 $H_{N_t-1}(k)$ 是指把 $H(k)$ 的第 p_1 個行移除而獲得的一個 $N_r \times (N_t - 1)$ 之矩陣。而 $S_{N_t-1}(k)$ 是把 $S(k)$ 的第 p_1 個資料移除，獲得一個長度為 $N_t - 1$ 的向量。

由步驟 I 到 III，每得到一個新的 H 就要重新計算(3-11)以獲得新的排序。經由算出來的 $p_2, ..., p_{N_t}$ 順序，並搭配更新後的接收向量 $Y_2(k), ..., Y_{N_t}(k)$，如此遞迴的運算直到所有 $S(k)$ 的資料都判斷出來後才停止。

3.1.4 軟性反對映(Soft de-mapping)

符元對映(symbol mapping)在 BICM 的系統中扮演著很重要的角色，在非遞迴解碼裡[15]，証明了葛雷碼(Gray-code)對映的最佳性。在使用葛雷編碼下，如果使用最大可能性解碼時，則反對映(de-mapping)則有明顯簡化的方法[16]，我們將在下面說明。

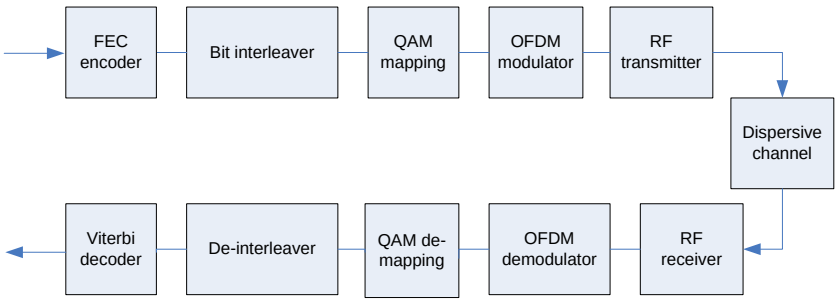


圖 3-1 單天線傳送接收器

所謂的軟性反對映(soft de-mapping)即在決定在每個接收到的 symbol 裡的位

元是 0 或是 1 的機率，metric 的值愈大便代表著位元是 1 的機率愈大，反之值愈小的話便代表著位元是 0 的機率大。在說明應用在 MIMO 的情況之前，我們先說明在單天線的情況。

單天線傳送接收器如圖 3-1 所示，在頻域上訊號可以表示為:

$$Z(k) = H(k)S(k) + N(k)$$

$$Y(k) = S(k) + \frac{N(k)}{H(k)} = S(k) + N'(k)$$

(3-14)

$H(k)$ 是通道頻率響應(Channel Frequency Response, CFR)在第 k 個子載波的複數係數，S 是傳送訊號，N 則是指外加雜訊，$\sigma_{N'}^2 = \frac{\sigma_N^2}{\|H[k]\|^2}$。傳送訊號 S 是一個從有限星狀圖(constellation) $\Upsilon = \{S_1, S_2, ...S_{|\Upsilon|}\}$ 取出的 QAM symbol， 由資料位元 $b = [b_0, b_1, ...b_M]$ 所對映而來。位元數 M 則是根據使用的 QAM 大小所決定。在接收到每個 $Z(n)$ 都有 $4M$ 個 metric 需要計算，in-phase 和 quadrature 位元 $b_{I,k}$、$b_{Q,k}$ 各需計算 0 和 1 的可能性，對 $b_{I,k}$ ($b_{Q,k}$ 是同樣的)而言，我們將 QAM 集合 Υ 分成二部分，$S_{I,k}^{(0)}$ 代表著在位置 (I,k) 的位元是 0 的點，也就是在 in-phase(I)位元群中第 k 個位元。相對的，是 1 的點便用 $S_{I,k}^{(1)}$ 表示。如此這二個 metrics 即可由下式表示:

$$m_c(b_{I,k}) = \max_{\alpha \in S_{I,k}^{(c)}} \log p(Z(k) \mid S(k) = \alpha), c = 0,1$$

(3-15)

$Z(k)$ 的機率是個高斯函數

$$p(Z(k) \mid S(k) = \alpha) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{1}{2}\frac{|Z(k) - H(k)\alpha|^2}{\sigma^2}\right\}$$

(3-16)

且 $Z(n)=H(n)Y(n)$ ，如此(3-15)即可改寫成

$$m_c(b_{I,k})=\left|H(k)\right|^2 \min_{\alpha\in S_{I,k}^{(c)}}\left|Y(k)-\alpha\right|^2 , c=0,1$$

(3-17)

log likelihood ratio (LLR):

$$L(b_{I,k})\equiv \log\frac{p(b_{I,k}=1\,|\,Z(k))}{p(b_{I,k}=0\,|\,Z(k))} = \log\frac{\sum_{\alpha\in S_{I,k}^{(1)}} p(S(k)=\alpha\,|\,Z(k))}{\sum_{\alpha\in S_{I,k}^{(0)}} p(S(k)=\alpha\,|\,Z(k))}$$

(3-18)

提供了第 (I,k) 的位元到底是 0 或是 1 的可靠度計算方法，由 $L(b_{I,k})$ 的正負號便可決定第 m 個位元是 0 或是 1，而值的大小則表示著可靠度。

當雜訊並非很大時，我們可以將 log-sum 簡化成: $\log\sum_j Z_j \approx \max_j \log Z_j$ ，如此(3-18)可寫成:

$$L(b_{I,k})= \log\frac{\max_{\alpha\in S_{I,k}^{(1)}} p(S(k)=\alpha\,|\,Z(k))}{\max_{\alpha\in S_{I,k}^{(0)}} p(S(k)=\alpha\,|\,Z(k))}$$

(3-19)

將(3-17)代入(3-19)可得:

$$L(b_{I,k})= \frac{\left|H(k)\right|^2}{2\sigma^2}\{\min_{\alpha\in S_{I,k}^{(0)}}\left|Y(k)-\alpha\right|^2-\min_{\alpha\in S_{I,k}^{(1)}}\left|Y(k)-\alpha\right|^2\}$$

(3-20)

在圖 3-2 我們以 16QAM 為例子，為了計算 $L(b_{I,k})$ 而在 $(S_{I,k}^{(1)},S_{I,k}^{(0)})$ 這二個集合裡找離接收訊號最近的點時，我們可以發現在這二個集合裡最近的點都是會在同一條垂直的線上，同樣地為了計算 $L(b_{Q,k})$ 在 $(S_{Q,k}^{(1)},S_{Q,k}^{(0)})$ 裡找最近的點時，這二點便會在同一條水平線上。因此(3-20)可以簡化成:

$$
\begin{aligned}
L(b_{I,k}) &= \frac{|H(k)|^2}{2\sigma^2}\{\min_{\alpha\in S'^{(0)}_{I,k}}|Y(k)-\alpha|^2 - \min_{\alpha\in S'^{(1)}_{I,k}}|Y(k)-\alpha|^2\} \\
&= \frac{|H(k)|^2}{2\sigma^2}\times 4\times\frac{1}{4}\times\{\min_{\alpha\in S'^{(0)}_{I,k}}|Y(k)-\alpha|^2 - \min_{\alpha\in S'^{(1)}_{I,k}}|Y(k)-\alpha|^2\} \\
&= \frac{2|H(k)|^2}{\sigma^2}\times\frac{1}{4}\{\min_{\alpha\in S'^{(0)}_{I,k}}|Y(k)-\alpha|^2 - \min_{\alpha\in S'^{(1)}_{I,k}}|Y(k)-\alpha|^2\} \\
&\equiv CSI\times D_{I,k}
\end{aligned}
\tag{3-21}
$$

式中子集合 $S'^{(c)}_{I,k}$ 定義為: $S'^{(c)}_{I,k}\equiv\Re\{S^{(c)}_{I,k}\}$ ， $c=0,1$。因此在計算 $L(b_{Q,k})$ 便是先由通道和雜訊算出在(3-21)中所需的 CSI，而 $D_{I,k}$ 可利用圖 3-2 而計算得(3-22):

$$
D_{I,1}=\begin{cases}
\frac{1}{4}\left[(Y_I(k)+1)^2-(Y_I(k)-1)^2\right]=Y_I(k), & |Y_I(k)|\le 2\\
\frac{1}{4}\left[(Y_I(k)+1)^2-(Y_I(k)-3)^2\right]=2(Y_I(k)-1), & Y_I(k)>2\\
\frac{1}{4}\left[(Y_I(k)+3)^2-(Y_I(k)-1)^2\right]=2(Y_I(k)+1), & Y_I(k)<-2
\end{cases}
$$

$$
D_{I,2}=\begin{cases}
\frac{1}{4}\left[(Y_I(k)-3)^2-(Y_I(k)-1)^2\right]=-Y_I(k)+2, & Y_I(k)>0\\
\frac{1}{4}\left[(Y_I(k)+3)^2-(Y_I(k)+1)^2\right]=Y_I(k)+2, & Y_I(k)<0
\end{cases}
\tag{3-22}
$$

由(3-22)可整理成(3-23):

$$
\begin{aligned}
D_{I,1} &= \begin{cases}
Y_I(k), & |Y_I(k)|\le 2\\
2(Y_I(k)-1), & Y_I(k)>2\\
2(Y_I(k)+1), & Y_I(k)<-2
\end{cases}\\
D_{I,2} &= -|Y_I(k)|+2
\end{aligned}
\tag{3-23}
$$

我們可以很容易地推出 $D_{Q,k}$ 的式子，只要將(3-23)中的 $Y_I(k)$ 改成 $Y_Q(k)$ 即可。

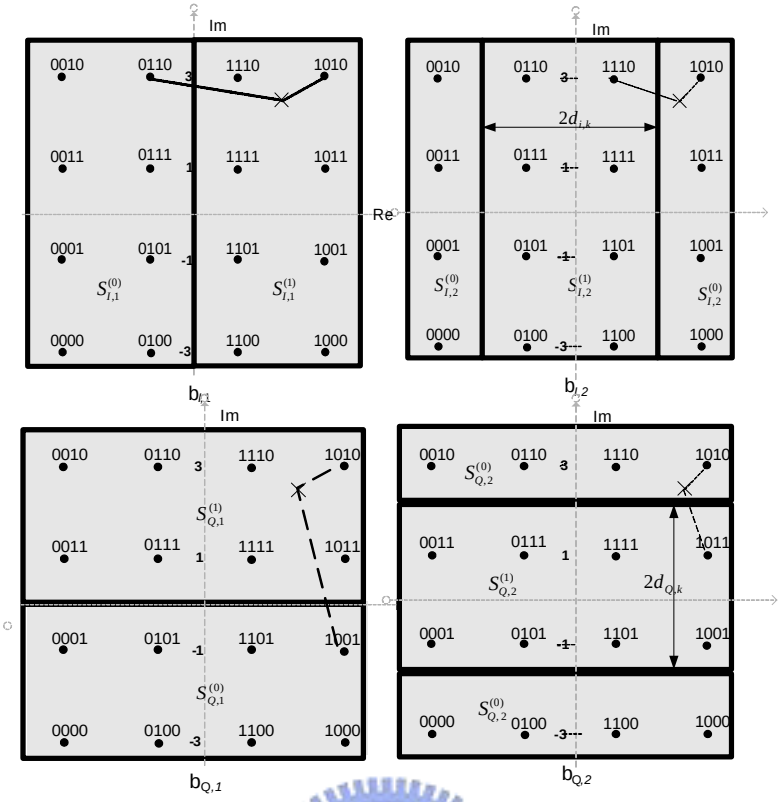


圖 3-2 16QAM 星狀圖的分割示意圖

(3-23)在計算上還是令人困擾，但如果犧牲一點效能的話，則可以再進一步化簡為:

$$D_{I,1} \cong Y_I(k)$$
$$D_{I,2} = -\left|Y_I(k)\right| + 2 \qquad \text{(3-24)}$$

(3-23)及(3-24)所算出之 $D_{I,1}$ 及 $D_{I,2}$ 的比較如圖 3-3 所示:

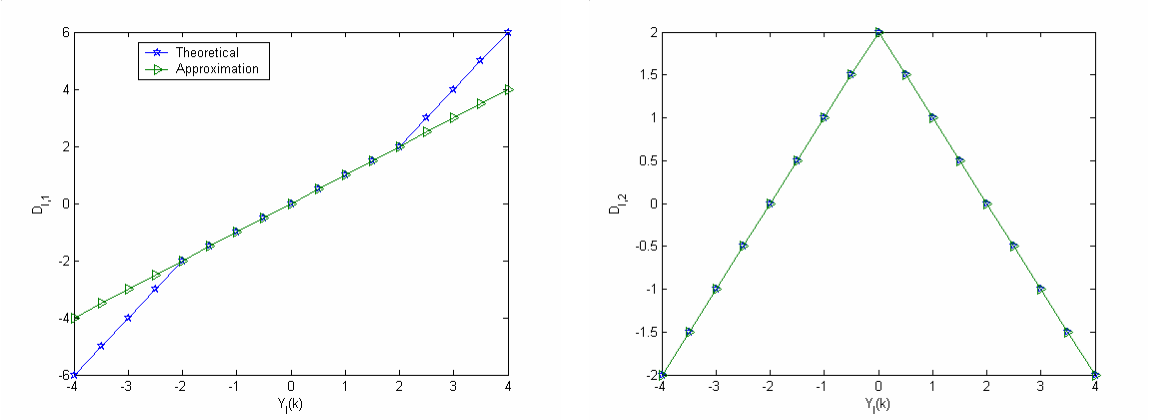


圖 3-3 In-phase 位元中，16QAM 之簡化對精確 LLR 計算方法比較圖

相同地，在 64QAM 的情況下可以導出(3-25)及其簡化式(3-26)

$$D_{I,1}=\begin{cases} Y_I(k), & |Y_I(k)|\le 2 \\ 2(Y_I(k)-1), & 2<Y_I(k)\le 4 \\ 3(Y_I(k)-2), & 4<Y_I(k)\le 6 \\ 4(Y_I(k)-3) & Y_I(k)>6 \\ 2(Y_I(k)+1) & -4\le Y_I(k)<-2 \\ 3(Y_I(k)+2) & -6\le Y_I(k)<-4 \\ 4(Y_I(k)+3) & Y_I(k)<-6 \end{cases}$$

$$D_{I,2}=\begin{cases} 2(-|Y_I(k)|+3), & |Y_I(k)|\le 2 \\ 4-|Y_I(k)| & 2<|Y_I(k)|\le 6 \\ 2(-|Y_I(k)|+5) & |Y_I(k)|>6 \end{cases}$$

$$D_{I,3}=\begin{cases} |Y_I(k)|-2, & |Y_I(k)|\le 4 \\ -|Y_I(k)|+6, & |Y_I(k)|>4 \end{cases}$$

(3-25)

$$D_{I,1}\cong Y_I(k)$$
$$D_{I,2}\cong -|Y_I(k)|+4$$
$$D_{I,3}\cong -|-|Y_I(k)|+4|+2$$

(3-26)

(3-25)及(3-26)所算出之 $D_{I,1}$、$D_{I,2}$ 及 $D_{I,3}$ 的比較如圖 3-4 所示:

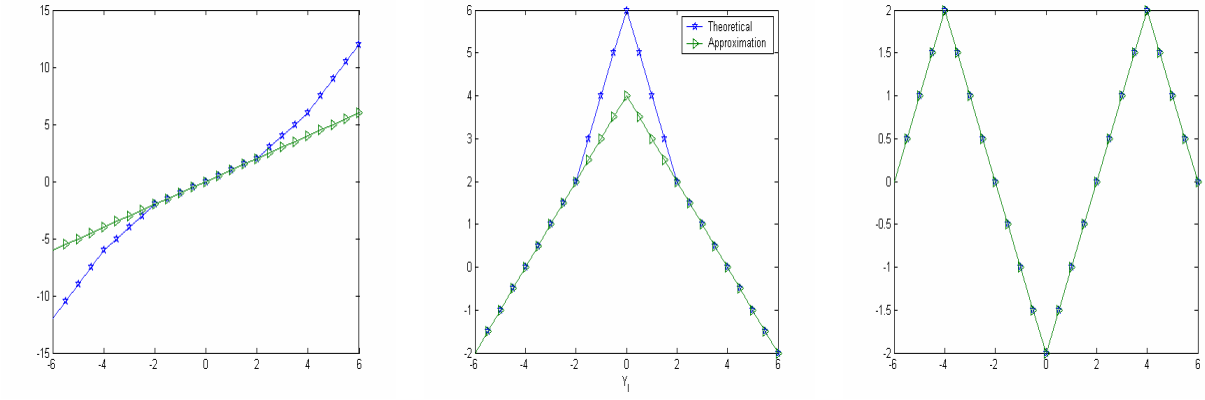


圖 3-4 In-phase 位元中，64QAM 之簡化對精確 LLR 計算方法比較圖

(3-26)可以推展到使用類似葛雷編碼的任意平方 QAM 的星狀圖上，假設 $d_{I,k}$ 及 $d_{Q,k}$ 為當 $k>1$ 時，二個集合之間距離的一半當如圖 3-2 所示，則從(3-23)到(3-26)

便可一般化為:

$$D_{I,k}=\begin{cases}Y_I(k), & k=1\\ -\left|D_{I,k-1}\right|+d_{i,k}, & k>1\end{cases}$$

$$L(b_{I,k})=CSI\times D_{I,k}$$

and

$$D_{Q,k}=\begin{cases}Y_Q(k), & k=1\\ -\left|D_{Q,k-1}\right|+d_{Q,k}, & k>1\end{cases}$$

$$L(b_{Q,k})=CSI\times D_{Q,k}$$

(3-27)

3.1.5 軟性輸入軟性輸出之 ZF 接收機

這個接收器如圖 3-5 所示，為了方便說明，這裡用 2×2 的模型，也很容易推廣到多根傳送接收天線模型。相對於將接收訊號視為訊號向量，我們直接使用線性的接收器將此向量分成二條獨立的資料流，並分開地計算其軟性輸出(Soft output)。

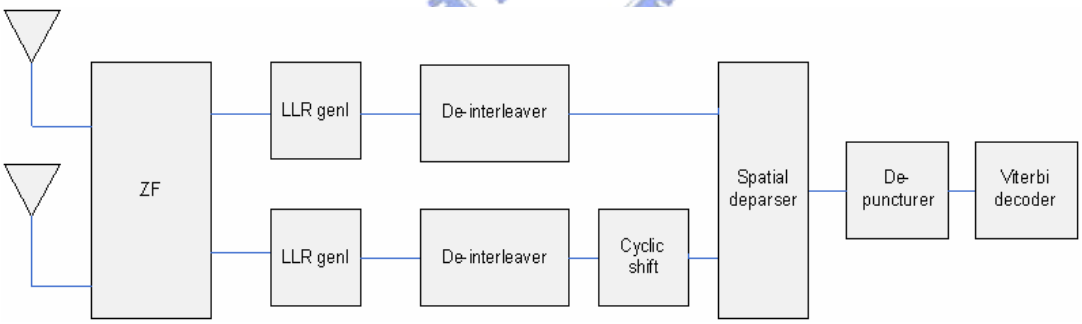


圖 3-5 2×2 之軟性輸入軟性輸出 ZF 接收器

在 2×2 的情況下(2-8)中的接收訊號可以表示為:

$$Y_k=\begin{bmatrix}h_{1,k} & h_{2,k}\end{bmatrix}\begin{bmatrix}s_{1,k}\\ s_{2,k}\end{bmatrix}+N_k$$

(3-28)

$h_{i,k}$ 是 $s_{i,k}$ 的通道影響，也就是 H 的第 i 列。而 ZF 的等化器則可以表示為

$$W = (H_k)^{-1} = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix}$$

(3-29)

則此等化過後的訊號即變成 $Z_k = WY_k$，假設 w_i^T 是 W 的第 i 行，則對第 i 根天線而言，等化後的訊號即可表示為:

$$Z_{i,k} = s_{i,k} + \underbrace{w_i^T N_k}_{N_{eff}}$$

(3-30)

使用(3-30)中 Z_k、及 N_{eff} 則可得等效之雜訊變異數為:

$$\sigma_{N_{eff}}^2 = \left\| w_i \right\|^2 N_0$$

(3-31)

如此(3-30)便和在 3.1.4 中所介紹軟性反對映有相似的格式，只是少了 $H(k)$。注意本方法的計算複雜度不論是在偵測器的實現上或是 CSI 的計算上都較 MMSE 偵測器為低，且在這裡我們不需考慮其它根天線的殘餘干擾訊號所造成的影響。

3.1.6 軟性輸入軟性輸出之MMSE接收機

這個接收器如圖 3-6 所示，為了方便說明，這裡用 2×2 的模型，也很容易推廣到多根傳送接收天線模型。相對於將接收訊號視為訊號向量，我們直接使用線性的接收器將此向量分成二條獨立的資料流，並分開地計算其軟性輸出(Soft output)。

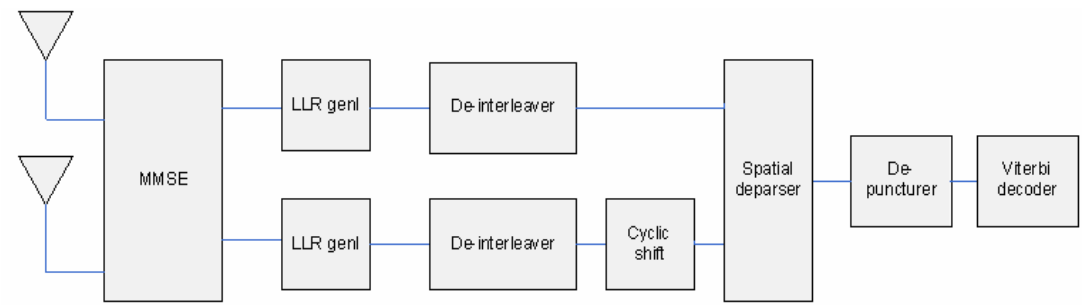


圖 3-6 2×2 之軟性輸入軟性輸出 MMSE 接收器

在 2×2 的情況下(2-8)中的接收訊號可以表示為:

$$Y_k = \begin{bmatrix} h_{1,k} & h_{2,k} \end{bmatrix} \begin{bmatrix} s_{1,k} \\ s_{2,k} \end{bmatrix} + N_k$$

(3-32)

$h_{i,k}$ 是 $s_{i,k}$ 的通道影響，也就是 H 的第 i 列。而 MMSE 的等化器則可以表示為

$$W = (H_k^H H_k + \frac{2}{\rho} I)^{-1} H_k^H = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix}$$

(3-33)

ρ 是 SNR$\frac{E_S}{N_0}$,則此等化過後的訊號即變成 $Z_k = WY_k$ ，假設 w_i^T 是 W 的第 i 行，則對第 i 根天線而言，等化後的訊號即可表示為:

$$Z_{i,k} = \underbrace{w_i^T h_{i,k}}_{H_{eff}} s_{i,k} + \underbrace{w_i^T h_{j \neq i,k} s_{j,k} + w_i^T N_k}_{N_{eff}}$$

(3-34)

使用(3-34)中 Z_k 、 H_{eff} 及 N_{eff} 則可得等效之雜訊變異數為:

$$\sigma_{N_{eff}}^2 = w_i^H h_{j \neq i,k} h_{j \neq i,k}^H w_i + \left\| w_i \right\|^2 N_0$$

(3-35)

如此(3-34)便和在 3.1.4 中所介紹軟性反對映有相同的格式。注意本方法的計算複雜度跟天線數成線性增加，且在這裡我們將其它根天線的殘餘干擾訊號當成雜訊處理。

3.2 Viterbi 解碼(Decoding)

Viterbi 演算法是在 1967 年由 Viterbi 所提出來的[23]來對迴旋碼進行解碼的，近年來，Viterbi 演算法則有非常廣泛的應用，例如數位通訊系統、語音辨識等。此演算法為最大近似解碼(Maximum Likelihood Decoding ; MLD)，簡單地說便是依據接收訊號來尋找在格狀圖上所有可能的路徑，最後選擇一條最短路徑來當作最佳解碼路徑。演算法中包括了計算各個路徑與所收到訊號之間的相似度，並且去除掉一些不可能成為最佳選擇的路徑。當幾個路徑進入同一狀態時，在這幾個路徑之間的其中一個會因有「適當」的計量值(metric)而被選擇，而這被選擇的路徑也就是存活路徑(survivor path)。所謂的「適當」則要根據計量值的定義。在格狀圖中的所有狀態都要進行這樣的存活路徑選擇。在接下來的時間也是根據這樣的法則來去除不可能的路徑，如此，在收集到一定長度的訊號後便可以將解碼位元輸出了。底下我們用一個例子來說明這演算法。

為了描述 Viterbi 演算法是如何動作之前，我必需先定義出格狀圖，這是根據編碼器而產生的。如圖 3-7 所示，我們用一個簡單的$(n,k,m)=(2,1,2)$的迴旋碼編碼器來舉例說明，它由 2 個移位暫存器，2 個模 2(Modulo-2)加法器所組成，而模 2(Modulo-2)的加法器可以由 XOR 閘來實現。迴旋碼編碼器基本上是一個有限狀態機(Finite state Machines)，我們可以用狀態圖來描述其輸入輸出的關係，由之編碼器可以建立如圖 3-8 所示的狀態圖。其中每一狀態點為編碼器內暫存器之值，輸入與輸出位元則表為$v^{(1)}v^{(2)}/u$。通常我們會將這狀態圖轉成格狀圖來說明，因為對於時間與狀態之間的轉換有較好的表達能力。如圖 3-9 所示，實線代表輸入 0，虛線代表輸入 1。對圖 3-9 來說，因為編碼器有二個移位暫位器，所以共會有 4 個狀態，分別為 $S_0(0,0)$，$S_1(1,0)$，$S_2(0,1)$，$S_3(1,1)$，每一個狀態有 2 條路徑指向下一個狀態，一條為當輸入位元 0 時，另一條則是輸入位元為 1 時。

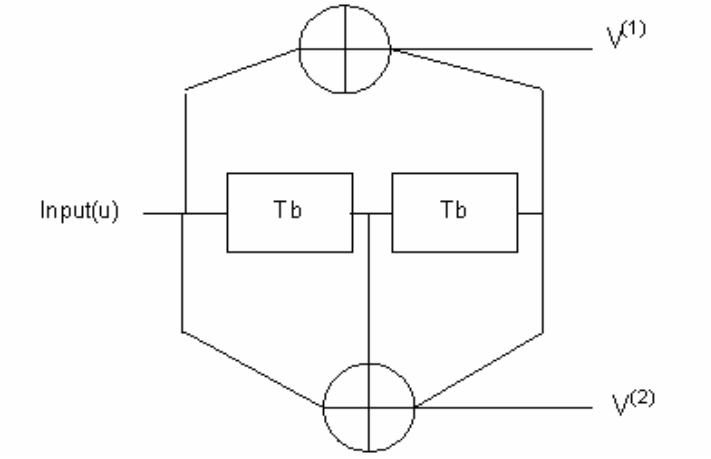
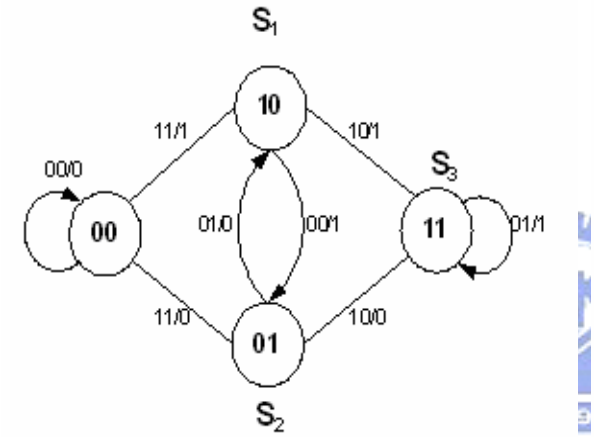


圖 3-7 (2,1,2)之迴旋碼編碼器



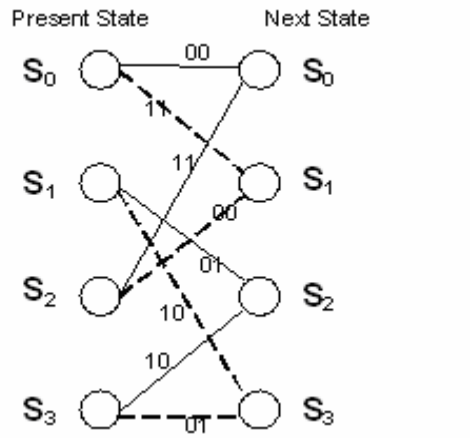


圖 3-8 (2,1,2)迴旋碼編碼器之狀態圖　　圖 3-9 (2,1,2) 迴旋碼編碼器之格狀圖

為了說明方便，我們使用 Hard Decision 的 Viterbi 來說明 Viterbi 是如何運作的，然而在吾人的實作中是使用 Soft Decision 的 Viterbi，而二者的差別只需將底下所使用的漢明距離改為歐幾里德(Euclidean)距離即可。在說明 Viterbi 演算法之前，我們先定義一些名詞:

- 碼句(code word) $Cw(i, j)$:由狀態 S_i 轉移到狀態 S_j 所對應的編碼器輸出。
- 分支計量值(Branch metric) $Bm(i, j, t)$:在時間 t，接收到的訊號($r(t)$)和碼句之間的相似度(Likelihood)，可用漢明距離表示。
- 路徑計量值(Path metric) $Pm(i, j, t)$:在時間 t，接收到的碼和由狀態 S_i 轉

移到狀態 S_j 之間的相似度。

- 存活計量值(Survivor metric) $Sm(j,t)$:在時間 t，進入狀態 S_j 的最小路徑計量值。

$Cw(i,j)$ 、 $Bm(i,j,t)$ 、 $Pm(i,j,t)$ 、 $Sm(j,t)$ 之間可表示如下:

$$Bm(i,j,t) = Cw(i,j) ** r(t)$$

$$Pm(i,j,t) = Bm(i,j,t) + Sm(j,t-1) \quad (3\text{-}36)$$

where ** is hamming distance calculation

假設有一全零資料序列 u =(0000000)，經過編碼後，我們可以得到編碼後的輸出序列為 v =(00,00,00,00,00,00,00)，而 v 序列在經過通道的輸輸過程中受到雜訊的干擾，使得第 1 及第 3 位元由 0 變成 1，而解碼器所收到的序列為 (10,00,00,00,10,00,00)，經由以下的 Viterbi 演算法，可求得解碼之結果為 $\hat{u}$ =(0000000)= u 。

I. 如圖 3-10 所示，在 $t = 0$ 時，從狀態 S_0 開始，其存活計量值為 0。若輸入 0，則路徑向上，在 $t = 1$ 時，狀態 S_0 輸出 00。若輸入 1，則路徑向下，在 $t = 1$ 時，狀態點 S_1 ，輸出為 11。若輸出位元與 r 之所對應的位元相同時，則我們給此位元之計量值為 0，反之為 1。因此，我們可以算出分支計量值 $Bm(0,0,1) = 1$ 及 $Bm(0,1,1) = 1$。因為在 t = 1 時，每一狀態只有一條路徑進入，所以存活計量值 $Sm(0,1) = 1$ 、 $Sm(1,1) = 1$ 。

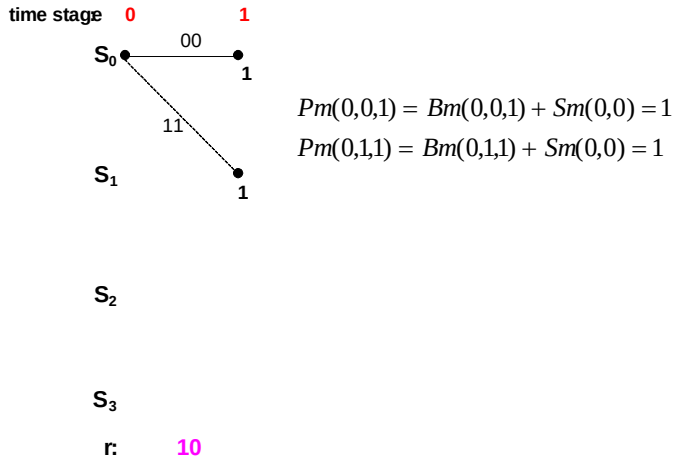
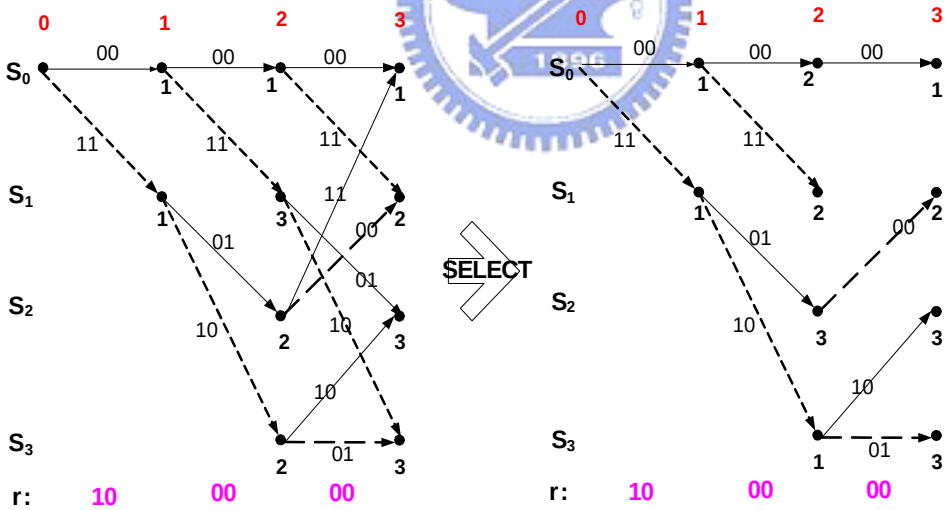


圖 3-10 Viterbi 演算法步驟 I

II. 重覆 I。如圖 3-11 所示，在 $t = 3$ 時，每一狀態都有二條路徑進入，我們選擇路徑較小的為存活計量值，此一路徑稱為存活路徑，同時把另一較大的路徑刪除。若兩路徑計量值相同時，則任取一條路徑為存活路徑。因此，S_0、S_1、S_2、S_3 的存活計量值 $Sm(0,3)$、$Sm(1,3)$、$Sm(2,3)$、$Sm(3,3)$ 分別為 1、2、3、3。



$Sm(0,3) = Min\{Pm(0,0,3), Pm(2,0,3)\} = 1$
$Sm(1,3) = Min\{Pm(0,1,3), Pm(2,1,3)\} = 2$
$Sm(2,3) = Min\{Pm(1,2,3), Pm(3,2,3)\} = 3$
$Sm(3,3) = Min\{Pm(1,3,3), Pm(3,3,3)\} = 3$

圖 3-11 Viterbi 演算法步驟 II

III. 重覆 II，經過時間 $t = 4 \sim 7$。如圖 3-12 所示，在最後 $t = 7$ 時，我們挑選有

存活計量值的狀態所走的路徑為存活路徑，由存活路徑，也就是在圖中線條較粗者的，得到解碼之結果為 $\hat{u}$ =(0000000)= u ，在此路徑中，實線為 0 而虛線為 1。

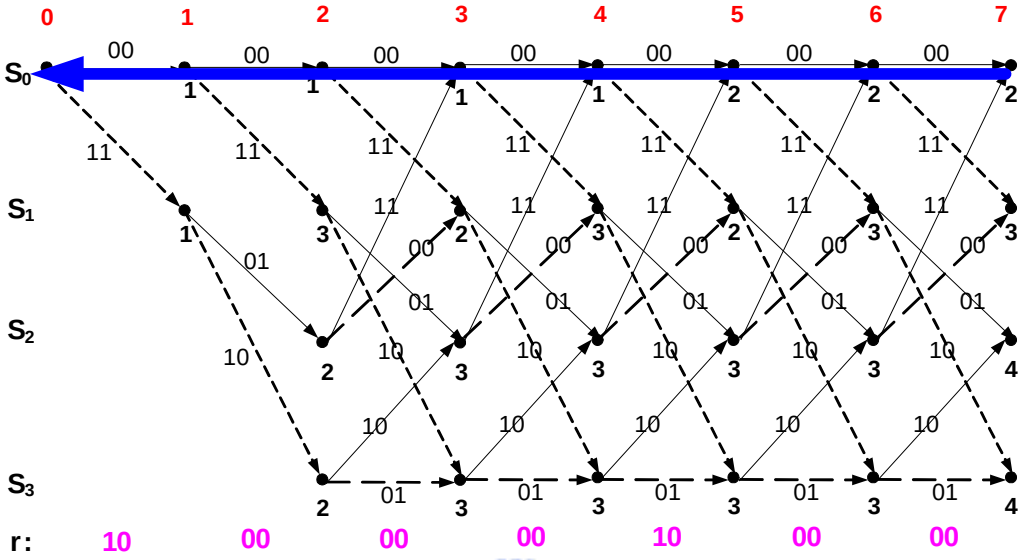


圖 3-12 Viterbi 演算法步驟 III

最後 Vitebi 演算法總結如下:

I. 從單元時間 t=1 開始，計算進入每一狀態之單一路徑的路徑計量值(path metric) $Pm(i, j, t)$ ，並記錄每一狀態之存活路徑和存活計量值(survivor metric) $Sm(i, j, t)$ 。

II. t 增加 1，對於每一個狀態中，比較進入此狀態的路徑，且將含有最小路徑計量值的路徑和其值記錄起來，而這個記錄起來的計量值便是存活計量值，而其它路徑便給予刪除。

III. 如果 $t<L+m$，則重覆步驟 2，否則停止。其中 L 為資料的長度，m 為編碼器中移位暫存器的個數。最後，比較在所有狀態的存活路徑計量值，有最小計量值的存活路徑便可以當作解碼路徑。

在 Viterbi 演算法中，我們必須將所有狀態及其對應的存活路徑儲存起來，

若路徑很長(也就是訊號序列長)，則需要一個很大的記憶體。因此，我們必須在路徑到達一定長度時加以截斷。當我們要截斷最前面的路徑時，必須要將其資料取出(解碼)。一般而言，若路徑之截斷長度(也就是被截斷存活路徑的長度)為 5.8*m* 時，不管任何一種路徑取出資料，所增加的解碼錯誤率是可以忽略的，也就是說，其最後面的一個分支都會重合成一條，如圖 3-13 所示。

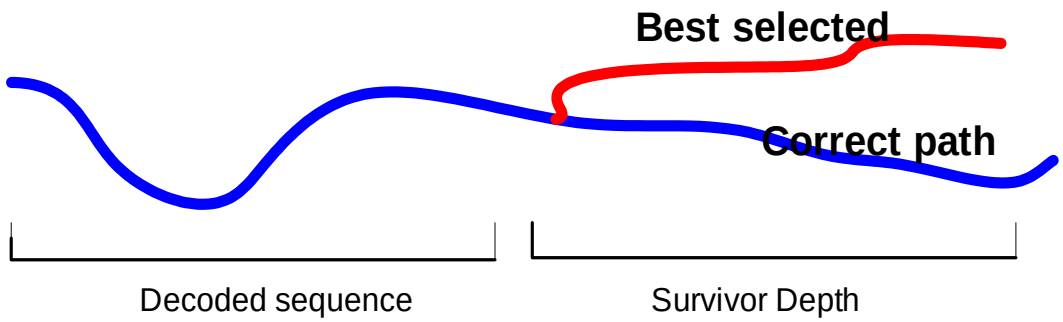


圖 3-13 Viterbi 演算法之截斷長度示意圖

3.3 模擬結果

在本節中，我們主要模擬 11n 提案中的 2×2 的基本模式，詳細的環境已在第二章中說明，在這樣的環境下我們模擬第二章及第三章所提及的各種演算法。模擬分兩部分，在第一部分中我們使用一個簡單的通道，假設在每根傳送天線 i 跟每根接收天線 j 之間的通道可以模擬為一個 FIR 濾波器，並且可以用序列 $[...H_{ij,l}...], l=0,\cdots,L$ 表示，L 代表最大 tap 數，且這些 taps 都是由獨立複數高斯隨機變數所產生。而每個 tap 可用表示如下:

$$H_{ij,l} = N(0,\sigma_l^2) + j \cdot N(0,\sigma_l^2)$$
$$\sigma_l^2 = \sigma_0^2 \exp(-lT_s / T_{RMS})$$
$$\sigma_0^2 = 1 - \exp(-T_s / T_{RMS}) \quad (3\text{-}37)$$

我們令 $T_s = 50ns$ 且 $T_{RMS} = 150ns$，且在模擬用的 SNR 是定義在接收到每個 OFDM 符碼後，計算出經過通道後的訊號能量加入在這 SNR 下該有的雜訊能量。

模擬的演算法主要分成 2 部分，第一部分主要在模擬比較 ZF、MMSE 及 V-BLAST 演算法加上 Soft-decision Viterbi 解碼在上述的環境中的效能表現，而 CSI 便是影響 ZF、MMSE 及 V-BLAST 的效能最主要的因素，這部分的模擬結果包括了圖 3-14 至圖 3-19。圖 3-14 比較了 ZF 及 MMSE 在不接 Viterbi 解碼器下之效能，而在圖中 only 表示不接 Viterbi 解碼器而單獨使用 ZF 或 MMSE。圖 3-15 比較了 2 種 ZF 作法，ZF 的 CSI 計算方式是使用 3.1.5 所介紹，而在圖中 NO CSI ZF 則是令 CSI 為 1 的情況。圖 3-16 比較了 2 種 MMSE 作法，MMSE 的 CSI 計算方式是使用 3.1.6 所介紹，而在圖中 NO CSI MMSE 則是令 CSI 為 1 的情況。而圖 3-17 則綜合比較 ZF、MMSE 及 V-BLAST 加上 Soft-decision Viterbi 解碼在上述環境下的表現，其中 CSI1 是在 V-BLAST 演算法中，假設前一級的決策是正確的情況下，使用一級一級降低的通道矩陣所計算出來的 CSI，而 CSI2 則是

直接使用第一級 MMSE 的計算出來的 CSI。圖 3-18 是使用 ZF 演算法，比較在 Single-Input Single-Output (SISO)、2×2、3×3 和 4×4 的天線組合下之效能比較。圖 3-19 是使用 ZF 演算法，在不同通道雜訊(channel noise)下之效能比較，其中 Perfect 表示 Perfect channel，而 CN1%、CN0.5%和 CN0.1%分別代表加上 zero mean 而功率分別為 1%、0.5%和 0.1%的高斯通道雜訊。從圖 3-14 模擬結果發現，不接 Viterbi 解碼器而單獨使用 ZF 或 MMSE 下，MMSE 較 ZF 有較好的效能，而 SNR 越高時，ZF 與 MMSE 的效能也就越來越近。從圖 3-15 至圖 3-17 模擬結果發現，ZF 及 MMSE 皆受 CSI 影響甚鉅，有使用 CSI 及沒有使用 CSI 在 ZF 及 MMSE 約造成 3dB 的影響，另外，由於整個系統皆有連接 Viterbi 解碼器，因為 Viterbi 解碼器擁有錯誤更正(error correction)的能力，導致 ZF 及 MMSE 演算法在效能上非常接近，這也是吾人在硬體實現上採用 ZF 偵測器的原因。從圖 3-17 模擬結果發現，MMSE 的效能較 V-BLAST 好，這結果與一般的認知相反，這可能是 Soft V-BLAST 在計算 CSI 上並不容易，而假設前一級的決策是正確的情況下，所計算出來的 CSI1 效能又不如預期。而使用 CSI2 的 Soft V-BLAST 所表現出來的效果又和單純使用 MMSE 差不多。從圖 3-18 模擬結果發現，ZF 演算法的效能亦隨著天線數的增加而降低，主要是因為天線數增加，資料傳輸量也跟著增加導致。從圖 3-19 模擬結果發現，ZF 演算法的效能亦隨著通道估測(channel estimation)所造成的 channel noise 增加而降低。

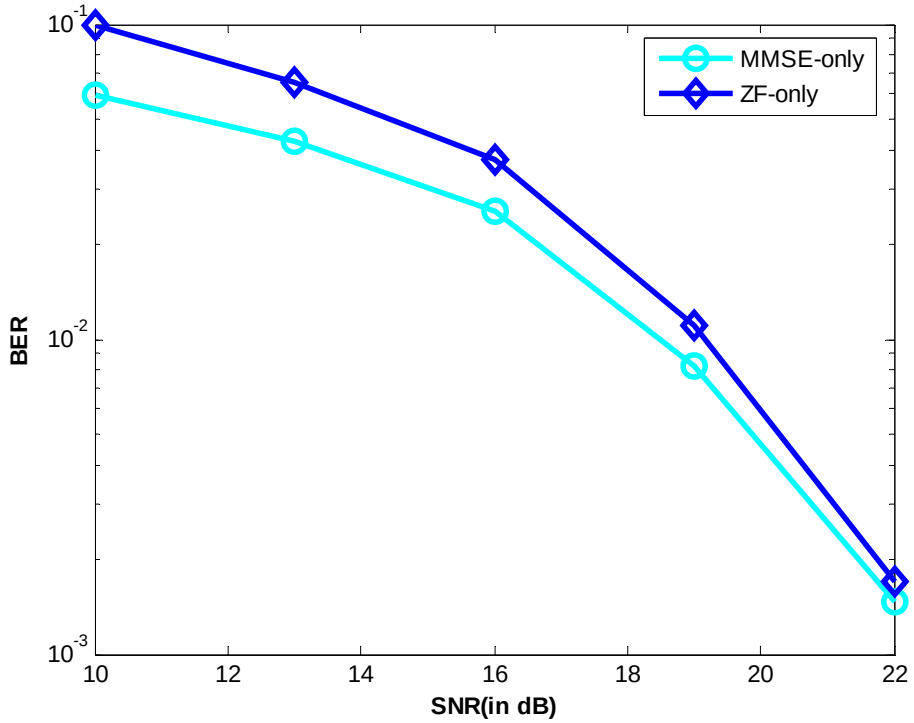


圖 3-14 不接 Viterbi 解碼器而單獨使用 ZF 或 MMSE 之效能比較圖 (64QAM)

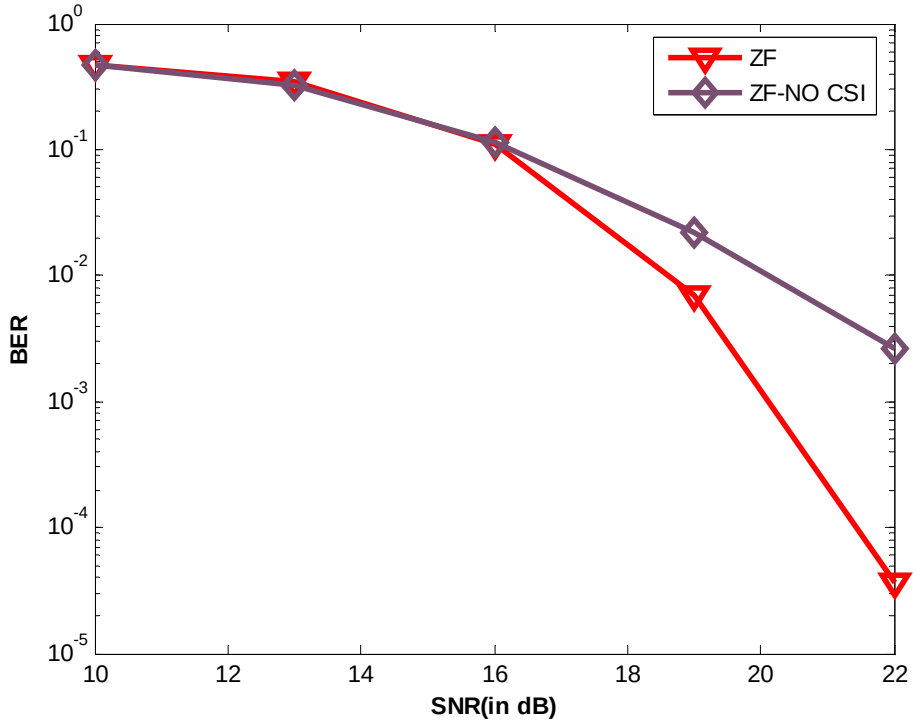


圖 3-15 ZF 之 CSI 與 NO CSI 的比較圖 (64QAM)

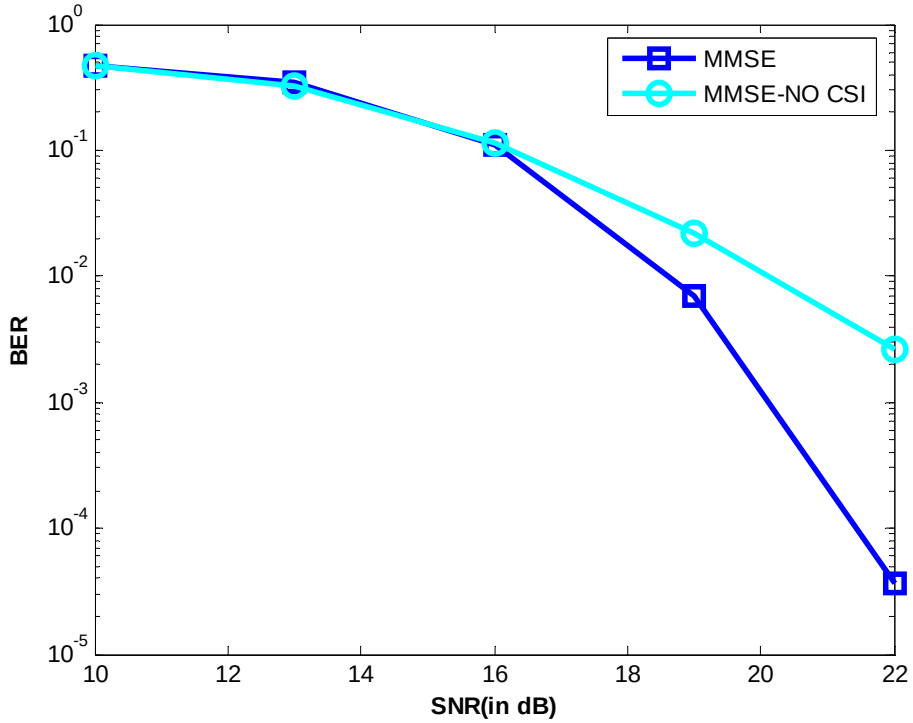


圖 3-16 MMSE 之 CSI 與 NO CSI 的比較圖 (64QAM)

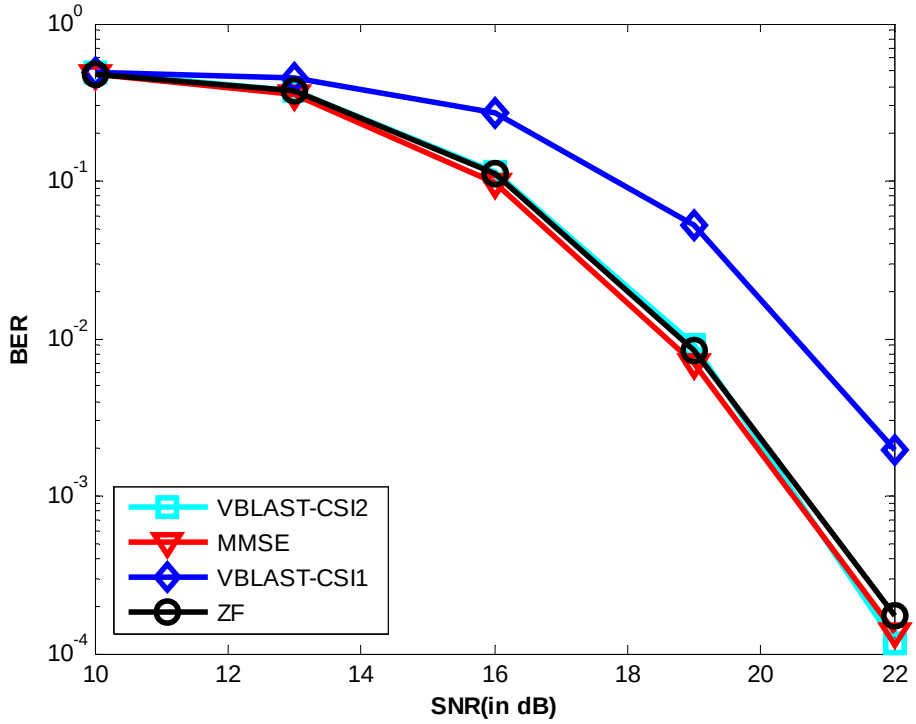


圖 3-17 ZF、MMSE 及 V-BLAST 之比較圖 (64QAM)

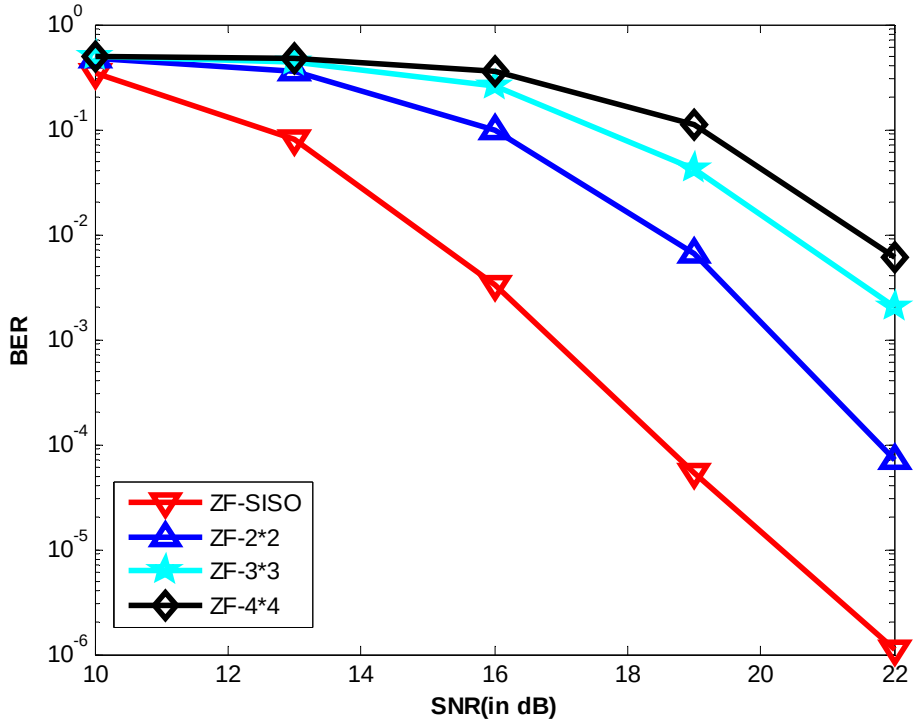


圖 3-18 SISO、2×2、3×3 及 4×4 之比較圖 (64QAM)

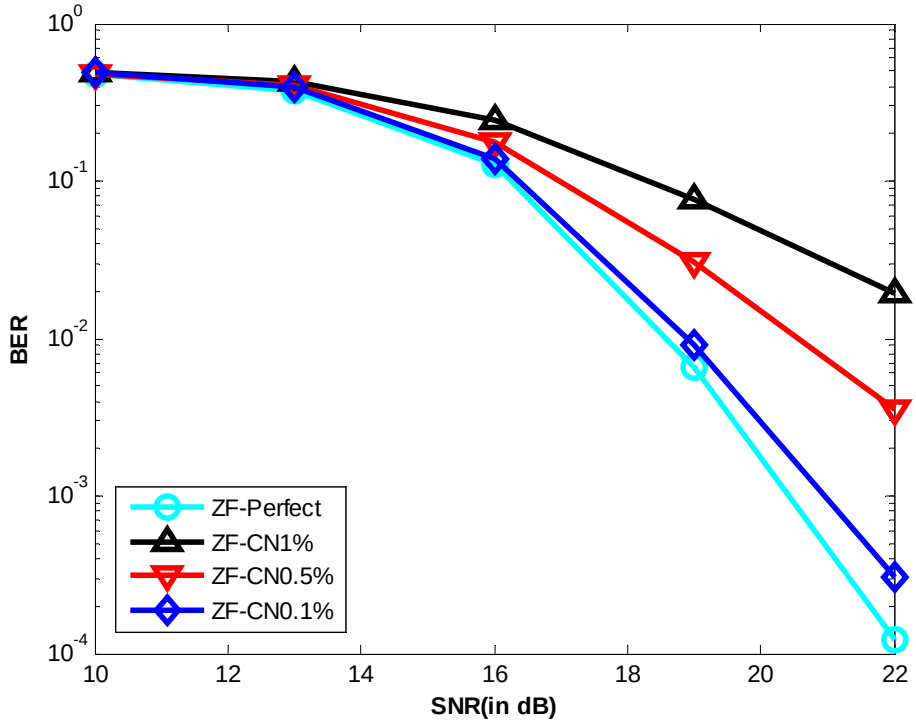


圖 3-19 不同 Channel Noise 下之效能比較圖 (64QAM)

第二部分的模擬則是針對802.11n TGn Sync所定義的通道模型B-F做更進一步的模擬，首先我們先設定系統為2×2的基本模式，並且通道頻寬為20MHz，而且假設通道效應為已知。再則，根據TGn Sync所定義，每96個OFDM symbol通道會變化一次，而通道的tap數會因通道模型B-F而各有不同，另外定義一個packet是由1000Bytes構成，也就是說一個packet有8000bits。我們將採用Packet error rate (PER)做效能之評估標準。圖 3-20是使用ZF演算法和MMSE演算法在不同通道模型下之比較，而使用的編碼率為1/2，其中RAN為第一部分所用的獨立複數高斯隨機變數所產生的通道，而B-F則為802.11n TGn Sync所提出的通道模型B-F。圖 3-21是使用ZF演算法和V-BLAST演算法在不同通道模型下之比較，其中B、D和E分別代表通道模型B、D和E，CSI2表示直接使用第一級MMSE所計算出來的CSI。圖 3-22是使用ZF演算法在獨立複數高斯隨機變數通道與AWGN通道下之比較，其中RAN如上述圖 3-20的說明，而AWGN表示AWGN通道。圖 3-23至圖 3-27是ZF演算法在不同的QAM數和不同的編碼率在相同或不同的通道模型中所表現出的不同效能。其中B、D和E分別代表通道模型B、D和E，而Q1、Q2、Q3和Q4分別代表BPSK、QPSK、16QAM和64QAM，C1、C2、C3和C4則分別代表編碼率1/2、3/4、2/3和5/6。從圖 3-20模擬結果發現，ZF演算法和MMSE演算法在不同的通道下，彼此仍有非常相近的效能，其原因已在第一部分解釋過，而通道模型B-F因為通道之間存在有相關性，故較獨立通道條件下所造成的效能為差，另外，通道模型B-F的效能排序由高到低依序為D、E、B、F和C。從圖 3-21模擬結果發現，V-BLAST演算法除了對通道模型B有較明顯的效能改善外，對通道模型D、E均無明顯的效能改善。從圖 3-22模擬中發現，AWGN通道效能較佳。從圖 3-23至圖 3-27模擬結果發現，不同的QAM數，由於QAM數的增加，造成系統使用更多的bits去表示一個QAM symbol，自然會造成效能的變差，而不同的編碼率，隨著編碼率的變高，將造成打掉不送再於接收端補上無意義的位元之比例增加，造成錯誤率的增加，相對的效能也會因此變差，另外通道模型效能的排序也如圖 3-20模擬結果所示。

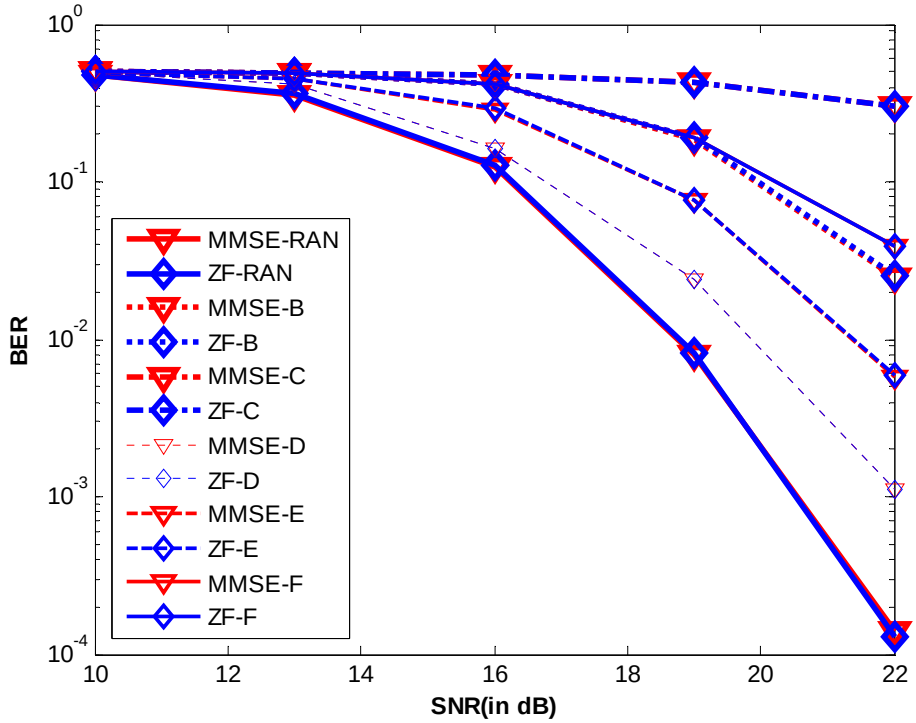


圖 3-20 ZF 與 MMSE 在不同通道之比較圖

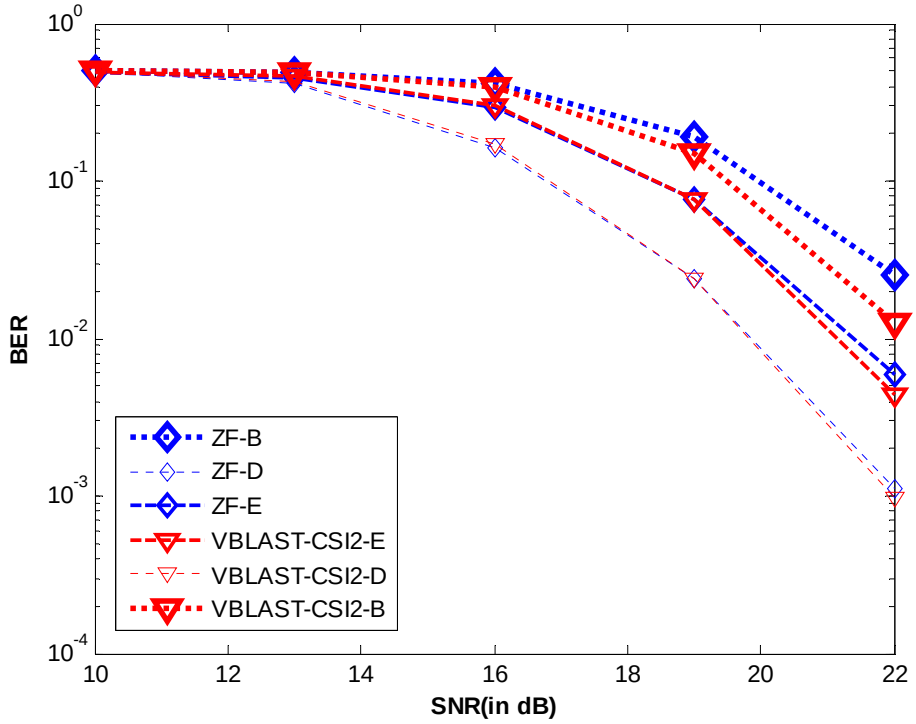


圖 3-21 ZF 與 VBLAST 在不同通道之比較圖

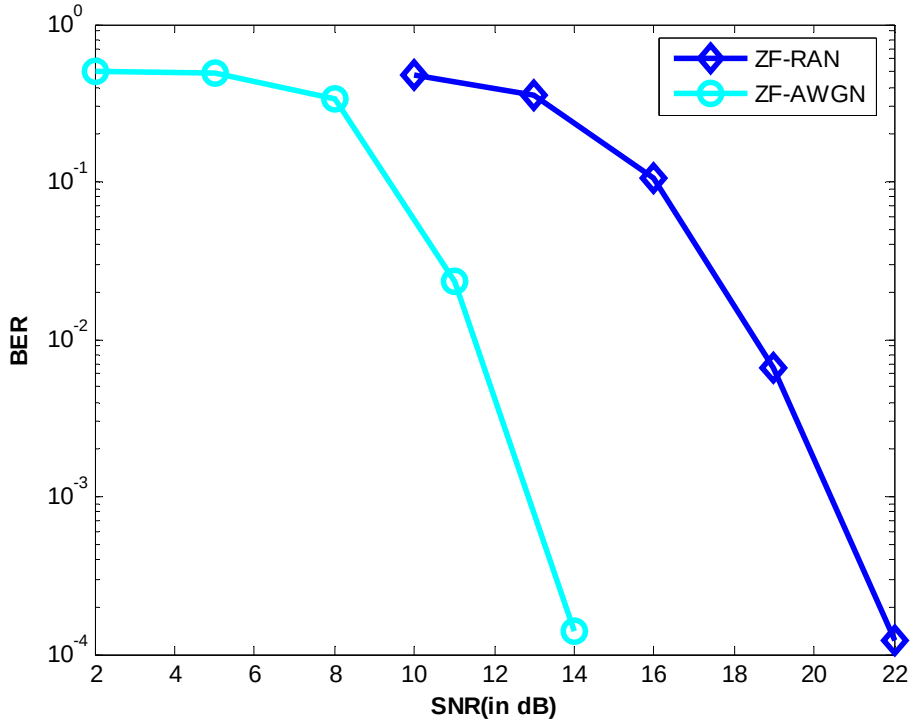


圖 3-22 ZF 在獨立複數高斯隨機變數通道與 AWGN 通道下之比較圖

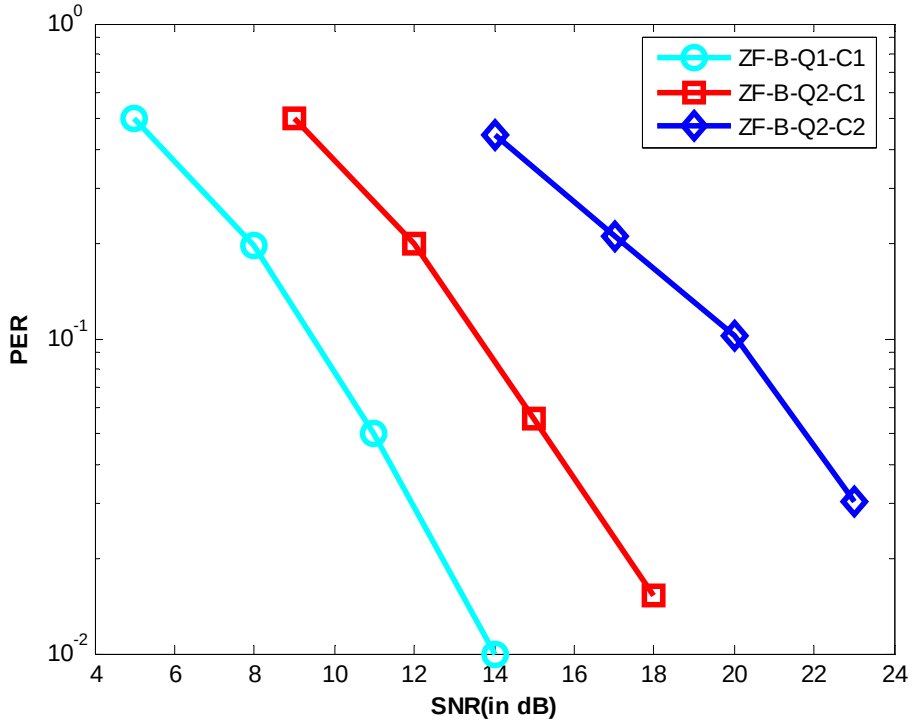


圖 3-23 BPSK 與 QPSK 在不同編碼率下之比較圖

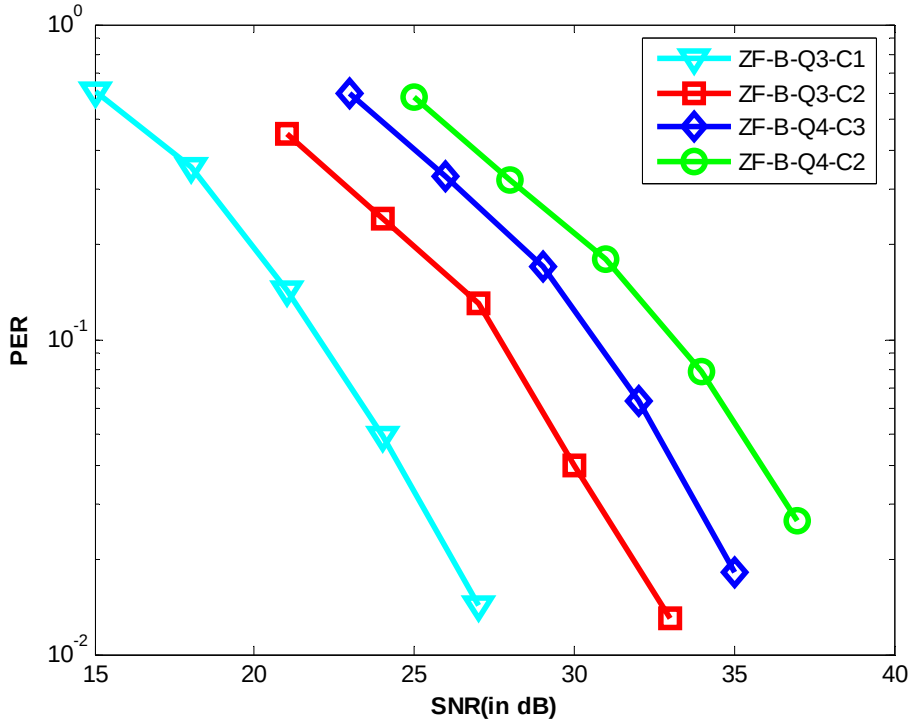


圖 3-24 16QAM 與 64QAM 在不同編碼率下之比較圖

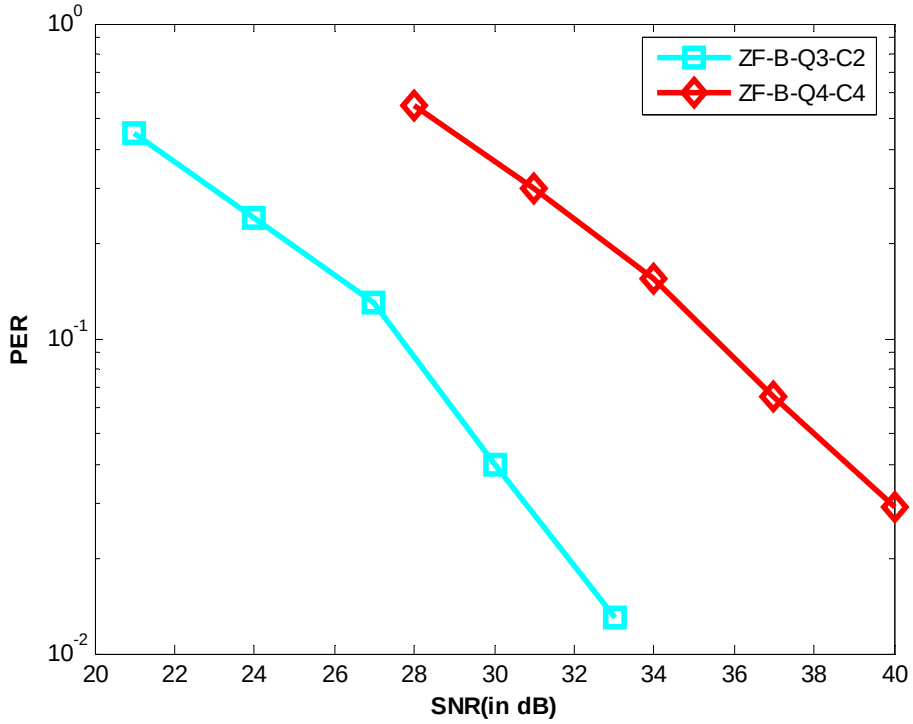


圖 3-25 16QAM 與 64QAM 在不同編碼率下之比較圖(channel B)

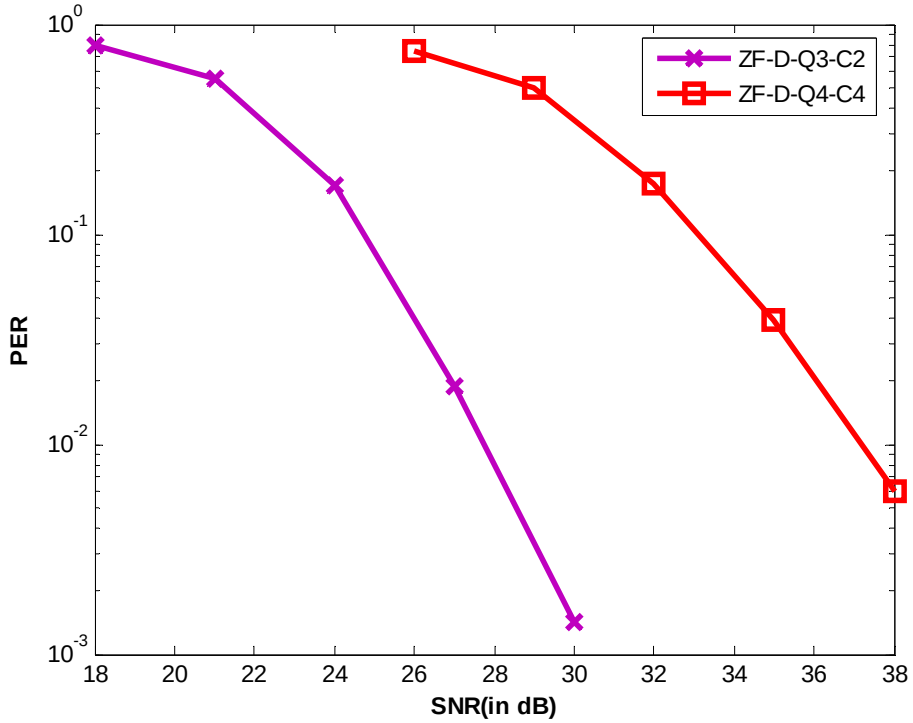


圖 3-26 16QAM 與 64QAM 在不同編碼率下之比較圖(channel D)

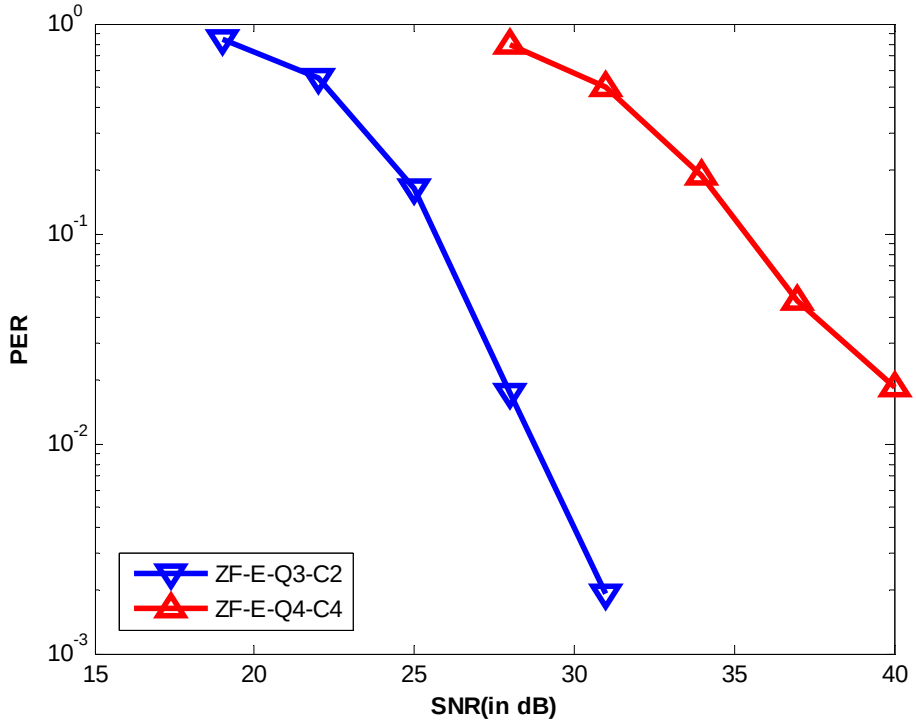


圖 3-27 16QAM 與 64QAM 在不同編碼率下之比較圖(channel E)

第4章硬體實現

4.1 設計流程

吾人使用電腦輔助工具來做模擬硬體實現，設計流程如圖 4-1 所示。設計流程主要分成軟體及硬體二部分，軟體使用 Matlab 模擬而硬體模擬使用 Modelsim 及 ISE 6 模擬。Matlab 模擬包括了浮點系統的模擬及定點(fixed-point)系統的模擬。浮點系統的模擬可以檢查各函數的正確性和設定參數，也可以評估系統的效能。而定點系統的模擬，用來決定各個訊號的所需要的表示位元數(word length)。各個信號表示位元數確定以後，便進入硬體實現模擬。在硬體實現模擬中，吾人使用 Verilog 作為硬體描述語言做程式編寫，且使用 Modelsim 作硬體的行為模擬來驗證 Verilog 程式碼的正確性，用 ISE 6 作為合成軟體。

硬體實現模擬流程中，首先使用硬體描述語言編輯(HDL Editor)模擬電路輸入，此步驟可由兩種方式進行之，第一種方式以硬體模擬為主，但須完成相對應的傳送端硬體，其測試平台(test bench)如圖 4-2 所示，第二種方式為較簡易的方式，不須另外完成傳送端硬體，其 test bench 如圖 4-3 所示。將訊號輸入電路後，接下來可做行為模擬(Behavioral simulation)，這可以快速的了解所設計電路正確與否，並做修改更正。在這一步驟時，會參照由 Matlab 定點系統模擬所產生的數據來幫助驗證函數功能正確。當硬體的行為模擬正確之後，便使用合成軟體 ISE 6 加入 VirtexE xcv-2000e 的 library 將 Verilog 程式轉為 register transfer level (RTL)，合成完之後，就可做時序模擬(Timing simulation)，模擬所設計電路再燒入 FPGA 晶片後，所造成的邏輯延遲及繞線延遲是否符合所需，在此便完成了所有的硬體模擬。

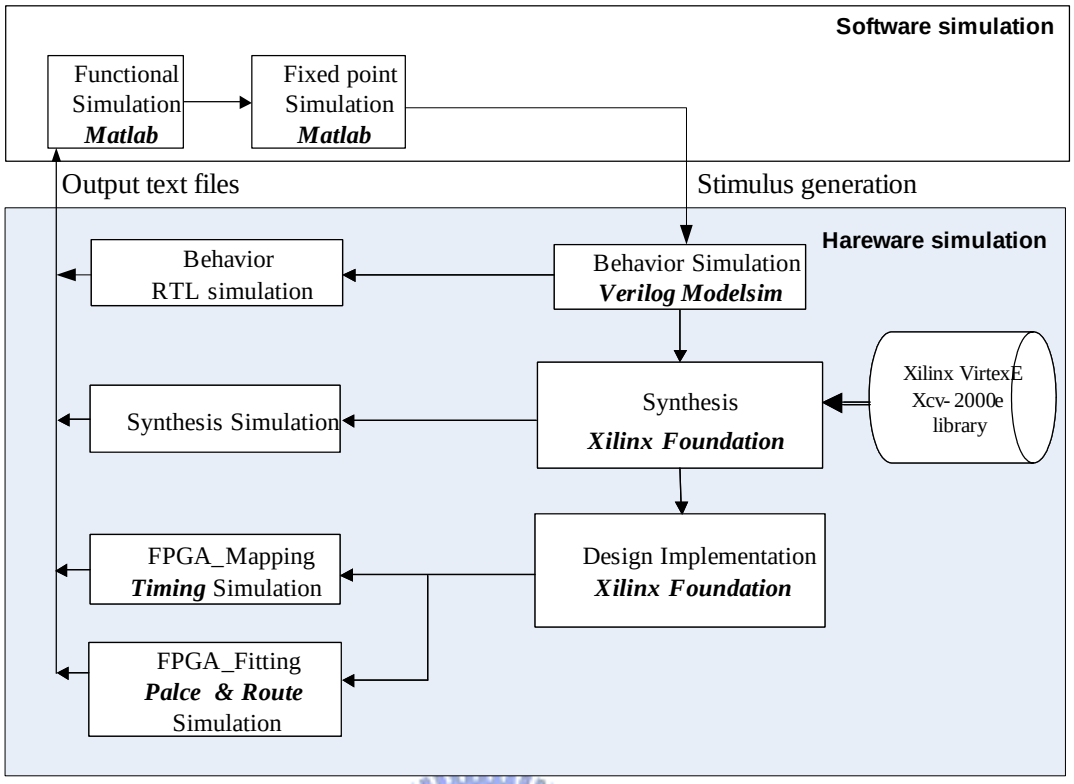


圖 4-1 硬體實現流程圖

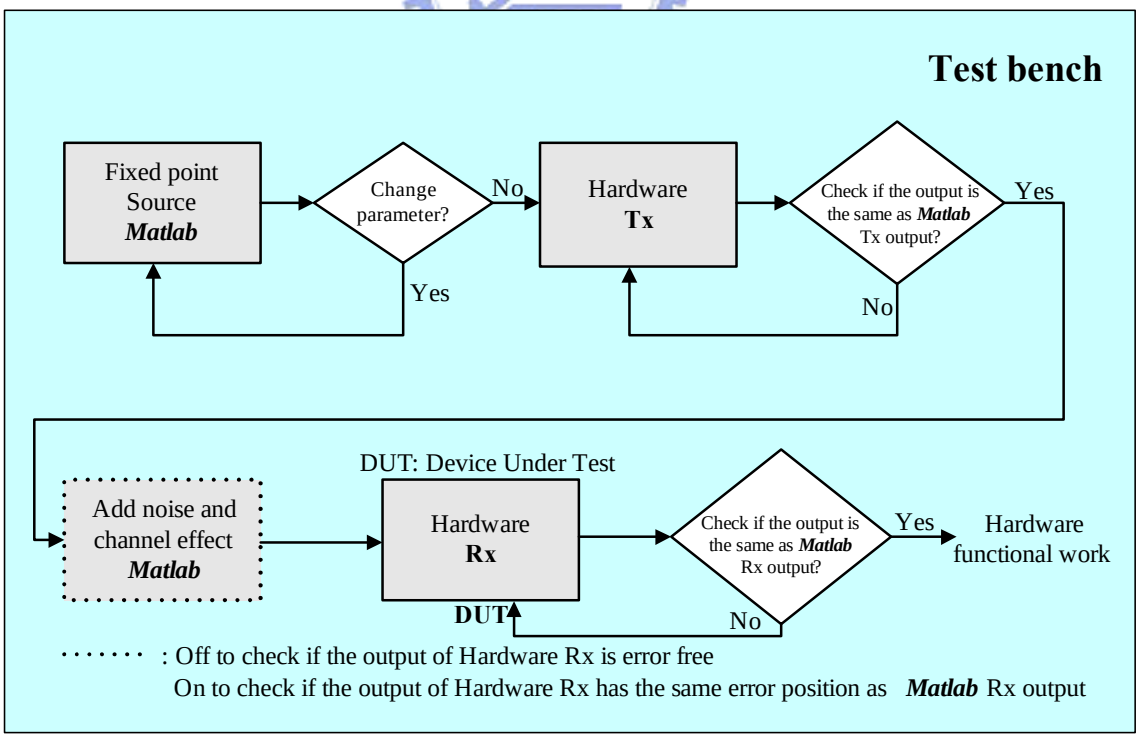


圖 4-2 以硬體模擬為主之 Test bench

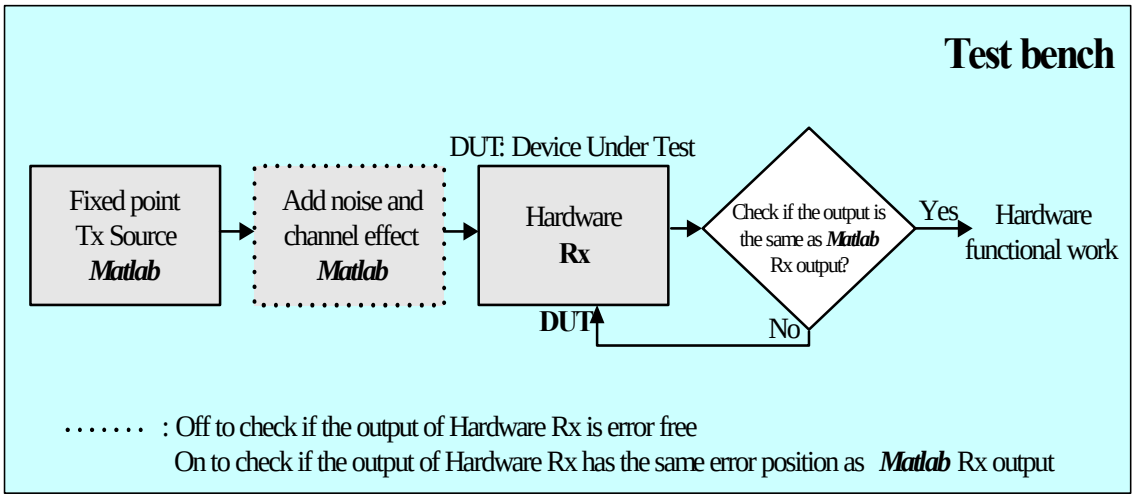


圖 4-3 僅含 DUT 較簡易之 Test bench

4.2 組成元件設計

吾人實作的接收機便如圖 4-4 所示，這是在 3.1.5 軟性輸入軟性輸出之 ZF 接收機那一節中所介紹的接收機的簡化版，主要實作元件包括了 Zero forcing 估測(ZF)、軟性反對映(Soft demapping)、反交錯器(De-interleaver)及 Viterbi 解碼器。這樣的模式是在 11n 提案中基本的模式(2 根接收天線、20MHZ、編碼率 1/2、64QAM)，訊號流程大致和 3.1.5 中所推導的相同。底下則對這種情況下的訊號處理及在為了實作上的需求而做的修改做詳細介紹。在 4.2 中的安排中，會先對所要實作的接收機訊號流程做介紹，再分別介紹各元件如何實作。

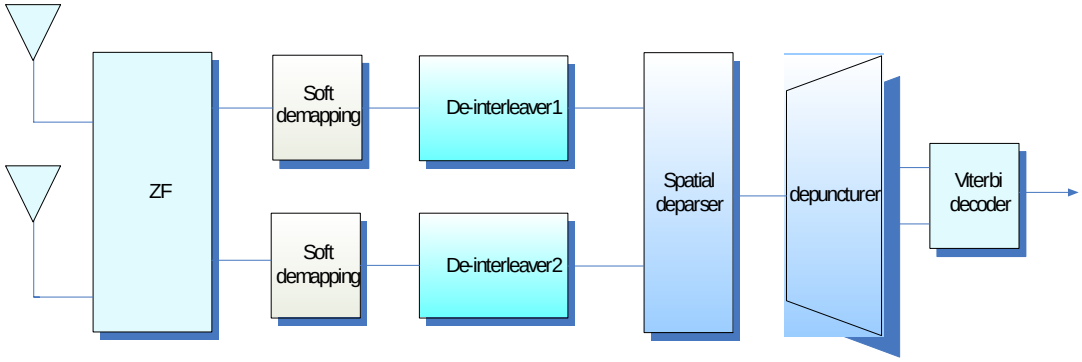


圖 4-4 實作之接收器方塊圖

4.2.1 接收機訊號流程

如同在 3.1.5 推導的，接收訊號在2×2 的情況下可以表示為:

$$Y_k = \begin{bmatrix} h_{1,k} & h_{2,k} \end{bmatrix} \begin{bmatrix} s_{1,k} \\ s_{2,k} \end{bmatrix} + N_k \tag{4-1}$$

$h_{i,k}$ 是 $s_{i,k}$ 的通道影響，也就是 H 的第 i 列。而 ZF 的等化器則可以表示為

$$W = (H_k)^{-1} = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix} \tag{4-2}$$

則此等化過後的訊號即變成 $Z_k = WY_k$，假設 w_i^T 是 W 的第 i 行，則對第 i 根天線而言，等化後的訊號即可表示為:

$$Z_{i,k} = s_{i,k} + \underbrace{w_i^T N_k}_{N_{eff}} \tag{4-3}$$

如此一來，$\sigma_{N_{eff}}^2 = \left\| w_i \right\|^2 N_0$，由等化過後的在第 i 根天線下第 k 個子載波的 QAM symbol($Z_{i,k}$)要反對映到位元($v_{k,i}^n$)時，相對於(3-21)可得:

$$v_{k,i}^n = \mathrm{CSI_{k,i}} \times D(Z_{i,k}, n) \tag{4-4}$$

n 代表著第幾個位元，在吾人的實作中使用 64QAM 為主，所以 $n = 1 \cdots 6$。而 CSI 則可以代入其定義:

$$CSI_{k,i} = \frac{2}{\sigma_{N_{eff}}^2} = \frac{2}{\left\| w_i \right\|^2 N_0} \tag{4-5}$$

而 $D(Z_{i,k}, n)$ 在 64QAM 如同在 3.1 節中推導可表示成:

$$D(Z_{i,k},n)=\begin{cases} R\{Z_{i,k}\} & n=1 \\ -\left|R\{Z_{i,k}\}\right|+4 & n=2 \\ -\left|\left|R\{Z_{i,k}\}\right|-4\right|+2 & n=3 \\ I\{Z_{i,k}\} & n=4 \\ -\left|I\{Z_{i,k}\}\right|+4 & n=5 \\ -\left|\left|I\{Z_{i,k}\}\right|-4\right|+2 & n=6 \end{cases} \tag{4-6}$$

$R\{Z_{i,k}\}$是指$Z_{i,k}$的實部，而$I\{Z_{i,k}\}$則是指$Z_{i,k}$的虛部。再來位元訊號經過反交錯器及 Viterbi 解碼器後便完成了整個訊號處理。

我們整理上述的訊號處理流程，接收訊號首先會經過 ZF 將混合在一起的訊號分開二個獨立處理的資料流，因此必須利用(4-2)算出由在頻域的通道係數所組成的 ZF 等化矩陣，再來便是使用(4-5)算出由 ZF 等化矩陣係數所組成的 CSI。因此在吾人的實作中，圖 4-4 中的 ZF 方塊，即是先算出由 52 組(11n 提案中所定義的一個 OFDM symbol 中的資料個數) 2×2 頻域通道係數所組成的 52 組 2×2 ZF 等化器係數，及 52 組 2×1 的 CSI，然後將這 52 組等化器係數及 CSI 放在 RAM 中等待之後的接收訊號使用。而圖 4-4 中的 Soft Demapping 方塊便是接收由 ZF 方塊算出的 CSI 及等化過後訊號再利用(4-4)及(4-6)算出位元訊號v。最後再經過反交錯器及解碼器便完成了整個接收機了。我們將於 4.2.2-4.2.6 中詳細介紹 ZF 方塊、Soft Demapping、反交錯器、反壓縮器及 Viterbi 解碼器是如何實作出來的。

4.2.2 Zero forcing 估測

我們令在第 k 個子載波上的通道矩陣係數H_k為:

$$H_k=\begin{bmatrix} h_{k,1} & h_{k,2} \\ h_{k,3} & h_{k,4} \end{bmatrix} \tag{4-7}$$

因為底下的推導都是在同樣的子載波上，為了表示方便，我們令 $h_{k,m} = h_m$，則通道矩陣可以改寫為:

$$H_k = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix}$$

由於 ZF 偵測器再經過 Viterbi 解碼器與 MMSE 偵測器再經過 Viterbi 解碼器，在前一章模擬中已得會有非常相近的效能，故在本節中將針對 ZF 偵測器與 MMSE 偵測器的複雜度做一比較，以確定 ZF 偵測器相對於 MMSE 偵測器有較低的複雜度。

由(4-2)，可得所要求的 ZF 等化器的矩陣係數則可寫成:

$$\begin{aligned} W &= \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix} \\ &= (H_k)^{-1} \\ &= \left(\begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix} \right)^{-1} \end{aligned} \tag{4-8}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \frac{1}{AD - BC} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \tag{4-9}$$

利用(4-9) 2×2 的反矩陣公式則(4-8)可改寫成(4-10)及(4-11)

$$\begin{aligned} W &= \left(\begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix} \right)^{-1} \\ &= \frac{1}{g} \begin{bmatrix} h_4 & -h_2 \\ -h_3 & h_1 \end{bmatrix} \\ &= \frac{1}{|g|^2} \left(\begin{bmatrix} g^* h_4 & -g^* h_2 \\ -g^* h_3 & g^* h_1 \end{bmatrix} \right) \end{aligned} \tag{4-10}$$

$g = h_1 h_4 - h_2 h_3$ 且

$$|g|^2 = gg^* = (h_1 h_4 - h_2 h_3)(h_1 h_4 - h_2 h_3)^* = (h_1^* h_1)(h_4^* h_4) + (h_2^* h_2)(h_3^* h_3) - (h_1^* h_4^*)(h_2 h_3) - (h_2^* h_3^*)(h_1 h_4) \tag{4-11}$$

從(4-10)中發現，吾人將W的分母項表示成$|g|^2$，最主要是因為g有可能是一個複數，而我們希望將W之分母吸收至軟性反對應的計算當中，以避免除法的運算，這只有在此分母為實數時才有可能，因此需要將g乘以g^*。相對的，對於MMSE 偵測器，令$\alpha = \frac{2}{\rho}$，如此由(3-33)，所要求的 MMSE 等化器的矩陣係數則可寫成:

$$\begin{aligned} W &= \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix} \\ &= (H_k^H H_k + \frac{2}{\rho} I)^{-1} H_k^H \\ &= \left(\begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix} + \alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \\ &= \begin{pmatrix} h_1^* h_1 + h_3^* h_3 + \alpha & h_1^* h_2 + h_3^* h_4 \\ h_2^* h_1 + h_4^* h_3 & h_2^* h_2 + h_4^* h_4 + \alpha \end{pmatrix}^{-1} \begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \end{aligned} \tag{4-12}$$

利用(4-9) 2×2 的反矩陣公式則(4-12)可改寫成(4-13)及(4-14)

$$\begin{aligned} W &= \begin{pmatrix} h_1^* h_1 + h_3^* h_3 + \alpha & h_1^* h_2 + h_3^* h_4 \\ h_2^* h_1 + h_4^* h_3 & h_2^* h_2 + h_4^* h_4 + \alpha \end{pmatrix}^{-1} \begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \\ &= \frac{1}{g} \begin{bmatrix} h_2^* h_2 + h_4^* h_4 + \alpha & -(h_1^* h_2 + h_3^* h_4) \\ -(h_2^* h_1 + h_4^* h_3) & h_1^* h_1 + h_3^* h_3 + \alpha \end{bmatrix} \begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \\ &= \frac{1}{g} \left(\begin{bmatrix} h_1^*(h_4^* h_4) - h_4(h_2^* h_3^*) & h_3^*(h_2^* h_2) - h_2(h_1^* h_4^*) \\ h_2^*(h_3^* h_3) - h_3(h_4^* h_1^*) & h_4^*(h_1^* h_1) - h_1(h_2^* h_3^*) \end{bmatrix} + \alpha \begin{bmatrix} h_1^* & h_3^* \\ h_2^* & h_4^* \end{bmatrix} \right) \end{aligned} \tag{4-13}$$

$$g = (h_1^* h_1)(h_4^* h_4) + (h_2^* h_2)(h_3^* h_3) - (h_1^* h_4^*)(h_2 h_3) - (h_2^* h_3^*)(h_1 h_4) + \alpha(h_1^* h_1 + h_2^* h_2 + h_3^* h_3 + h_4^* h_4) + \alpha^2 \tag{4-14}$$

比較(4-10)，(4-11)及(4-13)，(4-14)可得 ZF 偵測器及 MMSE 偵測器的複雜度比較，如下表所示:

	除法器	乘法器	加(減)法器
W of ZF	4	4	2
W of MMSE	4	8	6
$\|g\|^2$ **of ZF**	0	3	1
g **of MMSE**	0	9	6

表 4-1 單一子載波上，ZF 及 MMSE 複雜度之比較

然而在(4-10)或表 4-1 中，W 要算出來需要四個複數除法，因此我們讓 ZF 矩陣係數整個放大$|g|^2$倍，而避免除法:

$$W' = \begin{bmatrix} w'^{T}_{1} \\ w'^{T}_{2} \end{bmatrix} = |g|^2 W = \left(\begin{bmatrix} g^* h_4 & -g^* h_2 \\ -g^* h_3 & g^* h_1 \end{bmatrix} \right) \tag{4-15}$$

由(4-15)算出放大的 ZF 矩陣後，則第一根天線的 CSI_1 及第二根天線的 CSI_2 便可由(4-5)算出:

$$CSI_1 = \frac{2}{\sigma_{N_{eff}}^{\ 2}} = \frac{2}{\left\| w'_1 \right\|^2 N_0}$$

$$CSI_2 = \frac{2}{\sigma_{N_{eff}}^{\ 2}} = \frac{2}{\left\| w'_2 \right\|^2 N_0}$$

將 CSI 中的 2 及 N_0 正規化後，並考慮放大$|g|^2$倍所造成的整體影響，我們可以得最後 CSI 的計算方式:

$$CSI_1 = \frac{|g|^2}{w'^{\ 2}_{11} + w'^{\ 2}_{12}}$$

$$CSI_2 = \frac{|g|^2}{w'^{\ 2}_{21} + w'^{\ 2}_{22}} \tag{4-16}$$

如果 $W' = \begin{bmatrix} w'_{11} & w'_{12} \\ w'_{21} & w'_{22} \end{bmatrix}$。

由表 4-1 中，吾人可以確定 ZF 偵測器相對於 MMSE 偵測器有較低的複雜度。

另外如(4-16)所示，ZF 之 CSI 即是令 MMSE 之 CSI 的 $\sigma^2_{N_{eff}} = \|w_i\|^2 N_0$ 且 $H_{eff} = 1$ 所得到的簡化結果，故 ZF 在 CSI 上亦可得到較 MMSE 簡化的結果，至於 CSI 中所使用到的除法如何避免掉以及 W 為奇異矩陣時要如何處理，將在 4.2.4 中做較詳細的介紹。

ZF 方塊圖如圖 4-5 所示: 在 ZF 這個方塊中，一開始由頻域上通道係數計算 52 組的 W'、CSI 及 $|g|^2$ 並將這些值存在 RAM 中，而之後資料訊號 Y 進來時便從 RAM 中讀出 W' 由 $Z' = W'Y$ 做矩陣相乘而可得到 Z'。而 Z' 便可進入軟性反對映了。值得注意的是，ZF 這樣子的計算複雜度還是太高，於是在實現時吾人提高 ZF 系統的時脈變為原來的四倍，共用乘法器而能有效減小面積。

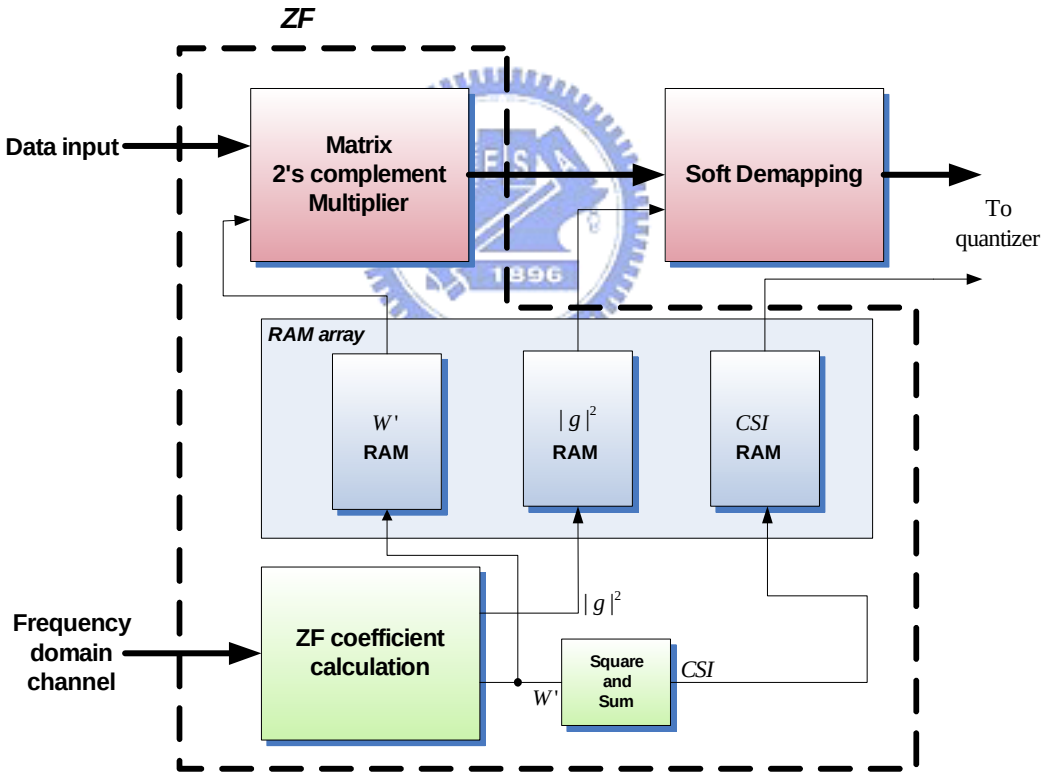


圖 4-5 ZF 結構方塊圖

4.2.3 Soft Demapping

如 4.2.2 所示，使用W'取代W會使得訊號經過 ZF 等化過後放大$|g|^2$倍，而且 CSI 中的分子項亦放大了$|g|^2$倍，然而我們發現在軟性反對映中，只需要修改(4-6)為(4-17)便不影響其機率值。

$$D(Z_{i,k},n)=\begin{cases} |g|^2\ R\{Z_{i,k}\} & n=1 \\ -\left||g|^2\ R\{Z_{i,k}\}\right|+4|g|^4 & n=2 \\ -\left|\left||g|^2\ R\{Z_{i,k}\}\right|-4|g|^4\right|+2|g|^4 & n=3 \\ |g|^2\ I\{Z_{i,k}\} & n=4 \\ -\left||g|^2\ I\{Z_{i,k}\}\right|+4|g|^4 & n=5 \\ -\left|\left||g|^2\ I\{Z_{i,k}\}\right|-4|g|^4\right|+2|g|^4 & n=6 \end{cases} \tag{4-17}$$

由於一個 QAM symbol 隨著 QAM 數的增加，所轉換成的機率位元數也會增加，如(4-17)中，因為 QAM 數為 64QAM，故其所轉換成的機率位元數就是 6 位元。當然這就造成在硬體實現上，輸入端和輸出端的時脈速度不相同，在(4-17)中，輸出端的時脈速度就是輸入端的 6 倍，最好的非同步時脈的處理方式為使用 RAM 當緩衝器(buffer)，以達到並行到序列轉換(Parallel to serial mapping)的目的。另外針對非同步時脈間的控制訊號，務須做到雙重同步(Double synchronization)，以避免設定時間違背(set-up time violation)，雙重同步如圖 4-6 所示。而 Soft Demapping 方塊圖如圖 4-7 所示。

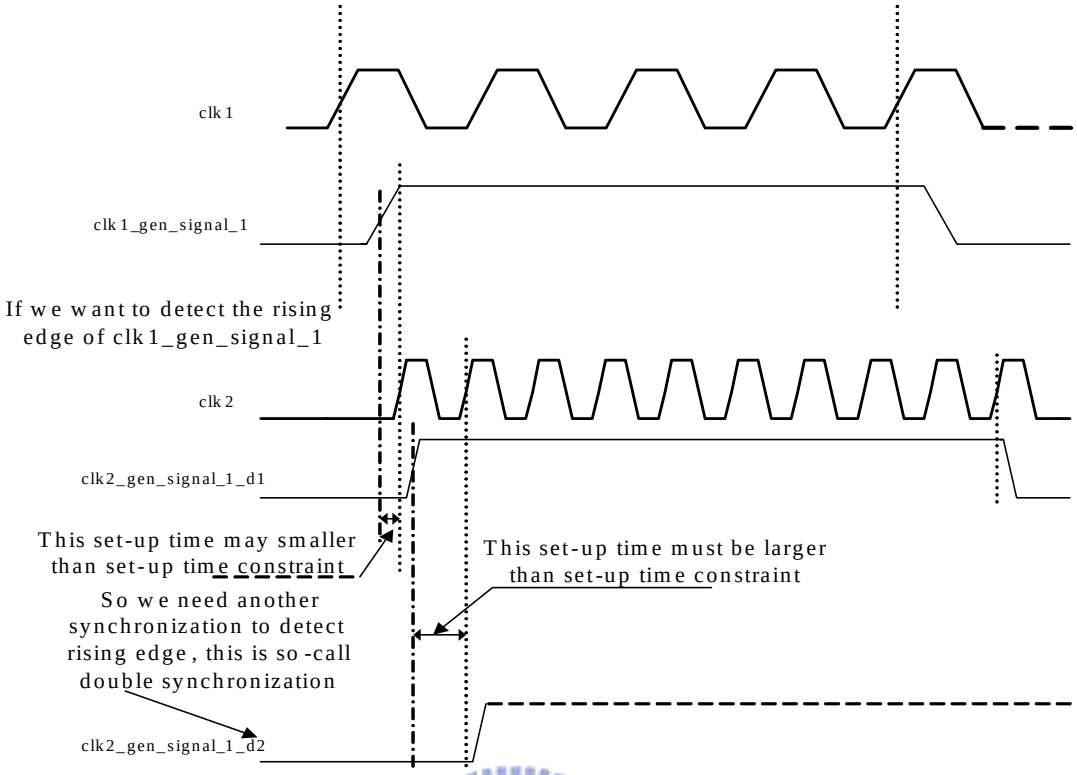


圖 4-6 雙重同步 Timing Diagram

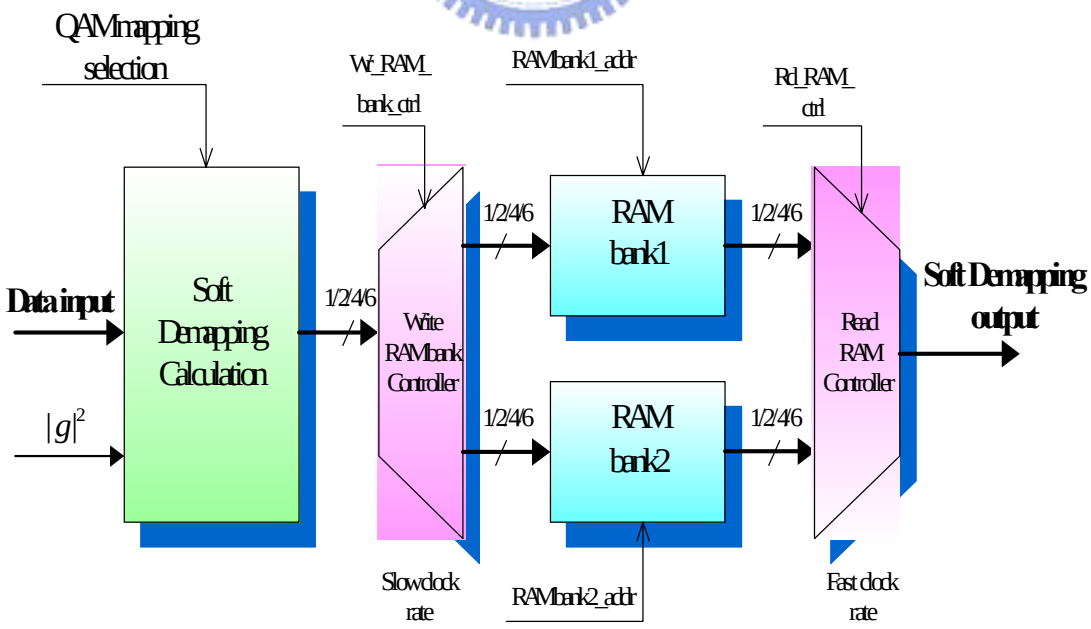


圖 4-7 Soft Demapping 方塊圖

4.2.4 反交錯器

如同 2.2.2.3 中所描述，反交錯器是由(2-13)至(2-15)所定義的三次對調所組成。假設在第一次對調之前資料的索引(index)為 r，而在第一次對調之後第二次對調之前的索引為 j，i 則為第二次對調及第三次對調之間的索引，k 是第三次對調之後 QAM 的反對映之前的索引。三次對調如下所示:

$$j=\left(r+\left(\left(2\times i_{ss}\right)\bmod 3+3\times \text{floor}(i_{ss}/3)\right)\times N_{rot}\times N_{BPSC}\right)\bmod N_{CBPS},\qquad r=0,1,\ldots,N_{CBPS}-1$$

$$i=s\times\text{floor}\left(j/s\right)+\left(j+\text{floor}\left(N_{column}\times j/N_{CBPS}\right)\right)\bmod s,\qquad j=0,1,\ldots,N_{CBPS}-1$$

$$k=N_{column}\times i-\left(N_{CBPS}-1\right)\text{floor}\left(i/N_{row}\right),\qquad i=0,1,\ldots,N_{CBPS}-1$$

我們使用四塊 RAM 來完成上述的對調。然而在吾人的實作中，使用 64QAM 及 2 根接收天線及 20MHz 所以上述三次對調所需要的參數便可以決定為:

$$N_{BPSC}=6$$

$$N_{CBPS}=52\times N_{BPSC}=312$$

$$s=\max(\frac{N_{BPSC}}{2},1)=3$$

$$N_{row}=13$$

$$N_{colmn}=4\times N_{BPSC}=24$$

$$N_{rot}=11$$

如此，我們便可以將二根天線個別經過三次對調的索引值存在 ROM 中，當作所使用的 RAM 的讀取位址。如圖 4-8 所示。

另外，在反交錯器的輸入端亦需做一些處理，其主要因為 Viterbi 解碼器並不需要非常精確的定點數位元來加以表示，而只需要 8 階(level)，也就是 3 位元的定點數來加以表示[18]，其模擬將在 4.3 節中介紹，而從反交錯器的輸入端之後到 Viterbi 解碼器輸入端之前，資料本身就不再做任何運算，僅就位置上做一些交錯，故可在反交錯器的輸入端前加上一個量化器(Quantizer)，即可節省非常多的定點數位元，進而達到節省面積的目的。量化器區間規格如圖 4-9 所示。由

(4-4)及(4-16)，並取(4-16)的 CSI_1 及圖 4-9 的區間 5 為例，可得下式，其中 $Q[v_{k,i}^n]$ 為 $v_{k,i}^n$ 經過量化器後的輸出，<=> 表示等效(equivalent):

$$Q[v_{k,i}^n]=5, \text{if } 0.5<v_{k,i}^n \le 1.0 \ ; \text{in this case } v_{k,i}^n=\text{CSI}_1 \times D(Z_{i,k},n)$$

$$<=> Q[v_{k,i}^n]=5, \text{if } 0.5<\text{CSI}_1 \times D(Z_{i,k},n) \le 1.0 \ ; D(Z_{i,k},n) \text{ as (4-17)}, CSI_1=\frac{1}{w'_{11}{}^2+w'_{12}{}^2}$$

$$<=> Q[v_{k,i}^n]=5, \text{if } (0.5\times(w'_{11}{}^2+w'_{12}{}^2))<D(Z_{i,k},n) \le (1.0\times(w'_{11}{}^2+w'_{12}{}^2)) \quad (4\text{-}18)$$

由(4-18)的等效過程中可得，原本 CSI 所使用到的除法，已在等效過程中轉換成原分母項 $w'_{11}{}^2+w'_{12}{}^2$ 的乘法，如此一來即可避免掉 CSI 的除法，由(4-10)中可知，當 $|g|^2=0$ 時，W 則為奇異矩陣，此時須令 $w'_{11}{}^2+w'_{12}{}^2=1$，讓所有的資料全部設置為零，這樣即解決了 W 為奇異矩陣所產生的問題了。量化器區間規格可以修正如圖 4-10 所示。

在反交錯器之後，尚須加上反空間分離器(Spatial deparser)，其主要是將原本在兩條天線上的資料流，按照 $s=\max(\frac{N_{BPSC}}{2},1)=3$ 的方式，合併為單一資料流，輸入反壓縮器，反空間分離器所造成的資料流合併，如圖 4-11 所示。而反空間分離器的方塊圖，如圖 4-8 所示。整體反交錯器結構方塊圖也如圖 4-8 所示。

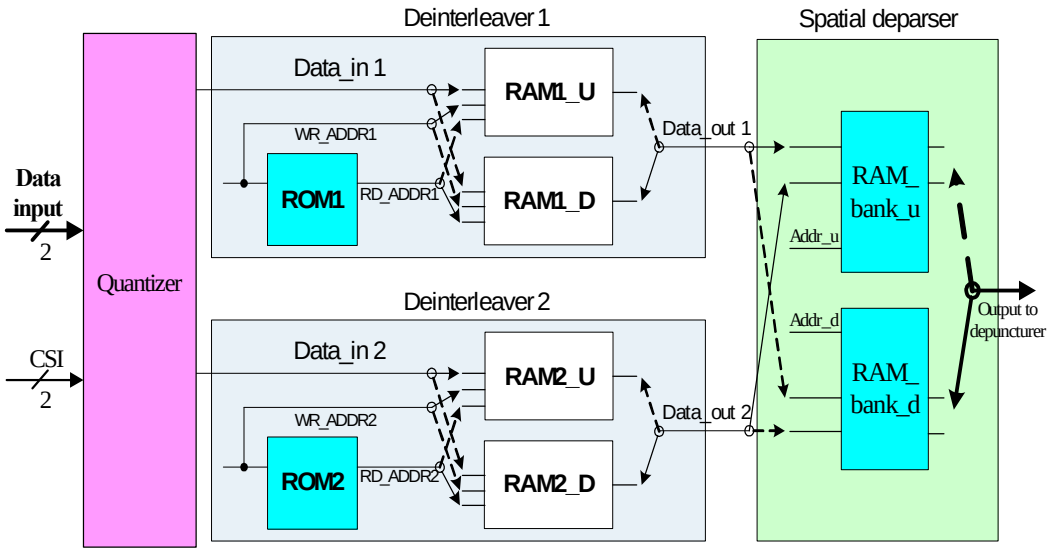


圖 4-8 反交錯器結構方塊圖

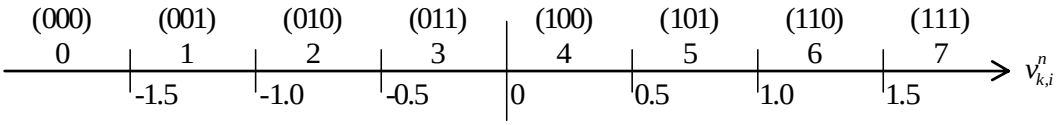
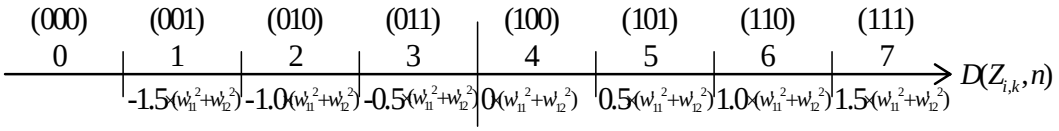


圖 4-9 量化器區間規格



If W is singular, we must let $w'_{11}{}^2+w'_{12}{}^2=1$

圖 4-10 量化器區間規格 – Modified

If $s=\max(\frac{N_{BPSC}}{2},1)=3$ and $N_r=2$, the data flow combination is as follow

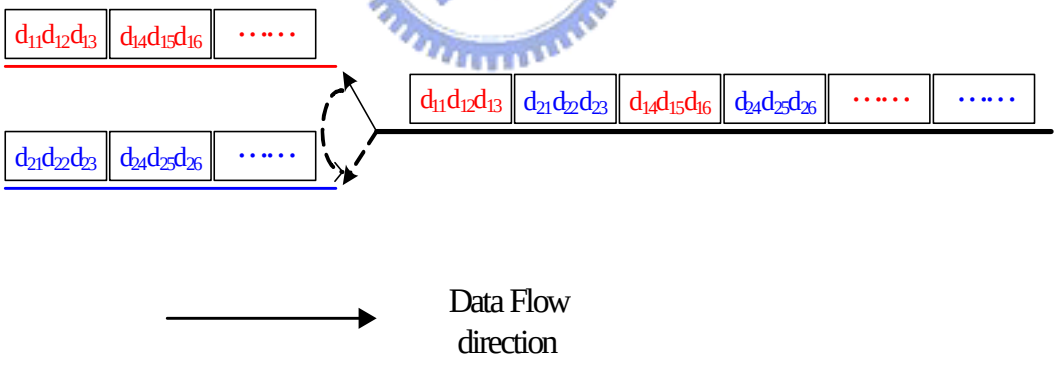
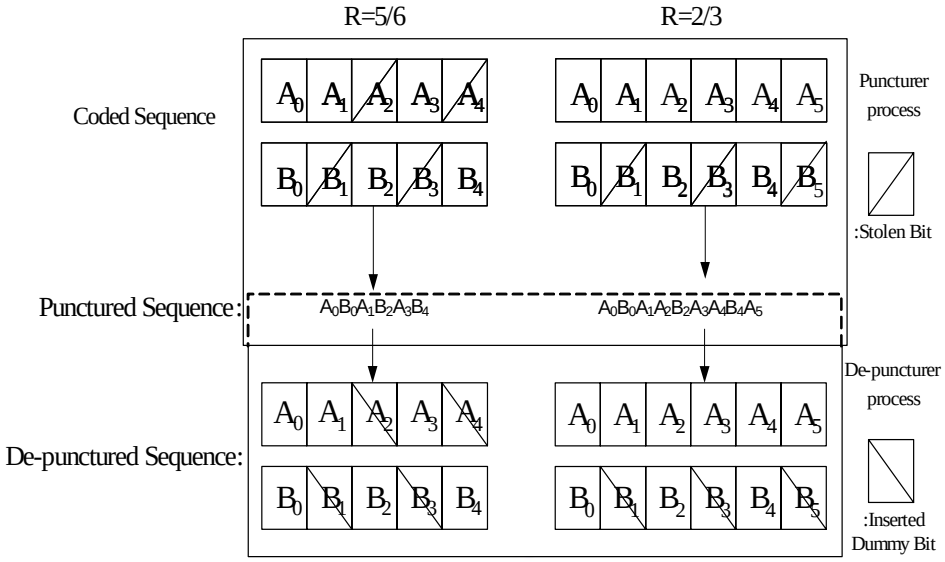


圖 4-11 資料流合併示意圖

4.2.5 反壓縮器

反壓縮器為壓縮器的反運作，其主要將原本經由壓縮器打掉的位元，在相同的位置上再補上無意義的位元。吾人欲使用 3 個位元來表示 Viterbi 解碼器的輸入位元，故可用 $(8)_{10} = (1000)_2$ 四個位元來表示補上的無意義位元。反壓縮流程圖如圖 4-12 所示。



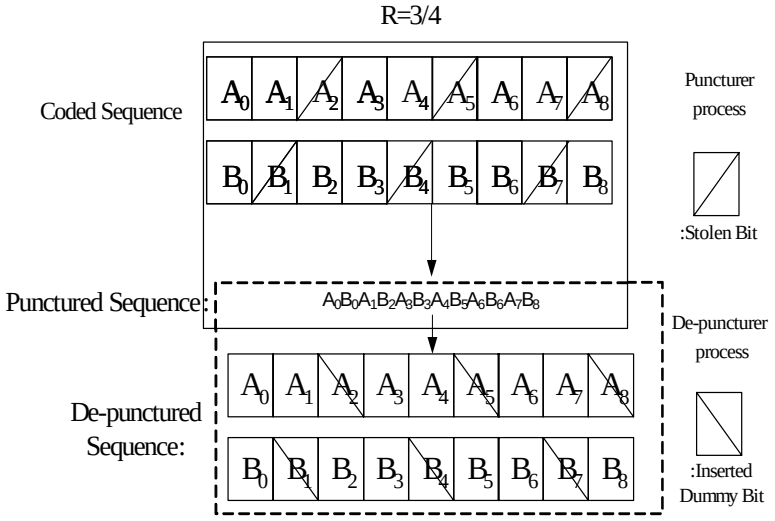


圖 4-12 反壓縮流程圖

而在反壓縮器中最主要的方塊為延遲單元(Delay Unit)，其功用為將輸入訊號做延遲，以便於適當位置加上無意義的位元，再重新組合，然後輸入 RAM 中。延遲單元功能示意圖如圖 4-13 所示。而反壓縮器方塊圖如圖 4-14 所示。

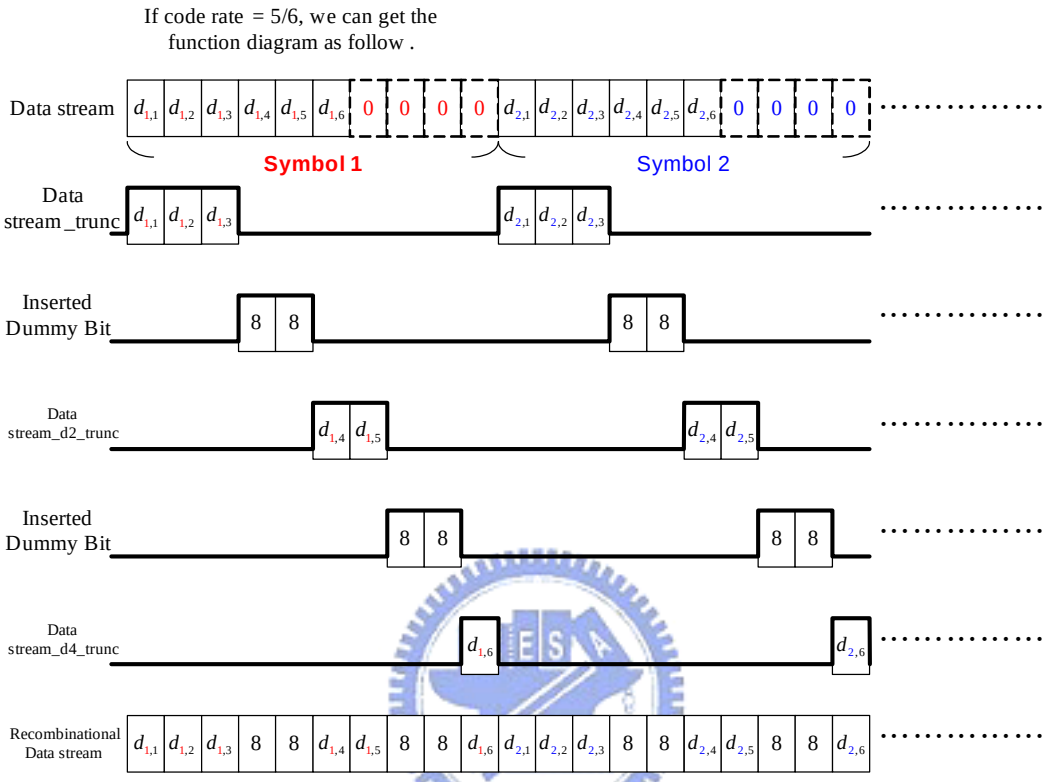


圖 4-13 延遲單元功能示意圖

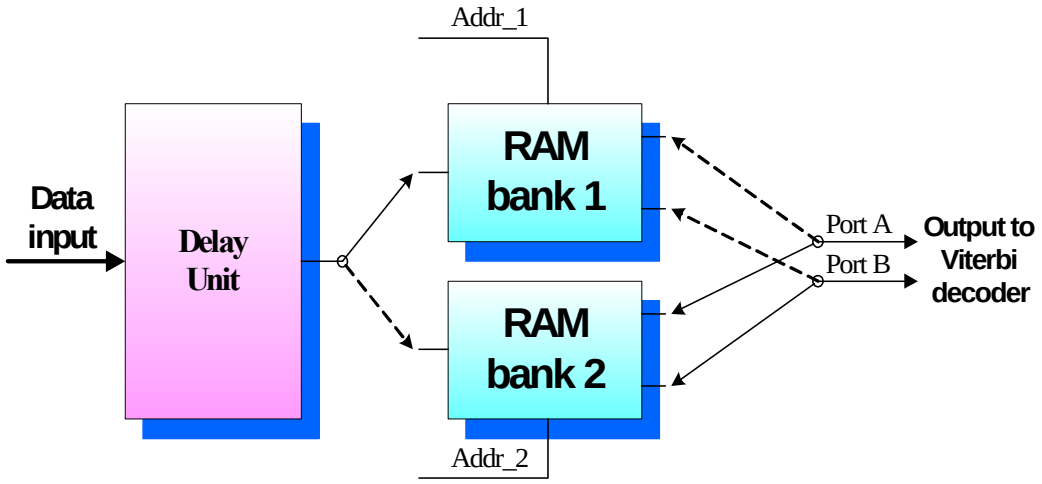


圖 4-14 反壓縮器方塊圖

4.2.6 Viterbi 解碼器

在吾人的實作當中，吾人選擇了 Viterbi 演算法當作在 802.11n 中所使用的迴旋編碼器的解碼器。一般而言，為了達到模組化及管線式處理，可以將解碼器的電路分成四個主要的處理單元，如圖 4-15 所示。其中包括:分支計量值單元 BMU(Branch Metric Unit ;BMU)、相加-比較-選擇單元(add-Compare-Select Unit ; ACSU)、路徑計量值記憶單元(Path Metric Memory Unit ; PMMU)及存活路徑記憶單元(Survivor Memory Unit ; SMU)。各單元的功能簡述如下:

I. 分支計量值單元(BMU):根據所收到的值計算在格狀圖(trellis)每一級的分支計量值。

II. 相加-比較-選擇單元(ACSU):對每個狀態執行相加-比較-選擇運算，用來計算所有狀態的存活計量值和產生決策位元(或存活路徑)

III. 路徑計量值記憶單元(PMMU):用來儲存存活計量值，以供下一級在 ACSU 中計算路徑計量值及存活計量值。

IV. 存活記憶單元(SMU):負責記錄 ACSU 所產生的決定位元，並找出存活路徑中最早時間點的位元，而當作解碼輸出。

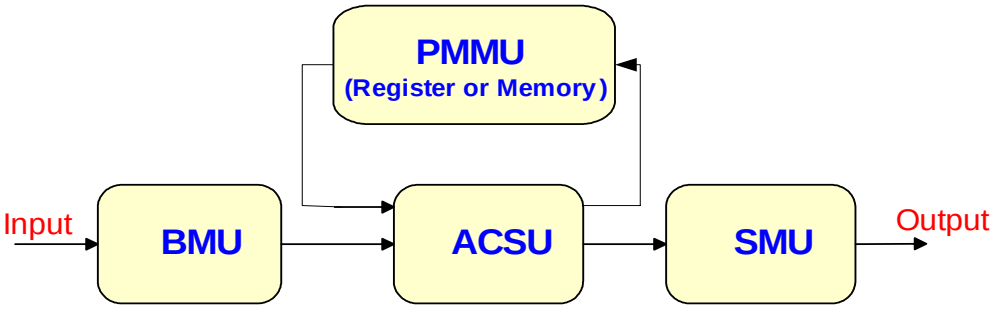


圖 4-15 Viterbi 解碼器方塊圖

在本節的安排中，由於我們在前一章已先對 Viterbi 的原理做過介紹，所以僅針對上述的四個單元一一做較詳細的介紹，並且將導入基數-4(Radix-4)之 Viterbi 解碼器架構。

4.2.6.1 分支計量值單元

分支計量值單元(BMU)，是根據所收到的值計算在格狀圖(trellis)每一級的分支計量值，也就是在計算在格狀圖上任二個節點之間的計量值。舉例說明，在圖 3-12 中，由節點 S_1 走到節點 S_2 的路徑是 01，所以假設我們收到的訊號是 $r(t)$=01，則此二節點的計量值便是 0，如果 $r(t)$=11，則此二節點的計量值便是 1。而在這個單元中的目的便是要計算出任二節點之間的計量值，而在圖 3-12 中，任二節點之間的路徑只有 00、01、10、11 四種，所以只要計算所收到的訊號跟 00、01、10、11 之間的距離，而任二個節點之間的計量值，便是這四個值的其中一個。在 VLSI 中，BMU 主要有二個實現的架構: (1)由記憶體產生，也就是說 BMU 是以查表的方式由 ROM 或 RAM 實現，而節省硬體面積。(2)由運算單元產生，將接收訊號直接運算出在格狀圖中所需的計量值，而這種方法的硬體複雜度則根據計量值的定義。在吾人的實作中是以絕對值，做為計算計量值的方法，且採用(2)由運算單元產生計量值的架構。圖 4-16 便是吾人使用的 BMU 架構。

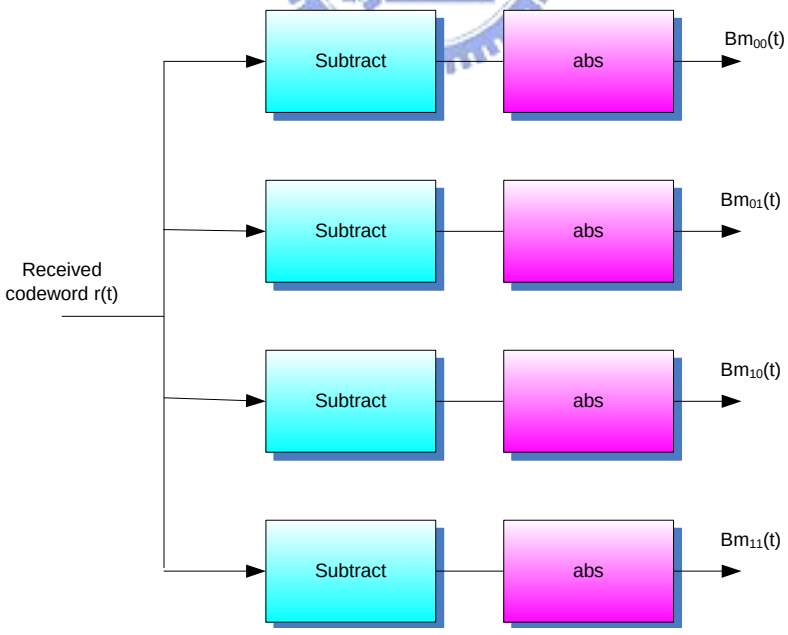


圖 4-16 BMG 方塊圖

4.2.6.2 相加-比較-選擇單元(ACSU)

ACSU 是整個 Viterbi 解碼器的核心，在高速傳輸的應用中(如 WLAN)，平行架構的 ACSU 則是必要的，因此在格狀圖的每一節點，便代表著一個 ACS 處理單元。

圖 4-17 為 ACSU 架構的例子，這是在 3.2 節中所使用(2,1,2)的架構。這 ACS 處理單元有四個輸入(2 個分支計量值及 2 個存活計量值)和 2 個輸出(下一級的存活計量值及決定位元)。我們先由格狀圖畫分出有幾個蝴蝶模組，且決定各蝴蝶模組的輸入輸出關係，而一個蝴蝶模組是由二個 ACS 所組成。在 ACS 中，其主要功能有二:

1. 在路徑計量值中選擇最小的當作存活路徑值，並儲存之。
2. 產生存活路徑之決定位元，並送至存活路徑記憶單元(SMU)。

決定位元是在 ACS 中所產生最重要的資訊。它暗示了下一級的存活計量值是由哪一組存活計量值及分支計量值之和所產生的。假設決定位元為 0，代表著在 ACS 中上面那條是被選擇為下一級的存活路徑。反之決定位元為 1 的話，存活路徑便是挑下面那條。代表著所有節點的 ACS 所產生的決定位元，將會被送到 SMU 中做解碼而產生出最後的輸出。

然而在吾人的實作中，是使用 6 個移位暫存器的編碼器，這代表了格狀圖會有 64 個狀態節點，便有 32 個蝴蝶模組、64 個 ACS，蝴蝶模組的目的為指示 ACS 應該抓取怎樣的路徑計量值及分支計量值，其架構如圖 4-18 所示。

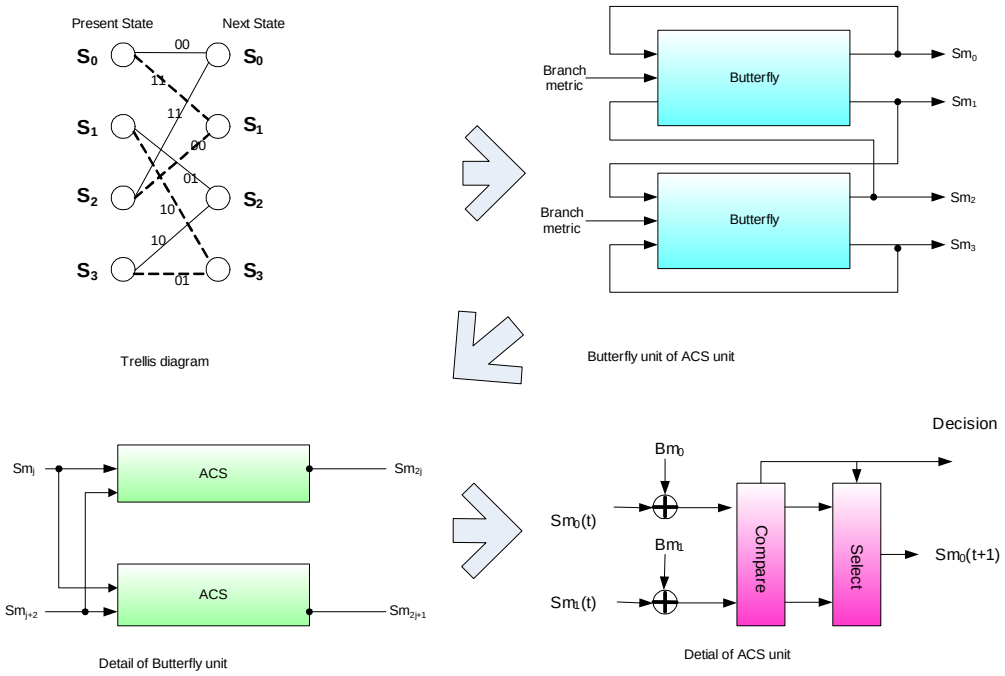


圖 4-17 ACS 設計流程圖

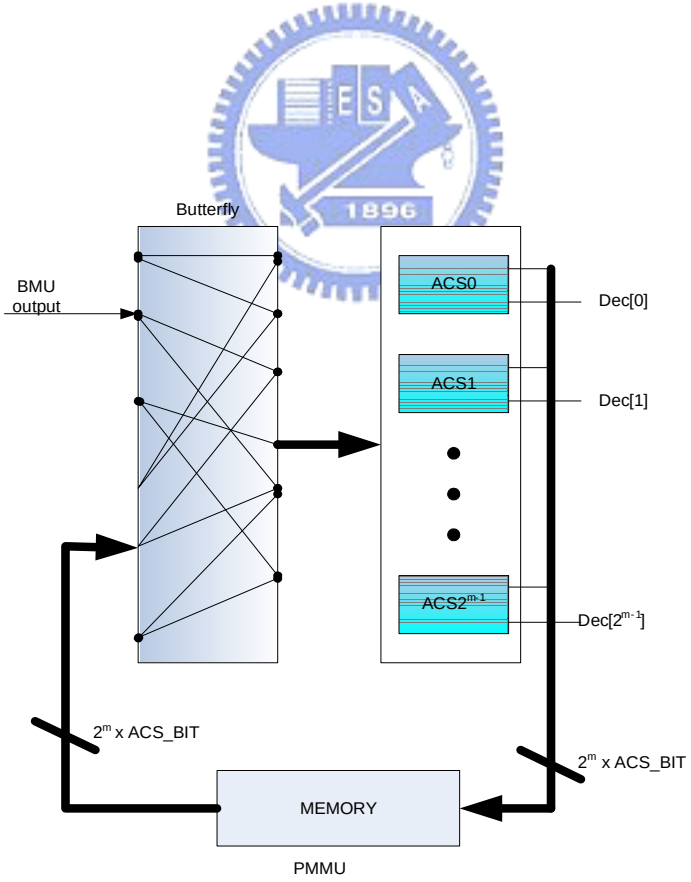


圖 4-18 平行 ACS 架構方塊圖

然而當資料不斷的進來，因為存活計量值會一直累加，為避免溢位(overflow)所造成的錯誤，我們必須對存活計量值做正規化，而正規化的方法有很多，最直覺的想法是將所有狀態之存活計量值同減去一個固定的值，但這造成了額外的硬體面積，於是在吾人的實作中採用[24]的修改過的模正規化(Modified Modulo Normalization)，其架構如圖 4-19 所示。

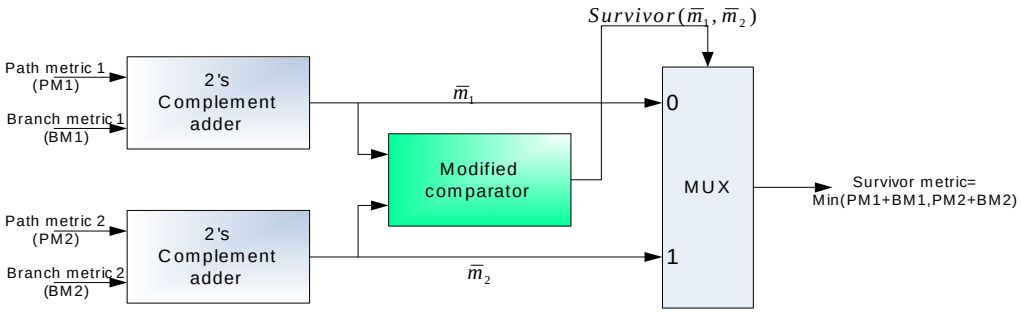


圖 4-19 模正規化架構方塊圖

在模(modulo)的技術中，所有的計量值(metric) m_j 都要依(4-19)被轉換為正規化過的計量值 $\bar{m}_j$ 。

$$\bar{m}_j = (m_j + \frac{C}{2})(\text{mod } C) - \frac{C}{2} \tag{4-19}$$

C 表示由路徑計量值所能表示的最大值。而比較法則決定了要給 SMU 的存活位元的值。這法則表示如下:

$$Survivor(\bar{m}_1, \bar{m}_2) = \bar{m}_{1p} \oplus \bar{m}_{2p} \oplus y(\bar{m}_1, \bar{m}_2) \tag{4-20}$$

$y(\bar{m}_1, \bar{m}_2)$ 表示 $\bar{m}_1$ 及 $\bar{m}_2$ 的非負(unsigned)比較。$\bar{m}_{1p}$ 及 $\bar{m}_{2p}$ 分別表示 $\bar{m}_1$ 及 $\bar{m}_2$ 的最高位元。也就是說當 $\bar{m}_1$ 及 $\bar{m}_2$ 有同樣的正負號時， $Survivor(\bar{m}_1, \bar{m}_2) = y(\bar{m}_1, \bar{m}_2)$ ，其單位為 1 位元。而當 $m_1 \le m_2$ 時， $Survivor(\bar{m}_1, \bar{m}_2) = 0$ 。

4.2.6.3 存活記憶單元

存活記憶單元(SMU)的目的最主要就是記錄從 ACS 中送來的決定位元或存活路徑。再根據這些資料進行解碼。這是在 Viterbi 解碼器中，最具有討論設計空間的地方。現存有二種著名的 SMU 做法，分別是暫存器交換方法[25](register-exchange method ; REM)及後向追溯方法[26]-[28](trace back method ; TBM)。暫存器交換方法有著架構簡單、快速的優點，但使用的暫存器所增加的面積卻是在設計中最需考量的地方。然而在另一方面後向追溯方法使用記憶體來儲存由 ACS 送來的存活路徑，而使得所須的面積較暫存器交換方法小，但有著較長的延遲及較複雜的控制電路，且所需要的功率(power)也較大。在吾人的實作中，綜合考量高系統時脈(clock)及所需面積大小，吾人採用在[28]中所提出的以記憶體實作 TBM 的其中一種，稱為“*k* pointer even”。底下則介紹這種方法。

首先我們先介紹在 TBM 中一定會被定義的三種動作:

1. 寫入(WR):從 ACSU 中所送過來的存活路徑將會被會寫入與狀態互相對應的記憶體中，比如在吾人的實作中，ACSU 共有 64 個 ACS，代表著 64 狀態，相對的也有 64 組存活路徑產生而需要被記錄。
2. 回溯讀取(Trace-back Read ; TB):TB 這動作主要是將讀取寫在記憶體中的存活路徑，將讀出的值當作一個指標，而這指標指向上個狀態，而能找到上一個狀態數。這動作的目的為希望能讓所有路徑重合。
3. 解碼讀取(Decode Read ; DC):這動作和 TB 相同，同樣讀出一個指標而找到上個狀態。但是這個動作的對象是更舊的資料，而且是從已經作過完整的 TB 之後所找到的狀態數開始做 DC 這動作。在 DC 中，讀出的指標值除了指向上一個狀態數之外，在搭配上現在的狀態數，便可以得到當初輸入而解碼輸出了。

k=3 的 k point even 方法如圖 4-20 所示。在這方法中使用了 *k* 個讀取指標(包括回溯讀取指標及解碼讀取指標)，由 2*k* 塊大小為 M/2 的記憶體所組成，*M* 是指

截斷長度。寫入的指標是從左到右的，而讀取指標是由右而左的。在一個時間點中，六塊記憶體中有一個是執行寫入、一個解碼讀取、二個回溯讀取及二個閒置。第一個位元輸出是在當寫入指標寫到整個記憶體的最後時，所以全部的延遲為 $3M$。然而在因為解碼讀取是由右而左的，所以位元是以一個倒反的順序輸出的，這需要另一塊記憶體當作 LIFO(Last In First Out)而讓輸出順序正確。

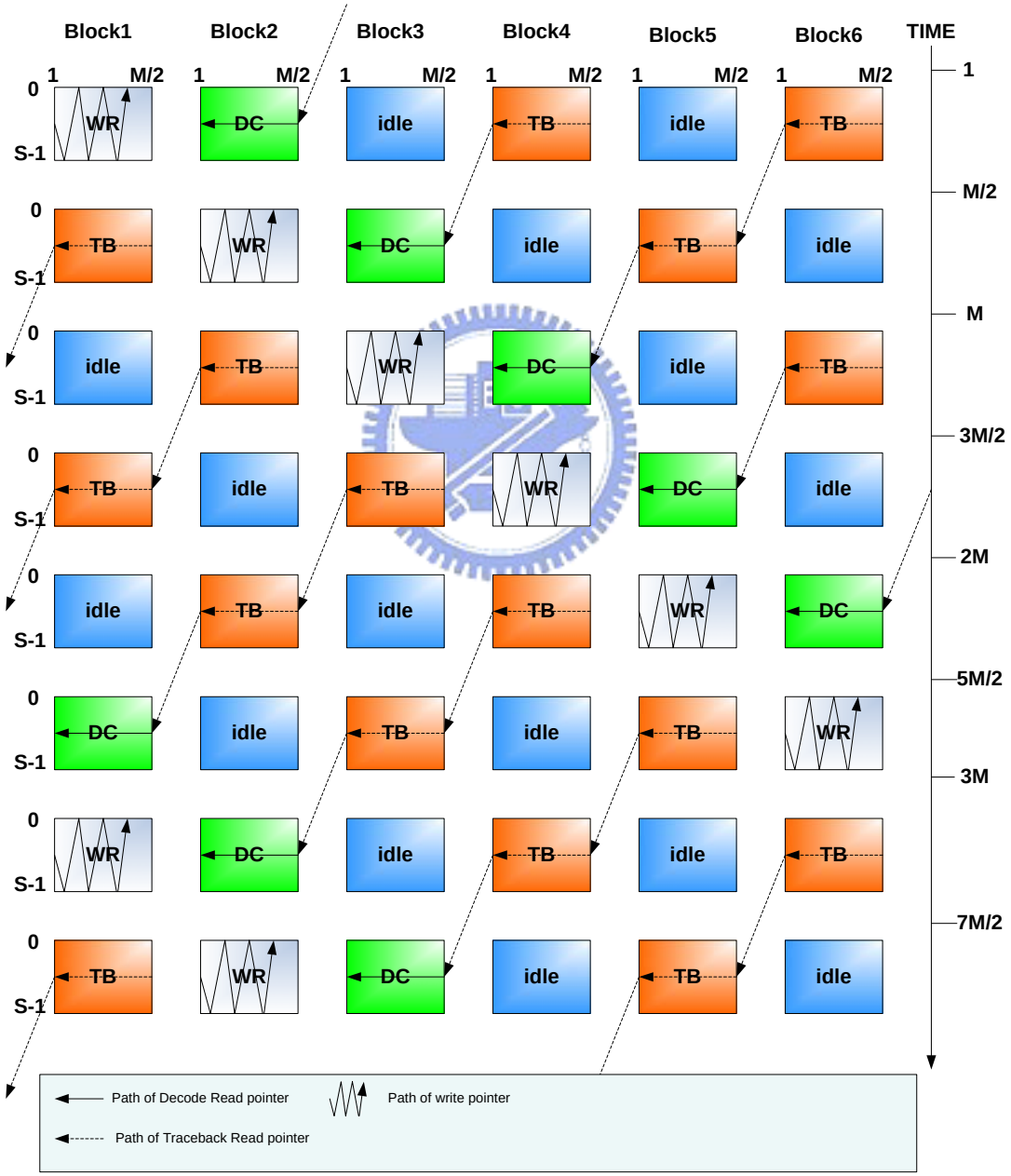


圖 4-20 TBM k point even 流程圖

4.2.6.4 Radix-4 之 Viterbi 解碼器架構

針對 Viterbi 解碼器，吾人使用 Radix-4 ACSU 搭配 Radix-4 之 BMU 及 Radix-4 之 Pre-SMU 和 Radix-16 之 SMU，以達到降低時脈速度或提昇資料的傳輸速度之目的。而針對位元階層(bit level)，吾人欲使用 8 level，也就是 3 位元的輸入位元[19]，來實現 Viterbi 解碼器，其模擬將在 4.3 節中介紹，至於在 bit level，如何打斷關鍵路徑(critical path)來加快 Viterbi 解碼器的資料傳輸速度，在[20]-[21]有較詳細的介紹，由於此部分牽涉到加法器的架構變動，吾人在此並不多做探討。

如何將前述的 Viterbi 解碼器架構轉換成 Radix-4 之 Viterbi 解碼器架構，其實就是將原本兩級的路徑合而為一級的路徑，一個 8 個狀態(eight-state)的 Radix-2 trellis to equivalent Radix-4 trellis 的例子，如圖 4-21 所示。至於 trellis Radix 的大小對資料的傳輸速度及計算複雜度的影響，如表 4-2 所示，由表 4-2 中，吾人可發現 Radix-4 之 Viterbi 解碼器架構剛好位於計算複雜度對 trellis Radix 所造成的指數增加曲線(exponentially increasing curve)以及傳輸速度對 trellis Radix 所造成的理想線性增加曲線(ideal linear increasing curve)的交點，所以吾人可使用 Radix-4 之 Viterbi 解碼器架構當做 Viterbi 解碼器之硬體實現架構，再者，Radix-4 之 Viterbi 解碼器架構是唯一增加 throughput 而維持相同面積效率(area efficiency)的架構。

Radix-4 之 BMU 即是將原本輸入訊號做一級延遲，用此延遲訊號輸入 Radix-2 之 BMU 所造成的 4 組輸出與原本輸入訊號輸入 Radix-2 之 BMU 所造成的 4 組輸出做一組合，可得到 4×4＝16 組分支計量值，Radix-4 之 BMU 方塊圖如圖 4-22 所示。將圖 4-21 中的 Radix-4 trellis 一般化，吾人可得知每個輸出狀態都有 4 個前一級來的狀態，進而可知路徑計量值在 Radix-4 ACSU 裡，能用 4 路(four-way)的 ACS 來更新，而每一組決定位元乃由 2 個位元所組成，Radix-4 ACS 設計流程圖如圖 4-23 所示。而為了使 trace back 時都能回到狀態 0，吾人將所有為零的輸入(all zero inputs)送至 Radix-4 之 Viterbi 解碼器，可得 64 個穩態

(steady-state)路徑計量值，並將此 64 個穩態路徑計量值當作初始狀態(initial state)路徑計量值，64-state 初始狀態路徑計量值如表 4-3 所示。Radix-4 之 Pre-SMU 即是將兩級的決定位元，合而為一個由 4 個位元組成的決定位元，其設計流程圖如圖 4-24 所示。而針對 Radix-4 之 Pre-SMU，吾人須搭配 Radix-16 之 SMU，如此可以將 SMU 的 clock rate 做進一步的下降，Radix-16 之 SMU 方塊圖如圖 4-25 所示。Radix-4 之 Viterbi 解碼器方塊圖如圖 4-26 所示。

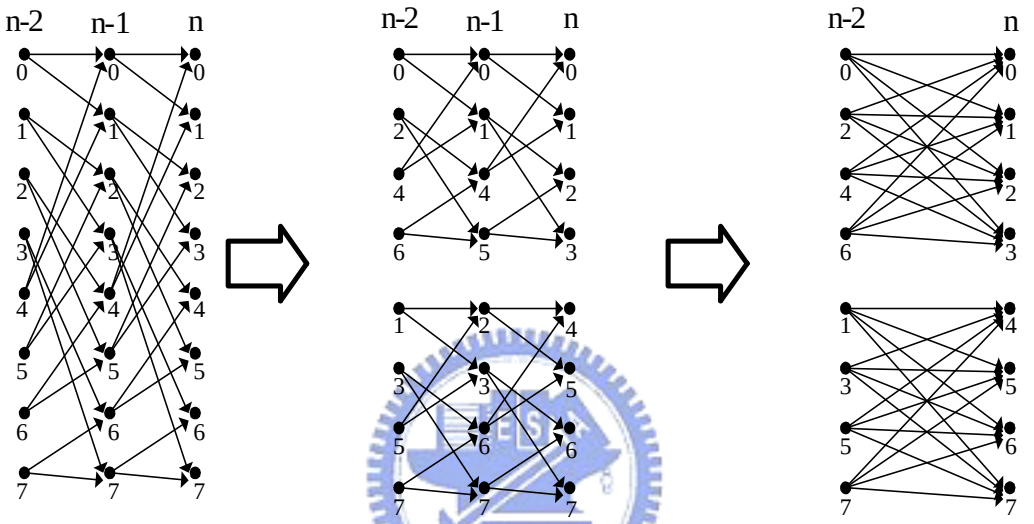
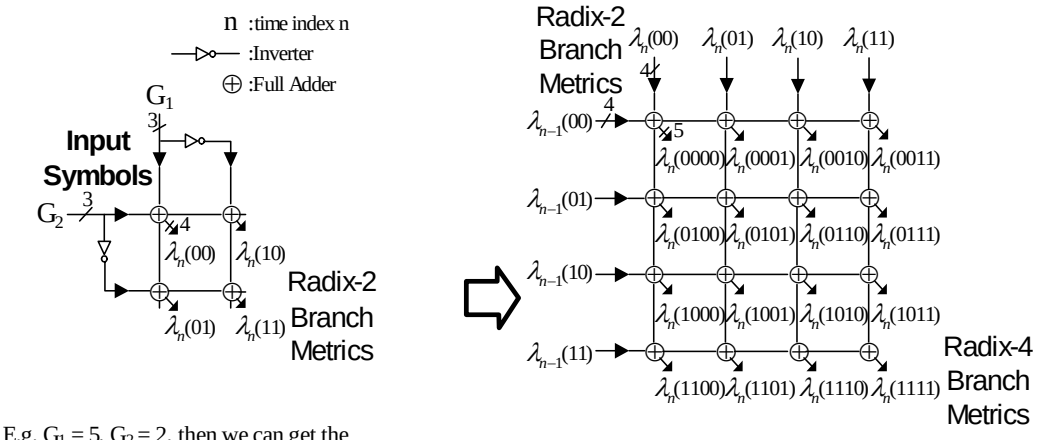


圖 4-21 Radix-2 trellis to equivalent Radix-4 trellis 示意圖

Radix-2^k COMPLEXITY SPEED MEASURES				
Radix	k(No. of path combination)	Ideal Speedup	Complexity Increase	Area Efficiency
2	1	1	1	1
4	2	2	2	1
8	3	3	4	0.75
16	4	4	8	0.5

表 4-2 trellis Radix 的大小對資料的傳輸速度及計算複雜度的影響比較表



E.g. $G_1 = 5$, $G_2 = 2$, then we can get the branch metric $\lambda_n(11) = |5 - 7| + |2 - 7| = 2 + 5 = 7$

Using Radix-2 branch metric calculation, we can get the same result. The calculation is as follow.

$\lambda_n(11)$ = Inverter($(5)_{10}$) + Inverter($(2)_{10}$)
=Inverter($(101)_2$) + Inverter($(010)_2$)
=$(010)_2$ + $(101)_2$
=$(2)_{10}$ + $(5)_{10}$ = $(7)_{10}$

圖 4-22 Radix-4 之 BMU 方塊圖

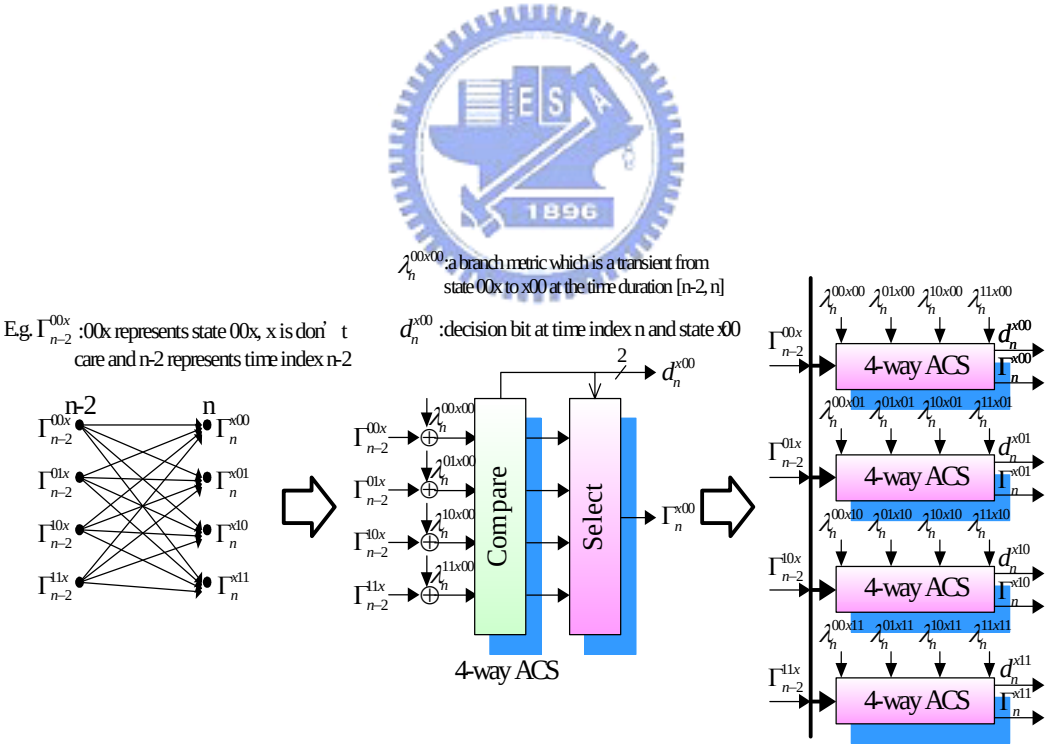


圖 4-23 Radix-4 ACS 設計流程圖

64-STATE ACS INITIALIZATION								
State	0	1	2	3	4	5	6	7
Init. Metric	0	14	21	21	35	21	28	28
State	8	9	10	11	12	13	14	15
Init. Metric	49	35	28	28	28	42	35	35
State	16	17	18	19	20	21	22	23
Init. Metric	49	35	42	42	28	28	35	35
State	24	25	26	27	28	29	30	31
Init. Metric	42	28	49	49	35	49	42	42
State	32	33	34	35	36	37	38	39
Init. Metric	56	56	49	35	49	49	42	28
State	40	41	42	43	44	45	46	47
Init. Metric	35	35	28	42	42	42	49	35
State	48	49	50	51	52	53	54	55
Init. Metric	49	49	42	28	56	56	49	49
State	56	57	58	59	60	61	62	63
Init. Metric	42	42	49	49	49	49	42	42

表 4-3 64-state 初始狀態路徑計量值

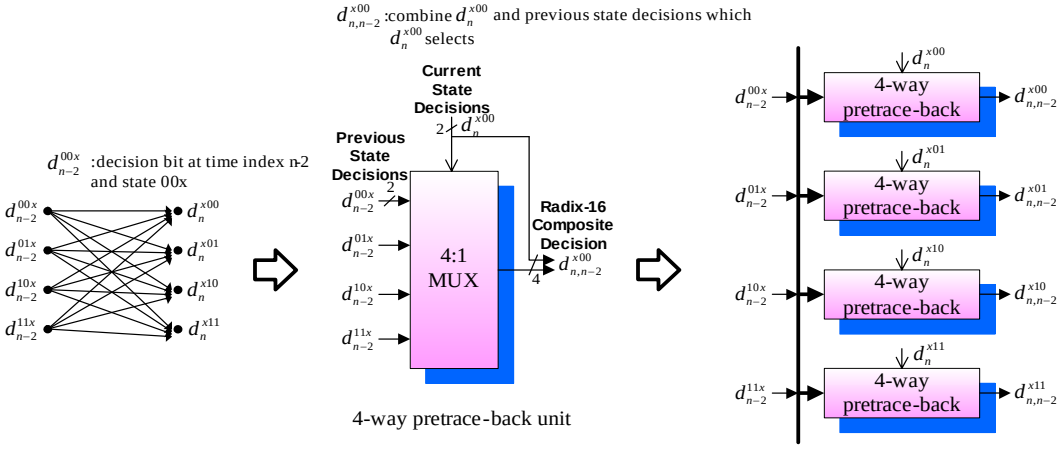


圖 4-24 Radix-4 之 Pre-SMU 設計流程圖

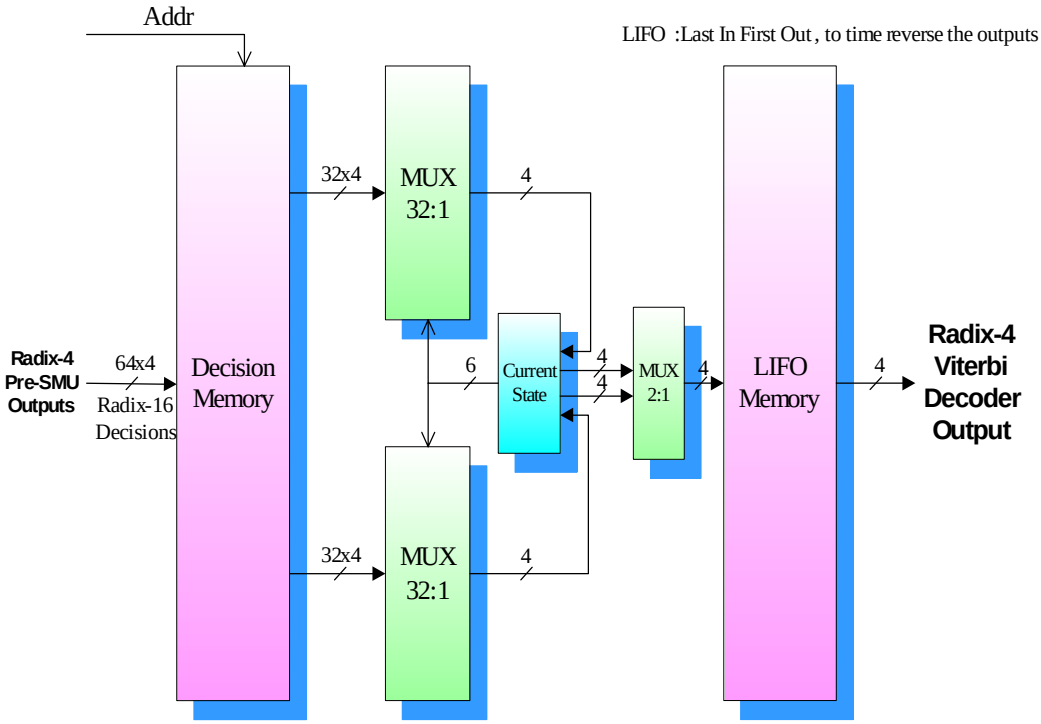


圖 4-25 Radix-16 之 SMU 方塊圖

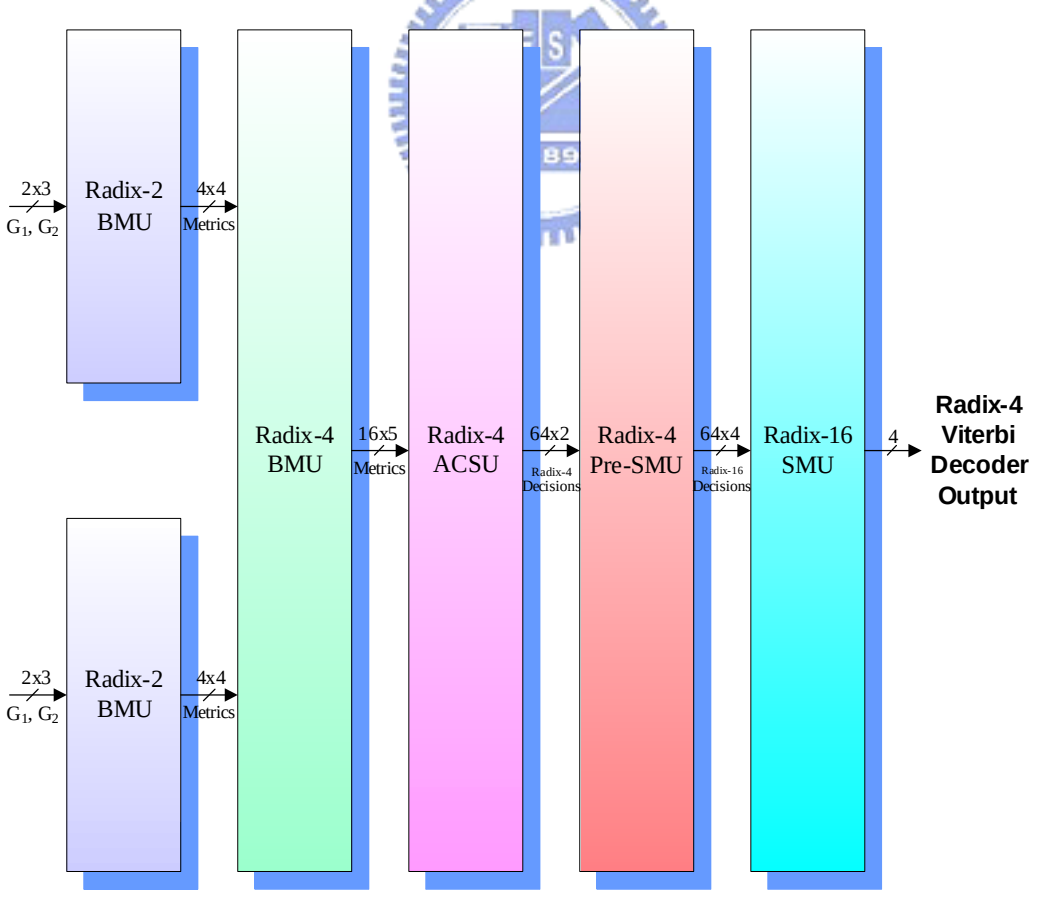


圖 4-26 Radix-4 之 Viterbi 解碼器方塊圖

4.3 定點數(Fixed-point)模擬結果

在本節中我們將使用 MATLAB 做定點模擬，主要是使浮點模擬轉換成定點模擬能有可接受的效能差，並期能用最少的位元數來完成硬體的實現，也就是達到節省面積，降低成本的目的。圖 4-27 中為採用 SISO、BPSK 以及純粹 Viterbi 解碼器的系統，而 floating 及 Q8 分別代表沒有經過量化及經過 8 階、3 位元表示的量化器，還有 CL7 表示 constrain length 為 7。圖 4-28 至圖 4-31 中為針對 ZF 偵測器，以各種不同的位元數表示之浮點數模擬對定點數模擬之比較圖，其中系統為 2×2、64QAM 以及編碼率為 1/2，而 floating point 代表浮點數模擬， fixed point-1 為使用 11 位元來表示通道係數以及進入 ZF 偵測器之位元數，而 CSI 的部分則直接採用定點運算過後之結果， fixed point-2 為使用 11 位元來表示通道係數以及進入 ZF 偵測器之位元數，而 CSI 的輸出則使用 13 位元表示， fixed point-3 則是使用 7 位元來表示通道係數，而用 11 位元及 13 位元來分別表示進入 ZF 偵測器之位元數及 CSI 的輸出，至於 fixed point-4 是使用 7 位元來表示進入 ZF 偵測器之位元數，而用 11 位元及 13 位元來分別表示通道係數及 CSI 的輸出，還有 fixed point-5 則是使用 9 位元來表示通道係數，而用 11 位元及 13 位元來分別表示進入 ZF 偵測器之位元數及 CSI 的輸出。由圖 4-27 模擬發現，使用 8 階、3 位元的量化與不使用量化器的效能上是非常相近的，故可在 Viterbi 解碼器前，加上 8 階、3 位元的量化器，以減少輸入端之位元數。由圖 4-28 模擬發現， fixed point-1 與 fixed point-2 對 floating point 都造成約 1dB 的效能影響。由圖 4-29 至圖 4-30 模擬發現，通道係數定點位元數減少對效能影響較小，而進入 ZF 偵測器之位元數減少則對效能影響較大。由圖 4-31 模擬發現， fixed point-5 與 fixed point-2 在效能上有非常相近的結果。故吾人決定使用 9 位元來表示通道係數，而用 11 位元及 13 位元來分別表示進入 ZF 偵測器之位元數及 CSI。

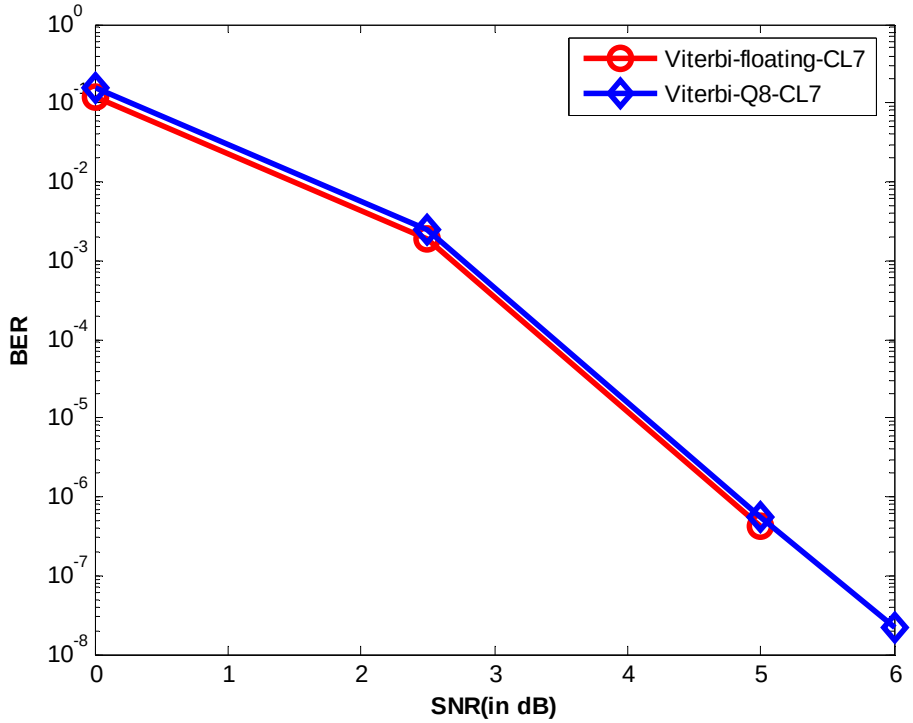


圖 4-27 純粹 Viterbi 解碼器浮點模擬對定點模擬之比較圖

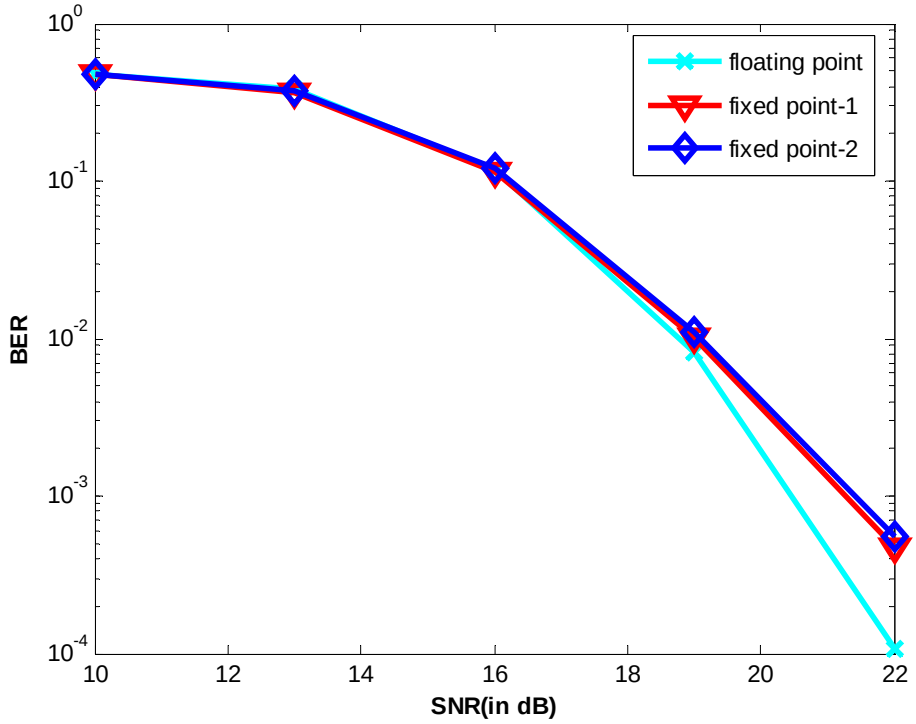


圖 4-28 接收機浮點模擬對定點模擬之比較圖 –(1)

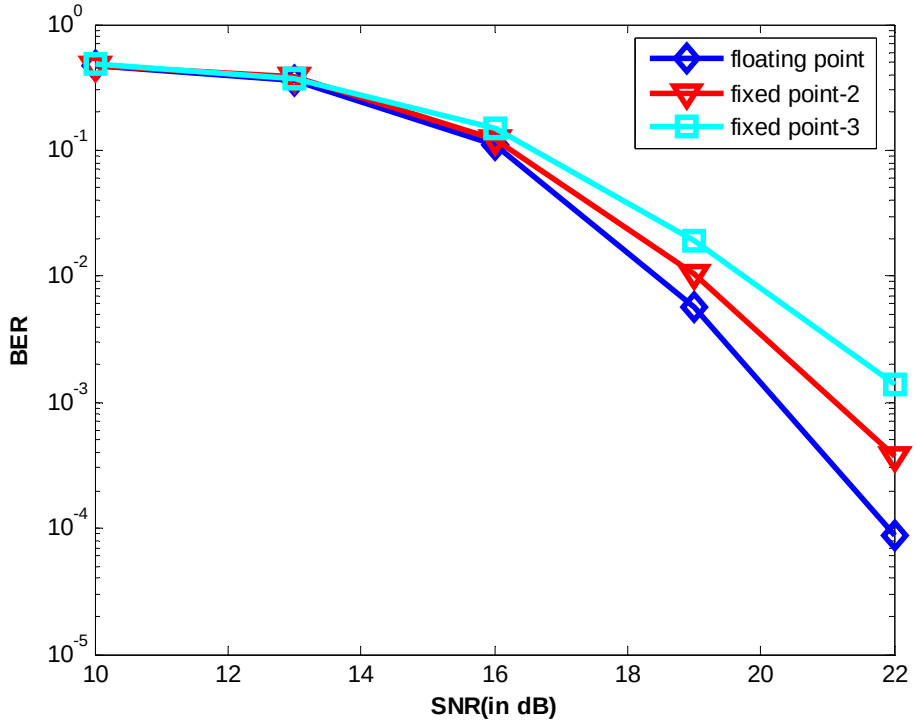


圖 4-29 接收機浮點模擬對定點模擬之比較圖 –(2)

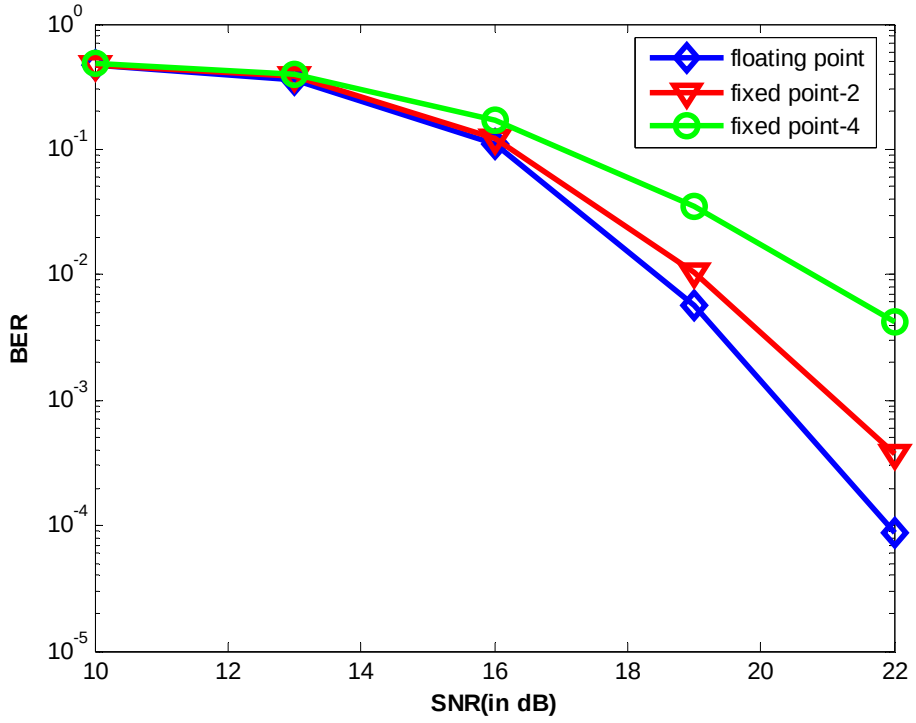


圖 4-30 接收機浮點模擬對定點模擬之比較圖 –(3)

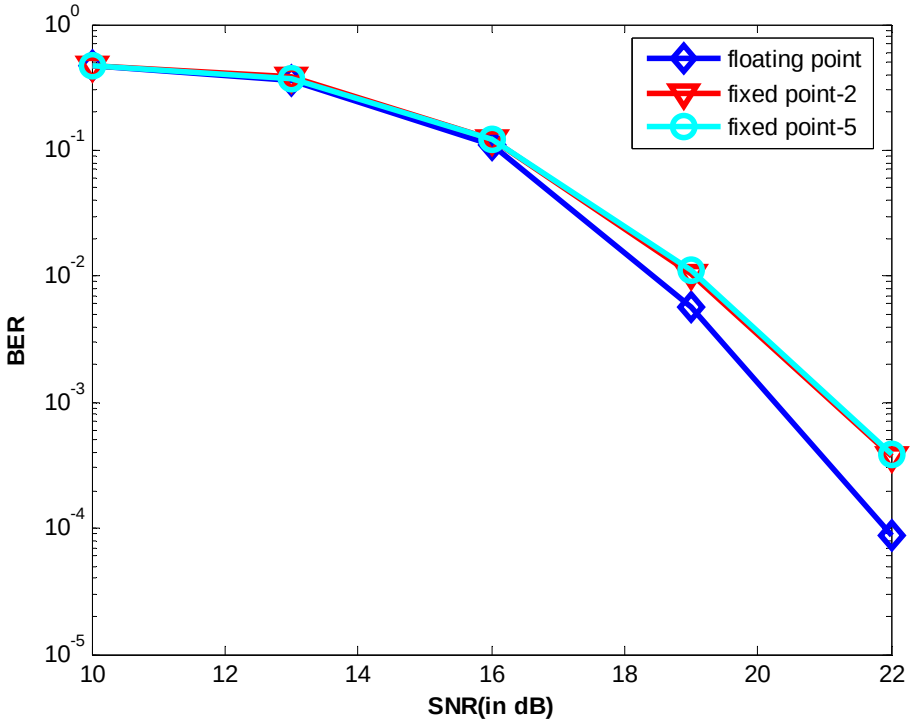


圖 4-31 接收機浮點模擬對定點模擬之比較圖 –(4)

4.4 硬體模擬結果

在本節中我們依照 4.1 所描述的設計流程將所設計之接收機用 Verilog 語言寫成行為模式(behavioral model)的硬體架構，流程主要有以下幾點:

1. 使用 MATLAB 做定點模擬，在和浮點模擬有可接受的效能衰減下，決定各個單元中變數的位元數，而定點模擬和浮點模擬之比較如上一節中所示。
2. 以 Modelsim 軟體模擬驗證此行為模式是否正確,再和步驟 1 中的 MATLAB 定點模擬比較來驗證。
3. 行為模式正確後就可將此 Verilog 程式輸入 Xilinx 的 ISE 6 軟體就可得到合成後的 RTL 巢狀(netlist)架構檔，此是根據所撰寫行為模式下 Verilog 程式所做合成後的輸出檔案,而所設計之接收機之 RTL 架構圖如圖 4-32 至圖 4-34 所示。

然而吾人在硬體實現中並沒有做 FPGA download 後之驗証，僅針對其進行硬體模擬驗證。最後由 ISE 6 所產生的 Mapping report 及 Timing report 如圖 4-35 及圖 4-36 所示。

最後，吾人將部分硬體模擬圖放上，用以說明硬體模擬驗證的結果，圖 4-37 至圖 4-39 為 Viterbi 解碼器的硬體模擬圖，其中 clk 為解碼器之輸入訊號所用的時脈，rst 為解碼器外部的重置(Reset)訊號，encoder_en 為 1 時，表示整個解碼器開始輸入，mes_i 為輸入 encoder 的訊號，其來源為 Matlab 輸出之 stimulus，g_1_i、g_2_i 為 mes_i 經由 Matlab 完成之無雜訊系統,至反壓縮器後的輸出,來當 Viterbi 解碼器硬體的輸入，start_of_packet_o 為解碼器內部的 Reset 訊號，decoder_o 為解碼器的輸出，message_shift_reg[121]為在 Test bench 上將 mes_i 做 122 級的延遲，而 err_ind 為在 Test bench 上做一機制來顯示 decoder_o 和 message_shift_reg[121]是否相同。圖 4-40 至圖 4-41 為反交錯器的硬體模擬圖。圖 4-42 至圖 4-43 為反空間分離器至反壓縮器的硬體模擬圖,其中 n_ss_para_i=1

表示資料流為 2，s_para_i=2 表示 $s=\max(\frac{N_{BPSC}}{2},1)=3$，rate_para_i=3 表示編碼率 $R=5/6$，input_packet0_i、input_packet1_i 表示反空間分離器的輸入訊號，output_stream_o_conv 表示反空間分離器的輸出訊號，也就是反壓縮器的輸入訊號，output_stream_u_o_depun、output_stream_d_o_depun 表示反壓縮器的輸出訊號。從圖 4-37 至圖 4-38 硬體模擬結果發現，decoder_o 在剛開始並不會輸出，而在 122 級的延遲後才會輸出，這是因為 Viterbi 解碼器硬體實現上，我們使用截斷長度 $M=40$，而由 4.2.6.3 可知，全部的延遲為 $3M=120$，再加上 RAM 輸出時造成的延遲，以及 decoder_o 輸出時多加的一級延遲，總共是 122 級延遲，而 decoder_o 會和 message_shift_reg[121]相同，這是因為 Matlab 所使用的是無雜訊系統。從圖 4-39 硬體模擬結果發現，start_of_packet_o 會在 decoder_o 輸出結束時，也就是輸入延遲 122 級後，由 1 變為 0，這是為了將 Viterbi 解碼器內部的暫存器(Registers)在沒有使用時，全部 Reset 為 0。從圖 4-40 硬體模擬結果發現，反交錯器每個資料流都使用 2 塊 RAM 的架構，2 塊 RAM 的讀寫控制都是交替產生的，如圖，當 wen_sel_u(表示寫上面的 RAM)由 1 變 0 時，rd_sel_u(表示讀上面的 RAM)才會由 0 變 1，這樣可以確保每塊 RAM 都是寫完再讀且同一時間只做讀或寫，另外 packet_start_o(表示開始輸出)在 rd_sel_u 由 0 變 1 起才會由 0 變 1，以確保輸出的為寫入以後之資料。從圖 4-41 硬體模擬結果發現，n_ss_para_i 為 1 時，表示資料流為 2，onto_no_i 為 $(11)_2$ 時，表示 64QAM，由 4.2.4 節可知，$N_{CBPS}=52\times N_{BPSC}=312$，故 wen_sel_u 下的 addr_ram_1_u(表示資料流 1 的上面那塊 RAM 的位址)，其範圍為 0.....311，而 rd_sel_u 下的 addr_ram_1_u 為 ROM 中所放的索引值，最後將 output_stream_1_o、output_stream_2_o(資料流 1、2 的輸出)與 Matlab 中的模擬結果比較即可驗證。從圖 4-42 硬體模擬結果發現，反空間分離器硬體模擬結果與 4.2.4 節之圖 4-11 中 $s=\max(\frac{N_{BPSC}}{2},1)=3$ 的情形相吻合。從圖 4-43 硬體模擬結果發現，反壓縮器硬體模擬結果與 4.2.5 節之圖 4-12 中編碼率 $R=5/6$ 的情形亦有相同的結果。

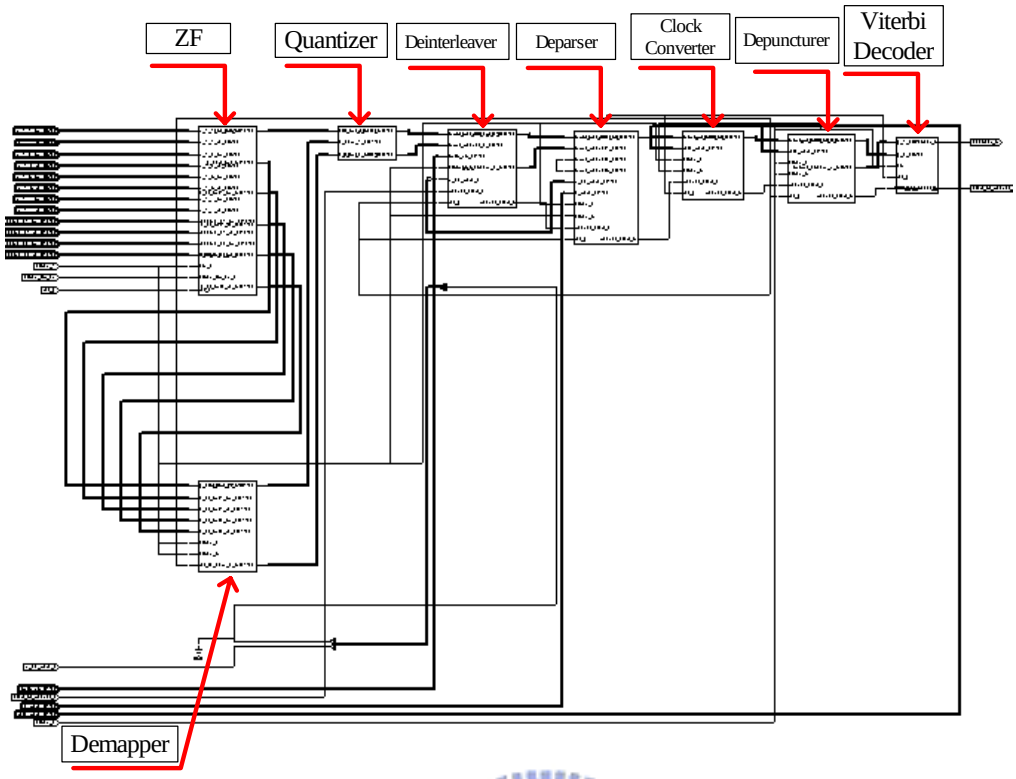


圖 4-32 接收機之 RTL 架構圖

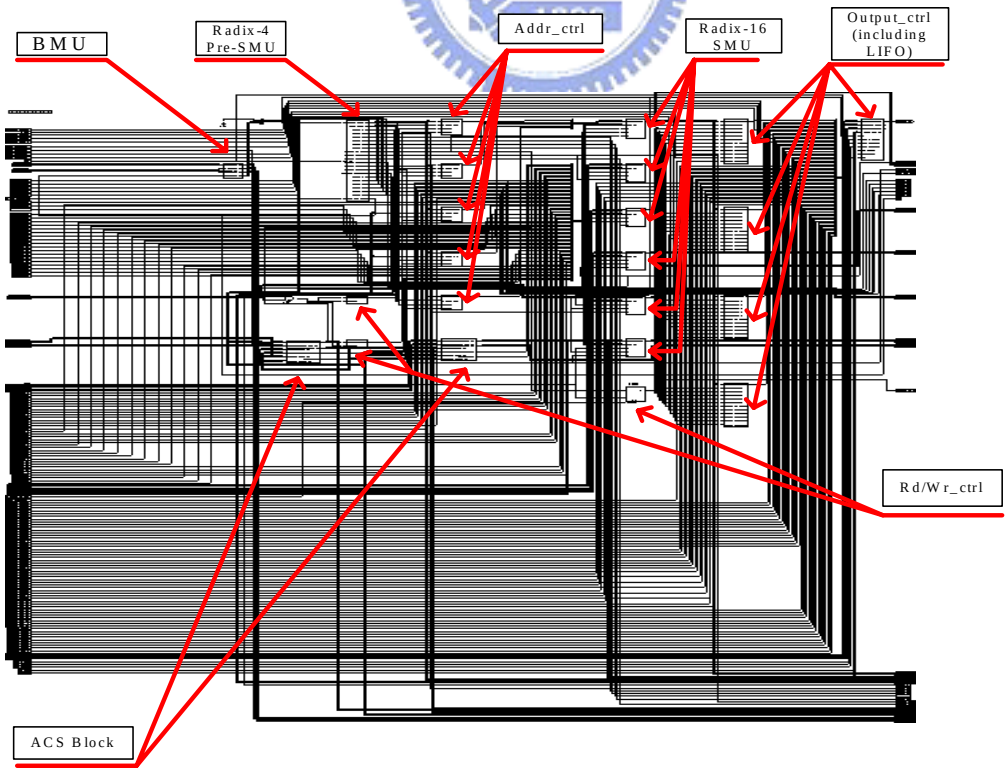


圖 4-33 Viterbi 解碼器之 RTL 架構概圖 – (1)

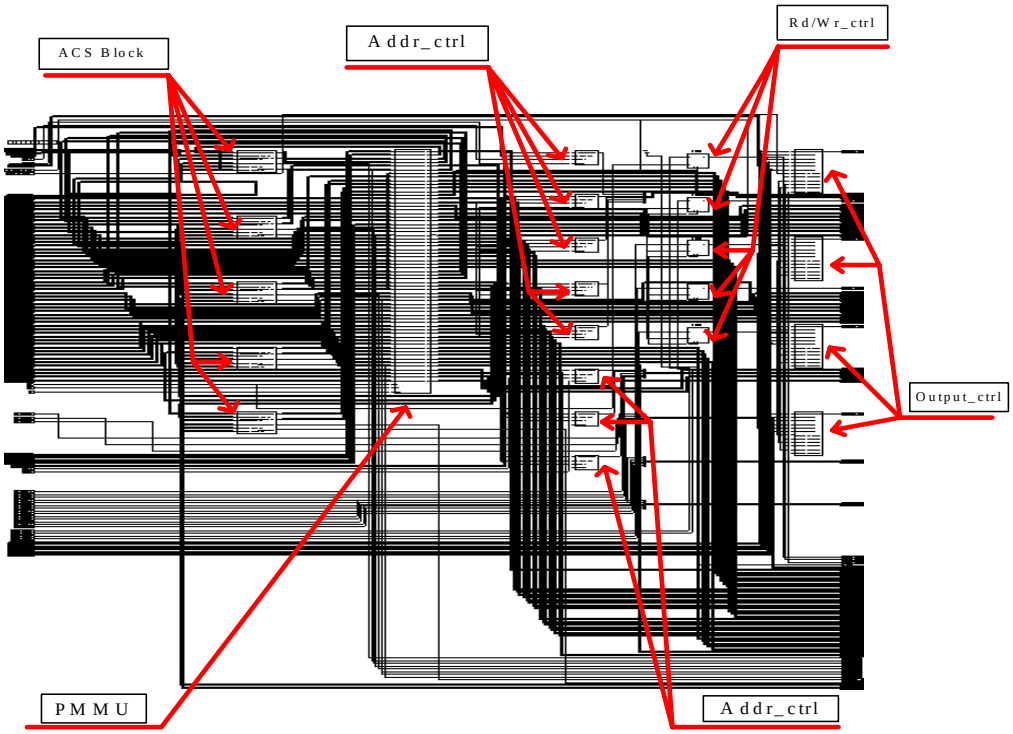


圖 4-34 Viterbi 解碼器之 RTL 架構概圖 – (2)

```
Design Summary
--------------
Number of errors:     0
Number of warnings:   0
Logic Utilization:
  Number of Slice Flip Flops:    1,921 out of 38,400    5%
  Number of 4 input LUTs:        4,550 out of 38,400   11%
Logic Distribution:
    Number of occupied Slices:                  2,925 out of 19,200   15%
    Number of Slices containing only related logic:  2,925 out of 2,925  100%
    Number of Slices containing unrelated logic:      0 out of 2,925    0%
       *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:       5,391 out of 38,400   14%
     Number used as logic:              4,550
     Number used as a route-thru:         555
     Number used for 32x1 RAMs:           240
     (Two LUTs used per 32x1 RAM)
     Number used as 16x1 RAMs:             46
   Number of bonded IOBs:         128 out of   404   31%
     IOB Flip Flops:                      2
   Number of Block RAMs:           12 out of   160    7%
   Number of GCLKs:                2 out of     4   50%
   Number of GCLKIOBs:             2 out of     4   50%

Total equivalent gate count for design:  407,173
Additional JTAG gate count for IOBs:  6,240
Peak Memory Usage:  160 MB
```

圖 4-35 接收機之 Mapping report

Timing Summary:

Speed Grade: -8

Minimum period: 9.038ns (Maximum Frequency: 110.644MHz)
Minimum input arrival time before clock : 9.019ns
Maximum output required time after clock : 7.901ns
Maximum combinational path delay : No path found

圖 4-36 接收機之 Timing report

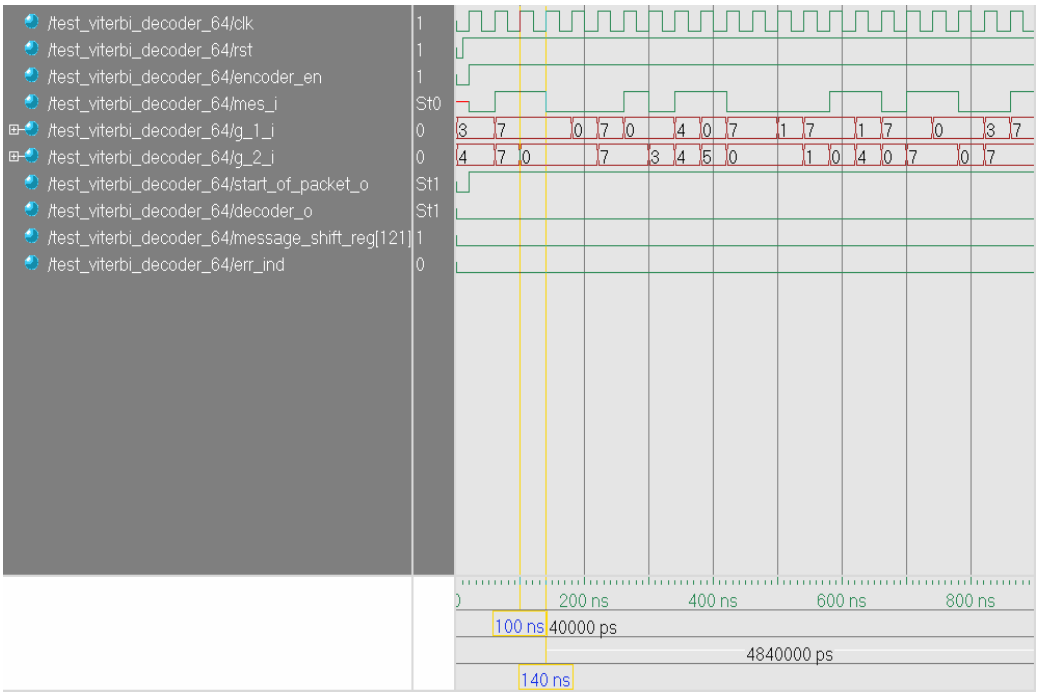


圖 4-37 Viterbi 解碼器硬體模擬圖 – (1)

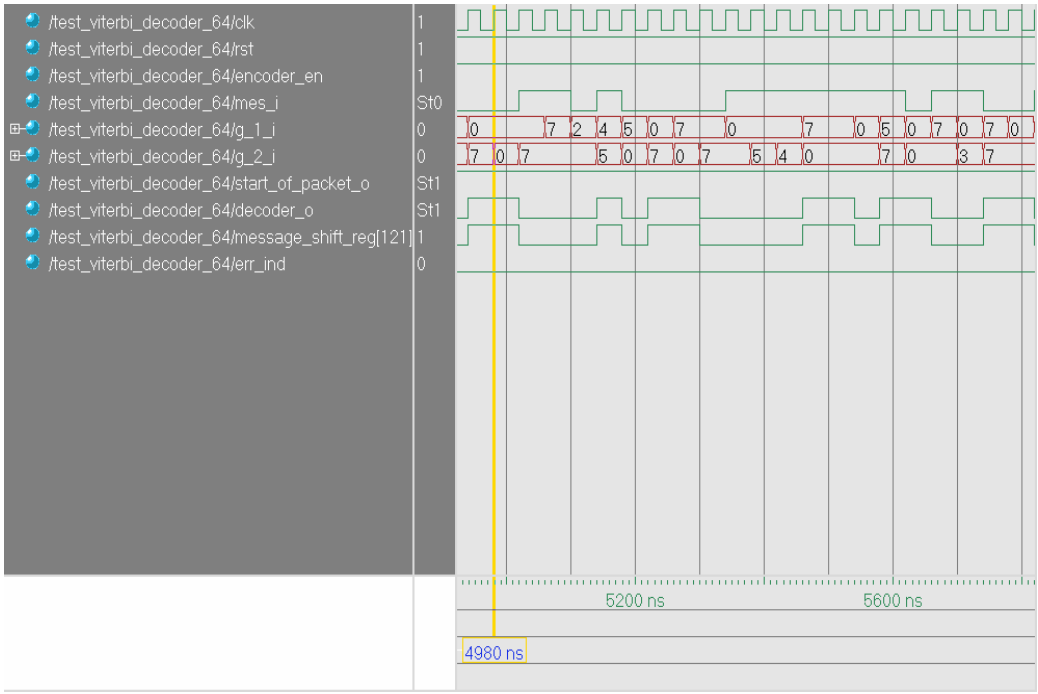


圖 4-38 Viterbi 解碼器硬體模擬圖 – (2)

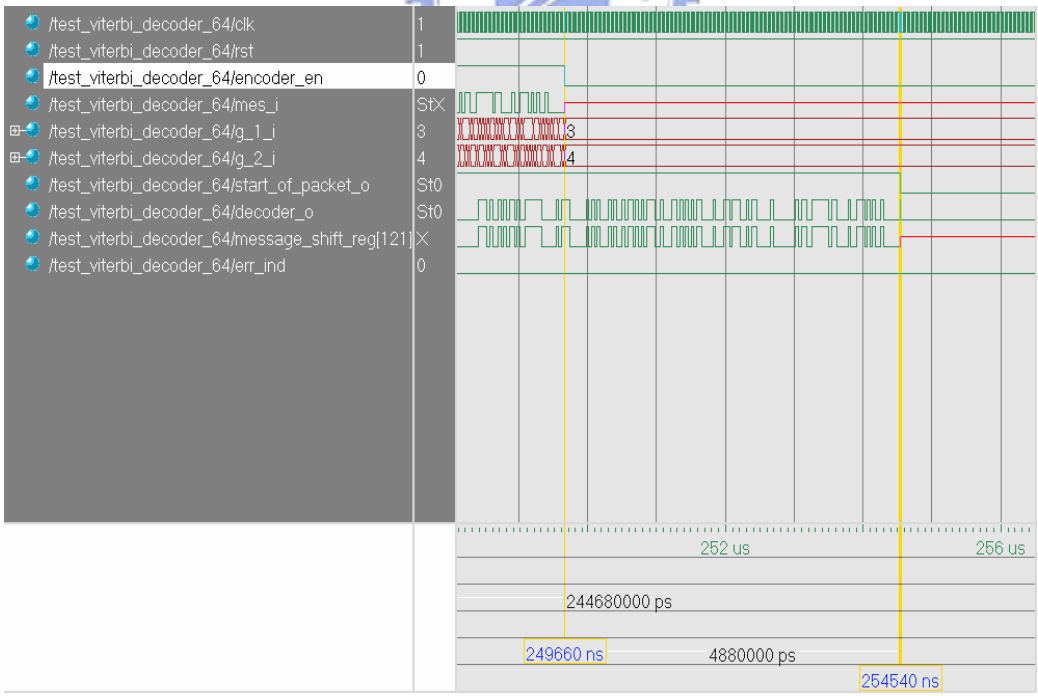


圖 4-39 Viterbi 解碼器硬體模擬圖 – (3)

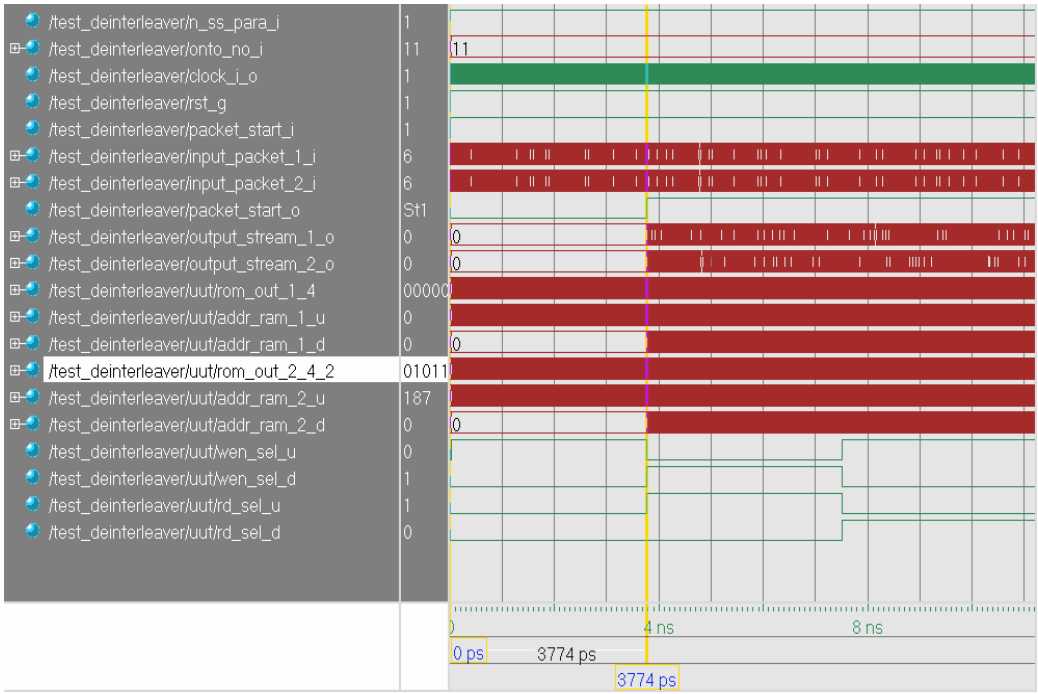


圖 4-40 反交錯器硬體模擬圖 –(1)

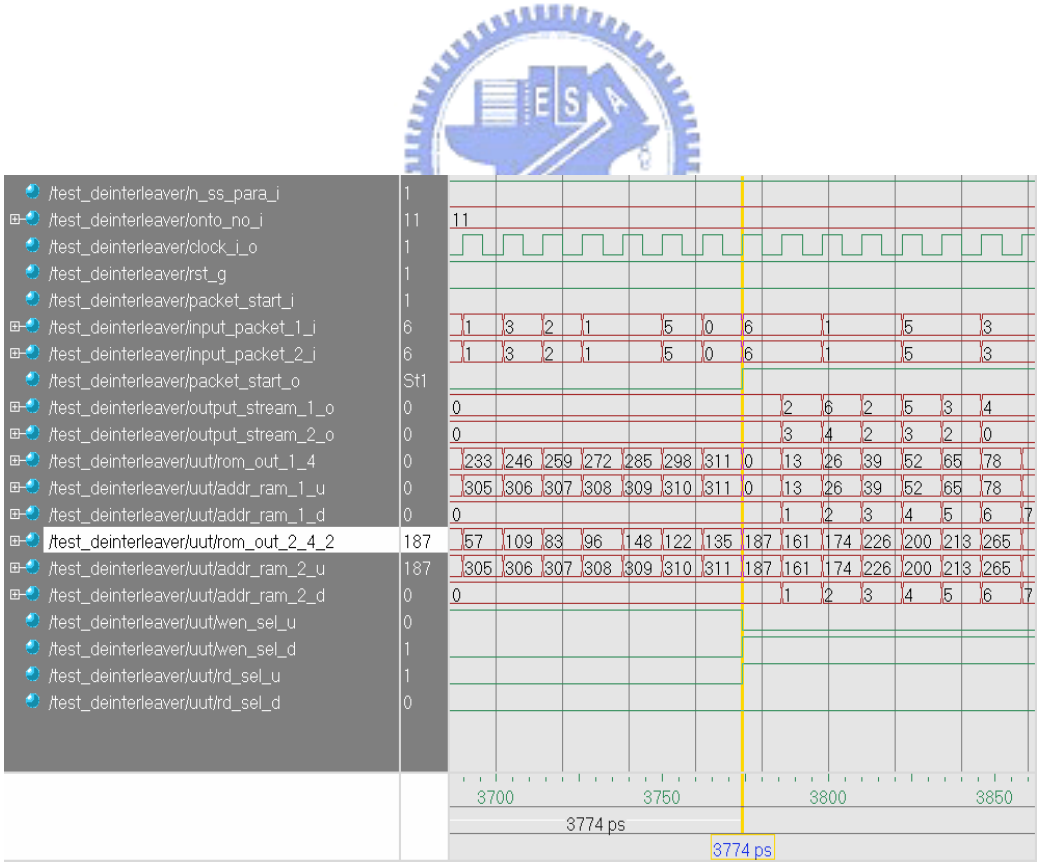


圖 4-41 反交錯器硬體模擬圖 –(2)

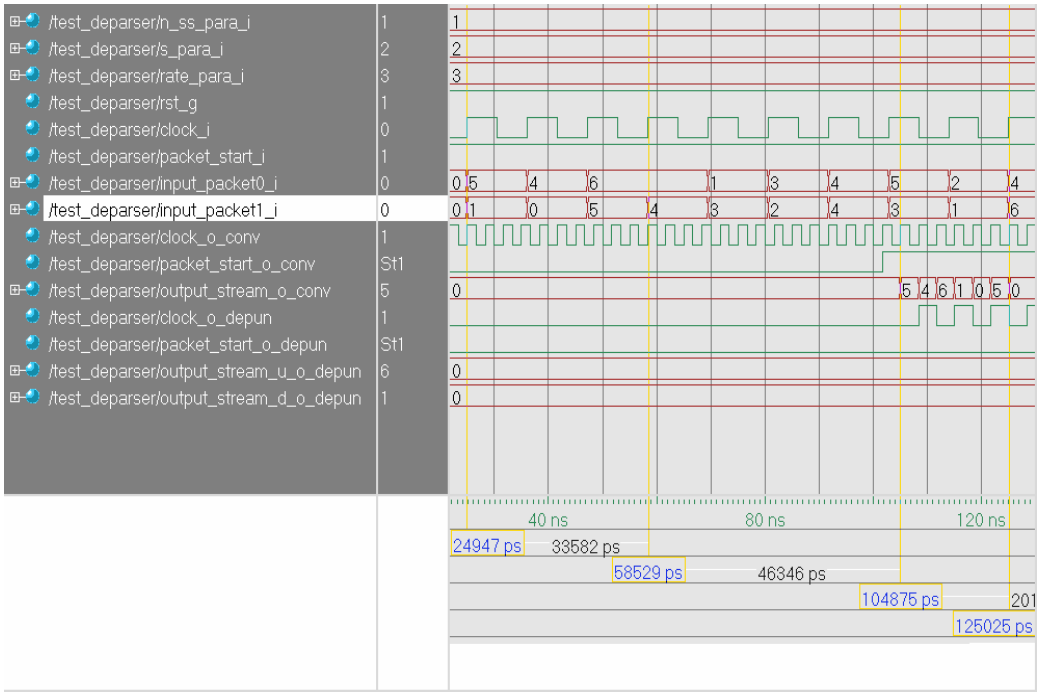


圖 4-42 反空間分離器硬體模擬圖

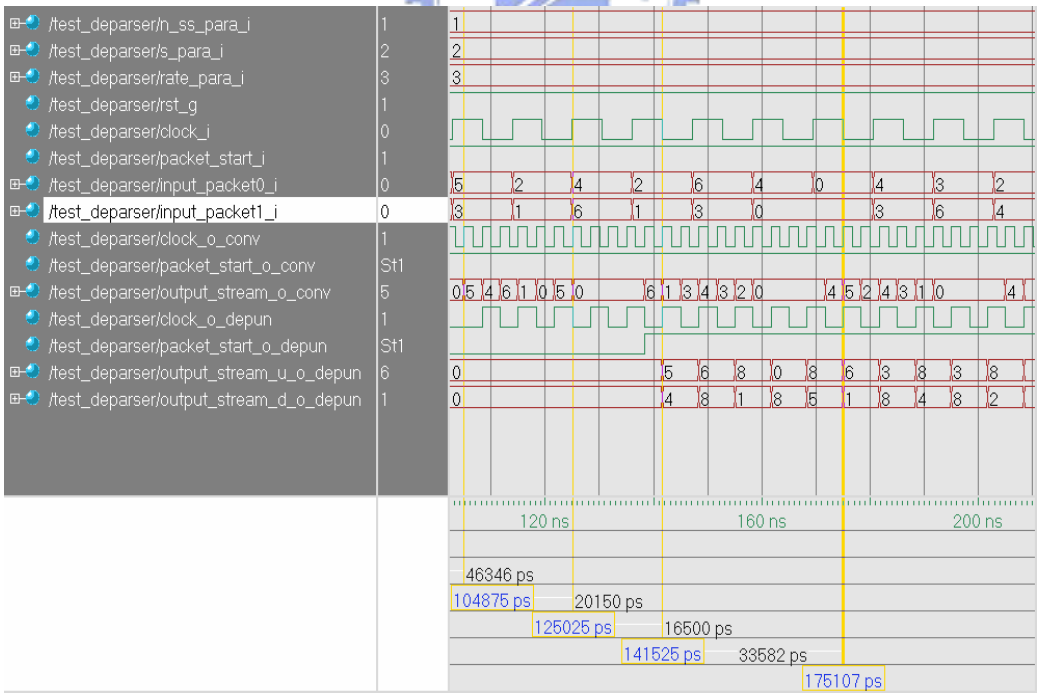


圖 4-43 反壓縮器硬體模擬圖

第5章結論

在本論文中，吾人考量如何在 BICM MIMO-OFDM 系統中結合 Viterbi 解碼器及偵測單元，而做一個整體的設計。論文主要的工作環境是在由 TGn Sync 所提出的 11n 草案中之傳送架構，在這個傳送架構下，吾人的設計重點包括了偵測演算法的選擇、CSI 計算與選擇及軟性反對映以及 Viterbi 解碼器。因此本論文在第 2 章中首先介紹了 11n 的系統，其中包括天線對應轉換之三種模式、11n 在不同頻寬模式下之 Tone 的配置以及 11n Preamble 格式之介紹，另外介紹了 BICM MIMO-OFDM 系統，其中包括了 11n 草案中傳送端之編碼器、壓縮器、交錯器等單元的介紹。而在第 3 章中，吾人介紹了在 MIMO-OFDM 中常用的偵測方法包括了 ZF、MMSE 及 V-BLAST 三種。接著介紹簡化之軟性反對映，並且在這樣的軟性反對映下，介紹了軟性輸入軟性輸出之 ZF 偵測單元以及 MMSE 偵測單元。另外介紹了 Viterbi 解碼器之演算法，而在第 3 章最後的模擬結果發現，ZF 演算法和 MMSE 演算法在效能上是非常接近的，故吾人決定在硬體實現上使用 ZF 演算法來降低計算的複雜度，進而達到節省面積降低成本的目的，而模擬結果分兩部分，在第一部分中，吾人使用一個簡單通道做模擬，而在第二部分中，吾人則針對 11n TGn Sync 所定義的通道模型做更進一步的模擬。

在論文的最後，吾人依一般 FPGA 設計流程使用 Verilog 實現第 3 章所介紹的 ZF 接收機，主要包括 ZF 偵測單元及 Viterbi 解碼器。在硬體實現上，ZF 偵測器相較於 MMSE 偵測器的確有較低的複雜度，而在 Viterbi 解碼器方面，Radix-4 之 Viterbi 解碼器架構的確在維持相同的面積效率下，提昇了資料的傳輸速度。然而吾人僅針對 2×2 之 MIMO-OFDM 系統做實現，當天線數增加時，ZF 偵測單元實現上所產生的乘法器個數將大大的提高，而 Viterbi 解碼器的資料傳輸速度也將提高，因此降低偵測單元的運算複雜度及提高 Viterbi 解碼器資料傳輸的速度將是未來實現的主要改進空間。

參考文獻

[1] G.. J. Foschini and M.J. Gans, “On limits of wirelesss communications in a fading environment when using multiple antennas, ” *Wireless Pres. Commun.*, vol. 6, no. 3, pp.311-335, Mar. 1998

[2] A. van Zelst, “Space division multiplexing algorithms,” in *Proc. 10th Mediterranean Electrotech. Conf.*, vol. 44, pp. 744-756, Mar, 1998.

[3] A. van Zelst, R. van Nee, and G. A. Awater, “Space division multiplexing(SDM) for OFDM system,” In *Proc. IEEE Veh. Technol. Conf.*, May 200, pp. 1070-1074.

[4] B. Hassibi, “A fast square-root implementation for BLAST,” in Conf. Rec. Thirty-Fourth Asilomar Conf. Signals, Syst. Comput., 2000, pp. 1255–1259.

[5] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, “V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel,” in *Proc. ISSSE, 1998*, pp. 295–300.

[6] G. D. Golden, C. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, “Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture,” *Electron. Lett.*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[7] V. Tarokh, N. Seshadri, and A. R. Calderbank, “Space-time codes for high data rate wireless communication: Performance criterion and code construction,” *IEEE Trans. Inform Theory*, vol. 44, pp. 744–756, Mar.1998.

[8] “TGn Sync proposal technical specification, ” *TGn Sync*, Mar. 2005.

[9] *IEEE 802.11a Stand., ISO/IEC 8802-11:1999/Amd 1:2000(E*)

[10] B. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K. D. Kammeyer, “Efficient algorithm for decoding layered space-time codes,” *Electron. Lett.*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[11] Choi , Jinho, “A bi-directional zero-forcing BLAST receiver” *IEEE transactions on Signal Processing*, vol. 52, no.9, pp.2670-2673, Sept. 2004.

[12] D., Wubben, , Bohnke, R., Kuhn, V., Kammeyer, K.-D.,“MMSE extension of V-BLAST based on sorted QR decomposition,” *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th* , vol. 1, Oct. 2003.

[13] Viterbo, E., Bouros, J., “A universal lattice code decoder for fading channels,” *IEEE Transactions on Information Theory*, vol. 45, pp. 1639 – 1642, July 1999.

[14] H. Vikalo and B. Hassibi, “The Expected Complexity of Sphere Decoding, Part I: Theory, Part II: Applications,” *IEEE transactions on Signal Processing, submitted for publication*, 2003.

[15] G. Caire, G. Taricco and E. Biglieri, “Bit-interleaved coded modulation,” *IEEE Trans. Info. Theory,* vol. 44, pp. 927-946, May 1998.

[16] F. Tosato and P. Bisaglia, “Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2,” *Proc. IEEE Int’l. Conf. on Comm. 2002*, vol. 2, pp. 664-668 , 2002.

[17] P.J.Black, and T.H.-Y.Meng, “A 140Mb/s, 32-state, radix-4 Viterbi decoder,” in *ISSCCDig. Tech. Paper*, Feb. 1992, pp. 70-71.

[18] J. A. Heller and I. M. Jacobs, “Viterbi decoding for satellite and space communication, ” *IEEE Trans. Commun. Technol.*, vol. COM-19, no. 5, pp. 835-848, Oct. 1971.

[19] Yu-xin You, Jin-xiang Wang, Feng-chang Lai, Yi-zheng Ye, “VLSI design and implementation of high-speed Viterbi decoder,” *Proc. IEEE Int’l. Conf. on Comm. 2002*, vol. 1, July 2002, pp. 64-68

[20] Fettweis G., Meyr H., “A 100 Mbit/s Viterbi decoder chip: novel architecture and its realization,” *Communications, 1990. ICC 90, Including Supercomm Technical Sessions. SUPERCOMM/ICC '90. Conference Record., IEEE International*

Conference 1990, vol. 2, April 1990, pp. 463-467

[21] Gemmeke T., Gansen M., Noll T.G., "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *Solid-State Circuits, IEEE Journal 2002*, vol. 37, Issue 7, July 2002, pp. 941-948

[22] O. Oteri, A. Paulraj, W. J. Chimitt, K. Holt, "SPACE-TIME-FREQUENCY CODING FOR OFDM-BASED WLANs," http://whitepapers.zdnet.co.uk/0,39025945,60131031p-39000516q,00.htm

[23] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm" *IEEE Transactions on Information Theory* ,vol. 13, issue 2, Apr 1967, pp.260 – 269.

[24] C. B. Shung, Gottfrried Ungerboeck, H. K. Thapar, "VLSI Architectures for Metric Normalization in the Viterbi Algorithm", *IEEE International Conference* vol.4, 1999, pp. 1723 –1728.

[25] Stephen B. Wicker, "Error Control Systems for digital communication and storage", 1995 by Prentice-Hall, Inc.

[26] C. M. Rader, "Memory management in a Viterbi algorithm" *IEEE Trans. Commun.*, vol. 29, Sept.1981, page(s) : 1399-1401.

[27] Gennady Feygin and P. G. Gulak, "Architectural Tradeoffs for Survivor Sequence Memory Management in Viterbi Decoders", *IEEE Trans. Commun*, vol. 41. No. 3. March 1993.

[28] Michael Horwitz and Robin Braun, "A generalized Design Technique For Trace back Survivor Memory Management In Viterbi Decoders", *IEEE*, 1997.

簡歷

姓　　名：　蔡耀毅

性　　別：　男

出生日期：　民國 60 年 12 月 28 日

出生地　：　新竹市

學　歷　：

新竹市立民富國小　　(1977.9~1983.6)

新竹市立成德國中　　(1983.9~1986.6)

省立新竹高級中學　　(1986.9~1989.6)

國立台北工專電子科　　(1989.9~1992.6)

國立台灣科技大學電子系　　(1996.9~1998.6)

國立交通大學電信工程研究所碩士班(2002.9~2005.10)

公元 2005 年 10 月獲得碩士學位