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　教授

以 FPGA 實現基於部分距離搜尋法之競爭式學習系統

FPGA Implementation of Competitive Learning with Partial Distance Search

研究生：　李　惠　雅　撰

中華民國　九十七　年　七　月

中文摘要

本論文針對 k 贏家通吃競爭式學習法之場域可程式化閘陣列(FPGA)實作提出一新演算法。k 個得以進行更新的獲勝神經元，為每一個輸入向量在小波域(wavelet domain)中執行部分距離搜尋(partial distance search)所找出的最近似者。在大多數的應用裡，PDS 以軟體方式被用於神經元搜尋的加速。此章節將提出一個適於硬體實現的新 PDS 演算法。此演算法使用子空間搜尋(subspace search)、有限精度計算(finite precision calculation)、多係數累積(multiple-coefficient accumulation)、和查表式除法(lookup-table based division)等技巧來有效降低面積複雜度與運算延遲。也提出一個新的排序架構，用於 PDS 步驟後 k 個獲勝神經元的判定。

在此提出的硬體架構將以專用邏輯區塊電路(custom logic block)的方式內嵌於 Nios 軟核心中央處理器的算術邏輯單元(ALU)中。Nios 處理器所提供的客製指令(custom instruction)便是用於存取專用邏輯區塊電路的方式。我們已測量出，Nios 軟核心中央處理器執行用於「k 贏家通吃競爭式學習訓練」之部分距離搜尋程式客制指令所需的 CPU 時間。實驗結果顯示 CPU 時間低於未搭配部份距離搜尋硬體電路的 Pentium IV 處理器。

Abstract

This paper presents a novel algorithm for the field programmable gate array (FPGA) realization of the competitive learning (CL) algorithm with *k*-winners-take-all activation. The *k* winning neurons for updating are those best matching the input vector in the wavelet domain with partial distance search (PDS). In most applications, the PDS is adopted as a software approach for attaining moderate codeword search acceleration. In this chapter, a novel PDS algorithm well-suited for hardware realization is proposed. The algorithm employs subspace search, finite precision calculation, multiple-coefficient accumulation, and lookup-table based division techniques for the effective reduction of the area complexity and computation latency. A novel sorting architecture is also proposed for identifying the *k* winning neurons after the PDS process.

The proposed implementation has been adopted as a custom logic block in the arithmetic logic unit (ALU) of the softcore NIOS processor. The custom instructions are also derived for accessing the custom logic block. The CPU time of the NIOS processor executing the PDS program with the custom instructions for *k*-winners-take-all CL training is

measured. Experiment results show that the CPU time is lower than that of Pentium IV processors executing the PDS programs without the support of custom hardware.

誌謝

首先要感謝我的指導教授黃文吉教授，在過去一年的時光裡，給予我課業與研究上的細心指導與關懷，使得本論文得以順利完成，在此獻上最誠摯的謝意。同時也感謝國立中央大學通訊所張寶基教授、國立台北科技大學資訊工程所楊士萱教授、中央研究院資科所陳伶志助研究員撥冗在論文上所給予的評閱與建議。

此外，要感謝校內外各位學長、同學——御廷、汶錫、定寬、祐琦、正岳、凱富、智傑、韋德、永隆、嘉儀、正存、嘉隆、鵬傑、昇遠等，於平日生活及課業上的協助與切磋，在此一併致謝。

最後，更要感謝家人、男友與所有關心我的朋友，有了你們的支持、關懷與鼓勵，讓我有勇氣去面對各種的挫折與壓力，並順利地完成學業。謹將此論文成果獻給所有關心我的人，希望大家與我分享這份喜悅與榮耀。

目 錄

附圖目錄

附表目錄

第一章 緒論

此章探討本論文的研究背景與動機目的，並概述各章節的主要內容及重要特性。

1.1 研究背景與動機目的

資料分群(Data clustering)對於一些窮究樣式分析(exploratory pattern analysis)、分組(grouping)、決策(decision making)及機器學習(machine learning)的情況，包含資料探勘(data mining)、文件檢索(document retrieval)，樣式分類(pattern classification)、影像分割(image segmentation)和資料壓縮(data compression)等，是非常有用的。在類神經網路(artificial neural networks，ANNs)中[10]，分群常以競爭式學習(competitive learning，CL)法則來實現[9, 16]。即網路中的神經元(neurons)藉由彼此競爭而被活化或代謝；而每個神經元所對應的權重向量(weight vector)，等於它在輸入特徵空間(input feature space)內的接受區(receptive field)中心。競爭式學習的目標，是將分群分析/樣式分類的錯誤[13, 21]，或是向量量化(vector quantization，VQ)的量化失真(quantization distortion)[8, 15] 最小化。

已經發展出來的眾多競爭式學習法則，可依其競爭方法及學習規

則來做區別。最簡單的競爭式學習演算法是以贏家通吃(winner-take-all，WTA)法則為基礎[11]，使神經元演化僅發生在最相似於輸入樣式的贏家上。這種簡易贏家通吃式學習的主要缺點為可能存在死點(dead nodes)——亦稱為利用不足(under-utilization)問題。在這類例子裡，一些神經元會因為不正確的初始化而永遠不能成為贏家。所以，它們不但對學習無法給予貢獻，還可能降低競爭式學習的效能。此外，這種競爭式學習演算法的另一個缺點是高計算複雜度。當神經元的數量很大，及/或每個神經元對應之權重向量的維度很高，鑑別出與輸入向量最相似神經元所需的平均 CPU 時間就會變得很長。而冗長的訓練時間對於即時應用(real-time applications)而言，可能就會變成一種侷限了。

解決利用不足問題的一個有效方法，是以 *k* 贏家通吃(*k*-winners-take-all，*k*-WTA)取代 WTA [20]，藉以降低對權重向量初始區位(initial location)之依賴度。簡易的贏家通吃競爭式學習法已然需要冗長的訓練時間，然而，由於排序操作在發掘 *k* 個最相似神經元的過程中是必須的；因此就計算成本而言，*k* 贏家通吃競爭式學習法更高過贏家通吃競爭式學習法。如此一來，*k* 贏家通吃法可能依然不適於期望能做到快速訓練的競爭式學習分群。

高計算成本的缺點或可藉由加速搜尋過程的方式來避免，比方說，在原有或轉換後的域上使用部分距離搜尋(PDS) [3, 12]。PDS 演算法因僅計算部分距離，只需少量乘法運算便能淘汰不需要的神經元。當一個神經元的部分距離大於現有最相配神經元的全距離(full distance)時，這個神經元便會被判淘汰。但這些軟體方法僅能達到普通程度的加速效果。因此便有學者選擇使用 VLSI 來實現搜尋引擎[16, 18]，把原有為了 VQ 編碼器所設計的各式心脈陣列(systolic arrays)拿來進行平行搜尋。相較於軟體實現，硬體電路提供了較高的產量(throughput)。然而，這些硬體設計的彈性有限。當套用到競爭式網路，大多數的心脈結構會被最佳化到僅供固定維度的權重向量，及/或固定數目的神經元使用；而且也只能鑑別出最相似的神經元。因此這些結構就只能用在贏家通吃競爭式學習法了。而這些特定應用積體電路(application specific integrated circuit，ASIC)也有著不可重複規劃(reconfigurable)的缺點。不同維度數及/或神經元數的競爭式網路必須有不同的 ASIC 實體，同時也就提高了開發成本。

有鑒於上述情況，本論文提出一個 *k* 贏家通吃競爭式學習法的新硬體架構，簡稱 *k*CL。此架構可以快速完成訓練，並以現場可程式邏輯陣列(field programmable gat array，FPGA)為實現平台[4, 5]，因此可

在重複規劃後套用到不同的競爭式網路上。基於 FPGA 的可重組硬體幾乎可像軟體般來進行程式設計，且相較於傳統的 ASIC 硬體實現，它最吸引人的地方莫過於彈性佳、成本低。另外，FPGA 的硬體實現可對競爭式學習法的演算進行平行處理，從而降低訓練時間。

除了使用 FPGA，我們還在小波域上進行有限精度的子空間部分距離搜尋法[18]，以求達到高產量、高彈性及低面積複雜度等特性。由於小波轉換具正交性，且轉換後可將大部分能量集中在低頻的幾個係數，所以在小波域上執行部分距離搜尋的效能遠優於在原域上進行。在本論文提出的硬體架構中，對所有輸入的測試樣本都會進行 Haar 小波轉換。在所有的小波轉換當中，Haar 小波是具正交性且最簡單的一種方法，擁有低計算複雜度的特性，適合用硬體來實現。由於小波高頻的係數僅保留了輸入測試樣本的少許能量，所以在進行部分距離搜尋運算前會捨去這些高頻的係數。不僅如此，低頻係數最小顯著位元 (Least significant bits，LSBs) 的部分位元平面也會被捨去，而這個動作只會對部分距離搜尋的結果造成極小的影響。

根據 PDS 的結果，我們提出一種新式排序電路，供 *k*-WTA 中找尋 *k* 個最佳神經元的操作使用。許多現有的排序架構[6, 7]，僅只是單純地在硬體上實現軟體排序演算法；所以當 *k* 值很大時，這些架構的

計算及/或面積複雜度可能會變得很高。而我們的電路兼具了低計算時間及低面積複雜度的特性。事實上，我們的電路包含了 k 個排序模組，其中第 i 個模組的責任，就是負責找到對於輸入樣式的第 i 個最佳神經元，故而面積複雜度僅隨 k 值線性增加。每一個排序模組只包含一個 buffer、一個 comparator 和一個 multiplexer。由於所有模組同時運作，因此排序操作中不存在傳遞延遲(propagation delay)，計算時間也與 k 值大小無關。故結合子空間搜尋和本論文所提出的排序電路，搜尋速度便可有效提升，並仍然保有低面積成本的優點。

最後，因為除法運算在更新神經元時無法避免的，因此為了降低更新神經元時的計算延遲，我們事先將除法運算的所有可能結果存於一查詢表(lookup table)中，如此一來，即可利用查表法得知除法運算的結果，也就技巧性地避開耗時的除法運算。

本論文所提出的硬體架構將以專用邏輯區塊電路 (custom logic block) 的方式內嵌於 Nios 軟核心中央處理器的算術邏輯單元 (ALU) 中[2]，然後整合入可程式化系統晶片 (System on Programmable Chip，SOPC) 平台來完成硬體實現與測試。Nios 處理器所提供的客製指令 (custom instruction) 便是軟體端用來存取專用邏輯區塊電路的方式。我們已成功的實現並測量出 Nios 軟核心中央處理器執行部分距離搜

尋專用硬體電路所需要的平均時間，同時在實驗數據中顯示出 50MHz 的 Nios 處理器搭配部份距離搜尋硬體電路的效能優於未搭配部份距離搜尋硬體電路的 Pentium IV 3.0GHz 處理器。

1.2 全文架構

本篇論文總共分為五章，各章的內容安排如下：

【第一章】 緒論

說明本論文的研究背景與動機目的。

【第二章】 基本理論介紹

敘述本論文提出架構的理論基礎與背景知識，包括 k 贏家通吃競爭式學習法、離散小波轉換原理以及部分距離神經元搜尋演算法則。

【第三章】 *k*CL 之架構與硬體實現

詳細說明本論文所提出的 *k*CL 硬體電路，及其內部各單元的架構，包含 DWT 運算單元硬體電路、向量平方距離計算單元硬體電路、排序電路、部分距離搜尋硬體電路。同時解析硬體實現時所採用的子空間搜尋、有限精度計算、多係數部分距離累積，以及查表式除法等四項技巧。

【第四章】 實驗數據與效能比較

將本論文提出的單模組與多模組的硬體架構以專用邏輯區塊電路的方式實現於 Altera FPGA (Stratix EP1S40F780C5) 發展板中，驗證本論文提出架構的效能，並與 Pentium IV 處理器以純軟體實現方式的結果作效能比較分析與探討。

【第五章】 結論

說明本論文的貢獻。

第二章 基礎理論介紹

本章將探討本論文架構的理論基礎與背景知識，包括 k 贏家通吃競爭式學習法、離散小波轉換原理，以及在小波域上進行運算的部分距離神經元搜尋演算法則。

2.1 k 贏家通吃競爭式學習法

競爭式學習指的是一種誤差修正的學習過程。各神經元以其對應權重向量與訓練向量的相似性做競爭；獲勝神經元的對應權重向量，就利用其與訓練向量的差異來做調整。這個學習過程就是在使權重向量慢慢趨近於訓練向量，相當於在學習訓練向量的平均值。

回顧贏家通吃競爭式學習，我們可以把學習過程分為兩個階段：

(1) 競爭階段(competitive phase)：根據訓練向量 x 與權重向量 $\mathbf{y}^j$ 兩者間的最小歐幾里得距離來挑出贏家 $\mathbf{y}^{j*}$。贏家將滿足

$$j^* = \text{argmin}_{1 \le j \le N} D(\mathbf{x}, \mathbf{y}^j) \tag{2.1}$$

(2) 學習階段(learning phase)：調整獲勝神經元的權重向量－－也就是對獲勝神經元的權重向量加上一個學習量。規則如下：

$$\mathbf{y}^{j*} \leftarrow \mathbf{y}^{j*} + \eta^{j*}(\mathbf{x} - \mathbf{y}^{j*}) \tag{2.2}$$

其中 η^{*} 是獲勝神經元的學習率(learning rate)，會隨著時間增加而嚴格遞減。由於學習率是一個嚴格遞減函式，故而神經元會因為學習率的遞減而穩定收斂。

然而，只有一個神經元擁有學習機會是有欠公平的，也很容易有利用不足的問題，因此將贏家通吃法延伸至 k 贏家通吃法，讓最相似訓練向量的 k 個神經元都擁有學習的機會。

由上述公式可看出搜尋時所做的比對是全搜尋的運算，神經元數目及/或維度的增加，將會大幅提升計算及儲存的複雜度，因此如何有效降低這兩樣複雜度是一個重要的課題。

2.2 離散小波轉換

在這一小節我們將簡單的介紹小波轉換。在訊號轉換處理的過程中，訊號被投影到一組基底函數上，相對應於基底函數的係數正是我們要處理的部份。有效的處理方式能將訊號藉由轉換過程使轉換後的訊號能量集中在少數的係數上，而小波轉換正符合此特性，使得我們

可以捨棄不重要的高頻係數資訊而不會嚴重影響分類的正確性。

令 **X** 是維度 $2^n \times 2^n$ 的向量 x 做 n 階離散小波轉換 (n-stage DWT) 之後的結果，如圖 2-1 所示，**X** 亦為 $2^n \times 2^n$ 的區塊且由 $\mathbf{x}_{L0}$ 和 $\mathbf{x}_{Vi}$、$\mathbf{x}_{Hi}$、$\mathbf{x}_{Di}$ 所組成，其中 $i = 0,1,\cdots,n-1$。在離散小波轉換的過程當中，區塊 $\mathbf{x}_{L(m\text{-}1)}$ (低頻區塊，$m = 1,2,\cdots,n$) 以及 $\mathbf{x}_{V(m\text{-}1)}$、$\mathbf{x}_{H(m\text{-}1)}$、$\mathbf{x}_{D(m\text{-}1)}$ (高頻區塊，$m = 1,2,\cdots,n$) 是由 $\mathbf{x}_{Lm}$ 遞迴獲得，而 $\mathbf{x}_{Ln}$ 等於輸入向量 x，$\mathbf{x}_{Lm}$、$\mathbf{x}_{Vm}$、$\mathbf{x}_{Hm}$ 和 $\mathbf{x}_{Dm}$ 這些區塊的維度是 $2^m \times 2^m$。也就是說，這些區塊 $\mathbf{x}_{L0}$ 和 $\mathbf{x}_{Vi}$、$\mathbf{x}_{Hi}$、$\mathbf{x}_{Di}$，其中 $i = 0,1,\cdots,m-1$，是由 $\mathbf{x}_{Lm}$ 經過 m 階的離散小波轉換後得來。而 $\mathbf{x}_{Lm}$ 可經由一個簡單的鏡像濾波器 (quadrature mirror filter，QMF) 分解成 $\mathbf{x}_{L(m\text{-}1)}$、$\mathbf{x}_{V(m\text{-}1)}$、$\mathbf{x}_{H(m\text{-}1)}$和 $\mathbf{x}_{D(m\text{-}1)}$四個區塊[10]。

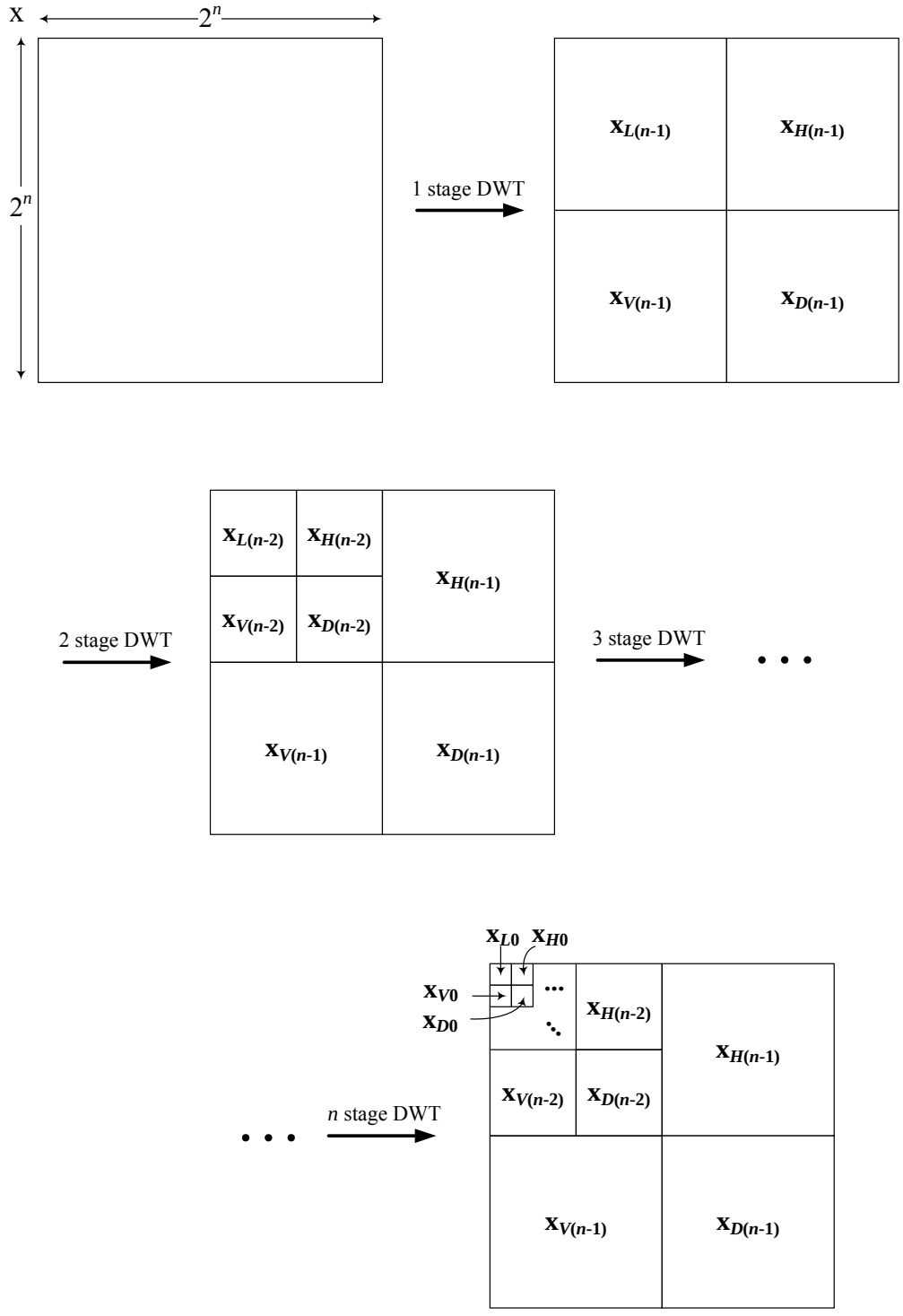

圖 2-1　*n* 階離散小波轉換示意圖

2.3 部分距離神經元搜尋法

神經元的搜尋是計算其對應權重向量與訓練向量間的量化誤差，也就是歐幾里得平方距離，以找出 k 個最小量化誤差的神經元。搜尋的過程包含了加法、乘法、邏輯判斷以及排序等運算，稱之為運算複雜度(computational complexity)。運算複雜度可以說是評估 CL 的一個重要指標，而搜尋 k 個最相似神經元的程序是 CL 最費時的一環，於是降低運算複雜度便成為增進 CL 效能最重要的工作。在這一節裡，我們將介紹以小波轉換為基礎的部分距離搜尋演算法則(Partial Distance Search，PDS)來達到降低運算複雜度的目的。

假設 $\mathbf{y}^j$ 代表第 j 個神經元的權重向量, $j = 1, 2, \ldots, N$。回想贏家通吃競爭式學習法則，給定一個訓練向量 x，則滿足(2.1)式的權重向量將遵照(2.2)式做調整。尋找的過程需要遍歷 N 個權重向量，所以當神經元的數量很大，及/或每個神經元對應之權重向量的維度很高，CL 演算法的計算複雜度就會變得很高。

令 X 與 Y^j 分別代表 x 與 y^j 做 n 階離散小波轉換後的結果，同時令 $D(u,v) = \sum_i (u_i - v_i)^2$ 表示向量 u 和 v 的平方距離。對於一個正交的離散小波轉換(orthogonal DWT)，我們可以得到這樣的結果：

$$D\left(x, y^j\right) = D\left(X, Y^j\right) \tag{2.3}$$

讓$\mathcal{F}_k = \{f_1, f_2, \ldots, f_k\}$代表與輸入向量$x$最接近的$k$個神經元的index，且$D\left(X, Y^{f_1}\right) \le D\left(X, Y^{f_2}\right) \le \ldots \le D\left(X, Y^{f_t}\right)$。

部分距離搜尋演算法則的目的是在減少搜尋與 $\mathcal{F}_k$ 的運算時間。一般來說部分距離搜尋演算法則包含兩個步驟。給定一個下限值(lower bound)，第一個步驟我們會先計算輸入向量與目標神經元部分維度的平方距離，若部分維度的平方距離超過此下限值，則目標神經元不需與訓練向量經過所有維度的平方距離運算即可排除成為獲勝神經元的可能性，否則才需要進行第二個步驟，即算出訓練向量與目神經元所有維度的平方距離來進行是否獲勝的判斷。

令X_i與Y_i^j分別代表X與Y^j的第i個小波係數，這些係數在小波領域是以zig-zag的順序來標記，如圖2-2所示：

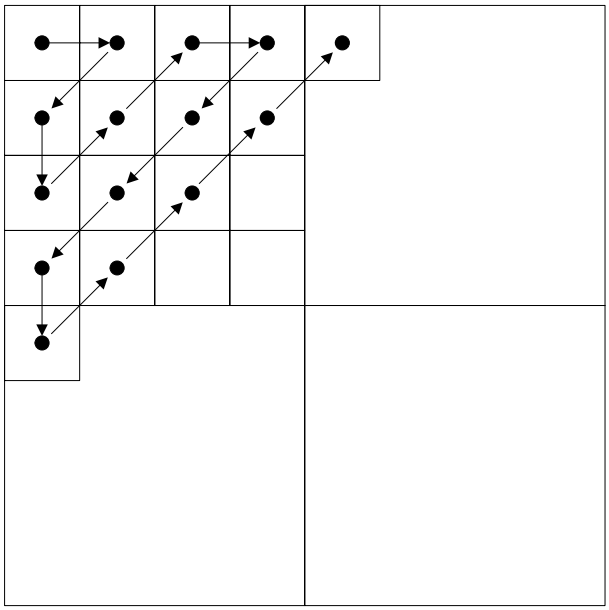

圖 2-2　zig-zag 排序方式

此外，令 $D^q\left(X,Y^j\right)=\sum_{i=1}^{q}\left(X_i-Y_i^j\right)^2$ 代表 X 與 Y^j 之間的部分距離，當 $D\left(X,Y^j\right)>D^q\left(X,Y^j\right)$，代表的是：

$$D\left(x,y^j\right)>D^q\left(X,Y^j\right) \tag{2.4}$$

令 D_i 代表在 PDS 過程中訓練向量 x 與目前最近第 i 筆距離神經元 y^{f_i} 的平方距離，$i=1,2\ldots,k$，則 $D_1\le D_2\le\ldots\le D_k$。在執行 PDS 之前，我們將 z 的值設定為 0，將 D_i 設定為無限大，$i=1,2\ldots,k$，所有的 f_i 的值將在 PDS 執行過程中產生。

PDS 找到 $\mathcal{F}_k$ 的流程如下：檢查從 Y^1 到 Y^N 為止的每一個向量，對於每一個向量，做 $\left|Y_1^j-X_1\right|$ 的計算，如果

$$\left|Y_1^j - X_1\right| > \sqrt{D_k} \tag{2.5}$$

根據 (2.4) 式，可以得到$\sqrt{D(x, y^j)} > \sqrt{D_k}$，於是可以得知$y^j$不屬於$k$個獲勝神經元，然後進行下一個神經元的比對運算。如果 (2.5)式不成立，則繼續進行接下來的 PDS 運算。從$q=2$開始，首先計算$D^q(X, Y^j)$，若是$D^q(X, Y^j) > D_k$，根據(2.4)式得知$\sqrt{D(x, y^j)} > \sqrt{D_k}$且剔除$y^j$屬於$k$個獲勝神經元的可能性；否則進行下一個$q$值重覆相同的處理程序直到$q = 2^n \times 2^n$為止。其中，部分距離$D^q(X, Y^j)$可以展開如下：

$$D^q(X, Y^j) = D^{q-1}(X, Y^j) + \left(X_q - Y_q^j\right)^2 \tag{2.6}$$

因此，對於新值q的部分距離可以累加前一個q值的部分距離而僅須計算$\left(X_q - Y_q^j\right)^2$。

重複的 PDS 處理程序會持續到y^j被剔除為k個獲勝神經元的可能或是q值達到$2^n \times 2^n$為止。當$q = 2^n \times 2^n$時$D^q(X, Y^j)$會與D_k進行比較，此時$D^q(X, Y^j) = D(X, Y^j)$，若是$D(X, Y^j) < D_k$，則f_k將從 $\mathcal{F}_k$ 中移除並將 j 加入 $\mathcal{F}_k$，再對 $\mathcal{F}_k$ 中的所有元素重新排序。當所有神經元都完成搜尋處理後，$\mathcal{F}_k$ 中的所有元素便是 k 贏家通吃法則中的贏家，這便完成了以 PDS 與 k 贏家通吃法則為基礎的競爭式學習法。

第三章 *k*CL 架構與硬體實現

我們的目標是為 *k* 贏家通吃競爭式學習法提供一個新的 PDS 架構。雖然直接將 PDS 實現在小波域上是可能的，但卻有許多缺點。例如，當權重向量的維度及/或神經元的數目很高，硬體就會需要龐大的記憶體來儲存這些權重向量，而這可能造成 FPGA 實現上的高面積成本。此外，在最基本的 PDS 演算法中，部分距離的計算是以全精度來執行；而全精度的 PDS 硬體實現也同樣具有高面積複雜度，故有限精度的 PDS 便有其需要性。又，這種最基本的 PDS 在做部分距離運算時，小波係數的加總一次只加總一個，而硬體實現因為允許多係數部分距離加總，故可達到高產量的搜尋。再者，這種最基本的 PDS 也僅適用於 WTA。而為了做到快速的 *k*-WTA，還需要有效率的排序電路。最後，學習率函數中帶有除法，而除法操作不論對於軟體或硬體實現均是耗時的運算，因此可將事先算好的學習率存於查詢表 (lookup table)，如此一來，即可降低學習階段的計算延遲。根據上述幾點，本論文提出的硬體架構將具有以下特性：子空間搜尋、有限精度計算、多係數部分距離累積、快速排序，以及查表式除法。

3.1 子空間搜尋(Subspace search)

因為 DWT 可以將向量的能量聚集在低頻的部分係數，所以在小波域上進行部分距離搜尋時僅需搜尋低頻的部分係數，藉此可更進一步加速部分距離搜尋處理程序，大量降低神經元搜尋的運算複雜度。同時子空間搜尋的權重向量集合僅需儲存Y_{Lm}^{j}，也就是y_{Lm}^{j}經由離散小波轉換後的部分低頻係數，其中$j=1,2,\cdots,N$。這一點明顯地降低權重向量集合的儲存空間，亦是搭配子空間搜尋的 *k*CL 在 VLSI 硬體實現的優點。

雖然Y_{Lm}^{j}，$j=1,2,\cdots,N$，可以事先由原始權重向量集合經過離散小波轉換後取部份係數再儲存於硬體中，但是訓練向量 x 的離散小波係數X_{Lm}，卻需於 *k*CL 中的運算求得，所以在子空間搜尋 *k*CL 硬體中仍需具備 DWT 運算單元。使用 Haar 小波轉換來做 DWT 運算單元硬體實現，是因為它具有簡單的低通濾波器 (impulse response，$\left\{\frac{1}{2},\frac{1}{2}\right\}$) 和高通濾波器 (impulse response，$\left\{-\frac{1}{2},\frac{1}{2}\right\}$)，同時用來實做 DWT 運算單元也不需要乘法處理。圖 3-1 顯示了 Haar 小波運算單元的 VLSI 硬體架構，其中 x 與X_{Lm}的維度分別為8×8以及4×4，也就是說在這個架構中$n=3$，$m=2$。X_{Lm}是由 $\mathbf{x}_{L0}$和 $\mathbf{x}_{Vi}$、$\mathbf{x}_{Hi}$、$\mathbf{x}_{Di}$所組成，$i=0,1$。DWT 運算單元運算的過程中，首先輸入向量 x 經由一階的 Haar DWT

後捨去 $\mathbf{x}_{V2}$、$\mathbf{x}_{H2}$、$\mathbf{x}_{D2}$ 等高頻部份，保留低頻的 $\mathbf{x}_{L2}$。然後如圖 3-1 所示，對 $\mathbf{x}_{L2}$ 繼續做兩階的 Haar DWT 後求得 $\mathbf{x}_{L0}$ 和 $\mathbf{x}_{Vi}$、$\mathbf{x}_{Hi}$、$\mathbf{x}_{Di}$，其中 $i=0,1$。

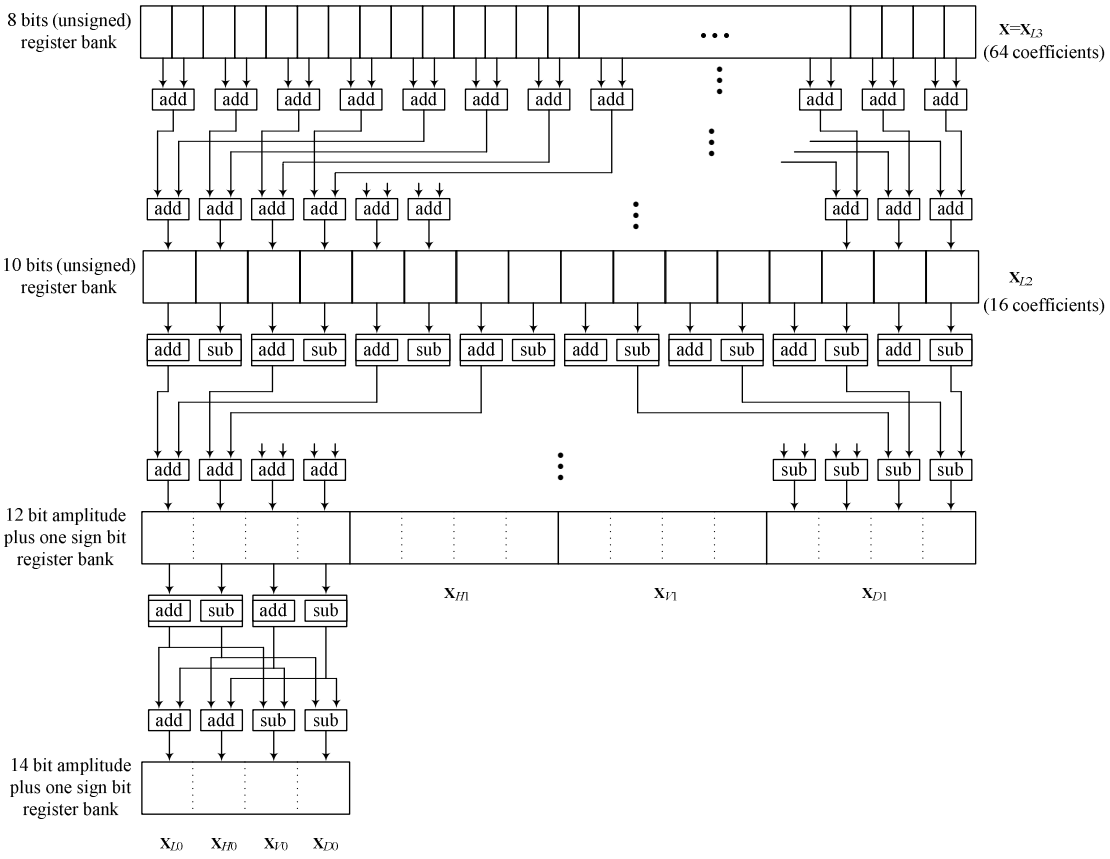

圖 3-1　DWT 運算單元硬體架構圖

3.2 有限精度計算(Finite precision calculation)

除了簡單易於硬體實現之外，Haar 小波還有另一個優點就是它的小波係數是有限精確度的。以圖 3-1 的小波轉換運算為例來說，給定輸入向量 x 的位元平面(bitplane)數量為 8，每做一階的 Haar 小波轉換後會增加兩個位元平面，也就是說做完三階 Haar 小波轉換後得到的 X_{Lm} 位元平面數量為 14，這些係數都會被精確地儲存於 RAM 裡用來做部分距離搜尋運算。

雖然要以硬體來實現全距離搜尋是可能的，但這會導致儲存 Y_{Lm}^{j} 的面積複雜度變高。而事實上，移除部分LSB (least significant bit) 位元平面對於*k*-WTA的效能只會造成極小的影響，可是卻能大幅降低硬體實現時 Y_{Lm}^{j} 儲存空間的面積複雜度，所以小波係數 X_{Lm} 的所有位元平面並不需要被精確的儲存。

在有限精度計算中，計算精度的縮減程度是以 l 來表示，其代表移除計算係數LSB位元平面的數目。由此可知，在有限精度計算中，X_{Lm} 及 Y_{Lm}^{j} 都會被縮減 l 個位元平面。

此外，運用有限精度計算的技巧，亦有助於降低平方距離計算及

排序電路的面積複雜度。

3.3 多係數部分距離累積 (Multiple-coefficient partial distance accumulation)

基本的部分距離搜尋演算法則在累加部分距離時一次只累加一個係數的距離，如(2.6)式所示，因此在硬體實現的時候僅需要一個乘法器。為了要提高 k 贏家通吃競爭式學習系統的產量，我們採用一次計算並累加 δ 個係數的方式來加速部分距離搜尋的計算，也就是在硬體實現中計算平方距離的單元使用 δ 個乘法器平行同時計算後累加。在子空間部分距離搜尋中，部分距離的計算如(3.1)式所示：

$$D^{q}\left(X_{Lm},Y_{Lm}^{j}\right)=D^{q-\delta}\left(X_{Lm},Y_{Lm}^{j}\right)+\sum_{i=q-\delta+1}^{q}\left(X_{i}-Y_{i}^{j}\right)^{2} \tag{3.1}$$

其中 X_i 跟 Y_i^j 分別是 X_{Lm} 與 Y_{Lm}^j 的第 i 個係數，這些係數的標記順序如前一章的圖 2-5 所呈現的 zig-zag 方式來排序。另外，低頻子頻帶的大小 $2^m \times 2^m$ 也必須是 δ 的整數倍。

本研究用向量平方距離計算單元(vector squared distance computation，VSDC) 來處理(3.1)式中右半部份 $\sum_{i=q-\delta+1}^{q}\left(X_i - Y_i^j\right)^2$ 的運算，硬體架構圖如圖 3-2 所示。

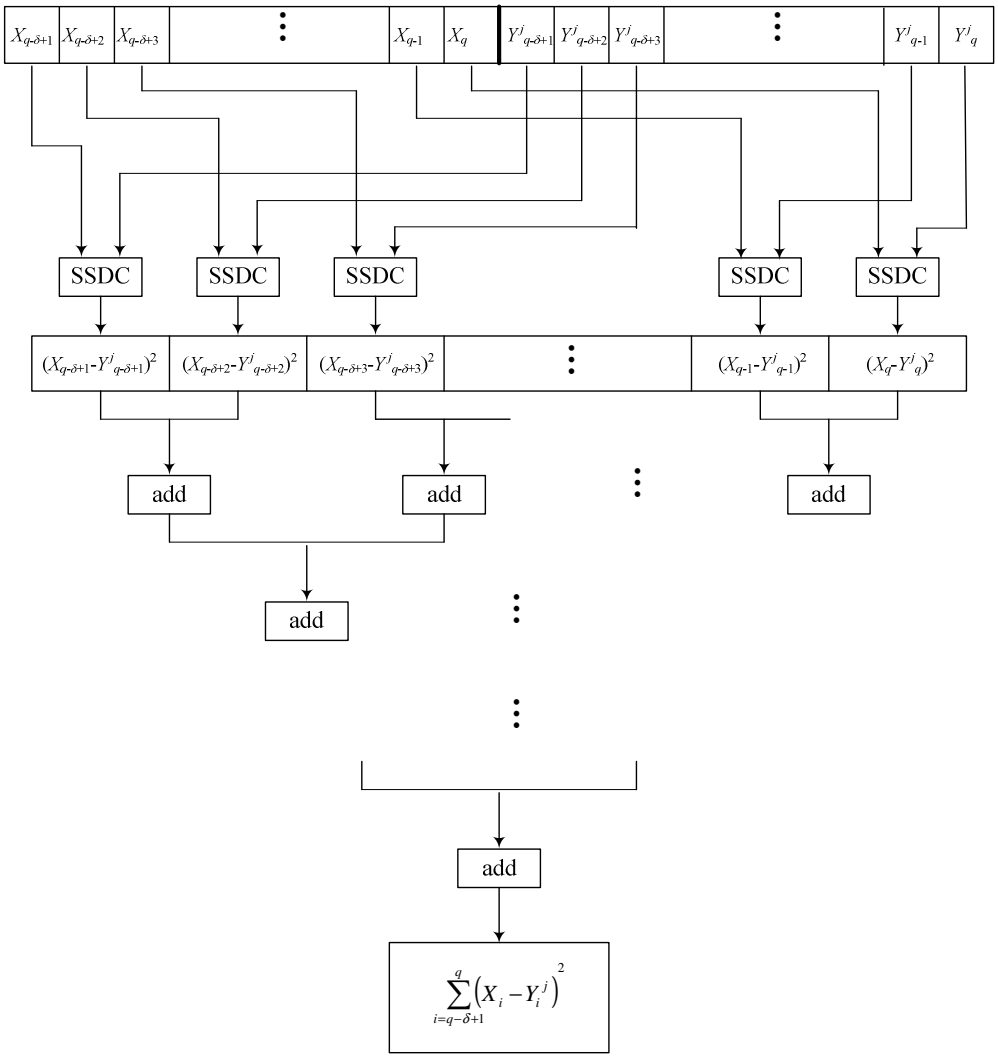

圖 3-2　向量平方距離計算單元 VLSI 架構示意圖

向量平方距離計算單元中包含了δ個用來計算兩個係數平方距離的純量平方計算單元(scalar squared distance computation，SSDC)，再透過樹狀加法的方式加總δ個 SSDC 單元的計算結果。其中 SSDC 的硬體架構如圖 3-3 所示。

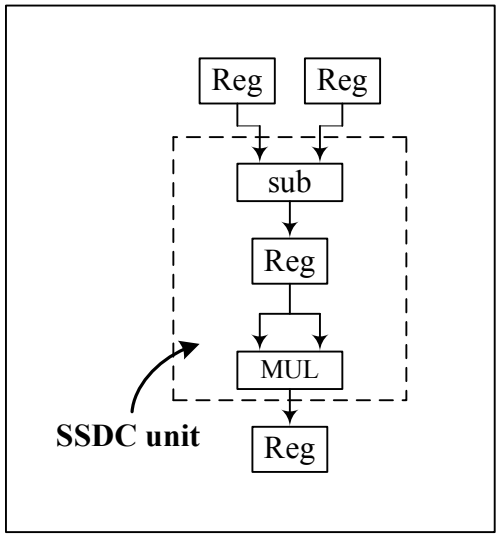

圖 3-3　純量平方距離計算單元 VLSI 架構示意圖

很明顯地，VSDC 單元的面積複雜度會隨著δ的增加而呈現倍數的急遽成長，另外位元平面縮減的程度l 也同樣的對面積複雜度有所關聯給定一個固定的係數個數δ，隨著有限精度縮減程度l 的增加可以不斷的降低面積複雜度。

3.4 排序電路

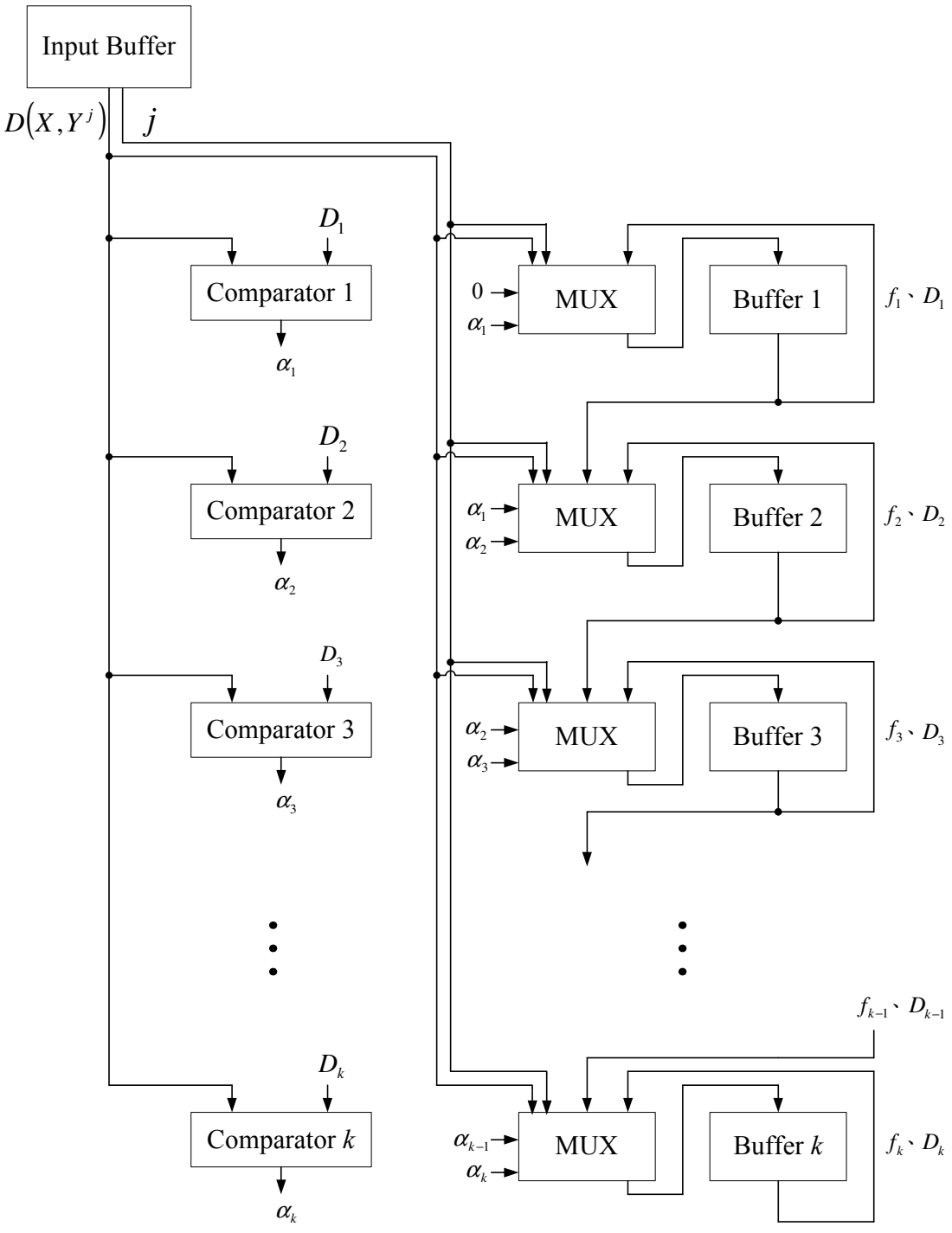

圖 3-4　排序電路架構示意圖

在 PDS-Based 的 *k*-WTA 流程中，如果目標神經元被判定需要更新，即代表要將 f_k 從 $\mathcal{F}_k$ 中移除並把 j 加入 $\mathcal{F}_k$，再對 $\mathcal{F}_k$ 中的所有元素重新排列。此處我們提出一個全新的排序電路，只需要 buffer、comparator 和 multiplexer 三種硬體元件即可在 2 個 clock cycles 的時間內完成重新排序的動作，硬體架構示意如圖 3-4。

我們在小波域進行 PDS 搜尋程序，目標神經元根據 $D(X,Y^j)$ 被判定為需要更新後，排序電路將被啟動，而排序過程分為兩個步驟。第一個步驟將 $D(X,Y^j)$ 和 D_i 做比較，比較結果 α_i 將被作為選擇線傳送至 MUX，$i=1,2,\ldots,k$，設定 $\alpha_0=0$。第二個步驟 MUX 根據選擇線內容決定 MUX 輸出內容，第 i 個 MUX 的選擇線與輸出內容關係如表 3-1。

α_{i-1}	α_i	output
x	0	f_i 、 D_i
0	1	j 、 $D(X,Y^j)$
1	1	f_{i-1} 、 $D_{(i-1)}$

表 3-1　排序電路內部多工器的真值表

在這邊以一個簡單的範例當說明，如圖 3-5，$k=5$，D_1 到 D_5 預設值都是無限大，第一個輸入值 3，同時和 D_1 到 D_5 做比較，結果輸出

α_1 到 α_5 都是 1，於是將輸入值 3 放到 D_1，D_2 到 D_5 的值則由 D_1 到 D_4 平移而來。第二個輸入值 4，同時和 D_1 到 D_5 做比較，結果輸出 $\alpha_1 = 0$，α_2 到 α_5 都是 1，於是將輸入值 4 放到 D_2，D_1 保持不變，D_3 到 D_5 則由 D_2 到 D_4 平移而來，以此類推完成六筆輸入值後，D_1 到 D_5 的值分別等於 1、2、3、4、5。

input	3		4		1		5		5		2		
D_1	∞	1	3	0	3	1	1	0	1	0	1	0	1
D_2	∞	1	∞	1	4	1	3	0	3	0	3	1	2
D_3	∞	1	∞	1	∞	1	4	0	4	0	4	1	3
D_4	∞	1	∞	1	∞	1	∞	1	5	0	5	1	4
D_5	∞	1	∞	1	∞	1	∞	1	∞	1	5	1	5

圖 3-5　排序範例

3.5 查表式除法 (lookup-table based division)

在 *k*CL 學習過程的第二階段，獲勝神經元的對應權重向量將進行調整的動作。所謂的調整，是先將學習率乘上權重向量與訓練向量的差值，然後再加回去原始權重向量。學習率為一個嚴格遞減函式，將隨著時間增加而嚴格遞減，因而影響整個類神經網路收斂的速度，故學習率的決定關係到整個學習成效的好壞。本論文採用的學習率函數源於[14]，可以下式表示：

$$\eta_i(r) = \frac{1}{4 \times r} \quad , \; i = 1,2,\ldots,N \tag{3.2}$$

其中 r 代表的是第幾次更新。

由上式可知，學習率函數的計算中帶有除法，而這不管對於軟體或硬體而言都是不小的負擔。為此，我們技巧性地將學習率的公式以查表及移位等運算來完成。

請觀察(3.3)式。對於任何大於 0 的整數 w 而言，

$$\eta_i(r) = \frac{1}{4 \times r} = \frac{2^w}{r} \times 2^{-(w+2)} \tag{3.3}$$

給定一個正整數 w，即代表查詢表 ROM 中每一個實體(entity)的位元寬度為 w，且 ROM 中的第 r 個實體，其值為 $\frac{2^w}{r}$。因此，對任何小於 2^w 的 r 而言，$\frac{2^w}{r}$ 的值都可在 ROM 中經由查詢得到。把 ROM 的輸出結果乘上權重向量與訓練向量的差值，然後再右移(right shift)共 $(w+2)$ 個位元，即完成學習量 $\eta^{j*}(\mathbf{x} - \mathbf{y}^{j*})$ 的求取。

值得注意的是，這個查詢表 ROM 的空間大小是可以縮減的。在採用有限精度計算的情況下，給定 2^w，當 r 大到一定程度，則不同的 r 都會擁有相同的 $\frac{2^w}{r}$ 值，因此這些擁有相同 $\frac{2^w}{r}$ 值的 r，可以藉由共享 ROM 中一個實體的方式來節省儲存空間。而這種方式只要在 ROM 中加入一個編碼器就可達到。這個編碼器會將所有共享同一實體的 r 全都對應到相同的位址，因此可讓我們省下可觀的儲存空間。舉例來說，當 $w = 9$，則 ROM 中僅需存有 170 個實體即可，而不用存到 2^9 個那麼多。

再者對 $\frac{2^w}{r}$ 而言，若我們採用的是有限精度計算，則 shift 的操作就會變得很簡單。在本論文的架構中，$\frac{2^w}{r}$ 和權重向量都是以整數的

格式做儲存。在這種情況下，對$(\mathbf{x} - \mathbf{y}^{j*})\times\frac{2^{w}}{r}$做右移$(w+2)$個位元的操作，就等於將相乘的結果捨棄$(w+2)$個 LSB 位元。

利用查表法及位移運算所得到的學習率近似值對於調整結果僅有極小的影響，但卻能省下可觀的計算時間，也不會造成硬體面積複雜度的過度擴張。

3.6 子空間部分距離搜尋法硬體實現

本論文的子空間部分距離搜尋硬體架構，其運作流程如下所列：

假定條件：給定m、l、δ的值

神經元$Y_{Lm}^{1},\cdots,Y_{Lm}^{t}$已先計算好，並以有限$l$精度計算處理後儲存於記憶體中。

Step1：擷取輸入向量x

計算小波係數X_{Lm}

移除小波係數X_{Lm}的l個 LSB 位元平面

Step2：初始化 $j=0$

初始設定 J = {0,…,0} (i.e., j_i=0, i=1,…,k.)

初始設定 $D_i=\infty$ ，$i=1,2,\ldots,k$

Step3：如果 $j=N$ ，則停止

否則， $j=j+1$

初始化 $q=\delta$

Step4：根據(3.1)式計算 $D^q\left(X_{Lm},Y_{Lm}^j\right)$

比較 $D^q\left(X_{Lm},Y_{Lm}^j\right)$ 與 D_k

如果 $D^q\left(X_{Lm},Y_{Lm}^j\right)>D_k$ ，則到 Step3 進行下一個神經元的搜尋，否則就繼續往下一個步驟執行

Step5：如果 $q=2^m\times 2^m$

將 f_k 從 $\mathcal{F}_k$ 中移除並把 j 加入 $\mathcal{F}_k$

將 $D\left(X,Y^j\right)$ 輸入排序電路並移除 D_k

根據 D_i 對 $\mathcal{F}_k$ 內所有元素 f_i 做排序，$i=1,2,\ldots,k$

然後到 Step3 進行下一個神經元的搜尋

否則就繼續往下一個步驟執行

Step6：$q = q + \delta$

回到 Step4 繼續部分距離的計算累加跟比較

當整個部分距離搜尋的流程完成後，系統保留的 $\mathcal{F}_k$ 就是相對於訓練向量 x 的 k 個最近距離神經元(final k closest neurons to x)。

從上面的流程可以觀察到，神經元的搜尋是在小波域上面進行(X_{Lm} 是 x 的小波低頻係數，$Y_{Lm}^1,\cdots,Y_{Lm}^t$ 是所有神經元的小波低頻係數)，所以這是一個子空間部分距離搜尋演算法的架構。這個架構使用了有限精度計算的技巧，在部分距離搜尋處理程序之前會先移除 X_{Lm} 以及 $Y_{Lm}^1,\cdots,Y_{Lm}^t$ 的 l 個 LSB 位元平面，最後才根據(3.1)式計算 $D^q\left(X_{Lm},Y_{Lm}^j\right)$ 的值。同時，部分距離搜尋的部分是使用多係數部分距離累積的技巧。

總結來說，本文針對 k 贏家通吃競爭式學習法的硬體實現提出了四個極佳的技巧，包括子空間搜尋、有限精度計算、多係數累積，以及查表式除法，這四個技巧能夠大幅度的降低硬體實現所需的面積複雜度，提高系統產量。

3.7　*k*CL 之 VLSI 硬體架構

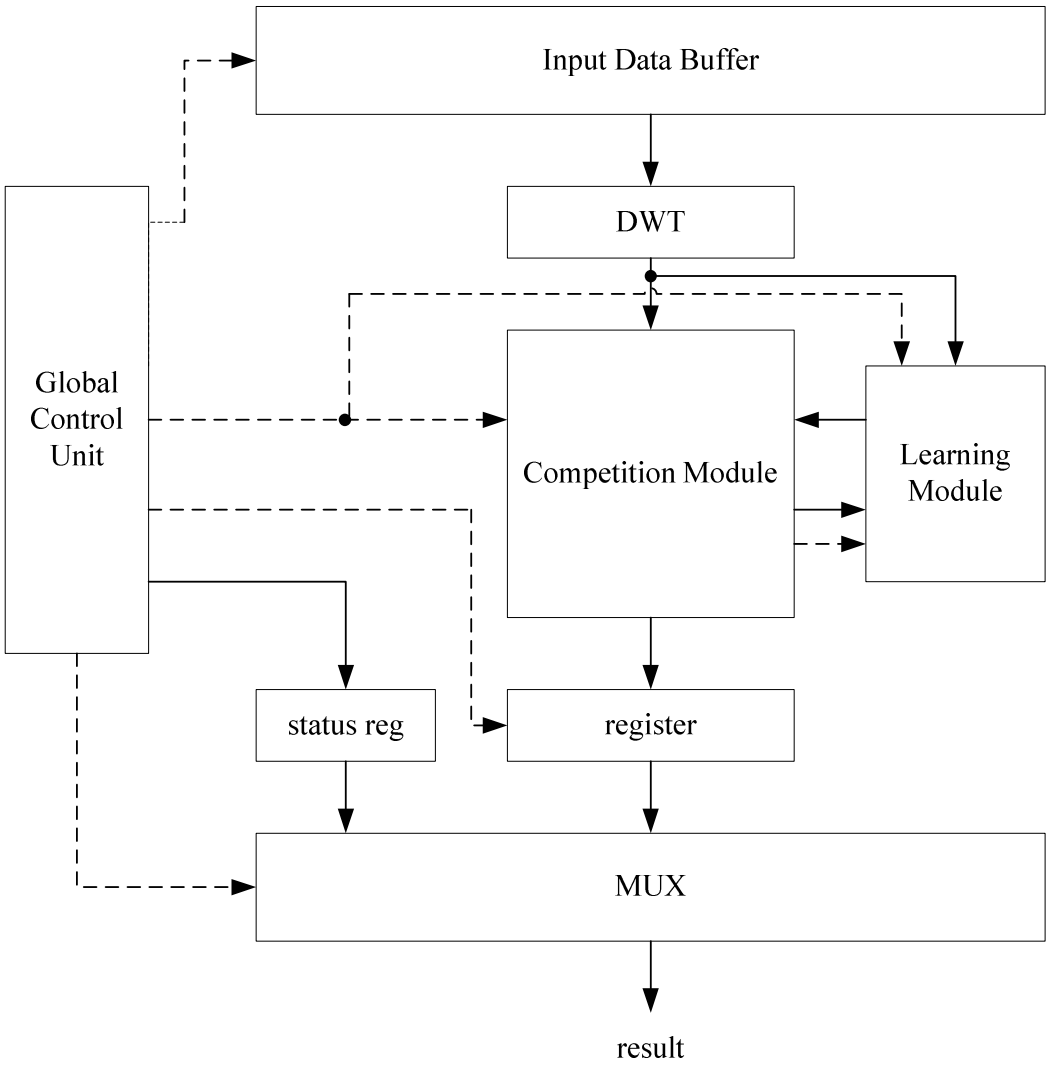

圖 3-6　*k*CL 之 VLSI 系統架構

本論文提出了適合硬體實現的 k 贏家通吃競爭式學習法，圖 3-6 顯示這個法則的基本 VLSI 硬體架構，包含一個全域控制單元(Global Control Unit，GCU)、一個 DWT 運算單元、一個競爭模組(Competition Module)、一個學習模組(Learning Module)、一個多工器，以及一些儲

存中間值、暫時結果、暫存資料的暫存器。全域控制單元用於協調各個單元的運作；DWT 運算單元對訓練向量 x 進行 Haar 小波轉換獲得低階的小波係數 X_{Lm}；競爭模組用來執行 *k*CL 的競爭階段；學習模組除了儲存神經元的對應權重向量 $Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$，用以提供給競爭模組進行搜尋比對之外，也負責執行 *k*CL 的學習階段；多工器的輸出則為系統的狀態，軟體端可從此輸出得知系統是否完成處理、是否允許接收資料。

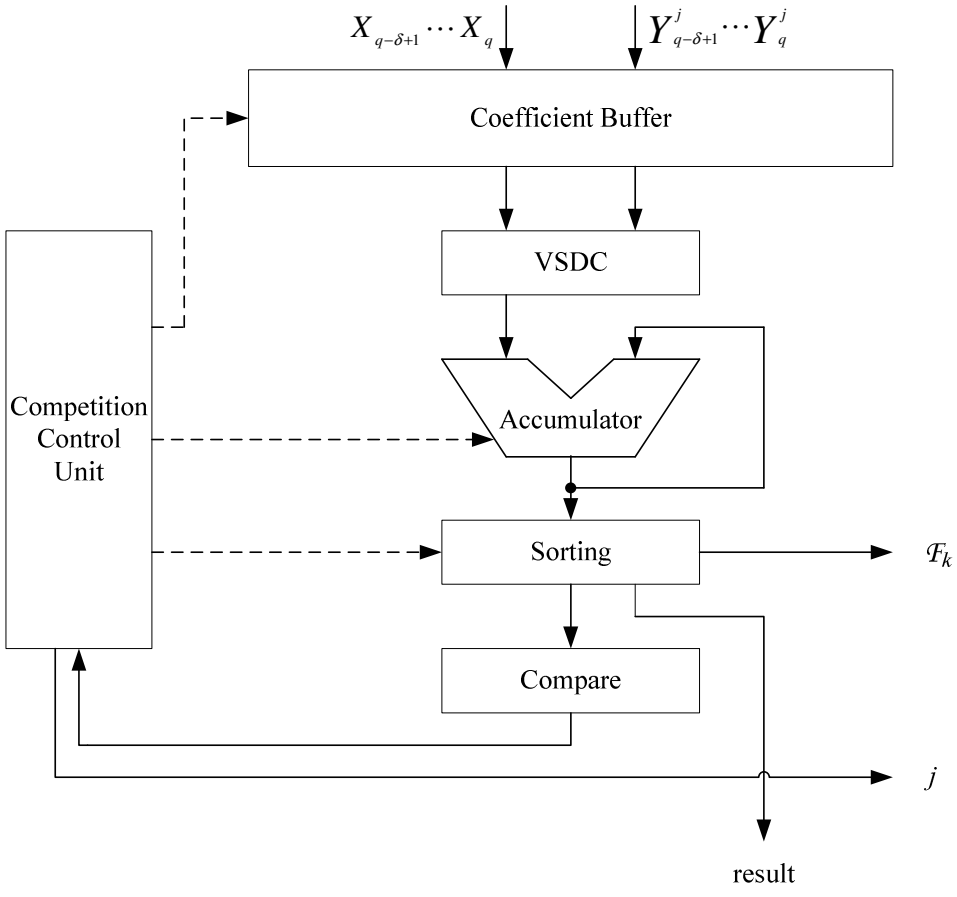

圖 3-7　競爭模組之 VLSI 架構

競爭模組的硬體架構如圖 3-7 所示。含有一個計算向量平方距離的 VSDC 運算單元、一個累積器、一個比較器、一個競爭模組控制單元、以及一個排序電路。VSDC 運算單元以及累積器可以用來計算和儲存 $q=\delta,2\delta,\cdots,2^m\times 2^m$ 時的部分距離 $D^q\left(X_{Lm},Y_{Lm}^j\right)$。比較器則用來比較部分距離 $D^q\left(X_{Lm},Y_{Lm}^j\right)$ 跟目前第 k 小量化誤差 D_k。比較器的結果會傳送到競爭模組控制單元，而競爭模組控制單元會根據這個結果，決定是否要重置累積器及更新 $\mathcal{F}_k$，各單元間的運作由競爭模組控制單元和全域控制單元協調控制。

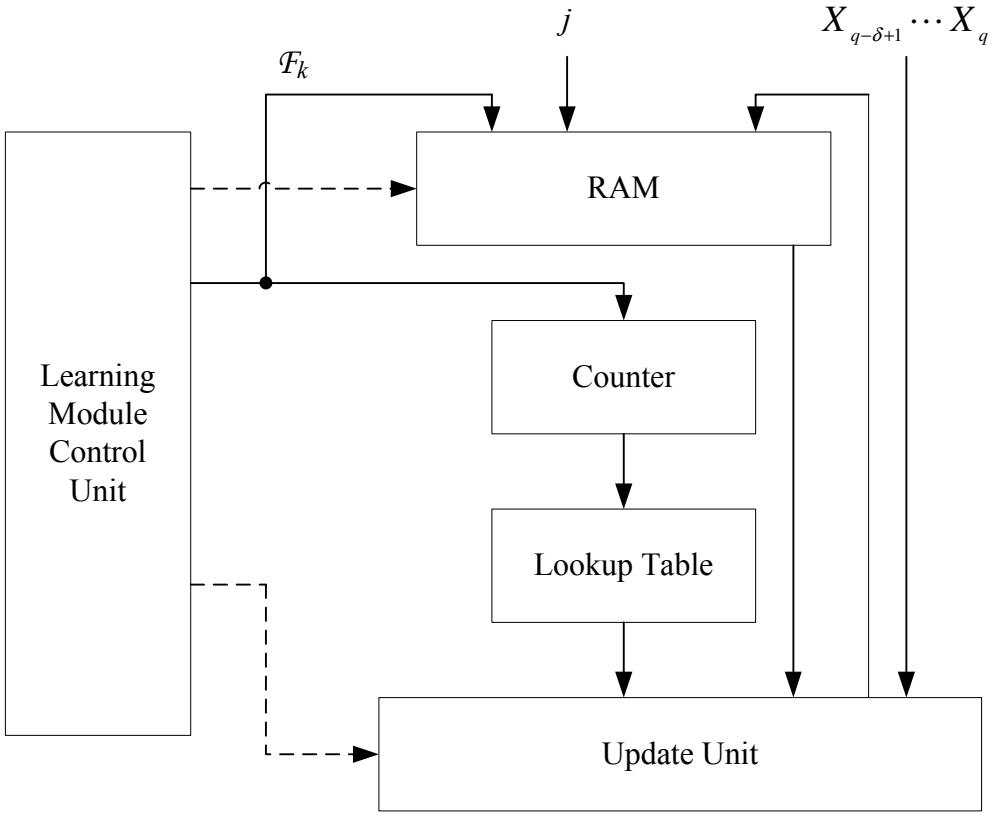

圖 3-8　學習模組之 VLSI 架構

在競爭模組完成計算後會啟動學習模組中的更新電路來進行權重向量的調整。圖 3-8 為學習模組的硬體架構圖。

學習模組含有一個學習模組控制單元、一個由 Altera 所提供之 LPM megafunction 所產生的 RAM、*N* 個計數器、一個查詢表、一個更新單元。RAM 中儲存的是 $Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$，當收到來自競爭模組的神經元索引值 j，RAM 即會輸出存於位址 j 的權重向量 Y_{Lm}^{j}；當學習模組收到來自競爭模組的 $\mathcal{F}_k$，也就是有權重向量需要被調整時，圖 3-9 的更新電路就會被啟動。當更新電路完成計算，並將計算結果寫回 RAM 後，我們就完成了 *k*CL 演算法中的學習階段。

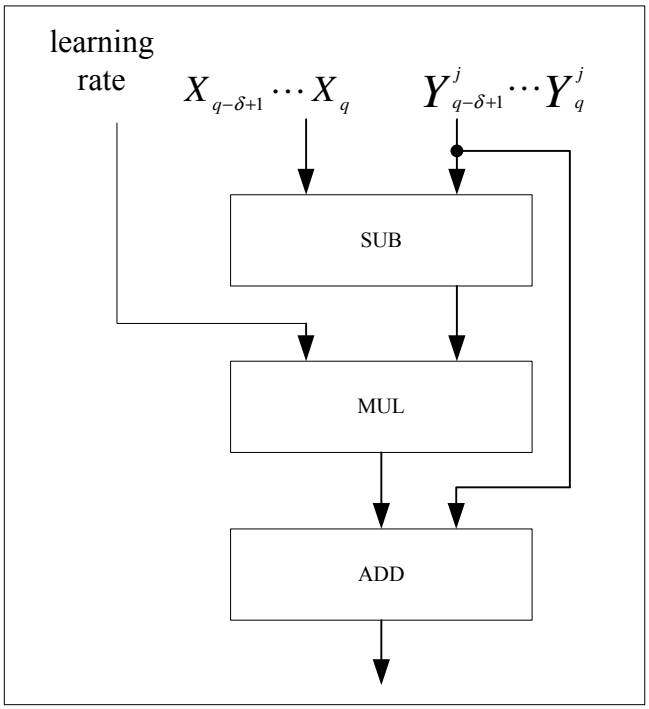

圖 3-9　權重向量調整電路之 VLSI 架構

3.8 軟硬體共同設計(Hardware/Software Codesign)

在本論文使用一個狀態暫存器配合全域控制單元(Global Control Unit，GCU)來協調各個單元的運作。在我們的設計中，*k*CL 模組具有兩個旗標用來表示狀態：Module_Available 表示模組是否有空，Result_Available 表示模組是否完成 *k*CL 運算，這些旗標有助於 GCU 控制系統流程。

這個狀態暫存器裡面記錄了以下幾種狀態：

(1) Ready_for_Input：是否可以輸入資料到硬體。

(Ready_for_Input = Module_Available & Input_Buffer_Available)

(2) Input_Buffer_Available：是否可使用 data_buffer 存放輸入向量。

(3) Input_Data_Available：訓練向量 x 是否已完整送進 data_buffer。

(4) Ready_for_Output：是否可以從硬體讀取運算結果。

(5) Write_Back_Result_Complete：軟體是否已接收運算結果。

全域控制單元根據狀態暫存器的狀態變化負責協調資料的存取和各單元的運作，完整的狀態轉換流程如圖 3-10 和圖 3-11 所示：

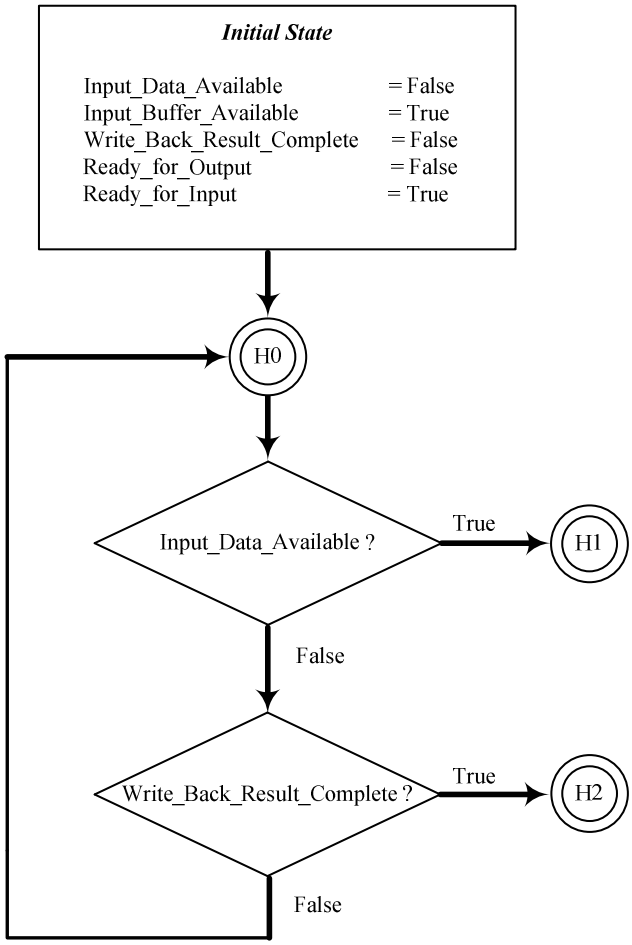

圖 3-10　全域控制單元的狀態流程之一

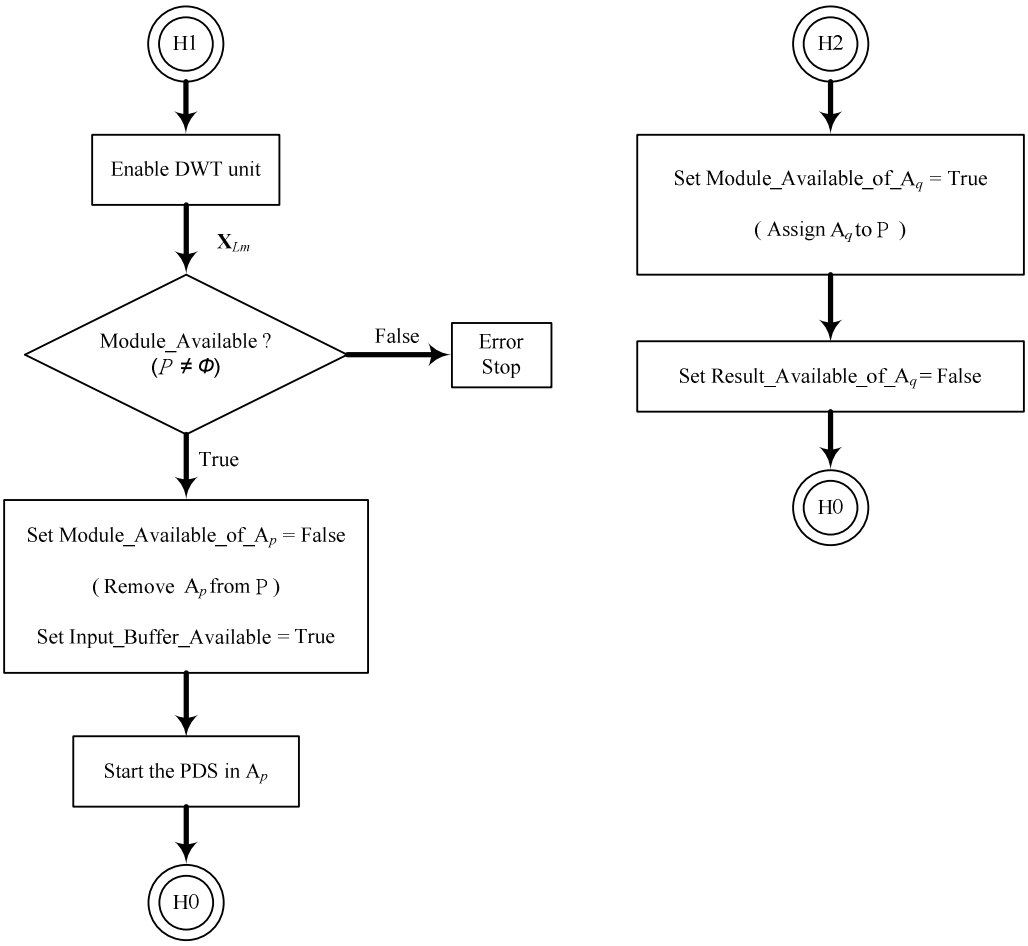

圖 3-11　全域控制單元的狀態流程之二

系統的初始狀態定義在圖 3-10 的一開始 Initial State，系統啟動之初 data_buffer 是空的，允許寫入資料 (Input_Buffer_Available = True)，未有資料放入 data_buffer 等待運算 (Input_Data_Available = False)，模組也沒有結果完成 (Ready_for_Output = False)，系統也不會接收到軟體端完成讀取運算結果後的回傳訊息 (Write_Back_Result_Complete = False)，*k*CL 模組有空且允許寫入資料

(Ready_for_Input = True)。

當軟體端已經將輸入向量完全傳送至 data_buffer 後，會將狀態設成 Input_Data_Available = True，硬體的全域控制單元會進入 H1 的流程，啟動 DWT 單元進行小波轉換運算得到低階小波係數 X_{Lm} 後，*k*CL 模組便開始進行部分距離搜尋的運算。如果 *k*CL 模組已完成分類運算且軟體端已接收結果，則 Write_Back_Result_Complete 會被軟體端設為 True，此時若 Input_Data_Available = False，硬體端便會進入 H2 的流程將已被取走分類結果的 *k*CL 模組設為有空，使其可準備繼續進行下一筆訓練向量的運算程序。

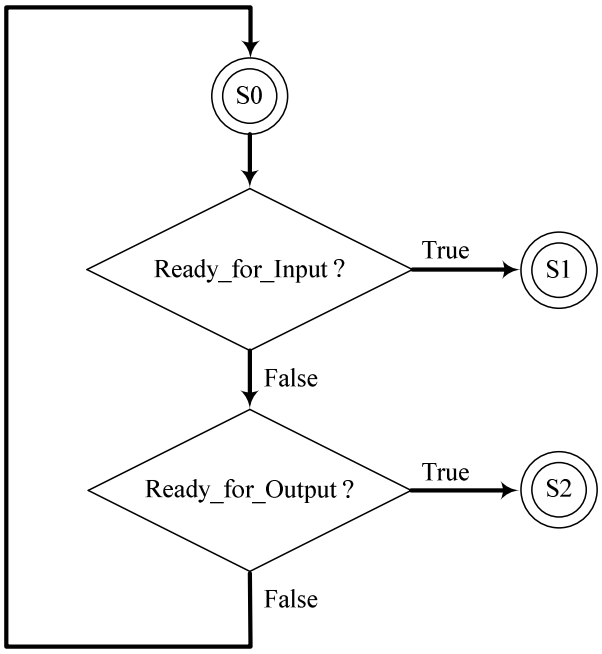

圖 3-12　軟體端的狀態流程之一

軟體端可以透過呼叫軟體巨集的方式，使用客製指令(Custom instruction) 來存取使用者專用邏輯區塊電路。我們用簡單的 C 語言配合客製指令來開發與專用硬體電路溝通的軟體，可以傳送資料到 *k*CL 專用硬體中，也可以從專用硬體接收 *k*CL 的運算結果。軟體端的流程如圖 3-12 和圖 3-13 所示，若 *k*CL 模組有空且 data_buffer 可供暫存資料，此時的狀態暫存器顯示 Ready_for_Input = true，各種狀態成立的條件已經討論定義，軟體可進入 S1 流程輸入資料到專用硬體電路中。若 *k*CL 模組沒空，但有執行結果等候軟體端接收，此時狀態暫存器顯示 Ready_for_Output，軟體可進入 S2 流程從專用硬體電路擷取執行結果。

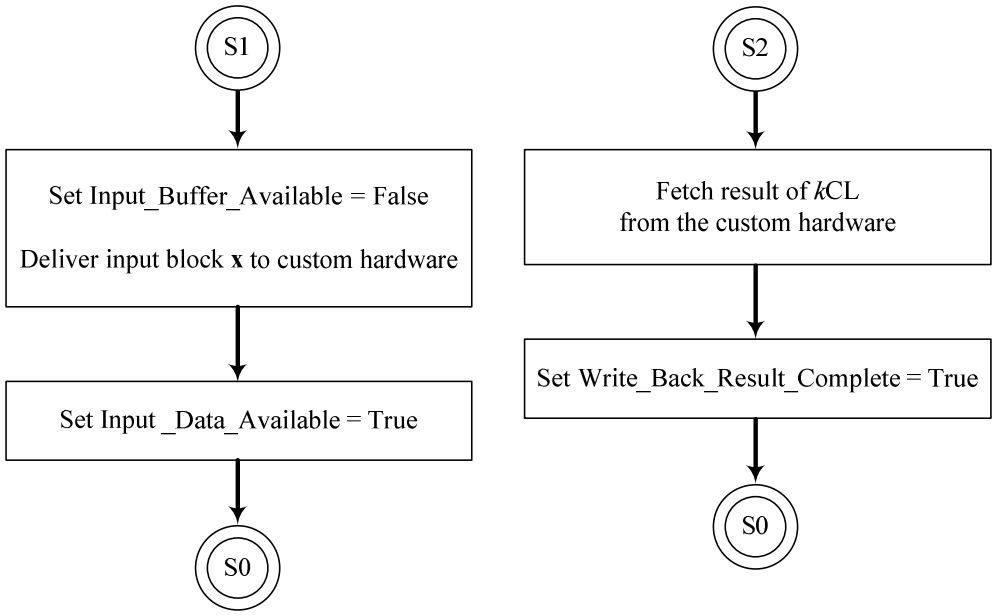

圖 3-13　軟體端的狀態流程之二

3.9 內嵌於軟核心處理器的 PDS 使用者自訂邏輯區塊

為了要實際量測本論文提出架構的效能，我們把部分距離搜尋演算法以使用者專用邏輯區塊電路(user custom logic block)的方式實現，然後內嵌於軟核心 Nios CPU 中成為算術邏輯單元的一部份。Nios CPU 是一個五級管線化的一般用途精簡指令集(RISC)處理器。軟核心處理器和一般微處理器主要的差異點在於軟核心處理器是一個可重設定(re-configurable)的架構，因此自訂的 VLSI 架構可以內嵌於軟核心處理器中用來做實際效能的量測。

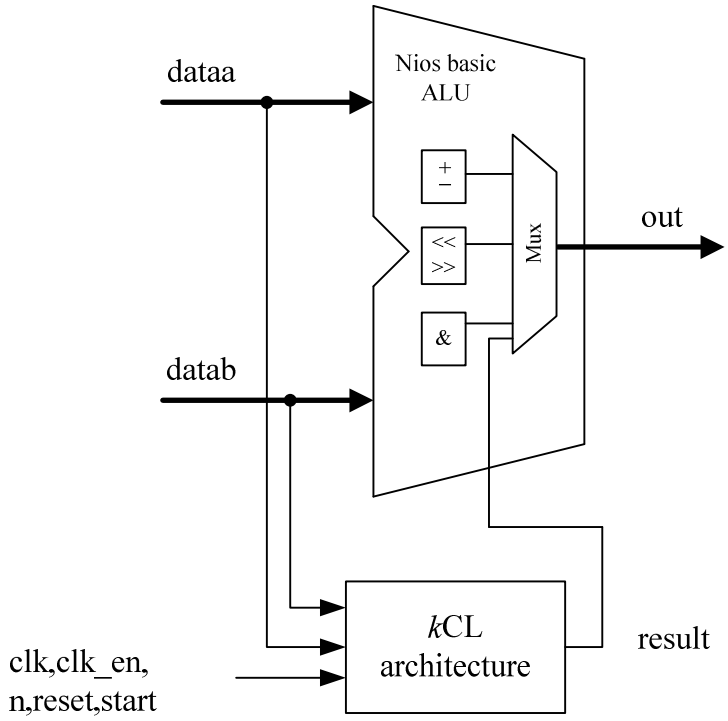

圖 3-14　使用者專用邏輯區塊在 Nios CPU 中的位置

本論文以使用者專用邏輯區塊電路(user custom logic block)實現部分距離搜尋演算法則，圖 3-14 中顯示了這個使用者專用邏輯區塊電路在 Nios 處理器 ALU 中的位置。從圖中可以觀察到這個使用者專用邏輯區塊電路和基本算術邏輯單元使用同一條輸入和輸出匯流排，而這個使用者專用邏輯區塊電路是用客製指令(Custom Instruction)來進行存取的動作。這個架構以部分距離搜尋專用硬體電路為例，其介面如圖 3-15 所示。專用邏輯區塊電路和 Nios 處理器之間的同步是靠“clk”、“clk_en”、“reset”和“start” [2]這幾根接腳來完成，而“dataa”和“datab”這兩根接腳是用來接收向量 *x* 的輸入資料，而傳送專用邏輯區塊電路向量量化器編碼結果或是狀態暫存器資料到 Nios 處理器是由“result”接腳負責，另外透過“n”接腳可以定義多種資料控制訊號溝通專用邏輯區塊電路與處理器。

Nios 是 32 位元的處理器，“dataa” 和 “datab” 腳位的寬度就是 CPU 資料匯流排的寬度 32 位元。對於一個維度大且位元平面數量多的訓練向量 *x* 而言，要利用 “dataa” 和 “datab” 來傳送 *x* 的資料可能會需要多個時脈週期。舉例來說，考慮一個維度 8×8 的輸入向量 *x*，假設 *x* 有 8 個位元平面，則共需 512 個位元，利用 “dataa” 和 “datab” 來傳送 *x* 到使用者專用邏輯區塊電路需要花費 8 個時脈週期，因此在

使用 DWT 單元進行小波轉換之前，要在專用硬體電路裡設計 data_buffer 用來儲存每一次利用 “dataa” 和 “datab” 傳送進來的輸入資料，確定一個輸入向量完整輸入到 data_buffer 後才能啟動模組進行運算。

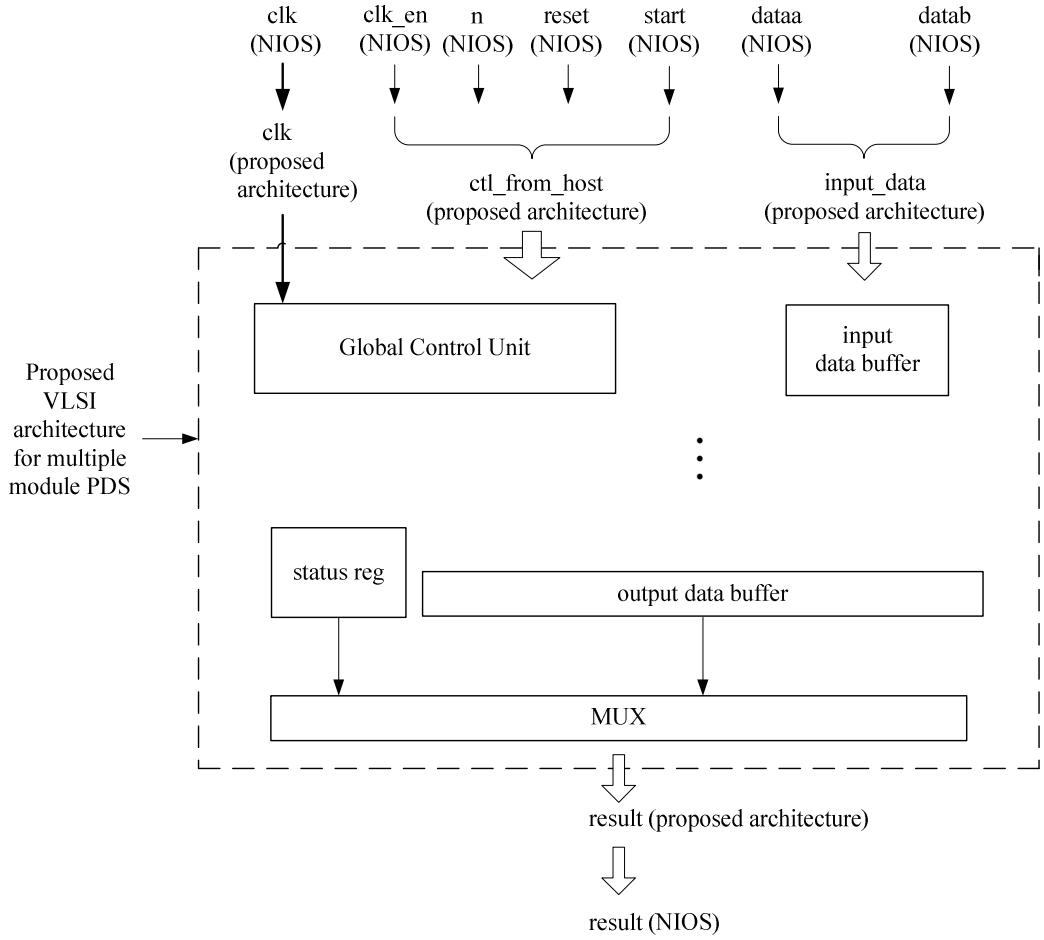

圖 3-15　硬體架構介面圖

第四章 實驗數據與效能比較

我們已對本論文所提出之架構完成實際效能的量測。本章將呈現這些實際效能量測的實驗數據，並做一個效能比較。

整個 *k*CL 系統是在 Altera 的 Stratix EP1S40F780C5 FPGA 發展板上合成硬體電路，此發展板提供的邏輯單元數(logic elements，LEs) 最大值為 41250 LEs [1]，而我們也將以 LEs 的數量來做為面積複雜度的量測基準。選擇在 Stratix FPGA 上實現驗證硬體電路，是因為可規劃系統晶片設計(System On a Programmable Chip，SOPC) 可以快速地將硬體設計實現並完成驗證，具備快速上市與系統再修改等特性，非常適合做雛型設計。

首先我們進行一項實驗，用以探討有限精度計算對影像品質與神經元集合 RAM 大小的影響。在這個實驗裡，我們給定神經元數目 $N=256$，向量維度 8×8 (i.e., $n=3$)，子空間搜尋的 X_{Lm} 和 Y_{Lm}^{j} 維度為 4×4(i.e., $m=2$)。其中神經元集合中的權重向量是由 LBG 演算法對兩張 512×512 的圖像 “Lena” 和 “Zelda” 做訓練求得。實驗數據中的 PSNR 值是分別對 “Girl”、 “Baby” 和 “House”三張測試影像做 VQ 編解碼後還原求得。RAM 是以 Altera 所提供的 LPM megafunction 所

產生，所以是使用 memory bits 來做為面積複雜度的量測標準。

Subspace		**No**	**Yes**			
l		0	0	2	4	6
PSNR	Girl	30.2219	30.0063	30.0036	29.9647	29.8685
	Baby	26.9533	26.6825	26.6782	26.6729	26.5793
	House	25.9510	25.6674	25.6667	25.6648	25.6115
神經元集合 memory bits		131072	61440	53248	45056	36864

表 4-1　有限精度計算對影像品質與 RAM 大小的影響

從表 4-1 中我們可以觀察到與基本全搜尋編碼程序相較之下，使用子空間部分距離搜尋進行編碼程序時 PSNR 值只會輕微地降低約 0.25 dB，但是神經元集合 RAM 的面積複雜度卻可以大幅度的降低（從 131072 memory bits 降到 61440 memory bits）。另外採用有限精度計算的技巧可以更進一步的降低儲存空間，當權重向量係數忽略 6 個 LSB 位元平面的精度時，只要付出 PSNR 值降低約 0.5 dB 的代價便可降低面積複雜度約 60%（從 61440 memory bits 降到 36864 memory bits）。

表 4-2 顯示了不同的δ與l 對 VSDC 單元面積複雜度的影響。從中可觀察到，雖然δ值較高的 VSDC 單元可以有較高的產量，但是它

所需的面積複雜度卻是呈倍數成長。尤其是當$\delta=16$、$l=0$時，硬體實現所需的面積複雜度為 1212 個 LEs，雖然這樣的 VSDC 單元在不用做部分距離累加的情況下可以一次就就算出$D\left(X_{Lm},Y_{Lm}^{j}\right)$的值$(m=2)$，但是它卻需花費較多的 LEs，也因此$\delta$值較高的 VSDC 單元在硬體實現時比較不合乎效益。相對地，較小的δ值(i.e., $\delta=4$)所需要的面積複雜度也較低。因此在本研究實現的架構裡，採用的是一次計算四個係數($\delta=4$)搭配位元平面縮減程度$l=6$的 VSDC 單元，僅使用了 170 個 LEs，與一次計算 16 個係數($\delta=16$)搭配位元平面縮減程度$l=0$的 VSDC 單元相較之下只用了 14%左右的比重。

	l	**0**	**2**	**4**	**6**
	1	46	40	34	28
	2	123	107	92	75
δ	**4**	278	242	206	170
	8	589	513	437	361
	16	1212	1056	900	744

表 4-2　不同參數對 VSDC 單元面積複雜度的關係表

表 4-3 觀察不同的δ值對於 VLSI 架構的 Latency（$\mathcal{L}$）的影響。令$\mathcal{T}(\mathrm{x})$代表對一個輸入向量x進行 3.6 節討論到的子空間部分距離

搜尋運算所需花費的平均時脈週期數，由於執行 Step1 耗費的時間與外部輸出入匯流排有所關聯，所以 Step1 這部份花費的時間並沒有計算在內。所以 Latency（$\mathcal{L}$）被定義如(4.1)式所示：

$$\mathcal{L} = \frac{1}{NW}\sum_{j=1}^{w} \mathcal{T}\left(x^{j}\right) \tag{4.1}$$

其中W代表的是輸入的訓練向量個數，N代表的是設計集中的神經元數目。對每一個訓練向量而言，比較每一個神經元所花費的平均時脈週期數記做 L。在這個實驗裡面，權重向量的個數為 1024 (i.e., $N=1024$)，共有 20480 個訓練向量(i.e., $w=20480$)。向量 $\mathbf{x}$ 和 $\mathbf{y}^{j}$ 的維度是 8×8 (i.e., $n=3$)，而 $\mathbf{x}_{Lm}$ 和 $\mathbf{y}_{Lm}^{j}$ 的維度則是 4×4 (i.e., $m=2$)。

δ	**1**	**2**	**4**	**8**	**16**
$\mathcal{L}$	1.4836	1.2096	1.0864	1.0307	1

表 4-3　Latency：比較單一神經元平均時脈週期數

從表 4-3 中可觀察到搭配多係數部分距離累積技巧的部分距離搜尋硬體實現，在使用不同的參數δ時對系統效能的影響。平均而言，判斷一個神經元是否需淘汰，在參數δ值些微大於 1 時所花費的 Latency 比 1 大但接近於 1。另外從表 4-2 和表 4-3 也可以觀察到搭配

較小參數δ和較大參數l的子空間部分距離搜尋演算法可以降低硬體實現的面積複雜度。因此我們所提出具有低面積複雜度和高產量的子空間部分距離搜尋硬體電路，可以達到計算平方距離和判斷神經元是否需被淘汰所需耗費的 Latency 接近於 1 的程度。

對於我們所提出的排序電路，其 k 值之於面積複雜度的關係可從表 4-4 獲得。從表中可觀察到，我們所提出的排序電路具有低面積複雜度的特性，且面積複雜度的成長率僅隨 k 值線性增加。加之其可在 2 個 clock cycles 的時間內完成排序動作，這些事實都將有助於 k 值很大的 k-WTA 設計。

k	**5**	**8**	**16**	**32**	**64**	**128**	**256**	**1024**
LEs	113	167	311	599	1175	2327	4631	18455

表 4-4　排序電路在不同 k 值下的面積複雜度(以 LEs 表示)

接下來要探討的是本論文提出架構的整體效能。如前所述，在硬體實現時，神經元集合內儲存的是各個神經元所對應的權重向量。在我們以下的實驗裡，神經元的數目共 1024 個，而這些神經元對應的權重向量，是由 Baboo、Bridge 兩張 512×512 的灰階圖經過計算所得來(i.e., $N=1024$)。權重向量的維度是 8×8 (i.e., $n=3$)，並以位元寬度 9 來表示(i.e., $b=9$)。分析 RAM 中所儲存的$Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$，其中

$m=2$，也就是說Y_{Lm}^{j}的維度是 4×4。有限精度計算的參數$l=6$，VSDC 運算單元一次計算四個係數(i.e., $\delta=4$)。

表 4-5 條列了 PDS 在 Pentium IV 處理器上以軟體實現，和本論文提出的架構在 Nios 處理器上以硬體實現所需的執行時間數據。實驗時共輸入了 20480 個訓練向量(i.e., $w=20480$)，訓練向量的維度亦為 4×4，且使用的是 5-WTA ($k=5$)。

CPU type	**Pentium IV 3.0 GHz**		**NIOS II 50 MHz**
Algorithm	Full Search without PDS	Subspace PDS	Subspace PDS
Implementation	Software	Software	Hardware/Software Codesign
CPU Time (μSec)	20877.41	350.93	86.19

表 4-5　軟硬體 *k*CL 效能比較

此處執行時間定義為對每個訓練向量完成 *k*CL 整個過程所需花費的平均 CPU 時間(uS)。在表 4-5 中，Pentium IV 處理器相關呈現的數據包含了兩種純軟體搜尋的方式，一種是單純的 *k*CL 演算法則，沒有使用子空間搜尋或部分距離搜尋法則；另一種則是在小波域上做子空間的部分距離搜尋。軟體實現法在進行權重向量調整時用的是原始除法，而非查表式除法。此外，因為訓練向量的個數眾多，為了避

免神經元的學習收斂速度過快，我們將 *r* 值的累加速度放慢，也就是只有在神經元獲勝 8 次後，才把 *r* 值加 1。

從表 4-5 中可以看到，在 Nios 處理器內嵌部分距離搜尋硬體電路上的執行效果，比在 Pentium IV 處理器上以純軟體的方式實現相同演算法則來的好。一般基本的 *k*CL 演算法則在 Pentium IV 3.0GHz 處理器上以純軟體的方式實現，平均執行時間要 20877*uS* 左右，使用子空間部分距離搜尋法則的 *k*CL 實現，則可將平均執行時間降低到 351*uS* 左右；而我們以 Nios 50MHz 處理器實現子空間部分距離搜尋 *k*CL 法則卻僅需要 86*uS*，執行時間等同是採用子空間部分距離搜尋法則之 Pentium IV 3.0GHz 處理器的 24.5%左右。此外，系統中的神經元數目高達 1024 個，整個系統所需的 LEs 卻只要 23431 個，約等於發展板最大容量的 56.8%。

我們也做了一些實驗來探討查表式除法對於學習結果的影響。若我們分別以“House”、“Lena”、“Tree”等三張 512×512 的灰階圖當訓練向量，輸入使用原始除法及查表式除法兩種 *k*CL 來訓練神經元，然後將訓練結果拿去當作 VQ 編碼器的碼簿(codebook)，並且對原始圖檔做 VQ 編碼再求其 PSNR 值，則我們可以從表 4-6 獲得查表式除法對學習結果的影響程度。

Image name	Division	Lookup-table based division
House	26.3675	26.3656
Lena	24.2825	24.2814
Tree	21.5550	21.5478

表 4-6　原始與查表式除法對學習結果的影響比較(以 PSNR 表示)

表 4-6 實驗用的神經元是由“Baboo”、 “Bridge”兩張 512x512 的灰階圖經過計算所得來，數量是 64 個。我們可以發現，查表式除法對於學習結果的影響幾乎可以被忽略，且整個查詢表僅需 58 LEs，以此少許的面積複雜度來換取除法大幅的計算複雜度是相當划算的。

由以上的各項實驗結果來看，本論文提出的架構對於硬體實現非常的有效益，不論是在效能或面積複雜度上都有優異的表現。

第五章 結論

本論文所提出的架構，對於 k 贏家通吃競爭式學習法的硬體實現具有下列三樣優點：低面積複雜度、高產量和高可調性。

使用子空間搜尋和有限精度計算兩項技巧，可以大量降低實現神經元集合 RAM 和 VSDC 運算單元的面積複雜度，整個硬體架構中最佔面積複雜度的也正好是屬於記憶體的神經元集合 RAM 和擁有大量數學計算的 VSDC 運算單元。使用多係數累積的技巧，可以降低部分距離計算的 Latency，事實上當 $\delta = 4$ 時，搜尋一個神經元的平均 Latency 幾乎接近只要一個時脈週期。我們也使用了查表式除法來大幅降低調整權重向量的計算延遲，但其對於更新結果的影響卻幾乎可被忽略。

實際的效能測試顯示出我們提出的架構內嵌於 Nios 50MHz 的處理器上時執行效能優於沒有硬體支援的 Pentium IV 處理器。

參考文獻

[1] Altera Corporation (2005). *Stratix Device Handbook*, http://www.altera.com/literature/lit-stx.jsp

[2] Altera Corporation (2002). *Custom Instructions for NIOS Embedded Processors*, Application Notes 188. http://www.altera.com/literature/lit-nio.jsp

[3] C.D. Bei and R.M. Gray, "An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization." *IEEE Trans. Communication*, Vol. COM-33, pp.1132-1133, Oct. 1985.

[4] Bondalapati K. and Prasanna V.K., "Reconfigurable computing systems," *Proceedings of the IEEE*, Vol. 90, pp.1201-1217, 2002.

[5] M. D. Ciletti, "*Advanced Digital Design with the Verilog HDL*," Prentice Hall, 2003

[6] Colavita A. A., Cicuttin A., Fratnik F., Capello G. SORTCHIP, "A VLSI Implementation of a Hardware Algorithm for Continuous Data Sorting," *IEEE Journal of Solid-State Circuits*, Vol. 38, pp 1076-1079, 2003.

[7] Cole R. and Seigel A. R., "Optimal VLSI circuit for sorting," *Journal of ACM*, Vol. 35, pp.777–809, 1998.

[8] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*. Kluwer, Norwood, Massachusetts, 1992.

[9] Grossberg S, "Competitive Learning: From Interactive Activation to Adaptive Response," *Cognitive Science*, Vol. 11, pp.23-63, 1987.

[10] Haykin S., "*Neural Networks: A Comprehensive Foundation*," Prentice Hall: Engle Cliffs, NJ, 1998.

[11] Hertz J., Krogh A., Palmer R.G., "*Introduction to the Theory of Neural Computation*," Addison-Wesley: New York, NY, 1991.

[12] Hwang W. J., Lin F.J., Zeng Y.C., "Fast Design Algorithm for Competitive Learning," *Electronics Letters*, Vol.33, pp.1469-1470, 1997.

[13] Hwang W. J., Ye B. Y., Lin C. T., "A Novel Competitive Learning Algorithm for the Parametric Classification with Gaussian Distributions," *Pattern Recognition Letters*, Vol.21, pp.375-380, 2000.

[14] Hofmann T., Buhmann J.M., "Competitive Learning Algorithm for Robust Vector Quantization," *IEEE Trans. Signal Processing*, Vol. 46, pp. 1665-1675, 1998.

[15] Kohonen T., "The self-organizing map," *Proc. IEEE*, Vol. 78, pp.1464-1480, 1990.

[16] Park H., Prasanna V. K., "Modular VLSI architectures for Real-Time Full-Search-Based Vector Quantization," *IEEE Trans. Circuits Syst. Video Technology*, Vol.3, pp.309-317, 1993.

[17] Vetterli M. and Kovacevic J., "*Wavelets and Subband Coding*," Prentice Hall: Engle Cliffs, NJ, 1995.

[18] Wang, C.L.; Chen, L.M. A New VLSI Architecture for Full-Search Vector Quantization, *IEEE Trans. Circuits and Systems for Video Technology* 1996,Vol. 6, pp.389-398.

[19] Wolfe W. J., Mathis D., Anderson C., Rothman J., Gottler M., Brady G., Walker R., Duane G., "K-Winner Networks," *IEEE Trans. Neural Networks*, Vol. 2, pp.310-315, 1991.

[20] Xu L., Krzyzak A., Oja A. E., "Rival penalized competitive learning for clustering analysis, RBFnet, and curve detection," *IEEE Trans. Neural Networks*, Vol. 4, pp.636-649, 1993.