第四章　通道特性的估計與快速傅立葉轉換

本章是最後一章在解說硬體的設計與實現，在第二章中，我們說明了如何偵測封包的存在與頻率偏移的估計與補償，但是在補償頻率偏移的時候，還是會留下一固定的偏移無法解決；除此之外，在第三章中，封包的同步亦無法達到盡善盡美的地步，而可能會有正交分頻多工符元起始位置偏移的問題；上述的兩個效應我們都可以歸納成通道的特性，而本章的重點在於通道特性的估計，目的就是為了能補償上述兩者的問題以及實際通道所帶來的影響。

在本章中，除了估計通道的特性外，也會介紹正交分頻多工技術中，最重要的運算元：快速傅立葉轉換。一般來說，快速傅立葉轉換都會採用記憶體基礎的架構，但是在本論文的硬體設計當中，為了減少記憶體的使用，所以我們將用管線為基礎的快速傅立葉轉換來替代一般採用的記憶體基礎，藉此降低硬體的負擔。

4.1 通道特性的估計

承上所述，通道特性所產生的問題可能來自於三個地方：第一個是由於頻率偏移補償時，所造成的一固定偏移無法消弭；其來自於接收端求得頻率偏移的時候，已經無法得知之前通過的取樣點個數；比如來說，假設在此次的頻率偏移補償中，接收機於第 D 個取樣點求得了頻率的偏移，但是在補償時，我們無法得知 D 值的大小，因此只能對在求得頻率偏移後的取樣點，用一倍、兩倍……等的頻率偏移來補償，而這就會造成雖然已經估出正確頻率偏移，但是依舊會有一固定的殘存餘偏無法補償的現象。當然要不利用通道特性的特性，來克服這個問題也是可以，但這勢必需要一個記數器，從接收機運作開始，一值進行記數的動作，用來求得 D 值的大小，但是我們根本無法估計 D 值的範圍，因此這個記數器的

字元長度根本無法定義，可行的方法就是取一個很大的字元長度，但是這樣依然會有錯誤的可能，而且也不切實際，因此一般來說都會將這個固定的殘存餘偏無法補償的效應，視為通道特性的一環。

第二個可能的原因來自於封包取樣點的同步，在第三章中，我們說過取樣點的同步，只需要將符元的起始位置同步於循環字首內即可，如此便不會有不同區塊間干擾的問題(Inter-Block Interference)，但是同時會造成資料在頻域上會有角度的旋轉，但是這個角度的旋轉對於在同一個位置上的頻譜取樣點是固定的，因此亦可以將其視為通道特性的影響之一。

最後一個可能的原因就是來至於本身通道的特性，由於本身是無線傳輸的技術，所以傳送端與接收端的通道特性，會受到兩者間的建築物、雜訊……等，而使得通道的特性非常複雜，若不估計出通道的特性而加以補償的話，則經過解調後的資料錯誤率將會大增；因此本章的目的就在於求得通道的特性以利補償之用，藉此降低資料的錯誤率。

4.1.1 估計通道特性的演算法

在估計通道特性的演算法上，我們依然是採用調整符元已知的特性：因為調整符元是傳送端與接收端都已知的，因此當接收端將已知的調整符元，和接收到的調整符元作比較，便可以找到通道的特性，進而作補償的動作。但是調整符元有兩個：短調整符元和長調整符元；該選取哪個來作通道特性的估計比較好呢？基本上兩者都可以，但是對於硬體在時間上設計的考量，我們將使用長調整符元作為通道特性估計的依據。

承上所述，我們的目的就在於要將接收到的長調整符元和已知的長調整符元，作比較以求得通道的特性；在比較的演算法上，我們採用最大可能性估計法(Maximum Likelihood Estimation，ML)；在沒有其他雜訊的干擾下，假設當初傳送端傳送的時間訊號為 $x(t)$ 、通道特性的脈衝響應(Impulse Response)為 $h(t)$ ，則接收到的訊號 $y(t)$ 即可由下式(4.1.1.1)而得：

$$y(t) = x(t) * h(t) \qquad \text{式(4.1.1.1)}$$

也就是 $y(t)$ 可以由 $x(t)$ 和 $h(t)$ 的迴旋積而得到，若將上式轉換為頻域上來看，可得到下式(4.1.1.2)：

$$Y(f)=X(f)\cdot H(f) \qquad \text{式(4.1.1.2)}$$

其中 $Y(f)$、$X(f)$ 和 $H(f)$ 分別是 $y(t)$、$x(t)$ 和 $h(t)$ 在頻域上的轉換；當然上式是考慮在理論的情況下，實際的情形還需要將雜訊的干擾考慮在內；因此，我們必須要應用最大可能性估計法，才可以找出最有可能的通道特性 $\hat{H}(f)$；依照最大可能性估計法，所求得的 $\hat{H}(f)$ 如下式(4.1.1.3)：

$$\hat{H}(f)=Y(f)\big/X(f) \qquad \text{式(4.1.1.3)}$$

或是 $$\hat{H}[k]=Y[k]\big/X[k] \qquad \text{式(4.1.1.4)}$$

在上式(4.1.1.3)是使用在當輸入訊號為連續訊號之時，而式(4.1.1.4)則是使用在當訊號為離散訊號的時候，其中 $\hat{H}[k]$ 表示的是第 k 根載波、利用最大可能性估計法估計出來的通道特性，而 $X[k]$ 和 $Y[k]$ 分別代表的是已知的離散訊號 $x[n]$，和接收到的離散訊號 $y[n]$ 經過快速傅立葉轉換後的第 k 根載波上所載有的資料；因此，只要我們將接收到的長調整符元經過快速傅立葉轉換後，在配合著已知的長調整符元的頻譜響應，便可以利用式(4.1.1.4)求得估計出來的通道特性。

4.1.2 內插法

基本上，利用前一節的最大可能性估計法(Maximum Likelihood Estimation)便可以有效的估計出通道的特性，而這估計出來的通道特性式包括了本章前言中所述的三種成分：頻率偏移補償的固定殘存餘偏、封包取樣點同步的失誤以及真正的通道特性。乍看之下，似乎已經足夠把全部二百五十六根的通道特性估計出來，但是實際上這是無法達成的，其中最主要的關係是因為長調整符元在頻譜響應上的特性；誠如第一章所述：在頻譜上，長調整符元載波所載有的資料，每兩個有效的資料間，會夾雜有一個不載有資料的載波，而這些不載有資料的載波便

成為我們無法估計出通道特性的關鍵；在式(4.1.1.4)中，該式的成立要在當 $X[k]$ 不為零的情況下，也就是說，必須要在當 $X[k]$ 不為零的情況下，才可以估計出通道的特性；可是今天不幸的，在原本長調整符元頻譜上所載的訊號裡，原本兩百五十六根的載波上，僅只有兩百根載有資料；而這兩百根載波上，更是有一百個所載的資料為零；因此這一百個通道的特性，將無法利用最大可能性估計法估計出來，所以我們必須要採用另外一種演算法來估計這些所載資料為零的通道特性。

在此我們所使用的方法是：一階拉格朗日(First Order Lagrange)內插法，其實也就是線性內插法，來獲得這一百根無法估計出來的通道特性。能使用內插法來求得這些無法估計出來的通道特性，最主要的原因有二：第一是因為這些無法估計的載波一定位於兩個可以估計的載波間；第二是通道的脈衝響應一定是連續的，而且一般來說具有平滑的特性，也就是不會在極短的頻段內有劇烈的上下震盪。當然也不一定要使用一階的拉格朗日，如果需要估計出更精準的通道特性的話，使用更高階的拉格朗日內插法也是可行的，只是相對的會增加硬體的負擔。

下表(4.1.2)是長調整符元所求得的一百根通道特性 $\hat{H}_L[k]$，與經過線性內插法後所求得的兩百根通道特性 $\hat{H}[k]$ 的對照表：

$\hat{H}[1]$	$\hat{H}_L[1]$	$\hat{H}[101]$	$\hat{H}_L[51]$
$\hat{H}[2]$	$\hat{H}_L[1]$	$\hat{H}[102]$	$\left(\hat{H}_L[51]+\hat{H}_L[52]\right)/2$
$\hat{H}[3]$	$\left(\hat{H}_L[1]+\hat{H}_L[2]\right)/2$	$\hat{H}[103]$	$\hat{H}_L[52]$
$\hat{H}[4]$	$\hat{H}_L[2]$	$\hat{H}[104]$	$\left(\hat{H}_L[52]+\hat{H}_L[53]\right)/2$
⋮	⋮	⋮	⋮
$\hat{H}[97]$	$\left(\hat{H}_L[48]+\hat{H}_L[49]\right)/2$	$\hat{H}[197]$	$\hat{H}_L[99]$
$\hat{H}[98]$	$\hat{H}_L[49]$	$\hat{H}[198]$	$\left(\hat{H}_L[99]+\hat{H}_L[100]\right)/2$

$\hat{H}[99]$	$\left(\hat{H}_L[49]+\hat{H}_L[50]\right)/2$	$\hat{H}[199]$	$\hat{H}_L[100]$
$\hat{H}[100]$	$\hat{H}_L[50]$	$\hat{H}[200]$	$\hat{H}_L[100]$

表 4.1.2 兩百根經過內插法後得到的通道特性

4.1.3 估計通道特性的硬體架構

下圖(4.1.3.1)所示即是整個估計通道特性的硬體架構圖，在此硬體設計中，主要可以分為三個模式：第一個模式是將第一個長調整符元所解出來的$\hat{H}_{L1}[k]$寫入記憶體當中；第二個模式是將兩個長調整符元所解出來的$\hat{H}_{L1}[k]$和$\hat{H}_{L2}[k]$取平均，藉此減少錯誤的產生；第三個模式是將$\hat{H}[k]$求出，並同時對解調出來的資料，作通道特性的補償，詳細的運作情形，我們將會在本章 4.1.3.2 介紹。

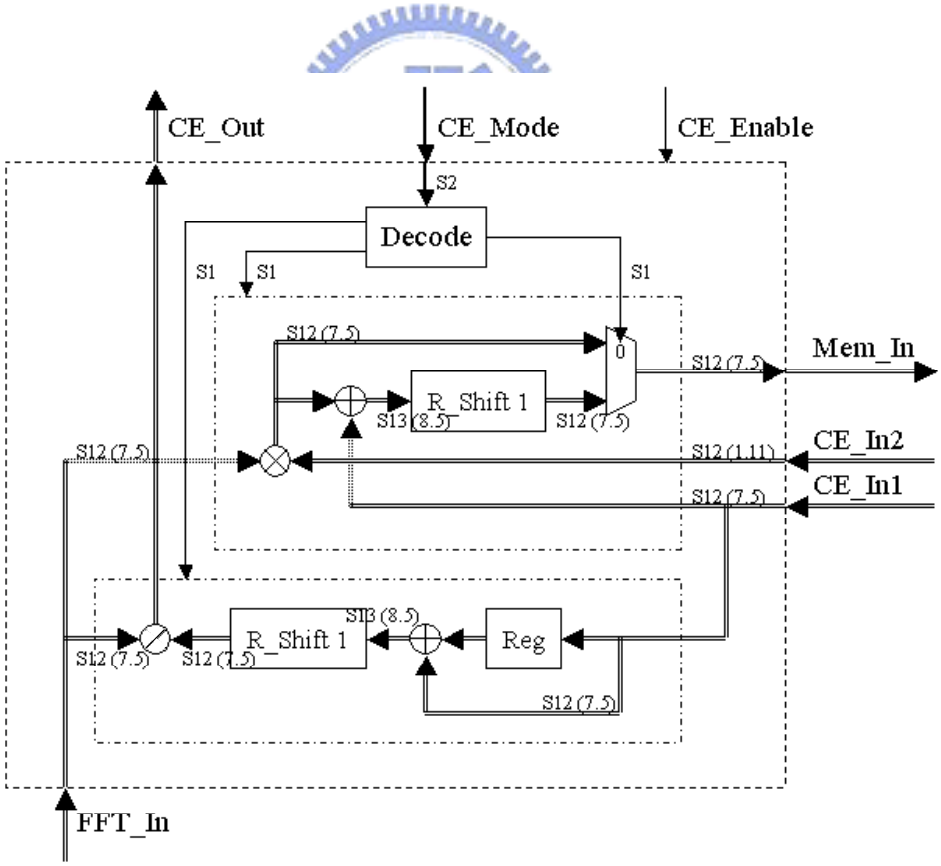

圖 4.1.3.1 估計通道特性的硬體架構

4.1.3.1 輸入輸出以及控制訊號

資料輸入訊號：

- CE_In1：從隨機存取記憶體中讀取$\hat{H}_{L1}[k]$或$\hat{H}_{L}[k]$，有實部、虛部訊號之分
- CE_In2：從唯讀記憶體中讀取已知訊號$X[k]$，只有實部訊號
- FFT_In：從傅立葉轉換運算元接收調整訊號$Y[k]$，有實部、虛部訊號之分

資料輸出訊號：

- CE_Out：經過通道特性補償後的資料，有實部、虛部訊號之分
- Mem_In：將$\hat{H}_{L1}[k]$或$\hat{H}_{L}[k]$寫入隨機存取記憶體中，有實部、虛部訊號之分

控制訊號：

- CE_Mode：輸入控制訊號，控制硬體元轉的模式，只有實部訊號
- CE_Enable：輸入控制訊號，控制通道估計與補償是否運作，只有實部訊號

4.1.3.2 硬體運作的機制

誠如之前所述，整個通道特性估計硬體可以分為三種模式；第一個模式是針對第一個長調整符元內的週期區間、其在頻譜上所載的資料$Y_{L1}[k]$，作通道特性的評估，而所獲得的通道特性$\hat{H}_{L1}[k]$如下式(4.1.3.2.1)：

$$\hat{H}_{L1}[k]=Y_{L1}[k]\big/X[k] \qquad \text{式(4.1.3.2.1)}$$

其中$X[k]$是我們已知長調整符元，在頻譜上所應載有的資料；在上式中，我們需要一個除法器來完成求取通道特性的機制，但誠如之前所述，除法器的設計會需要龐大的硬體負擔，所以我們亦想如之前一般，使用乘法器來替代除法器的使用；由於$X[k]$是我們已知的資料，所以我們亦可以事先求得$X^{-1}[k]$，並將其儲存於唯讀記憶體中，當$Y_{L1}[k]$經由快速傅立葉轉換獲得後，利用式(4.1.3.2.1)的變形、式(4.1.3.2.2)，獲得$\hat{H}_{L1}[k]$：

$$\hat{H}_{L1}[k]=Y_{L1}[k]\cdot X^{-1}[k] \qquad \text{式(4.1.3.2.2)}$$

因此，第一個模式的運行機制便是從 FFT_In 訊號線取得$Y_{L1}[k]$，同時與從

CE_In2 所獲得的 $X^{-1}[k]$ 相乘，用以獲得 $\hat{H}_{L1}[k]$ 並經由 Mem_In 的訊號線寫回隨機存取記憶體中，利於將來第二個模式的實現。

第二個模式則是承接著第一個模式而來，除了同第一個模式用第二個長調整符元內的週期區間、其在頻譜上所載的資料 $Y_{L2}[k]$，求得其對應的通道特性 $\hat{H}_{L2}[k]$ 外，還有將 $\hat{H}_{L1}[k]$ 和 $\hat{H}_{L2}[k]$ 取平均值以獲得 $\hat{H}_L[k]$ 的機制，其數學式如下式(4.1.3.2.3)：

$$\hat{H}_{L2}[k] = Y_{L2}[k] \cdot X^{-1}[k]$$

$$\hat{H}_L[k] = \left(\hat{H}_{L1}[k] + \hat{H}_{L2}[k]\right) \Big/ 2 \qquad \text{式(4.1.3.2.3)}$$

因此，第二個模式的機制便是將從 FFT_In 訊號線而來的 $Y_{L2}[k]$ 資料，與 CE_In2 訊號線傳來的 $X^{-1}[k]$ 資料相乘、作通道特性的評估以獲得 $\hat{H}_{L2}[k]$，同時亦從 CE_In1 的訊號線，將剛第一個模式寫入隨機存取記憶體的 $\hat{H}_{L1}[k]$ 資料讀取出來，並與剛求得的 $\hat{H}_{L2}[k]$ 利用加法和右移暫存器作平均的動作，而獲得 $\hat{H}_L[k]$ 的結果，最後將結果透過 Mem_In 的訊號線，寫回隨機存取記憶體內，準備第三個模式的使用。

第三個模式的目的，主要是用來實現通道特性補償以及內插法的架構；其運作的機制是從 CE_In1 訊號線中讀取欲求得 $\hat{H}[k]$ 所需的兩個 $\hat{H}_L[k]$ 資料，接著如同第二個模式一般，利用加法器和右移暫存器，實現線性內插法的機制；而 $\hat{H}[k]$ 和所需的兩個 $\hat{H}_L[k]$ 資料的關係，可藉由表(4.1.2)中得知；舉例來說，如果現在需要求得 $\hat{H}[1]$，則控制單元便會控制隨機存取記憶體，經由 CE_In1 的訊號線，送出兩次 $\hat{H}_L[1]$ 的資料，再經過相加、右移一位元後，即可獲得滿足表(4.1.2)關係式的 $\hat{H}[1]$；同理，若現在需要求得的是 $\hat{H}[3]$，則控制單元便會控制隨機存取記憶體分別送出 $\hat{H}_L[1]$ 和 $\hat{H}_L[2]$，然後依照上面同樣的原理，取得正確 $\hat{H}[3]$ 的資料。

上面所述的是第三個模式用來實現線性內插法的機制，但是第三個模式除了作線性內插外，還要能補償通道的特性；假設接收機接收到在頻譜上，第 k 個載波所載的資料為 $Y[k]$，而該載波所對應到的通道特性為 $\hat{H}[k]$，則經過補償通道

特性所還原的資料$\hat{X}[k]$需要滿足下式(4.1.3.2.4)：

$$\hat{X}[k] = Y[k] \Big/ \hat{H}[k] \qquad \text{式(4.1.3.2.4)}$$

因次當載波資料$Y[k]$經過 FFT_In 的訊號線送入此硬體架構後，其相對應的$\hat{H}[k]$也會利用前述的方法同時求得，接著經過除法器作通道特性的補償，而獲得的還原的資料$\hat{X}[k]$便會經由 CE_Out 訊號線送出，以利後端的處理。

4.1.3.3 字元長度的選擇

在第二章中，我們已經說過我們使用的記憶體、其寬度為十二個位元大小，所以 CE_In1、CE_In2 和 Mem_In 等的字元長度皆為十二個字元，以利於記憶體的存取；因次不論是在第二個模式裡，針對$\hat{H}_{L2}[k]$和$\hat{H}_{L2}[k]$取平均以獲得$\hat{H}_{L}[k]$、亦或者是在第三個模式裡，對$\hat{H}_{L}[k]$作線性內插；在其加法器後的字元長度，都需要十三個位元來表示，而其後端的右移暫存器則會使其結果捨去一位元，而還原成原本的十二個字元，剛好配合於記憶體的存取。

4.2 快速傅立葉轉換

接著我們來看正交分頻多工技術中，運算元的核心：快速傅立葉轉換。所謂的正交分頻多工技術其實就是將要傳送的資料，載放到頻譜上的一種方法，這和以往將資料載在時域上傳送的觀念不同，由於必須要將時域上的資料，轉換到頻域上，所以傅立葉轉換是不可或缺的運算單元；可是在快速傅立葉轉換的硬體架構出來前，雖然正交分頻多工的理論非常的純熟，但是受限傅立葉轉換龐大的硬體負擔，所以當時並未有效地在實際的硬體上實現；直到快速傅立葉轉換的問世，正交分頻多工技術的調變才大放光彩。

快速傅立葉轉換的硬體實現，其實有許許多多種的形式，端看使用者的需要來決定；一般來說，都是採用記憶體基礎的架構，也就是將欲經過傅立葉轉換的資料，先行存入記憶體中，接著利用資料位置產生器(Data Address Generator)產生欲讀取資料的位置以及欲寫入資料的位置，將資料讀取出來，經過蝴蝶

(Buutterfly)硬體架構的運算後，將運算的結果寫回記憶體中，以利下次讀取的動作，直到完成整個傅立葉轉換。

記憶體基礎的快速傅立葉轉換在記憶體的使用上也可以分為兩種，一種是採用單記憶體的方式，而另外一種是採用雙記憶體的方式；單記憶體的方式，適用於當讀取資料的位置與寫入資料的位置相同的時候，如此可以有效的利用該塊記憶體，用以完成傅立葉轉換的運算；但是由於一般的隨機存取記憶體在一個時脈(Clock)下，只能作儲存或是讀取的動作；因此對於一個蝴蝶架構的運算，我們將會需要兩個時脈來完成；反觀若是採用雙記憶體的方式，讀取的資料與寫入的資料將不用要求在相同的位置，而且一個蝴蝶架構的運算，由於讀取和寫入的位置，是在兩個不同的記憶體，故可以在一個時脈下完成；但是相較於單記憶體的負擔上，會需要使用多一倍大小的記憶體。下圖(4.2.1)即是兩者的式意圖：

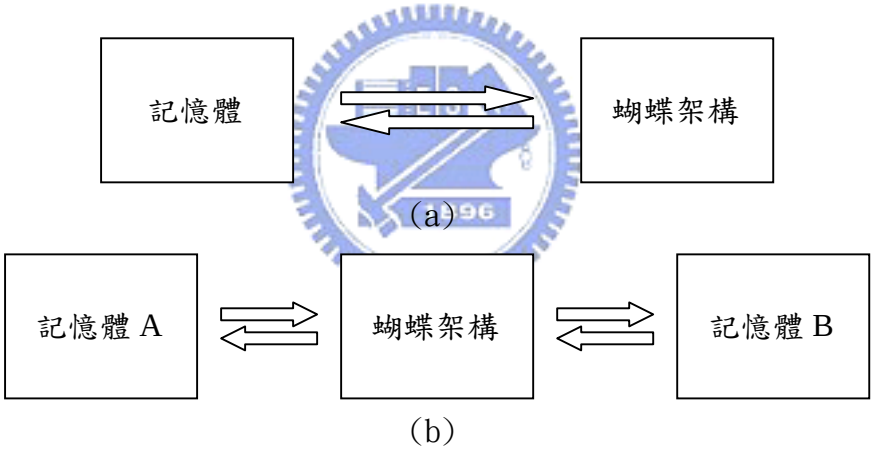

圖 4.2.1 (a)單記憶體、(b)雙記憶體式意圖

以上就是以記憶體為基礎的架構、其運作的基本模式；但是這對我們在時間上吃緊的硬體設計，不論是採用單記憶體或是雙記憶體，都不適合。因此我們需要使用另外一種基礎的架構，名為管線快速傅立葉轉換(Pipeline Fast Fourier Transform)；假設現在需要作兩百五十六點的傅立葉轉換，則管線快速傅立葉轉換的優點便在於當欲轉換的兩百五十六點資料都送入該硬體後，下一個時脈的運作便會開始將經過傅立葉轉換後的結果，隨著時脈的起伏一一送出，期間不會有任何的延遲；這樣的硬體非常符合我們的需要，這也是我們選用管線快速傅立葉

轉換架構的原因。

4.2.1 管線快速傅立葉轉換的硬體架構

在看管線快速傅立葉轉換的硬體架構前，我們先來看一下Radix-2^2分頻快速傅立葉轉換的流程圖，即下圖(4.2.1.1)：

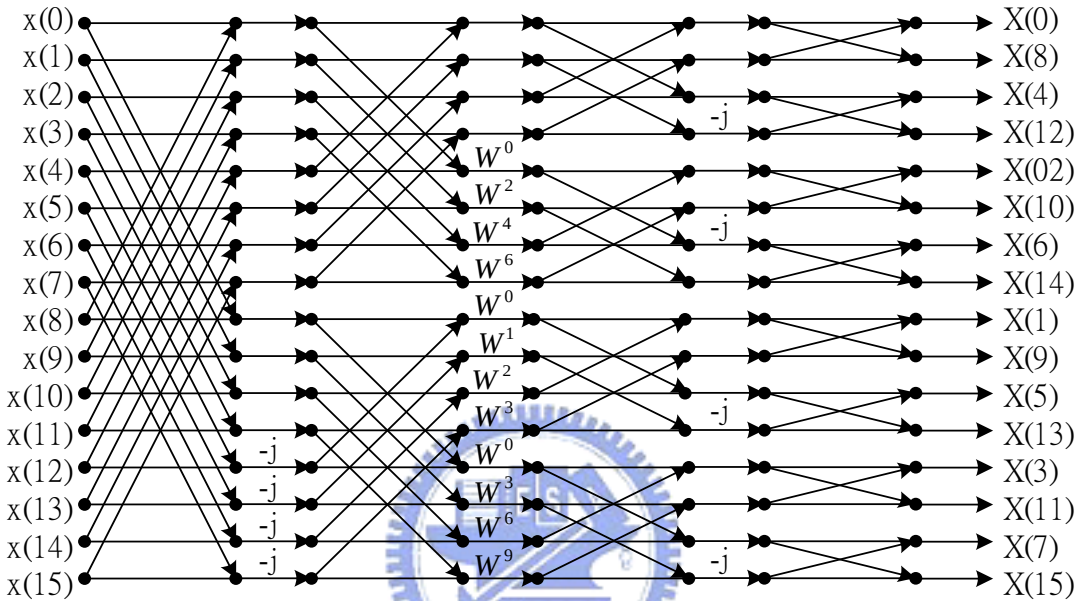

圖 4.2.1.1 Radix-2^2分頻快速傅立葉轉換的流程圖

在圖中是以十六點的快速傅立葉轉換為基本架構，訊號$x(k)$表示的是在時域上的取樣點，而$X(k)$則表示該取樣點，經過傅立葉轉換後所得到在頻域上的結果；W^k表示的十六點快速傅立葉轉換所對應的轉動係數(Twiddle Factor)，其數學表示式如下式(4.2.1.1)：

$$W^k = e^{-j(2\pi/16)k} \qquad for \qquad k = 0,1,2.....15 \qquad \text{式(4.2.1.1)}$$

在圖中，我們亦可以發現在時域上的取樣點$x(k)$，是依照取樣的順序來排列的，因此我們可以將此架構在時間上展開成下圖(4.2.1.2)的架構，而這也就是我們所使用的管線基礎快速傅立葉轉換的硬體；在圖(4.2.1.2)上方的數字：8、4、2、1，表示的是將輸入的資料，延遲的時脈個數；相關的運作，我們將在下一章節中，有詳細的介紹。

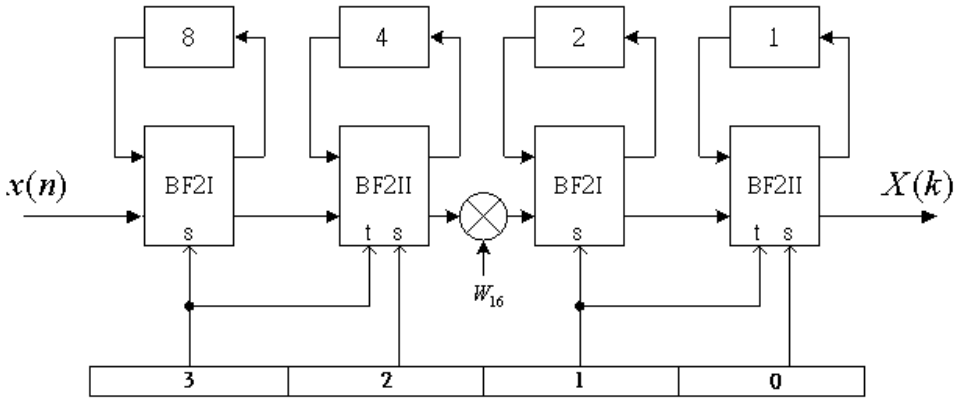

圖 4.2.1.2 管線基礎的快速傅立葉轉換硬體

下圖(4.2.1.3)和圖(4.2.1.4)則是在圖(4.2.1.2)當中所使用的兩種蝴蝶架構硬體的內視圖。

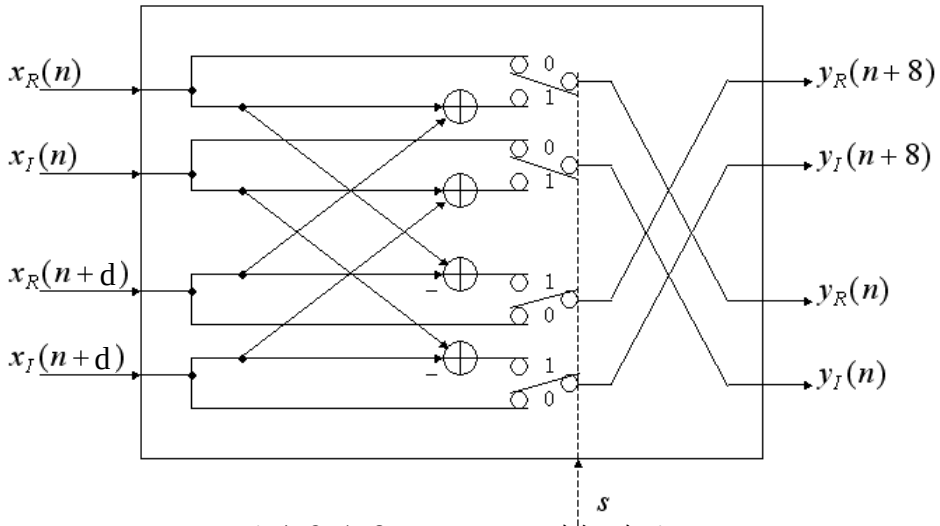

圖 4.2.1.3 BF2I 的硬體架構圖

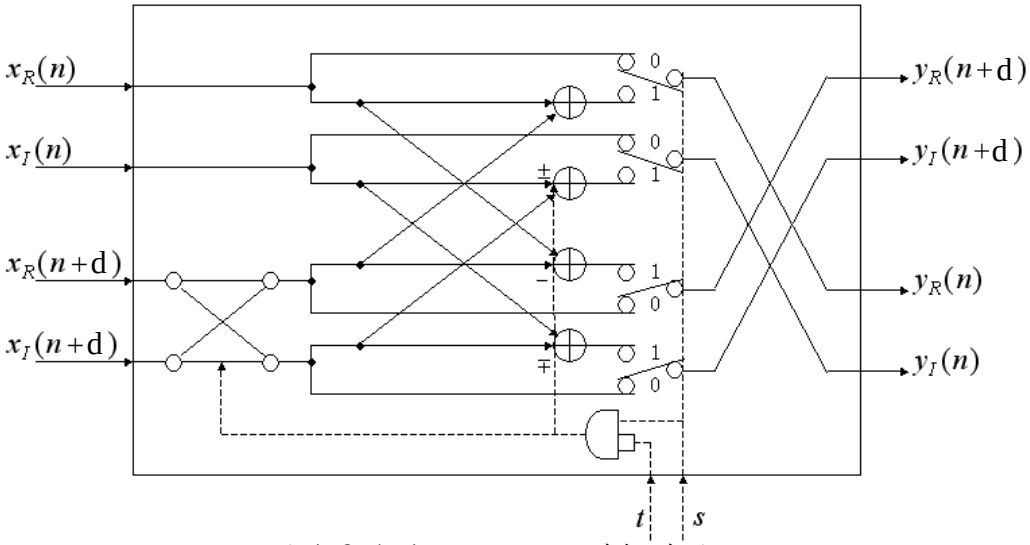

圖 4.2.1.4 BF2II 的硬體架構圖

4.2.2 管線快速傅立葉轉換的運行機制

基本上，圖(4.2.1.2)的架構就是來自於圖(4.2.1.1)中，Radix-2^2分頻快速傅立葉轉換的流程圖，因此本章的重點就在於介紹圖(4.2.1.2)中的架構將如何的運行，以實現圖(4.2.1.1)中的流程。

整個管線快速傅立葉轉換的運作，除了由時脈訊號(Clock)來控制暫存器的閂鎖(Latch)外，其運行的情況則是由另一個四位元的記數器來決定；圖(4.2.1.2)中，下方的 3、2、1、0，即是代表著這四位元記數器的輸出值，其中 3 表示的是最高有效位元(Most Significant Bit)，而 0 表示的是最低有效位元(Least Significant Bit)；這四個位元值的位準將會決定架構圖(4.2.1.2)中，蝴蝶架構的運行模式；以 BF2I 而言：控制訊號s將控制該蝴蝶架構的運行模式為將輸入訊號由旁路(Bypass)通過，或是執行快速傅立葉轉換的蝴蝶運算。而在 BF2II 中，除了控制訊號s控制剛在 BF2I 中所述的運行模式外，亦還有另外一個控制訊號t；這t控制訊號則是用來決定是否要將輸入的$x(n+d)$訊號，先乘上一個複數值$(-j)$，然後再執行後續的蝴蝶運算；而這個乘上$(-j)$的動作，我們又可以利用簡單的電路將其化簡，其中的原因在於：若將一個複數$A+B\cdot j$乘以$(-j)$，則得到的結果會為$B-A\cdot j$；所以我們僅只需要將$x(n+d)$訊號的實部與虛部對調，同時再控制蝴蝶運算中的加、減法使用，如此便可以省卻乘法器的使用。而這就是圖(4.2.1.4)中，硬體架構運作的機制。

接著我們來看圖(4.2.1.2)的架構圖中，訊號的流程；在前八個時間取樣點的值輸入時，由於四位元記數器僅只從零累加到七，所以對於這四位元記數器輸出的最高有效位元，也就是 3 的電壓位準，將會是低電位；因此造成在圖(4.2.1.2)中，由 3 所控制的 BF2I 進行將輸入訊號由旁路通過的模式而不作任何的處理；所以當$x(0)$、$x(1)$……$x(7)$輸入後，會經過 BF2I 寫入其上方，八個連續的暫存器內，準備作接下來的運算。

然後，當第九個取樣點，也就是$x(8)$輸入時，由於記數器累加到八，所以 3 的電壓準位會由低電位轉為高電位，使得 3 所控制的BF2I將進行快速傅立葉轉換的蝴蝶運算；而此時該BF2I的兩個輸入分別為$x(0)$和$x(8)$，形成與圖(4.2.1.1)

的Radix-2^2分頻快速傅立葉轉換的流程中，第一階段的第一個蝴蝶架構相同；而其運算出來的結果，會如圖(4.2.1.2)中所示：蝴蝶架構上方路徑的輸出會傳入下一個由 3 和 2 所控制的BF2II蝴蝶架構，而下方路徑的輸出，則會回傳寫入暫存器內。此時，由於記數器的輸出還是維持在八，所以 2 的電壓準位為低電位，使得由 3 和 2 所控制的BF2II蝴蝶架構會運作為旁路模式，將由 3 所控制的BF2I而來的輸入訊號，寫入該蝴蝶架構上方的四個連續暫存器內。

接下來的第十、十一、十二個取樣點輸入時，由於記數器所對應到的 3 和 2 的電壓位準沒有改變，所以維持著和第九個資料取樣點輸入時，同樣的運行模式；也就是 $x(1)$ 和 $x(9)$ 、 $x(2)$ 和 $x(10)$ 、 $x(3)$ 和 $x(11)$ 分別執行蝴蝶運算形成流程圖(4.2.1.1)中，第一階段的第二、第三、第四個蝴蝶架構，而蝴蝶運算的輸出結果，將會分別寫入圖(4.2.1.2)由左算起、第一個 BF2I 和第一個 BF2II 架構上方的暫存器內，如同之前所述。

接著當第十三個取樣點輸入時，由於 2 的電壓位準會由低電位轉為高電位，所以第一個 BF2II 的架構將不再是旁路模式；相反的，他會將此前一級 BF2I 輸入為 $x(3)$ 和 $x(11)$ 的蝴蝶運算結果，與經過暫存器延遲四個取樣點、$x(0)$ 和 $x(8)$ 的蝴蝶運算結果，再經過一次蝴蝶運算，形成圖(4.2.1.1)中，第二個階段的第一個蝴蝶架構。

接下來的硬體運作流程，基本上和上面所述的雷同，差別只在於暫存器所造成的延遲不同而已；因此，當第十四個取樣點輸入時，會形成流程圖(4.2.1.1)中第一階段的第六個蝴蝶架構，同時也會形成第二個階段的第二個蝴蝶架構；而第十五個取樣點輸入時，除了形成第一階段的第七個蝴蝶架構，和第二個階段的第三個蝴蝶架構外，亦會形成第三個階段的第一個蝴蝶架構。

最後就是當第十六個取樣點輸入時，除了形成第一階段的第八個蝴蝶架構外，還有第二個階段的第四個蝴蝶架構、第三個階段的第二個蝴蝶架構，以及最後一個階段、第四階段的第一個蝴蝶架構；因此當第十六個取樣點輸入時，我們將可以在圖(4.2.1.2)中，最右方的輸出訊號線得到第一個經過傅立葉轉換的結果 $X(0)$ ，而沒有像以記憶體為基礎的快速傅立葉轉換架構中，需要資料讀取的

延遲時間，這正是我們使用管線基礎的快速傅立葉轉換的原因。

在求得$X(0)$後，接下來的每一個時脈都會運算出一個快速傅立葉轉換的結果，但是這些結果並非是依序(In Order)的，而是照著圖(4.2.1.1)中，右方的輸出順序輸出；因此當輸入資料的第十六取樣點輸入後，我們將可以在其後的第十五個時脈，獲得完整的傅立葉轉換結果。