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 Kernel 為基礎之模糊分群演算法硬體架構實現

FPGA Implementation for Kernel-Based Fuzzy C-Means algorithm with Spatial Constraint

研究生： 歐 浩 聲 撰

中華民國 壹佰零壹 年 柒 月

中文摘要

本論文根據文獻[12]以及文獻[17]，以此兩則文獻中提到的 FCM-SC 分群演算法的硬體架構和 KFCM 演算法的硬體架構為基礎，實作以非線性高斯核函式為核距離計算之 KFCM[12] 再加上空間資訊[17] 後的分群演算法硬體電路，具有管線化以及可以同時計算所有分群之權重係數的能力。此架構改良了以往 KFCM 分群演算法對於有雜訊的資料做分群的問題，並且配合 KFCM 本身可以對非線性資料分群效果較好的能力，所以能夠廣泛地使用在許多的分群資料上，並且都有良好的辨識率。本論文使用 FPGA 實現我們提出的硬體架構，並使用人工雜訊圖片作為實驗測試資料。實驗結果顯示本架構對於有雜訊的非線性資料分群效果確實較 KFCM 佳，且架構簡單提供了日後高度的延伸性。

關鍵字：可程式邏輯陣列，KFCM演算法，FCM-SC演算法，系統程式晶片設計，KFCM-SC演算法。

Abstract

Based on the FCM-SC (Fuzzy *C*-Mean with spatial constraint) architecture in reference [12] and the KFCM (Kernel-Based Fuzzy C-Means) architecture in reference [17], KFCM-SC (Kernel-Based Fuzzy C-Means with spatial constraint) hardware architecture is proposed here with non-linear Gaussian kernel function and spatial constraint. Moreover, the KFCM-SC architecture also takes the advantage of the pipeline and it can compute all of the membership coefficients and centers concurrently. Compared to KFCM architecture, KFCM-SC architecture improves the segmentation ability for noisy data by computing the spatial information. With these advantages, it can deal with the non-linear data due to the kernel function, KFCM-SC architecture can be applied to wide of data and it can achieve better segmentation results. KFCM-SC architecture is implemented on FPGA and tested with noisy picture data. The segmentation result shows that KFCM-SC architecture definitely has a better ability with non-linear noisy data compared to KFCM. Because of the simple architecture of the KFCM-SC, it can be extended easily.

Keywords: FPGA, KFCM algorithm, FCM-SC algorithm, SOPC, KFCM-SC algorithm.

誌謝

感謝 黃文吉老師在這兩年內對我不辭辛勞的細心指導，無論是在研究上或者是待人處事方面均獲益良多，尤其感謝老師願意讓我在完成碩士論文的口試之後，支持我去瑞典烏普薩拉當交換學生一年，完成了我人生中去歐洲讀書、生活的夢想，在此致上最深的謝意，真的是很感謝!同時感謝國立台灣師範大學光電科技研究所　鄭超仁博士、清雲科技大學電子工程系　歐謙敏博士，能夠百忙之中撥冗參加本人的口試審查，並給予本人更多的建議與指導。

再來是感謝一起生活兩年的同窗們，偉豪、哲誠、子昕、嘉翎、陳昊還有相處融洽、令人懷念的已畢業學長們，銘彥、侃翰、宗懋、坤宏、士彰和斯閔，在課業、研究與生活上，一起成長與學習!

要感謝愛我的爸爸、媽媽、兩個弟弟、阿姨以及其他家人和許多好朋友們，因為你（妳）們一路上的支持與陪伴，使我得到許多精神上的鼓勵，並且完成研究所的學業。最後，要感謝可愛的女朋友林瑜庭，謝謝妳一直陪伴著我，做我的後盾給我安慰與鼓勵，使我有安定的力量和前進的動力。兩年的時光過得很快，縱使在碩士論文完成前經歷了一段一個多月使我精疲力盡的日子，但這一切對我來說都是一種成長，我相信往後回首這段求學時光，會是帶著微笑的。

本論文謹獻給一路上愛我和我愛的你們。

目 錄

附表目錄

附圖目錄

第一章 緒論

本章主要是說明本論文的研究背景、動機以及目的。

1.1 研究背景

Fuzzy C-Means (FCM) 演算法 [1, 2] 是一種能夠有效的將資料做分群的技術。以模糊理論為基礎，FCM 將每筆資料歸屬於分群過程中的每個質心，而不是只被歸屬於某一個質心。每筆資料對於每次分群過程中的質心都有一個對應的權重係數，而此係數也代表著該資料對於該質心的歸屬程度。相較於其他一般的分群技術，FCM 的好處是對於初始質心的敏感度比較低，也意味著可以避免只有處理到部分資料的分群最佳化。

然而，由於 FCM 演算法是使用距離平方來量測資料與質心的歸屬程度，所以當資料以圓形分布，則 FCM 分群的效果並不會很好。Kernel- FCM (KFCM) 演算法 [2, 3, 4] 可以用來處理這種問題。在 KFCM 中，是用 Kernel 函式做為計算權重係數以及質心的依據。雖然 KFCM 有較好的分群結果，它的計算複雜度在計算權重係數以及質心時卻很高。此外，權重係數矩陣也隨著要被分群的資料量和分群數的增加而跟著增加。如此龐大的權重係數矩陣會使得 KFCM 在實作上面遇到困難。另外，雖然 KFCM 分群法則對於影像的分割有不錯的效果，但影像分

割正確率卻會隨著雜訊的增加而跟著降低。在 FCM 中加上空間(Spatial)資訊可以有效的提高有雜訊影響下的影像分割正確率 [2, 5]。

有些對於基礎 FCM 可以快速計算的演算法 [6, 7, 8]，也可以用來加速 KFCM 的計算速度。然而，這些快速的演算法是用於軟體的實作上。因此只能加速一定範圍內的效果。雖然有許多對於 FCM 的硬體架構 [9, 10, 11] 已經被提出，這些架構卻僅僅只用於基礎的 FCM 演算法則。此外，在論文 [9] 的硬體架構中，該設計是以類比電路為基礎。因此分群的結果很難直接被用在數位電路的應用上。雖然在論文 [10] 的架構是用於數位電路，而該架構卻僅僅用於分群數為兩群的狀況，對於要分比較多群的狀況會不敷使用。在論文 [11, 12] 中，其所提出的 FCM 硬體架構，是用於數位電路且可以做分群數為兩群以上的電路。然而，在 [11, 12] 中所提出的電路架構並沒有加入空間資訊的設計。另外，在現有的論文 [10, 11, 12] 所提出的硬體架構中，都是建立於定點數的表示。若在實作時使用的是浮點數的表示，FCM 的分群結果或許可以被提升。

1.2 研究動機與目的

基於上一節所敘述，本論文的目標就是對於 KFCM 演算法提出一個創新的硬體架構用於處理在雜訊影響下的影像分割。在本論文提出的 KFCM 中，使用的是高斯的核函式 [2, 3] ，因為其對於圖形能夠做有效的分群和處理。空間資訊(Spatial Constraint)也被加入了進來，用來強化對於雜訊抵抗的能力。而本論文提出的硬體架構，使用的是浮點數的計算方式，也因此分群出來的結果可以和軟體算出來的結果相符合。此電路架構可以有效地計算核函式、權重係數以及質心，像是同步地計算權重係數以強化電路的效率。此外，此電路架構把在對於每筆資料做分群所需計算的權重係數矩陣以及質心合併成一次電路的流程，也節省了許多的儲存空間。本論文所提出的電路藉由現場可編程輯閘陣列 (FPGA) [13] 實作。在實作時，藉由可程式化系統晶片 (SOPC) 上將所提出的客製化電路和整塊板子的電路結合，可讓板子上的處理器運算時使用此塊客製化電路，達到量測上面的加速效果。所提出的架構，相較於軟體的 Intel i5 處理器，擁有很高倍的加速結果。而和傳統的 FCM 演算法比起來，此架構也包含了在處理有雜訊的圖片下，可以得到相當好的圖片分割結果。

1.3 全文架構

本篇論文共分為五章，以下為各章的內容概述：

【第一章】緒論

說明本論文的研究背景、研究動機、研究目的和全文的架構。

【第二章】理論基礎與技術背景

簡介本論文使用的基礎理論 KFCM 與 KFCM with spatial constraint 演算法及技術背景。

【第三章】基礎電路架構介紹

詳細說明本論文所提出的 KFCM with spatial constraint 演算法的基礎電路，及其內部各單元架構。

【第四章】實驗結果與數據探討

呈現本論文所提出的電路設計成果數據、以及討論。

【第五章】結論

對本論文做最後的總結。

第二章 理論基礎與技術背景

本章將討論本論文的理論基礎與技術背景，首先介紹基礎的 KFCM 分群演算法與 KFCM with spatial constraint 演算法，以及介紹 FPGA 技術與 SOPC 系統整合設計，方便讀者對本論文有初步的了解與認識。

2.1 Kernel-Based Fuzzy C-Means 演算法

假設現有一集合 $\boldsymbol{X} = \{x_1, \ldots, x_t\}$，此集合要被分割到 c 個群組，其中每一個群組 i，$1 \leq i \leq c$，都被各自的質心 v_i 所代表。在影像的分割應用上，$\boldsymbol{X}$ 代表著一張要做影像分割的圖片，x_k是在 $\boldsymbol{X}$ 中的某一個區塊，t 是 $\boldsymbol{X}$ 所有的區塊數，c 要分群的群數。在 KFCM 中的目標即是要最小化目標函式 [2]：

$$J = \sum_{i=1}^{c} \sum_{k=1}^{t} u_{i,k}^{m} \left\| \Phi(x_k) - \Phi(v_i) \right\|^2 \quad (1)$$

Φ是一個映射函數，$u_{i,k}^{m}$是x_k在第 i 群中的權重係數，$m>0$ 代表著法則的模糊程度，在此定義一個核函式$K(x,y) = \Phi(x)^T \Phi(y)$為$\Phi$的映射函數，因此:

$$\left\| \Phi(x_k) - \Phi(v_i) \right\|^2 = K(x_k, x_k) + K(v_i, v_i) - 2K(x_k, v_i) \quad (2)$$

而代入常用的高斯核函式，此式為:

$$\mathrm{K(x,y)} = \mathrm{e}^{-\frac{\|\mathrm{x}-\mathrm{y}\|^2}{2\sigma^2}} \quad (3)$$

將公式(3)帶入公式(2)，得到$\left\| \Phi(x_k) - \Phi(v_i) \right\|^2 = 2 - 2K(x_k, v_i)$，因此公式(1)可以被改寫成

$$J = 2\sum_{i=1}^{c}\sum_{k=1}^{n} u_{i,k}^{m}\,(\,1 - K(x_k, v_i)\,) \tag{4}$$

此目標函式 J 在 KFCM 中被簡化為兩層的重複計算，在第一層計算中，固定質心 $v_1, \ldots, v_c$，接著計算對應的最佳化的權重係數為:.

$$u_{i,k} = \left(\frac{1}{1-K(x_k,v_i)}\right)^{\frac{1}{m-1}} \Big/ \sum_{j=1}^{c}\left(\frac{1}{1-K(x_k,v_j)}\right)^{\frac{1}{m-1}} \tag{5}$$

在第二層計算中，反過來固定權重係數，而對於每個質心 i 的公式為:

$$v_i = \sum_{k=1}^{t} u_{ik}^{m} K(x_k, v_i) x_k / \sum_{k=1}^{t} u_{ik}^{m} K(x_k, v_i). \tag{6}$$

資料x_k在每一個質量重心 v_i 的歸屬程度介於 0~1 之間，即$\sum_{i=1}^{c} u_i = 1$反覆計算公式(5)與公式(6)使公式(1)的 J 值的改變量小於某一臨界值時，則視其為一穩定狀態，最後得到一組新的質量中心點$\{v_1,\ldots,v_c\}$，下方文字為 KFCM 演算法的流程，下一頁為 KFCM 演算法的流程圖。

Step1: 設定初始質量重心 $v_1, v_2, \ldots v_c$，分群的個數 c 群，收斂條件 ε 以及模糊程度 m 值，與核函式的變異係數值σ。

Step2: 計算權重係數如公式(6)，以得到更新過後所有資料再全部質量重心的權重係數值。

Step3: 將 *Step2* 的結果帶入公式(7)，計算新的質量中心點。

Step4: 將 *Step3* 的結果帶入公式(1)，計算新的目標函式 J 值。

Step5: 判斷 J 值的運算結果與前一次運算結果之間的差值比例是否小於收斂條件，若達到收斂條件，表示此演算法已經趨於穩定；反之，則將新計算出的質量重心點取代舊的質量重心點，回到 *Step2* 繼續。

Step5: 將收斂條件後的質量重心代入(9)運算式，得到收斂後資料中每個點的分群結果。

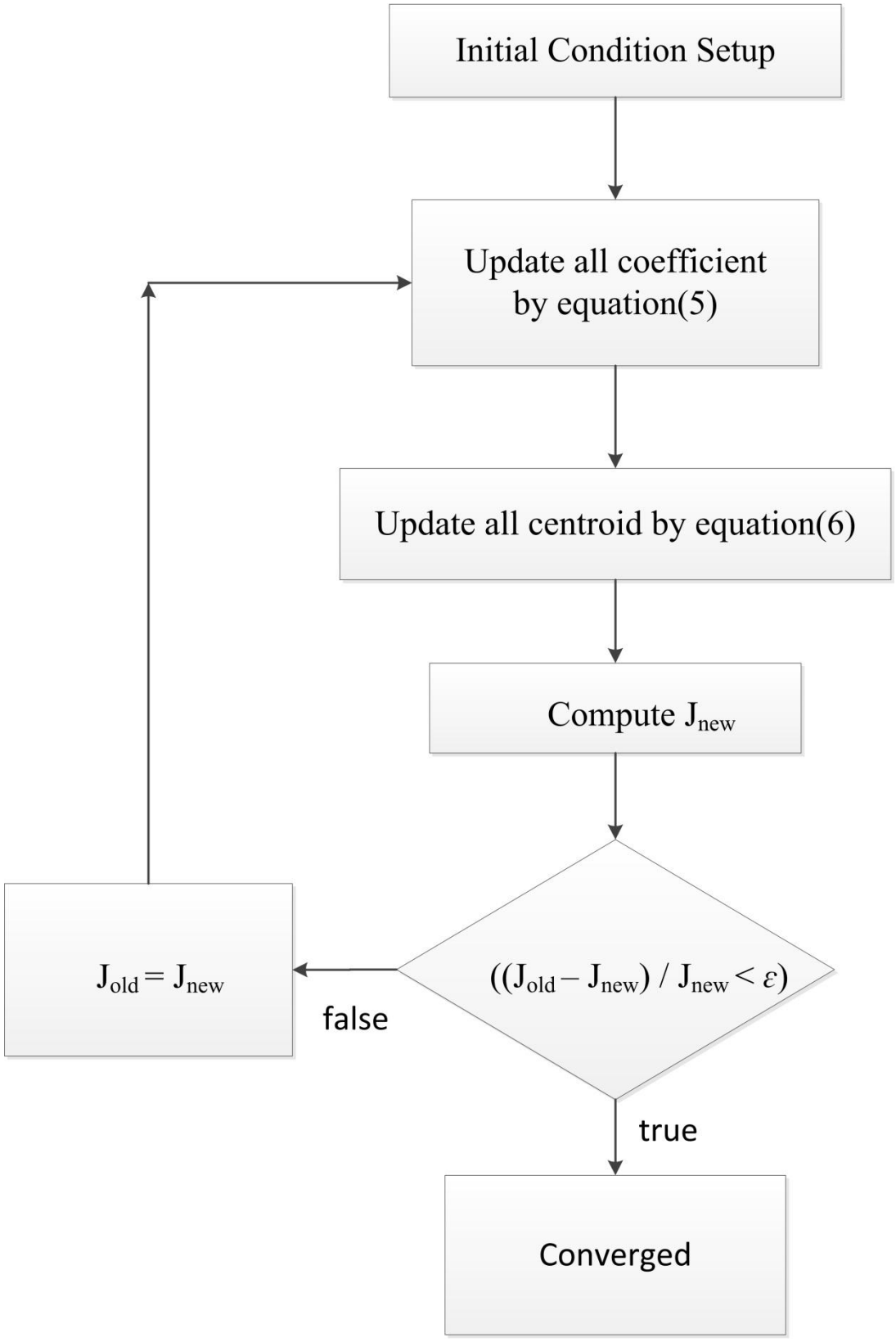


圖 2.1 KFCM 演算法流程圖

2.2 Kernel-Based Fuzzy C-Means with spatial constraint 演算法

儘管 KFCM 是有效的，在做影像分割時卻沒有加入空間資訊。因此，當現有一張受到雜訊干擾的圖片要做分割時，分割的成功率就會降低。加入空間資訊後的 KFCM 為 KFCM with spatial constraint (KFCM-SC) 可以用來解決這個問題。而 KFCM-SC 可以視為 KFCM 和 FCM-S 的結合 [2, 12]，而 KFCM-SC 是要最小化目標函式以下:

$$J = \sum_{i=1}^{c}\sum_{k=1}^{t} u_{i,k}^{m}\,(\,1 - K(x_k, v_i)\,) + \alpha\sum_{i=1}^{c}\sum_{k=1}^{t} u_{i,k}^{m}\,(\,1 - K(\bar{x}_k, v_i)\,) \tag{7}$$

假設現有一區塊以x_k為中心，$\bar{x}_k$即代表著該區塊內的平均值，而α的大小代表著空間資訊在分群過程中所佔的比重。若現有一組質心$v_i, i = 1,..,c$，則計算對應的最佳化的權重係數改寫為:

$$u_{i,k} = \frac{\left(\frac{1}{(1-K(x_k,v_i))+\alpha(1-K(\bar{x}_k,v_i))}\right)^{\frac{1}{m-1}}}{\sum_{j=1}^{c}\left(\frac{1}{(1-K(x_k,v_j))+\alpha(1-K(\bar{x}_k,v_j))}\right)^{\frac{1}{m-1}}} \tag{8}$$

計算質心公式也接著化簡為:

$$v_i = \frac{\sum_{k=1}^{t} u_{ik}^{m}(K(x_k,v_i)x_k + \alpha K(\bar{x}_k,v_i)\bar{x}_k)}{\sum_{k=1}^{t} u_{ik}^{m}(K(x_k,v_i) + \alpha K(\bar{x}_k,v_i))} \tag{9}$$

KFCM 和 KFCM-SC 都需要很多的浮點數運算，從公式(4)到(6)，需要儲存權重係數去計算質心和目標函式J，再由於權重係數總數會隨著 t 和 c 增加而跟著增加，因此在計算 KFCM 和 KFCM-SC 的過程中，當需要分群的資料總數增多或者

分群的群數變多，所需要的空間會非常大。

2.3 SOPC 系統整合設計

隨著科技的進步與近年來微電子技術及其應用快速的發展，產品生命週期日益下降，功能日益複雜，如何在短時間內開發出好的產品是個重要的議題。由於在傳統的系統設計上，軟體與硬體是分開設計的，若在整合軟硬體時出現錯誤，除錯會變得非常困難，必須花費大量的時間重新檢查軟硬體，也因此延遲了產品開發時間，降低了競爭力。為了改善軟硬體分開設計的缺點，可程式化系統晶片設計(System On Programmable Chip)是一種新的軟硬體整合設計技術，帶來的靈活性，使過去耗費、刻板的硬體設計變得像軟體設計一樣容易除錯修改，可以加速產品的開發設計，提升整體的競爭力，大幅降低設計的時間與成本。

Altera 公司根據不同使用者的需求，開發出許多不同系列的 FPGA 開發板，並且推出 PLD(programmable logic device)設計軟體 Quartus II 供使用者使用，Quartus II 是一個在做硬體開發時很常被使用的開發環境。我們可在此環境寫硬體描述語言 Verilog HDL、VHDL，並藉由模擬圖、RTL 圖、自動機圖…等不同的工具去協助我們做電路的驗證，是一款很好用的硬體開發環境。本研究是在 NIOS development kit 中的 Stratix Ⅲ EP3SL150F1152C2 系統開發板上實現我們所提出的硬體電路。

在開發客製化電路上，也提供使用者許多需要的設計元件，使用者可以針對

自己的需求做選擇，並藉由 SOPC 系統將電路和自己設計的客製化電路整合，如圖 2.2。在此是將使用者的客製化電路掛載到 FPGA 板子上面的匯流排(Avalon Bus)，如圖 2.4。該客製化電路的所有輸入以及輸出訊號線都藉由此匯流排與開發板上面的系統作溝通，編譯完後即可燒錄到開發板上使用。

而在燒錄到開發板上後，最後藉由 Altera 公司提供的 NIOSⅡ IDE 視窗介面的軟體開發工具，設計者就可以在上面編寫程式碼、編譯、除錯及觀察程式執行結果。整體的實作流程如圖 2.3 所示。

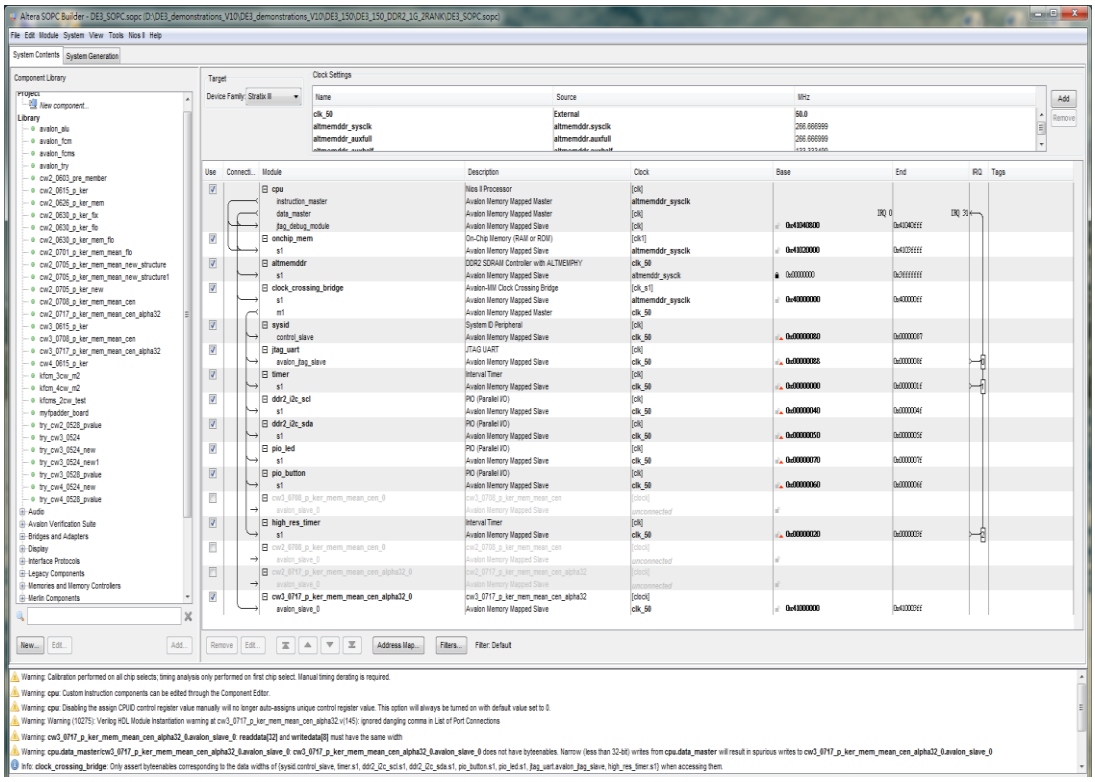


圖 2.2 SOPC 系統介面

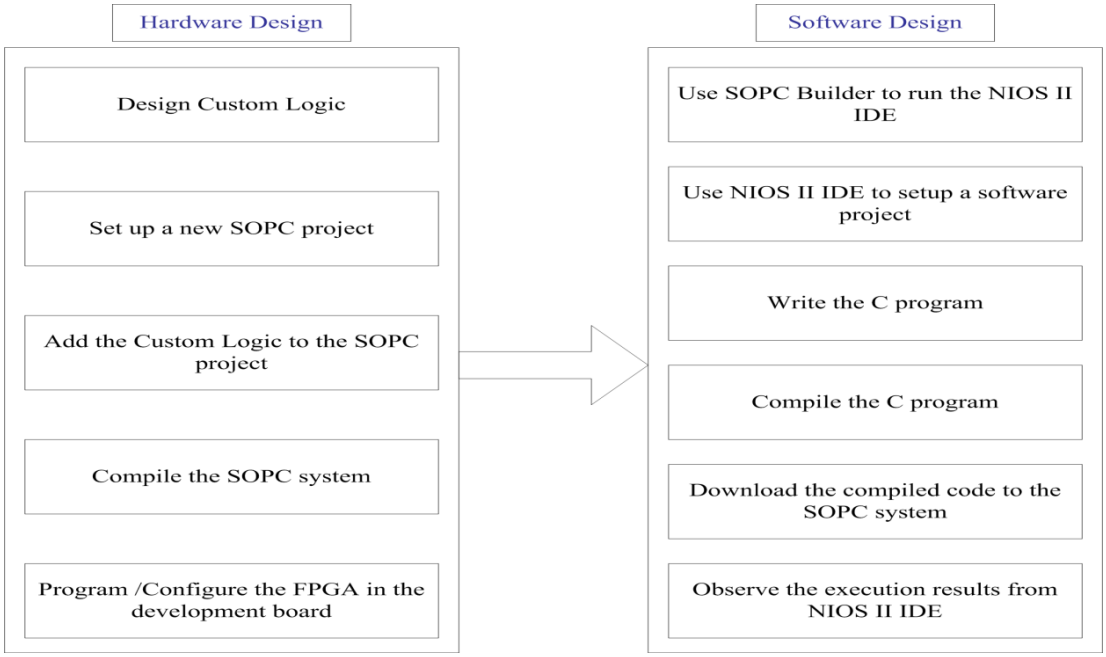


圖 2.3 軟硬體共同設計流程圖

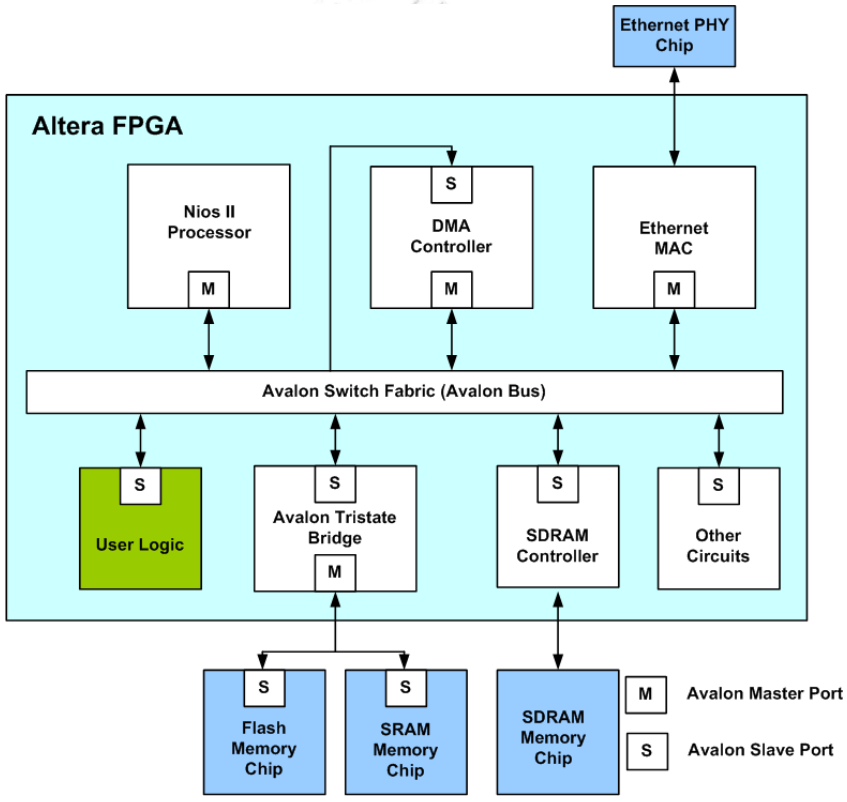


圖 2.4 FPGA 上的 Avalon System

第三章 基礎電路架構介紹

在沒有雜訊的環境下，KFCM 分群演算法適合用在影像分割上，但考量實際情況，將 KFCM 分群演算法進行調整是必要的。因此 KFCM-SC 演算法加入了每筆資料點x_k的周遭間資訊進行分群的計算，以減少雜訊影響影像分割後的結果。

綜觀本論文提出的 KFCM-SC 的硬體架構，其包含兩個單元，如圖 3.1，首先是 Mean Computation Unit，此單元的目的是計算每筆資料x_k與其周遭空間資訊的平均值$\bar{x}_k$；另一個就是 KFCM-SC 演算法的計算部分電路。本節接下來將討論此兩大單元。

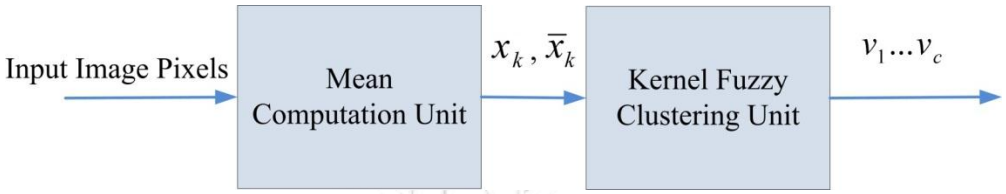


圖 3.1 KFCM with spatial constraint 架構圖

3.1 Mean Computation Unit

給定一個資料x_k，$\bar{x}_k$是在空間上以x_k為中心的鄰居共佔 3x3 個區塊的加總平均。假設圖檔是由 *NxN* 的長與寬所構成，此電路是由 2*N*+3 個 stage 的位移暫存器所組成，負責傳遞資料x_k，如圖 3.2。

在 Mean Computation Unit 中含有一個輸入埠與兩個輸出埠: 輸入埠為資料x_k，輸出埠為資料x_k和其與周遭的平均值$\bar{x}_k$。當第一個時脈週期來時，第一筆資料x_1進入 cell 1，即第一個位移暫存器，第二個時脈來臨時，原先在 cell 1 的訓練

向量傳遞至 cell 2，同時第二筆的訓練向量x_2進入 cell 1，以此類推。由於圖形邊界的資料並沒有滿足 3x3 個區塊的鄰居數，因此不考慮邊界平均值處理的情況。所以從 Mean Computation Unit 輸出且有意義的值是從第 *N*+2 個時脈之後出現的資料，而此向量的所有鄰居在第 2*N*+3 個時脈後才會全部充斥在管線化的架構中，因此我們需要 2*N*+3 個 stage 保留所有的需要的鄰居資訊。

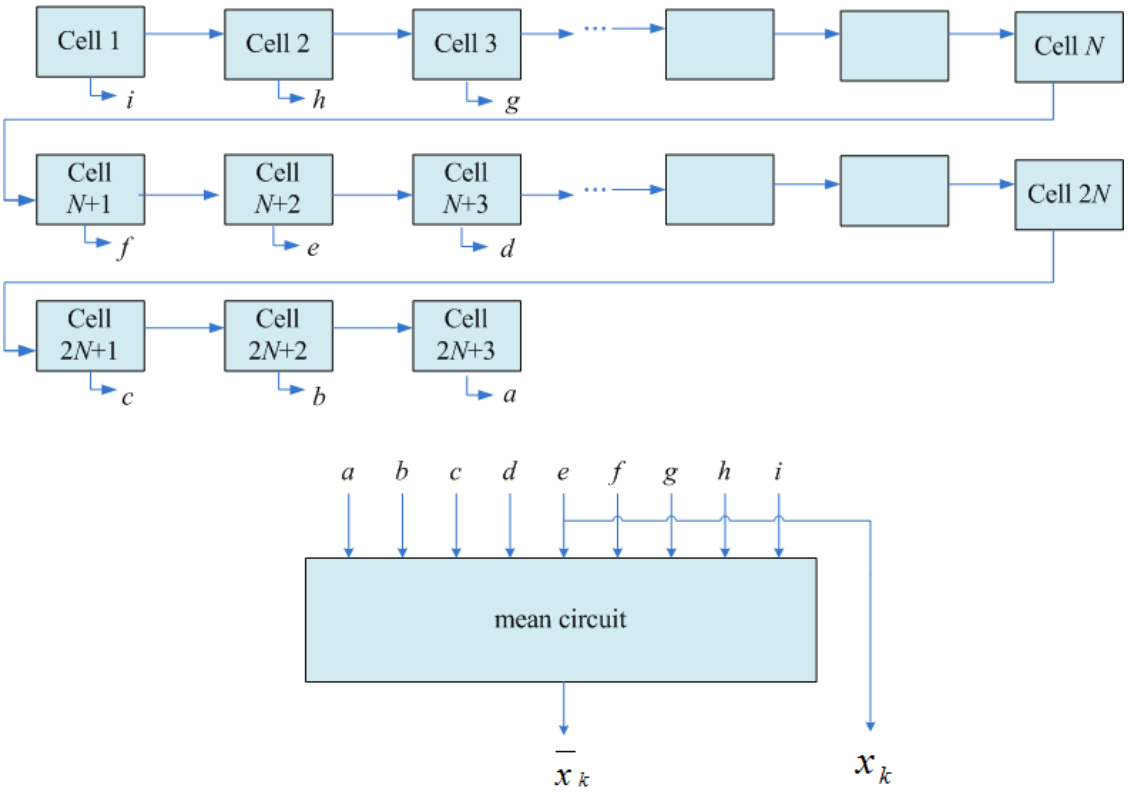


圖 3.2 Mean Computation Unit 架構

此電路的設計優點為可以百分之百的利用資料，當每一個資料離開最後一個 stage 的位移暫存器時，代表不再需要參與每個資料的平均值計算。

3.2 KFCM-SC 分群演算法電路

本論文提出的 KFCM-SC 分群演算法電路和 KFCM 電路大致上有著相同的架構組合， 其中包含 Pre-Computation Unit， Membership Coefficients Updating Unit， 和 Center Updating Unit，而 KFCM-SC 比 KFCM 多出一塊計算影像區塊平均值的電路，如圖 3.3 所示。此外 KFCM 可視為 KFCM-SC 中的$\alpha = 0$的特例。以下我們將詳細介紹 KFCM-SC 的電路。

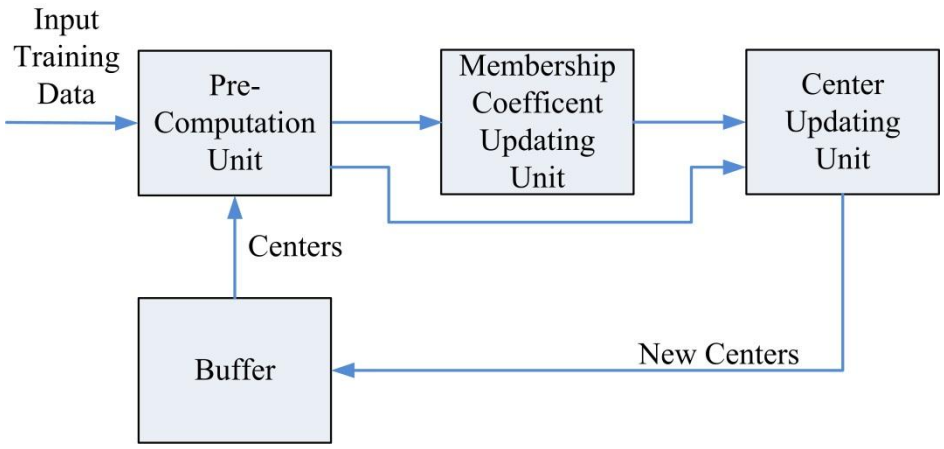


圖 3.3 KFCM-SC 計算流程圖

3.2.1 Pre-Computation Unit

此塊電路主要用來降低在計算權重係數和質心時的計算複雜度，在此使用模糊係數 *m*=2，因此公式(8)可以簡化為:

$$u_{i,k} = \Big(\big(1 - K(x_k, v_i)\big) + \alpha\big(1 - K(\bar{x}_k, v_i)\big)\Big)^{-1} P_k^{-1} \tag{10}$$

而

$$P_k = \sum_{j=1}^{c} \frac{1}{\left(1-K(x_k,v_j)\right)+\alpha\left(1-K(\bar{x}_k,v_j)\right)} \tag{11}$$

因此，計算權重係數的複雜度即可在此單元藉由計算此P_k而降低。在公式(11)中，可以注意到包含了公式(3)核函式的計算，圖 3.4 所示即為核函式的架構。

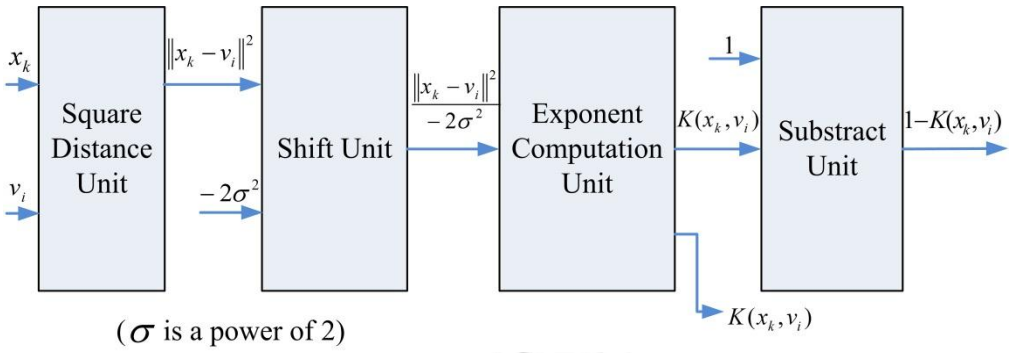


圖 3.4 核函式(Kernel Computation Unit) 架構圖

其中包含了計算距離平方以及做高斯指數運算的部分。要特別注意的是，在公式(3)中除以σ^2的除法運算，可以藉由設定此σ^2為 2 的次方而避免掉。做高斯指數運算的部分，使用的是 Altera 公司提供的浮點數指數(ALTPF_EXP) 計算功能 (megafunction) [15] ，而根據核函式的計算，Pre-Computation Unit 可以被分成 c 個 stage，其中每個 stage i 如圖 3.5，所示。

每一個 stage i 會接收來自前一個 stage i-1 所計算完的$P_k(i-1)$並會計算 $P_k(i)$ 和$\Delta P_k(i)$。$P_k(i-1)$和$\Delta P_k(i)$如下定義:

$$P_k(i-1) = \sum_{j=1}^{i-1} \left(\left(1-K\left(x_k,v_j\right)\right)+\alpha\left(1-K\left(\bar{x}_k,v_j\right)\right)\right)^{-1} \tag{12}$$

$$\Delta P_k(i) = \Big(\big(1 - K(x_k, v_i)\big) + \alpha\big(1 - K(\bar{x}_k, v_i)\big)\Big)^{-1} \quad (13)$$

因此，

$$P_k(i) = P_k(i-1) + \Delta P_k(i) \quad (14)$$

從公式(10)到公式(13)，可以觀察到

$$u_{i,k} = \Delta P_k(i) P_k^{-1} \quad (15)$$

所以在 Pre-Computation Unit 中的第 stage i ，會傳遞$P_k(i)$和$\Delta P_k(j), j = 1, \dots, i$到下一個 stage i+1 及接下來的所有 stage。到了最後一個在 Pre-Computation Unit 中的 stage 時，在此即是指 stage c，就可以得到最終要傳遞給接下來要計算的權重係數所需的$P_k(i)$以及$\Delta P_k(j), j = 1, \dots, c$。除了上述的$P_k(i)$和$\Delta P_k(i)$，在 Pre-Computation Unit 中的第 stage i，也會產生$\Delta R_k(i)$和$\Delta S_k(i)$，定義如下:

$$\Delta R_k(i) = K(x_k, v_i) x_k + \alpha K(\bar{x}_k, v_i) \bar{x}_k \quad (16)$$

$$\Delta S_k(i) = K(x_k, v_i) + \alpha K(\bar{x}_k, v_i) \quad (17)$$

此兩個參數是為了方便之後要計算質心位置所需，所以在 Pre-Computation Unit 中的最後一個 stage，這兩個參數$\Delta R_k(j)$和$\Delta S_k(j), j = 1, \dots, c$也會被一起傳遞到之後的 Center Updating Unit 去計算最後新的質心。

這裡的 Pre-Computation Unit 使用管線化的架構去更進一步的增強電路的產能，圖 3.6 是有三個 stage 的 Pre-Computation Unit (*c=3*)管線化架構。

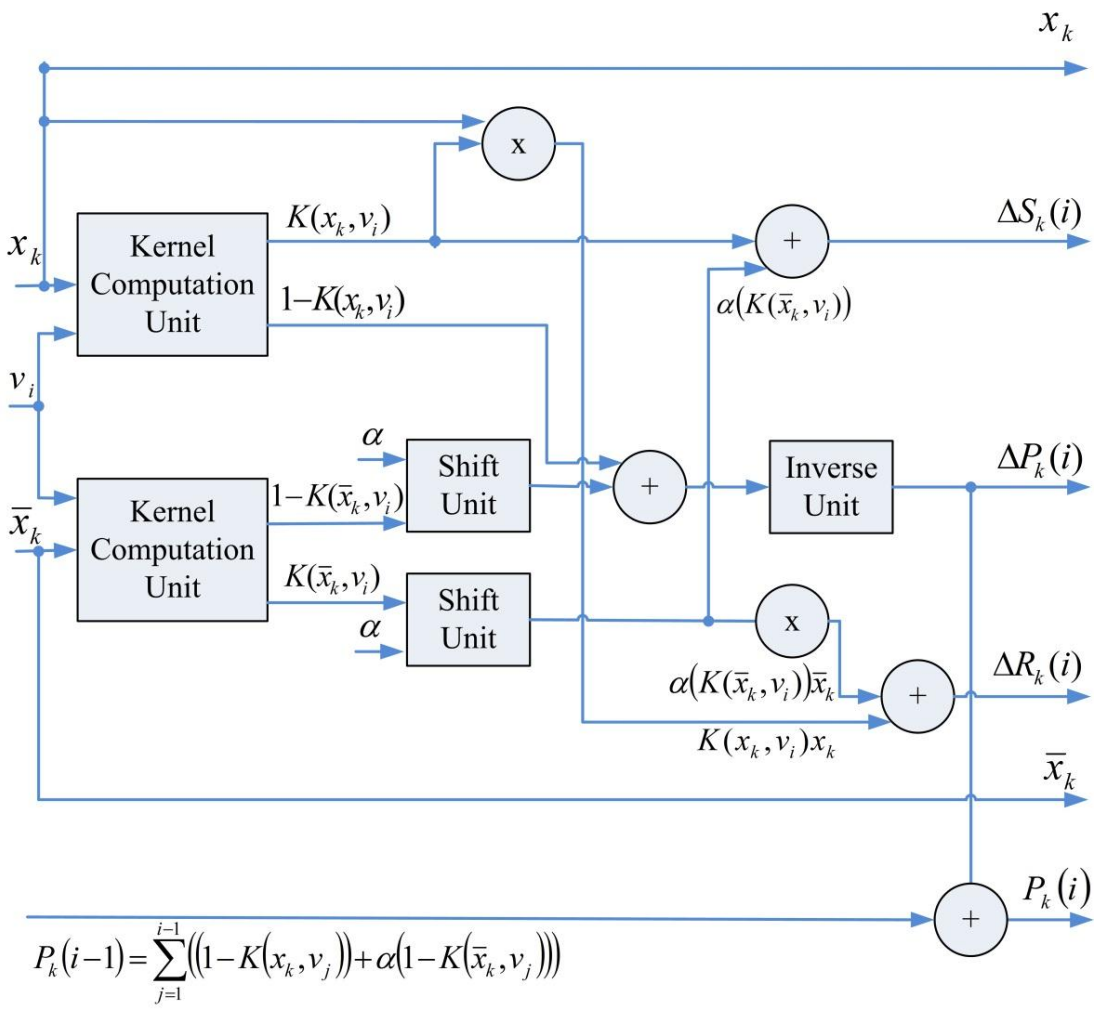


圖 3.5 Pre-Computation Unit 內部架構圖

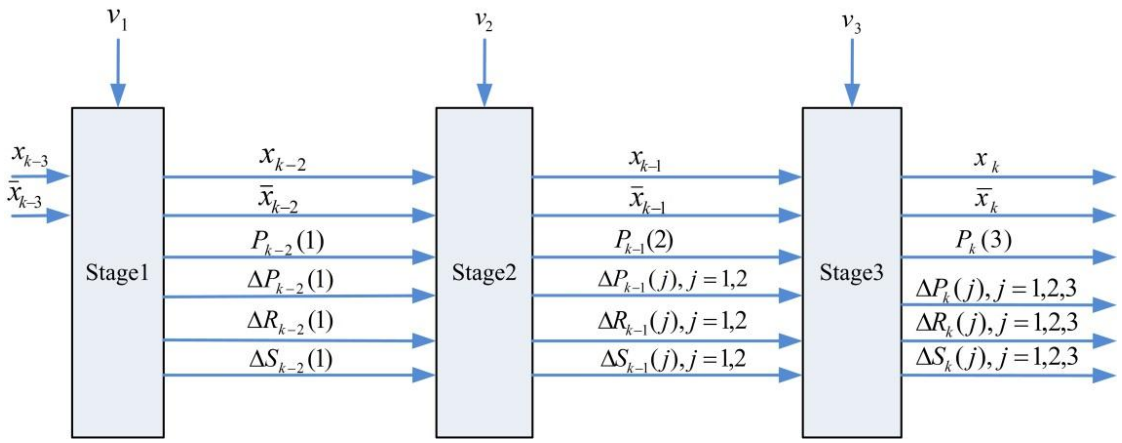


圖 3.6 管線化 Pre-Computation Unit (以 *c=3* ，即 3 群為例)

3.2.2 Membership Coefficient Updating Unit

對於每一個資料x_k，在計算權重係數時，都是同步地在此 Membership Coefficient Updating Unit 做計算；因此，在此 Unit 中，包含了 c 塊模組，如圖 3.8 所示，每一塊模組分別負責計算一筆對應的權重係數，而在圖 3.7 表示的是每一塊計算權重係數模組的內部架構。

根據公式(15)，可以觀察到每一塊計算權重係數的模組會從前面的 Pre-Computation Unit 接收到$\Delta P_k(i)$和P_k，在此模組中包含了乘法器以及一個做倒數運算用的運算單元，以用來計算所需的$u_{i,k}^2$。Membership Coefficient Updating Unit 的最後輸出$u_{i,k}^2$，將會被傳遞到下一個 Center Updating Unit 去計算新的質心。

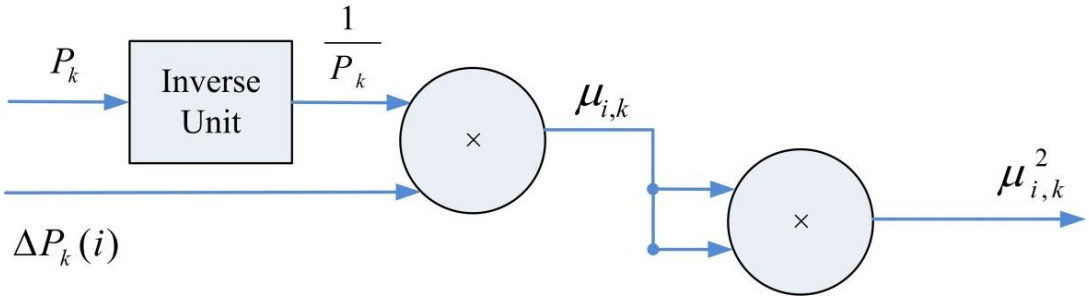


圖 3.7 Membership-Coefficient Updating Unit 架構

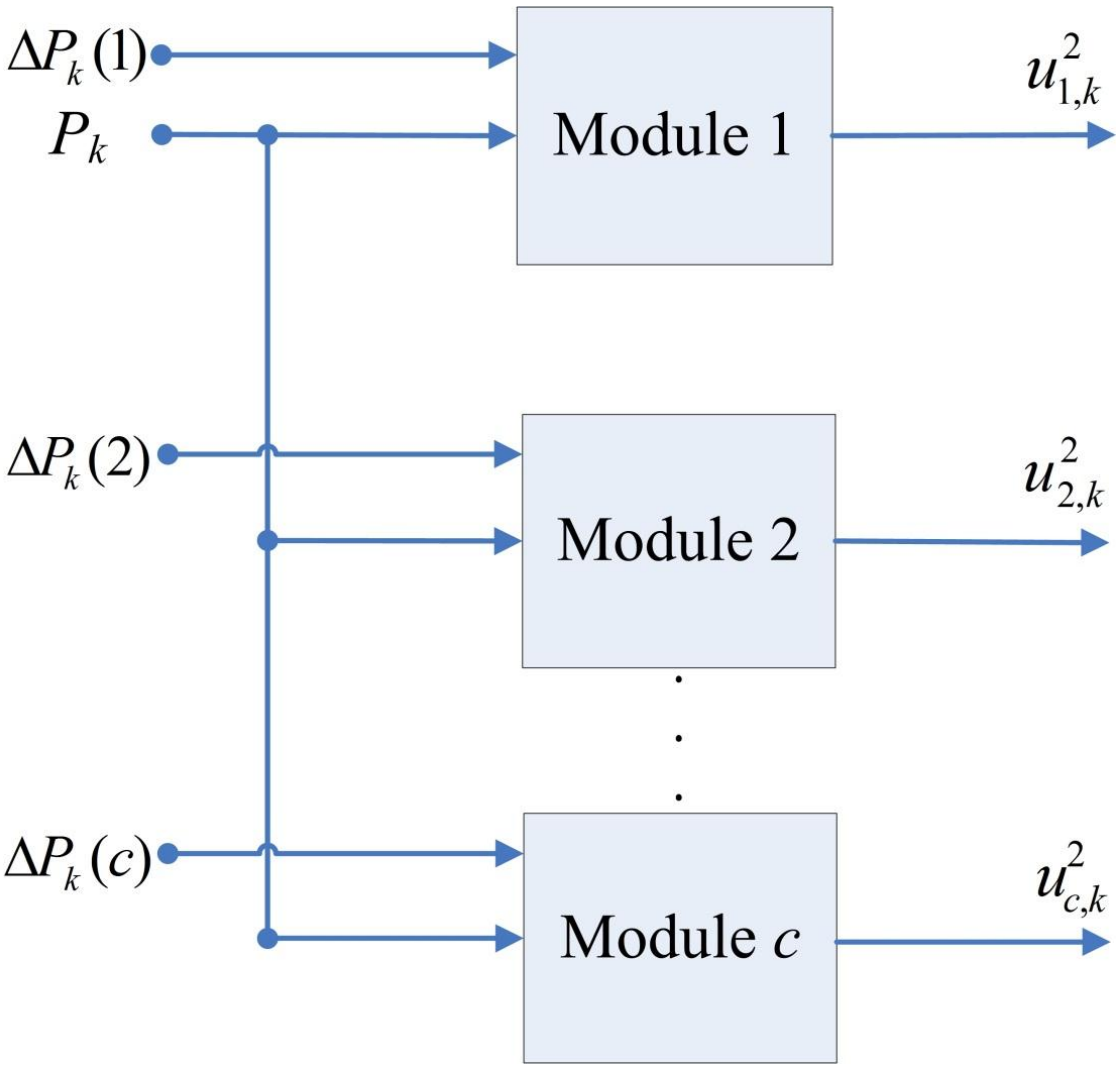


圖 3.8 管線化的權重係數更新單元架構(以 c 群為例)

在此圖中，說明的是同一時間對於 c 個質心來說，有 c 個模組同時在計算其對應的權重係數值，將每個模組所需的P_k廣播到此 c 個模組內，並傳遞每個模組各自的需求值$\Delta P_k(c)$，而之後也會同時算出對應的權重係數$u^2_{i,k}$。

3.2.3 Center Updating Unit

類似前面提到的 Membership Coefficient Updating Unit，這裡的 Center Updating Unit 也包含了 c 塊模組，而這 c 塊模組對於每一個資料x_k也都是同步地進行質心的運算，如圖 3.10。而在這裡採用累加的方式去計算新的質心最主要的好處是，可以不需要像在公式(9)中提到的大量空間去儲存整塊權重係數矩陣後再來做質心的運算。因此該公式在此可以再定義成:

$$v_i(k) = \frac{\left(u_{ik}^2(K(x_k,v_i)x_k+\alpha K(\bar{x}_k,v_i)\bar{x}_k)+\sum_{n=1}^{k-1}u_{in}^2(K(x_n,v_i)x_n+\alpha K(\bar{x}_n,v_i)\bar{x}_n)\right)}{\left(u_{ik}^2(K(x_k,v_i)+\alpha K(\bar{x}_k,v_i))+\sum_{n=1}^{k-1}u_{in}^2(K(x_n,v_i)+\alpha K(\bar{x}_n,v_i))\right)} \quad (18)$$

需要注意的是在公式(18)中的分子: $\sum_{n=1}^{k-1}u_{in}^2(K(x_n,v_i)x_n+\alpha K(\bar{x}_n,v_i)\bar{x}_n)$ 以及分母: $\sum_{n=1}^{k-1}u_{in}^2(K(x_n,v_i)+\alpha K(\bar{x}_n,v_i))$，此兩項在計算完前一筆質心$v_i(k-1)$時就已經儲存在暫存器中了，也因如此，不需要為了每次要計算質心時而做重複地計算，如下頁圖 3.9 所示。此外，藉由公式(16)和公式(17)，可以把公式(18)，改寫成:

$$v_i(k) = \frac{\left(u_{ik}^2\Delta R_k(i)+\sum_{n=1}^{k-1}u_{in}^2\Delta R_n(i)\right)}{\left(u_{ik}^2\Delta S_k(i)+\sum_{n=1}^{k-1}u_{in}^2\Delta S(i)\right)} \quad (19)$$

$\Delta R_k(i)$和$\Delta S_k(i)$ 可以從 Pre-Computation Unit 獲得，而權重係數u_{ik}^2可以從 Membership Coefficient Updating Unit 中獲得。因此，可以大大地簡化計算$v_i(k)$的方式。

比較公式(9)和公式(18)，可以輕鬆地發現，在模糊係數 *m=2* 的情況下，當累

加到 $k=t$，$v_i(k) = v_i$，新的質心就被本論文提出的電路算出來了。

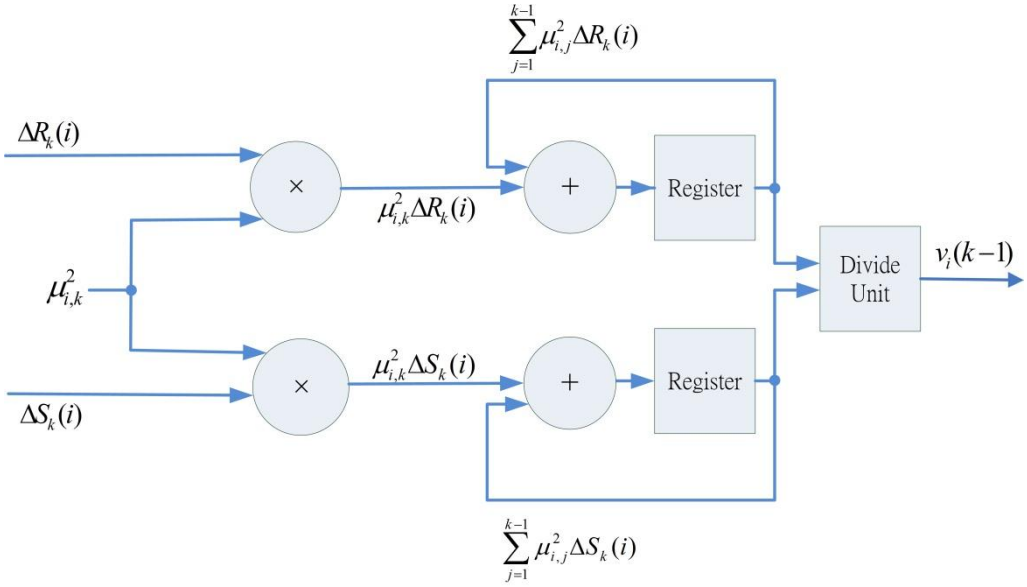


圖 3.9 Center Updating Unit 內部架構圖

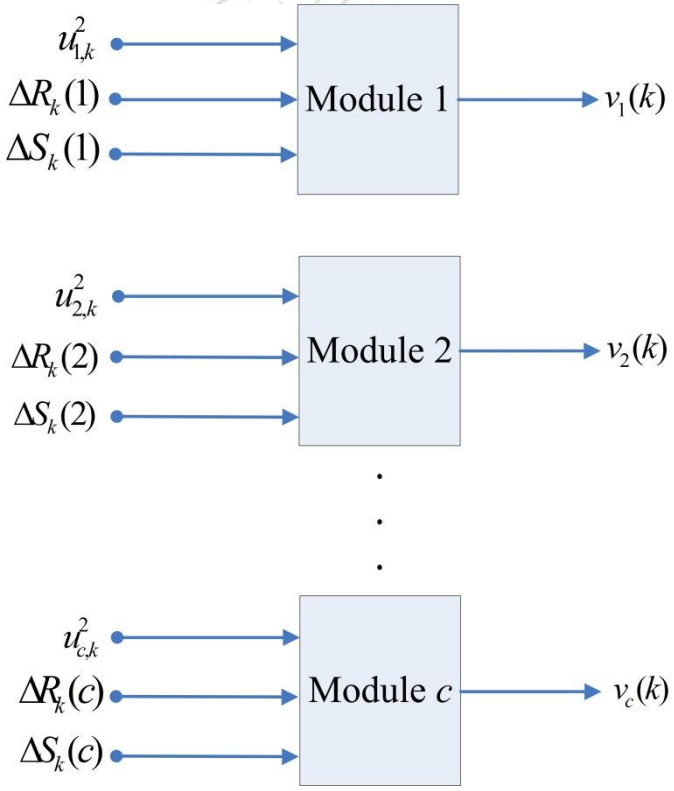


圖 3.10 管線化 Center Updating Unit 架構

第四章 實驗結果與數據探討

本章節主要為介紹實驗環境以及呈現我們的實驗結果。

4.1 開發平台與實驗環境介紹

由於 FPGA 具有可程式化的彈性，可以重複修改與快速上市等優點，使設計人員更容易建置客製化的硬體電路在 SOPC 系統上，因此 FPGA 非常適合用來實現本論文提出的硬體架構。表 4.2 為軟、硬體的實做環境。圖 4.1 為本論文實驗時所使用的 FPGA 開發板，由 Terasic 公司生產，該版型號為 DE3，而 FPGA 上面使用的晶片為 Altera 公司製造的 EP3L150F1152C2。表 4.1 為 Altera DE3 EP3L150 FPGA 開發板的詳細規格資訊：

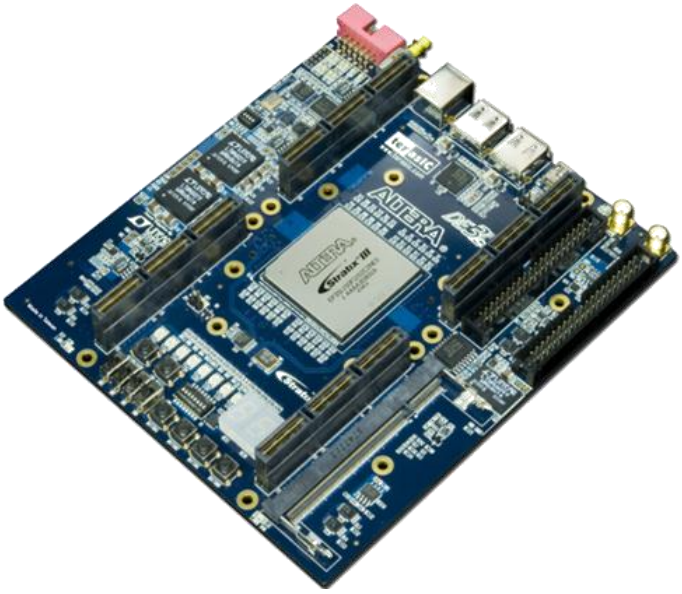


圖 4.1 Altera DE3 EP3L150 實驗開發板

表 4.1 DE3 EP3L150 FPGA 開發板的詳細規格

Device feature	Number
Combinational ALUTs	113,600
Memory ALUTs	56800
Dedicated logic registers	113600
Total pins	744
Total block memory bits	5630976
DSP block 18-bit elements	384
Total PLLs	8
Total DLLs	4

表 4. 2 軟、硬體實現環境

軟體實現環境	硬體實現環境
處理器：Intel i5 2410M 2.3GHz	處理器 ：NOIS II 50MHz
記憶體：DDR3 8.0GB 1333MHz	記憶體 ：DDR2 1.0GB
程式語言:C 語言	程式語言: Verilog-HDL 語言
編譯器：DevC++ 4.9.9.2	開發板 ：Altera DE3 EP3SL150

4.2 實驗數據的呈現與討論

此節討論數據方面，將分為兩個部分，首先會討論複雜度的議題， 接著會討論分群後結果的分析。

4.2.1 實驗的複雜度

首先評估本論文所提出的電路所需的空間複雜度和時間上的延遲,由於加法器、乘法器、除法器、指數運算單元以及暫存器為電路的基本組成元件，所以空間複雜度的部份分成五個類別來評估，分別為: 加法器總數、乘法器總數、除法器總數、指數運算單元總數以及暫存器總數，而時間延遲在此定義的是 KFCM 做分群的時間。表 4.5 為本硬體架構所需之空間複雜度以及時間延遲的整理。由表 4.5 可以觀察到，空間複雜度隨著分群數 *c* 增加而線性成長。然而，由於使用了管線化的架構，分群的群數並不會直接影響延遲時間的部分，而只有要計算的資料量 *t* 才是影響延遲時間的主因。

接著，考慮本架構實作方面議題。我們的實驗平台是用 Altera 公司提供的 Quartus II 10.1 和其內部的 SOPC 建立器以及該公司提供的 NIOS II IDE，表 4. 4 所示為在不同的分群數 *c* 的情況下，KFCM-SC 所需的資源消耗。其中包含了三種類別的資源: Adaptive Look-Up Tables（ALUTs)，區塊記憶體位元數(Block Memory Bits)和 DSP 區塊(DSP Blocks)，在編譯時所選的裝置為: Altera Stratix

III EP3SE110F [16] ，其詳細規格如表 4.3。如同表 4.3 所示，隨著分群數 c 的增加，資源消耗也隨之成長。

表 4.3 EP3SE110F FPGA 詳細規格

Device feature	Number
Combinational ALUTs	85200
Memory ALUTs	42600
Dedicated logic registers	85200
Total pins	744
Total block memory bits	8248320
DSP block 18-bit elements	896
Total PLLs	8
Total DLLs	4

表 4.4 KFCM-SC 硬體架構在 EP3SE110F 所需的資源消耗

	ALUTs	DSP Blocks	Block Memory Bits
c = 2	13904/85200 (16%)	218/896 (24%)	27648/8248320 (<1%)
c = 3	20739/85200 (24%)	326/896 (36%)	41472/8248320 (<1%)
c = 4	27245/85200 (32%)	435/896 (49%)	61028/8248320 (<1%)

表 4.5 KFCM-SC 硬體架構所需複雜度

	Adders	Multipliers	Dividers	Exponent Operators	Registers	Latency
Pre-Computation Unit	$O(c)$	$O(c)$	*0*	$O(c)$	$O(c)$	
Membership Coefficients Updating Unit	$O(c)$	$O(c)$	$O(c)$	*0*	$O(c)$	
Center Updating Unit	$O(c)$	$O(c)$	$O(c)$	*0*	$O(c)$	
Total	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(t)$

本論文為了要有更好的分群結果，這裡提出的架構是基於浮點數運算的架構，所有的資料在電路裡面都是用 32 位元的單精密度表示法儲存。也因如此，跟其他使用定點數表示法的架構比起來，本論文提出的架構所算出來的值，會非常相似軟體模擬出來的值，但也會有更多的資源消耗。

表 4.6 即是在分群數 c=2 的情況下，不同架構的資源比較，儘管 KFCM-SC 的架構有較高的資源消耗，但這會使它擁有更好的分群結果，因為浮點數的表示可以讓計算中的值克服因位元數不夠而造成的 overflow，underflow 和 truncation 的問題。

表 4.6: 在 c=2 時，不同架構所需的資源消耗

Architectures	Data Format	FPGA Device	ALUTs	DSP Blocks	Block Memory Bits
FCM[11]	8-bit Fixed Point	Stratix IIEP2S60	3468	20	38944
FCM-SC[12]	8-bit Fixed Point	Stratix II EP2S60	4152	20	38944
KFCM[17]	8-bit Fixed Point	Stratix III EP3SE110F	9453	98	38912
KFCM-SC	32-bit Floating Point	Stratix III EP3SE110F	13904	218	27648

表 4.7 顯示的是在不同的分群數 c=2, c=3 分別在測試資料點總數 t=102400 時的時間複雜度比較，可以發現 KFCM-SC 在速度方面都比軟體還快很多，加速可以到達 13 倍以及 43 倍。

而接著在表 4.8 以及圖 4.2 可以看到隨著要分群的測試資料的增加，硬體計算時間與軟體計算時間的差異會越來越大，更可以突顯出本論文提出的硬體架構的加速效果。這其實是因為這裡的硬體架構在計算資料時，都是利用硬體中管線化的機制提高產能，對於每一個資料x_k在計算時，其下一個資料x_{k+1}也同時進入硬體架構中做計算，不需要等x_k算完才做計算，這點使地當我們隨著資料量增大時，節省下來的計算時間也隨之越多，也就是硬體電路中的單次迴圈計算時間會小於甚至遠小於軟體單次迴圈計算時間。此外，當分群的群數增加時，可以注意

到，硬體計算時間並沒有明顯地增加，而軟體計算時間卻增為原先的 3.27 倍，因此可以合理的預測當分群的群數增加時，速度上能比軟體有更好的表現。

表 4.7 資料量 *t* 為 102400 時，2 群和 3 群的軟硬體速度比較

Cluster numbers	One loop period for KFCM-SC Software	One loop period for KFCM-SC	Speed Up
***c* = 2**	434.9 ms	32.96 ms	13.2
***c* = 3**	1424 ms	32.96 ms	43.2

表 4.8 三群(*c*=3)時，不同的資料量大小所需的軟硬體時間

資料量	51200	102400	153600	204800
軟體	715 ms	1424 ms	2136 ms	2869 ms
KFCM-SC 硬體架構	16.48 ms	32.96 ms	49.44 ms	65.92 ms

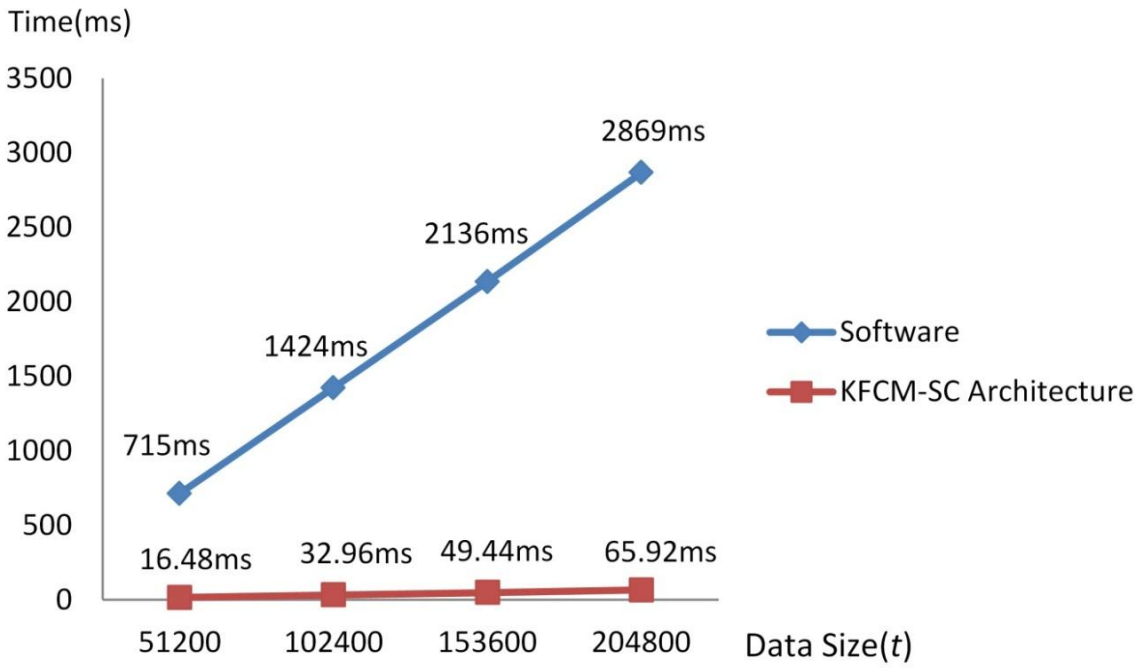
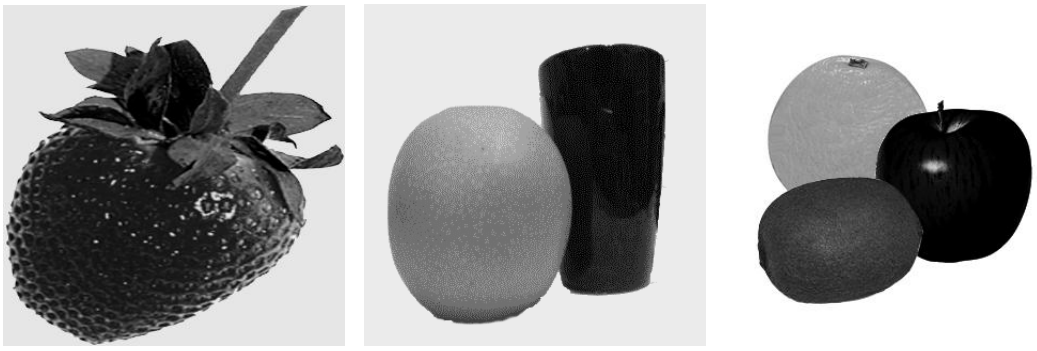


圖 4.2 三群下(*c*=3)軟、硬體時間的比較

4.2.2 實驗的分群結果

本論文將會針對有雜訊的圖片在經過所提出的 KFCM-SC 分割後的結果作探討，而實驗所使用的圖片為灰階影像，大小為 320 x 320 的灰階點陣圖。其中的每個資料點x_k都代表著一個灰階影像區塊，共有 102400 個資料點x_k，值分別都介在 0 到 255 之間。在實驗參數的設定方面，模糊係數 m=2.0、空間資訊的比重值 α=2.0、高斯函數$e^{(\frac{-\|x-y\|^2}{2\sigma^2})}$中的分母$\sigma$=128；此外，本實驗使用的是 uniform distribution noise 雜訊干擾影像，範圍介於$\pm b$之間，其平均值為 0，是一種 additive 的雜訊。首先定義平均雜訊能量 (average noise energy)： $\frac{1}{t}\sum_{i=1}^{t} w_i^2$ ，表示雜訊的強度，t 為雜訊的個數，w_i為隨機產生的雜訊值，表 4.7 為雜訊能量的強度表。



(a) (b) (c)

圖 4.3 測試資料圖檔

(a)草莓圖 (b)梨子和杯子圖 (c)三個水果圖

表 4.9 不同的雜訊能量強度表

b value	10	20	40	60	80
Avg. noise energy	33.33	133.33	533.33	1200	2133.33

為了驗證演算法的正確性，我們在每次分群完畢後，會計算分群錯誤率，作為比較其分群效果的依據。在此定義錯誤率的計算方式: 計算出新的質心後，將每筆輸入的資料點與新的質心的距離作計算，距離最近的質心即為該筆資料點所歸屬的群。令 t 為全部資料點的總數，e 為錯誤之分群個數，定義錯誤率為:

$$Error\ Rate = \frac{e}{t}$$

使用 FCM 以及 FCM_SC 演算法時，我們使用平方距離作為每筆資料點與質心的分群依據；而在使用 KFCM 以及 KFCM-SC 演算法時，則是使用核距離作為分群的依據。

首先，我們先對圖 4.3(a)的草莓圖作分群，這是一張 2 群(c=2)的圖，其紋理明顯且有枝葉向上突出，對於一般人眼判斷不會有太大的影響，可是在電腦的灰階數值表達上，卻不能以單純的 Thresholding 來分辨出前景以及背景 2 群。而且我們的測試圖片還加入雜訊，這也會使分群的難度增加。表 4.10 及圖 4.4 即是對草莓圖的分割錯誤率在不同 b 值情況下，使用硬體架構分群後結果的比較。

表 4.10 不同 FCM 演算法架構對草莓圖的分割錯誤率比較

Architectures	10	20	40	60	80
FCM [11]	2.39%	2.54%	3.29%	4.95%	6.61%
FCM-SC [12]	2.12%	2.16%	2.27%	2.58%	3.04%
KFCM [17]	2.31%	2.42%	2.77%	3.48%	4.95%
KFCM-SC	1.28%	1.45%	1.56%	1.80%	2.08%

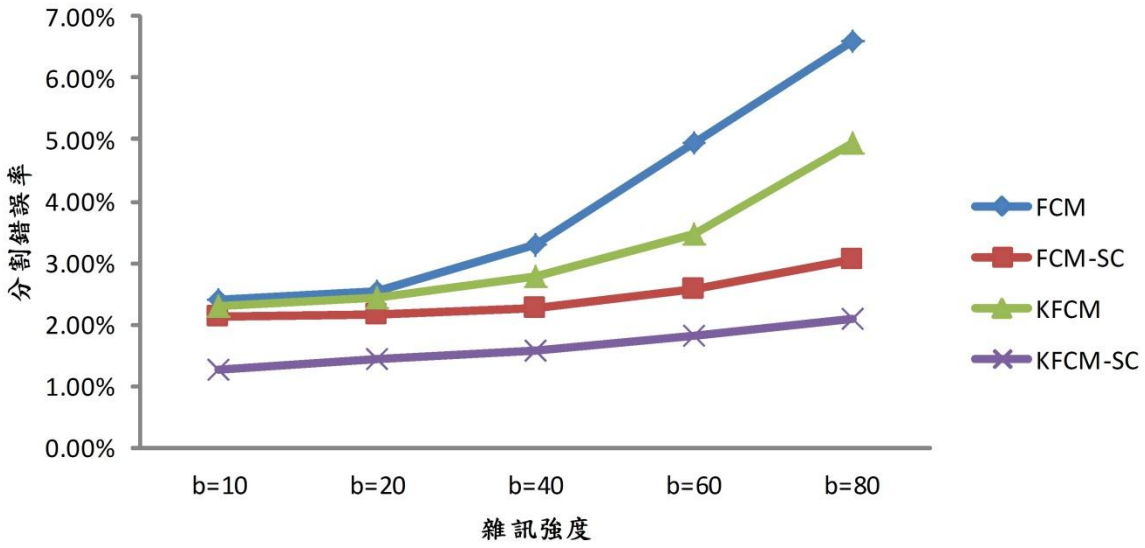


圖 4.4 不同 FCM 演算法架構對草莓圖的分割錯誤率比較

從表 4.10 中觀察到，不論雜訊的強度為何，KFCM-SC 硬體架構皆勝於其他現有的硬體架構，和基本的 FCM 以及 KFCM 有較大的差距，和現有最佳的 FCM-SC[11] 也略勝一個百分點左右。在圖 4.5 及圖 4.6 將會具體呈現經過分群後草莓圖還原後的結果，並會和其他三者做比較還原後的結果作比較。

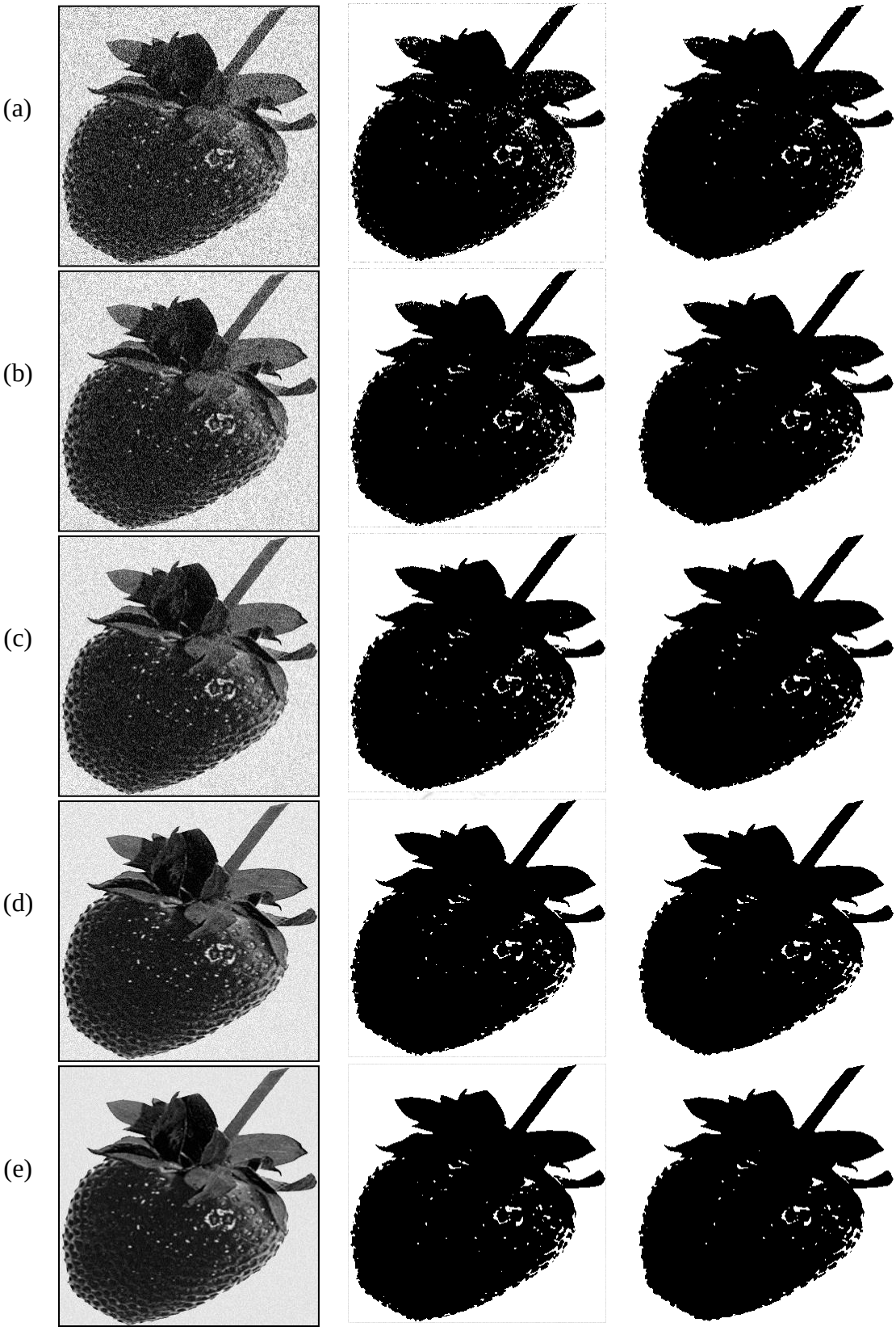


圖 4.5 草莓圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 *b*=80，*b*=60，*b*=40，*b*=20，*b*=10，左邊為加入雜訊後的原始圖片，中間為 FCM-SC 分群結果，右邊為 KFCM-SC 分群結果

(a)

(b)

(c)

(d)

(e)

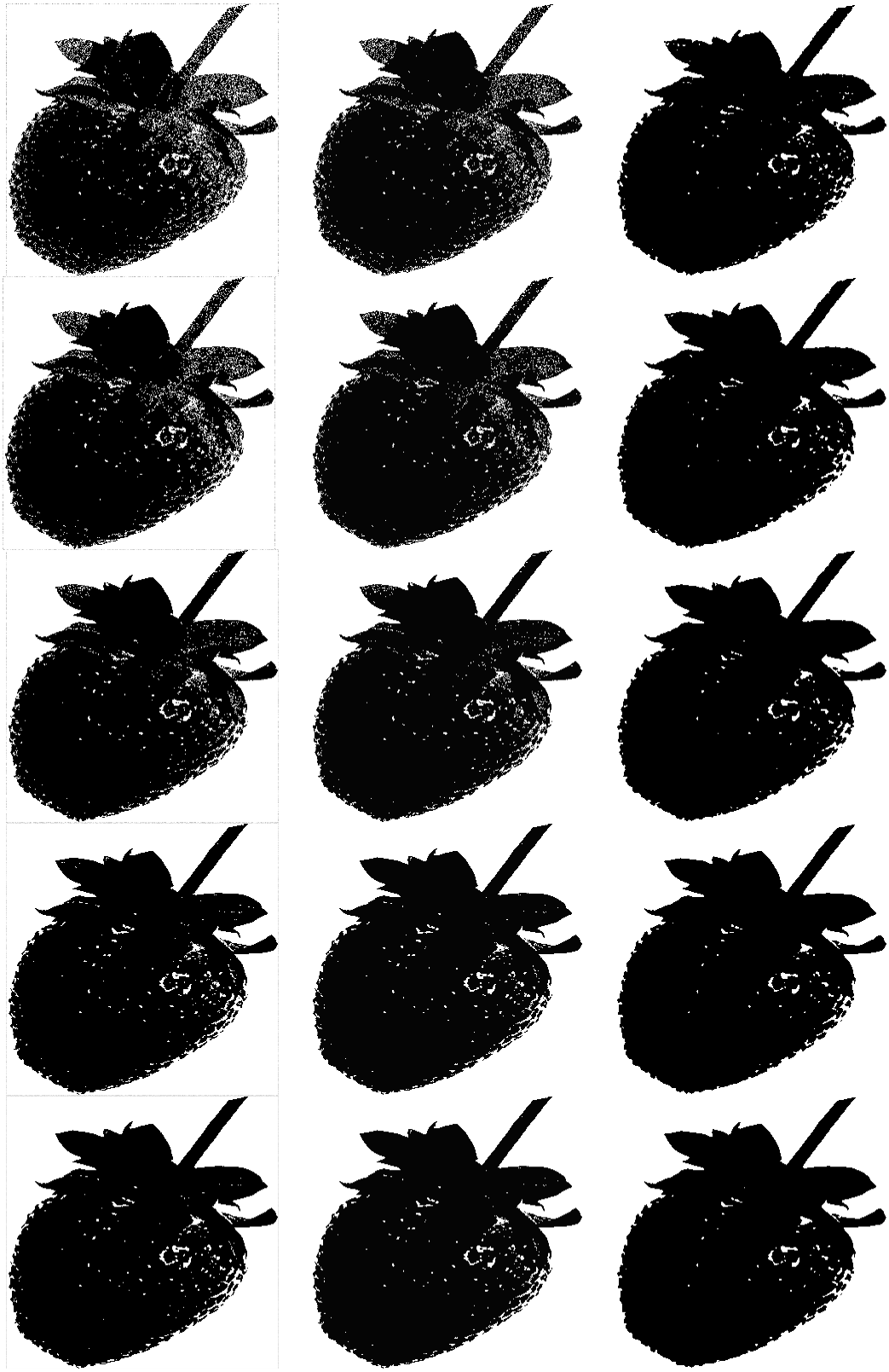


圖 4.6 草莓圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 *b*=80，*b*=60，*b*=40，*b*=20，*b*=10，左邊為 FCM 分群結果，中間為 KFCM 分群結果，右邊為 KFCM-SC 分群結果

接下來，將對圖 4.3(b)的梨子和杯子圖作分析，肉眼可以很明顯的分辨出這是一張前景為梨子和杯子，再加上背景總共三群(*c*=3)的圖片，而梨子和杯子也有其自然存在的紋理，表 4.11 及圖 4.7 即是此圖加入雜訊後分割的效果。

表 4.11 不同 FCM 演算法架構對梨子杯子圖的分割錯誤率比較

Architectures	*b*=10	*b*=20	*b*=40	*b*=60	*b*=80
Basic FCM [11]	2.86%	3.12%	5.97%	16.01%	26.25%
FCM-SC [12]	2.25%	2.37%	3.27%	3.90%	5.37%
KFCM [17]	0.65%	2.16%	5.09%	13.2%	19.99%
KFCM-SC	1.36%	1.33%	1.49%	2.52%	4.84%

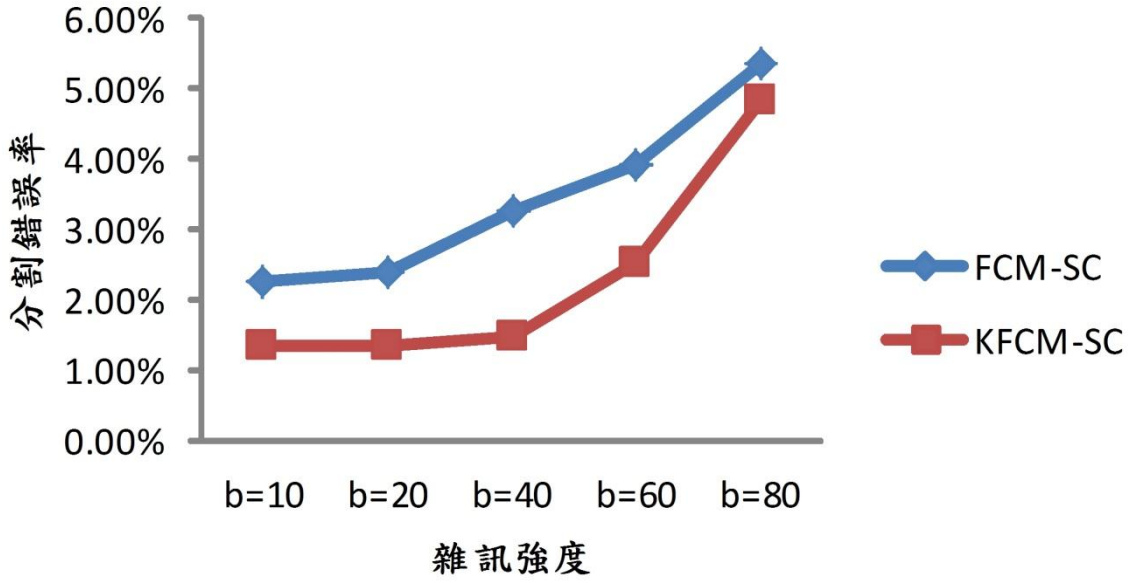


圖 4.7 不同 FCM 演算法架構對梨子杯子圖的分割錯誤率比較

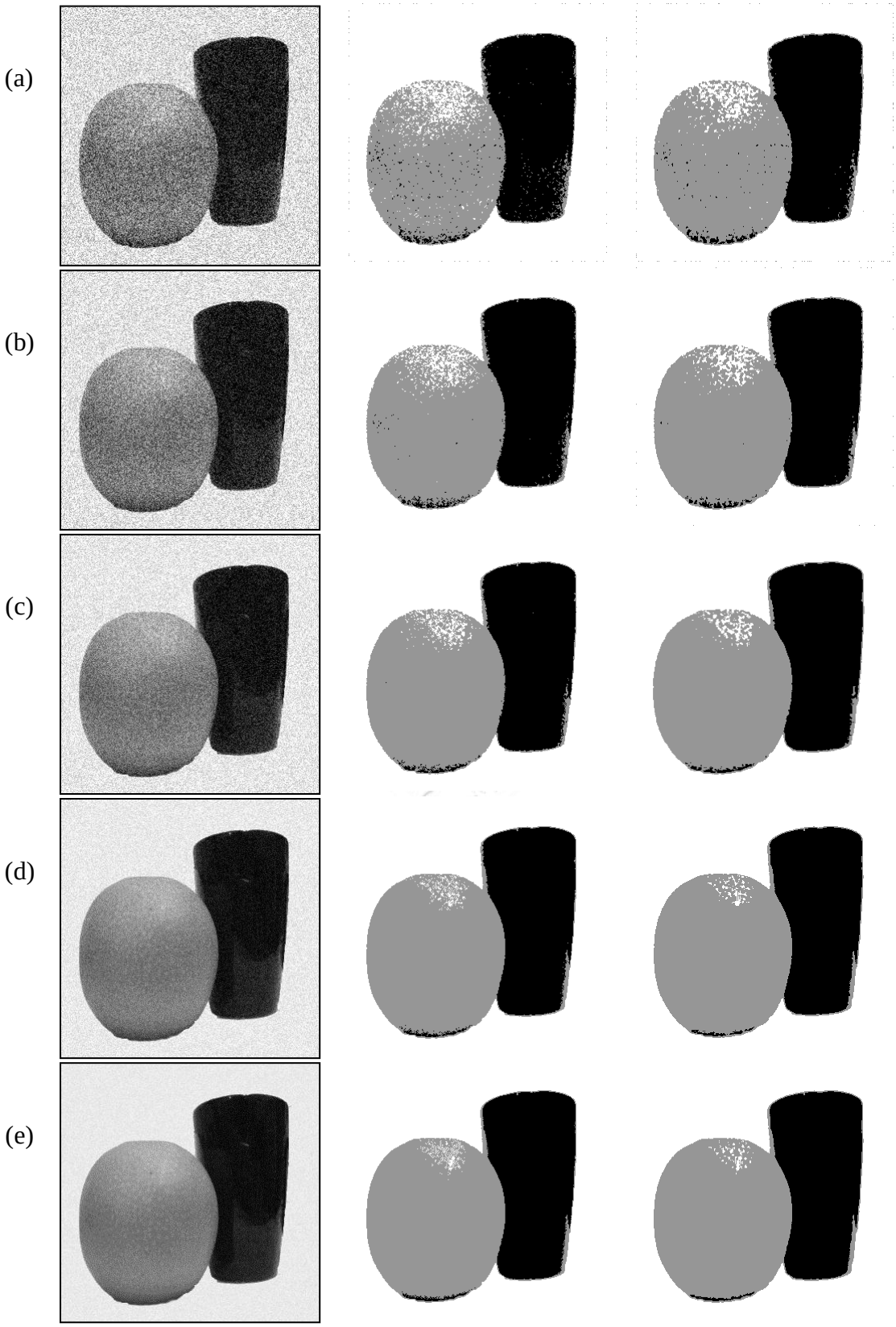


圖 4.8 梨子杯子圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 *b*=80，*b*=60，*b*=40，*b*=20，*b*=10，左邊為加入雜訊後的原始圖片，中間為 FCM-SC 分群結果，右邊為 KFCM-SC 分群結果

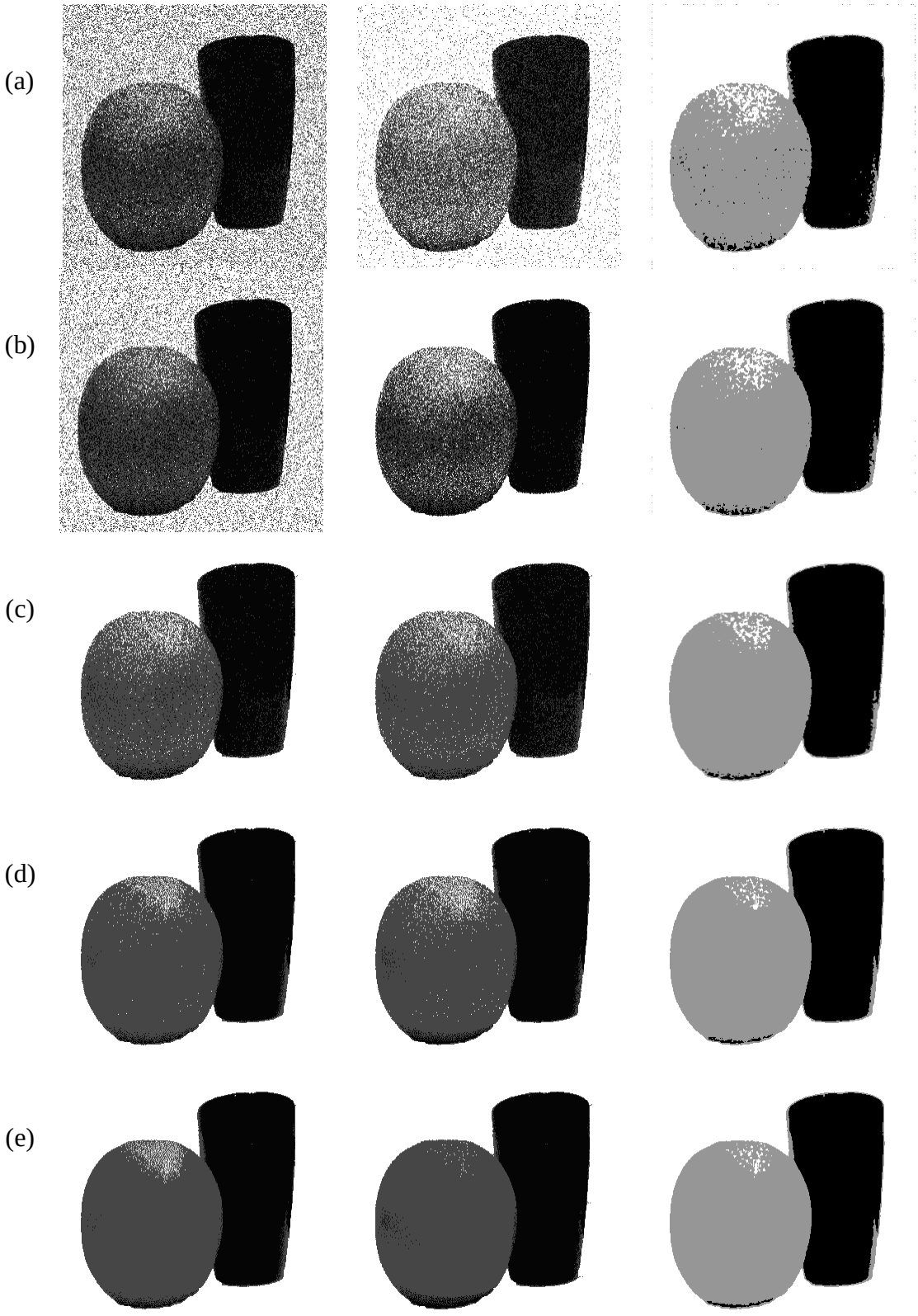


圖 4.9 梨子杯子圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 *b*=80，*b*=60，*b*=40，*b*=20，*b*=10，左邊為 FCM 分群結果，中間為 KFCM 分群結果，右邊為 KFCM-SC 分群結果

從表 4.11 以及圖 4.7 中觀察到，不論雜訊的強度為何，在對三群的圖片做分割時，KFCM-SC 硬體架構亦皆勝於其他現有的硬體架構，且和二群的圖片比較起來，和基本的 FCM 以及 KFCM 有著更明顯的差距，但和現有最佳的 FCM-SC[11] 大致上仍可略勝一個百分點左右。圖 4.8 和圖 4.9 為具體呈現經過分群後梨子杯子圖還原後的結果，之中也和 FCM-SC[11]還原後的結果作比較。

接著將對四群的圖片作分析，這是一張前景有三個水果的圖片，大概看得出來分別為檸檬、蘋果以及橘子，而每個水果其也都有自然紋理在上面，此外右邊的蘋果上面有一小圈的光澤再加上此圖在雜訊的影響下，尤其是當雜訊值 b 為 80 的時候，整張圖片的背景與後面的橘子上面的紋理已經有很大的重疊，蘋果和檸檬上面的紋理也因此變得很接近，雖然肉眼仍可輕易分辨此三個物品，但對電腦的灰階圖來說，這將會使分割此圖變得更為困難，表 4.12 以及圖 4.10 為此圖加入雜訊後的還原效果。

表 4.12 不同 FCM 演算法架構對圖 4.4(c)三個水果圖的分割錯誤率比較

Architectures	*b*=10	*b*=20	*b*=40	*b*=60	*b*=80
Basic FCM [11]	1.21%	2.19%	6.51%	22.46%	32.53%
KFCM [17]	1.24%	2.33%	6.37%	19.53%	31.58%
KFCM-SC	0.62%	1.24%	4.33%	6.46%	18.52%

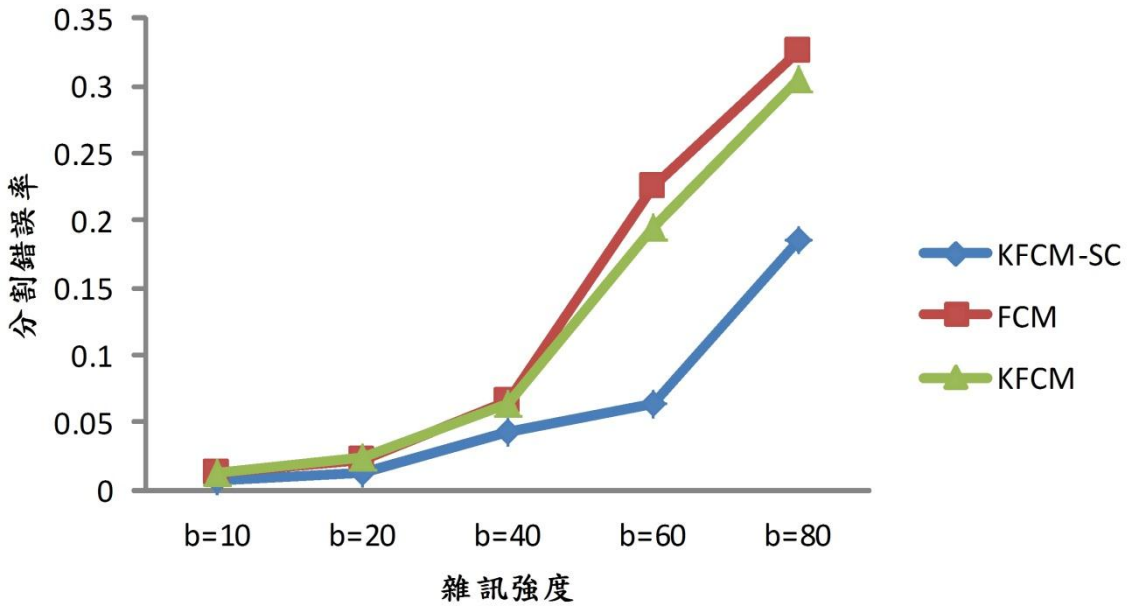


圖 4.10 不同 FCM 演算法架構對三個水果圖的分割錯誤率比較

由於論文[12]中並沒有提到對此圖的錯誤率分析，因此這裡僅引用在論文[17] 中提到的數據做參考的對象，可以從此結果中發現 KFCM-SC 在四群的分割錯誤率明顯大大地優於傳統的 FCM 以及 KFCM 法則，圖 4.11 將呈現此圖還原後的效果。

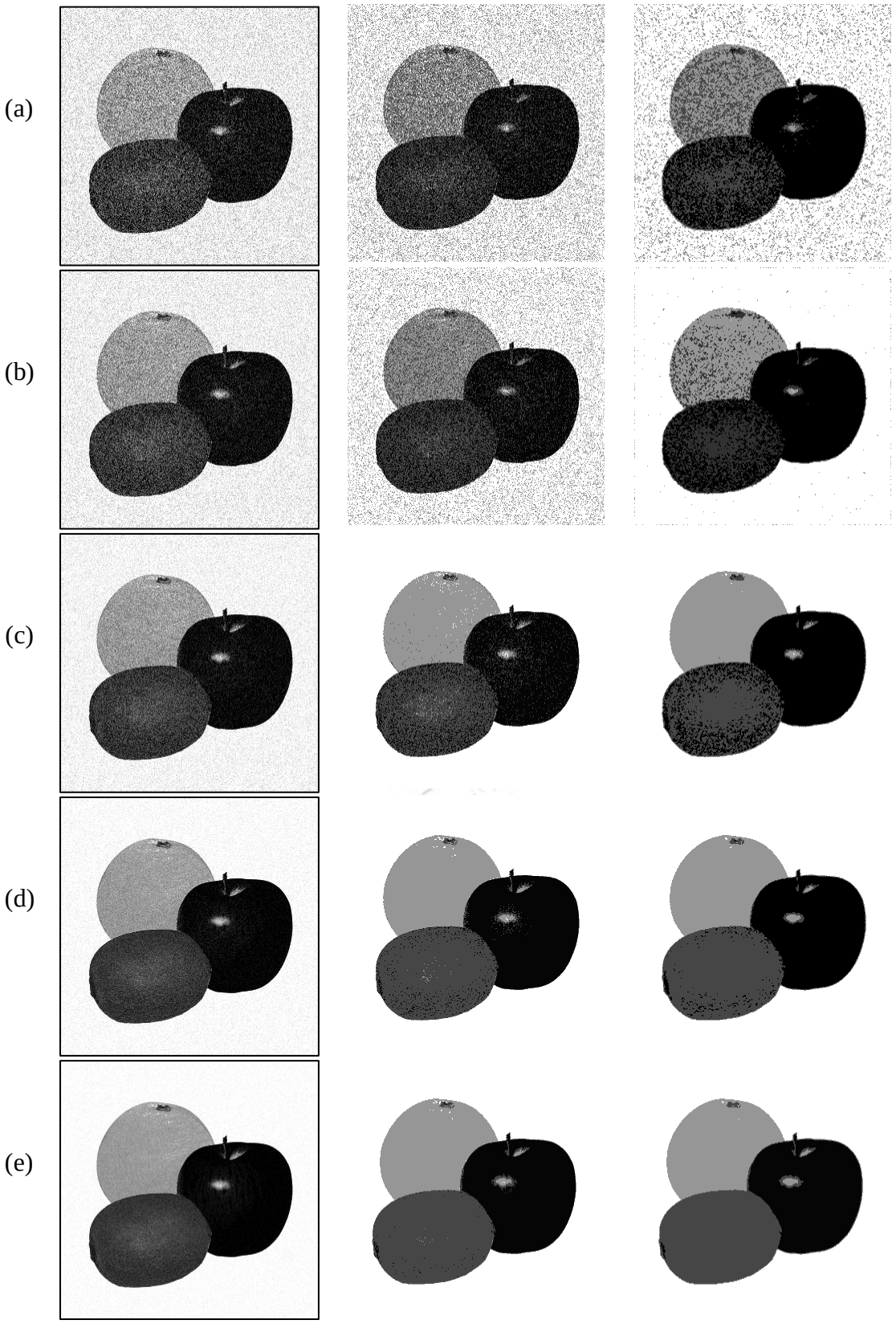


圖 4.11 三個水果圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 *b*=80，*b*=60，*b*=40，*b*=20，*b*=10，左邊為加入雜訊後的原始圖片，中間為 KFCM 分群結果，右邊為 KFCM-SC 分群結果

從三個水果圖中可以看到，儘管在雜訊為 b=80 之下，KFCM-SC 對於右邊那顆蘋果來說，除了上面原有的光澤難以還原外，幾乎是還原得很清楚，擁有很不錯的分群能力。

可以從上述的例子中觀察到，本論文提出的 KFCM-SC 硬體架構相較於其他現有的 FCM 硬體架構擁有比較低的分割錯誤率，KFCM-SC 擁有較優秀的分割表現，也是基於此演算法在分群過程中，同時使用了核函式以及空間資訊，而且，這裡提出的 KFCM-SC 硬體架構是基於浮點數的運算而非定點數，這點也使得此架構有更好的效果。此外也可以推測，當實驗的分群數(c)增加時，KFCM-SC 相較於其他 FCM 演算法架構的優勢將會更明顯。

接著要來討論的是在分群過程中，不同的 α值在雜訊 b 值不同時的狀況差異。表 4.13 以及圖 4.12 顯示的是當圖 4.3(b)在不同的α值時，對分割的結果所造成的影響。藉由此表 4.13 以及圖 4.12 可以發現，當雜訊增加時，α值越大，分割錯誤率會相較於α值小的還要來的低，這其實是因為α值代表的是 KFCM-SC 公式中空間資訊的比重，當此比重越大時，對於每一個資料x_k來說，在計算過程中以該資料x_k本身為中心加上周遭環境的資料後平均值$\bar{x}_k$，的影響力，會大於x_k本身在計算過程中所佔的影響。也因此可以推論，配合適當的α之下，此$\bar{x}_k$可以更有效地提高 KFCM-SC 分割過程中對於雜訊處理的抵抗能力。

表 4.13 不同α值對梨子和杯子圖在不同雜訊下的分割錯誤率比較

	b = 10	b = 20	b = 40	b = 60	b = 80
α = 2	1.36%	1.33%	1.49%	2.52%	4.84%
α = 128	1.48%	1.43%	1.5%	2.42%	3.16%

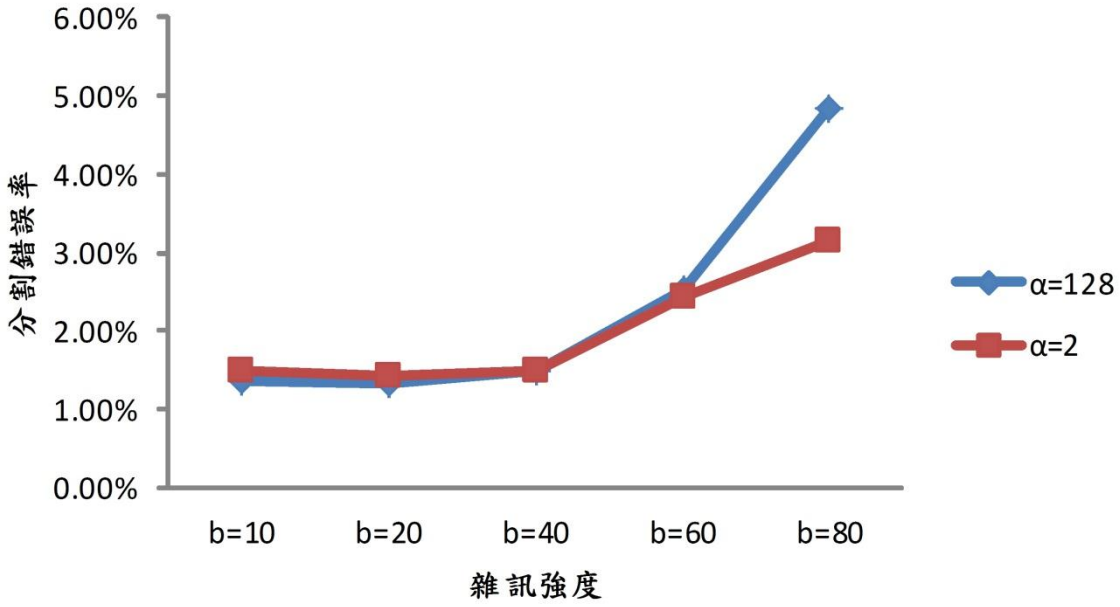


圖 4.12 不同的α值對梨子和杯子圖在不同雜訊下的分割錯誤率比較

圖 4.13 為梨子杯子圖在不同α值分群後的還原圖，可以觀察到在雜訊 b 值為 10、20、40 以及 60 時，兩者無論從圖表上的分析或者肉眼觀看還原後的圖片，皆較難看到有太大的差異性，唯獨在雜訊 b 值為 80 的情況下，兩者的差距也就慢慢地更突顯了出來。

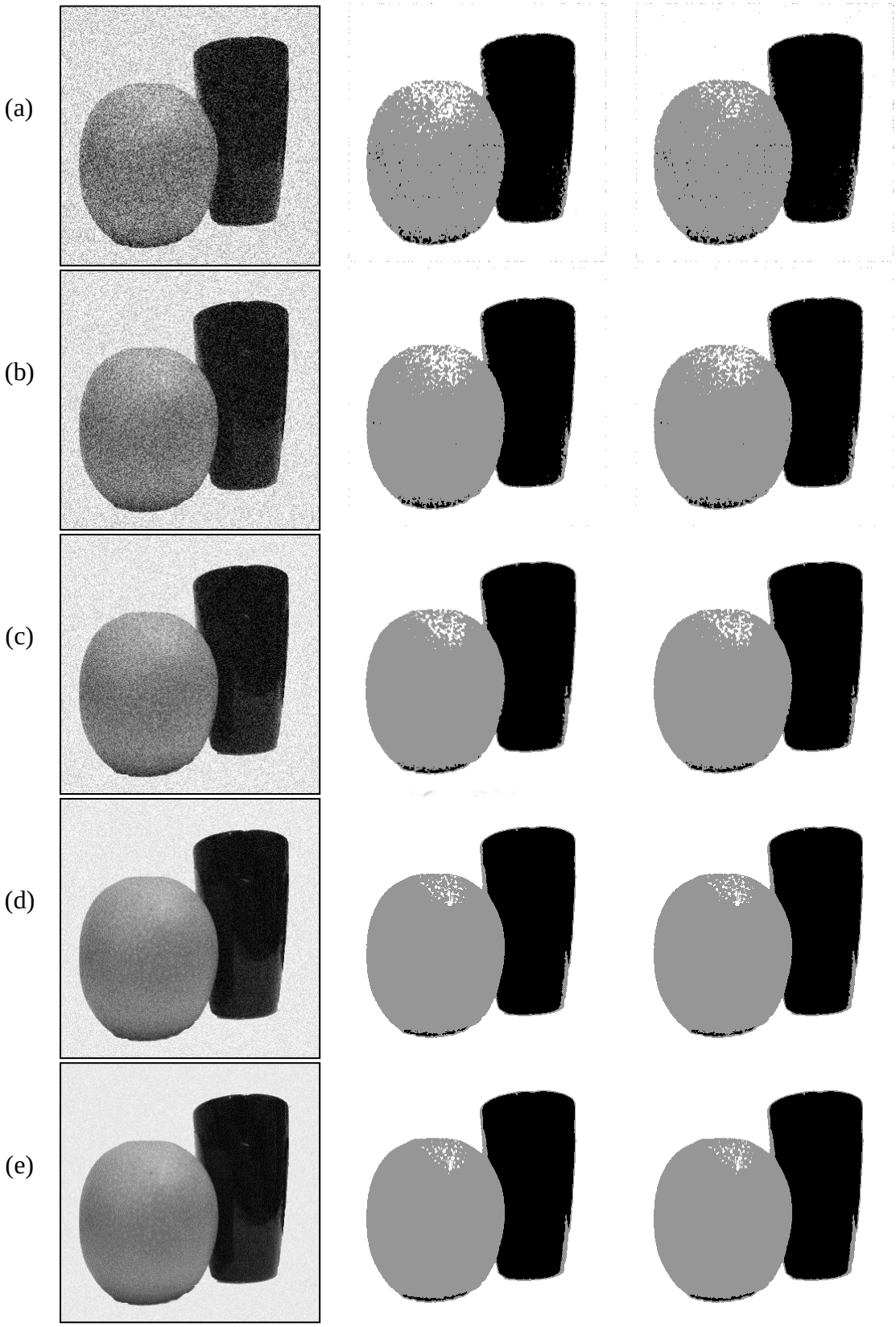


圖 4.13 三個水果圖使用 KFCM-SC 演算法硬體電路之分群結果顯示

由上而下為雜訊值 b=80，b=60，b=40，b=20，b=10，左邊為加入雜訊後的原始圖片，中間為α=2 分群結果，右邊為α=128 分群結果

第五章 結論

本論文呈現 KFCM-SC 之硬體架構，並實際測試此架構對於有雜訊的影像做分割的結果，相較於傳統的 FCM 分群演算法，在加入了核函式(Kernel)的高斯函數計算以及空間資訊(Spatial Constraint)後，分群錯誤率有顯著的降低，此外使用了浮點數做值的儲存，大大地增加了和軟體模擬出來結果的準確度，也使分群的結果更好。人們在還原後的圖片上用肉眼就可以輕易地辨別出圖片的分割結果和現有 FCM 相關分群演算法所算出的結果的差異。

此外，這裡呈現的 KFCM-SC 硬體架構，使用了管線化(Pipeline)的架構去提高產能的輸出，充分地運用了每個硬體元件，使得在計算過程當中，每個硬體元件都不會 idle，這點也使得和軟體比較起來，本論文提供的硬體架構在計算分群時能夠比軟體加速很多。

綜觀以上兩段敘述，本論文提出的硬體架構可以實際應用於做即時影像分割的行動裝置上。

參考著作

[1] J. C. Bezdek, *Fuzzy Mathematics in Pattern Classification*, Cornell University: Ithaca, NY, USA, 1973.

[2] S. Chen, D. Zhang, Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure, *IEEE Systems, Man, and Cybernetics*, pp.1907-1919, 2004.

[3] W. Cai, S. Chen, D. Zhang, Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation, Pattern Recognition, Vol. 40, pp. 825-838, 2007.

[4] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, A survey of kernel and spectral methods for clustering, *Pattern Recognition*, Vol. 41, pp. 176-190, 2008.

[5] K. S. Chuang, H. L. Tzeng, S. Chen, J. Wu, T. J. Chen, Fuzzy c-means clustering with spatial information for image segmentation, *Comput. Med. Imaging Graphics*, Vol. 30, pp.9-15, 2006.

[6] J. F. Kolen and T. Hutcheson, Reducing the Time Complexity of the Fuzzy C-Means Algorithm, *IEEE Trans. Fuzzy Systems*, Vol. 10, pp. 263-267, 2002.

[7] S. Eschrich, J. Ke, L. O. Hall, and D. B. Goldgof, Fast Accurate Fuzzy Clustering Through Data Reduction, *IEEE Trans. Fuzzy Systems*, pp.262-270, 2003.

[8] T. W. Cheng, D. B. Goldgof, L. O. Hall, Fast fuzzy clustering. *Fuzzy Sets Syst.*, Vol. 93, pp. 49-56, 1998.

[9] J. Garcia-Lamont, L.M. Flores-Nava, F. Gomez-Castaneda, J.A. Moreno-Cadenas, CMOS Analog Circuit for Fuzzy C-Means Clustering, *IEEE Proc. 5th Biannual World Automation Congress*, 2002.

[10] J. Lazaro, J. Arias, J. L. Martin, C. Cuadrado and A. Astarloa, Implementation of a Modified Fuzzy C-Means Clustering Algorithm for Realtime Applications, *Microprocessors and Microsystems*, pp. 375-380, 2005.

[11] H. Y. Li, C. T. Yang, and W. J. Hwang, Efficient VLSI Architecture for Fuzzy C-Means Clustering

in Reconfigurable Hardware, Proc. IEEE International conference on Frontier of Computer Science and Technology, p.168-174, 2009.

[12] H. Y. Li, W.J. Hwang and C.Y. Chang, Efficient Fuzzy C-Means Architecture for Image Segmentation, Sensors, Vol. 11, pp. 6697-6718, 2011.

[13] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann, USA, 2008.

[14] Altera Corporation, *NIOS II Processor Reference Handbook*, http://www.altera.com/ literature/ lit-nio2.jsp, 2012.

[15] Altera Corporation, Floating Point Exponent (ALTFP_EXP) Megafunction User Guide, 2008.

[16] Altera Corporation, *Stratix III Device Handbook*, http://www.altera.com/ literature/ lit-stx3.jsp, 2012

[17] 楊斯閔, ”Kernel-Based Fuzzy *c*-Means 分群演算法硬體架構實現”, 碩士論文, 國立臺灣師範大學資訊工程學系, 民國壹佰年