

國立交通大學

資訊科學與工程研究所

碩士論文

濾波反投影演算法之 FPGA 硬體平台實現

FPGA Implementation for Filtered Back Projection
Algorithm

研究生：張騰介

指導教授：荊宇泰 博士

中華民國 九十四 年 九月

濾波反投影演算法之 FPGA 硬體平台實現
FPGA Implementation for Filtered Back Projection Algorithm

研 究 生：張騰介

Student : Teng-Chieh Chang

指 導 教 授：荊宇泰

Advisor : Yu-Tai Ching

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文



A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年九月

濾波反投影演算法之 FPGA 硬體平台 實現

學生：張騰介

指導教授：荊宇泰博士

國立交通大學資訊科學與工程研究所



電腦斷層掃描自發明以來到現在，技術已經非常成熟，除了應用在醫療診斷外，也應用在其他領域。但是由於價格昂貴，體積龐大不易搬動，難以在其他領域堆廣。我們利用現今硬體製程進步，IC 技術發達之利，希望能設計一顆便宜的系統單晶片來執行影像重建的工作。我們使用濾波反投影演算法作為影像重組的理論基礎，並且使用 Xilinx 公司出產的 ML310 Development Board 作為我們實作的平台。實作出來的系統單晶片雛形在重建影像上效率符合預期，重建出來的影像也有良好的效果。

FPGA Implementation for Filtered Back Projection Algorithm

Student : Teng-Chieh Chang

Advisor : Dr. Yu-Tai Ching

Department of Computer and Information Science

National Chiao Tung Universal



Since Computer Tomography (CT) scanner was invented, many advances have been made in scanner technology as well as in the algorithm used for CT reconstruction. Today CT scanners are used in diagnostic medicine and other applications. But the CT scanner is expensive and too heavy to take it along, so CT is not popular except diagnostic medicine. Now the hardware and IC design progress rapidly, so we hope to design a cheap SoC for reconstructing image. The Filtered Back Projection Algorithm is the base theory of our design and we implement it on ML310 Development Board of Xilinx company.

Experimental results show that the efficiency of SoC model of the Filtered Back Projection Algorithm reach our anticipation and the quality of reconstructed image is feasible.



誌謝

在這裡首先我要感謝我的指導老師，荊宇泰教授，在這兩年中給予我自由學習的機會，也指導我研究的方向，更重要的是信任我的研究與學習。再來要感謝志陽、富祺、冠杰、書豪、方正、滄智學長們對我的關懷，尤其謝謝冠杰學長帶我接觸佛學的精神，讓我多些生活的體悟。謝謝滄智學長給我許多學習上的建議並與我一起努力共同渡過論文難產期。也謝謝這兩年來陪我一路走過來的同學，孟誌、樹偉、杰翰及雅榆。謝謝你們與我一起上課、一起熬夜寫程式、一起吃飯、一起休閒，在學習或生活上共同扶持。謝謝學弟們與我共同討論學習，尤其是秉章，對於研究的熱情讓我也學習不少。

最後感謝我的父母，我的家人以及我的姑姑。沒有你們的支持我無法考上交大資科所，沒有你們的鼓勵畢業的路也許更長。謝謝你們我的父母我的家人我的同學及朋友們，願將喜悅與你們分享。

目錄

中文摘要.....	i
英文摘照.....	ii
誌謝.....	iv
目錄.....	v
第一章 緒論.....	1
1.1 研究背景.....	1
1.2 論文架構.....	2
第二章 影像重組理論.....	4
2.1 影像重建演算法.....	5
2.1.1 Fourier Slice Theorem	6
2.1.2 濾波反投影演算法.....	9
2.1.3 濾波反投影演算法實作.....	12
第三章 系統單晶片設計與發展平台.....	17
3.1 系統單晶片設計.....	17
3.1.1 軟硬體共同設計.....	18
3.2 嵌入式系統 SoC 發展平台.....	19
第四章 FBP 演算法嵌入式系統.....	23
4.1 嵌入式系統軟體實作.....	24
4.1.1 FBP 演算法分析與實作.....	24
4.2 嵌入式系統硬體實作.....	25
4.2.1 嵌入式系統硬體架構.....	25
4.2.2 建立 FBP 週邊元件.....	28
第五章 FBP 元件實作.....	34
5.1 IP 介紹及 fbp_locate 模組實作.....	35
5.1.1 Single-Port Blcok Memory IP	35
5.1.2 Fast Fourier Transform v3.1 IP.....	38
5.1.3 fbp_locate VHDL 程式模組實作.....	44
5.2 user_logic VHDL 程式模組.....	46
5.2.1 user_logic 模組實作.....	46
5.2.2 軟體與硬體溝通介面.....	49
第六章 實驗與討論.....	52
6.1 系統驗證與實驗結果.....	52
6.2 結果討論與未來展望.....	54

圖目錄

圖 2-1 兩種投影方式示意圖。(a)平行投影，(b)扇形投影	4
圖 2-2 平行投影與投影資料關係圖	5
圖 2-3 傅立葉切片理論建立起投影資料與物體截面圖頻率域中射線的關係	8
圖 2-4 待測物體截面圖二維傅立葉頻率域	8
圖 2-5 這圖顯示從投影資料中得到的頻率域情形。(a)是理想狀況，(b)是經過傅立葉切片理論後	9
圖 2-6 背投影過程示意圖	12
圖 2-7 (a)是sinogram影像的示意圖，(b)是 256x256 pixels灰階的範例圖，(c)是(b)的sinogram	13
圖 2-8 FBP演算法程式流程	14
圖 2-9 頻率域中的濾波函數	16
圖 3-1 嵌入式系統SoC設計流程圖	18
圖 3-2 典型軟硬體設計流程	19
圖 3-3 Xilinx公司的ML310 Development Board	20
圖 3-4 以PowerPC 405 為核心的嵌入式系統架構	21
圖 4-1 EDK設計流程概觀	23
圖 4-2 基礎嵌入式系統架構圖	26
圖 4-3 FBP演算法嵌入式系統架構圖	28
圖 4-4 IPIF是IP模組與PLB bus之間的介面	29
圖 4-5 FBP元件與PLB bus間的模組示意圖	33
圖 4-6 使用者建立週邊元件並加入系統的過程	33
圖 5-1 FBP元件內的模組階層	34
圖 5-2 Core Schematic Symbol	36
圖 5-3 No-Read-on-Write Mode Waveform	37
圖 5-4 Core Schematic Symbol	39
圖 5-5 IP開始輸入資料的波形圖	43
圖 5-6 IP輸出資料的波形圖	43
圖 5-7 Core Schematic Symbol	44
圖 5-8 fbp_locate模組中的有限狀態機示意圖	45
圖 5-9 Core Schematic Symbol	46
圖 5-10 user_logic模組的有限狀態機示意圖	47
圖 5-11 user_logic模組功能方塊圖	48
圖 6-1 (a)是原始影像，(b)是圖(a)的sinogram影像，共投影 128 次，每次投影偵測 128 筆資料。	54

第一章 緒論

我們常說「眼見為憑 (To see is to believe.)」。從我們身體感官系統所直接感受到的訊息是最為直接，最容易讓我們信賴，視覺是最佳實例。映入眼簾的不只是一張圖片或影像而已，隨著時間演進，我們大腦可以從圖片或影像的動態變遷，串接成連續的資訊，讓我們了解身體外在的環境，其生動與變化遠遠勝過語言或文字描述。所以在影像處理 這門學問中有句名言：「一張圖片勝過千言萬語 (A picture is more than a thousand words.)」。

在醫學診斷方面也是如此，回想我們生病上醫院看病時的過程，若用工程觀點來分析醫師對我們的診斷治療過程，那醫師對病情資料的蒐集是首要的步驟。而為了顯現人體內部的病變，並且希望能夠以非侵入方式透視人體內部結構時，往往就需要仰賴透視人體的利器——電腦斷層掃描。

1.1 研究背景

電腦斷層掃描(Computer Tomography, CT)在一九七二年由 Godfrey Newbold Hounsfield 發明問市以來，在醫學臨床診斷上一直扮演著舉足輕重的角色，雖然 X 光有輻射危險之虞，但是此系統可以讓醫護人員以非侵入方式，透視人體各部分器官的形態變化，對於病變的診斷有很大助益，突破傳統解剖等醫療技術。

這項劃時代重大發明讓 Hounsfield 與建立電腦斷層掃描系統的線積分基礎的物理學家 Alan Cormack，共同分享一九七九年諾貝爾獎。不過其中有關線積分數學理論基礎，在一九一七年就被奧地利數學家 Johann Radon 推演過，因此

電腦斷層掃描技術中的物質密度函數沿著直線的線積分也被稱之為雷登轉換(Radon Transform)。以雷登轉換為基礎，利用傅立葉理論在頻率空間尋求影像重組的解決方法。而電腦斷層掃描中一般重建影像是利用最基本的反向投影法(Back Projection)配合影像重建濾波器(Filter)重組影像，稱為濾波反投影(Filtered Back Projection, FBP)是一種快速有效的影像重建演算法。

隨著時代的進步，電腦斷層掃描的發展相當的成熟且應用領域越來越廣，除了醫學外，也應用在天文學及一些非破壞性的偵測上例如機場，港口等因為安全因素對來往的旅客或是貨物進行偵測，或是應用在生物、農業上對動植物生長作觀察。然而，醫用的電腦斷層掃描儀器由於體積龐大不易搬運，再加上掃描方式不適當、造價昂貴等因素因此無法有效的普及化。

但是隨著半導體技術的快速發展及電路設計軟體的進步，目前的積體電路(Integrated Circuit, IC)設計主流——系統單晶片(System on Chip, SoC)，可以整合多個完整功能的電路，將構成整個系統的模組完整的放入晶片中，而不需要多顆晶片就能擁有完整的系統功能。這樣可以大大的減低以晶片做核心的儀器生產成本，也可以縮小儀器體積。

在此論文中，我們以 Xilinx 公司的 ML310 Development Board 為發展平台搭配 Xilinx 公司的 Embedded Development Kit 軟體設計一個可以重建 128x128 pixels 灰階影像的嵌入式系統 SoC 晶片雛形。希望藉由 SoC 上成本及體積的優勢，使電腦斷層掃描系統能變成攜帶容易，能掃描任意物體，且造價低廉的非醫用電腦斷層掃描系統，使得這項非破壞性檢驗技術能更廣泛地應用在各層面。

1.2 論文架構

我們論文共分為六章。第一章簡單說明電腦斷層掃描的發展，及我們的研究

動機與目的。第二章介紹影像重建理論與我們使用的演算法—濾波反投影演算法。第三章介紹系統單晶片設計與發展平台。第四章與第五章介紹我們實作濾波反投影演算法嵌入式系統 SoC 離型的過程。第六章討論我們的實作成果與未來可以改進的地方。



第二章 影像重組理論

電腦斷層掃描(Computer Tomography, CT)利用 x-ray、超音波或是放射線同位素等對待測物體進行不同角度的投影(Projection)，接著在投影的反面利用一連串的探測器對透射出的 x-ray 等能量進行衰減量的偵測。這些資料我們稱為投影資料(Projection Data)。透過一些影像重建演算法，我們可以利用這些不同投影角度得到的投影資料重建出待測物體的截面圖。

對物體進行投影而收集投影資料的方式約有兩種，(1)平行投影(parallel projection)、(2)扇形投影(fan beam projection)。示意圖如下

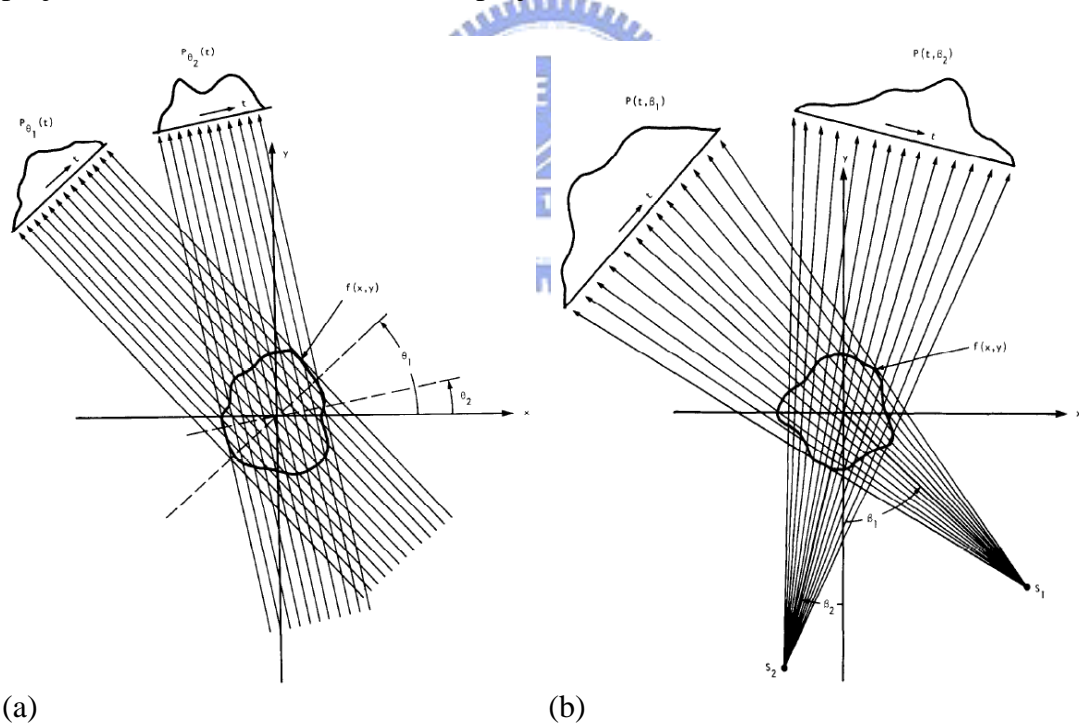


圖 2-1 兩種投影方式示意圖。(a)平行投影，(b)扇形投影

而下面我們要介紹的演算法是以平行投影為基礎的影像重建演算法。首先，我們會先介紹電腦斷層攝影的基本概念，最後將介紹實際使用的演算法，濾波反投影(filtered back projection, FBP)演算法。

2.1 影像重建演算法

影像重建演算法的主要問題在於如何利用投影資料來尋找物體截面圖上各個點的線性衰減係數。例如我們以 x-ray 對生物組織進行投影，則我們可以在投影面的另一邊偵測到 x-ray 的衰減量。在這裡我們考慮物體的每個點有不同的 x-ray 衰減係數(物質密度不同)，而每個偵測器得到的衰減量，就是對應的 x-ray 射線透射物體時直線經過的點的衰減係數總和(類比物質密度總和)，相當於是 x-ray 射線經過物體的那條路徑的線性積分(line integral)。如下圖所示

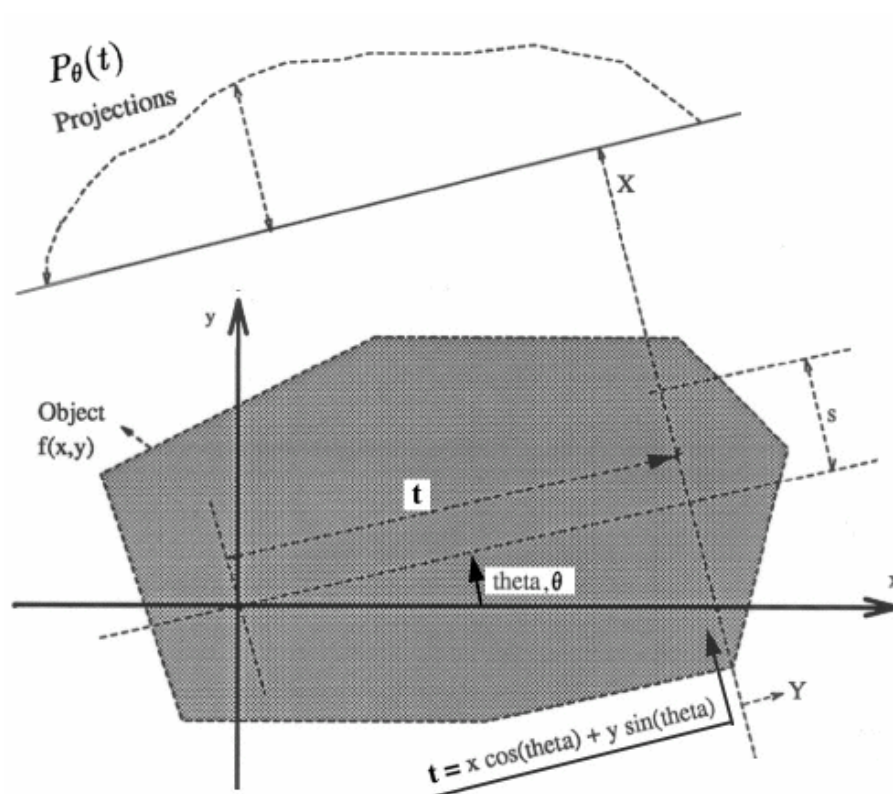


圖 2-2 平行投影與投影資料關係圖

由上圖可知我們以二維函數 $f(x, y)$ 來描述物體，加上 (θ, t) 參數，我們以線性積分 $P_\theta(t)$ 來描述 (θ, t) 參數對應的 x-ray 射線對物體作投影而得到衰減係數總和。

$$P_\theta(t) = \int_{(\theta, t)} f(x, y) ds \quad (2.1)$$

使用 delta function

$$\delta(n) = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases} \quad (2.2)$$

公式(2.1)可以改寫如下

$$P_{\theta}(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - t) dx dy \quad (2.3)$$

公式(2.3)就是函數 $f(x, y)$ 的雷登轉換(Radon Transform)。集合所有同一個角度 θ 的 $P_{\theta}(t)$ 值就成為角度 θ 的投影資料。

後面將介紹到的濾波反投影演算法是利用傅立葉理論(Fourier Theory)與投影資料來達到前述問題的封閉解(closed form solution)繼而重組出截面圖。連結傅立葉轉換(Fourier Transform)到待測物體的截面圖之間的基礎理論就是傅立葉切片理論(Fourier Slice Theorem)。

2.1.1 Fourier Slice Theorem

連結傅立葉理論到投影資料與待測物體截面圖之間的基礎理論是由 Bracewell[4]、Ramachandran和Lakshminarayana[2],[3]發展，接下來的證明則是由Kak和Slaney[1]推導的結果。二維傅立葉轉換定義如下：

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy \quad (2.4)$$

其中 u, v 是 *cycles/unit length*。

接著我們定義角度為 θ 的投影資料及其傅立葉轉換

$$S_{\theta}(\omega) = \int_{-\infty}^{\infty} P_{\theta}(t) e^{-j2\pi\omega t} dt \quad (2.5)$$

其中 ω 是 *radians/unit length*。

為了導出 Fourier Slice Theorem，我們需要定義新的座標系統 (t, s) ， (t, s) 座標是原有的座標系統 (x, y) 順時針旋轉 θ 角而來。

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.6)$$

參考圖 2-2，圖中的投影資料 $P_\theta(t)$ 是相對於 (x, y) 座標從 0 度以逆時針旋轉 θ 角對物體進行投影而得到的資料。若我們考慮新的座標系統 (t, s) ，則 $P_\theta(t)$ 是相對於 (t, s) 座標系統以 $\theta = 0$ 角度對物體進行投影得到的資料。此時在 (t, s) 座標系統中，考慮公式

$$P_\theta(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - t) dx dy \quad (2.3)$$

及 $\theta = 0$ ，沿著常數 t 定義出來的投影可以寫成

$$P_\theta(t) = \int_{-\infty}^{\infty} f(t, s) ds \quad (2.7)$$

將(2.7)代入(2.5)中我們可以得到

$$S_\theta(\omega) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(t, s) ds \right] e^{-j2\pi\omega t} dt \quad (2.8)$$

將整理公式(2.8)，以公式(2.6)將 (t, s) 座標系統轉回 (x, y) 座標系統，則我們可以得到下列公式

$$S_\theta(\omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi\omega(x \cos \theta + y \sin \theta)} dx dy \quad (2.9)$$

公式(2.9)的右手邊就是 2 維傅立葉轉換的定義，因此

$$S_\theta(\omega) = F(u, v) \quad (2.10)$$

從公式(2.10)可知道，在 θ 角度對待測物體截面圖作投影得到的投影資料，經過一維傅立葉轉換後相當於待測物體截面圖的二維傅立葉頻率域(frequency domain)中的一條通過低頻中心點的射線。下圖可以清楚的表示公式(2.10)的意義

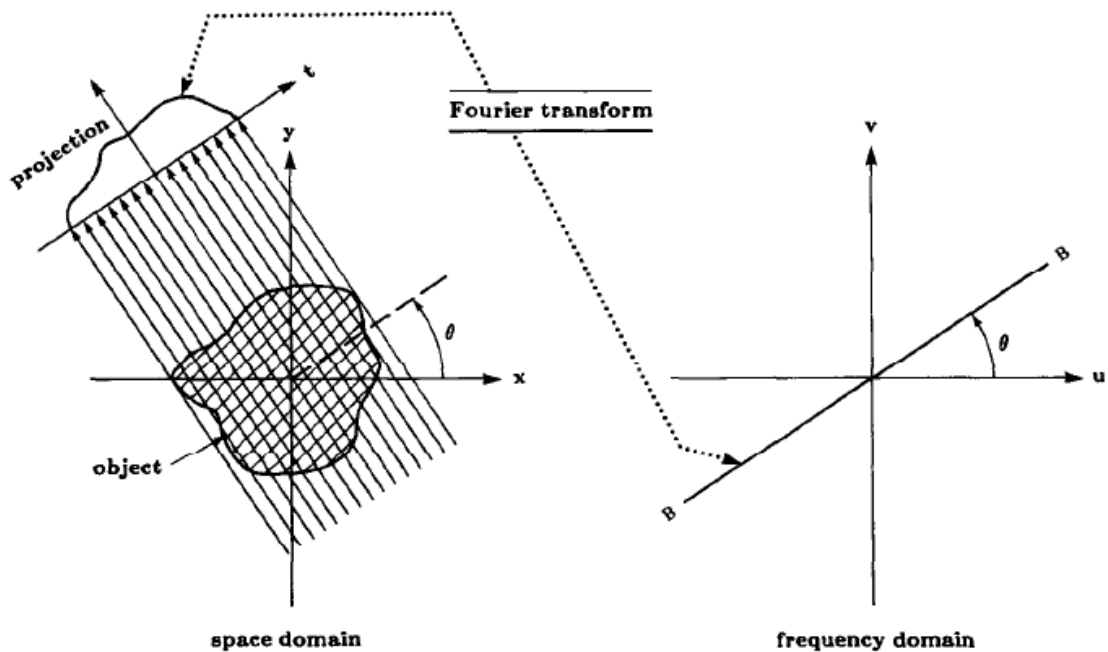


圖 2-3 傅立葉切片理論建立起投影資料與物體截面圖頻率域中射線的關係

有了上述結果，我們可以在 $\theta_1, \theta_2, \dots, \theta_k$ 角度對物體作投影，得到投影資料後經過一維傅立葉轉換對應到截面圖的二維傅立葉頻率域中。如下圖所示：

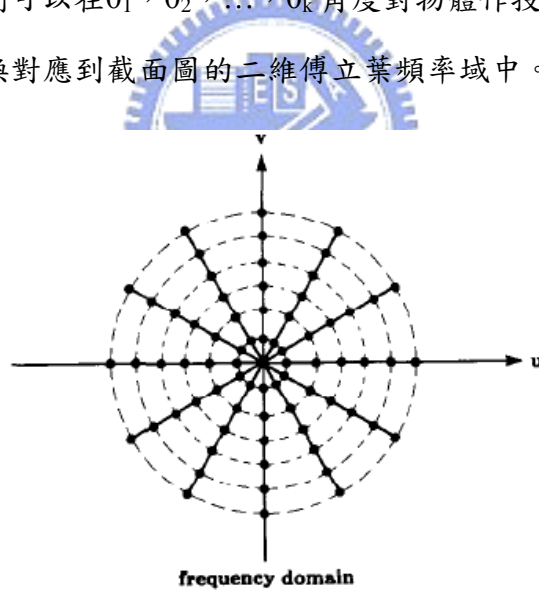


圖 2-4 待測物體截面圖二維傅立葉頻率域

假設我們可以作無限個不同角度的投影，那我們得到的資料經過一維傅立葉轉換後，可以填滿截面圖的頻率域，接著我們只要再作反傅立葉轉換(inverse Fourier Transform)就可以得到我們所希望的截面圖。

可是實際上，我們只能得到有限個不同投影角度的投影資料，且由圖 2-4可知，從投影資料經過一維傅立葉轉換對應到頻率域中的資料點是中心點(低

頻)密集，外圍(高頻)離中心點越遠越稀疏，因此頻率域沒有對應到的資料點就需要從已知資料點作內插法來當成預設資料。可是這其中會有相當大的誤差，尤其是越高頻的地方誤差越大，造成重建回來的截面圖會有影像剝蝕(image degradation)的情形。因此上面提到的是電腦斷層掃描重建影像概念上的模式，實作上我們得用不同的方式來完成。

2.1.2 濾波反投影演算法

這邊我們要使用來完成影像重建的演算法是濾波反投影(Filtered Back Projection, FBP⁽¹⁾)演算法。這演算法也是由 Fourier Slice Theorem 衍生而來，是藉由公式原理的改寫來達成不同的運算實作。這邊我們先了解濾波反投影演算法的想法。

由前一小節我們可以知道角度 θ 的投影資料作一維傅立葉轉換可對應到物體截面圖二維傅立葉頻率域中的一條射線，這相當於截面圖二維傅立葉頻率域乘於一個簡單的濾波器(filter)而得到如下圖(b)所示的情形。

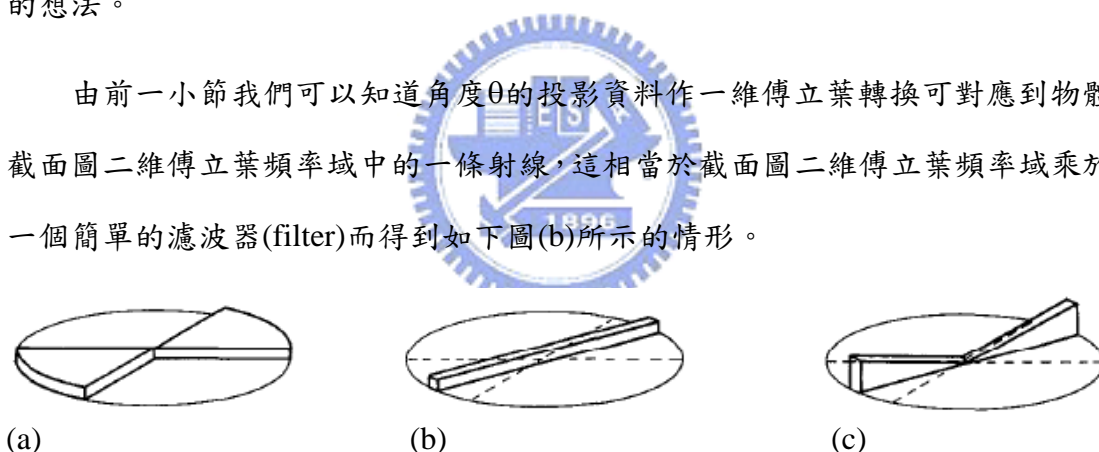


圖 2-5 這圖顯示從投影資料中得到的頻率域情形。(a)是理想狀況，(b)是經過傅立葉切片理論後得到的狀況，(c)是(b)圖經過濾波器加權後到的狀況，(c)圖中的資料狀況近似於(a)

但是對影像重建而言，我們希望得到的是投影資料的一維傅立葉轉換可以跟上圖(a)的派外型射線是相當的，這樣我們集合幾個這樣的射線就可以重建出高品質的影像。對於這想法，採用最接近的方式就是將投影資料的一維傅立葉轉換—— $S_{\theta}(\omega)$ ，乘以濾波器——圖 2-5(a)中的楔形物的寬度。假設在 180° 內共執行K次不同角度的投影，則在頻率 ω 下，楔形物的寬度是 $2\pi|\omega|/K$ 。 $S_{\theta}(\omega)$ 在經過 $2\pi|\omega|/K$ 的加權後，我們可以得到圖 2-5(c)的圖形，這跟圖 2-5(a)來比較，具有相近的”

質量”。在足夠的投影次數下，集合圖 2-5(c)的射線，可以重建出接近圖 2-5(a)所重建的影像。而濾波反投影演算法，聞其名可以知道，演算法是分為濾波(filtering)部分： $S_{\theta}(\omega)$ 在頻率域乘以濾波器作加權。及反投影(backprojection)部分：將加權後的 $S_{\theta}(\omega)$ 值重組回影像。底下我們要推導濾波反投影演算法的數學公式。

我們同樣以二維函數 $f(x, y)$ 來描述待測物體， $f(x, y)$ 的反傅立葉轉換(inverse Fourier Transform)定義如下：

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} dudv \quad (2.11)$$

將頻率域的標準座標系統轉成極座標系統(polar coordinate system)，如下

$$u = \omega \cos \theta, v = \omega \sin \theta \quad (2.12)$$

微分細節如下



$$dudv = \omega d\omega d\theta \quad (2.13)$$

公式(2.11)可以改寫如下

$$f(x, y) = \int_0^{2\pi} \int_0^{\infty} F(\omega, \theta) e^{j2\pi\omega(x\cos\theta+y\sin\theta)} \omega d\omega d\theta \quad (2.14)$$

我們將積分(integral)分為 0° 到 180° 及 180° 到 360° 兩部分，則

$$\begin{aligned} f(x, y) &= \int_0^{\pi} \int_0^{\infty} F(\omega, \theta) e^{j2\pi\omega(x\cos\theta+y\sin\theta)} \omega d\omega d\theta \\ &+ \int_0^{\pi} \int_0^{\infty} F(\omega, \theta + \pi) e^{j2\pi\omega[x\cos(\theta+\pi)+y\sin(\theta+\pi)]} \omega d\omega d\theta \end{aligned} \quad (2.15)$$

利用傅立葉原理

$$F(\omega, \theta + \pi) = F(-\omega, \theta) \quad (2.16)$$

可以簡化公式(2.15)如下

$$f(x, y) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} F(\omega, \theta) |\omega| e^{j2\pi\omega x} d\omega \right] d\theta \quad (2.17)$$

其中
$$t = x \cos \theta + y \sin \theta \quad (2.18)$$

將公式(2.17)中括號的二維傅立葉轉換 $F(\omega, \theta)$ 以投影資料的一維傅立葉轉換 $S_\theta(\omega)$ 來代替，我們得到


$$f(x, y) = \int_0^\pi \left[\int_{-\infty}^\infty S_\theta(\omega) |\omega| e^{j2\pi\omega t} d\omega \right] d\theta \quad (2.19)$$

整理公式(2.19)中的積分式子

$$f(x, y) = \int_0^\pi Q_\theta(x \cos \theta + y \sin \theta) d\theta \quad (2.20)$$

其中

$$Q_\theta(t) = \int_{-\infty}^\infty S_\theta(\omega) |\omega| e^{j2\pi\omega t} d\omega \quad (2.21)$$

其中公式(2.21)描述一個濾波的動作。在頻率域中使用 $|\omega|$ 當濾波器對 $S_\theta(\omega)$ 進行加權，所以公式(2.21)稱為濾波投影(filtered projection)。而公式(2.20)則對每個 $Q_\theta(t)$ 進行背投影(back projection)的動作。對於重建影像上的每個點 (x, y) ，在角度 θ 下可以找到對應的值 $t = x \cos \theta + y \sin \theta$ 。如  圖 2-6，由 θ_i

角可以找到對應的 $Q_{\theta_i}(t)$ ，在 $Q_{\theta_i}(t)$ 距離投影資料中心點 t_j 置，可以找到 LM 直線上所有 (x, y) 點累加的衰減係數總和 $Q_{\theta_i}(t_j)$ 值， $Q_{\theta_i}(t_j)$ 值可以平均分布到 LM 直線上的所有 (x, y) 點。假設在 180° 內對物體的投影有 K 次，則對重建影像的點 (x, y) 而言，每個點可以累加 K 次對應的 $Q_{\theta_i}(t_j)$ 值，計算出每個點的線性衰減係數，而重建出影像。公式(2.20)及(2.21)就是從投影資料到整個影像重建完成的計算過程。底下我們將介紹在個人電腦上如何以軟體實作此演算法。

備註 1：後面的章節中，濾波反投影演算法將以 FBP 演算法來簡稱。

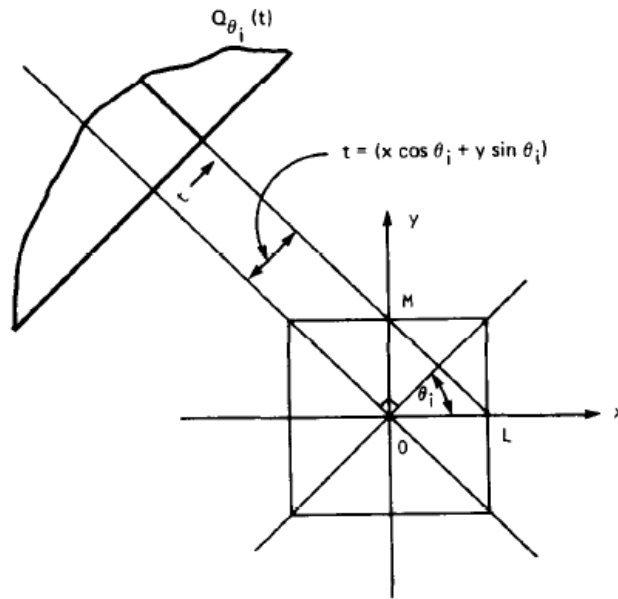


圖 2-6 背投影過程示意圖

2.1.3 濾波反投影演算法實作

FBP 演算法由前一小節可以知道，共分為濾波投影及備投影兩部分。根據

$$\text{Filtered Projection} \quad f(x, y) = \int_0^\pi Q_\theta(x \cos \theta + y \sin \theta) d\theta \quad (2.20)$$

$$\text{Back Projection} \quad Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(\omega) |\omega| e^{j2\pi\omega t} d\omega \quad (2.21)$$

兩程式，假設 0° 到 180° 之間對待測物體共作 K 次不同角度的投影，所以演算法簡單的流程如下：

I. 濾波投影

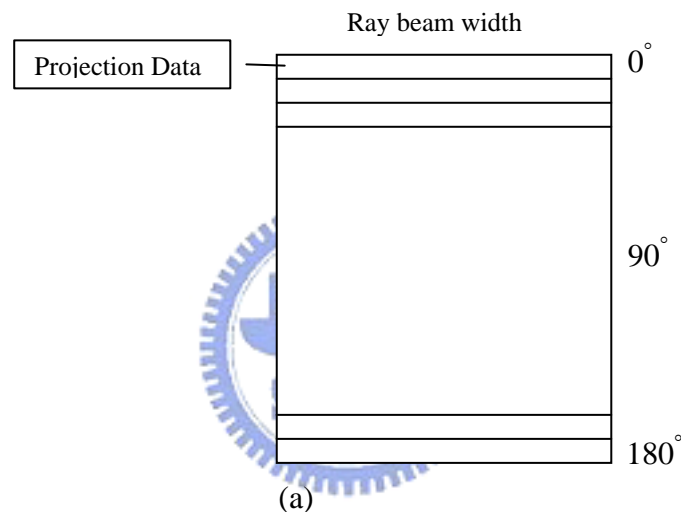
取得 θ_i 的投影資料 $P_{\theta_i}(t)$ ， i 從 0 到 $K-1$ ，執行下列工作共 K 次。

- i. 將 $P_{\theta_i}(t)$ 作傅立葉轉換到頻率域，成為 $S_{\theta_i}(\omega)$ 。
- ii. 對 $S_{\theta_i}(\omega)$ 乘與濾波器 $|\omega|$ 在頻率域作加權。
- iii. 將加權後的 $S_{\theta_i}(\omega)$ 值作反傅立葉轉換回空間域(space domain)，成為 $Q_{\theta_i}(t)$ 。

II. 背投影

- i. 從點 (x, y) 及角度 θ_i 算出 $t_j = x \cos \theta_i + y \sin \theta_i$ ， i 從 0 到 $K-1$ ，共 K 次。
- ii. 累加每個 $Q_{\theta_i}(t)$ 中，位置 t_j 上得到的 $Q_{\theta_i}(t_j)$ 值，共累加 K 次，得到 (x, y) 點原本的線性衰減係數。

在實作上，電腦斷層掃描取得的投影資料會排列成一張影像，我們稱為 sinogram。sinogram 的排列，是按照 0° 到 180° 依序將投影資料以一系列一列的方式擺列成一張影像，其中若平行投影光束有 256 束，則每列就有 256 筆偵測器得到的偵測資料。sinogram 示意圖如圖 2-7(a) 所示：



(b)

(c)

圖 2-7 (a) 是 sinogram 影像的示意圖，(b) 是 256x256 pixels 灰階的範例圖，(c) 是 (b) 的 sinogram 影像。

圖 2-7(b) 是 256x256 的灰階影像，其中以座標(100,80)為中心點有個 3x3 pixels

的小正方形。而圖 2-7(c)就是對圖 2-7(b)進行 256 次投影，每次投影偵測到 256 筆偵測資料而得到的sinogram影像，因此sinogram影像也是 256x256 pixels大小，在此我們重建影像要求得的pixel點衰減係數就是pixel的灰階值。

我們參考前面的提到的 FBP 演算法流程，先在個人電腦上以 C Language 實作此演算法，以驗證 FBP 演算法的正確性。根據演算法流程，FBP 演算法程式設計流程圖如下所示：

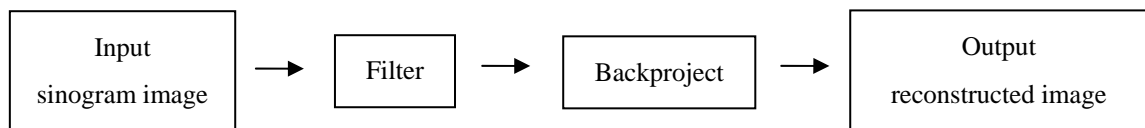


圖 2-8 FBP 演算法程式流程

所以程式上的內容主要也是分成四個部分，以底下的 pseudo code 程式來表示我們軟體的實作內容：

表 2-1 FBP 演算法的 pseudo code

```

#define angle=128; /* 旋轉 128 個角度 */
#define rays=128; /* rays beam 與 image 的 raw pixel 個數一樣 */
const int x_max = rays;
const int y_max = rays;
const int x_cen = x_max/2; /* x_cen 表示 sinogram 影像 x 軸的中心位置 */
const int y_cen = y_max/2; /* y_cen 表示 sinogram 影像 y 軸的中心位置 */
double sonogram[angles][ rays]; /* 儲存原始 sinogram 影像的陣列 */
double filtered_rel[angles][rays]; /* 儲存完成 filter 後的 singogra 影像的陣列 */
double reconstructed[rays][ rays]; /* 儲存重建完成影像的陣列 */
main()
begin
/*****從 sinogram 影像中讀取投影資料*****/
    read sinogram image to sinogram[angles][rays];

/*****底下是filtered projection的pseudo code *****/
    /* 作 1D FFT 及 IFFT 之前要作 2 幕次方的 ZERO Padding */
    int logn = int(ceil(log10(rays)/log10(2))+1);
    int fftsize = int(pow(2,logn));
  
```

```
/* 定義FFT及IFFT中存實部，虛部的array */
```

```
double *rel = new double [fftsize+1];
```

```
double *img = new double [fftsize+1];
```

```
/* 定義 window值 */
```

```
double *window = new double [fftsize+1];
```

```
for(i=1; i<fftsize/2; i++)
```

```
    window[i+1] = window[fftsize-i+1] = i;
```

```
window[0] = window[1] = 1;
```

```
window[fftsize/2+1] = fftsize/2;
```

```
for( a=0 ; a<angle ; a++ ) loop
```

```
    memset(rel,0,sizeof(double)*(fftsize+1));
```

```
    memset(img,0,sizeof(double)*(fftsize+1));
```

```
    memcpy(rel, sinogram[a],sizeof(double)*rays);
```

```
    FFT(rel, img);
```

```
/* 做完 FFT 後與 windows 相乘 */
```

```
for(i=1;i<fftsize+1;i++) loop
```

```
    rel[i]*=(window[i]/(double)fftsize*2);
```

```
    img[i]*=(window[i]/(double)fftsize*2);
```

```
end loop;
```

```
IFFT(rel, img);
```

```
    memcpy(filtered_rel[a],rel,sizeof(double)*rays);
```

```
end loop;
```

```
/****** 底下是back projection的pseudo code *****/
```

```
/* the mid-point是投影光束的中心位置 */
```

```
int midpoint = rays/2;
```

```
for(y=0; y<y_max; y++) loop
```

```
    for(x=0; x<x_max; x++) loop
```

```
        double sum = 0;
```

```
        for(int i=0; i<angles; i++) loop
```

```
            theta = i*PI/angles;
```

```
            t =cos(theta)*((x-x_cen)/(double)(x_cen))
```

```
                +sin(theta)*((y-y_cen)/(double)(y_cen));
```

```

/* mid_bin表示當角度為theta時，重建影像(x,y)pixel對應於 $Q_\theta(t)$ 的位置*/
mid_bin = (t* mid_point)+ mid_point;
/* mid-bin可能在 $Q_\theta(t)$ 的整數位置low-bin跟high-bin之間，所以要求
mid-bin在low-bin與high-bin之間的位置比例，方便以後作內插法 */
lb = (int)mid_bin;
hb = lb + (mid_bin>lb? 1:0);
frac = mid_bin - lb;
/* mid_bin可以能在low_bin跟high_bin中間，所以作gray level要作內插法
計算 */
sum += (1-frac)*filtered_rel[i][lb] + frac*filtered_rel[i][hb];

end loop;
reconstructed [y][x] = sum;
end loop;
end loop;
/*****將重建好的截面圖pixel灰階值寫回重建影像中*****/
write reconstructed[rays][ rays] to reconstructed imgae;

end;

```

上面的pseudo code內容是針對 128x128 大小的sinogram影像(0° 到 180° 共 128 個投影角度，每次投影偵測有 128 筆偵測資料)可以重建出截面圖為 128x128 pixels大小的灰階影像。pseudo code中，在濾波投影(filtered projection)過程時，傅立葉轉換我們以快速傅立葉轉換(Fast Fourier Transform, FFT)來完成，且執行快速傅立葉轉換前，取樣資料需先執行Zero Padding擴大成 2 的冪次方。而在頻率域中需使用的濾波器 $|w|$ 函數，則如圖 2-9所示，將投影資料的一維傅立葉值 $S_\theta(\omega)$ ，乘與一個權重，低頻中心點權重值低，越往高頻則權重越高。

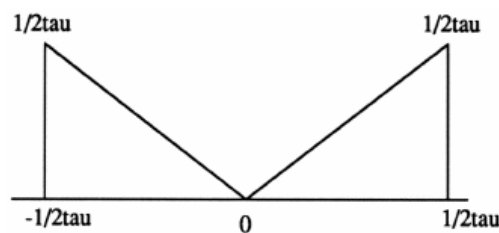



圖 2-9 頻率域中的濾波函數

第三章 系統單晶片設計與發展平台

系統單晶片(System on Chip, SoC)解決方案被譽為半導體業最重要的發展之一，目前，從數位手機和數位電視等消費類電子產品到高階通訊 LAN/WAN 設備中，這一元件隨處可見。過去，為了建立此類嵌入式系統，設計工程師不得不在處理器、邏輯單元和記憶體等三種硬體中進行選擇，而現在這些元件已合併為單一的 SoC 解決方案。

3.1 系統單晶片設計



硬體製程技術提高，晶片中可容納的電晶體數量不斷提昇。在考慮晶片功耗，面積及成本上，我們會希望將多個功能完整的電路整合成一顆積體電路，就是系統單晶片(SoC)。可整合到系統單晶片的電路包括中央處理器，記憶體，溝通介面，應用電路，數位訊號處理器(Digital Signal Processor, DSP)等。系統單晶片最直接的想法，就是將構成系統的所有電路模組完整的放入單晶片中，所以此類的系統單晶片也具有處理器，記憶體及應用電路，可以發展嵌入式系統的功能。

嵌入式系統SoC可採用現場可程式閘陣列(Field Programmable Gate Array, FPGA)或專用積體電路(Application Specific Integrated Circuit, ASIC)實現。傳統設計的方法，如圖 3-1(a)所示，在設計的早期就將軟硬體明顯分離獨立設計，最後階段才將軟硬體整合。由於軟體設計者不熟悉硬體的架構，硬體設計者不了解軟體設計的方法，若合成發生問題，常常都無法確定是軟硬設計上何種錯誤，導

致修改上的困難度提高，延遲了產品上市的時間。再者，隨著晶片容量增大，功能需求增加，嵌入式系統的複雜度也相對地提高，這種傳統設計方法，在最後邏輯合成時除了時間的延遲外，也會浪費人力和成本在系統整合上。

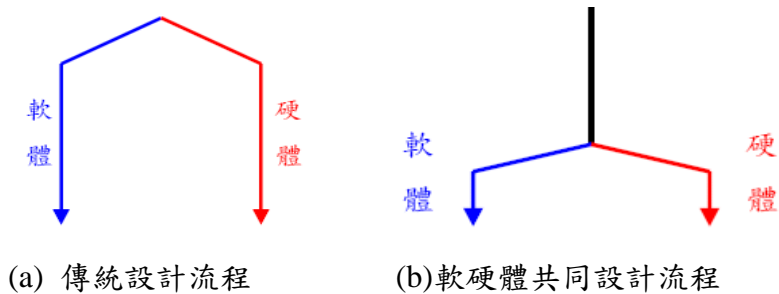


圖 3-1 嵌入式系統 SoC 設計流程圖

然而，晶片設計的生產力在這幾十年來是有目共睹，但是與晶片容量的成長，卻是有著越來越大的落差。因此，一個新的設計方法論就因應的誕生—軟硬體共同設計(Software/Hardware Co-design)。這是一個以系統觀點出發的方法論，強調軟硬體設計時的一致性與整合性，如圖 3-1(b)，對於軟體發展的驗證(Verification)更加容易，且硬體設計的錯誤也能提早發現，降低設計所需的時間與成本，大大縮短系統設計的時間，因應市場的即時性。

3.1.1 軟硬體共同設計

一般而言，典型的軟硬體共同設計流程，可以用圖 3-2 典型軟硬體設計流程來表示。

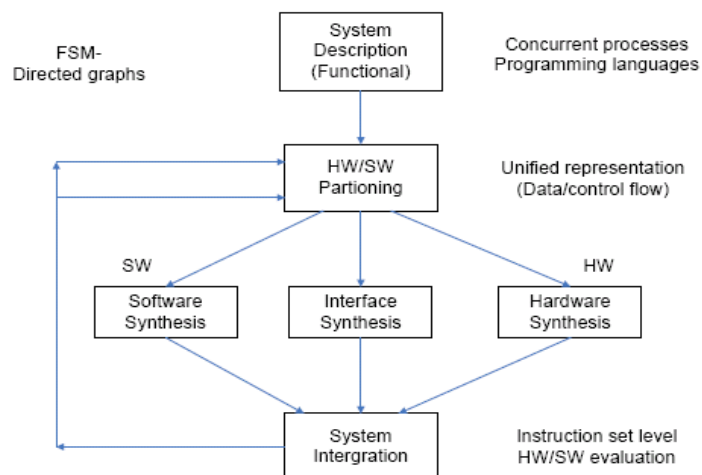


圖 3-2 典型軟硬體設計流程

軟硬體共同設計的目的，是為了軟體與硬體的協同描述、驗證與合成提供一個整合的環境，以圖 3-2 典型軟硬體設計流程可知，首先需對你所設計的系統作規格與功能上的分析，決定有哪些功能，需要哪些硬體元件及控制流程為何等。再來利用一套合適的演算法，將軟硬體部分切割，並設計一個介面供軟硬體溝通之用。最後便是將軟硬體與溝通的介面三個部份實作出來，並加以整合，做軟硬體共同的驗證(Verification)與測試(Testing)，再依設計成本、系統效能或低功率(Low power)等不同方向考量，若需要改進再重做軟硬體切割，直到最佳的效果。

在這邊我們的 FBP 演算法將採用現場可程式閘陣列(FPGA)來實現，設計流程也將採用上述的軟硬體共同設計的方式來完成我們的 FBP 演算法嵌入式系統的實作。



3.2 嵌入式系統 SoC 發展平台

在我們的實作中採用的是Xilinx公司生產的嵌入式系統發展平台—ML310 Development Board，如 圖 3-3所示：

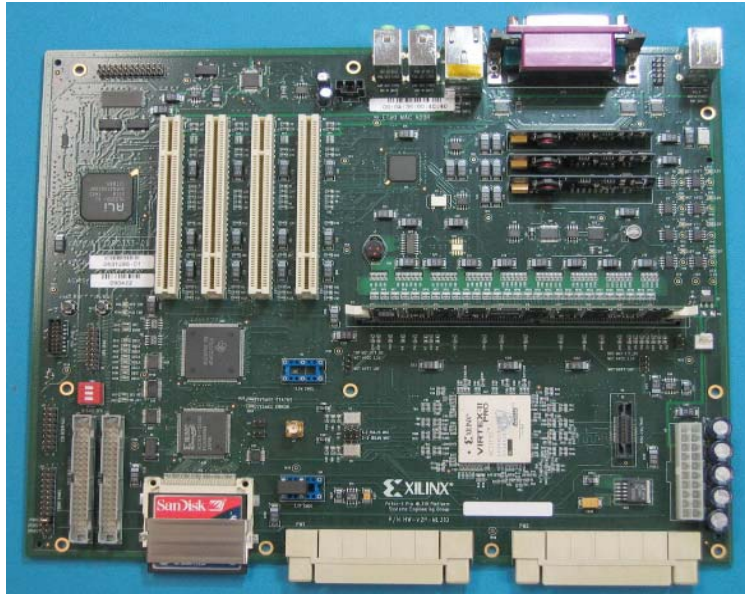


圖 3-3 Xilinx 公司的 ML310 Development Board

平台的核心是一顆 Virtex- II pro FPGA(Field Programmable Gate Array)，型號是 XC2VP30。這顆 FPGA 共有 30,816 Logic Cells⁽¹⁾提供邏輯合成的電路資源，且具有 2 顆內建的 PowerPC 405 處理器，可以當作嵌入式系統的核心處理器。此外，FPGA 內尚有內建的 18x18 bits 的乘法器 136 顆及 136 塊 18Kbits 的 Block RAM 提供現成的電路元件。

除了 FPGA 外，發展平台提供一個嵌入式系統相當完整的發展環境，除了電源與一些 I/O 介面，並利用匯流排將核心模組與邏輯模組整合在一起。平台上重要的邏輯模組像是 DDR Memory，System ACE CF Controller，CompactFlash Slot，IDE Drive Connectors，PCI Slots 等。

平台上主要的匯流排是 Process Local Bus(PLB)與 On-chip Peripheral Bus(OPB)，PLB bus 與 OPB bus 是 IBM CoreConnect 的標準匯流排，其中 PLB bus 是高速匯流排，而 OPB bus 是低速匯流排，兩個匯流排中間有個 Bridge 作連接，形成整個嵌入式系統的骨幹。

底下是是以 PowerPC 405 處理器為核心的嵌入式系統架構圖

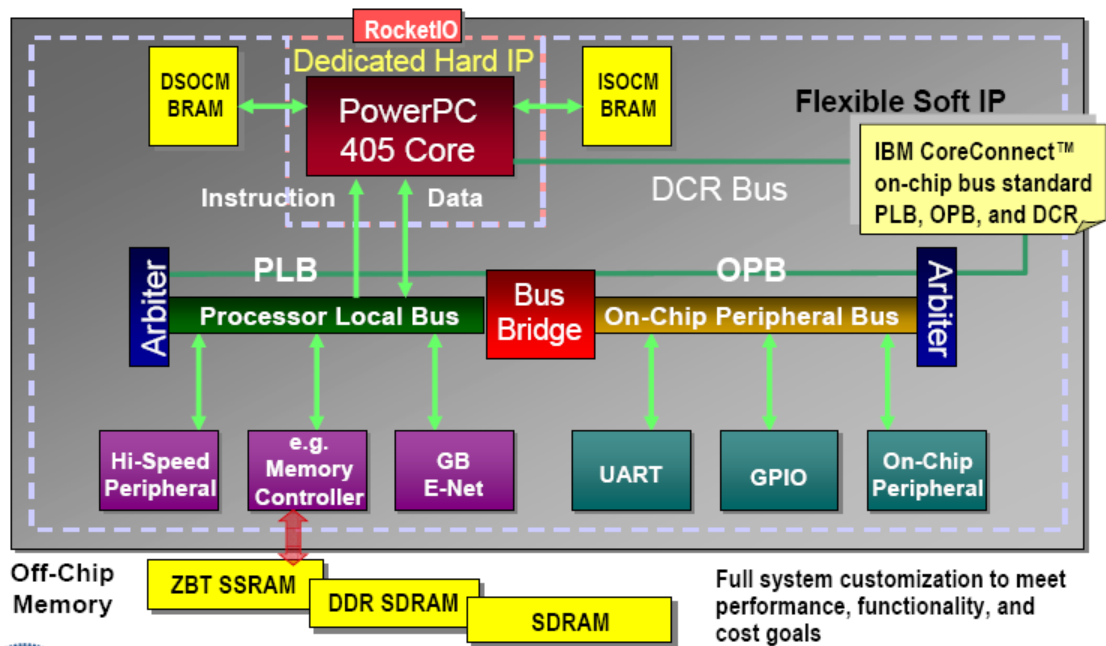


圖 3-4 以 PowerPC 405 為核心的嵌入式系統架構

我們配合 ML310 平台使用的發展工具是 Embedded Development Kit(EDK) 及 Integrated Software Environment(ISE)。EDK 提供以 PowerPC 405 為核心的嵌入式系統設計者豐富的設計工具及多樣的嵌入式系統週邊設備，如 Memory Controller、UART(Uni-versal Asynchronous Receiver/Transmitter)、GPIO(General Purpose Input/Output)等。EDK 發展工具主要包括下面幾部分：

- 嵌入式系統核心及週邊元件的硬體 IP(Intellectual Property)。
- 嵌入式系統軟體所需要的驅動程式(Drivers)，函式庫(Libraries)。
- 具有整合軟硬體研發環境的 Xilinx Platform Studio(XPS)。

其中 Xilinx Platform Studio 中有幾個重要的發展工具，在我們的 FBP 演算法嵌入式系統實作中將使用到，所以下面簡單介紹這些工具：

- The Base System Builder(BSB) wizard：這是一個軟體工具可以幫助使用者快速的建立特定平台上可以執行的基礎嵌入式系統。在執行完 BSB wizard 後，會產生兩個重要的檔案，一個是硬體規格檔(Microprocessor Hardware Specification，MHS)，另一個是軟體規格檔(Microprocessor Software

Specification, MSS)。MHS 檔案定義系統的架構及使用到的週邊元件及核心處理器。也定義了系統中週邊與核心元件間匯流排的連接關係，週邊元件在系統中的位置地圖(address map)及各個週邊元件的參數設定。MSS 檔案則是定義週邊元件的驅動程式，系統的標準輸入輸出設備，系統中斷處理常式及相關的軟體設定。

- The Platform Generator Tool(PlatGen)：輸入 MHS 檔案，建立系統的 netlist 檔案，並支援檔案下載到 FPGA 時所需的相關檔案，及新增週邊元件到系統中的相關工作。在執行 PlatGen 工具時，ISE 將自動被呼叫去完成整個硬體平台的邏輯合成，電路佈局到產生 netlist 檔案。
- The Library Generator Tool(LibGen)：輸入 MSS 檔案，建立並個人化週邊元件驅動程式、函式庫、系統中斷處理常式及檔案系統。
- The Bitstream Initializer tool：在 FPGA 中對處理器將存取的指令記憶體區塊進行初始化設定，而指令記憶體區塊則存在 Block RAM 中。這個工具將讀入 MHS 檔案，並且呼叫 ISE 的 Data2MEM 工具來初始化 FPGA 中使用到的 Block RAM。
- Create/Import Peripheral Wizard：幫助使用者建立自己的週邊元件並且加入嵌入式系統中。

ISE 是套 IC 設計的整合軟體環境，包含多個軟體套件，可以執行完整的設計流程。從硬體描述語言(Hardware Description Language)程式的編寫(如 VHDL 及 Verilog)，設計的輸入，邏輯模擬，邏輯合成，電路佈局，產生 netlist 檔案，產生可以載入 FPGA 配置電路的 BitStream 檔等工作。而 EDK 軟體中，有牽涉到 IC 設計流程的工作則是呼叫 ISE 來完成這些工作。

備註 1：Logic Cell = (1) 4-input LUT + (1)Flip-Flop + Carry Logic

第四章 FBP 演算法嵌入式系統

我們的目標是將FBP演算法在ML310 Development Board的架構下實作成嵌入式系統SoC。配合ML310 平台，我們使用EDK發展工具來完成我們的實作。而EDK軟體上的設計流程如圖 4-1所示，是軟硬體共同設計的方式。

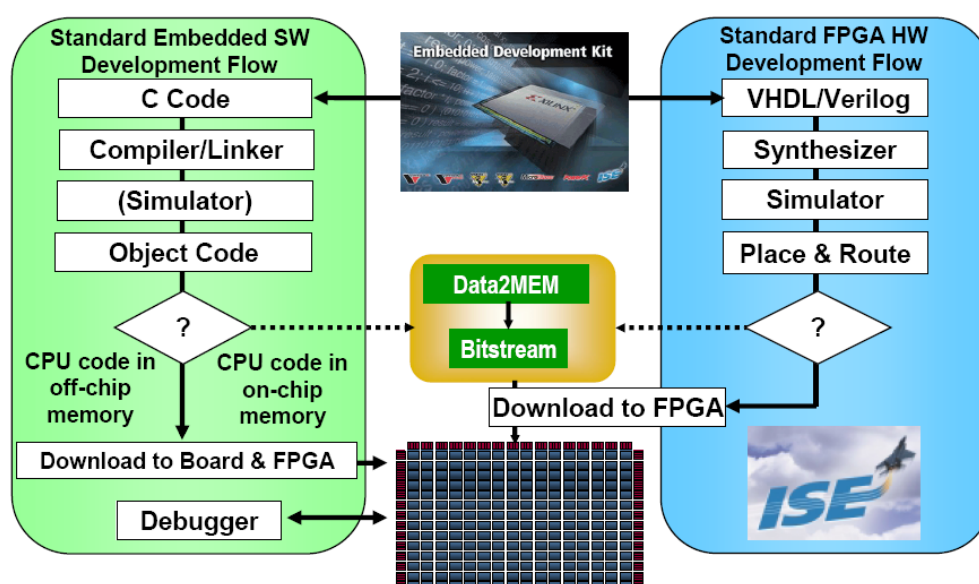


圖 4-1 EDK 設計流程概觀

由上面的流程圖可以知道，我們完全可以採用 Standard Embedded SW Development Flow，以 C Language 來完成 FBP 演算法所有功能，再與 ML310 Development Board 上的基本嵌入式系統元件如 PowerPC 405、PLB bus 及 DDR SDRAM 等硬體元件整合好，單純以 PowerPC 405 處理器執行所有運算，這樣就是一個完整的 FBP 演算法嵌入式系統了。可是這樣並不符合我們的需求，首先 PowerPC 405 處理器並不支援硬體的 float point 運算，而是以軟體模擬的方式來完成，加上 PowerPC 405 處理器的運算頻率並不高，這將使的 FBP 演算法運算的時間拉長。再且，FBP 演算法中有許多運算式是固定的且重複執行次數可預期，相當適合實作成硬體來加速整個演算法的運算。

因此我們要先分析 FBP 演算法，分析出哪些地方採用硬體實作比較好，哪些地方用軟體撰寫比較恰當。ML310 board 是一塊完整的嵌入式系統發展平台，因此我們可以依據我們的需求選擇需要使用的模組。並將 FBP 演算法中要以硬體加速的部分在 FPGA 中實作成硬體並加入嵌入式系統架構中。

4.1 嵌入式系統軟體實作

4.1.1 FBP 演算法分析與實作

假設 0° 到 180° 之間對待測物體共作 K 次不同角度的投影，則由表 2-1 FBP 演算法的 pseudo code 中，我們可以知道，程式主要分為 Filtered Projection 和 Back Projection 兩部分。Filtered Projection 的計算集中在 K 次的 FFT、 $S_o(\omega)$ 乘與 $|\omega|$ 作加權及 IFFT 運算。而 Back Projection 過程中需作一個三層 for-loop 的數學運算，以表 2-1 FBP 演算法的 pseudo code 內容為前提，我們要重建 128×128 的灰階影像，則三層 for-loop 最內層的數學運算我們要執行 $128 * 128 * K$ 次，這是相當大量的數學運算。

由於我們在個人電腦上已完成 FBP 演算法的軟體實作，所以我們暫時先在嵌入式軟體上完成 FBP 演算法。因為我們的實作平台 ML310 Development Board 不支援硬體的 float point 運算，所以程式的撰寫過程中我們以 fix point 的方式來處理帶有小數的資料，以加速 FBP 程式運算的速度。由於處理的資料是 sinogram 影像中 pixel 的灰階值，資料寬度是 8 bits。因此我們以 fix point 的方式處理資料的方法是將資料放大 $1024(2^{10})$ 倍。一方面是資料放大 1024 倍相當可以保存小數點以下三位的資料，不至於在數學運算過程中流失過多的資訊，使的重建回來的影像失真。取 2 的冪次方倍數，則是考慮以硬體實作 FBP 演算法中的數學運算

時可以使用位元移動的方式來作乘除法的動作，避免使用過多的乘除法器增加電路佈局的複雜度及所耗費的 FPGA 電路資源。

參照圖 2-8的FBP程式流程圖，其中關於Input sonogram image及Output reconstructed image的部分，我們利用ML310 Development Board的CompactFlash Slot上的 512MB CompactFlash SanDisk作為sinogram影像及完成重建後影像的儲存設備。

由於我們的 FBP 程式要讀取 SanDisk 上的 sinogram 影像及回存重建的影像，因此我們需要可以讀寫 SanDisk 的函式庫。Xilinx 公司所提供的函式庫名稱為 xilfatfs，我們只要在 EDK 的 Software Platform Settings 設定選項中勾選此函式庫，並在 FBP 程式中包含 xsysace.h 及 sysace_stdio.h 兩個標頭檔，即可順利讀寫 SanDisk。

完成FBP程式後經過圖 4-1中的Standard Embedded SW Development Flow過程可以得到一個附檔名為.elf的執行檔。附檔名為.elf的執行檔將與後面提到的系統硬體平台設定檔system.bit合併使用。



4.2 嵌入式系統硬體實作

4.2.1 嵌入式系統硬體架構

完成 FBP 程式後，我們要建立嵌入式系統的硬體架構。ML310 Development Board 是完整的嵌入式系統發展平台，因此具有多種硬體設備。在這我們的重點在於 FBP 演算法的運算，因此我們只要基本的嵌入式系統成員，如下所使示

- Processor：PowerPC 405 Core

- Bus System：Processor Local Bus 及 On-Chip Peripheral Bus
- Memory：DDR SDRAM 及 FPGA 內的 Block RAM
- I/O Device：RS232_Uart 及 SysAce_CompactFlash

其中RS232_Uart可以讓我們透過terminal程式經由serial port與系統溝通。SysAce_CompactFlash則讓我們可以透過System ACE CF Controller控制CompactFlash card，以達到讀寫的目的。系統的建立，我們可以透過3.2節中提到的EDK軟體中的功能，BSB wizard。透過此wizard從硬體選單中勾選我們需要的硬體成員，由EDK軟體幫我們建立基本的系統。其中細部設定上，RS232_Uart及SysAce_CompactFlash本身是比較慢的週邊設備，因此與OPB Bus連接，DDR SDRAM是高速的設備，因此與PLB Bus連接。

建立好的系統硬體架構圖如下：

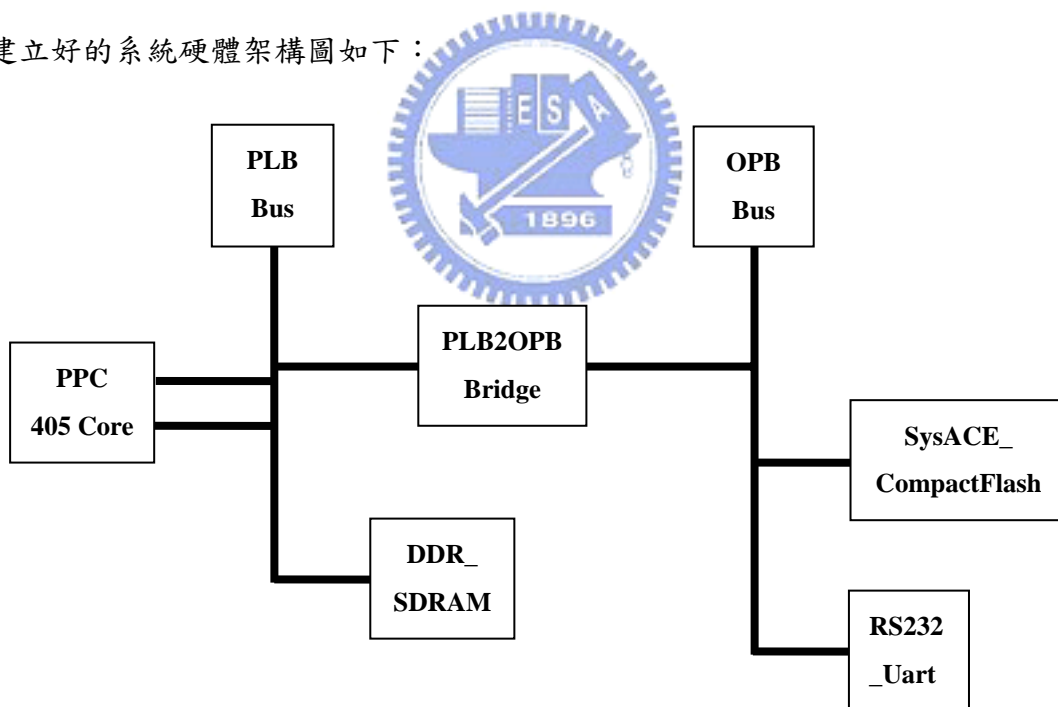


圖 4-2 基礎嵌入式系統架構圖

經過圖 4-1 中 Standard FPGA HW Development Flow 流程後會產生系統硬體平台設定檔 system.bit。將 4.1.1 節中，完成的 FBP 程式執行檔與 system.bit 合併可以產生系統設定檔 download.bit。將此 Bitstream 檔下載至 FPGA 中可以配置電路安裝整個 FBP 演算法嵌入式系統。但前面我們提到，此架構完全由 PowerPC 405 處理

器擔任FBP程式的所有運算。由於不支援硬體的float point運算且PowerPC 405 本身的運算頻率算不上快。因此我們將對FBP程式中運算負擔較重的地方實作成硬體，使用硬體來加速。

由演算法分析中我們可以知道 FBP 程式中，時間耗費最大，運算量最高的兩個地方是：


- Filtered Projection 過程中的 FFT 轉換、頻率域中的加權運算及 IFFT 轉換。
- Back Projection 過程中的 3 層 for-loop 運算。

因此我們希望以硬體加速的方式來完成上面兩個地方的運算。我們將設計一個名為 FBP 的週邊元件，此週邊元件是一個 IP 將在 FPGA 中合成此電路。此 FBP 元件有下列兩個功能

功能一：

Filtered Projection過程中從FFT開始到IFFT完成之間的所有數學運算。參考表 2-1 FBP演算法的pseudo code，內容如下：

表 4-1



```
/* Pθ(t)作傅立葉轉換成 Sθ(ω) */  
FFT(rel, img);  
  
/* Sθ(ω)與濾波器|ω|作加權運算 */  
for(i=1;i<fftsize+1;i++) loop  
    rel[i]*=(window[i]/(double)fftsize*2);  
    img[i]*=(window[i]/(double)fftsize*2);  
end loop;  
  
/* 將 Sθ(ω) |ω|作反傅立葉轉換成 Qθ(t) */  
IFFT(rel, img);
```

功能二：

Back Projection過程中 3 層 for-loop內的數學運算。參考表 2-1 FBP演算法的 pseudo code，內容如下：

表 4-2

```
/* 經由i值可以求出第i次投影的投影角度 $\theta$ (theta) */  
theta = i*PI/angles;  
t = cos(theta)*((x-x_cen)/(double)(x_cen))+sin(theta)*((y-y_cen)/(double)(y_cen));  
/* mid_bin表示當角度為theta時，重建影像(x,y)pixel對應於 $Q_0(t)$ 的位置*/  
mid_bin = (t* mid_point)+ mid_point;  
/* mid_bin可能在 $Q_0(t)$ 的整數位置low_bin跟high_bin之間，所以要求mid_bin在low_bin與  
high_bin之間的位置比例，方便以後作內插法 */  
lb = (int)mid_bin;          /* lb表示low_bin */  
hb = lb + (mid_bin>lb? 1:0); /* hb表示high_bin */  
frac = mid_bin - lb;
```

由於我們希望FBP的運算頻率越快越好，因此我們將週邊元件FBP IP與高速的PLB bus作連接。因此新的系統硬體架構圖如下：

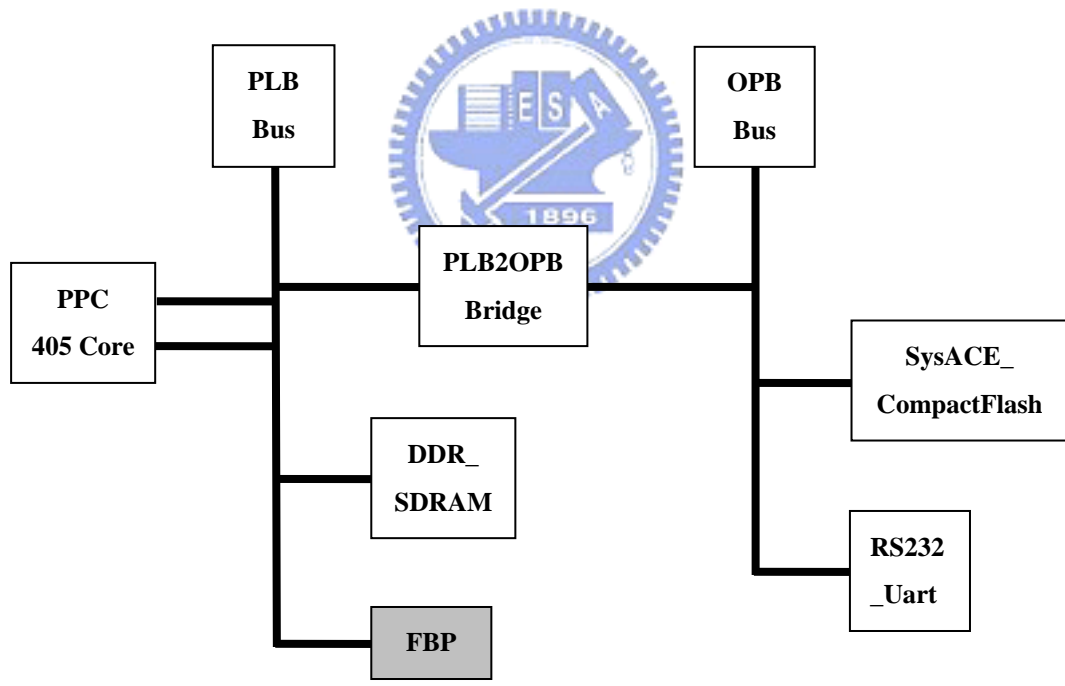


圖 4-3 FBP 演算法嵌入式系統架構圖

4.2.2 建立 FBP 週邊元件

EDK 軟體提供 Create/Import Peripheral Wizard 幫我們建立自己的週邊元件，在這我們要利用此 wizard 幫我們建立 FBP 元件並加入系統設定中。首先用 Create

Peripheral wizard 建立 FBP 元件。FBP 元件的硬體描述，目前 EDK 軟體只支援 VHDL 模組，我們將最上層電路的 VHDL 模組，取名為 fbp。接下來我們選擇將 FBP 元件連接上 PLB Bus。之後我們要決定 FBP 元件與 PLB bus 之間的溝通方式。EDK 提供 IP Interface(IPIF)框架，它可以提供使用者建立的 IP 模組與 on-chip buses 之間的溝通介面服務。由於連接的 Bus 是 PLB bus，因此我們使用的是 PLB IPIF。PLB IPIF 有兩個主要的介面，對 PLB bus 的介面及對 IP 模組的介面。

如下圖所示：

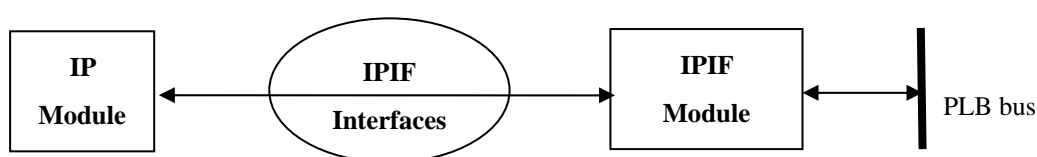


圖 4-4 IPIF 是 IP 模組與 PLB bus 之間的介面

IPIF 提供的溝通介面服務，決定使用者的元件與 Bus 間資料的溝通方式。IPIF 有下列幾項：

- S/W Reset and Module Information register(RST/MIR)
- Burst and Cacheline Transaction Support
- DMA
- FIFO
- User Logic Interrupt Support
- User Logic S/W Register Support
- User Logic Master Support
- User Logic Address Range Support

其中我們選用 S/W Reset and Module Information register(RST/MIR)及 User Logic S/W Register Support 的服務。S/W Reset and Module Information register(RST/MIR)

服務，會為了 FBP 元件定義一個唯寫的位址，當有特定的字元從軟體寫入此位址時將會單獨的將此元件重新啟動。另外會定義一個唯讀的暫存器儲存 FBP 元件的識別資訊。User Logic S/W Register Support 服務使 IPIF 模組在 fbp 模組內定義出軟體可以定址的暫存器。透過這些暫存器，我們可以從軟體將資料寫入暫存器供 FBP 元件作運算，也可以從暫存器取回計算完成的資料。在這我們定義了 32 個 64bits 的暫存器供我們使用。

IPIF 對使用者 IP 模組的介面稱為 IP Interconnect(IPIC)，它是 IPIF 與使用者 IP 模組連接的訊號集合。我們的 fbp 模組需要透過 IPIC 定義的訊號與 IPIF 連接才能夠與 PLB bus 作溝通。IPIC 定義的訊號集內容隨著我們選用 IPIF 介面服務的不同而有所不同，我們使用到的 IPIC 訊號定義如下：

表 4-3 PLB IPIF I/O Signals

Signal Name	I/O	Description
Bus2IP_Clk	O	IPIC clock provided to IP. This PLB_CLK
Bus2IP_Reset	O	Active high reset for use by the User IP.
Bus2IP_Data(0:C_IPIF_DWIDTH-1)	O	Write data bus to the User IP. Write data is accepted by the IP during a write operation by assertion of the IP2Bus_WrAck signal and the rising edge of the Bus2IP_Clk.
Bus2IP_BE(0:(C_IPIF_DWIDTH/8)-1)	O	Byte enable qualifiers for the requested read or write operation with the User IP. Bit 0 corresponds to Byte lane 0, Bit 1 to Byte lane 1, and so on.
Bus2IP_RdCE(0: see note 1)	O	Active high chip enable bus. Chip enables are assigned per the User's entries in the C_ARD_NUM_CE_ARRAY. These chip enables are asserted only during active read transaction requests with the target address space and in conjunction with the corresponding sub-address within the space.
Bus2IP_WrCE(0: see note 1)	O	Active high chip enable bus. Chip enables are assigned per the User's entries in the

		C_ARD_NUM_CE_ARRAY. These chip enables are asserted only during active write transaction requests with the target address space and in conjunction with the corresponding sub-address within the space.
Bus2IP_RdReq	O	Active high signal indicating the initiation of a read operation with the IP. It is asserted for 1 Bus2IP_Clk during single data beat transaction and remains high to completion on burst write operations.
Bus2IP_WrReq	O	Active high signal indicating the initiation of a write operation with the IP. It is asserted for 1 Bus2IP_Clk during single data beat transaction and remains high to completion on burst write operations.
IP2Bus_Data(0:C_IPIF_DWIDTH-1)	I	Input Read Data bus from the User IP. Data is qualified with the assertion of IP2Bus_RdAck signal and the rising edge of the Bus2IP_Clk.
IP2Bus_Retry	I	Active high signal indicating the User IP is requesting a retry of an active operation. This signal is currently unused by the PLB IPIF.
IP2Bus_Error	I	Active high signal indicating the User IP has encountered an error with the requested operation. This signal is asserted in conjunction with IP2Bus_RdAck or the IP2Bus_WrAck.
IP2Bus_ToutSup	I	Active high signal requesting suppression of the data phase time-out function in the IPIF for the active read or write operation.
IP2Bus_RdAck	I	Active high read data qualifier. Read data on the IP2Bus_Data Bus is deemed valid at the rising edge of Bus2IP_Clk and the assertion of the IP2Bus_RdAck signal by the User IP.
IP2Bus_WrAck	I	Active high write data qualifier. Write data on the IP2Bus_Data Bus is deemed accepted by the User IP at the rising edge of the Bus2IP_Clk and IP2Bus_WrAck asserted high by the User IP.

表 4-4 PLB IPIF Design Parameters

IPIF Bus Specification	Feature/Description	Parameter Name	Default Value
	IPIF Data Bus Width (bits)	C_IPIF_DWIDTH	64
	IPIF Address Bus Width (bits)	C_IPIF_AWIDTH	32

note1：在 IPIF 提供的 User Logic S/W Register Support 服務中，我們定義了 32 個暫存器。而 Bus2IP_RdCE 及 Bus2IP_WrCE 用來指定要讀寫的暫存器位置，所以這兩個 signal 的寬度皆是 32bits。

在 Create Peripheral wizard 建立好 FBP 元件後，此元件會有三個部分。

- Bus 介面：定義與 PLB Bus 之間的訊號埠集。包括 IPIF 對 PLB Bus 的介面及 IPIC。
- IP Interface(IPIF)元件的使用：在這我們選用 RST/MIR 與 User Logic S/W Register Support，此時 EDK 軟體會建立可以讀取暫存器的介面。
- 使用者定義的硬體元件功能：FBP 程式要實作成硬體的部份將在這實作完成。

建立好 FBP 元件後，會產生兩個重要的檔案，一個是 fbp.vhd 及 user_logic.vhd。fbp.vhd 是 FBP 元件的最上層 VHDL 模組，因此 FBP 元件的模組示意圖如下。

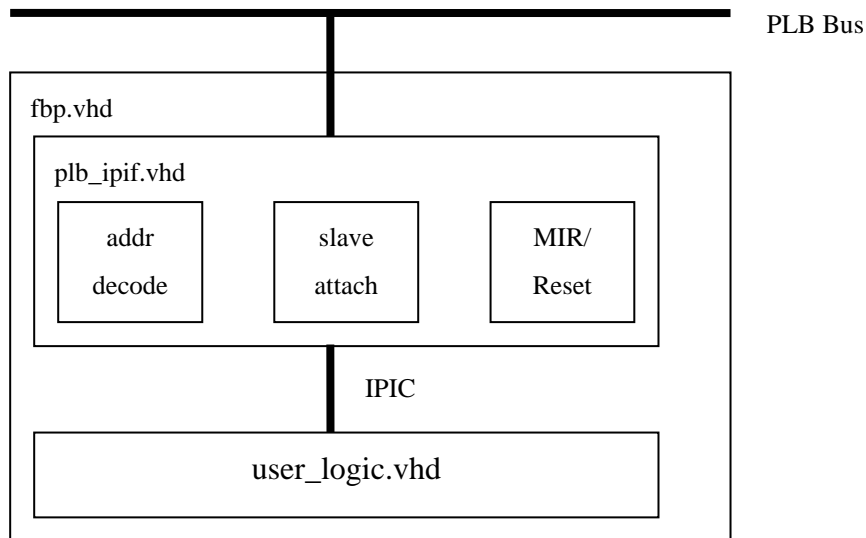


圖 4-5 FBP 元件與 PLB bus 間的模組示意圖

建立好 FBP 元件後，我們執行 Import Peripheral Wizard 將 FBP 加入系統中。FBP 元件是以 PLB Slave 的模式與 PLB Bus 連接。執行完 Import Peripheral Wizard 後，EDK 軟體會將 FBP 元件的資訊記錄在 MHS 檔案中。這樣我們就完成了 FBP 演算法嵌入式系統的架構。接下來我們將在 user_logic.vhd 檔案中的 user_logic VHDL 模組內完成 fbp 程式中欲實作成硬體的部份。

建立 FBP 元件並加入系統中的整個流程如下所示：

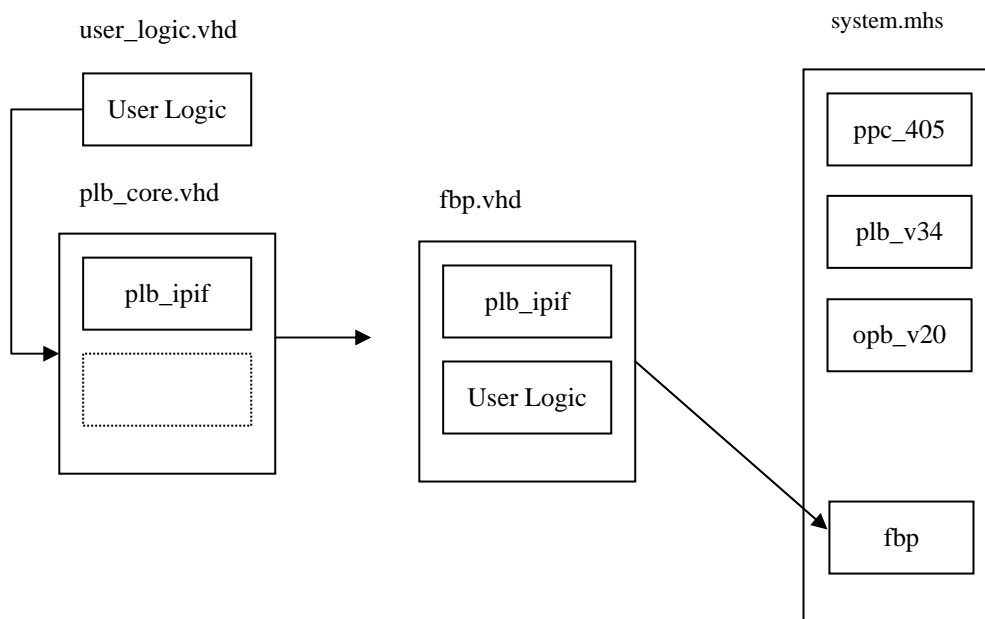


圖 4-6 使用者建立週邊元件並加入系統的過程

第五章 FBP 元件實作

從上一章節我們可以知道，我們要在user_logic VHDL模組內中完成4.2.1節提到的FBP元件功能一及功能二的VHDL程式撰寫。在實作的過程中，我們將使用到Xilinx公司提供的兩個IP，一個是Single-Port Block Memory v6.1 IP，另一個是Fast Fourier Transform v3.1 IP。另外我們為了FBP IP功能二另外建立了一個名為fbp_locate的VHDL 程式模組，在user_logic模組內對剛剛提到的兩個IP及fbp_locate模組作元件宣告(Component Declaration)後使用。因此FBP元件的模組階層如下：

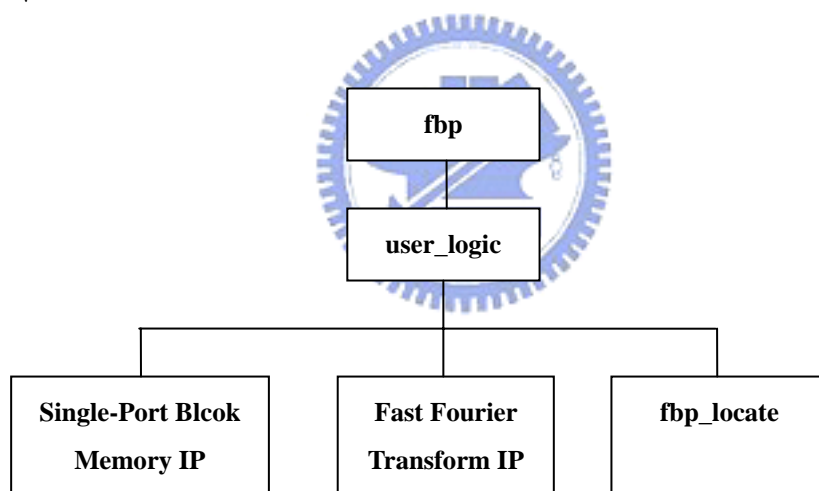


圖 5-1 FBP 元件內的模組階層

底下我們依序介紹 Single-Port Block Memory IP、Fast Fourier Transform IP 及我們自己建立的 fbp_locate 模組，然後介紹最重要的 user_logic 模組內部實作細節。

5.1 IP 介紹及 fbp_locate 模組實作

Xilinx 公司提供的嵌入式系統發展環境除了平台像是我們使用的 ML310 Development Board，及搭配的軟體如 EDK 及 ISE 之外。Xilinx 也針對 IC 設計者經常使用的特殊功能或是計算，實作成 IP 供使用者使用。這樣使用者可以節省設計開發的時間，也可以使得設計更為模組化。模組化的好處將使得電路設計的維護性增高，在模組電路的設計與驗證的過程中，可以比較容易找到設計錯誤的地方。

5.1.1 Single-Port Block Memory IP

ML310 Board 中的 FPGA 有 136 塊 18Kbits 的 Block Memory。由於 Block Memory 是 on-chip memory，因此在存取速度上會比 off-chip memory 要快。我們在設計 FBP 元件時要完成 4.2.1 節提到的兩個功能，其中 FBP 元件功能一需要一次讀進 sinogram 影像中某一系列的所有 pixel 灰階值。因此我們需要在我們的設計中增加一個記憶體空間來儲存這些讀入的資料。Signal-Port Block Memory V6.1 IP 可以幫我們使用 FPGA 中的 Block Memory 規劃出一塊可以供我們使用的記憶體空間。

此 Single-Port Block Memory V6.1 IP 的重要特色如下：

- 支援 Read-Only Memory 及 Random-Access Memory 的功能
- 根據選擇的硬體架構，記憶體大小的設定範圍由 2 到 2M words 之間。
- 可以針對 memory 的儲存內容作初始化。
- 記憶體寫入的模式有 Read-after-Write，Read-before-Write 及 No-Read-on-Write 三種模式。

底下是 Signal-Port Block Memory V6.1 IP 的線路符號代碼，其中黑色的部分代表在我們的設計中有使用到的訊號，灰色則是沒有選用到的訊號：

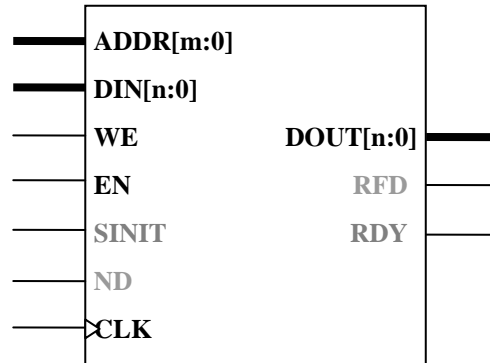


圖 5-2 Core Schematic Symbol

下列的訊號定義表僅列出我們使用到的訊號定義：

表 5-1 Single-Port Block Memory V6.1 IP Core Pinout

Port Name	Direction	Description
DIN[n:0] (Optional)	Input	Data Input: Data written into memory.
ADDR[m:0]	Input	Address: Memory location for data written to/read from.
WE (Optional)	Input	Write Enable: Allows data transfer into memory.
EN (Optional)	Input	Enable: Enables access to memory via read and write operations.
CLK	Input	Clock: All memory operations synchronous with rising or falling edge of clock input, depending on user configuration of the clock pin polarity. When memory is enabled, all control signals, input/output data are registered on the rising or falling edge of clock.
DOUT[n:0]	Output	Data Output: Synchronous output of the memory.

Single-Port Block Memory V6.1 IP 的功能介紹如下：

當 Block Memory 的致能訊號 EN 是 1 時，所有記憶體的動作將發生在時脈訊號 CLK 的上升緣(Rising Edge)或是下降緣(Falling Edge)變化時。當致能訊號 EN 為 0 時，則記憶體的設定及輸出訊號 DOUT 將不變。讀出的動作在致能訊號 EN 為

1 時，輸出訊號 DOUT 將輸出位置訊號 ADDR 所指定的記憶體位置內的值。當寫入訊號 WE 為 1 時，在輸入訊號 DIN 的值將存在位置訊號 ADDR 所指定的記憶體位置。而剛剛提到寫入的模式有三種，Read-after-Write，Read-before-Write 及 No-Read-on-Write。這是輸出訊號的行為在記憶體為寫入的狀態時，有三種不同的輸出模式。分別敘述如下：

- Read-after-Write 是指在同一個時脈中，在輸入訊號 DIN 中的值除了存到指定的記憶體位置外也直接傳到輸出訊號 DOUT 中當成輸出。
- Read-before-Write 則是同一個時脈中，位置訊號 ADDR 所指定的記憶體位置內的值將傳送到輸出訊號 DOUT 中當成輸出。
- No-Read-on-Write 是指記憶體在寫入的狀態時，輸出訊號 DOUT 維持原狀不作改變。

在我們的設計中，Single-Port Block Memory V6.1 IP 的設定上我們選擇 No-Read-on-Write 的模式。記憶體的讀寫動作設定為發生在脈訊號 CLK 的上升緣。而因為 FFT 及 IFFT 的過程中有 Zero Padding 的動作，所以記憶體大小，我們設定為 256 words(128*2)，每個 word 是 32bits，且 word 的初始值設定為 0。至於 SINIT、ND、RFD 及 RDY 訊號，我們並不需要因此設定上選擇不用，以簡化設計。我們參考到的時序圖如下：

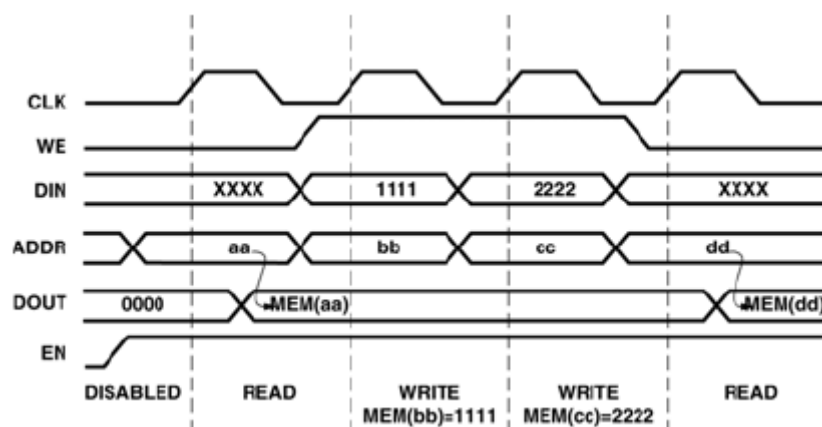


圖 5-3 No-Read-on-Write Mode Waveform

5.1.2 Fast Fourier Transform v3.1 IP

4.2.1節中提到的FBP元件功能一，包含FFT運算及IFFT運算。Xilinx公司也有提供FFT及IFFT的IP可以供我們使用，我們不需從頭設計FFT及IFFT的電路。

Xilinx公司提供的Fast Fourier Transform v3.1 IP的重要特色如下：

- 此 IP 可以在執行狀態中設定為 FFT 或 IFFT 運算。
- FFT 及 IFFT 可以支援的取樣點數目大小是 $N=2^m$ ， $m=3$ 到 16。
- 每一筆輸入資料的精確度 b_x 可以是 8、12、16、20、24 位元寬度。
- FFT 運算過程中使用到的 phase factor 精確度 b_w 可以是 8、12、16、20、24 位元寬度。
- 計算過程中計算的形式有：
 - Unscaled (full-precision) fixed point
 - Scaled fixed point
 - Block floating point
- 在 butterfly 運算過程中產生的數值可以使用 Rounding 或是 Truncation 的方式進位到最後一位。
- IP 的硬體架構有三種選擇：
 - Pipelined, Streaming I/O
 - Radix-4, Burst I/O
 - Radix2, Minimum Resource



底下是 Fast Fourier Transform v3.1 IP 的線路符號代碼，黑色的部分表示我們有使用到的訊號，灰色則代表在我們的設計中沒有選用這些訊號：

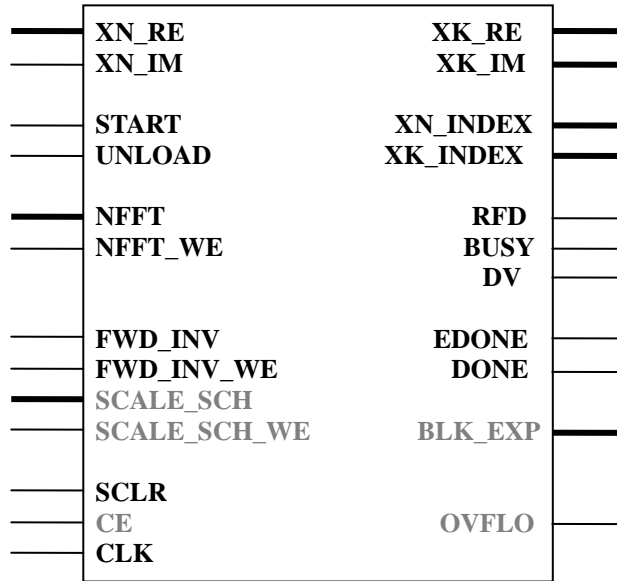


圖 5-4 Core Schematic Symbol

下列的訊號定義表僅列出我們使用到的訊號定義：

表 5-2 Fast Fourier Transform v3.1 IP Core Pinout

Port Name	Port Width	Direction	Description
XN_RE	b_{xn}	Input	Input data bus: Real component ($b_{xn}=8,12,16,20,24$)
XN_IM	b_{xn}	Input	Input data bus: Imaginary component ($b_{xn}=8,12,16,20,24$)
START	1	Input	FFT start signal(Active High): START is asserted to begin the data loading and transform calculation (for the Burst I/O architecture). For continuous data processing, START will begin data loading, which processing directly to transform calculation and the data unloading.
UNLOAD	1	Input	Result unloading (Active High): For the Burst I/O architecture, UNLOAD will start the unloading of the results in normal order. The UNLOAD port is not necessary for the Pipelined, Streaming I/O architecture.
NFFT	5	Input	Point size of the transform: NFFT can be the size of the transform or any smaller point size. For example, a 1024-point FFT

			can compute point size 1024, 512, 256, and so on. The value of NFFT is \log_2 (point size). This port is optional for Pipelined, Streaming I/O architecture.
NFFT_WE	1	Input	Write enable for NFFT (Active High): Asserting NFFT_WE will automatically cause the FFT core to stop all processes and to initialize the state of the core. This port is optional for Pipelined, Streaming I/O architecture.
FWD_INV	1	Input	Control signal that indicates if a forward FFT or an inverse FFT is performed. When FWD_INV=1, a forward transform is computed. If FWD_INV=0, an inverse transform is performed.
FWD_INV_WE	1	Input	Write enable for FWD_INV (Active High).
SCLR	1	Input	Master reset (Active High): Optional port.
CLK	1	Input	Clock
XK_RE[(B-1):0]	b_{xk}	Output	Output data bus: Real component.
XK_IM[(B-1):0]	b_{xk}	Output	Output data bus: Imaginary component.
XN_INDEX	$\log_2(\text{point size})$	Output	Index of input data.
XK_INDEX	$\log_2(\text{point size})$	Output	Index of output data.
RFD	1	Output	Ready for data (Active High): RFD is High during the load operation.
BUSY	1	Output	Core activity indicator (Active High): This signal will go High while the core is computing the transform.
DV	1	Output	Data valid (Active High): This signal is High when valid data is presented at the output.
EDONE	1	Output	Early done strobe (Active High): EDONE goes High one clock cycle immediately prior to DONE going active.
DONE	1	Output	FFT complete strobe (Active High): DONE will transition High for one clock cycle at the end of the transform calculation.

Fast Fourier Transform v3.1 IP 的功能簡單介紹如下：

Xilinx 公司提供的這個 IP，可以設定為執行 FFT 或是 IFFT 運算。此 FFT 是一個計算離散傅立葉轉換 (Discrete Fourier Transform, DFT) 高效率的演算法。但是注意的是 FFT 使用的函式是 DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-jnk2\pi/N} \quad k = 0 \dots N-1 \quad (5.1)$$

而 Fast Fourier Transform v3.1 IP 執行 IFFT 運算使用的函式是

$$x(n) = \sum_{k=0}^{N-1} X(k) e^{jnk2\pi/N} \quad n = 0 \dots N-1 \quad (5.2)$$

而不是和函式(5.1)對應的IDFT(inverse Discrete Fourier Transform)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{jnk2\pi/N} \quad n = 0 \dots N-1 \quad (5.3)$$

換句話說，如果作 128 筆取樣值的 FFT 轉換接著作 IFFT 轉換後，得到的資料將比原取樣值大 128 倍，這是我們使用此 IP 要注意的地方。

此 IP 的內部架構有三種形式可以供我們選擇，說明如下：

- Pipelined, Streaming I/O：此架構可以連續性的處理資料，也就是讀入資料，做轉換及資料的輸出可以同時執行。此架構下執行 FFT 或是 IFFT 運算所需的時間最少，但是此架構下 IP 要合成硬體所需要的硬體資源也是最多的。此架構使用的傅立葉演算法為 Radix-2 butterfly。
- Radix-4, Burst I/O：此架構下執行 FFT 或是 IFFT 運算有兩個階段，一個是資料的輸入輸出，另一個是資料的轉換運算過程。此架構下所需要的硬體資源會比 Pipelined, Streaming I/O 架構要少，但運算的時間會增加。內部使用的傅立葉演算法為 Radix-4 butterfly。
- Radix2, Minimum Resource：此架構跟 Radix-4, Burst I/O 架構一樣有資料的輸入輸出及轉換運算過程兩個階段，但是因為內部的傅立葉演算法為 Radix-2 butterfly。因此整個運算的時間最長，但所需要的硬體資源也最少。

另外，執行 FFT 或是 IFFT 運算時，在 radix-4 或是 radix-2 butterfly 轉換過程中，

數值可能會增大，因此考慮可能發生位元溢位的狀況下。運算的過程中數值的計算有三種形式：

- Unscaled (full-precision) fixed point：整個計算過程，不針對數值作任何比率的縮小，此情形在數值位元寬度不足時可能發生溢位狀況。
- Scaled fixed point：在作 FFT 或是 IFFT 會有多個 radix-4 或是 radix-2 butterfly 轉換過程，每個過程都作比率的縮小，縮小的倍數可能是 1、2、4、8，代表的是將數值作位元右移操作。每個過程要縮小的倍數定義在 SCALE_SCH 訊號中。
- Block floating point：此計算形式是指完成 FFT 或是 IFFT 運算後，由 IP 自動決定要縮小幾倍，而縮小的倍數會儲存在 BLK_EXP 訊號中。

同樣的上述的三種數值計算形式也會影響到此 IP 合成硬體所需要的資源。

在我們的設計中，FBP 軟體將需要將取樣值經由暫存器傳給 FBP IP 中再由 Fast Fourier Transform v3.1 IP 執行 FFT 運算，因此 Pipelined, Streaming I/O 架構無法跟我們的設計配合，因此我們選擇 Radix-4, Burst I/O 架構，使得資料的輸出入跟資料作傅立葉轉換分成兩階段。另外，我們處理的資料是 pixel 的灰階值，範圍是 0~255，只要 8bits 就可以表達。但是之前我們提到要使用 fix point 的方式來處理 float point 的資料，因此將會乘上 2 幕次方倍數來保留運算中產生的小數位的資訊，再加上考慮計算過程中可能發生的溢位問題，因此在輸入資料的位元寬度及 phase factor 我們設定為 24 位元寬度。資料輸出則由 IP 自動設定為 33 位元寬度。為保留計算中數值所代表的資訊完整性，我們選擇 Unscaled (full-precision) fixed point 計算形式，避免縮小倍數的同時而失去部分資訊。至於我們要執行多少取樣數的 FFT 及 IFFT 運算呢？前面我們提到原本的 128 pixels 的灰階值在執行 FFT 運算前要先作 Zero Padding 的動作，因次我們將取樣數設為 256 筆。這樣我們就完成 Fast Fourier Transform v3.1 IP 的參數設定。

底下是我們參考到的時序圖：

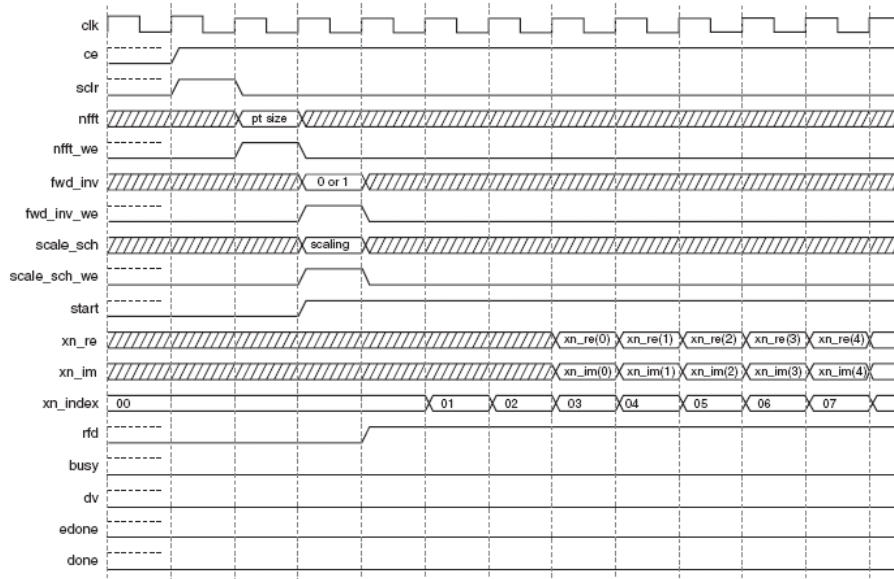


圖 5-5 IP 開始輸入資料的波形圖

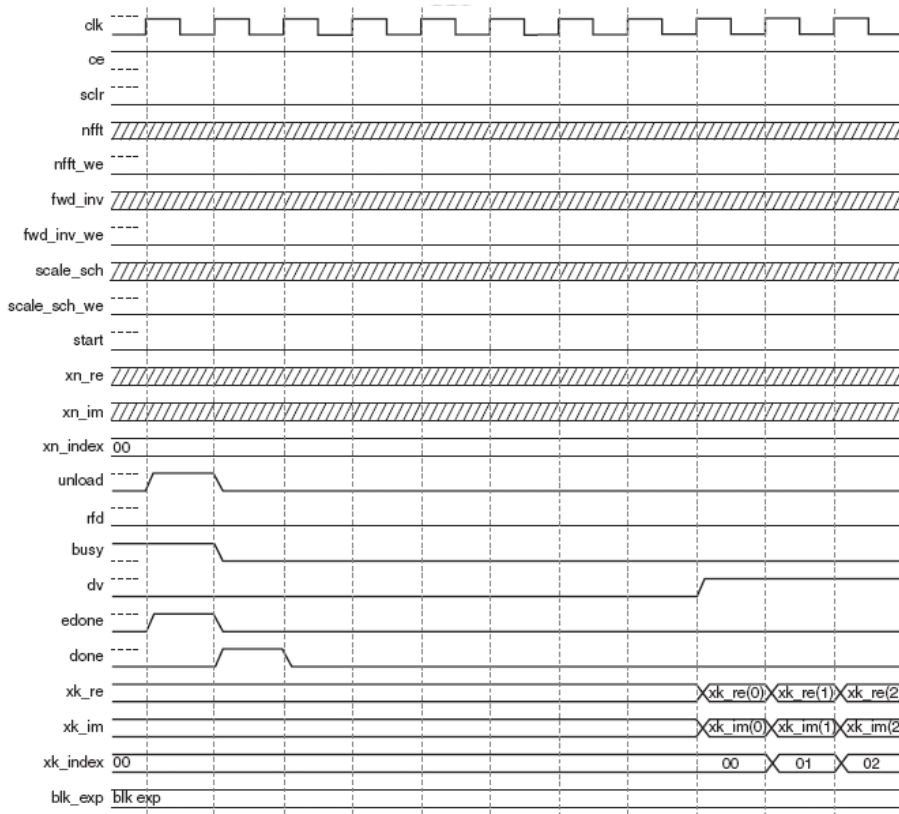


圖 5-6 IP 輸出資料的波形圖

其中要注意的是當 start 訊號拉成 1 時，從下一個 clock 開始的第四個 clock 才能輸入資料到 Fast Fourier Transform v3.1 IP 中。當 dv 訊號為 1 時，則開始輸

出傅立葉轉換後的資料。

5.1.3 fbp_locate VHDL 程式模組實作

我們建立fbp_locate模組來完成4.2.1節提到的FBP IP功能二的實作，而FBP元件功能一則寫在user_logic.vhd檔案中的user_logic模組中。FBP元件功能二的工作內容是輸入三階for-loop的索引值y、x及i，其中y及x代表的是重建影像上某個pixel的座標索引(x, y)，而i則表示sinogram影像上的列索引值，也就是斷層掃描從0°到180°共K次的投影中的第i次投影，經由i值可以算出投影的角度 θ (theta)。fbp_locate模組收到此三個索引值後經過計算，可以得到兩個數字，lbin及frac，而frac則是mid-bin在lbin及hbin的位置比例值(參考表 4-2)。線路符號代碼如下：

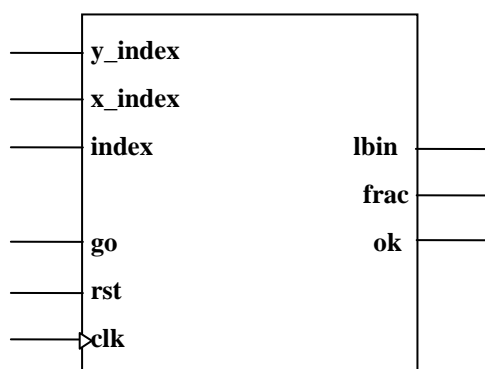


圖 5-7 Core Schematic Symbol

訊號定義如下表所示：

表 5-3 fbp_locate module Core Pinout

Port Name	Type	Direction	Description
y_index	integer	Input	表示重建影像的y軸座標值。(參考表 4-2)
x_index	integer	Input	表示重建影像的x軸座標值。(參考表 4-2)
index	integer	Input	表示sinogram影像的列索引值。(參考表 4-2)
go	std_logic	Input	當 go 訊號為 1 時，開始運算。

rst	std_logic	Input	當 rst 訊號為 1 時，fbp_locate 模組的 port 訊號及內部訊號均為 0。
clk	std_logic	Input	Clock 訊號。
lbin	integer	Output	mid-bin 相對於 $Q_0(t)$ 中心位置的前一個整數位置 (low-bin)。(參考表 4-2)
frac	integer	Output	mid-bin 位於 high-bin 及 low-bin 之間的位置比例。(參考表 4-2)
ok	std_logic	Output	當 ok 訊號為 1 時，表示運算完成，否則為 0。

fbp_locate 模組的功能是作一連串的數學運算，我們參考表 4-2 的內容，對於這些數學運算式，我們稍微拆解成為較短的算式，並對於沒有相依性的算式，重新排列計算的順序。這些算式當中有 $\sin(\theta)$ 及 $\cos(\theta)$ 的三角函數運算，這邊我們建立一個表格儲存 θ 從 0° 到 90° 共 91 個 $\sin(\theta)$ 值，改寫算式以查表的方式來完成三角函數計算。

我們以有限狀態機 (Finite State Machine, FSM) 設計一個循序邏輯電路 (Sequential Logic)，在狀態機的每個狀態 (state) 中執行之前拆解好的短算式，這樣就可以循序執行完這些數學運算式。下面是狀態圖 (State Diagram) 的示意圖：

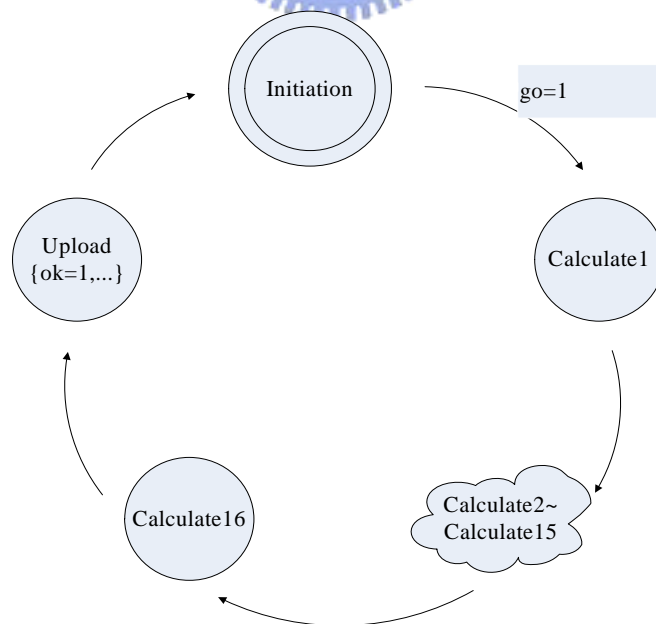


圖 5-8 fbp_locate 模組中的有限狀態機示意圖

其中 Initiation 狀態是 fbp_locate 模組的初始狀態，初始化模組內的參數及訊

號，當 fbp_locate 模組的訊號(port signal)go 為 1 時，狀態循序改變完成每個狀態的工作，到了 Upload 狀態，將訊號 ok 設為 1，表示工作完成，然後跳回 Initiation 狀態。

5.2 user_logic VHDL 程式模組

5.2.1 user_logic 模組實作

由於 FBP 元件功能一要件作 FFT，加權運算及 IFFT，而 FBP 元件在 PLB bus 系統中是 Slave mode，所以我們使用 IPIF 框架提供的服務建立一組暫存器作為資料溝通的方式。而 user_logic 模組內有定義好的暫存器讀寫功能，因此我們不為了 FBP 元件功能一另外建立一個 VHDL 模組，避免 FBP 元件內模組之間作多餘的資料交換動作。

user_logic 模組的線路符號代碼如下，而訊號的定義請參考表 4-3。

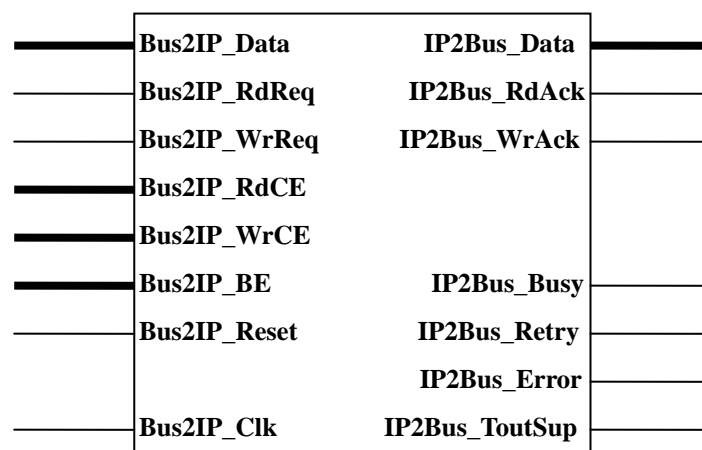


圖 5-9 Core Schematic Symbol

根據 FBP 元件功能一及功能二，我們需要規劃一個有限狀態機來控制 FBP 元的工作流程，圖 5-10 是有限狀態機的主要狀態圖。其中 IDLE 狀態，是 user_logic

模組的初始狀態，在IDLE狀態中的工作就是初始化user_logic模組內的參數及內部訊號。

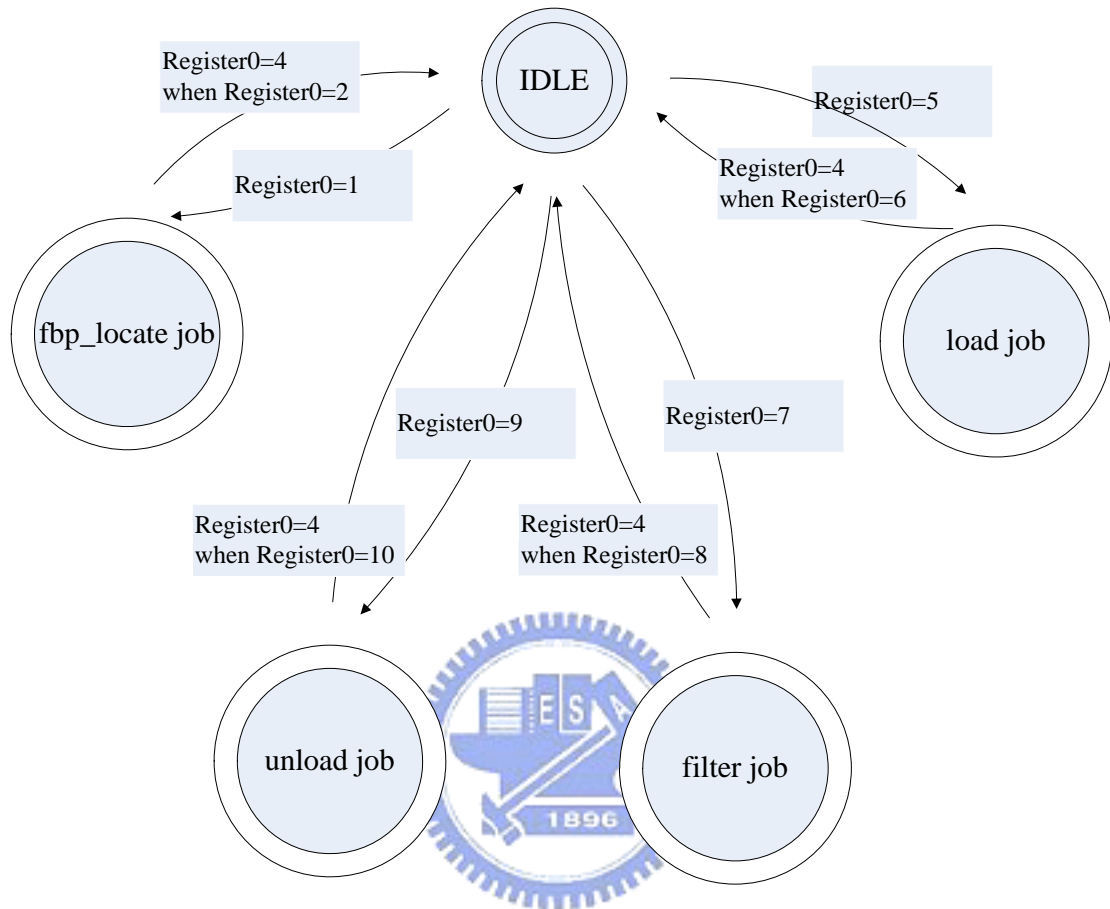


圖 5-10 user_logic 模組的有限狀態機示意圖

接著我們將FBP元件功能一及功能二細分為四大部分，load job，filter job，unload job，fbp_locate job。每一部份都包含一連串的狀態(state)，不同的狀態執行不同的工作，利用狀態的改變循序完成整體工作，其中有關公式計算的部份作法如同5.1.3節一樣分解成較短的公式分配到各個狀態中執行。四大部分的工作內容依執行順序分述如下：

- load job：從暫存器中讀取 sinogram 影像的資料並儲存到 Signal-Port Block Memory IP 合成的 Block RAM 中。
- filter job：
 - 從 Block RAM 中將資料讀出並依序輸入 Fast Fourier Transform v3.1 IP

邏輯合成的電路中作 FFT 轉換。

- Fast Fourier Transform v3.1 IP 邏輯合成的電路輸出完成 FFT 轉換的資料，乘與濾波器 $|\omega|$ 後存回 Block RAM 中。
- 從 Block RAM 中將資料讀出並依序輸入 Fast Fourier Transform v3.1 IP 邏輯合成的電路中作 IFFT 轉換。
- Fast Fourier Transform v3.1 IP 邏輯合成的電路輸出完成 IFFT 轉換的資料，直接存回 Block RAM 中。
- unload job：將 Block RAM 中完成 filter process 的資料寫回暫存器，由軟體讀回資料。
- fbp_locate job：從暫存器中讀入三階 for-loop 的索引值 y 、 x 及 i ，並傳入 fbp_locate VHDL 模組中進行計算，得到 lbin 及 frac 兩個值後寫回暫存器中，由軟體讀回資料。

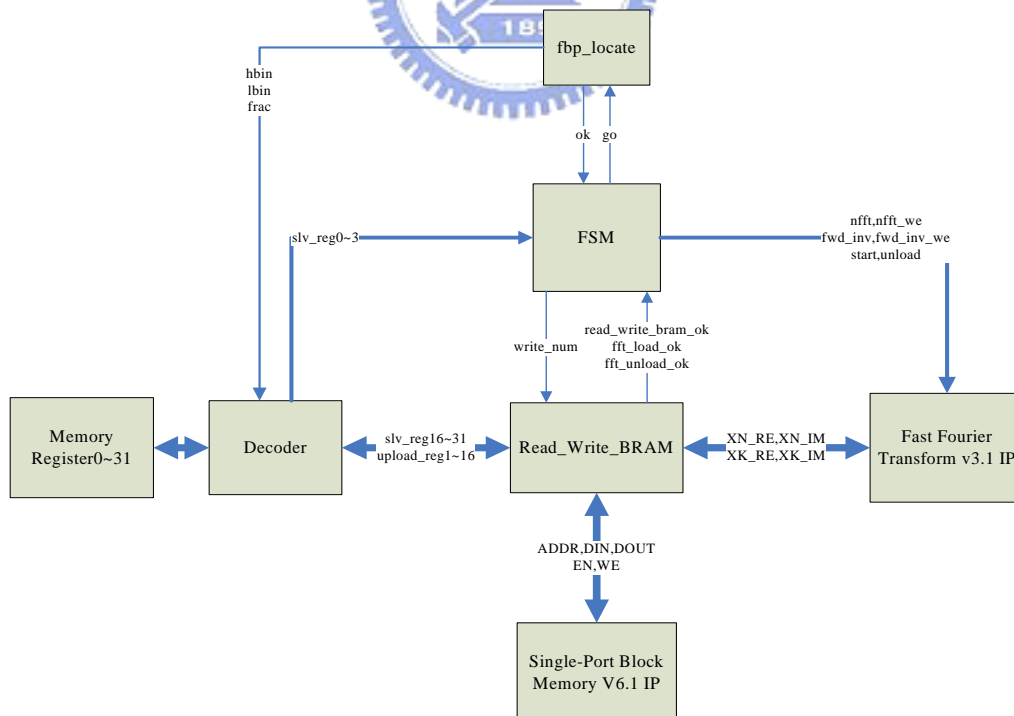


圖 5-11 user_logic 模組功能方塊圖

圖 5-11是user_logic模組內的功能方塊圖(function diagram)，依功能分類描述

user_logic模組內的主要架構，其中比較重要的內部訊號我們也列在方塊圖中。其中方塊FSM包含一個fsm process，此process的內容是一個有限狀態機控制狀態改變如圖 5-10所示。而方塊Read_Write_BRAM則包含一個read_write_bram process，此process搭配Single-Port Block Memory v6.1 IP與Fast Fourier Transform v3.1 IP用來完成前面提到的load job，filter job及unload job三個工作。fsm process藉由write_num訊號來告訴read_write_bram process，在此狀態下該執行那樣工作。當工作完成後read_write_bram process則回報fsm process工作完成。read_write_bram process介於Single-Port Block Memory v6.1 IP與Fast Fourier Transform v3.1 IP之間，除了擔任此兩個IP的資料收送外，也負責資料乘與濾波器 $|\omega|$ 作加權動作。在這邊要特別注意的是，read_write_bram process在使用Fast Fourier Transform v3.1 IP的電路執行FFT或是IFFT時，此電路的參數設定由fsm process直接設定後，read_write_bram process才將資料輸入電路中。

而第四個工作fbp_locate job則是執行fbp_locate VHDL 模組來完成，同樣的也是由fsm process發出go訊號啟動fbp_locateVHDL 模組中的狀態機，循序執行工作。完成後fbp_locateVHDL 模組發出ok訊號通知fsm process工作完成。

圖 5-11 中的 Decoder 方塊包含了 SLAVE_REG_READ_PROC 與 SLAVE_REG_WRITE_PROC 兩個process，主要的工作就是從暫存器中讀資料回FBP元件中，或是將資料由FBP元件寫回暫存器中。由圖 5-10和圖 5-11，我們可以清楚知道user_logicVHDL模組的結構與工作流程。

5.2.2 軟體與硬體溝通介面

在4.2.2節中，當我們建立好FBP演算法嵌入式系統的架構後，我們可以執行EDK軟體的LibGen工具。此時LibGen會幫我們建立FBP週邊元件的驅動程式，此驅動程式根據4.2.2節中選擇的IP Interface服務，會建立兩支程式，FBP_mWriteReg(BaseAddress, RegOffset, Data)與FBP_mReadReg(BaseAddress,

RegOffset)。此兩支程式可以讓我們從軟體中對暫存器進行讀寫的動作，而參數 BaseAddress和RegOffset則定義在標頭檔fbp.h中。

因為我們要將表 4-1及表 4-2的運算交由FBP週邊元件來作運算的加速，因此我們需要從軟體來控制FBP週邊元件的工作流程。參考圖 5-10，我們定義一些特定的數字寫入暫存器來指揮FBP週邊元件的工作流程。我們定義 32 個暫存器中的第 1 個暫存器Register0，作為軟硬體之間控制訊號的介面。參考圖 5-10底下我們列出一些我們從軟體用來控制FBP元件中有限狀態機流程的控制訊號。

表 5-4 軟硬體介面訊號定義

數字	意義
1	當 Register0=1 時，user_logic 模組的 FSM 跳至 fbp_locate job 狀態中，開始執行 fbp_locate 模組功能。
2	當 Register0=2 時，表示 fbp_locate job 已經完成。
4	當 load 或 filter 或 unload 或 fbp_locate job 完成後，軟體收到完成通知後，由軟體寫入 4 到 Register0 中通知 FSM 跳回 IDLE 狀態。
5	當 Register0=5 時，user_logic 模組的 FSM 跳至 load job 狀態中，開始執行工作。
6	當 Register0=6 時，表示 load job 已經完成。
7	當 Register0=7 時，user_logic 模組的 FSM 跳至 filter job 狀態中，開始執行工作。
8	當 Register0=8 時，表示 filter job 已經完成。
9	當 Register0=9 時，user_logic 模組的 FSM 跳至 unload job 狀態中，開始執行工作。
10	當 Register0=10 時，表示 unload job 已經完成。

根據上面我們自己的定義，及FBP_mWriteReg和FBP_mReadReg兩支程式，

我們改寫4.1.1節中原先已完成的嵌入式軟體有關表 4-1與表 4-2的部份，由軟體呼叫硬體執行運算加速，這樣我們的fbp演算法嵌入式系統就完成了。



第六章 實驗與討論

我們在第4章、第5章建立起我們的FBP週邊元件，在user_logic.vhd檔案中實作4.2.1節提到的FBP元件功能一及功能二，之後將元件加入我們所建立的嵌入式系統中完成系統硬體架構。我們也利用IPIF框架定義的 32 個暫存器作為軟硬體資料交流的介面，並利用EDK軟體建立的FBP週邊元件的驅動程式讓我們對FBP嵌入式程式進行改寫，這樣我們完成了軟體的架構。接下來我們要依照圖 4-1的EDK設計流程在ML310 平台上完成系統單晶片雛形。

6.1 系統驗證與實驗結果



依照圖 4-1的設計流程，我們的硬體平台經過Standard FPGA HW Development Flow的各個步驟，會產生一個system.bit的Bitstream檔。而FBP嵌入式軟體經過Standard Embedded SW Development Flow的各個步驟後會產生副檔名.elf的執行檔。接下來我們執行EDK軟體的The Bitstream Initializer工具，將此兩個檔案合併成為download.bit檔。接下來我們將此檔案存放在CompactFlash SanDisk中，插入ML310 平台上的CompactFlash Slot中。我們開啟ML310 平台上的電源，SanDisk中的download.bit檔案會下載至FPGA中，並對FPGA中的電路做規劃，接著FBP嵌入式系統就準備完成可以執行影像重建的工作。工作開始之後，FBP嵌入式系統會至CompactFlash SanDisk中讀取事先存好的sinogram影像，接著執行影像重組，然後將重建好的影像回存到SanDisk中完成整個工作。

我們將比較有使用FBP元件作硬體加速，及沒有使用FBP元件純粹使用軟體

實作FBP演算法的嵌入式系統之間的執行效率。底下我們以表 2-1 FBP演算法的 pseudo code 中的Filter及Backproject兩段程式碼執行時間作為比較的目標。

表 6-1

項目	Filter	Backproject
使用 FBP 元件執行硬體加速	約 1 秒	約 24 秒
未使用 FBP 元件執行硬體加速	約 2 分鐘	約 1 分鐘

由表可知我們將表 4-1與表 4-2的程式碼實作成FBP週邊元件的功能一及功能二而獲得的硬體加速使我們的系統執行時間縮短 7.2 倍左右。底下我們列出FBP嵌入式系統SoC雛型在FPGA電路資源中的使用率及系統的時脈等相關資訊。

- 時脈資訊：

系統時脈：81.773MHz

最長路徑延遲(Maximum path delay from/to any node)：2.223ns

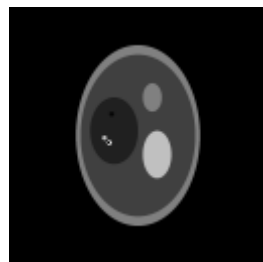
- FPGA 資源使用率

18x18 bits 的乘法器： 55 out of 136 40%

18Kbits Block RAM： 77 out of 136 56%

SLICES⁽¹⁾： 8621 out of 13696 62%

底下我們以 128x128 pixelx 大小的”head phantom”圖作為我們影像重建的範例。



(a)



(b)

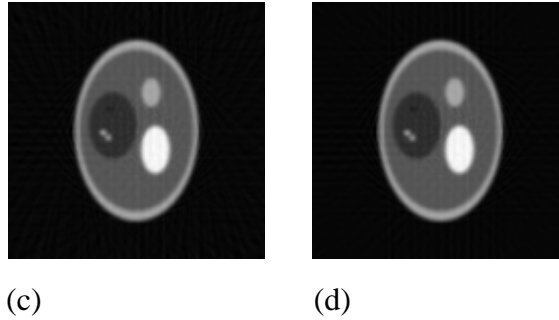


圖 6-1 (a)是原始影像，(b)是圖(a)的 sinogram 影像，共投影 128 次，每次投影偵測 128 筆資料。

(c)是使用 FBP 元件作硬體加速後重建回的影像，(d)是純用嵌入式軟體重建回的影像。

備註：每個 SLICE 內有兩組(1) 4-input LUT + (1)Flip-Flop + Carry Logic

6.2 結果討論與未來展望

在我們的 FBP 嵌入式系統 SoC 設計上對於 FBP 演算法的內容上並無多大的修改，只有資料精確度因為 fix point 的關係而不同。我們在硬體實作上對於資料的精確度保持到小數點以下三位，因此重建的影像與 PC 上使用軟體重建的影像幾乎相同。

在系統的效能上我們處理 128x128 pixels 大小的 sinogram 影像(0 度到 180 度共 128 個投影角度，每次投影偵測有 128 筆偵測資料)重建出 128x128 pixels 大小的灰階影像需要 25 秒左右，在效能上仍有改善的空間。要改善效能，可以在兩方面做改進，一個是演算法結構，一個是在系統設計上。FBP 演算法是相當成熟的演算法因此修改的空間不多。而系統設計上，FBP 元件在 PLB bus 系統是 Slave 模式，因此無法獲得匯流排控制權去存取記憶體的资料。因此我們需要定義 32 個暫存器，由 PowerPC405 處理器控制 PLB bus 系統，將需要處理的资料傳到暫存器後再由硬體自暫存器取得資料。因此資料在軟硬體一來一往的傳送之間，浪費不少的時間。

未來在系統設計上，改善效能的目標是了解 PLB bus 系統中，master 元件的

架構，使FBP元件以master模式接上PLB bus系統，則可以由FBP元件獲得匯流排控制權，進而主動存取記憶體資料，這將省卻不少資料傳送的時間，且可以控制記憶體，則我們可以修改表 4-1與表 4-2中的數學公式，以更簡潔的運算來達成相同的功能，相信這樣必能大大的改善FBP演算法嵌入式系統的效能。



參考文獻

- [1] A.C. Kak and M. Slaney, Principles of Computerized Tomographic Imaging. IEEE, Inc., New York: IEEE Press, 1988.
- [2] A.V. Lakshminarayanan, "Reconstruction from divergent ray data," tech. rep., Dept. Computer Science, State University of New York at Buffalo, 1975.
- [3] G.N. Ramachandran and A.V. Lakshminarayanan, "Three dimensional reconstructions from radiographs and electron micrographs: Application of convolution instead of fourier transforms," Proceedings of the National Academy of Sciences, vol. 68, pp. 2236 - 2240, 1971.
- [4] R.H. Bracewell and A.C. Riddle, "Inversion of fan beam scans in radio astronomy," *Astrophysics Journal*, vol. 150, pp. 427 - 434, 1967.
- [5] Raman Rao, Ronald D. Kriz, "Parallel Implementation of the Filtered Back Projection Algorithm for Tomographic Imaging, "
http://www.sv.vt.edu/xray_ct/parallel/Parallel_CT.html .
- [6] Srdjan Coric, Miriam Leeser, Eric Miller, Marc Trepanier, "Parallel-Beam Backprojection: An FPGA Implementation Optimized for Medical Imaging, " Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays.
- [7] "Xilinx: The Programmable Logic Company™, " <http://www.xilinx.com/> .
- [8] "電子工程專輯, " <http://www.eettaiwan.com/> .