國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：　黃 文 吉　博士

以菲涅耳轉換及相位展開為基礎之
數位全像顯微鏡在 FPGA 上之實現

研究生：　莊　子　昕　　撰

中華民國　一百零一　年　七　月

中文摘要

本論文旨在提出一硬體架構可以將數位全像片還原成原始影像相位圖，此硬體架構適用於嵌入式的數位全像顯微鏡(Digital Holographic Microscopy, DHM)系統，能夠加快運算來即時取得正確的還原全像影像。

本硬體架構採用皆以快速傅立葉轉換(FFT)為基礎的菲涅耳轉換搭配相位展開法則演算法來達到全像圖重建的目的。其中快速傅立葉轉換為高複雜度計算，對於一些需要即時顯示還原影像的應用往往會遇到很大的困難，因此本論文使用硬體電路架構來執行相關運算，以克服一般嵌入式系統上運算能力的限制，以縮短相位重建影像運算所需要花費的時間。另外，為克服硬體常見精確度不足問題，本硬體電路中大多使用IEEE 754浮點數格式來提升計算的精確度。

最後我們以現場可程式化邏輯閘陣列(Field Programmable Gate Array ,FPGA)為開發平台實現並實際測量硬體電路的資源消耗以及運算時間；實驗的結果顯示了本論文所提出的相位展開法則硬體架構能夠得到正確的還原結果，並且有效的降低還原相位圖運算所需要花費的時間以及擁有低硬體資源消耗的優點，因此適合使用於嵌入式的 DHM 系統。

關鍵詞：數位全像顯微鏡、系統晶片設計、FPGA、菲涅耳轉換、相位展開法則

誌謝

首先感謝研究所兩年來的指導教授　黃文吉博士，給予我研究啟發，以及在我研究遇到瓶頸之時提供我許多建議，讓我不只在專業領域上獲益良多，也學習到正確的研究態度，在此致上最深的謝意。同時也感謝台灣師範大學光電科技研究所　鄭超仁博士以及師大光電所　賴信吉同學，給予我研究上許多背景知識協助，還有清雲科技大學副教授　歐謙敏博士，百忙之中撥空在論文上所給予評閱與建議。

再來感謝一起生活兩年的同窗摯友：范范、嘉翎、李先生、歐歐和陳昊，不僅在課業上可以互相協助，也讓研究室生活天天都很有趣且充滿許多歡笑；還有感謝學弟們：建廷、清志、聖穎、國璿、任軒和瀚逸，在口試時給予許多幫忙協助；額外感謝師大分部7-11和神祕花園阿姨，讓我們研究生活多出一些可以去走動的地方。

最後感謝，我的家人，給予我最大支持與鼓勵；我的男友建廷，不管在研究上或精神上都給予我全力協助與包容，讓我有更多力量面對壓力與挫折；我的貓PI，每天撒嬌、呼嚕一下，煩躁的心情都消失了。

還有其他許多給予過我鼓勵，或是讓我更有信心堅持下去的人、事、物，內容繁多不及備載，在此也一併致謝，因為有你們我才能夠順利完成研究。

目 錄

附圖目錄

附表目錄

第一章 緒論

本章節主要在探討本論文的研究背景與動機、研究目的與方法，並大略說明各章節的主要內容與重要特性。

1.1 研究背景與動機目的

在科技產業研發或醫學工程上生物細胞檢測等應用中，光學測量技術因為具有非侵入性質的優點而受到廣泛使用，現今的數位全像顯微鏡(Digital Holographic Microscopy, DHM)[1]即為其中一種非侵入性質的影像擷取工具，其不只可以測量細微之物體，還可以針對物體的三維影樣測量相位的改變，對於測量三維物體有很大的貢獻。全像術是由Dennis Gabor於1948年提出[2]，當初是為了解決電子顯微顯微鏡技術及性能上的問題，由於此技術可以記錄完整光波資訊，理論上可以把完整的三維影像重新產生出來，因此被命名為全像術(Hologram)。隨時代進步，全像片記錄介質已由傳統的底片等化學材料演變為數位化記錄方式，而數位全像顯微鏡能夠有效的讀取光學上透明（transparent）的或是反射性（reflective）的樣本之振幅（amplitude）和相位（phase）等資訊，並且利用感光藕荷元件（charge-coupled device, CCD）或是互補式金屬氧化層半導體（complementary metal oxide semiconductor, CMOS）記錄包含樣本資訊的數位全像圖，基於這樣的特性，其目前已被廣泛的使用在微結構、活細胞與微生物等方面的分析與檢測。

從全像圖到還原圖形，需要經過圖形重建，可用菲涅耳轉換近場繞射式、捲積法或角頻譜繞射等方法，然而利用數位全像顯微鏡所重建出來的三維度影像相位圖，其相位數值會被壓縮在$(-\pi,\pi]$的範圍之內，造成相位圖中產生不連續點(discontinuity)，所畫出的相位圖呈現非線性的分布，而無法立即被使用於相關的應用中。為了消除這些不連續點重建出真實的相位，利用相位展開法則(phase unwrapping method)是一種最常用的解決方式，藉由相位展開法則運算，使得全部的相位資料和真實的影像互相符合，而一般常見的技術像是Synthetic Aperture Radar (SAR)/Interferometric Synthetic ApertureRadar (InSAR)[3,4]、核磁共振攝影(MRI)[5]等方面，也可利用相位展開法則來重建相位資訊。

一般而言，以捲積法進行光波繞射重建全像圖，必須進行三次傅立葉轉換，其計算複雜度很高，而利用菲涅耳轉換[9,10,11]只需做一次傅立葉轉換，即可完成數位全像的數值重建。另外，以光柵掃描（raster-scan）為基礎的相位展開法則，雖然能快速的執行相位展開運算，但當影像受到雜訊的干擾或破壞時，影像中的某些相位數值會因為雜訊的破壞而產生誤差，這些錯誤的相位資訊將會在往後的運算過程中不斷累積，導致最後幾筆的相位數值產生嚴重的誤差。因此許多健全的相位展開法則[6,7,8]已經被提出來修正誤差累積的問題，例如最小平方誤差解（minimum squared error solution）等方式，即使影像受到雜訊的破壞，也能夠有效的還原正確的相位值。

不管是全像圖重建或是相位展開法則，此類型的演算法則大多有很高的計算

複雜度，對於一些將數位全像顯微鏡架設於可攜式的嵌入式系統上，且希望能夠立即顯示還原後影像的相關應用中，若使用這些計算複雜度高的演算法，往往在設計上會遇到幾種常見的困難。其一是受限於一般嵌入式系統有限的運算能力，對於實現這些高計算複雜度的演算，可能會有執行時間過長問題，即使透過個人電腦以軟體來實現的方式，也未必能在極短的時間內立即取得還原過後的影像，尤其當影像的維度越大時，所花費的時間將會更加冗長；其二是受限於嵌入式系統的有限資源，當設計越複雜硬體電路時，通常所使用資源也會消耗越多，隨著影像維度越大，資源消耗也會增加，又由於資源問題，對於電路中資料表示也必須錙銖必較，若使用太多位元表示一資料，可能造成資源的浪費，相反的，若使用太少位元表示一資料，則可能造成計算精確度的下降，導致最後結果不佳。

為了解決以上這些問題，本論文提出一套由菲涅耳轉換搭配相位展開法則的硬體電路架構來執行全像圖重建運算，利用這樣的硬體架構設計，能夠克服一般嵌入式系統上運算能力的限制，大幅縮短運算所需要花費的時間，且此硬體電路大多使用浮點數計算，因此能夠迅速且正確的得到與軟體實現相似的還原結果。同時本論文所提出的硬體電路架構具有高精確度、低硬體資源消耗、低功率的優點，因此非常適合使用於嵌入式的DHM系統。

1.2 研究方法

為了使設計出來的電路具有較低的硬體資源消耗、較低功率消耗以及高精確度等優點，且必須求出正確相角並修正受到雜訊破壞的影像相位資訊，本論文根據以傅立葉轉換(Fourier transform)為基礎的全像圖重建演算法及相位展開法則演算法[12]提出一個硬體架構，最後計算出的結果會得到一個最小平方誤差解(minimum squared error solution)。

本論文提出的硬體電路架構主要分為三大部分，分別是菲涅耳轉換電路(Fresnel transform circuit)、相位展開法則電路(Phase unwrapping circuit)以及嵌入式記憶體(on-chip RAM)，其中菲涅耳轉換電路又分為菲涅耳轉換前單元(pre-transform unit)、快速傅立葉轉換單元(FFT unit)、菲涅耳轉換後單元(post –transform unit)，而相位展開法則電路則分為相位展開轉換前單元(pre-transform unit)、離散餘弦轉換單元(DCT unit)、相位展開轉換後單元(post –transform unit)。

菲涅耳轉換電路及相位展開法則電路為運算單元，負責執行菲涅耳轉換和相位展開法則中所需的運算，並且都是以管線化(pipeline)架構的方式實現，來最大化(maximize)電路整體的throughput。On-chip RAM 則是負責儲存各運算單元的運算結果與提供各單元運算的來源資料，搭配各運算單元各自的控制器(controller)以及位置產生器(address generator)自動產生訊號和記憶體位置來存取on-chip RAM中的資料，減少花費在處理資料存取上的時間。

在硬體電路設計中，運算時通常使用定點數(Fixed-point)來進行運算，其中所存在的問題是，設計者必須決定要將小數點設定在哪一個位數，因為表示資料的寬度有限，若是設計不佳則容易導致最後運算結果不佳，尤其是一些會使用到之前運算結果的計算，因為精確度不足所導致的誤差會不斷累積；因此本論文中除了相位展開轉換前單元之外，其餘電路皆以IEEE 754單精度浮點數格式進行運算，其可表示範圍較大、誤差較低，可降低誤差累積，提高整體電路運算精確度，另外，浮點數表示具有固定格式，不會隨數字變大而使位元數變多，可以確保資源消耗量不會隨圖片增大而增加。

本論文提出的硬體電路架構整合在以FPGA(Fields Programmable Gate Array)為基礎的可程式化系統晶片(System on Programmable Chip, SoPC)[13]平台，來實際測量電路的運算時間以及驗證執行結果。而使用SoPC的優點在於可以重複的更改與驗證設計中的電路，使得設計的彈性佳，帶給設計者極大的便利性。除此之外，也會使用Intel Core I7的CPU以軟體實作相同的演算法則，此目的除了將軟體的執行時間與經由本論文提出的硬體架構的執行時間相比較，也用來驗證本論文硬體架構運算結果之正確性。

1.3 全文架構

本篇論文共分為五個章節，以下為各章節內容概述：

【第一章】緒論

說明本論文的研究背景、動機、目的、方法及本文架構。

【第二章】基礎理論及技術背景介紹

簡介本論文所使用之演算法、理論基礎、技術背景及本論文在數位全像顯微鏡中的應用。

【第三章】系統架構

介紹所提出的菲涅耳轉換電路及相位展開法則電路的設計架構，並提供各電路內部的討論與說明。

【第四章】實驗數據與效能比較

包含了系統環境的說明、相關實驗數據分析以及軟硬體效能比較。

【第五章】結論與未來展望

對於所提出之硬體架構以及實驗的結果做一個總結以及未來發展方向。

第二章 基礎理論及技術背景介紹

本章節將介紹本論文所使用的基礎理論與技術背景。首先，本章的第一節將介紹本文所使用的菲涅耳轉換法(Fresnel Transform)基本運作流程，接著在第二節中介紹相位展開演算法的基本運作流程，第三節中則會討論菲涅耳轉換與相位展開法則在數位全像顯微鏡中的應用，最後，第四節介紹FPGA技術與SoPC系統整合設計，方便讀者對本論文能有初步的了解與認識。

2.1 菲涅耳轉換運算流程

在菲涅耳轉換中，以全像片平面的複數振幅資訊為輸入，而取得全像片中物體的複數振幅資訊之後，首先要先進行數值重建，因此，本節將討論本論文所採用的 Fresnel Transform 近場繞射公式及其數值重建方法。

菲涅耳轉換式(Fresnel transform)又可稱菲涅耳近似式(Fresnel approximation)，當觀測平面至孔徑平面間的距離遠大於波長 λ 時，可以由 Huygens-Fresnel principle[14]的積分式推導出菲涅耳轉換式，然而進行數位計算時，必須將此菲涅耳轉換式表達成離散形式，如公式(2.1)所示。

$$\varepsilon_{u,v} = \frac{j}{\lambda z} e^{-j\frac{2\pi}{\lambda}z} e^{-j\frac{\pi}{\lambda z}\Delta_p{}^2(u^2+v^2)} \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left[\eta_{x,y} e^{-j\frac{\pi}{\lambda z}\Delta_f{}^2(x^2+y^2)}\right] e^{-j2\pi\left(\frac{xu}{N}+\frac{yv}{N}\right)} \quad (2.1)$$

假設影像之長與寬皆為 N，先令$\eta_{x,y}$為某一數位全像片之影像資訊，其中

$0 \le x < N$ 且$0 \le y < N$，λ 為波長，z 為傳播距離，Δ_p為計算公式時所需要的常數值，三者皆設為定值，而Δ_f可由Δ_p導出，其也為一定值，關係式如式(2.2) ，

$$\Delta_f = \frac{\lambda z}{N\Delta_p} \tag{2.2}$$

在公式(2.1)中，與$\eta_{x,y}$相乘之指數函數可以依指數加法律整理得公式(2.3)，並將其重新定義為 μ，如公式(2.4)，

$$\mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_f{}^2(x^2+y^2)} = \mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_f{}^2(x^2)} \times e^{-j\frac{\pi}{\lambda z}\Delta_f{}^2(y^2)} \tag{2.3}$$

$$\mu_x = \mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_f{}^2(x^2)} \quad , \quad \mu_y = \mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_f{}^2(y^2)} \tag{2.4}$$

公式(2.4)與全像片影像資訊相乘後得函數$\rho_{x,y}$，如公式(2.5)，此為一週期函數，因此可將其作傅立葉轉換之後可得函數$\tau_{u,v}$，如公式(2.6)，本論文中，實作了快速傅立葉(FFT)及離散傅立葉(DFT)兩種架構，將於第三章系統架構和第四章實驗數據及效能比較中加以討論，

$$\rho_{x,y} = \eta_{x,y} \cdot \mu_x \cdot \mu_y \tag{2.5}$$

$$\tau_{u,v} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \rho_{x,y} e^{-j2\pi(\frac{xu}{N}+\frac{yv}{N})} \tag{2.6}$$

由於菲涅耳轉換式中，λ、z 皆為固定值，因此可定義一定值 α，如公式(2.7)，

$$\alpha = \frac{j}{\lambda z} \mathrm{e}^{-\mathrm{j}\frac{2\pi}{\lambda} z} \tag{2.7}$$

另外，與$\tau_{u,v}$相乘之指數函數也可仿公式(2.4)定義為 ω，可得公式(2.8)，

$$\omega_u = \mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_p{}^2(u^2)} \quad , \quad \omega_v = \mathrm{e}^{-\mathrm{j}\frac{\pi}{\lambda z}\Delta_p{}^2(v^2)} \tag{2.8}$$

接著重新整理式(2.1)，可得出菲涅耳公式複數數值點的解$\varepsilon_{u,v}$，如公式(2.9)。

$$\varepsilon_{u,v} = \alpha \cdot \omega_u \cdot \omega_v \cdot \tau_{u,v} \tag{2.9}$$

為了後續進行相位展開法則運算，必須將 ε 進行反正切函數(arctangent)運算來取得相角(phase)ζ，如公式(2.10)，

$$\zeta = tan^{-1}\varepsilon \tag{2.10}$$

在本文中反正切函數運算使用 arctangent approximation 公式來求得近似解，如公式(2.11)，關於詳細的公式說明請參閱文獻[15]，

$$tan^{-1}(\varepsilon) = \frac{\varepsilon}{1+0.28086\varepsilon^2} \quad , \quad -1 \le \varepsilon \le 1 \tag{2.11}$$

另外，由於菲涅耳轉換公式所求出的 ε 不一定符合公式(2.11)的條件 $-1 \le \varepsilon \le 1$，為解決此問題，本論文在進行反正切運算時，會將公式(2.11)、公式(2.12)及公式(2.13)搭配使用，其使用方法將在第三章詳細說明。

$$tan^{-1}(\varepsilon) = \frac{\pi}{2} - tan^{-1}(\frac{1}{\varepsilon}) \quad , \quad \varepsilon > 0 \tag{2.12}$$

$$tan^{-1}(\varepsilon) = -\frac{\pi}{2} - tan^{-1}(\frac{1}{\varepsilon}) \quad , \quad \varepsilon < 0 \tag{2.13}$$

總結以上的討論，本文所採用的菲涅耳轉換公式以及由轉換公式解求出相角，其運算流程以可歸納為以下步驟：

步驟零：給予輸入數位全像圖數值函數$\eta_{x,y}$，$0 \le x < N$ 且$0 \le y < N$。

步驟一：利用公式(2.5)，計算函數$\rho_{x,y}$。

步驟二：利用公式(2.6)，使用二維度傅立葉轉換計算$\tau_{u,v}$，二維度傅立葉轉換運算步驟如下：

步驟二之一：對於函數$\rho_{x,y}$中每一列 x 執行快速傅立葉轉換或離散傅立葉轉換，$0 \le x < N$，可得一運算結果陣列K_x，離散傅立葉轉換如公式(2.14)，。

$$K_x = \sum_{n=0}^{N-1}\left(\kappa_n \cos\frac{2\pi}{N}xn + j\cdot\kappa_n \sin\frac{2\pi}{N}xn\right) \quad (2.14)$$

步驟二之二：將陣列K_x去取代函數$\rho_{x,y}$中的第 x 個列向量。

步驟二之三：當$\rho_{x,y}$中每一個列向量都被取代之後，對於函數$\rho_{x,y}$中的每一行向量 y 執行步驟二之一到步驟二之二的動作。

步驟三：利用公式(2.9)，將二維度傅立葉轉換計算出的$\tau_{u,v}$，求出函數$\varepsilon_{u,v}$。

步驟四：使用公式(2.11)、(2.12)以及(2.13)，對函數$\varepsilon_{u,v}$作反正切函數運算，以求出相角$\zeta_{u,v}$，再交由相位展開演算法則電路繼續運算。

2.2 相位展開法則概述

相位展開演算法部分，本論文所採用的演算法是以快速傅立葉轉換（Fast Fourier Transform）為基礎的相位展開法則，此演算法所計算出的結果會是一個最小平方誤差解（minimum squared error solution），本節將針對此相位展開法則演算法[12]作基本運作流程介紹。

由於相位展開演算法中使用的輸入影像大小需為$(N+1)\times(N+1)$，但菲涅耳轉換時所使用的輸入及輸出影像長與寬皆為N，因此，在影像進行相位展開運算前，必須先將第$N+1$行及第$N+1$列補0，再進行相位展開運算。

接續上節所提到由反正弦函數計算出的相位值函數$\zeta_{u,v}$，其數值會落在$(-\pi,\pi]$，且同時滿足關係式 $e^{-j\zeta_{u,v}}=e^{-j\phi_{u,v}}$，在此$\phi_{u,v}$為相位展開後的實數函數。接著令$\psi_{u,v}$為函數$\zeta_{u,v}$利用鏡面反射（mirror reflection）技術所產生的週期函數，且滿足$0\le u\le 2N$和$0\le v\le 2N$，因此可將$\psi_{u,v}$詳細定義如下方程式(2.15)：

$$\psi_{u,v}=\begin{cases}\zeta_{u,v} & ,0\le u\le N,0\le v\le N\\ \zeta_{2N-u,v} & ,N\le u\le 2N,0\le v\le N\\ \zeta_{u,2N-v} & ,0\le u\le N,N\le v\le 2N\\ \zeta_{2N-u,2N-v} & ,N\le u\le 2N,N\le v\le 2N\end{cases} \tag{2.15}$$

接著定義$\Delta_{u,v}^{x}$與$\Delta_{u,v}^{y}$，如公式(2.16)，此公式表示壓縮相位差(wrapped phase difference)，也為週期性函數，因此壓縮相位差的值必須限制在$(-\pi,\pi]$，若運算

後的值超出範圍，則利用其週期函數的特性將數值加上$2n\pi$或減掉$2n\pi$，使之落於$(-\pi,\pi]$區間之中。

$$\Delta_{u,v}^{x}=\psi_{u+1,v}-\psi_{u,v}\quad,\quad\Delta_{u,v}^{y}=\psi_{u,v+1}-\psi_{u,v} \tag{2.16}$$

相位展開法則的目的是找到一周期函數$\bar{\phi}_{i,j}$，此函數為函數$\phi_{i,j}$的估計值，目的要最小化公式(2.17)的計算結果，

$$\Sigma_{u,v}(\bar{\phi}_{u,v+1}-\bar{\phi}_{u,v}-\Delta_{u,v}^{x})^{2}+\Sigma_{u,v}(\bar{\phi}_{u+1,v}-\bar{\phi}_{u,v}-\Delta_{u,v}^{y})^{2} \tag{2.17}$$

由公式(2.17)可知函數其實是一組最小平方誤差解（minimum squared error solution）。首先令$\gamma_{u,v}$為公式(2.18)，

$$\gamma_{u,v}=\Delta_{u,v}^{x}-\Delta_{u,v-1}^{x}+\Delta_{u,v}^{y}-\Delta_{u-1,v}^{y} \tag{2.18}$$

再將公式(2.17)微分後令結果等於零，得其結果如公式(2.19)

$$\left(\bar{\phi}_{u,v+1}-2\bar{\phi}_{u,v}+\bar{\phi}_{u,v-1}\right)+\left(\bar{\phi}_{u+1,v}-2\bar{\phi}_{u,v}+\bar{\phi}_{u-1,v}\right)=\gamma_{u,v} \tag{2.19}$$

由於$\bar{\phi}_{u,v}$和$\gamma_{u,v}$皆為週期性函數，所以可利用傅立葉轉換(Fourier Transform)來求得公式（2.19）的解$\bar{\phi}_{u,v}$，在本論文中則是使用餘弦轉換（Cosine Transform）來代替傅立葉轉換。對公式（2.19）等號兩邊的算式執行$N\times N$的二維度餘弦轉換運算，$\bar{\phi}_{u,v}$與$\gamma_{u,v}$可轉換成函數$\Phi_{m,n}$與函數$\Gamma_{u,v}$，並得到方程式(2.20)如下。

$$\Phi_{m,n}=\Gamma_{m,n}/(\left(2\cos\left(\frac{m\pi}{N}\right)+2\cos\left(\frac{n\pi}{N}\right)-4\right) \tag{2.20}$$

再根據公式（2.20），對函數$\Phi_{m,n}$執行二維度反餘弦轉換（2D-inverse Cosine Transform），即可算出欲求得的週期函數$\bar{\phi}_{u,v}$，最後只要把$\bar{\phi}_{u,v}$的範圍限制在$0\le u\le N$和$0\le v\le N$之間就可以得到最後還原後的相位結果$\phi_{u,v}$。

總結本節的討論，本文所採用的相位展開法則，其運算流程以可歸納為以下步驟：

步驟零：給予輸入影像相位數值函數$\zeta_{u,v}$，$0 \le u \le N$ 且$0 \le v \le N$。

步驟一：利用方程式（2.18），計算函數$\gamma_{u,v}$，$0 \le u \le N$ 且$0 \le v \le N$。

步驟二：使用二維度餘弦轉換函數（Cosine transform）計算$\Gamma_{u,v}$，$0 \le u \le N$ 且$0 \le v \le N$。二維度餘弦轉換運算如下：

步驟二之一：對函數$\gamma_{u,v}$中每一列u執行餘弦轉換，$0 \le u \le N$，如公式(2.21)，得到另一陣列Λ_u，$0 \le u \le N$。

$$\Lambda_u = \lambda_0 \cdot \cos\frac{\pi \cdot 0 \cdot u}{N} + 2\sum_{k=1}^{N-1} \lambda_k \cdot \cos\frac{\pi \cdot k \cdot u}{N} + \lambda_n \cdot \cos\frac{\pi \cdot N \cdot u}{N} \qquad (2.21)$$

步驟二之二：用陣列Λ_u去取代函數$\gamma_{u,v}$中的第u個行向量。

步驟二之三：當$\gamma_{u,v}$中每一個列向量皆被取代後，對$\gamma_{u,v}$中的每一個行向量執行步驟二之一到步驟二之二的動作。

步驟三：依據方程式（2.20）計算函數$\Phi_{m,n}$。

步驟四：對函數$\Phi_{m,n}$執行反餘弦轉換，可獲得還原後的相位數值函數$\bar{\phi}_{u,v}$。

2.3 討論菲涅耳轉換及相位展開於數位全像顯微鏡之應用

全像術主要分為紀錄、重建兩個步驟。傳統全像片通常須由底片拍攝而成，然而，早在40年前，Goodman、Laurence[21]就提出利用電腦紀錄及重建全像圖的想法，但當時受限於科技技術，電腦欠缺大量運算數值影像的能力，因此無法充分的表現還原結果，而現今由於電腦科技日趨成熟，可使用高效率且快速的電子式CCD/CMOS影像感測器取代傳統光學全像之記錄介質，並以數值計算方式重建出原來物體的完整波前(振幅與相位)。

若以肉眼觀察，全像片只是一張灰白條紋相間的圖片，這些條紋形狀與原物影像沒有任何幾何上的相似性，因此需要透過重建將影像還原出來，數位全像片其數值重建方法已有許多演算方式被提出，本論文即透過菲涅耳轉換搭配相位展開法則來重建一連續相位圖，其架構如圖(2.1)。

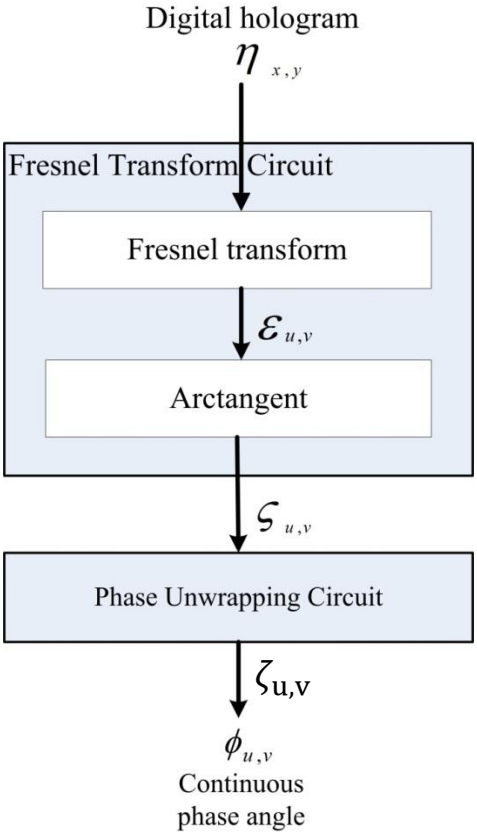

圖2.1 本論文重建全像圖流程

取得全像片平面的複數振幅資訊$\eta_{x,y}$之後，將之作為原始資料輸入至本章第一節所描述的菲涅耳轉換電路，藉由計算 Fresnel transform 公式獲得物體的複數振幅資訊$\varepsilon_{u,v}$，並利用反正切函數取得相位值$\zeta_{u,v}$，此時的相位值$\zeta_{u,v}$數值會被壓縮在$(-\pi, \pi]$之間，造成相位圖中產生不連續點(discontinuity)，使整張相位圖呈現非線性的分布，因此必須再利用本章第二節所描述的相位展開法則電路，將不連續點消除以重建出真實的相位$\bar{\phi}_{u,v}$。

2.4 FPGA 系統設計

近年來，積體電路(IC)產品的發展愈來愈快，然而卻造成其生命週期日益下降，所要求的功能複雜度也提高許多，為了滿足這些應用本身的挑戰，特別是隨著設計專案規模越來越大時，設計師們需要一個易於使用、靈活的設計環境來探索不同的設計實現，以達到其成本、功率消耗和性能指標，因此FPGA 越來越常出現在對成本敏感、功耗敏感的大量應用。

使用硬體描述語言(Hardware description language, HDL)搭配可重複程式設計的現場可程式化邏輯閘陣(FPGA，Field programmable gate array) 晶片設計數位電路，可縮短研發時間且更有彈性，經過簡單的合成繞線佈局，可快速重覆地燒錄至FPGA上進行測試，是現代IC設計的技術主流。目前FPGA系統的開發，允許軟體與硬體共同設計，來達成系統化的IC設計，具有開發時間短及系統可修改的優點。

Altera公司根據不同使用者的需求，提供許多不同系列的FPGA開發板，本論文是使用NIOS development kit中的Stratix III EP3SL150系統開發板來實現菲涅耳轉換及相位展開法則電路。

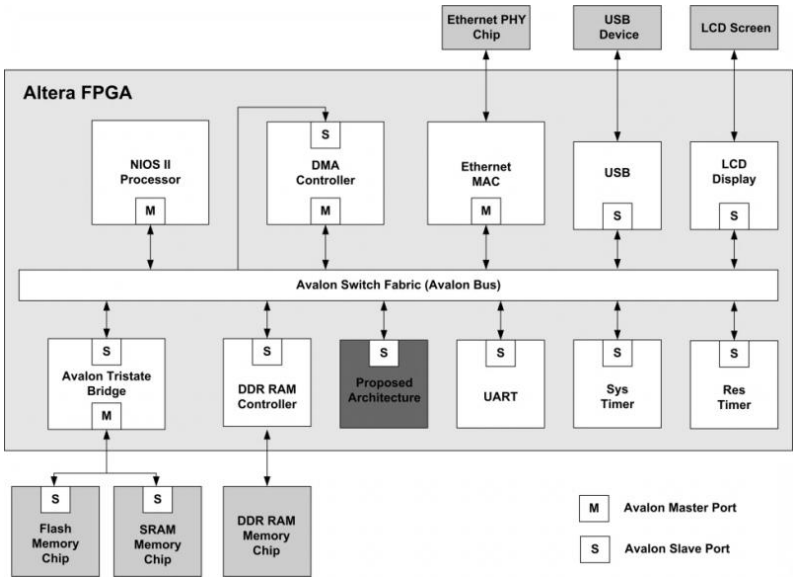

圖2.2 SOPC系統架構圖

圖2.2為一個基於NIOS系列嵌入式處理器的SoPC系統架構圖，在NIOS系統中提供了一套專門給NIOS處理器使用的匯流排(Avalon bus)，其交換架構是一種定制的內部互聯，它由SOPC Builder設計工具自動生成，將系統中所有的主設備和從設備端口連接在一起。開發人員設計出的硬體電路被視為一個客製化邏輯電路(custom logic circuit)，透過此匯流排上的各個訊號線，將電路掛在Avalon bus上與整個系統溝通。除此之外，Stratix III開發板上也提供DDRII SDRAM、flash memory、Ethernet controller與I/O 裝置等，供開發人員使用。

NIOS系統擁有下列優點：

1. Altera公司所提供的NIOS Ⅱ IDE是一個視窗介面的開發工具，設計者可於其上編輯程式碼，更可編譯及除錯及觀察程式執行結果。此程式除了包含所有C 語言的函式庫之外，還有HAL(hardware abstraction layer)函式庫。其中HAL函式庫提供設計者一個呼叫系統相關裝置API(Application Program Interface)，

如圖2.3所示，設計者可用C 語言撰寫基本程式並透過HAL API呼叫來讓特定的裝置運作。

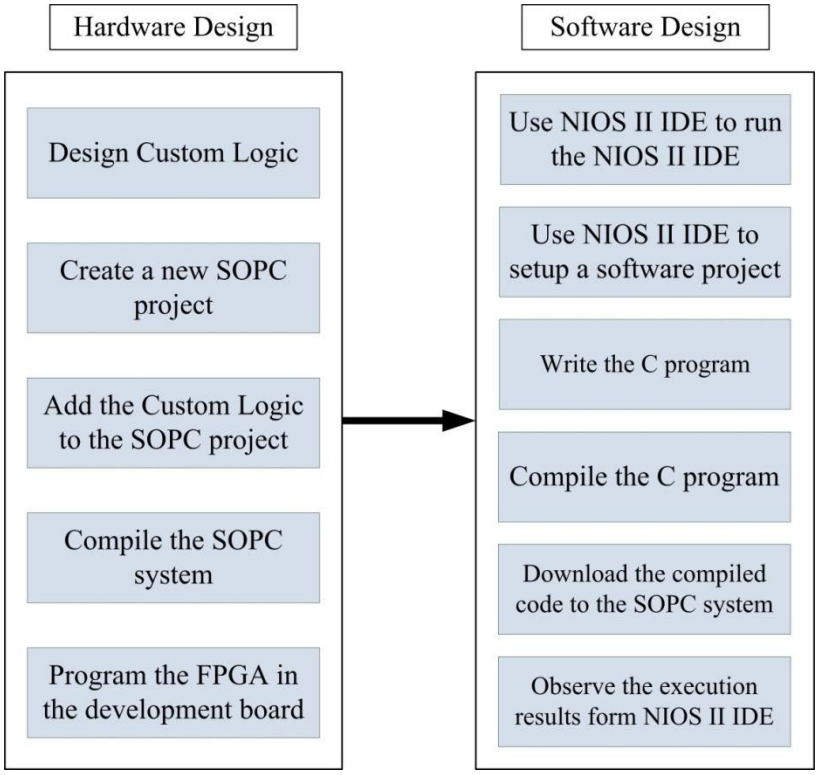

圖2.3 軟硬體共同設計圖

2. 提供了Compact Flash卡(以下簡稱CF卡)存取功能，讓設計者可以將資料置於CF卡中，並透過HAL API讓設計者進行讀與寫的動作。

3. 提供DMA(direct memory access)機制讓設計者可加速資料在記憶體與系統周邊元件的傳輸速度且不佔用CPU資源。

4. lwIP提供基本的網路功能，除了符合嵌入式系統最基本的簡單、快速、不佔據過多系統資源之外，也讓設計者更容易修改或增刪想要的功能。

設計者可依自己的需求增加或刪除想要的電路功能，而增刪裝置也非常簡便，重新建置系統並編譯成硬體檔之後燒至板子即可。

第三章 系統架構

本章節將詳細介紹本論文所提出的菲涅耳轉換及相位展開法則硬體架構。本系統架構主要分成七個部分：嵌入式記憶體(on-chip RAM)、菲涅耳轉換前單元（Fresnel pre-transform unit）、傅立葉轉換單元（Fresnel Fourier Transform Unit）和菲涅耳轉換後單元（Fresnel post-transform unit）、相位展開法則換前單元（Phase unwrapping pre-transform unit）、餘弦轉換單元（Phase unwrapping cosine transform unit）和相位展開法則轉換後單元（Phase unwrapping post-transform unit）。接下來的各節將會依序對每個單元的硬體架構進行詳細描述。

3.1 菲涅耳轉換及相位展開法則之硬體電路架構

首先介紹整體的系統架構，本系統將電路分成兩大部分：菲涅耳轉換電路及相位展開法則電路。而菲涅耳轉換電路架構又包含三個部分：轉換前單元（Fresnel pre-transform unit）、傅立葉轉換單元（Fresnel Fourier Transform Unit）和轉換後單元（Fresnel post-transform unit）。相位展開法則電路架構中也包含三個部分：轉換前單元（Phase unwrapping pre-transform unit）、餘弦轉換單元（Phase unwrapping cosine transform unit）和轉換後單元（Phase unwrapping post-transform unit）。另外包含一塊嵌入式記憶體(on-chip RAM)由兩大塊電路所共用。

本系統架構中，on-chip RAM 的用途有三個，第一為儲存各個運算單元的資料輸入來源，其次為儲存計算過程中的暫時性資料，最後是儲存計算完成的最終結果。

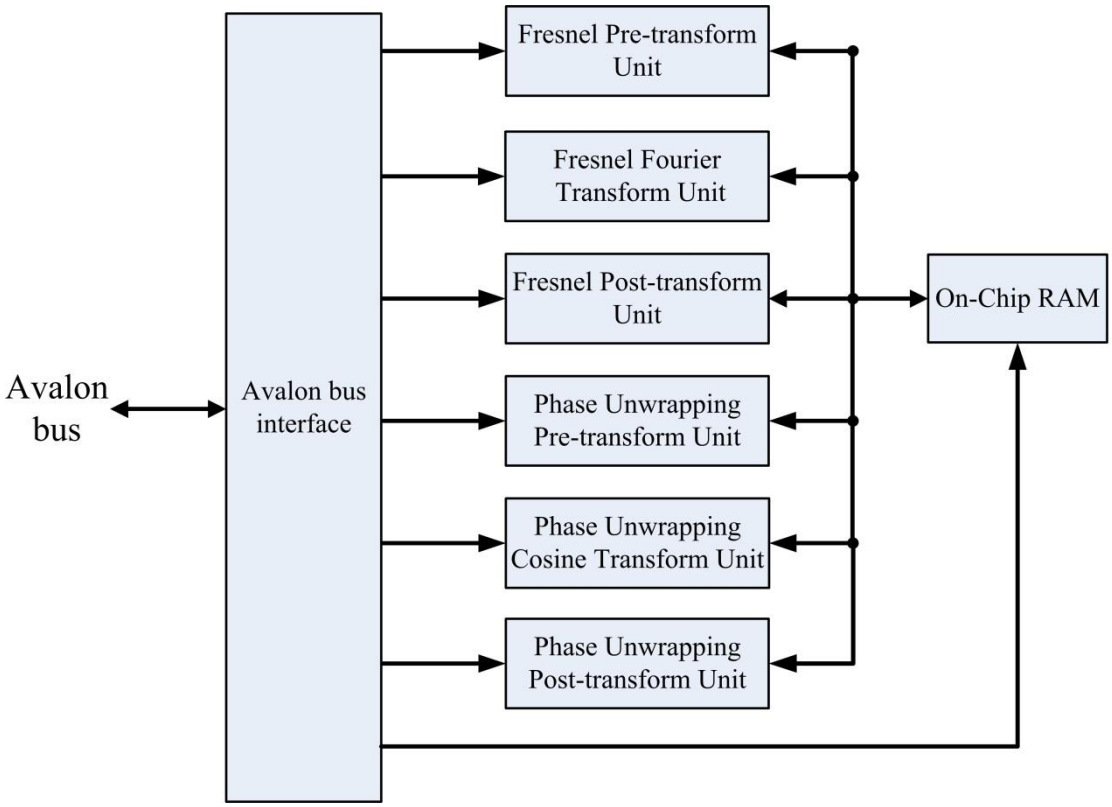

圖 3.1 菲涅耳轉換及相位展開法則硬體電路架構圖

在菲涅耳轉換電路中，轉換前單元從on-chip RAM中讀取數位全像片資料$\eta_{x,y}$，並根據2.1節中的公式(2.4)及公式(2.5)進行運算得到函數$\rho_{x,y}$，接著傅立葉轉換單元根據公式(2.6)計算得到函數$\tau_{u,v}$，最後，轉換後單元由公式(2.7)、公式(2.8)和公式(2.9)算出複數值點$\varepsilon_{u,v}$，經由反正切函數運算得相角ζ，並將相角ζ存入on-chip RAM中，再交由相位展開電路繼續運算。

相位展開電路中，轉換前單元從 on-chip RAM 中讀取菲涅耳轉換電路的計算結果$\zeta_{u,v}$，並根據 2.2 節中公式(2.15)、公式(2.16)和公式(2.18)進行運算，產生壓

縮相位差（wrapped phase difference）之和的結果$\gamma_{u,v}$，接著餘弦轉換單元由公式(2.21)的餘弦轉換公式計算出$\Gamma_{u,v}$，轉換後單元再依其結果計算出另一中間值$\Phi_{m,n}$，最後做反傅立葉轉換計算，得到欲求出的相位函數$\phi_{u,v}$，此相位展開法則演算法中的所有函數皆以二維陣列資料的形式來表示。圖 3.1 中的每個單元會在接下來的小節中討論。

3.2 嵌入式記憶體(On-chip RAM)

在本文所提出的系統架構中，on-chip RAM會由兩塊RAM 2-port module所組成，分別是on-chip RAM1及on-chip RAM2，並且用來儲存複數資料的實部值及虛部值。此二塊on-chip RAM都是大小為$(N+1)\times(N+1)$的記憶體陣列空間，主要存放各個運算單元的資料輸入來源以及儲存計算途中的暫時性資料，或是儲存計算完成的最終結果。

若不使用on-chip RAM而是利用電路外部SoPC系統上的資源如DDR2 SDRAM或Flash等元件作為資料儲存裝置，由於每次要從這些off-chip裝置讀取或寫入資料時，都需要經由NIOS CPU執行一次讀取或寫入指令才能完成，加上讀取與寫入耗費時間冗長，無法保證在一個時脈週期(clock cycle)內就能執行完成，因此，若要處理大量的資料時，傳輸部分所花費的時間將會非常可觀，有鑑於此，在本論文中選擇使用on-chip RAM作為儲存裝置，將記憶體空間放置在可程式化邏輯閘陣列(FPGA)晶片當中，使得CPU不需要額外參與資料的傳輸，也有效的減少本系統架構中電路元件從記憶體中讀寫資料所需要花費的時間。

在各運算單元中，用有各自的控制器(controller)和位址產生器(address generator)來自動產生對應控制訊號及記憶體位址與on-chip RAM進行資料存取溝通，但由於on-chip RAM的資料傳輸埠(port)及位址傳輸埠都只有一個，而欲輸入來源卻有多個，分別來自於菲涅耳轉換前單元、菲涅耳傅立葉轉換單元等單元，

因此，必須在on-chip RAM的資料傳輸埠以及記憶體位址傳輸埠前加上多工器，讓資料傳輸得以正確運作，如圖3.2所示；另外，利用SoPC 系統中的NIOS II CPU可以設定狀態暫存器(status register)內的數值，多工器即可根據狀態暫存器內儲存的數值來選擇對應的訊號，避免錯誤的資料值被寫入on-chip RAM中。

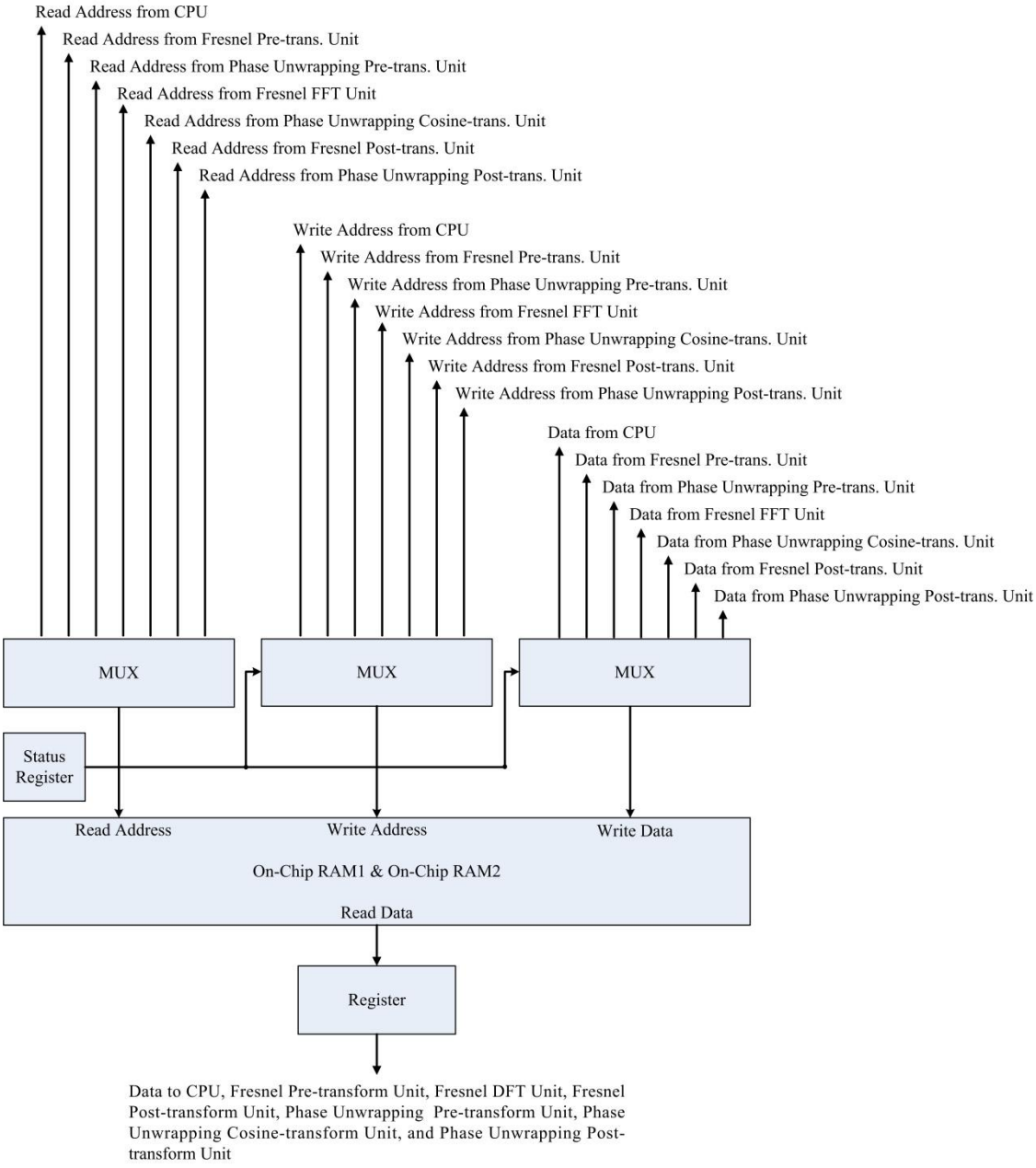

圖 3.2 On-chip RAM 硬體電路架構圖

3.3 菲涅耳轉換之轉換前單元(Pre-transform Unit)及轉換後單元(Post-transform Unit)

本節將介紹菲涅耳轉換前單元及轉換後單元，這兩個單元是以硬體電路實現菲涅耳轉換電路的步驟一及步驟三，因為此二單元電路架構相似，因此在本節共同討論。

首先介紹菲涅耳轉換前單元，其硬體架構如圖 3.3 所示，包含位址產生器、控制器、多工器，指數表(Exponential table)，以及兩個複數乘法模組(Complex number multiplication)。其中位址產生器輸出的記憶體位址用來讀取 on-chip RAM 的資料以及將運算結果存回其相對應的位置，另外記憶體位址也會輸入指數表進行查表的動作。

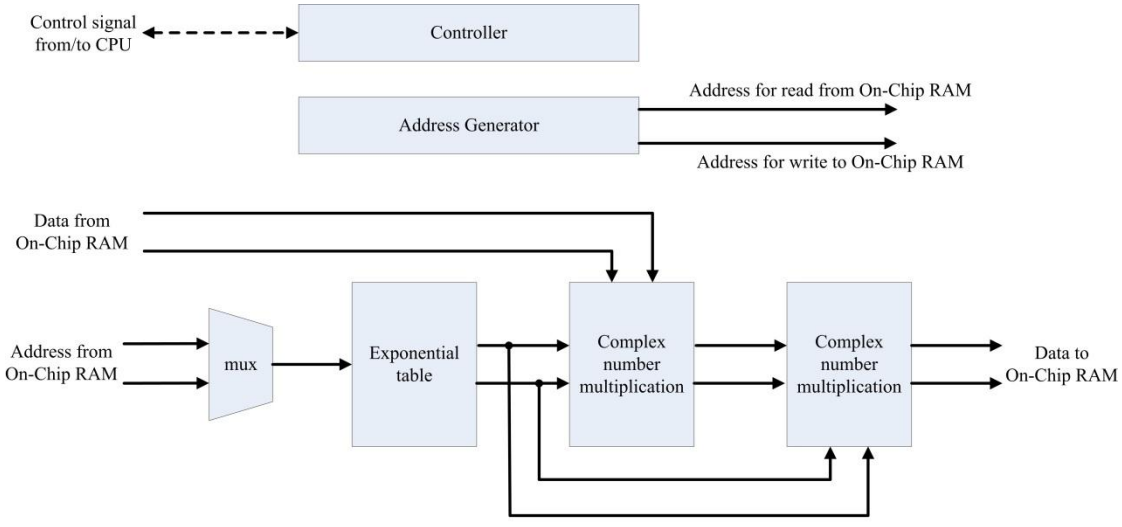

圖 3.3 菲涅耳轉換前單元硬體電路架構圖

根據公式(2.4)可知，μ_x及μ_y為相同的指數週期函數，根據不同的$\eta_{x,y}$所提供

的位址索引x和y至指數表取得對應的值，由於$0 \le x < N$且$0 \le y < N$，x和y的數值只有N種可能，因此可建構一個具有N個entry的指數表，當索引值傳入指數表後，指數表就可以根據所輸入的數值，將對應的指數值輸出。另外，因為x和y所查的都是同一張表，因此在指數表之前加上一個多工器，利用控制器給予多工器訊號得以選擇來源。

由於全像片平面的振幅資訊$\eta_{x,y}$皆為複數形式，因此在相乘時要做複數乘法，複數乘法的通解如公式(3.1)，

$$(A + Bj)(C + Dj) = (AC - BD) + (AD + BC)j \tag{3.1}$$

根據公式(3.1)的解設計出的電路架構如圖3.4，在此電路中需要四個乘法器搭配兩個加法器來完成複數乘法計算。

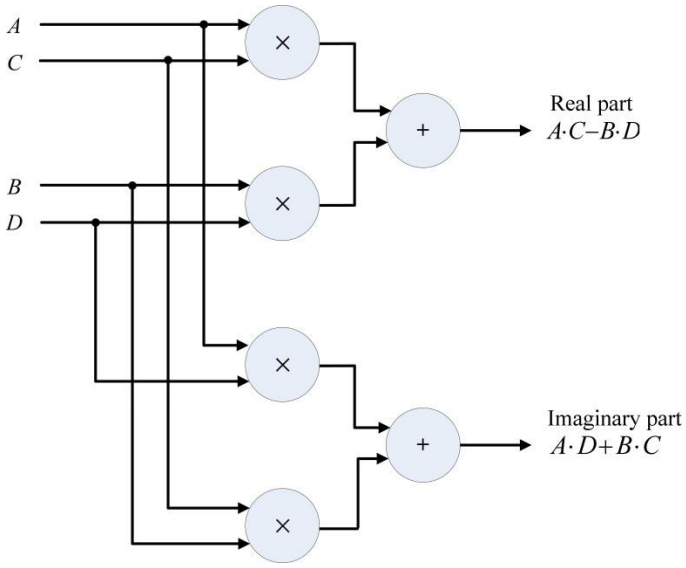

圖3.4 複數乘法器硬體電路架構

圖 3.5 及圖 3.6 為菲涅耳轉換前單元執行時的時序圖。當位址產生器產生位址 *x,y* 後，*x,y* 便會輸入至 on-chip RAM 中讀取對應資料$\eta_{x,y}$，此外，*x,y* 也會送至

多工器當成指數表的索引值，而控制器會先給予多工器訊號選擇索引 x，經由指數表查出對應的值μ_x，再將μ_x與$\eta_{x,y}$進行複數乘法得到$\eta_{x,y} \cdot \mu_x$，如圖 3.5 所示。

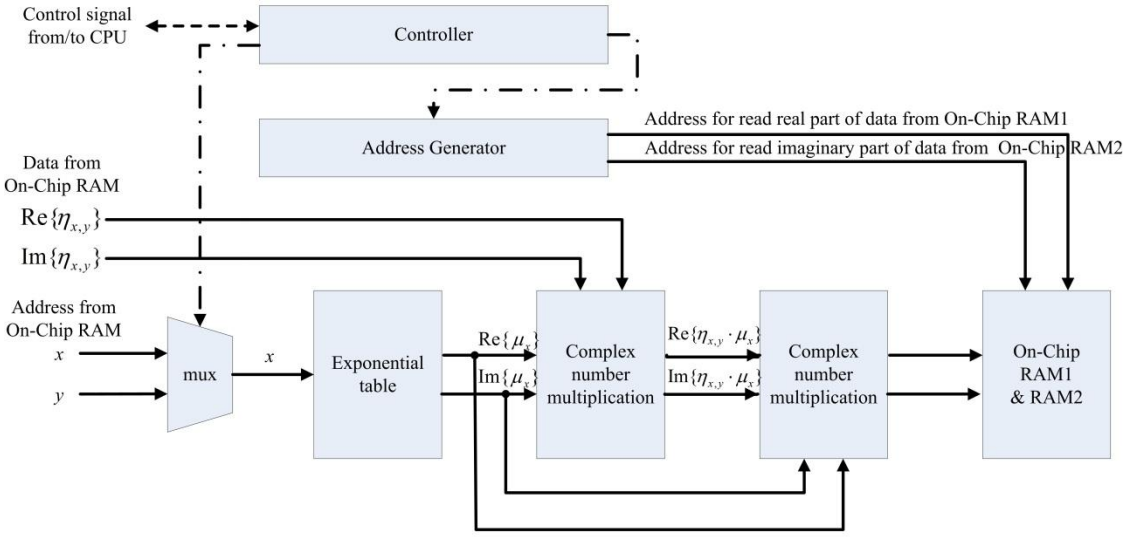

圖 3.5 菲涅耳轉換前單元資料流動時序圖(1)

接著，如圖 3.6，控制器再給予多工器訊號選擇索引 y，一樣經由指數表查出對應值μ_y之後，與上一步驟的複數乘法求出的值相乘，得到$\rho_{x,y}$，並將其結果的實部與虛部分別儲存到 on-chip RAM1 及 on-chip RAM2。

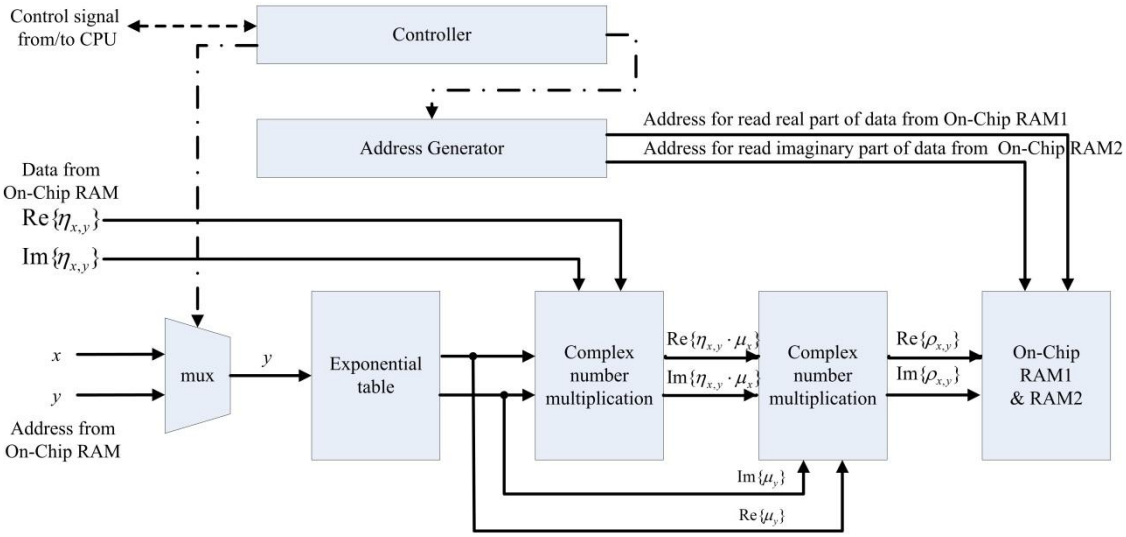

圖 3.6 菲涅耳轉換前單元資料流動時序圖(2)

菲涅耳轉換後單元的架構類似於菲涅耳轉換前單元，圖 3.7 為轉換後單元的架構圖，其包含位址產生器、控制器、多工器，指數表(Exponential table)、三個複數乘法模組(Complex number multiplication)，以及反正切函數模組(Arctan circuit)。其中記憶體位址也是用來讀取 on-chip RAM 的資料，將運算結果存回其相對應的位置，以及輸入指數表進行查表的動作。

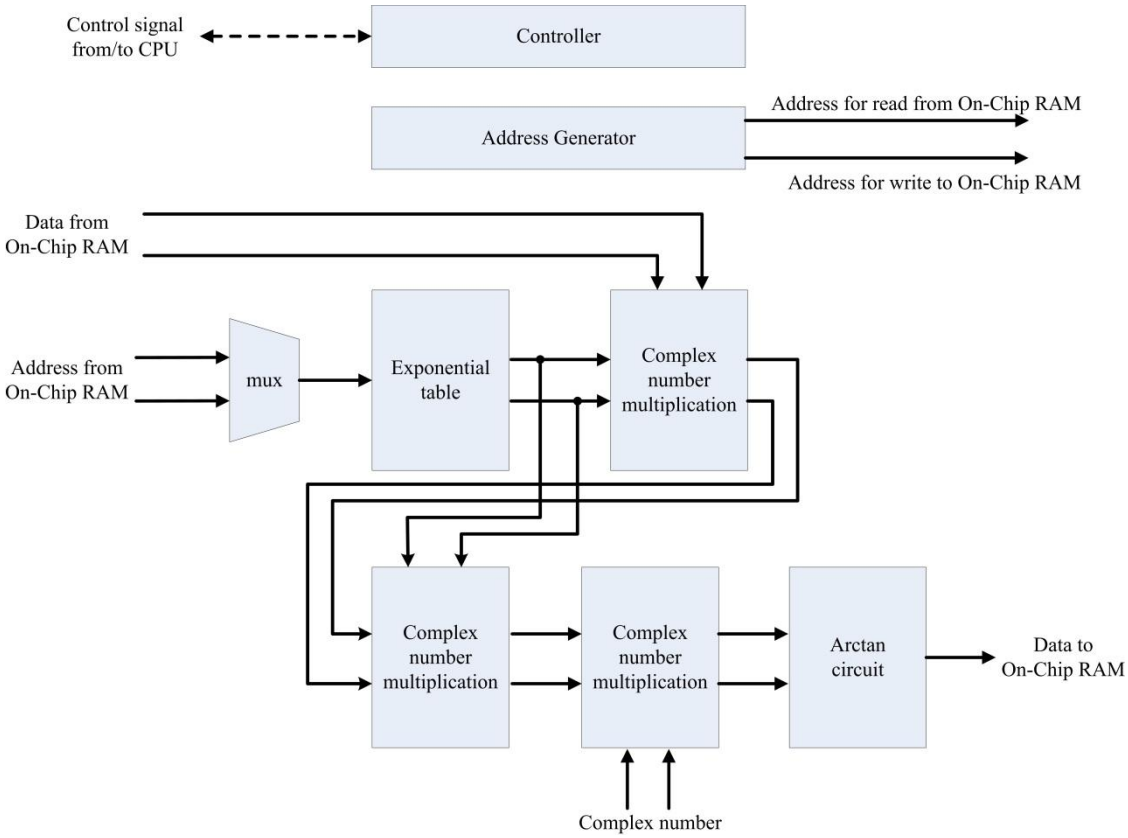

圖 3.7 菲涅耳轉換後單元硬體電路架構圖

根據公式(2.8)可知，轉換後單元也有兩個一樣的指數函數ω，u 和 v 都介於 0 至 N 之間，因此可以依公式製作一個 N 個 entry 的指數表，並藉由控制器來控制多工器輪流選擇索引值 u 或 v。

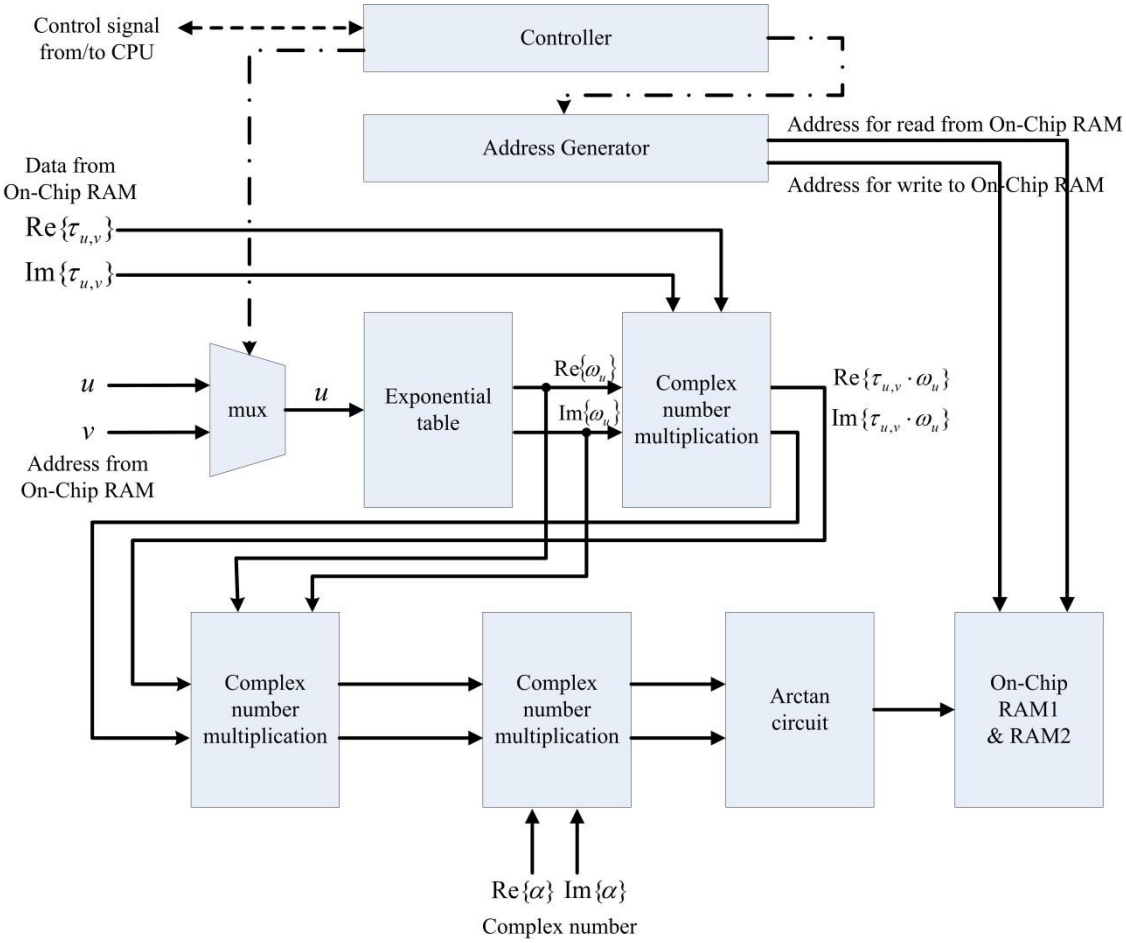

圖 3.8 菲涅耳轉換後單元資料流動時序圖(1)

圖 3.8 及圖 3.9 為菲涅耳轉換後單元執行時的時序圖。流程與菲涅耳轉換前單元類似，當位址產生器產生位址 u, v 時，一方面將位址送到 on-chip RAM 中讀取資料，另一方面也送到電路中的多工器當作索引值；控制器首先給予多工器訊號選擇索引 u，經指數表查出對應值ω_u，與 on-chip RAM 取得的資料$\tau_{u,v}$一起送入複數乘法模組作相乘，如圖 3.8，接著控制器再給予多工器訊號選擇索引 v，查出指數值ω_v，與前一步驟相乘的結果再做一次乘法，得到值$\omega_u \cdot \omega_v \cdot \tau_{u,v}$，緊接著再與定值 α 做第三次乘法，算出菲涅耳公式的解$\varepsilon_{u,v}$，如圖 3.9。

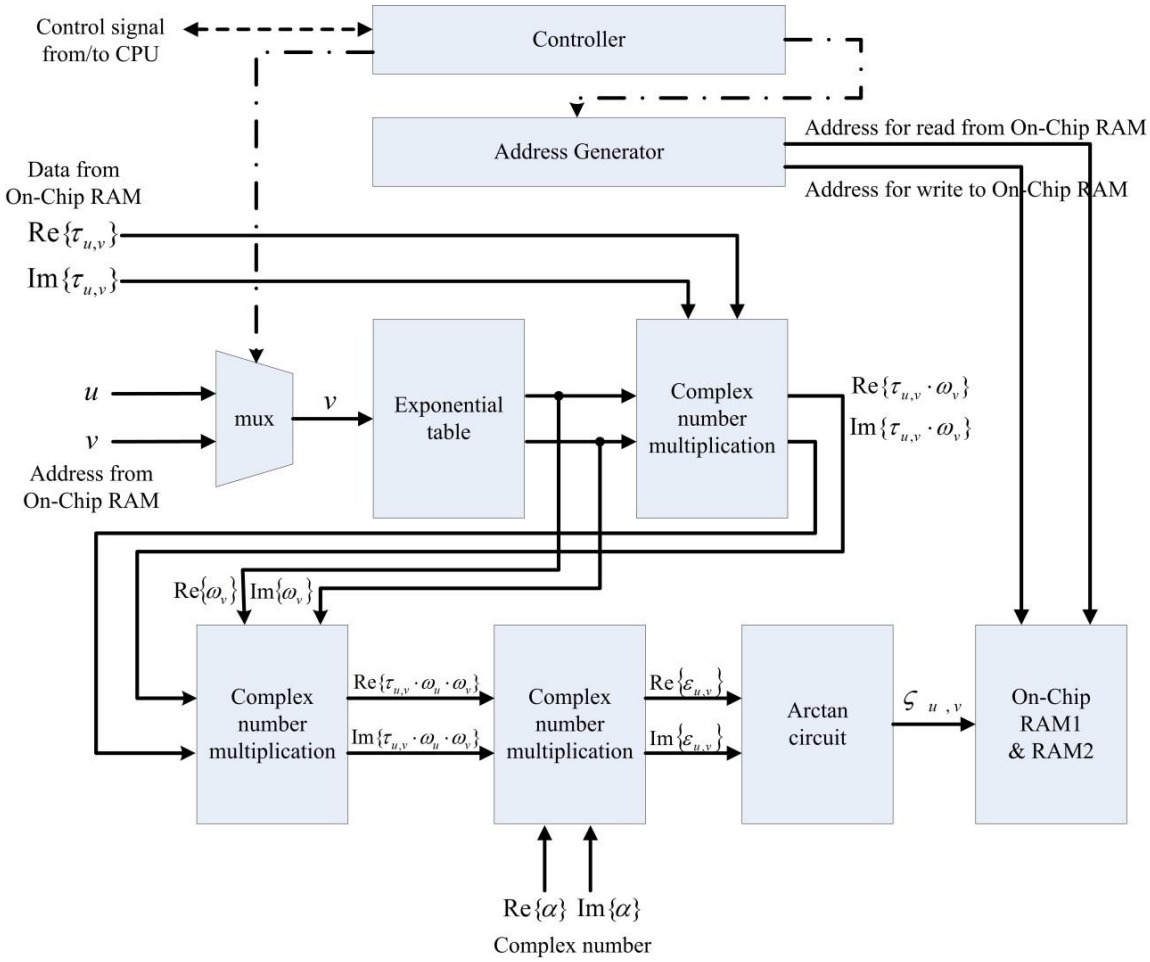

圖 3.9 菲涅耳轉換後單元資料流動時序圖(2)

觀察圖 3.9，在算出菲涅耳公式的解$\varepsilon_{u,v}$之後，根據公式(2.10)，還要作反正切函數運算來取得相角ζ，將複數數值$\varepsilon_{u,v}$的虛部值除以實部值可得其正切函數(tangent)的值ε，再對ε作反正弦函數運算得到的就是欲輸入相位展開電路的相角ζ。此部分則交由反正切函數電路(Arctan circuit)來運算，如圖 3.10，運算完的結果才會存回 on-chip RAM1 及 on-chip RAM2 之中。

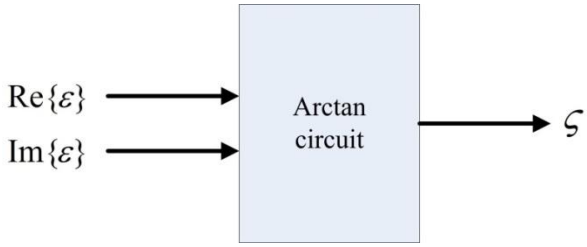

圖 3.10 反正切函數電路架構圖

圖 3.11 為反正切函數電路內部的架構圖，包含絕對值比較器(Absolute value compare)、符號位元擷取器(sign bit extraction)、資料交換器(Data exchange)、除法器(Divider)、反正切函數運算(Arctan function)以及反正切函數輸出值分析器(Arctan output analyse)，而此電路與菲涅耳轉換後單元共用位址產生器及控制器。

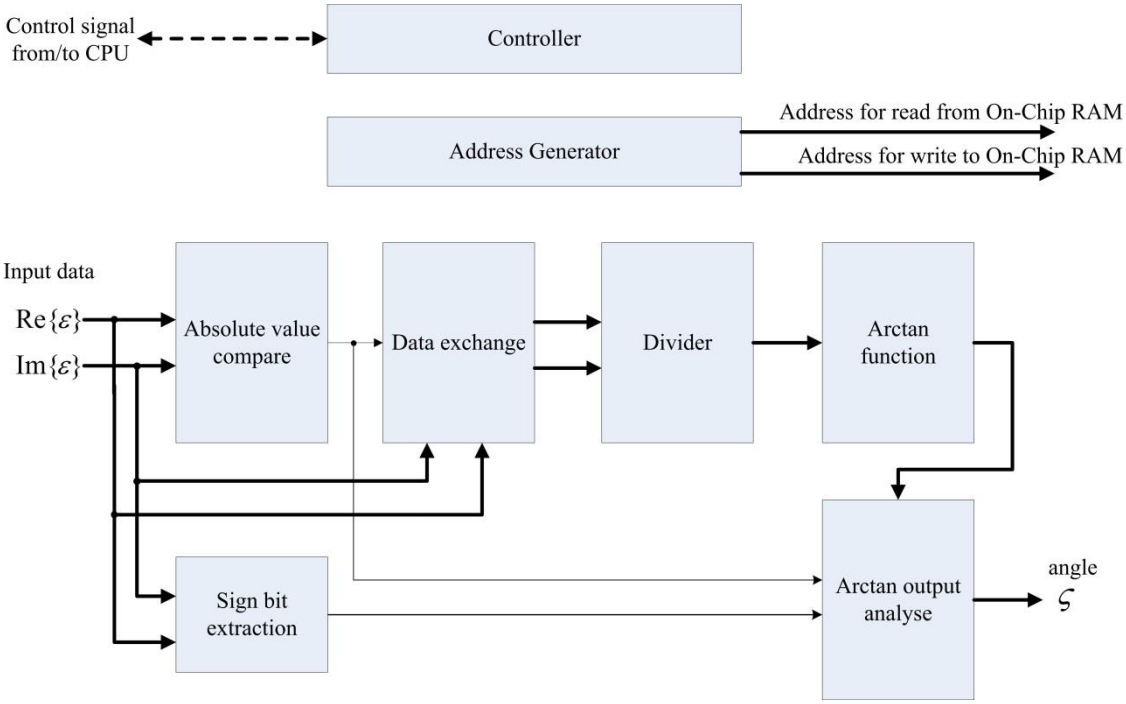

圖 3.11 反正切函數電路內部硬體電路架構圖

反正切函數電路是將公式(2.11)、公式(2.12)及公式(2.13)整合在一起之後進行實作。反正切函數運算單元的電路設計如圖 3.12，使用兩個乘法器，以及加法器與除法器各一，計算出的值即為公式(2.11)所求；

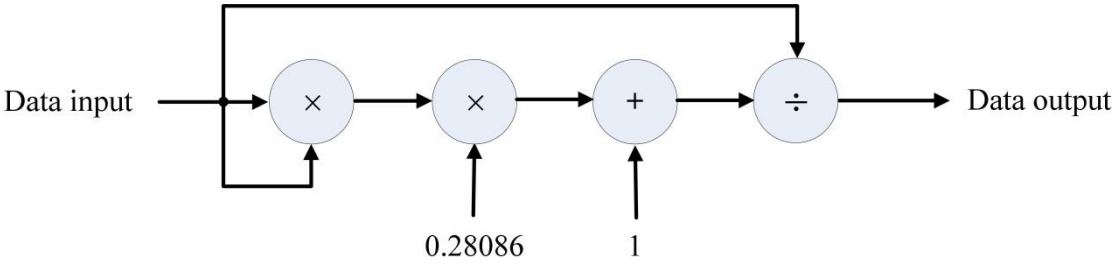

圖 3.12 反正切函數模組內部硬體電路架構

因為公式(2.11)中ε值必須介於-1 到 1 間，為了使超出範圍的ε值也可以運算，另搭配兩個反正切函數的倒數參數(Reciprocal arguments)關係式作為輔助，若$\varepsilon > 1$則使用公式(2.12)，若$\varepsilon < -1$則使用公式(2.13)，兩者都是先將ε倒數值經由反正切函數電路作運算之後，再加上一常數來獲得最後相角值ζ。

表 3.1 反正切函數公式輸入值對應除法器輸入選擇

實部值 >虛部值	ε 範圍	選擇反正切函數公式	反正切函數輸入值	被除數	除數
T	$-1 \le \varepsilon \le 1$	$\frac{\varepsilon}{1+0.28086\varepsilon^2}$	ε	虛部值	實部值
F	$\varepsilon > 1$	$\frac{\pi}{2} - tan^{-1}(\frac{1}{\varepsilon})$	$\frac{1}{\varepsilon}$	實部值	虛部值
F	$\varepsilon < -1$	$-\frac{\pi}{2} - tan^{-1}(\frac{1}{\varepsilon})$	$\frac{1}{\varepsilon}$	實部值	虛部值

判斷 ε 落在那一區間的方法是將 ε 值的實部值及虛部值取絕對值後輸入至絕對值比較器中，比較器會輸出一訊號表示實部值是否有大於虛部值，參考表 3.1，若實部值比較大表示$-1 \le \varepsilon \le 1$，則可使用公式(2.11)，利用資料交換器設定被除數及除數，其架構如圖 3.13，再把值傳給除法器，將虛部值除以實部值得到 ε；若虛部值比較大表示$\varepsilon > 1$或$\varepsilon < -1$，則可能使用公式(2.12)或公式(2.13)兩者之一，此時反正切函數輸入的是倒數值，則利用除法器將實部值除以虛部值取得$\frac{1}{\varepsilon}$。

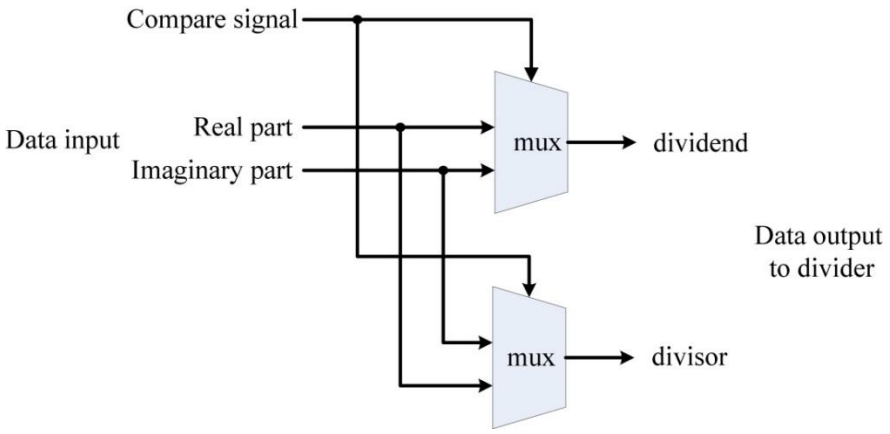

圖 3.13 資料交換器內部硬體電路架構

利用此機制，改變被除數及除數位置，只需使用一個除法器作一次除法，即可得到ε或$\frac{1}{\varepsilon}$，而非先算出ε再作第二次除法取得$\frac{1}{\varepsilon}$，此方法可以避免多餘資源浪費也可避免在計算倒數時增加 critical path 長度。

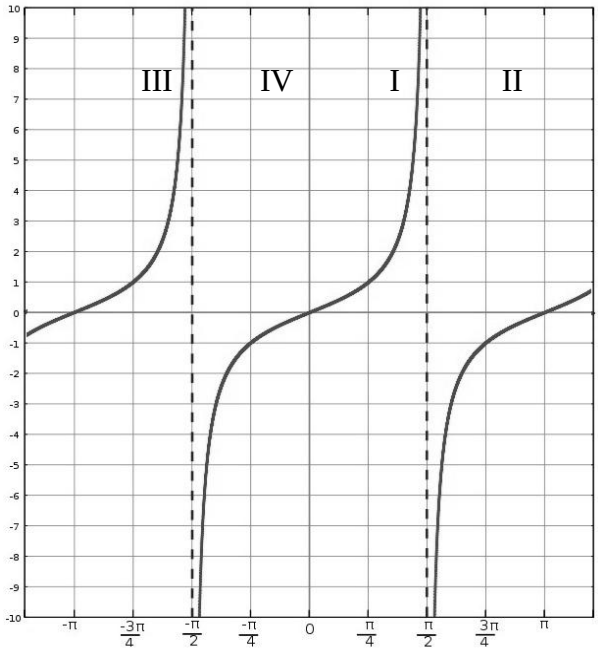

圖 3.14 正切函數圖形

反正切函數電路計算完成後，必須根據所選擇公式來加上常數值，以及根據ε所在的座標象限進行調整，觀察如圖 3.14 可發現，因反正切函數的值域為$\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$，但是相位展開電路所輸入的相位角$\zeta$其值域為$(-\pi,\pi]$，因此必須將座標位於第二

象限的值加上π平移到$\left(-\frac{\pi}{2},\pi\right]$，而座標位於第三象限的值則必須減掉$\pi$平移到$\left[-\pi,-\frac{\pi}{2}\right)$。

表 3.2 根據象限對應平移加減值

實部值	虛部值	象限	平移加減值
+	+	I	+0
+	-	II	+π
-	-	III	−π
-	+	IV	+0

綜合以上敘述，可整理出表 3.3，反正切函數單元計算完之後可能會有五種不同的加減值，分別為$+0$、$+\frac{\pi}{2}$、$-\frac{\pi}{2}$、$+\pi$及$-\pi$，反正切函數輸出值分析器會接受反正切函數電路計算出的值，並根據符號位元擷取器所取得的實部值及虛部值的正負號，搭配絕對值比較器的比較結果，三種訊號共有八種可能，依表 3.3 對應到不同加減值。

圖 3.15 為反正切函數輸出值分析器的電路架構，每次有資料輸入時，都會平行的將五種不同的加減值計算完成傳到多工器，再利用輸入的 sign bits 選擇對應的輸出值，即可得相角ζ。由於相角為一實數值，不會包含虛部的部分，因此將相角ζ存回 on-chip RAM1，而此時 on-chip RAM2 則存入數值 0，在相位展開電路部分只處理相角部分，因此不會再用到 on-chip RAM2。

表 3.3 反正切函數加減值統整表

實部值	虛部值	實部值 > 虛部值	所在象限及ε範圍	選擇反正切函數公式	求出反正切函數之後須加上的常數值	平移加減值	最後加減值
+	+	T	第 I 象限 $\varepsilon \le 1$	公式 2.11	$+0$	$+0$	$+0$
+	+	F	第 I 象限 $\varepsilon > 1$	公式 2.12	$+\frac{\pi}{2}$	$+0$	$+\frac{\pi}{2}$
-	+	T	第 II 象限 $\varepsilon \ge -1$	公式 2.11	$+0$	$+\pi$	$+\pi$
-	+	F	第 II 象限 $\varepsilon < -1$	公式 2.13	$-\frac{\pi}{2}$	$+\pi$	$+\frac{\pi}{2}$
-	-	T	第 III 象限 $\varepsilon \le 1$	公式 2.11	$+0$	$-\pi$	$-\pi$
-	-	F	第 III 象限 $\varepsilon > 1$	公式 2.12	$+\frac{\pi}{2}$	$-\pi$	$-\frac{\pi}{2}$
+	-	T	第 IV 象限 $\varepsilon \ge -1$	公式 2.11	$+0$	$+0$	$+0$
+	-	F	第 IV 象限 $\varepsilon < -1$	公式 2.13	$-\frac{\pi}{2}$	$+0$	$-\frac{\pi}{2}$

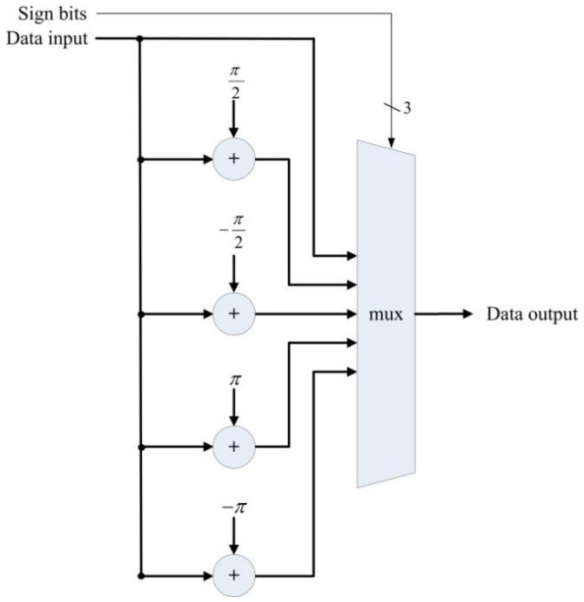

圖 3.15 反正切函數輸出值分析器內部硬體電路架構

3.4 傅立葉轉換單元(Fourier Transform Unit)及離散餘弦轉換單元(DCT Unit)

在本節中將介紹傅立葉轉換單元以及離散餘弦轉換單元，傅立葉轉換單元為硬體實現菲涅耳轉換電路中的步驟二，在此論文中實作了離散傅立葉轉換(DFT)以及快速傅立葉轉換(FFT)兩種架構。圖 3.16 為離散傅立葉轉換單元的硬體架構圖，由控制器、位址產生器、一維的離散傅立葉模組(1D-DFT module)和緩衝暫存器(Buffer register)所組成。

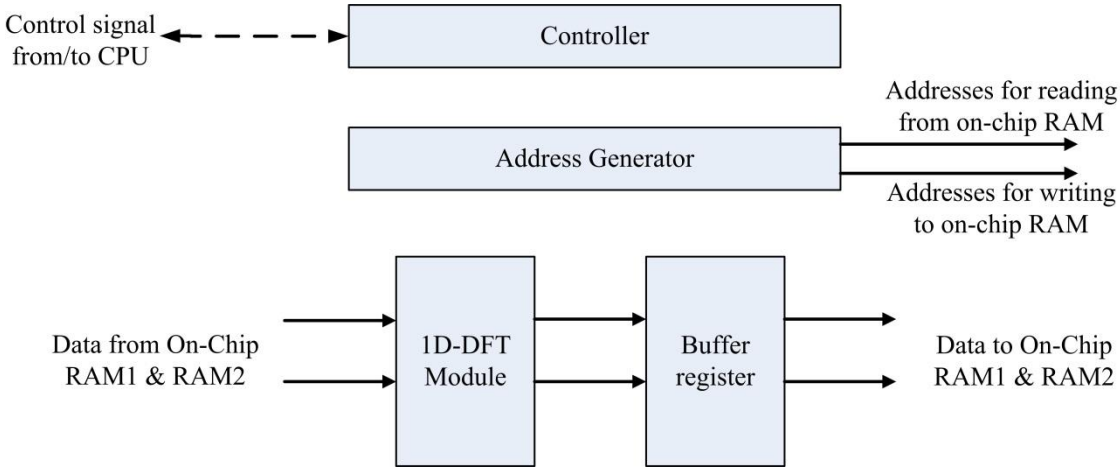

圖 3.16 離散傅立葉轉換單元硬體電路架構圖

依菲涅耳轉換電路運算流程的步驟二，離散傅立葉轉換單元會利用一維的離散傅立葉模組對函數$\rho_{x,y}$進行運算，在此過程中會先執行列運算，再執行行運算，在進行列運算時，每一列運算完的值會先儲存在緩衝暫存器之中，緩衝暫存器即為公式(2.14)中的K_x，直到一整列都計算完之後，才一次連續地將緩衝暫存器K_x中的值存入 on-chip RAM1 及 on-chip RAM2 中，取代原本的ρ_x，接著再對下一列進

行相同的列運算，當執行完所有列方向運算之後，再以相同方法對每一行做行運算，直到行運算執行結束之後，儲存在 on-chip RAM1 及 on-chip RAM2 之中的就是函數$\rho_{x,y}$經過二維傅立葉運算之後的值$\tau_{u,v}$，如圖 3.17。

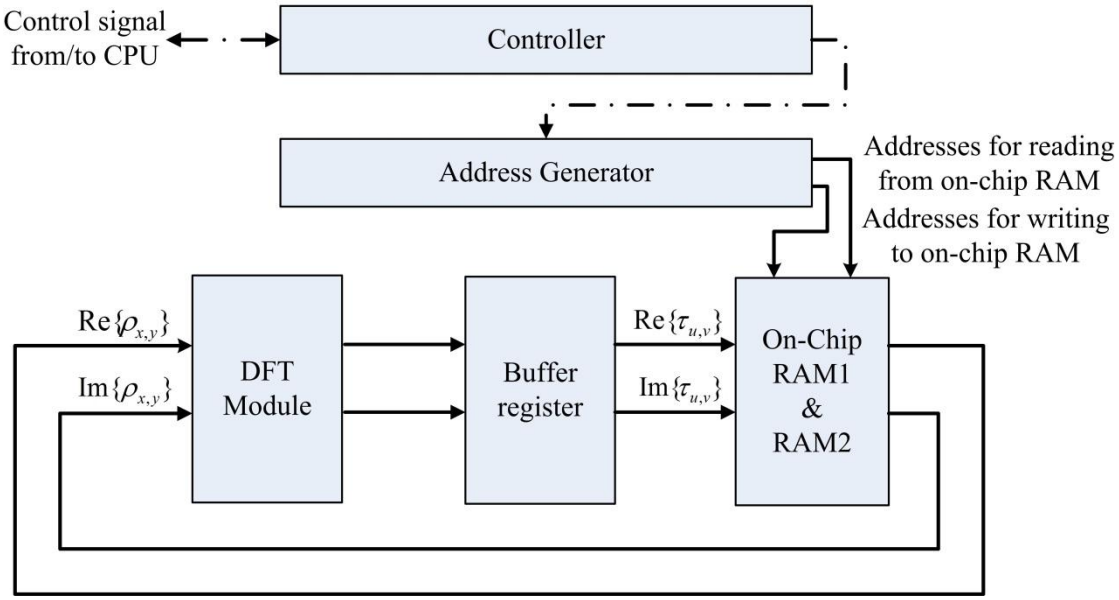

圖 3.17 離散傅立葉轉換單元資料流動時序圖

圖 3.18 為一維的離散傅立葉模組，此電路用來實現公式(2.14)，內部包含一個乘法器、正弦值表(Sine table)及餘弦值表(Cosine table)各一、一個複數乘法器以及兩個累加器(Accumulator)。而累加器內部架構如圖 3.19，是由一個加法器和一個暫存器所組成；加法器求出的值同時將訊號輸出，以及拉回輸入端當作輸入值之一，與另一值相加後可以達到累加作用。

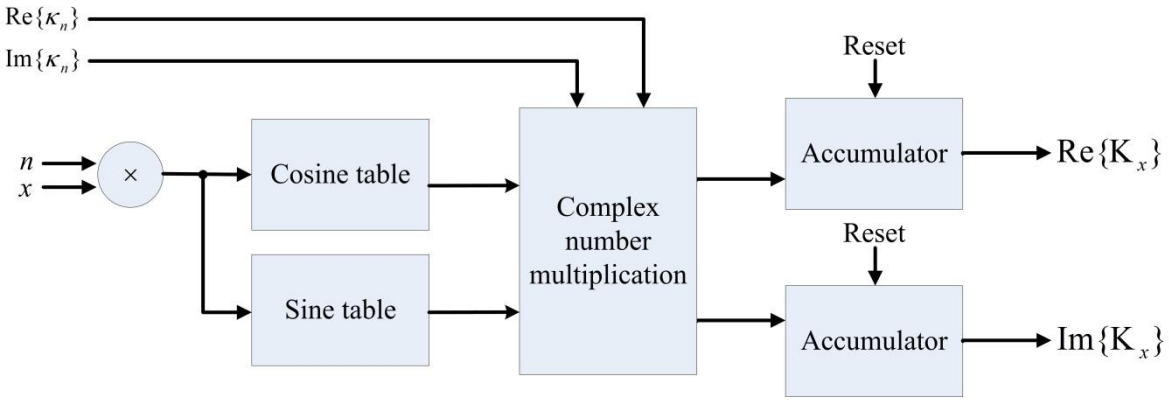

圖 3.18 一維的離散傅立葉模組內部硬體電路架構圖

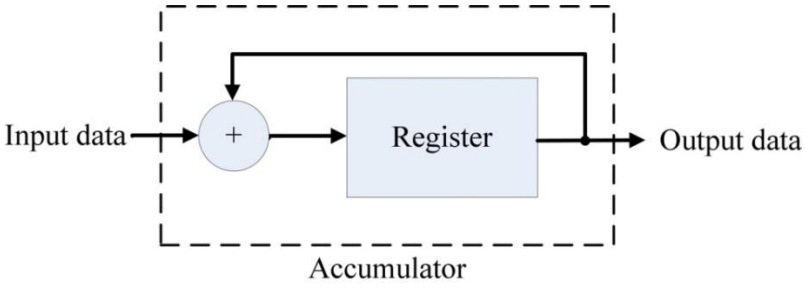

圖 3.19 累加器內部電路圖

根據公式(2.14)，先將輸入的 n 及 x 相乘後當成索引值，並利用此索引查出正弦值表和餘弦值表中對應的正弦值和餘弦值，因正弦和餘弦函數都是週期函數，因此，若$xn > N$，則對xn取 N 的餘數進行查表，查出來的值再與κ_n值相乘，接著交由累加器累加；計算每一列的第 x 個值K_x時，都會讀取到此列所有κ_n，$0 \le n < N$，換句話說，算每一筆資料都要累加 N 次，計算長度 N 的資料時，每一筆資料會被重複讀取到 N 次，位址產生器會重複產生 on-chip RAM 中某一列的記憶體位址共 N 次，因此，再算出某一列中的第 x 筆資料時，不能立即就取代 on-chip RAM 中對應的第 x 個資料，而是將之儲存到緩衝暫存器中的第 x 個位置，等計算完一整列之後再將一整列存回 on-chip RAM1 及 on-chip RAM2 中，而列方向運算則依此類推，每一列和每一行都做完運算之後，最後存在兩個 on-chip RAM

中的就是函數$\tau_{u,v}$。

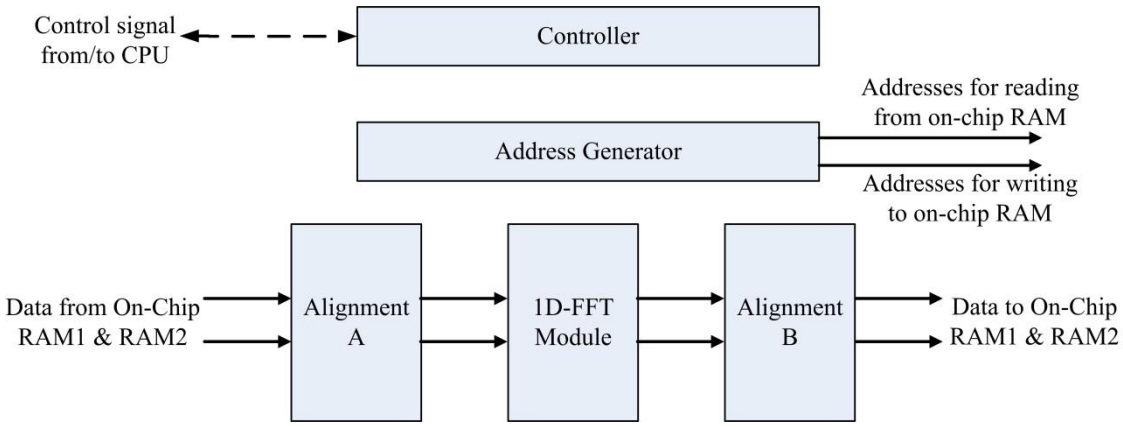

圖 3.20 快速傅立葉轉換單元硬體電路架構圖

圖 3.20 為快速傅立葉轉換單元的硬體架構圖，由控制器、位址產生器、一維的快速傅立葉模組(1D-FFT module)和兩個格式校準器(Alignment A, Alignment B)所組成。由前敘述之離散傅立葉轉換電路可知，計算一維的離散傅立葉需要花費大量時間，且需要額外產生暫存器來儲存未算完的中間值，因此本論文中除了實作離散傅立葉轉換電路，也實作了快速傅立葉轉換電路，而兩者之間比較則會在第四章再詳細說明。

本論文所使用的一維快速傅立葉模組是使用，Altera 公司所提供的FFT MegaCore function[]。此模組為單一複數資料輸入及單一複數資料輸出的形式，資料轉換長度為N，並且選用資料流(streaming)的模式，此種模式可以連續的輸入N筆資料到一維快速傅立葉模組，並且待轉換完成後也能連續的將結果連續的輸出。資料流模式的傅立葉模組輸入所使用的是定點數(Fixed-point)格式，輸出則是採用分段浮點數(Block Floating Point)格式，為了配合菲涅耳轉換前單元以及菲涅耳轉換後單元所使用的IEEE 754浮點數格式，因此在一維快速傅立葉模組前後都

各加上一塊格式校準器做為輔助。

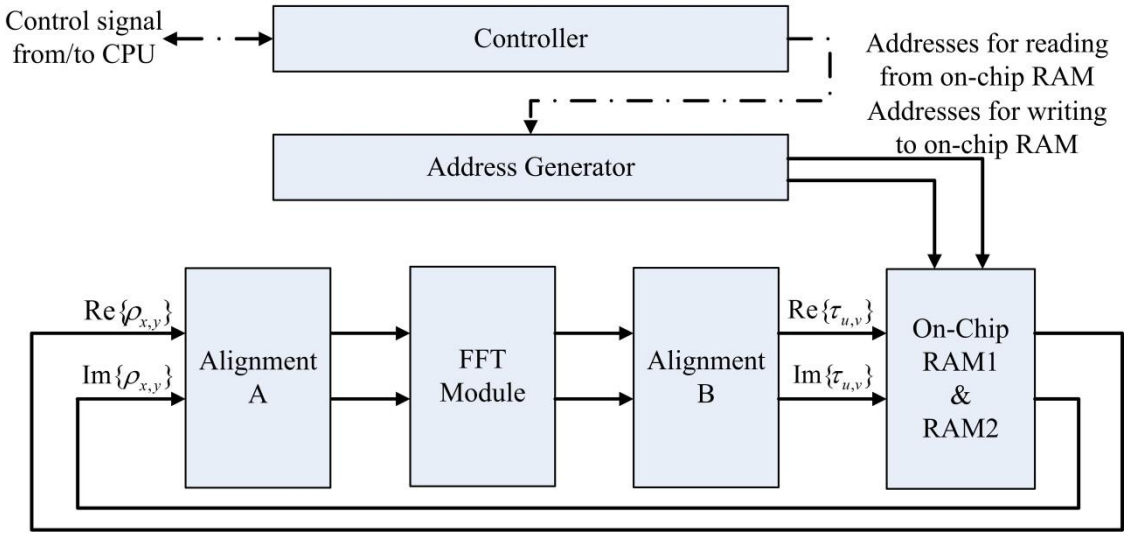

圖 3.21 快速傅立葉轉換單元資料流動時序圖

快速傅立葉轉換單元資料流動情況如圖 3.21，菲涅耳轉換前單元計算結果$\rho_{x,y}$為 IEEE 754 單精度浮點數格式，輸入格式校準器 A，將單精度浮點數格式轉換成定點數格式，再將轉換好的值送到快速傅立葉轉換單元中進行運算，快速傅立葉單元的輸出為分段浮點數格式，為格式校準器 B 的來源值，格式校準器 B 將分段浮點數格式轉為單精度浮點數格式，才將轉換好的值存回 on-chip RAM 之中。使用快速傅立葉轉換單元，其計算方法與離散傅立葉轉換單元步驟相同，首先進行列方向運算，每次取一行依順序輸入快速傅立葉轉換模組中，並將計算結果存回 on-chip RAM 中對應的記憶體位址，當列方向都計算完成後再進行行方向計算，直到所有行都計算完成，可得傅立葉轉換單元結果$\tau_{u,v}$。

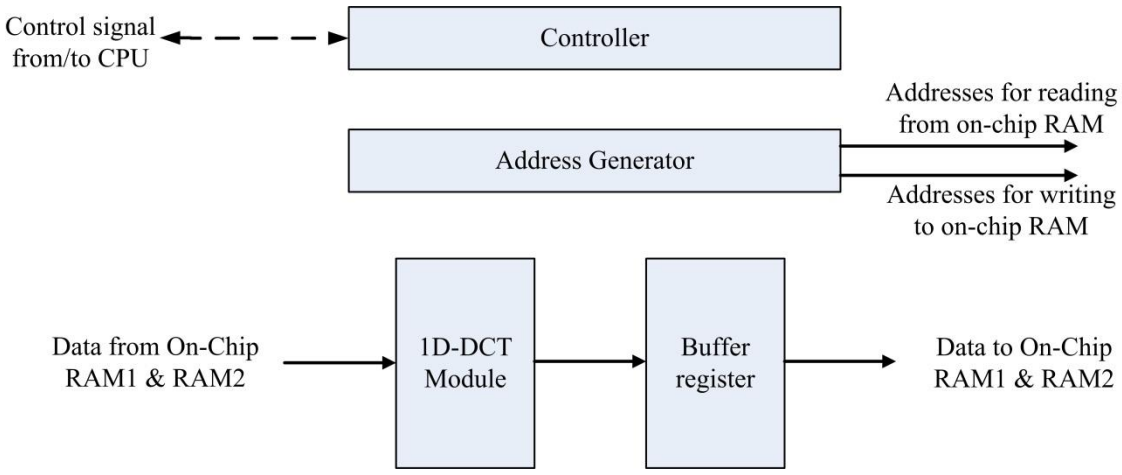

圖 3.22 離散餘弦轉換單元硬體電路架構圖

圖 3.22 為離散餘弦轉換單元的硬體架構圖，目的在於以硬體實現相位展開法則電路中的步驟二。其中包含控制器、位址產生器、一維的離散餘弦轉換模組(Cosine transform unit)和緩衝暫存器(Buffer register)。

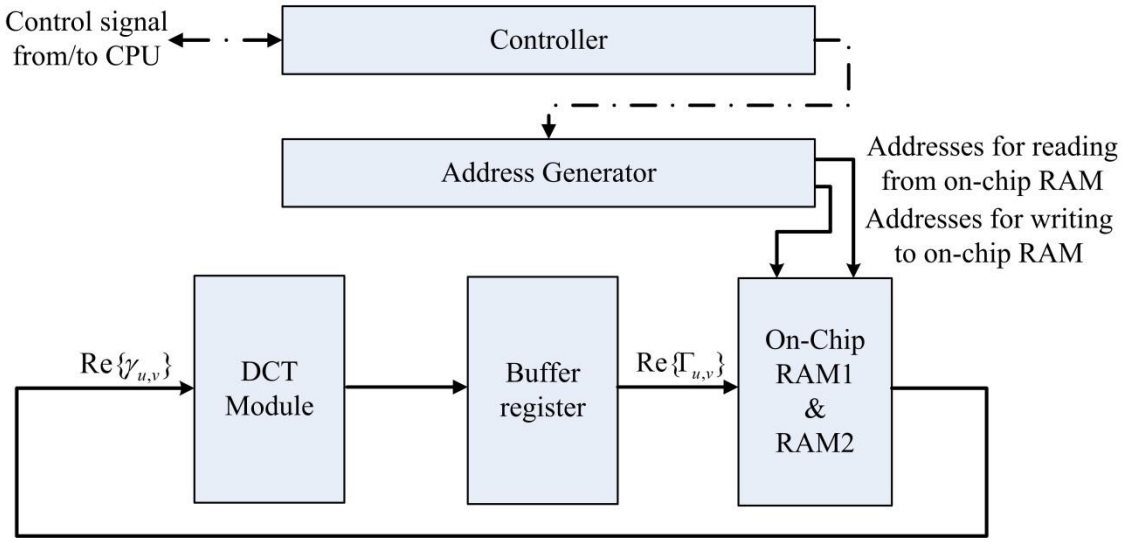

圖 3.23 離散餘弦轉換單元資料流動時序圖

如圖 3.23，離散餘弦轉換單元執行方法類似於離散傅立葉單元，利用一維離散餘弦轉換模組對函數$\gamma_{u,v}$中的每一列進行列運算，算完之後先將值儲存在緩衝暫存器中，直到一整列算完，再將緩衝暫存器中的值存回 on-chip RAM，取代原

本的一列γ_u，當所列方向的列運算執行完成之後，再對每一行執行行運算，直到運算結束，儲存在 on-chip RAM1 及 on-chip RAM2 之中的就是函數$\gamma_{u,v}$經過二維餘弦轉換運算之後的值$\Gamma_{u,v}$。

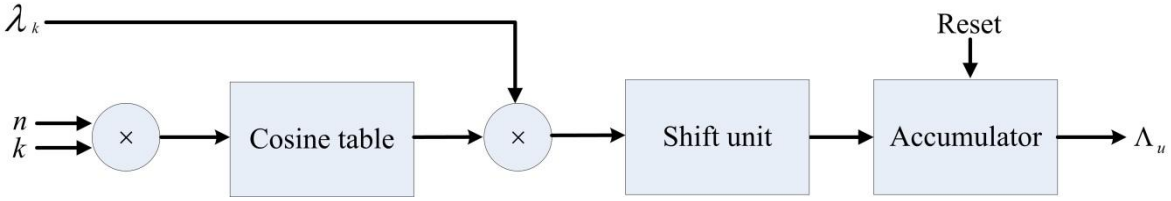

圖 3.24 一維的離散餘弦轉換模組內部硬體電路架構圖

一維離散餘弦轉換模組的電路設計如圖 3.24，包含兩個乘法器、一個餘弦值表、一個位移單元(Shift unit)以及一個累加器。其中餘弦值表以及累加器都與一維離散傅立葉轉換模組一樣。

根據公式(2.21)，先將輸入的 n 與 k 相乘，若$xk > N$，則對xk取 N 的餘數進行查表，查出的餘弦值再與相位差λ_k相乘，公式(2.21)有部分值必須在累加後乘以 2，在此設計中則根據分配律將計算的值先輸入位移單元中乘以 2，再進行累加，以二進位的角度來看，乘以 2 即將所有位元左位移一個位元，因此不需要真的產生一個乘法器元件，使用位移方法是比較快速也比較節省資源的作法。

3.5 相位展開法之轉換前單元(Pre-transform Unit)及轉換後單元(Post-transform Unit)

本相位開法則之轉換前單元及轉換後單元分別以硬體電路實現相位展開法則演算法的步驟一及步驟三。

相位展開法則轉換前單元的硬體架構如圖3.25所示，包含了位置產生器、控制器、暫存器、加法器、相位壓縮模組(phase wrapping module)以及多工器。其中位置產生器是用來自動產生記憶體位置來讀取來源資料以及將運算結果存回on-chip RAM 中相對應的位置。

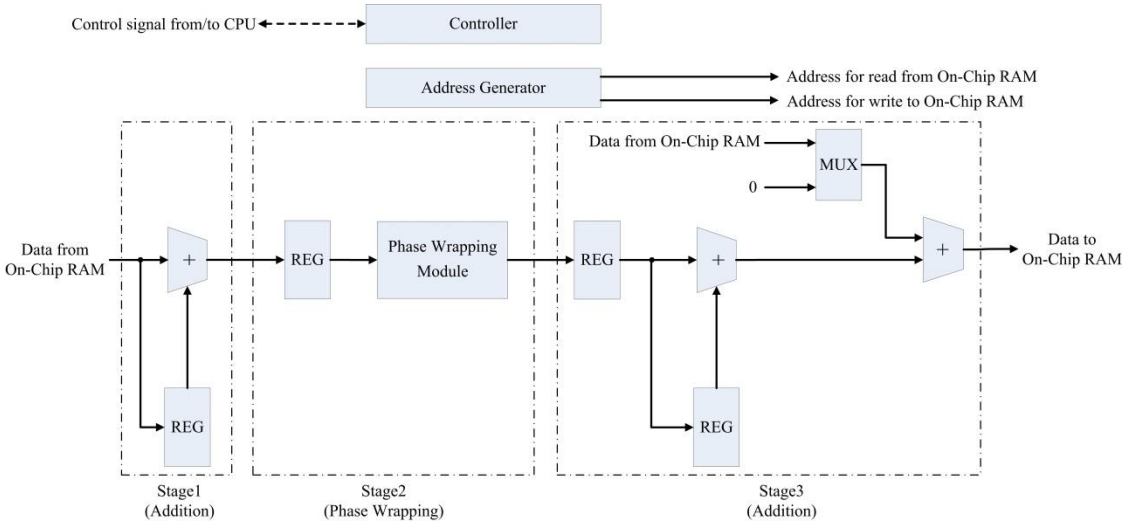

圖3.25 相位展開法轉換前單元硬體電路架構圖

由圖3.25可以得知轉換前單元是一個3級的管線化架構，根據公式(2.18)，求出$\gamma_{u,v}$之前必須先求出壓縮相位差$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$，相位差則是根據公式(2.16)所求出，並且其值必須限制在範圍$(-\pi,\pi]$，在相位展開法則轉換前單元的第一級，使用加

法器計算出$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$；而第二級則是利用相位壓縮模組將第一級計算出的$\Delta_{u,v}$值調整至範圍$(-\pi,\pi]$；最後在第三級時，也透過加法器將調整過後的$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$相加，得到最終相位差$\gamma_{u,v}$。

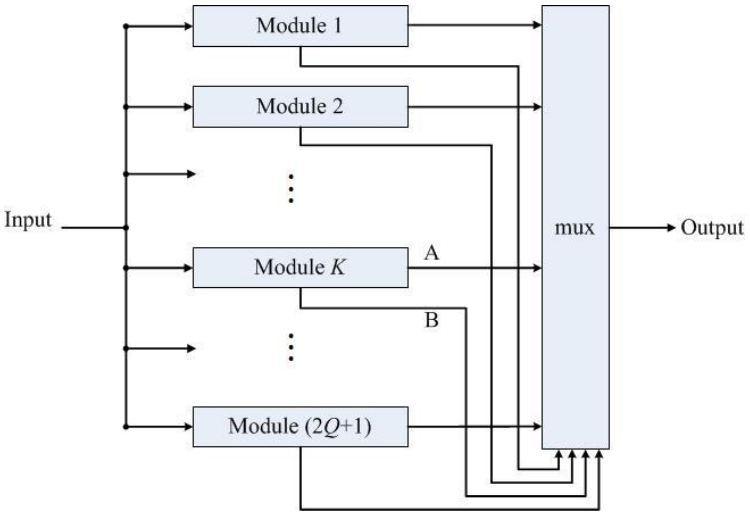

圖3.26　相位壓縮模組內部硬體電路架構

相位壓縮模組其內部架構如圖3.26，其中包含2Q+1個模組以及一個多工器。當電路第一級所計算出的$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$進入相位壓縮模組之後，會以廣播(broadcast)的方式發送給所有的模組並且做同步的運算，對於任一個模組k，k = 1,..,2Q+1，其各自表示介於$(-(2\mathrm{Q}+1)\pi+2(k-1),-(2Q+1)\pi+2k\pi]$的特定數字範圍；而每個模組內部架構如圖3.27，皆由兩個比較器、一個加法器和一個AND邏輯閘所組成。模組中comparator1會比較輸入資料$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$的數值是否小於等於$(k+2)\pi$，comparator2則是比較$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$的數值是否大於$k\pi$，藉此判斷出所輸入的數值是否位於該模組所代表的邊界數值範圍之內；若輸入的數值是介於該模組所代表的範圍之內，則兩個比較器的輸出值經過AND邏輯閘執行運算之後，所輸出的B值會為1，反之則為0。比較的同時間，模組中的加法器也會將所輸入的$\Delta_{u,v}^{x}$或$\Delta_{u,v}^{y}$加

上$-k\pi-\pi$，輸出的A值即為相加後的結果。每個模組運算後所產生的A值及B值會先輸出到多工器中，而B值會作為多工器的選擇訊號，由於每個模組所代表的數值範圍皆不相同，因此只有一個模組輸出的B數值會為1，多工器就會選擇將該模組所計算出的A值當作最終的輸出結果，得到的結果即為數值被調整在$(-\pi, \pi]$間的$\Delta_{u,v}^{x}$和$\Delta_{u,v}^{y}$。

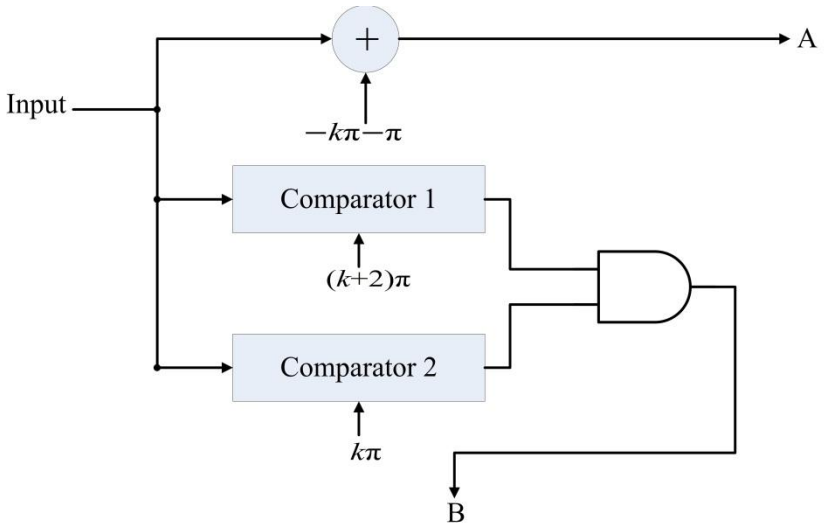

圖3.27 相位壓縮模組中模組硬體電路架構圖

在轉換前單元執行運算之前，我們會先將轉換前單元運算所需的來源資料放置在on-chip RAM1的上半部中，並且將轉換前單元計算所得到的結果分成兩部分儲存在on-chip RAM1上半部以及下半部之中。

根據公式(2.16)和公式(2.18)，$\gamma_{u,v}$其實是由$\psi_{u,v}$所計算出來，並非直接由$\zeta_{u,v}$計算而得。若要計算出$\gamma_{0,v}$、$\gamma_{N,v}$、$\gamma_{u,0}$和$\gamma_{u,N}$，則需要由鏡面反射運算得到的$\psi_{-1,v}$、$\psi_{N+1,v}$、$\psi_{u,-1}$和$\psi_{u,N+1}$這些資料，$0 \le u \le N$和$0 \le v \le N$，根據公式(2.15)可知，$\psi_{-1,v} = \zeta_{1,v}$、$\psi_{N+1,v} = \zeta_{N-1,v}$、$\psi_{u,-1} = \zeta_{u,1}$和$\psi_{u,N+1} = \zeta_{u,N-1}$，然而在實際的電路中，不需要真的在轉換前單元中特別設計一個執行鏡面反射運算的元件，只需要

利用轉換前單元中的位置產生器就可以達到同樣的目的，也就是說當電路中需要輸入$\psi_{-1,v}$、$\psi_{N+1,v}$、$\psi_{u,-1}$和$\psi_{u,N+1}$這些資料時，只需由位置產生器產生對應的記憶體位置從on-chip RAM1中讀取出$\zeta_{1,v}$、$\zeta_{N-1,v}$、$\zeta_{u,1}$和$\zeta_{u,N-1}$即可得到與執行鏡面反射運算相同的結果。

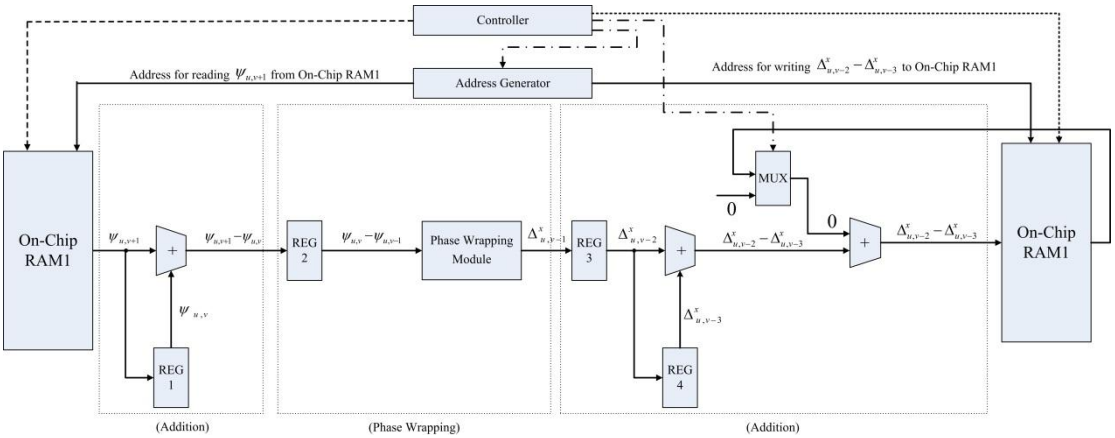

圖3.28 相位展開法則轉換前單元資料流動時序圖(1)

圖3.28為資料流動的情況。在管線內的第一級中，從on-chip RAM1上半部中讀取$\psi_{u,v+1}$進入轉換前單元，此數值會分別送往REG1儲存以及送往加法器來與原本儲存在REG1中的$\psi_{u,v}$相減，而相減得到的值會儲存於REG2中。同時在管線內的第二級會將原本儲存在REG2 的前一筆運算結果當$\psi_{u,v}-\psi_{u,v-1}$送至相位壓縮模組計算得到數值範圍介於$(-\pi,\pi]$之間的$\Delta^x_{u,v-1}$，並將結果存入REG3中。而管線內的第三級中，則會將原本儲存於REG3中的$\Delta^x_{u,v-2}$分別送往REG4中儲存以及送往加法器來與原本儲存在REG4中的$\Delta^x_{u,v-3}$相減，計算完會把此數值傳遞給on-chip RAM前的加法器，控制器會設定多工器的選擇訊號，在轉換前單元的步驟一中，控制器會讓多工器選擇數值0作為輸出，因此on-chip RAM1前的加法器會將$\Delta^x_{u,v-2}-\Delta^x_{u,v-3}$和0做相加之後，傳遞到on-chip RAM1的下半部中儲存。

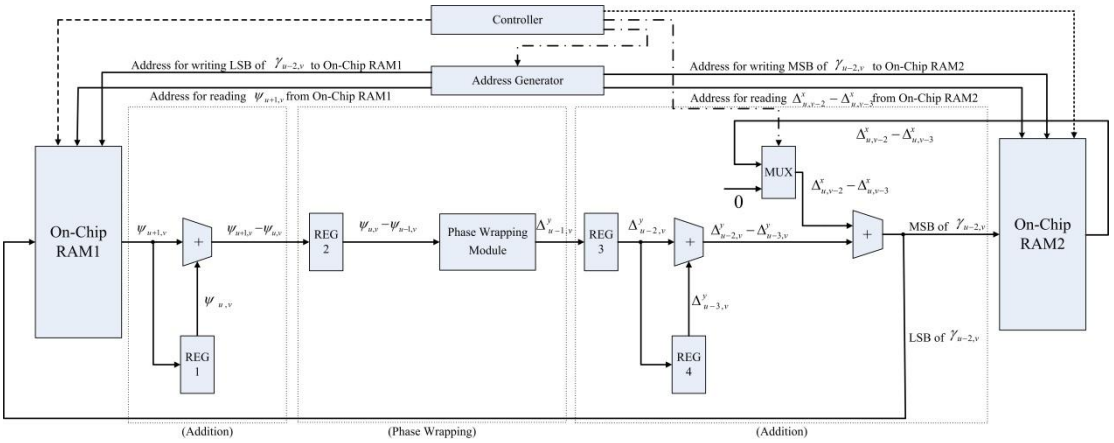

圖3.29 相位展開法則轉換前單元資料流動時序圖(2)

當所有x方向的相位差都處理完之後，電路會再繼續計算y方向的相位差，如圖3.29，其步驟都相同，除了最後控制器會讓多工器的訊號選擇從on-chip RAM1下半部讀取出的$\Delta^x_{u,v-2}-\Delta^x_{u,v-3}$作為加法器的其中一個輸入來源，並由加法器將$\Delta^x_{u,v-2}-\Delta^x_{u,v-3}$和$\Delta^y_{u,v-2}-\Delta^y_{u,v-3}$相加求出$\gamma_{u-2,v}$，將之存回on-chip RAM1之中。

圖3.30為轉換後單元的硬體架構，包含了控制器、位置產生器、兩個餘弦函數計算模組、加法器和除法器。其中餘弦函數計算模組是由位置轉索引轉換器(address to index converter)和Lookup Table 所組成。

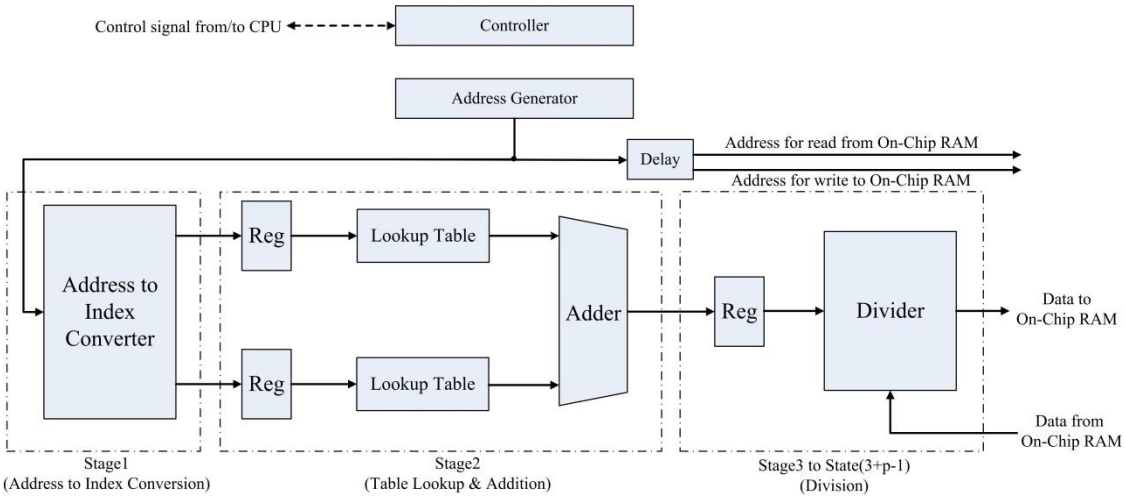

圖3.30 相位展開法轉換後單元硬體電路架構圖

此單元的餘弦函數計算模組設計方法類似菲涅耳轉換前單元及轉換後單元的餘函數值表，其中共有N個entry，利用位址產生器產生索引m及n，除了將之送到記憶體讀取其對應的資料外，也當成索引值輸入餘弦函數模組中進行查表。

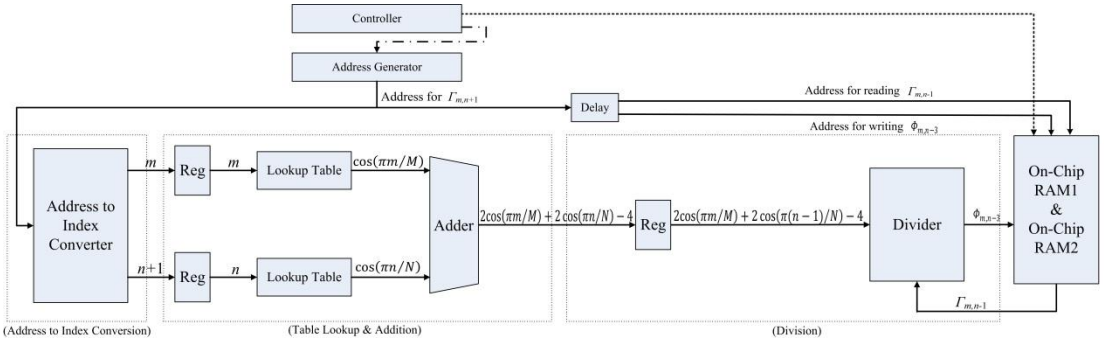

圖3.31 相位展開法則轉換後單元資料流動時序圖

圖3.31為轉換後單元資料流動狀況，首先根據位址產生器所產生的索引值，從餘弦函數計算單元中取得對應的值，接著利用加法器將兩輸出值相加，計算出值$2\cos\left(\frac{\pi m}{N}\right)+2\cos\left(\frac{\pi n}{N}\right)-4$，此數值即公式(2.20)中的分母部分。轉換後單元中的除法器是採用Altera公司所提供的Altera Floating Point Megafunction[]，此除法器具備p個管線化層級，能使得運算的整體效能提升。而除法器的目的是將由on-chip RAM讀出的$\Gamma_{m,n}$與加法器計算的結果相除，計算後的結果即為$\Phi_{m,n}$，最後將之存回on-chip RAM之中。

值得強調的是，轉換後單元一開始所輸入的運算資料並不是由on-chip RAM中讀取出快速傅立葉轉換單元的計算結果$\Gamma_{m,n}$，而是對應於on-chip RAM中$\Gamma_{m,n}$的記憶體位置m及n。我們將此記憶體位置分別送至餘弦函數計算模組，以及一個延遲(delay)裝置。此延遲裝置會在加法器計算完$2\cos\left(\frac{\pi m}{N}\right)+2\cos\left(\frac{\pi n}{N}\right)-4$之後，才將記憶體位置送入on-chip RAM中讀取$\Gamma_{m,n}$，並送至除法器與加法器計算結果。藉

由此設計，一個記憶體位置卻可以獲得多種資料數值，這種single-address-multiple-data的方式使我們可以減少對記憶體的存取次數，可以有效的提升轉換後單元的運算速度。

3.6 軟硬體共同設計

本論文提出的菲涅耳轉換及相位展開法則硬體電路架構在整個SoPC系統架構中會被當成一個客製化邏輯電路（custom user logic）。除了客製化的硬體電路之外，還包含了NIOS II CPU等元件，如圖3.32所示。

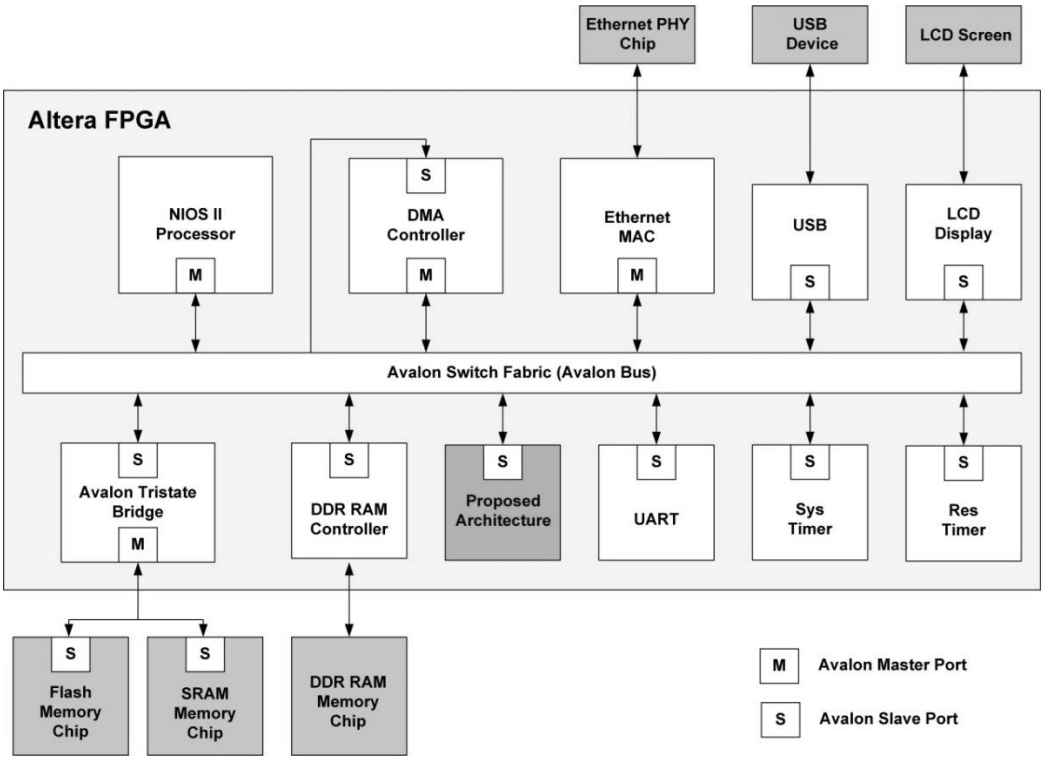

圖3.32 SoPC 系統電路架構圖

NIOS II CPU[]的功用在於控制SoPC系統架構中資料的流動，由於本論文所提出的硬體架構是由兩個on-chip RAM作為菲涅耳轉換電路及相位展開法則電路儲存運算結果的目的地，因此為了確保兩個on-chip RAM能夠儲存到正確的數值，CPU必須負責設定狀態暫存器的數值，讓兩個on-chip RAM前的多工器根據狀態暫存器內的數值選擇正確的輸入來源並將其計算結果存入on-chip RAM之中。

圖3.33顯示NIOS II CPU執行的流程。由此流程圖可以發現，NIOS II CPU除了負責設定狀態暫存器內的數值之外，只需要負責傳送控制訊號來開啟各個單元的運作，各單元在接收到由NIOS II CPU傳來的開啟訊號後，就會自動的開始執行運算，並將運算後的結果存回on-chip RAM當中，也因為NIOS II CPU只需要處理如此簡單的軟體指令，使得本硬體架構在執行菲涅耳轉換運算及相位展開法則運算的效能有很大的提升。

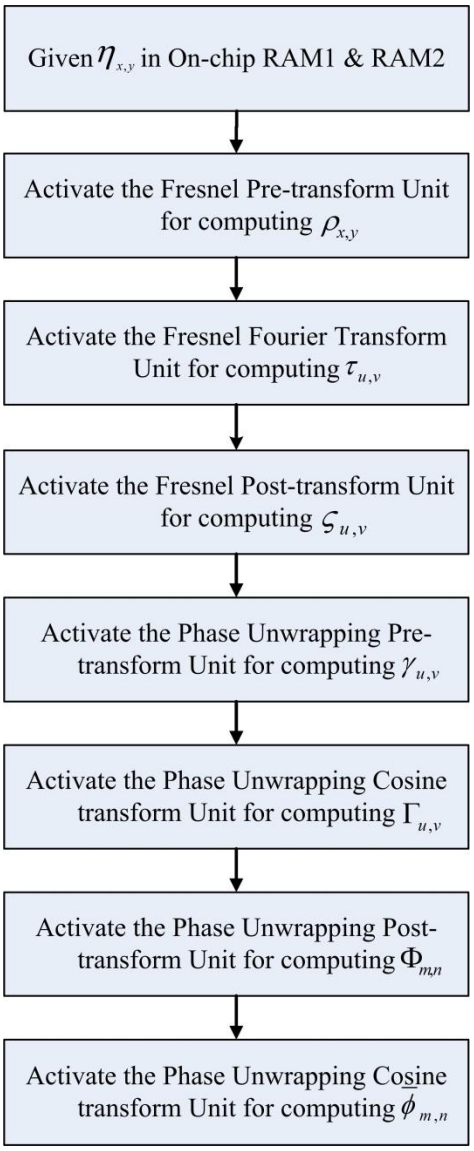

圖3.33 NIOS II CPU執行之軟體流程圖

第四章 實驗數據與效能比較

本章節將呈現本論文所提出的菲涅耳轉換及相位展開法則硬體架構之正確性、實際效能測量與比較，以及實驗環境的介紹。

4.1 開發平台與實驗環境介紹

本論文提出的硬體架構是以Altera 的Stratix III EP3SL150F1152C2N FPGA開發板為主要的實現平台，如圖4.1 所示。表格4.1 是Stratix III EP3SL150F1152C2N FPGA開發板的詳細規格資訊。而選擇以FPGA 開發板來實現與驗證硬體電路，是因為可程式化系統晶片(System on a Programmable Chip, SoPC)可以快速的將硬體架構設計實現與驗證，具備可重複修改與快速上市等優點，所以FPGA 非常適合實作本論文所提出的硬體架構。

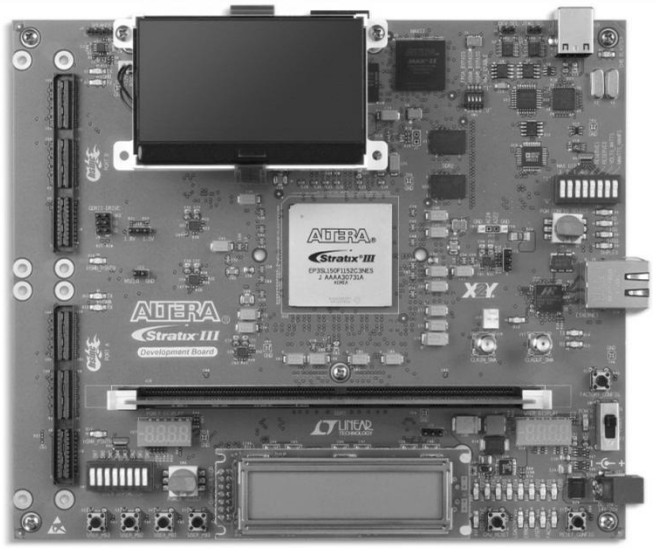

圖4.1 Altera Stratix III EP3SL150F1152C2N 開發板外觀

表4.1 Altera Stratix III EP3SL150F1152C2N 開發板規格表

Feature	Stratix III
Device	EP3SL150F1152C2N
Adaptive Logic Modules (ALMs)	56800
Combinational ALUTs	113600
Memory ALUTs	56800
Equivalent Logic Elements (LEs)	142000
Dedicated Logic Registers	27213
M9K Memory Blocks	355
M144K Memory Blocks	16
Total block memory bits	5,630,976
DSP block 18-bit elements	384
Total PLLs	8
Total DLLs	4
Total Pins	744

本論文使用Altera Quartus II 8.0作為撰寫Verilog硬體描述語言的平台；並且透過Quartus II提供的語法檢查、時序分析、邏輯元件的配置、產生規劃檔案、電路合成以及繞線佈局等強大的功能，可以快速建立系統架構，並藉由模擬訊號波型圖驗證其正確性。待系統架構設計完成後，就可以將系統架構掛載在SoPC system上，並且將掛載完畢的SoPC system燒錄於FPGA開發板後，就可以進行實質的訊號輸入/輸出驗證系統架構的效能與正確性。而SoPC system內許多元件包含CPU、

客製化電路(菲涅耳轉換及相位展開電路)、記憶體等等組成一個系統則是透過Altera SoPC Builder這套介面配置而成。Altera Quartus II 8.0所使用的PC規格為Intel® Core™2 i7 CPU 920 @ 2.67GHz、8G DDRIII記憶體。

而Altera公司也提供了採用Eclipse為基礎的NIOS II IDE軟體供使用者使用，，所有在系統中的軟體開發都可以在NIOS II IDE下完成。由於NIOS II系統提供設計人員在Altera FPGA中開發完整的Nios II嵌入式系統所需的軟體工具、應用程式、函式庫及驅動程式，方便使用者利用此IDE 軟體來與FPGA開發板溝通互動，加速SoPC 系統的開發。

本論文除了使用Verilog 硬體描述語言來實作所提出之硬體電路架構，也會使用MATLAB R2008a為平台，以軟體實現相同的演算法則來和硬體電路架構的執行結果互相比較。

※ 硬體實現環境

Device：Altera Stratix III EP3SL150F1152C2N

CPU：NIOS II 50MHz

Memory：64-MB DDR2 SDRAM

※ 軟體實現環境

CPU：IntelR Core.2 i7 CPU 920 @ 2.67GHz

Memory：DDRIII 8.0 G

4.2 實驗數據呈現與討論

本節將探討菲涅耳轉換硬體電路及相位展開法則硬體電路架構在輸入影像為64 × 64時，實際在FPGA開發板上測量到的效能以及資源消耗的情況。

在實際的SoPC系統上，硬體資源主要分成三大部分，分別為ALMs、block memory bits和DSP blocks，其中ALMs被用於實現暫存器以及運算單元；block memory bits主要用於實現記憶體元件；而DSP blocks則是被用來實現數值運算中像是除法器以及乘法器等元件。

表4.2為菲涅耳轉換電路個單元所消耗的ALMs資源情形，表4.3為相位展開法則電路中各個單元及On-chip RAM所消耗ALMs資源情況，會使用到ALMs資源的主要為浮點數加減法器、浮點數乘法、快速傅立葉模組以及暫存器，另外如菲涅耳轉換前單元中所查的指數表、菲涅耳離散傅立葉轉換單元中所查的正弦函數值以及餘弦函數值表、菲涅耳快速傅立葉轉換單元中的格式校準器等，也都會消耗資源。可以發現在菲涅耳轉換電路中資源消耗普遍較高，因為內部多使用查表方法，以及使用複數乘法運算，若使用複數乘法器，一次就必須使用用到四個浮點數乘法器，而相位展開電路中，餘弦轉換單元需要額外暫存器來記錄算到一半的值，因此消耗資源也相對偏高；另外，在傅立葉轉換單元中，若使用快速傅立葉模組，其模組也會消耗大量ALMs。在相位展開法則電路中ALMs則明顯消耗較低，主要都是暫存器所消耗的資源，其中只有相位展開轉換後單元內部有使用一個浮

點數除法器以及用來查出餘弦函數值的lookup table，因此資源消耗提高了一些。

表4.2 菲涅耳轉換電路中各單元ALMs資源消耗表

	Fresnel Pre-transform Unit	Fresnel Fourier-transform Unit		Fresnel Post-transform Unit
		DFT	FFT	
ALMs消耗	3409	4383	7346	7725

表4.3 相位展開法則電路及On-chip RAM中各單元ALMs資源消耗表

	Phase unwrapping Pre-transform unit	Phase unwrapping Cosine-transform unit	Phase unwrapping Post-transform unit	On-chip RAM
ALMs消耗	303	2192	664	95

表4.4和表4.5分別是菲涅耳轉換電路中各單元中block memory bits資源消耗，以及相位展開法則電路中各單元及On-chip RAM中block memory bits資源消耗表，block memory bits主位用來組成記憶體單元，因此可看出主要資源消耗單元為On-chip RAM和快速傅立葉模組，其餘部分則是被浮點數除法器所消耗；本架構中的on-chip RAM單元是由on-chip RAM1以及on-chip RAM2兩個RAM 2-port module所組成，而這兩個RAM module中每筆資料儲存的大小皆為32 bits，所以on-chip RAM 所消耗的block memory bits 為$(N+1)\times(N+1)\times 32$ bits$\times 2$，其中$N+1$為影像之長寬，因此block memory bits的資源消耗量會隨著影像大小的增加。而快速傅立葉單元也會因為處理長度增加而消耗資源量增加。

表4.4 菲涅耳轉換電路中各單元block memory bits資源消耗表

	Fresnel Pre-transform Unit	Fresnel Fourier-transform Unit		Fresnel Post-transform Unit
		DFT	FFT	
Block memory bits消耗	0	0	12288	9216

表4.5 相位展開法則電路及On-chip RAM中各單元block memory bits資源消耗表

	Phase unwrapping Pre-transform unit	Phase unwrapping Cosine-transform unit	Phase unwrapping Post-transform unit	On-chip RAM
Block memory bits消耗	0	0	4608	270400

表4.6和表4.7為菲涅耳轉換電路中各單元及On-chip RAM中DSP blocks資源消耗表，以及相位展開法則電路中各單元DSP blocks資源消耗表，主要會消耗DSP block的是浮點數乘法器、除法器及快速傅立葉模組，又以除法器及快速傅立葉模組消耗量較大，菲涅耳轉換電路中各單元都有使用到浮點數乘法器，而菲涅耳轉換後單元中有還有使用除法器，因此消耗量較高，而相位展開法則電路中，只有離散餘弦轉換單元用到一個乘法器，轉換後單元則有一個除法器，消耗量較低。

表4.6 菲涅耳轉換電路中各單元DSP blocks資源消耗表

	Fresnel Pre-transform Unit	Fresnel Fourier-transform Unit		Fresnel Post-transform Unit
		DFT	FFT	
DSP blocks 消耗	32	17	24	88

表4.7 相位展開法則電路及On-chip RAM中各單元DSP blocks資源消耗表

	Phase unwrapping Pre-transform unit	Phase unwrapping Cosine-transform unit	Phase unwrapping Post-transform unit	On-chip RAM
DSP blocks 消耗	0	5	16	0

表4.8為菲涅耳轉換電路及相位展開法則電路的總資源消耗，以及兩者整合後的資源消耗，其中菲涅耳轉換電路及相位展開法則電路資源不計on-chip RAM部分，因其由兩者所共用。若單看菲涅耳轉換電路(使用DFT Unit)或相位展開法則電路的話，都不太會消耗到block memory bits，主要在消耗ALMs及DSP資源；而on-chip RAM則相反，主要消耗block memory bits；但若是使用FFT Unit的菲涅耳電路，可發現三大類資源量都上升，因為快速傅立葉轉換模組本身就會消耗掉大量資源，尤其是ALMs以及block memory bits消耗量幾乎是使用離散傅立葉模組時的兩倍，且消耗量會隨轉換資料長度增長而增加，當處理較大型圖片時，on-chip RAM和快速傅立葉模組所消耗的block memory bits都會大幅增加，記憶體資源可能成為資源的瓶頸，因此需要審慎評估。在整合電路中，所計算的資源消耗則包含菲涅耳轉換電路、相位展開法則電路以及on-chip RAM三大部分，可發現，若在菲涅耳轉換電路中使用離散傅立葉轉換單元，其資源消耗量十分的低，對於擴大圖片尺寸有更多可用資源。

表4.8 部分電路與整合後電路總資源消耗表

	ALMs	Block memory bits	DSP blocks
菲涅耳轉換電路(使用DFT Unit)	17278(15%)	9216(<1%)	137(36%)
菲涅耳轉換電路(使用FFT Unit)	22605(20%)	21504(<1%)	144(38%)
相位展開法則電路	2574(2%)	0(0%)	21(5%)
On-chip RAM	95(<1%)	270400(5%)	0(0%)
菲涅耳轉換電路(使用DFT Unit)及相位展開法則整合電路	20946(18%)	280639(5%)	158(41%)
菲涅耳轉換電路(使用FFT Unit)及相位展開法則整合電路	25580(23%)	310073(6%)	165(43%)

將本系統架構掛載於SoPC時，NIOS software CPU也會消耗一些硬體資源，掛載完畢的整體系統架構資源消耗如表4.8所示。其中菲涅耳轉換電路及相位展開法則電路資源也不計on-chip RAM部分，而整合電路的數據則包含了兩大塊電路以及on-chip RAM總消耗。

表4.9 部分電路與整合後電路於SOPC系統中總資源消耗表

	ALMs	Block memory bits	DSP blocks
菲涅耳轉換電路(使用DFT Unit)	24343(21%)	779264(14%)	141(37%)
菲涅耳轉換電路(使用FFT Unit)	29469(26%)	1061952(19%)	148(39%)
相位展開法則電路	9199(8%)	746496(13%)	25(7%)
菲涅耳轉換電路(使用DFT Unit)及相位展開法則電路整合	34542(30%)	1527684(27%)	166(43%)
菲涅耳轉換電路(使用FFT Unit)及相位展開法則電路整合	39748(35%)	1874529(33%)	173(45%)

表4.10為菲涅耳轉換電路執行時間與軟體matlab執行時間做比較，是根據電路設計方法而寫出的軟體版本。觀察表可以發現Fresnel離散傅立葉轉換單元所花費時間非常長，且佔所有執行時間約99%，以matlab模擬離散傅立葉轉換單元電路所執行動作，其量測出的時間約為電路執行時間的兩倍；若以快速傅立葉轉換單元來執行計算動作，不管是電路或是以MATLAB模擬，其時間都會大幅縮短，快速傅立葉電路執行速度比MATLAB模擬快約8倍；若比較快速傅立葉以及離散傅立葉兩者執行速度，以MATLAB模擬互相比較，快速傅立葉電路比離散傅立葉電路快約97倍，而以電路互相比較，快速傅立葉電路則比離散傅立葉電路快323

倍之多。

表4.10 菲涅耳轉換電路與Matlab軟體模擬各階段影像執行時間比較

	Fresnel Pre-transform Unit	Fresnel DFT Unit	Fresnel FFT Unit	Fresnel Post-transform Unit
Circuit execution time(ms)	0.22902	76.54192	0.23692	0.19434
MATLAB execution time(ms)	0.98958	181.97647	1.88327	4.61207

表4.11為相位展開電路執行時間與軟體執行時間做比較。可以發現也是在Cosine transform unit所花費時間最長,若以matlab模擬DCT Unit電路所執行動作，其量測出的時間約快電路執行時間的2.5倍，但其若以快速傅立葉轉換(FFT)以及反快傅立葉(IFFT)來取代元DCT Unit所執行動作，時間則可縮短為原本軟體執行時間的十分之一倍。

表4.11 相位展開法則電路與Matlab軟體模擬各階段影像執行時間比較

	Phase unwrapping Pre-transform Unit	Phase unwrapping Cosine-transform Unit	Phase unwrapping Post-transform Unit	Phase unwrapping Inverse Unit
Circuit execution time(ms)	0.23188	79.29193	0.00792	79.29193
MATLAB execution time(ms)	5.53199	30.65038	0.19710	30.49417
MATLAB execution time(ms)	5.53199	3.87229 (FFT)	0.19710	4.83989 (IFFT)

最後，表4.12為菲涅耳轉換電路及相位展開電路個別執行時間，及其整合之後電路運算所花費總時間。由表可知，雖然兩塊電路中轉換前單元以及轉換後單元執行時間都比軟體來的快上許多，但是這兩大塊電路皆以傅立葉轉換執行時間最長，尤其可以看出使用DFT以及使用FFT運算其時間差相距非常大。

本論文實做了DFT以及FFT電路，進行資源消耗及運算時間分析之後，DFT擁有低資源消耗的優點，但是執行速度太慢，時間複雜度呈現$O(N^2)$成長，FFT則是消耗大量資源，但其執行速度很快，時間複雜度為$O(N\log N)$，利用FPGA來運算又更能凸顯其運算速度上的優勢。

表4.12 部分電路與、整合後電路與Matlab軟體模擬執行總時間比較

	Circuit execution time(ms)	MATLAB execution time(ms)
Fresnel use DFT	76.96528	187.57812
Fresnel use FFT	0.66068	7.48492
Phase unwrapping	158.82366	66.87364
Fresnel (DFT)及 Phase unwrapping 整合電路	235.78894	254.45176
Fresnel(FFT)及 Phase unwrapping 整合電路	159.48434	74.35856

為了驗證本論文所提出的菲涅耳轉換及相位展開法則硬體電路之正確性，本論文由國立台灣師範大學光電科技研究所鄭超仁教授的研究室提供全像片平面的複數振幅資訊，做為測試影像。圖4.2為利用matlab軟體模擬所獲得的相角值，其值被壓縮在$(-\pi,\pi]$，圖4.3則為經由菲涅耳轉換電路運算後所獲得的相角值。

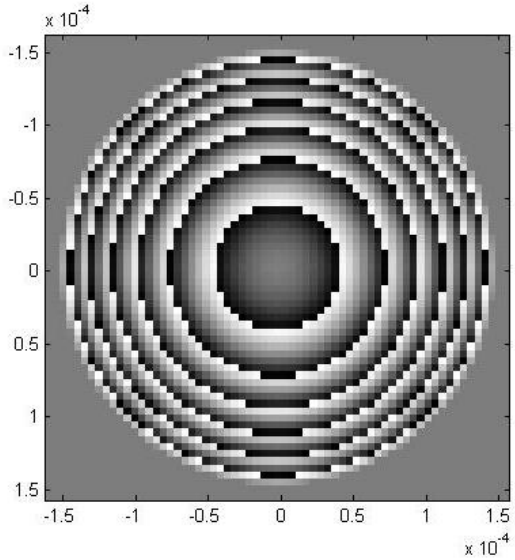

圖4.2 由MATLAB模擬獲得之相角值

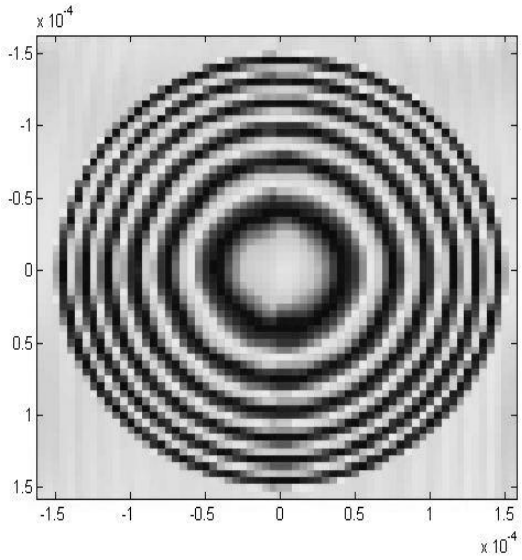

圖4.3 由硬體電路運算獲得之相角值

圖4.4為使用matlab模擬，執行相位展開後所得到的連續相位圖，而圖4.5則是執行相位展開法則電路之後所獲得的連續相位圖。

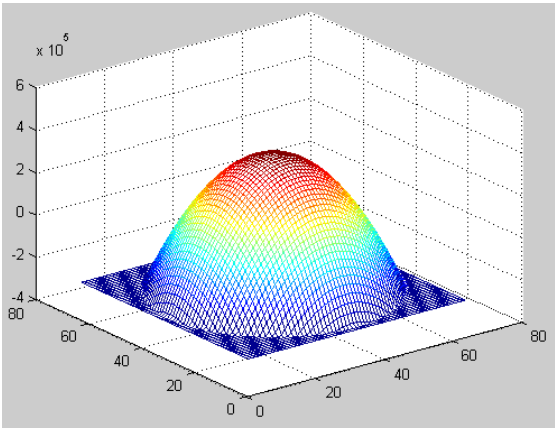

圖4.4 由MATLAB模擬獲得之連續相角還原圖

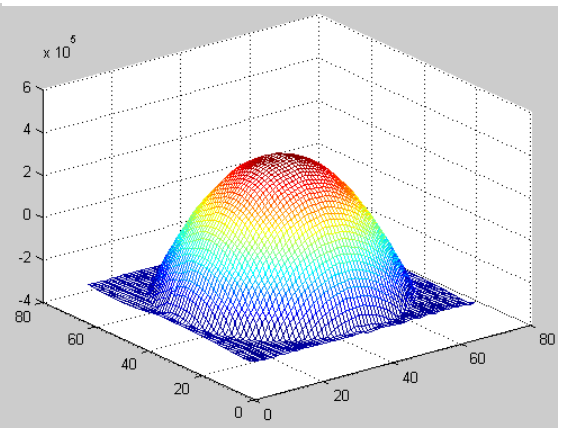

圖4.5 由硬體電路運算所得到之連續相位角還原圖

第五章 結論

本論文所實現的是可適用在嵌入式 DHM 系統上執全像圖重建之菲涅耳轉換及相位展開法則運算的硬體電路架構。在本論文所提出的硬體架構中，每個運算單元皆利用管線化硬體架構的方式來設計，使得電路整體的運作更有效率。除此之外，使用 on-chip RAM 作為提供各單元的運算來源資料以及儲存各運算單元執行後的結果，搭配位置產生器以及控制器來自動產生記憶體位置和控制訊號，有效減少存取資料所需要花費的時間，因此本論文所提出的硬體架構降低了運算所需的時間並且獲得正確的還原結果。本論文所提出的硬體架構擁有低運算時間以及低資源消耗的優點，對於一些需要即時運算的嵌入式的 DHM 系統是個不錯的選擇。

參考著作

[1] E. Cuche, P. Marquet and C. Depeursinge, Simultaneous amplitude-contrast and quantitative phase-contrast microscopy by numerical reconstruction of Fresnel of-axis holograms, Appl. Opt., Vol. 38, pp. 6994–7001, 1999.

[2] D. Gabor, "A new microscopic principle," Nature 161, 777-778 (1948).

[3] Z. Li, Z. Bao, and Z. Suo, A joint image coregistration, phase noise suppression, and phase unwrapping method based on subspace projection for multibaseline InSAR systems, IEEE Trans. Geoscience and Remote Sensing, Vo. 45, pp.584-591, 2007.

[4] Loffeld, O. , Nies, H. , Knedlik, S. , Yu Wang , Phase Unwrapping for SAR Interferometry A Data Fusion Approach by Kalman Filtering, IEEE Trans. Geoscience and Remote Sensing, Jan 2008.

[5] S. Chavez, Q.S. Xiang, and L. An, Understanding Phase Maps in MRI: A New Cutline Phase Unwrapping Method, IEEE Trans. Medical Imaging, Vol. 21, pp.966-977, 2002.

[6] J.M. Bioucas-Dias and G. Valadao, Phase Unwrapping via Graph Cuts, IEEE Trans. Image Processing, Vol. 16, pp.684-697, 2007.

[7] D. C. Ghiglia and M. D. Pritt, Two-Dimensional Phase Unwrapping: Theory,

Algorithms and Software, 605 Third Avenue, New York, NY, 10158-0012: Wiley Inter-Science, 1998.

[8] V. Katkovnik, J. Astola, K. Egiazarian, ―Phase Local Approximation (PhaseLa) Technique for Phase Unwrap From Noisy Data, IEEE Trans. Image Processing, Vol. 17, pp.833-846, 2008.

[9] M. Born & E. Wolf, Principles of Optics, 1999, Cambridge University Press, Cambridge

[10] H. Oberst, D. Kouznetsov, K. Shimizu, J. Fujita, F. Shimizu. Fresnel diffraction mirror for atomic wave, Physical Review Letters, 94, 013203 (2005).

[11] Light," by Richard C. MacLaurin, 1909, Columbia University Press

[12] M.D. Pritt and J.S. Shipman, Least-Squares Two-Dimensional Phase Unwrapping Using FFT's, IEEE Trans. Geoscience and Remote Sensing, Vol. 32, pp.706-708, 1994.

[13] S. Hauck, and A. Dehon, Reconfigurable Computing, Morgan Kaufmann, 2008.

[14] Stratton, Julius Adams: Electromagnetic Theory, McGraw-Hill, 1941. (Reissued by Wiley IEEE Press, ISBN 978-0-470-13153-4).

[15] Sreeraman Rajan, Sichun Wang, Robert Inkol, Alain Joyal,"Efficient Approximations for the Arctangent Function,"Signal Processing Magazine, IEEE, volume 23 page 108-111 (2006)

[16] D. Parshall and M. K. Kim, Digital holographic microscopy with dual-wavelength phase unwrapping, Applied Optics, Vol. 45, pp.451-459, 2006.

[17] T. Shimobaba, Y. Sato, J. Miura, M. Takenouchi, and T. Ito, Real-time digital holographic microscopy using the graphic processing unit, Opt. Exp. 16 (16), 11776-11781, 2008.

[18] Etienne Cuche et al, "Simultaneous amplitude-contrast and quantitative phase-contrast microscopy by numerical reconstruction of Fresnel off-axis holograms," Appl. Opt. 38,6994-7001 (1999).

[19] U. Schnars and W. Juepner, "Digital Holography," Springer (2005).

[20] E.N. Leith and J.Upatnieks,"Wavefront reconstruction with diffused illumination and three dimensional objects," JOSA 54 ,1295-1301 (1964).

[21] J. W. Goodman and R. W. Lawrence, "Digital image formation from electronically detected holograms," Appl. Phys. Lett. 11,77–79 (1967).

[22] W. S. Haddad et al, "Fourier-transform holographic microscope," Appl. Opt. 31, 4973–4978 (1992)

[23] K. Boyer et al, "Biomedical three-dimensional holographic microimaging at visible, ultraviolet and X-ray wavelength," Nat. Med. 2, 939–941 (1996).

[24] P.A. Karasev, D.P. Campbell, and M.A. Richards, Obtaining a 35x Speedup in 2D Phase Unwrapping Using Commodity Graphics Processors, Proc. IEEE Radar

Conference, pp. 574-578, April 2007.

[25] Y. C. Lin and C. J. Cheng, Determining the refractive index profile of micro optical elements using transflective digital holographic microscopy, J. Opt. 12, 115402, 2010.

[26] Y. C. Lin, C. J. Cheng and T.-C. Poon, Optical sectioning with a low coherence phase-shifting digital holographic microscope, Appl. Opt. 50(7), B25-B30, 2011.

[27] C. J. Mann, L. Yu, C.-M. Lo, and M. K. Kim, High-resolution quantitative phase-contrast microscopy by digital holography, Optics Express, Vol. 13, pp.8693-8698, 2005.

[28] P. Mistry, S. Braganza, D. Kaeli, and M. Leeser, Accelerating Phase Unwrapping and Affine Transformations for Optical Quadrature Microscopy using CUDA, Proc. Second Workshop on General Purpose Processing on Graphics Processing Units, 2009.

[29] S. Braganza and M. Leeser, An efficient implementation of a phase unwrapping kernel on reconfigurable hardware, Proc. International Conference on Application Specific Systems, Architectures and Processors, pp.138-143, 2008.

[30] H. Calderon, C. Elena, and S. Vassiliadis, Soft Core Processors and Embedded Processing: a survey and analysis, Proc. Safe ProRisc Workshop, pp.483-488, 2005.

[31] Altera Corporation, FFT MegaCore Function User Guide, 2011.

[32] Altera Corporation, Floating Point Mega Function User Guide, 2011.

[33] Altera Corporation, NIOS II Processor Reference Handbook, 2011.

[34] Inverse trigonometric functions,

http://en.wikipedia.org/wiki/Inverse_trigonometric_functions

[35] Digital holographic microscopy,

http://en.wikipedia.org/wiki/Digital_holographic_microscopy

[36] Holography, http://en.wikipedia.org/wiki/Holography

[37] Discrete cosine transform,

http://en.wikipedia.org/wiki/Discrete_cosine_transform