國立臺灣師範大學
資訊工程研究所碩士論文

在NoC上實現OSort演算法之硬體架構設計
Hardware Implementation for Spike Sorting Based on NoC with OSort Algorithm

指導教授：　黃　文　吉　博士

研究生：　徐　雅　姿　撰

中華民國　一零三　年　七　月

摘要

本論文針對棘波分類法則實現一套硬體架構，以提供大量資料快速運算。棘波排序是研究生物大腦及發展腦機介面(BMI, Brain Machine Interface)的基礎，棘波排序主要分為三個步驟：棘波偵測、特徵擷取以及分群，棘波分類則包含特徵擷取與分群部分,本論文採用OSort演算法將棘波偵測所採集到的數位訊號進行分類。

OSort演算法是一個模板比對（Template Matching）的非監督式分群法則，與其他棘波分類演算法相比，如PCA和K-means、小波轉換和SPC、GHA和FCM等等，不需做複雜的降低資料維度（Dimensionality Reduction）運算，更適用於即時分類；不用以分類指標（Cluster Validity Index）來計算最佳群集數，即可自動決定群集個數；不若於其他演算法需要一段時間的離線訓練（Offline Training），可立刻獲得棘波分類結果。

本論文保留OSort演算法核心的概念，簡化較消耗硬體資源的設計，並適用於多數的採樣數據做高速運算，透過Altera公司的系統開發工具使用NoC（Network on Chip）架構，實現並驗證於現場可程式化邏輯閘（FPGA, Field Programmable Gate Array）上，實驗結果證明，本論文所提出的棘波分類硬體架構具有一定精確度及高速運算的優點。

關鍵字： 可程式化系統晶片、棘波分類、OSort、FPGA、NoC

致謝

在師大資工求學的這兩年時間，感謝我的指導教授黃文吉博士在研究上給我很大的空間，讓我培養獨自解決問題的能力和學習待人處事的道理，並在我遇到難題時引導我和不厭其煩地解說，將課堂上的知識學以致用，也感謝師大提供良好的學習環境和豐沛的器材資源，使我的學習更有效率。在此特別感謝國立台北科技大學資工系的楊士萱教授、國立中正大學資工系的熊博安教授以及健行科技大學電子系的歐謙敏教授，撥冗來參加我的碩士學位口試，並給予實質的建議與回饋。

很高興能來到多媒體通訊暨系統晶片實驗室與大家一起學習成長，度過美好的時光，感謝已畢業學長、同學和學弟：（網頁上排序）建廷、國璿、清志、聖穎、翰逸、任軒、思淮、一修、皓棠、光耀、奇恩、烝祺、建旻、元品、承祐、信豪、柏佑和映綸，不論是學業或生活上的幫忙與支援。也感謝師大資工系排的學弟妹們，讓我在忙於課業之餘，能有打球放鬆的空間，很高興認識你們！

最後，要謝謝我的爸媽一路栽培我到碩士畢業，讓我無後顧之憂地讀書，以及關心我的朋友們，謝謝你們耐心聆聽我的情緒，給我支持、關懷與鼓勵，適時地讓我轉換心境再重新出發，沒有你們生活不會如此精采。很開心自己完成一個人生的重要里程碑，不論成果好壞，我將會帶著這些寶貴經驗，繼續前往下一個旅程。

目錄

附圖目錄

附表目錄

第一章 緒論

1.1 研究背景與動機

人類的大腦是一個產生意識、思想、情感和行為的器官，由神經元、膠質細胞、上皮細胞和血管所組成，神經元（Neuron）被認為是大腦的重要成分[1]，腦部包含大約 1000 億個神經元，每個神經元可與其餘神經元產生上千種不同的連結。

多數的神經元發出動作電位（Action Potential）來彼此溝通，動作電位在研究領域中又被稱為棘波（Spike），為一連串的離子通過細胞膜後，離子重新分配所造成的電位差，產生短暫且特殊的波形，持續的時間約為 1~4 毫秒(ms)[2]。透過分析和區別動作電位，也就是棘波排序（Spike Sorting），我們才得以更進一步的解密人類大腦[3]。

棘波排序是設計腦機介面（BMI, Brain Machine Interface）所需的步驟之一[4]，腦機介面是利用腦波來與外部設備溝通的介面，自 1970 年代，由UCLA開始研究至今，從動物實驗到人體實驗，應用的領域包含醫療、運動訓練、遊戲、教育等等，雖然腦機介面不斷地改良和推陳出新，但由於生物大腦是一個極為複雜的系統，我們並未全面了解它的奧秘，腦機介面的發展也還未達到顛峰，仍尚有進步

的空間。

在進行棘波排序前，我們可以透過三種方式獲取腦部的訊號，分為侵入式、半侵入式和非侵入式。侵入式方法為透過直接在大腦的灰質植入一個為電極（Microelectrode）或微電極陣列（MEA, Microelectrode Array），蒐集周圍神經元所發出的訊號，可獲取較強的棘波訊號，但容易引發免疫反應和愈傷組織，半侵入式與侵入式的不同之處在於將微電極植入在頭殼與灰質之間，雖然獲得較弱的訊號，但不會發生免疫反應和愈傷組織，此兩種方法皆需動手術才能使用，目前僅應用在生物和醫學研究上，以及幫助少數腦部正常身體卻癱瘓的病人，如：透過腦機介面操作機械手臂維持日常生活。

非侵入式的方法為早期在頭皮表層接上電極並塗抹導電膠來強化訊號，經過增幅氣放大訊號後，蒐集所有傳至頭皮的訊號，以電極數量的多寡可分為腦電波網（EEG Net）或腦電波帽(EEG Cap)等等，直到 2008 年，美國NeuroSky公司開發出第一款不需塗抹導電膠的乾電極腦波偵測器，並且可以過濾掉周圍雜音、電器設備、眼動雜訊及交流電訊號的干擾[5],腦機介面才開始獲得主流媒體的關注，最廣為人知的例子是與NeuroSky合作的日本公司Neurowear所推出的「貓耳朵」Necomimi。

雖然非侵入式的方法不需動手術，較符合大眾使用，但所蒐集到的訊號從神經元傳至頭皮表層已經相當細微且混合，以目前的設備和技術，從訊號中獲得的控制資訊相當有限，無法進行複雜的操作，也直接限制腦機介面的應用範圍，所以腦電圖較常拿來診斷病人是否罹患疾病，如：檢視棘波和判斷腦中是否異常放電，來確認病人是否罹患癲癇。

註：腦電圖（EEG）中的棘波與棘波（動作電位）意義不同，腦電圖是紀錄由數以千計的神經元放電所造成的細胞外場電位變化，體表測得的電位，是多處的總和，此棘波代表的意義為腦部異常放電所產生的訊號高凸現象，好發於癲癇病人腦中。

但不管透過何種方式獲取腦部訊號，首要面對的問題便是大量的雜訊干擾，由於體內、外雜訊與周圍神經元所發出的訊號干擾，直接影響棘波分類的正確率，增加區分不同棘波的困難度，再者數小時蒐集到的資料量往往非常的龐大，在個人電腦上運算也要耗時數小時甚至一天的時間，隨著腦機介面的發展，即時分類也成為必要的條件之一，在講求正確率與效率的情況下，使得棘波分類被視為一項不容易的工作。這也是 1929 年Hans Berger發明腦電圖和 1958 年Strumwasser發明微電極陣列後，經過數十年，還有如此多研究人員投入研究的原因。

鑒於此，一個理想的棘波排序系統必須具備高正確率和即時處理大量資料的能力。棘波分類主要分為三個步驟，分別為棘波偵測、特徵擷取和分群[6]，棘波分類包含特徵擷取和分群部分，本論文將專注棘波分類的研究，並使用微電極蒐集的訊號和模擬訊號作為測試數據。

常用的特徵擷取方法為主成分分析（PCA, Principle Components Analysis）[7]和小波轉換（Wavelet Transform）。PCA是一種簡化數據、降低資料維度的方法，透過對共變異數矩陣進行特徵分解，來得出數據的主成分（即特徵向量）與權值（及特徵值），此方法雖然能有效降低數據維度，但計算過程較於複雜。GHA（Generalized Hebbian Algorithm）[8]為PCA的變形，藉由突觸權重向量w依公式不停迭代計算後，w將趨近於輸入向量x的特徵值， x與w的內積為特徵擷取的結果。

小波轉換則有很多種方式，其概念為將資料由時域轉為頻域，較簡單的方法為Haar小波轉換，運算步驟分為水平與垂直分割，完成第一次水平切割與第一次垂直切割，則做完第一階的小波轉換，得到四個頻帶LL、LH、HL、HH，再對LL頻帶做一次水平與垂直切割，則完成第二階的小波轉換，第三階則以此類推，做完n皆小波轉換後，LL頻帶為最低頻的部分也就是特徵擷取的結果，在 2004 年 Quiroga et al.所提出的Waveclus中做了四階Haar小波轉換[9]。

以上簡要提出三種特徵擷取的方法，這些法則皆可搭配不同的分群演算法成為一套特有的棘波分類系統，最基本的分群演算法為K-means分群法也可稱為C-means分群法，從數據集中任選k個點作為群集的中心，將距離群集最近的數據歸類到此群集，再重新計算群集的中心，重複上述步驟即完成分群，此法則簡單快速，但缺點為對初始值敏感，對於不同的初始值k，可能導致差異極大的分群結果，在雜訊高和有異常點（Outlier）的情況下分群效果也不佳。

FCM（Fuzzy C-means）是根據C-means分群法衍生而來的法則，加入模糊理論的概念，使得數據不絕對的屬於任何群集而是以介於 0-1 之間的值來表示隸屬程度，期望提升分群的效果。其他還有SPC（Superparamagnetic Clustering）[9]、Bayesian Classification方法等等，這些法則雖然擁有一定的分群效果，但都有共同的缺點——必須指定分群數目，由於我們無法明確地知道微電極蒐集到的棘波共有幾種，為了增加分群的準確率，需額外利用分類指標(Cluster Validity Index)來計算最佳群集數，如以FCM為主的分類指標法則[10]。

上述的棘波分類法則，不論是PCA和K-means、小波轉換和SPC、GHA和FCM的搭配，皆需一段時間的離線訓練（Offline Training），雖有不錯的棘波分類精確度，但較難應用於即時分類，不適合用於需對棘波立即反應的腦機介面上。

1.2 研究目的

OSort演算法是廣為使用且公開資源的非監督式棘波分類法則，可以自動決定群集的數目，不需要一段時間的離線訓練（Offline Training），即可立即獲得棘波分類的結果，非監督式演算法也能去除監督式演算法需設定參數所花費的時間，運算速度較其他演算法快且能夠即時分類為其主要優勢[11]，表 1.1 為OSort演算法與其他棘波分類演算法之比較。

	PCA+K-means	WT+SPC	GHA+FCM	OSort
可用於即時分類	否	否	否	是
需離線訓練	是	是	是	否
需指定群集數量	是	是	是	否

表 1.1　棘波分類演算法之比較

但在使用多個微電極採樣數據，也就是多通道（Multi-Channel）的情況下，大量的數據讓OSort演算法的效率顯得不足。硬體化棘波分類演算法可以讓腦機介面有更好的效能與應用，也能加速大量離線資料的分類，所以硬體化OSort演算法有其必要性。雖然OSort演算法的概念簡單，本論文將於下一章詳細介紹，但實作硬體將會遇到三大挑戰，第一為硬體資源的分配，根據OSort演算法，群集（Cluster）的數目會不斷地增加或減少，硬體卻不擅長處理資源會擴增和削減的演算法，其二為電路Controller的設計，OSort演算法為一個有限狀態機（FSM, Finite State

Machine）不同狀態的切換跟Controller有密切的關係，如何解決複雜的資料流動，讓電路發揮最好的效能，最後則為執行效能的部分，為本論文硬體實現的主要目標，電路一段時間內所能處理的資料量（Throughput）是重要的量測目標，如何讓單通道至多通道蒐集的大量資料能夠即時處理是本論文的研究核心。

目前有許多新的棘波分類系統持續地被提出，因此在研究領域中能快速實作並驗證新的法則也是必要的，而ASIC（Application-Specific Integrated Circuit）一經生產便難以修改的特性，使得ASIC較難應用於棘波分類的研究領域。另外高NRE（Non-Recurring Engineering） Cost以及需要較長的時間設計與驗證，現場可程式化邏輯閘（FPGA, Field Programmable Gate Array）相對於ASIC顯得更佔優勢，可提供棘波分類系統在未來面臨修改或延伸時較高的彈性，所以本論文採用FPGA為硬體架構驗證和測試的平台。

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與動機、研究目的及本文架構。

【第二章】基礎理論與文獻探討

簡介棘波排序的理論基礎、本論文所使用之演算法及FPGA的技術背景。

【第三章】系統架構

介紹所提出的OSort演算法之電路架構，並提供電路細部的討論與說明。

【第四章】實驗數據與效能分析

包含了實驗環境的說明、相關實驗數據分析以及軟硬體效能比較。

【第五章】結論

對於所提出之硬體架構以及實驗的結果進行總結。

第二章 基礎理論與文獻探討

2.1 棘波排序（Spike Sorting）

棘波排序是一項分析電生理學（Electrophysiological）資料的技術，將一個或多個微電極所蒐集到的混合訊號——棘波序列（Spike Train），在雜訊干擾的背景下，將不同神經元所發出的訊號加以區分。依照訊號之波形作為分類的準則，傳統的棘波排序包含三個步驟：棘波偵測（Spike Detection）、特徵擷取（Feature Extraction）以及分群（Clustering），每個步驟有相同的重要性，直接影響棘波排序的結果。

詳細的流程圖如圖 2.1，在採樣頻率 20k-30kHz下，將微電極蒐集到的訊號，通過增幅器再經由ADC(Analog-to-Digital Conversion)將類比訊號轉為數位訊號，此訊號即為原始數據（Raw Data），先將原始數據通過代通濾波器，通常為300Hz-3000Hz，才做為棘波偵測的輸入；棘波偵測的方法分為振福偵測(Amplitude ）和能量計算（Energy）方式，再藉由制定閥值（Threshold）以及相關運算來偵測棘波，著名的法則有NEO（Nonlinear Energy Operator）[12]和Matched Filter[13]等等。

偵測到棘波後，透過對齊（Alignment）的動作，以峰值最高或最低點為基準，

來獲取完整的棘波，以 64 個資料點表示一個棘波為例，每個棘波的最高點或最低點會在相同的資料點位子上，透過對齊可以提高棘波分類的準確度。

為了分析棘波，經由特徵擷取降低數據維度，以特徵值分析用最少的資訊來表示棘波，減少分群演算法的計算時間和複雜度，常用的方法有PCA、GHA和小波轉換等等，再將數據依據距離的度量分成若干群，將相似波形的棘波分為一群，常用的方法有K-means、FCM、SPC等等。透過上述這些法則，我們可以建立一套完整的棘波排序系統。

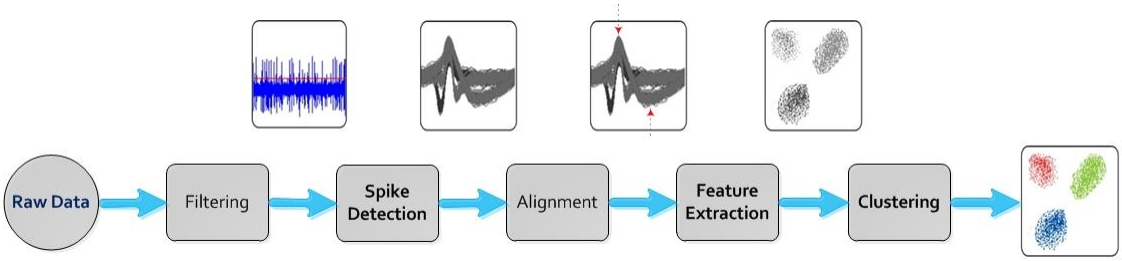


圖 2.1　棘波排序流程圖

2.2　OSort（Online Sorting）

OSort是Rutishauser在 2006 年針對棘波分類所提出的演算法[14]，目的在於提供即時分類，立即得知分類結果，不同於以往需分為特徵擷取與分群兩個部分，以模板比對（Template Matching）的方式，沒有複雜的計算式，用簡單的運算實作整個流程，如圖 2.2 所示。

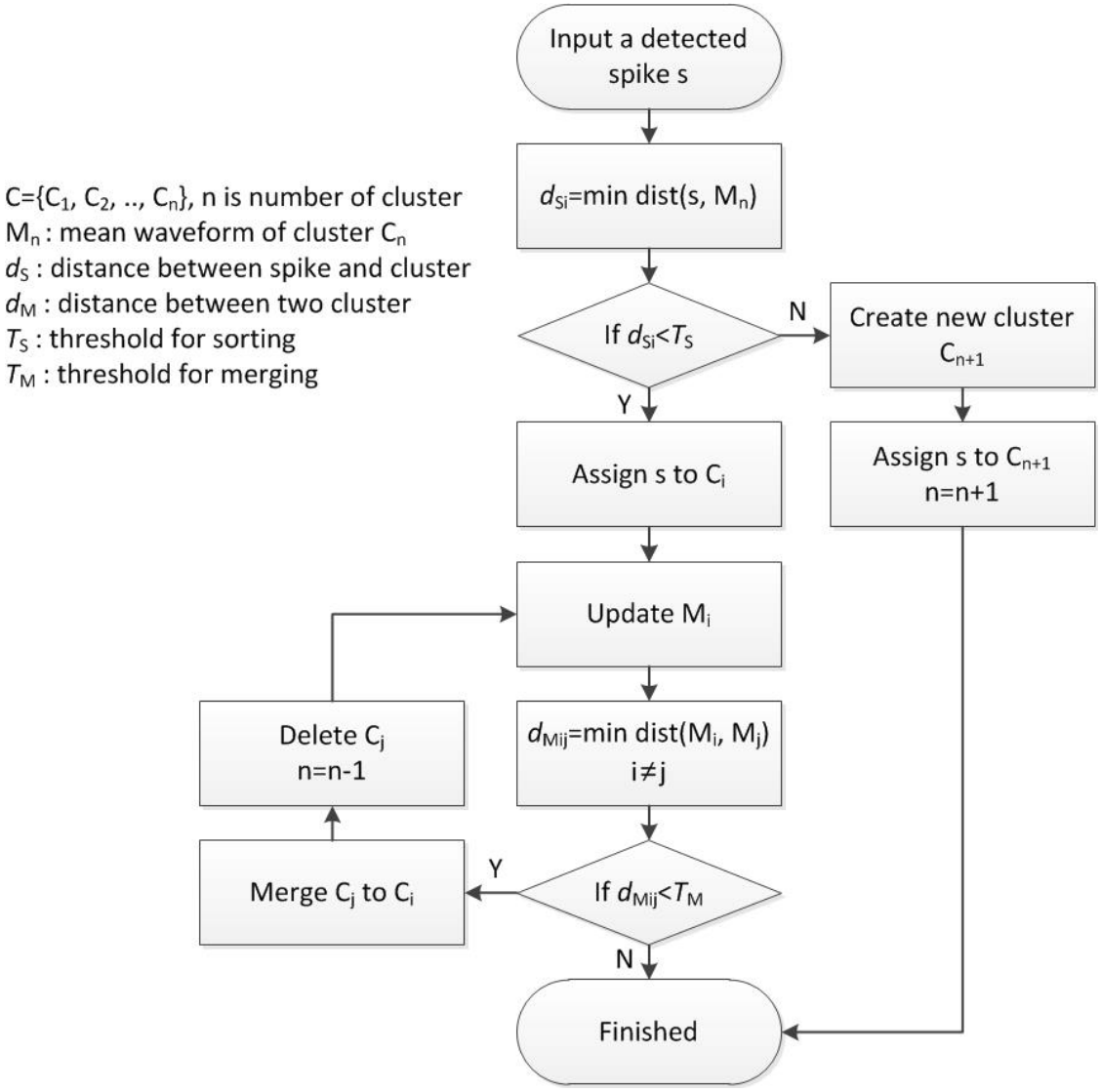


圖 2.2　Osort演算法流程圖

首先，先進行閥值的計算，T_S作為排序（Sorting）之用，而T_M則作為合併

（Merging）之用，Rutishauser提出了兩種計算閥值的方法，分別為近似法和估算法：

（1） 近似法（Approximated Method）：T_S=T_M=T，T的計算方法如公式(2.1)，σ_r為通過濾波器之信號的平均標準差，N為一個棘波的採樣點數量。

$$T = N(\sigma_r)^2 \tag{2.1}$$

（2） 估算法（Estimation Method）：需用到卡方分配計算T_S和T_M，有較高的正確率，因其公式較為複雜，故在此省略，有興趣讀者可參考原始論文。

C為群集之集合，假設目前存在n個群集，C={C_1, C_2, .., C_n}，每個群集的平均值計為M_i, i=1,…, n，在輸入一個偵測到的棘波s，計算s與所有群集C_i之平均值M_i的距離，如公式(2.2)，找出其中最小的距離d_{Si}和相對應的群集i，若最小的距離d_{Si}大於T_S，表示此棘波s與目前存在之群集較無相似度，則需建立一個新的群集，將棘波s分類到新的群集；若d_{Si}小於T_S，表示此棘波s與群集C_i較為相似，則將棘波s分類到最短距離的群集C_i。

$$d_s(\mathrm{s}, \mathrm{M}_i) = \sum_{k=1}^{N} (\mathrm{s}(k) - \mathrm{M}_i(k))^2 \tag{2.2}$$

$$d_{\mathrm{M}}(\mathrm{M}_i, \mathrm{M}_j) = \sum_{k=1}^{N} (\mathrm{M}_i(k) - \mathrm{M}_j(k))^2 \tag{2.3}$$

將s與C_i中的數據重新計算群集的平均值M_i後，再次計算此群集C_i與其餘群集C_j之平均值M_j的距離，如公式(2.3)，若最小的距離d_{Mij}小於T_M，表示此兩個群集已趨於相似，則將兩群合併（Merging），重複此步驟，直到此群集C_i與其餘群集C_j

之距離皆大於T_M，則棘波分類結束，即為OSort演算法之結果。

從上述流程可知OSort演算法不需設定群集個數，可以自動將數據分為若干群，由於棘波偵測法則可能無法達到百分之百的正確率，所以過濾偵測錯誤的棘波和Outlier的方法，可以將存在較少棘波數量的群集刪除，直接視為雜訊（Noise）。

2.3 FPGA 系統設計

自 1958 年美國德州儀器的Jack Kilby成功生產第一個積體電路（IC, Integrated Circuit）也就是將電晶體、電阻和電容放在一個半導體晶片上，在這數十年間IC的產品已經無處不在且發展越來越快，所要求的功能也趨於複雜，ASIC即是一種符合不同需求的特殊應用積體電路，它的優點是系統體積小、消耗功率低、保密性強和在大批量應用時，可顯著降低成本，但它的缺點也是開發時間長，一旦程式化即不能更改，事後發現設計錯誤會造成極大的成本損失，所以設計師需要一個易於使用、靈活的設計環境來因應不同設計實現，以成本、性能指標和功率消耗為考量下，在 1984 年由Xilinx首先推出FPGA，是一種可程式化系統晶片（SOPC, System-on-a-Programmable-Chip），也有人稱作可編成的ASIC。

對於一些需求量較小或處於開發階段的產品，FPGA無疑是最好的選擇，使用硬體描述語言(HDL, Hardware Description Language)搭配FPGA開發板設計數位電路，縮短研發時間且更有彈性，經過簡單的繞線佈局合成電路，可快速重覆地將系統燒錄至FPGA上進行測試。在摩爾定律下，FPGA晶片所能提供的邏輯單元數目不斷增大，所面臨的便是更複雜及龐大的系統，如何整合系統、讓每個元件快速地運作並相互溝通是一大挑戰。

Altera公司最早在他們開發的軟體Quartus II中，推出系統整合開發工具SOPC Builder，提升設計師開發的方便性，Qsys為新一代的系統整合開發工具採用單晶片網路系統（NoC, Network-on-a-Chip）架構，與SOPC Builder相比擁有更快的時序收斂、更快的完成驗程以及改進一些系統設計加速開發。NoC是利用區域網路的路由節點結構，將所有元件連結起來，透過類似於TCP/IP協定的封包形式傳輸。

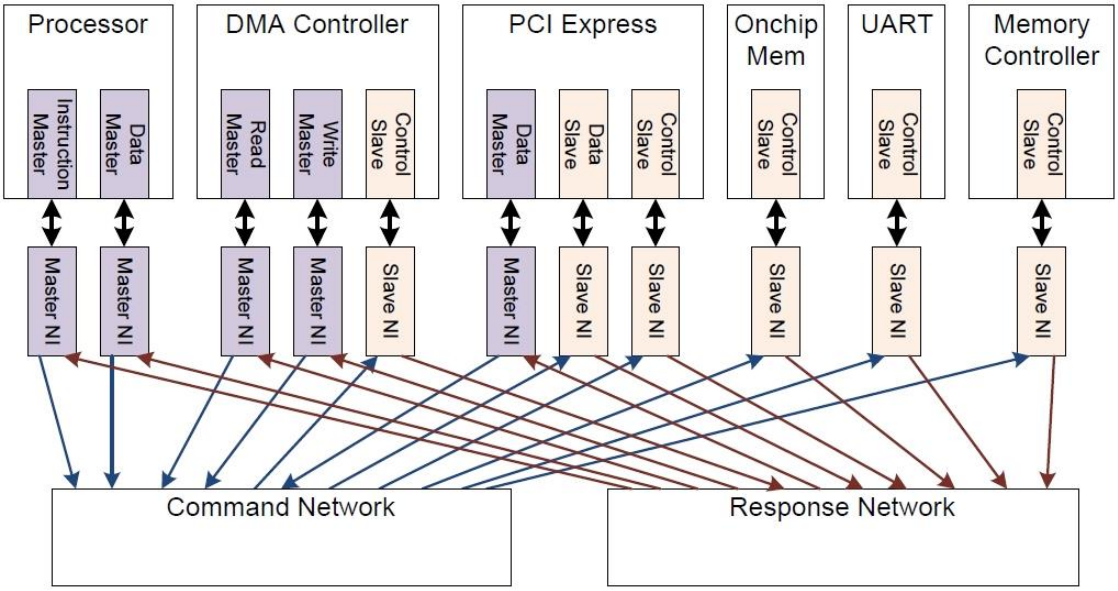


圖 2.3　NoC系統架構圖

圖 2.3 為NoC的系統架構圖，所有元件經由NI（Network Interface）來進行溝通，各個元件如：Processor、DMA Controller、PCI Express等等執行完程序後，只需將處理完的資料傳送給NI，不需考慮資料傳輸的流程，傳輸方式為Master Interface或Slave Interface會根據傳輸層（Transport Layer）協議，透過命令網路（Command Network）和回應網路（Response Network）拿取和傳送封包，再透

過NI將封包轉換成資料，傳遞給需要的原件，以此實作交易層（Transaction Layer），如圖 2.4 所示。

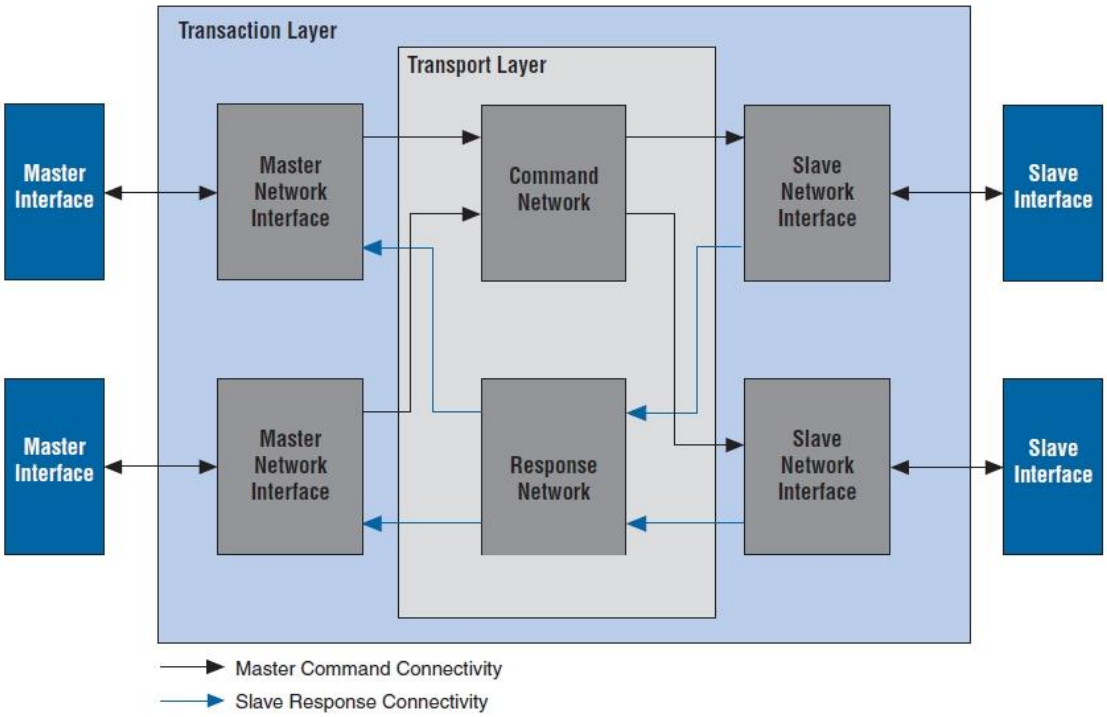


圖 2.4　Master Interface與Slave Interface的傳輸架構

透過Qsys開發工具，使我們不需擁有廣泛的Noc知識，也能使用Noc架構創造一個高性能的系統，再結合Altera公司所提供的NIOS II IDE（Integrated Development Environment），發展出一套軟硬體共同設計的流程，如圖 2.4 所示。我們將設計好的客製化電路，使用Qsys建立專案，透過GUI介面來設定所需的系統元件，如CPU、DMA Controller、On-chip Memory等等，再加入電路專案並重新編譯後，將Qsys專案燒錄於FPGA開發板，以上為硬體設計的部分。

軟體部分使用NIOS II IDE，首先我們必須建立一個NIOS II專案來與FPGA開

發板做連接，設定完系統參數後，撰寫C語言程式碼，並利用所提供的函式庫來與FPGA開發板也就是我們所設計的電路溝通,進行實質的訊號輸入及輸出測試，以驗證系統架構的正確性與效能。

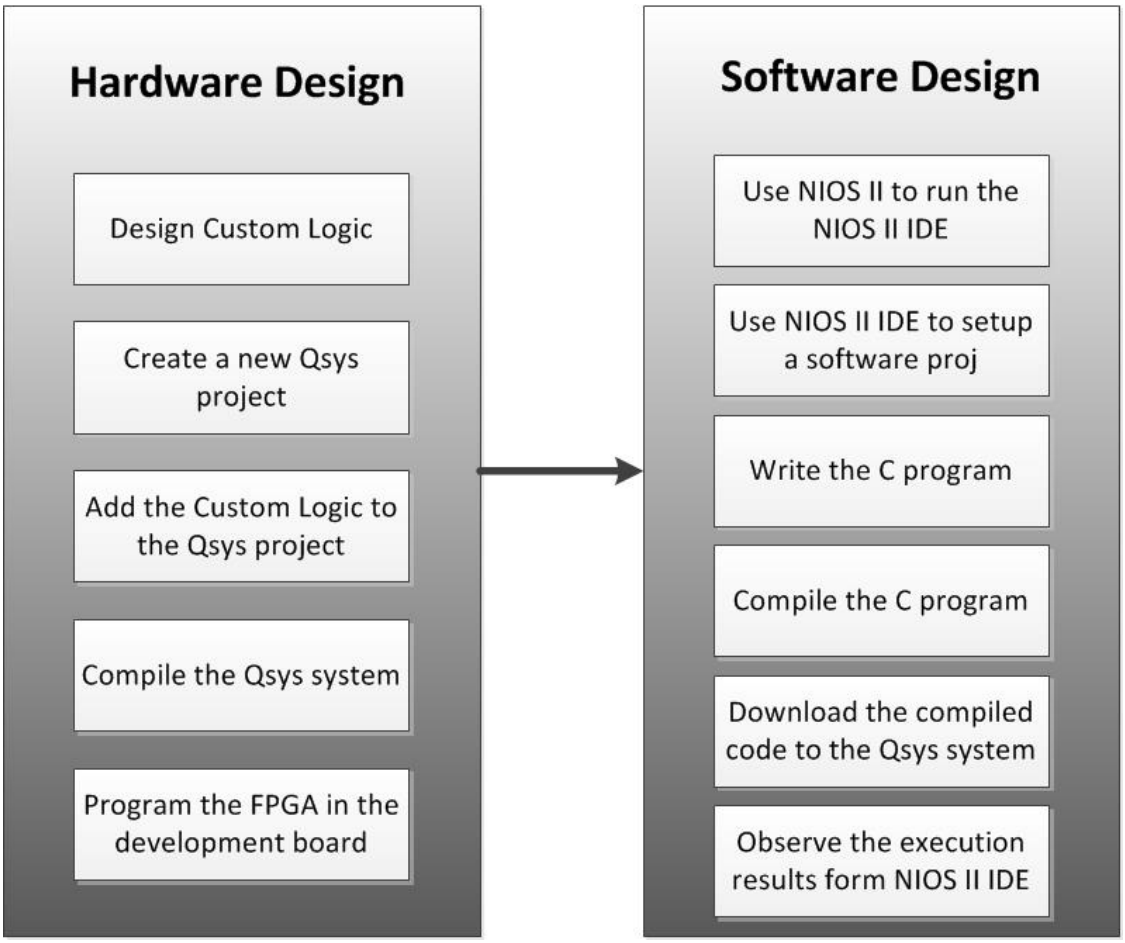


圖 2.5　軟硬體共同設計圖

2.4　回饋式模板架構

OSort演算法除了能做為棘波分類之用外，也能將分類的結果作為模板，回饋到棘波偵測步驟 [15]，在 2014 年王思淮發表的論文中，提出回饋式棘波偵測系統——Normalized Correlator[16]，在此篇論文中使用OSort演算法的分類結果作為Matched Filter的模板，來提高棘波偵測正確率，這是由於OSort演算法特殊的運算過程，在運算過後完整地保留整個棘波的形狀，不同於PCA、小波轉換和GHA等等特徵擷取的法則，這些法則在運算過後已將棘波資訊轉換壓縮成少量的資料，無法將結果作為模板，圖 2.6 為OSort硬體架構與Normalized Correlator硬體架構的關係，兩者透過NoC架構來傳遞資料。

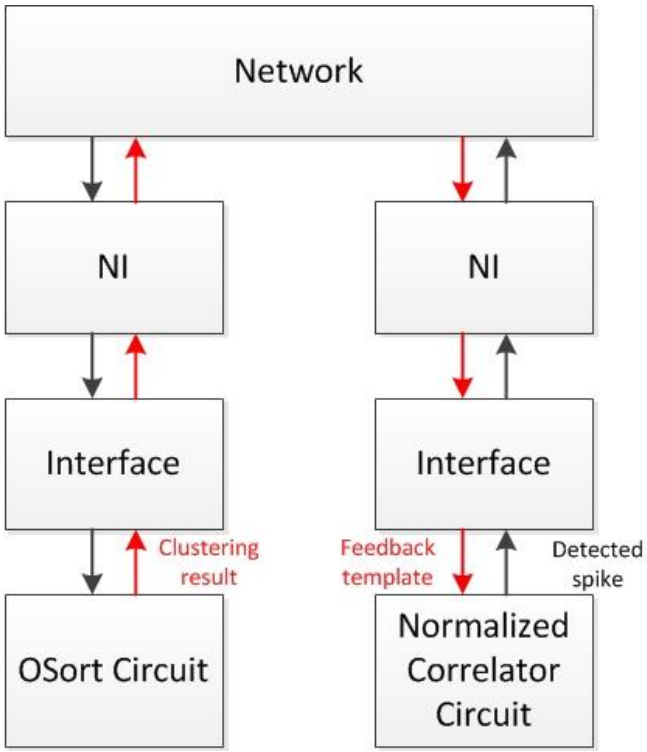


圖 2.6　棘波排序硬體架構

經由一段時間的棘波偵測，累積一定數量的棘波數目後，再將資料交由OSort演算法，將其分類結果，依群集中棘波的數量多寡作為優先權，將擁有最多棘波的群集之平均值傳遞Normalized Correlator系統，取代預設的模板，來提高棘波偵測的正確率。

第三章 系統架構

3.1 研究流程

本論文以軟體輔助硬體電路設計，先以軟體撰寫OSort演算法程式，在透過多次實驗制定相關參數，視其必要性修改演算法，達到最佳效能後，再以硬體實作，並將硬體實作結果與軟體實驗結果進行比較，以此來驗證此硬體電路的正確性，再與軟體以及相關研究進行效能比較，如圖 3.3 所示。

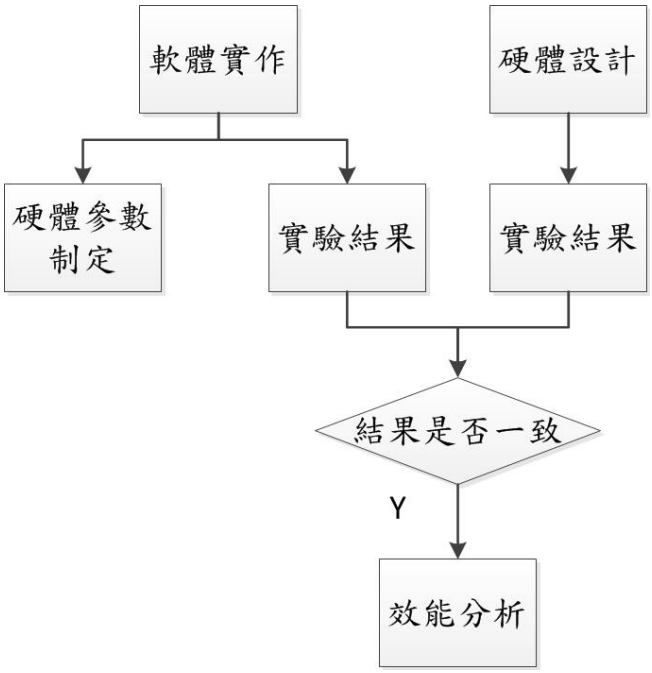


圖 3.1　研究流程圖

3.2 電路架構

在本節中將介紹整體的系統架構，依據OSort演算法，本論文將其所需之運算單元拆成五小單元，分別為記憶體單元、最小平方距離計算單元（MSD-CU, Minimum Squared Distance Computation Unit）、更新平均單元（MUU, Mean

Updating Unit）、比較器單元（Comparator）和控制單元（Controller），完整架構如圖 3.2 所示，接下來的各小節將會依序對每個單元進行詳細的描述。

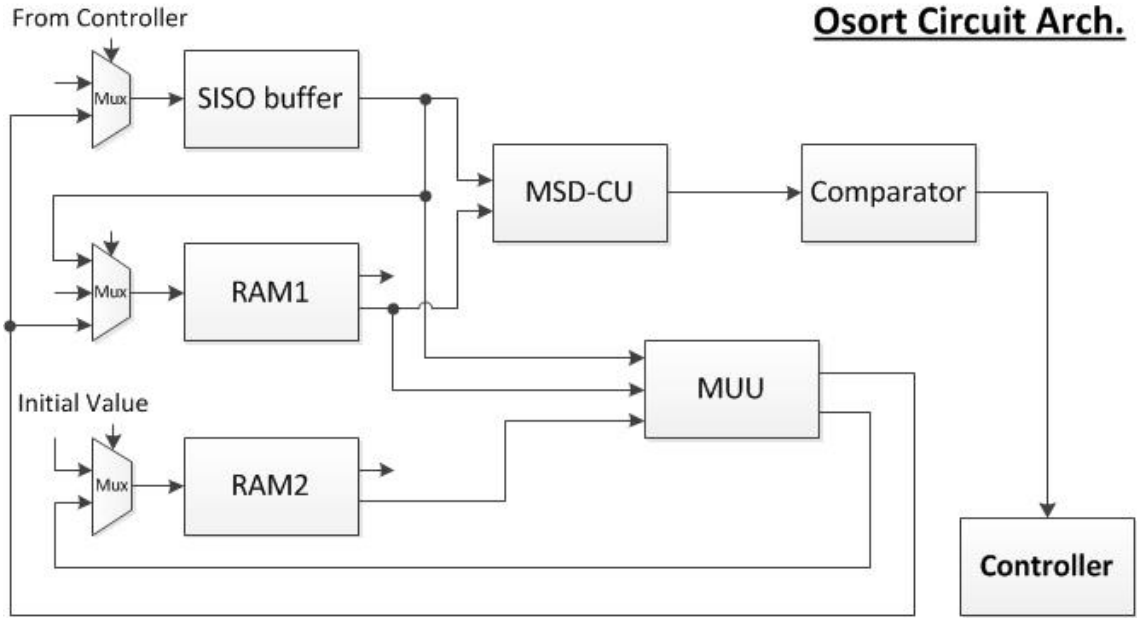


圖 3.2　OSort電路架構圖

3.2.1　記憶體單元

本節將介紹OSort電路中的記憶體單元，共有三部份分別為RAM1、RAM2 以及SISO buffer，由於硬體資源空間有限不能盡數記錄群集裡的棘波資訊，所以我們只保留各個群集中所有棘波的平均值來表示這個群集，RAM1 為儲存所有群集的平均值，RAM2 則記錄所有群集的棘波數量，以此來判斷群集大小以及後面相關運算使用。

在章節 2.2 提到OSort演算法可自動決定群集數目，根據流程圖可看出群集的數量是不斷地增減，經軟體實驗發現尤其以增加為常，當群集不斷地增加就需要

更多的硬體資源來記錄這些群集，但硬體無法提供無上限的資源供應，所以我們必須訂定一個最大的群集數目，32 個群集經實驗後認為是個可行的數字，在一個棘波以 64 個資料點、每個資料點 10bit紀錄的情況下，RAM1 包含 32*64*10 個暫存器來儲存資料，為了降低電路面積，本論文採用位移暫存器來傳遞資料，如圖 3.3 和圖 3.4 所示，多工器藉由Ctrl. A來選擇欲輸入或讀取，Ctrl. A則由Ctrl. 1 和 Ctrl. 2 經由Decoder後來決定，Index則控制欲選取的暫存器，控制訊號所代表的運作見表 3.1，M_i為群集之平均值。

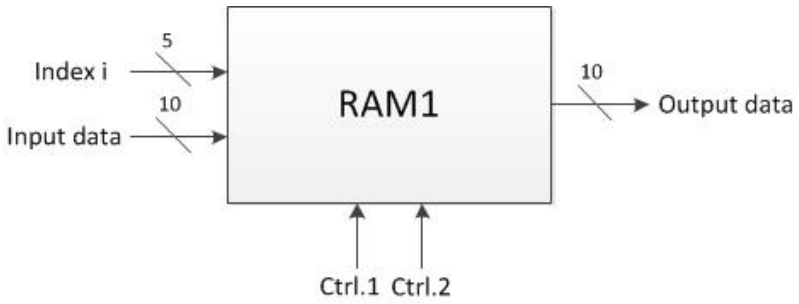


圖 3.3　RAM1 電路外觀圖（A）

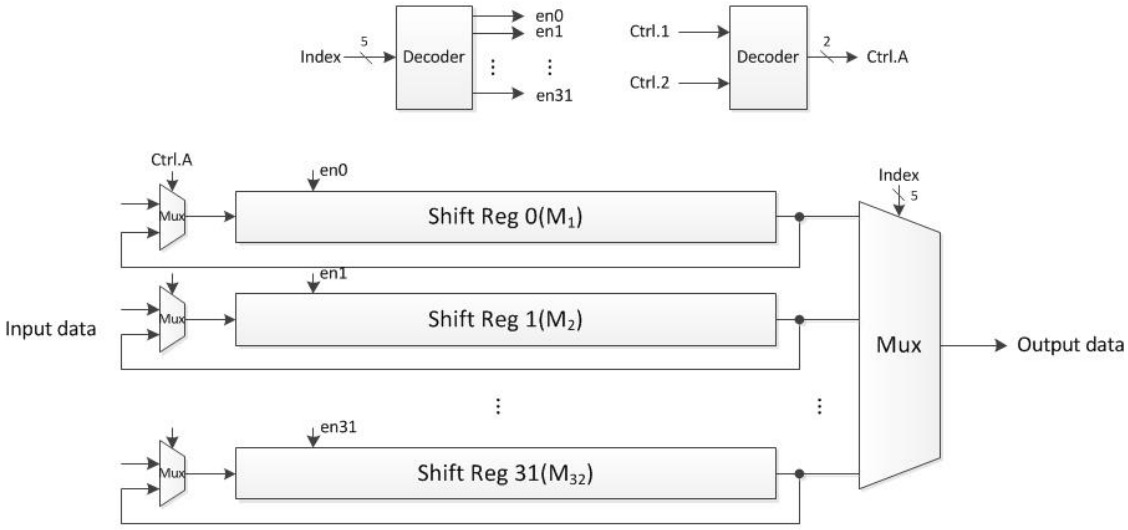


圖 3.4　RAM1 電路架構圖（B）

由於我們固定了群集的數目，為了有效利用記憶體空間，藉由新增控制訊號——有效位元（Valid bit）來記錄群集的使用狀況，再以 10bit來記錄棘波數量的情況下，RAM2 包含 32+32*10 個暫存器來儲存資料，如圖 3.4 所示，控制訊號所代表的運作見表 3.2，Q_i為群集之數量，V_i為有效位元。

Index	Ctrl. 1	Ctrl. 2	Function	Operation
i	0	1	Update M_i	New M_i→Input data
i	1	1	Output M_i	M_i→Output data
×	0	0	Nop	×
×	1	0	Nop	×

表 3.1　RAM1 運作表

SISO buffer（Serial-in-serial-out buffer）為存放外部新輸入之棘波數據的緩衝區，其架構與RAM1 相同，都是以位移的方式來傳遞資料，由 64*10 個暫存器所組成。

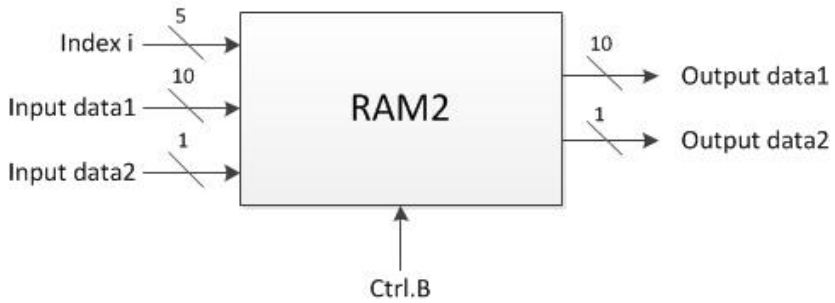


圖 3.5　RAM2 電路架構圖

Index	Ctrl. B	Function	Operation
i	0	Output Q_i and V_i	Q_i→Output data1 V_i→Output data2
i	1	Update Q_i and V_i	New Q_i→Input data1 New V_i→Input data2

表 3.2　RAM2 運作表

3.2.2　MSD-CU單元

本節將介紹最小平方距離計算單元的電路架構，此為OSort電路的主要計算單元，目的是在RAM1 裡找出與SISO buffer中之棘波最為相似的群集，如圖 3.6 所示。A為RAM1 所輸出之群集C_i的平均值，B為SISO buffer中的棘波數據，兩者相減再相乘後，不斷地累加即為群集C_i與此棘波之平方距離，透過比較器（Comparator）與目前儲存於D1 中最小距離的值相比較，若此值小於D1，則將值更新至D1 成為新的最小距離，D2 則等於i，是為記錄最相似的群集編號，直到與所有RAM1 中的群集做完運算後，再將D1 和D2 的值輸出。

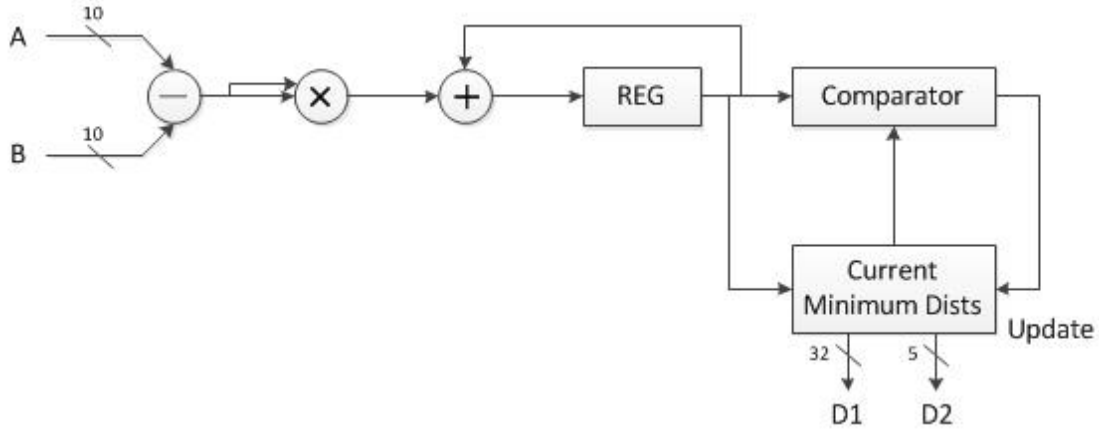


圖 3.6　MSD-CU電路架構圖

3.2.3　Comparator單元

圖 3.7　Comparator電路架構圖

在MSD-CU單元運算完成後，我們將最小平方距離D1 與閥值做比較，如圖 3.7 所示，若D1 小於等於閥值會輸出 1 給Controller，若D1 大於閥值則會輸出 0 給Controller，輸出的訊號代表的意義，之後小節會再作介紹。

3.2.4　MUU單元

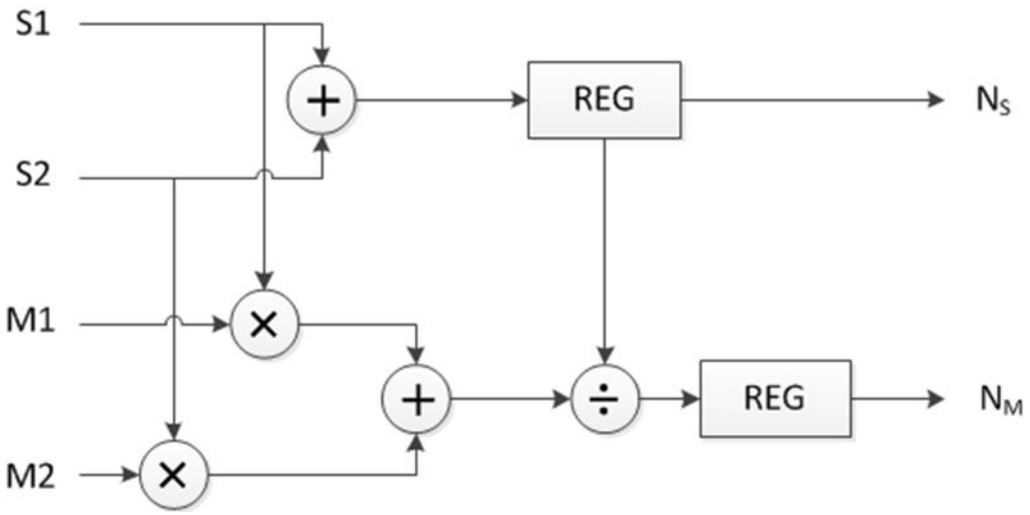


圖 3.8　MUU電路架構圖

本節將介紹更新平均單元的電路架構，此單元為棘波被分類到某個群集或兩個群集太過相似需要合併（Merge）時，所需之運算單元，負責重新計算群集之平均值，如圖 3.8 所示，S1、S2 為群集之數量，M1、M2 為群集之平均值，首先將M1 乘以S1、M2 乘以S2 來還原數據，再將兩者的值相加除以新的棘波數量N_S（S1 與S2 相加），得到的結果即為新的平均值N_M。

3.2.5　Controller單元

本節將介紹控制器的設計概念，以及在不同控制模式下OSort電路的運作流程。Controller在OSort電路中扮演重要的角色，它是電路中各個單元資料傳輸的樞紐，透過設計不同的模式，組合不同的單元來達成運算，本論文將Controller細分為 7 個模式，如表 3.3 所示。

狀態	使用到的單元	運作內容
Mode 1	RAM1、RAM2	將外部資料直接寫入記憶體單元，或作為第一個棘波輸入之用
Mode 2	RAM1、RAM2	讀取記憶體單元
Mode 3	SISO buffer、MSD-CU、Comparator	棘波輸入後計算最小平方距離和與閥值的關係
Mode 4	SISO buffer、RAM1、RAM2	創造一個新的群集
Mode 5	SISO buffer、MSD-CU、Comparator	將棘波分類到某個目前存在的群集，並計算此群集與其餘群集之關係
Mode 6	MUU、RAM1、RAM2	合併兩個相似的群集
Mode 7	RAM2	從記憶體單元中清空數量最少的群集

表 3.3　控制器之狀態表

OSort電路就是以上述這 7 個模式來控制運作，如圖 3.9 所示，首先，我們將電路的第一個棘波透過Mode 1，直接寫入RAM1 成為第一個群集，同時更新RAM2的數量資訊和有效位元；當第二個以後的棘波輸入時，皆透過Mode 3，先將棘波資訊寫入SISO buffer，並與RAM1 中所有的群集計算最小平方距離，透過MSD-CU單元得到此值後，再與閥值做比較，接著便傳送一個訊號給Controller。

當訊號為 0 時,表示此棘波與所有群集都相差甚遠,需要新增一個新的群集，首先，必須透過RAM2 中的有效位元檢視RAM1 可存放的群集空間是否已滿，若記憶體單元已滿，則透過Mode 7 來清出一個群集空間，將棘波數量最小的群集之有效位元設為 0，來達到釋放記憶體空間的目的，接著透過Mode 4 來新增群集，將儲存於SISO buffer中的棘波資訊寫入RAM1。

當訊號為 1 時，表示此棘波與某一個群集C_i相似，我們透過Mode 5 將此棘波分類到C_i，並經由MUU單元重新計算C_i之平均值，再將更新後的值寫回RAM1 和SISO buffer中，同時更新RAM2 中群集的棘波數量；接著將目前存在SISO buffer中更新後的群集平均值，再與RAM1 中的其餘群集計算最小平方距離，來確認群集間是否過於相近，同樣透過MSD-CU單元與Comparator單元來做運算，並傳送一個訊號給Controller。

當訊號為 0 時，表示並無群集與群集C_i過於相近；當訊號為 1 時，表示群集C_i與某個群集C_j相當接近，則需透過Mode 6 將此兩群集合併，以MUU單元將此兩群集合併後，將新的群集平均值寫回RAM1，同時更新RAM2，並將RAM2 中其中一群集的有效位元設為 0，來達到合併的目的。

根據上述的模式組合，我們可以不斷地輸入棘波來做分類，當所有棘波都分

群完畢後，可以透過Mode 2 來將各個群集最後的平均值以及群集所含的棘波數量輸出至電路外部，以提供棘波分類演算法的結果。

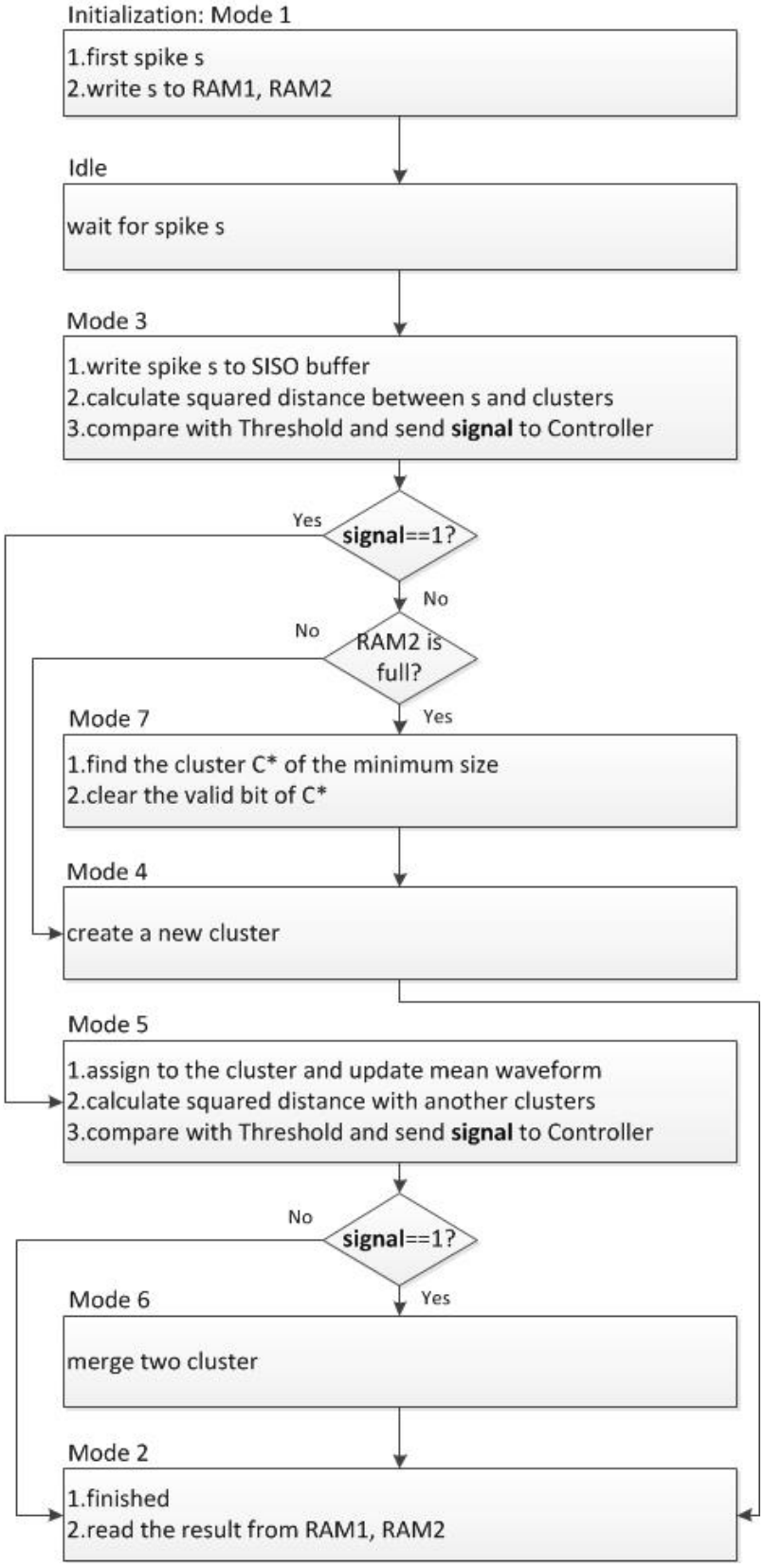


圖 3.9　OSort電路流程圖

圖 3.10 至圖 3.15 為控制器Mode 1 至Mode 7 在OSort電路架構圖中資源使用的狀況，以及使用到的線路圖。

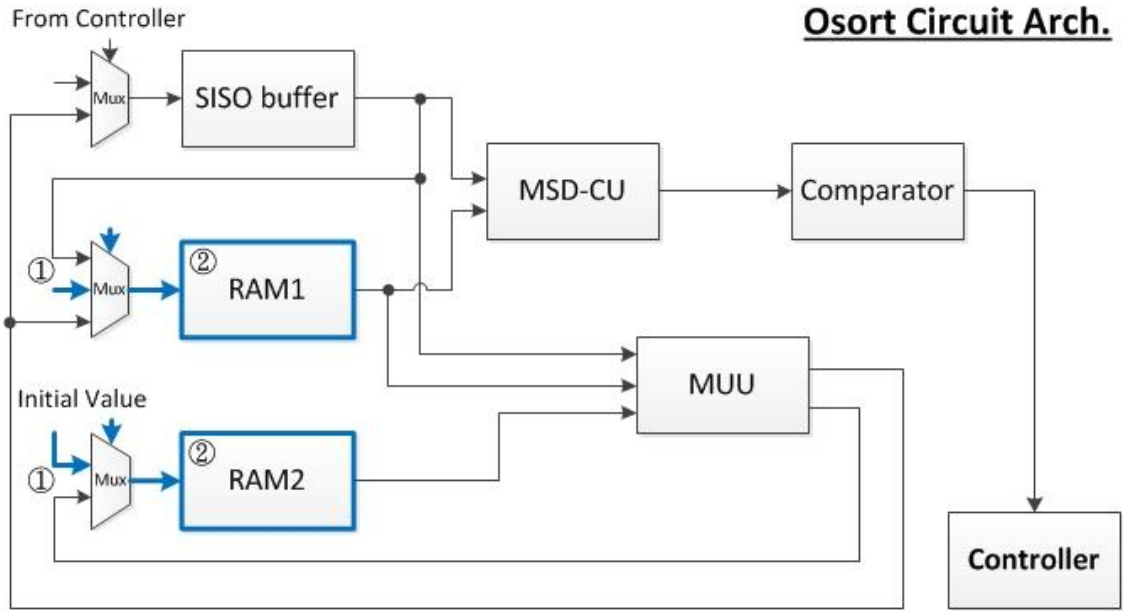


圖 3.10　Mode 1 資源使用狀況

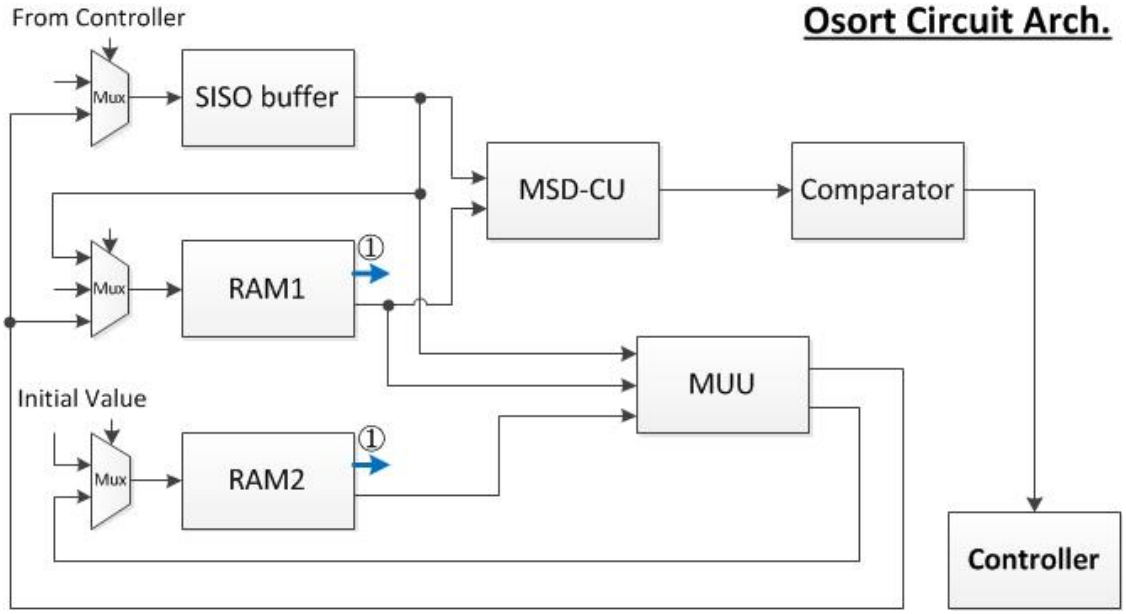


圖 3.11　Mode 2 資源使用狀況

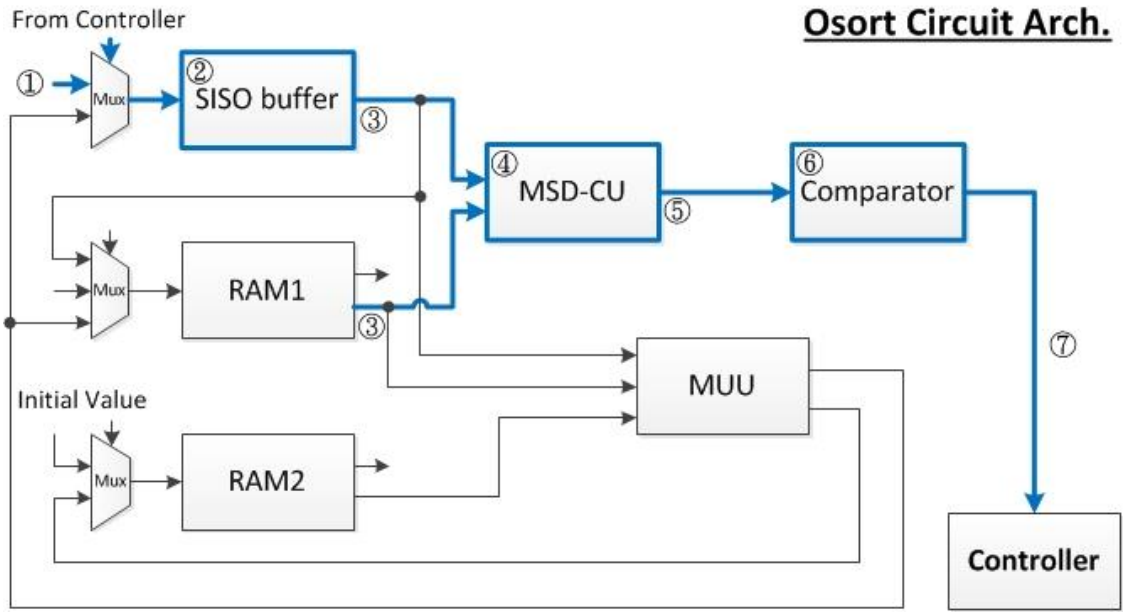


圖 3.12　Mode 3 資源使用狀況

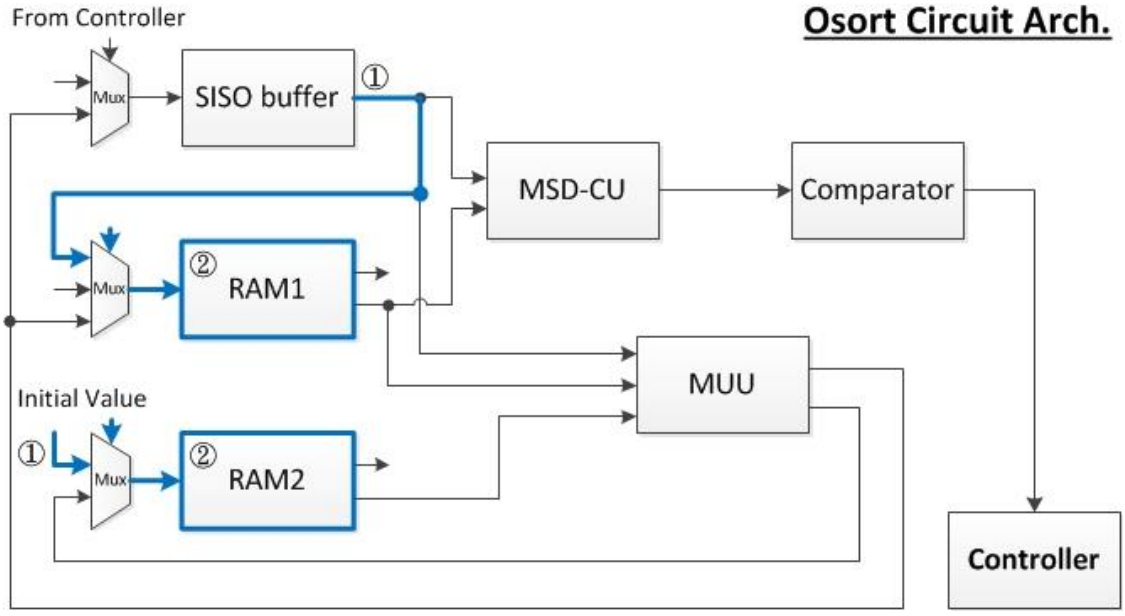


圖 3.13　Mode 4 資源使用狀況

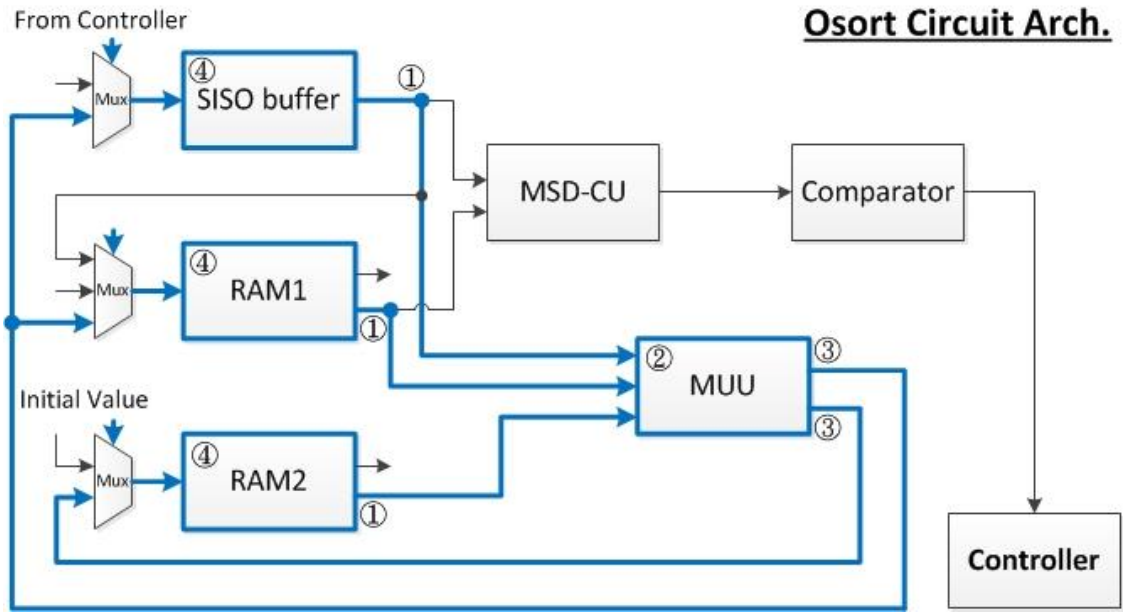


圖 3.14　Mode 5-1 資源使用狀況

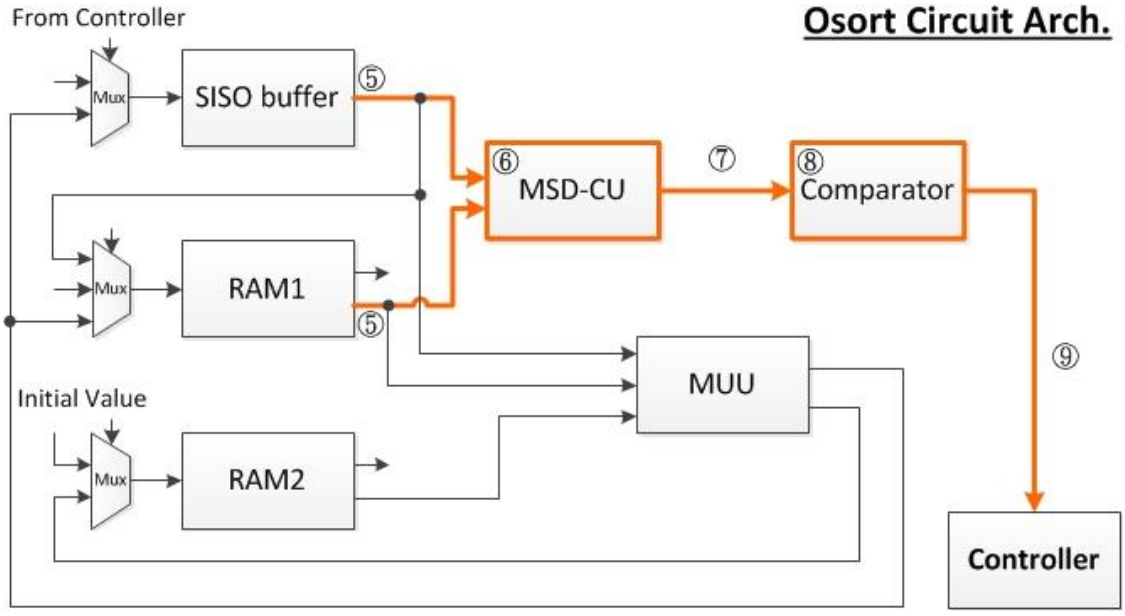


圖 3.15　Mode 5-2 資源使用狀況

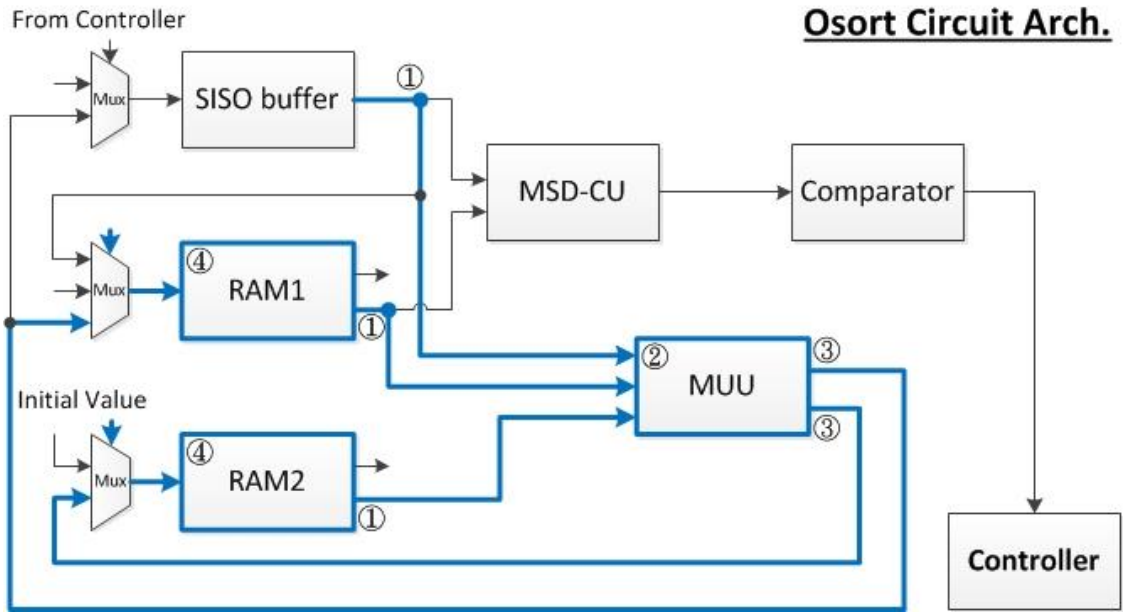


圖 3.16　Mode 6 資源使用狀況

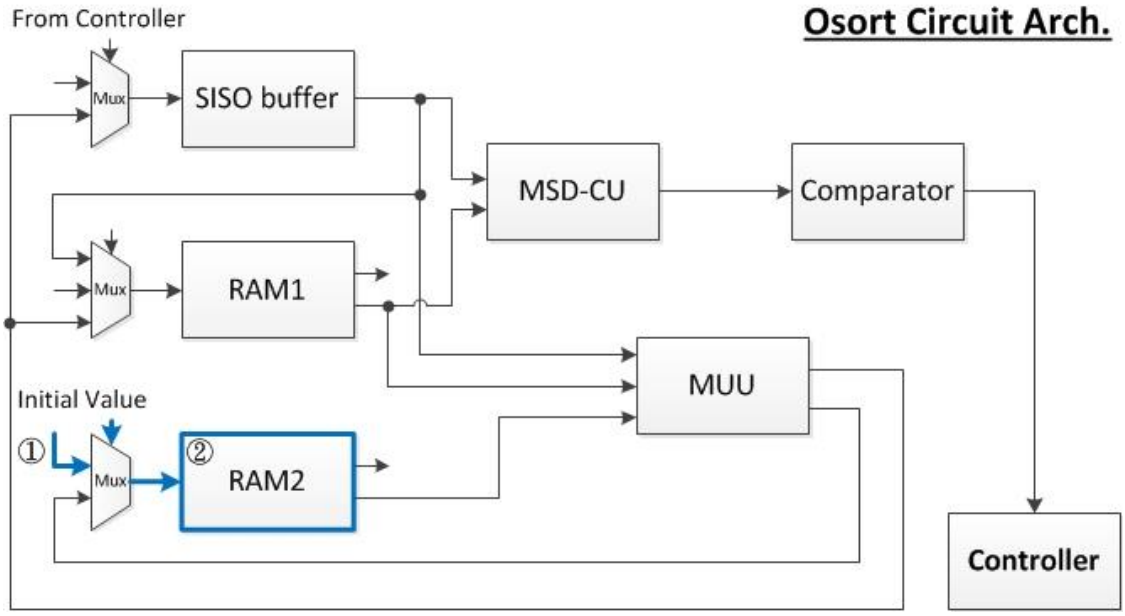


圖 3.17　Mode 7 資源使用狀況

第四章 實驗數據與效能分析

4.1 開發平台與實驗環境

本論文使用Altera公司推出的DE2-115 EP4CE115F29C7 開發板來實現棘波分類系統之硬體架構，如圖 4.1 所示，DE2-115 開發板擁有豐富多樣的周邊應用介面，常為教育、研究所使用，該開發板搭配Cyclone IV E系列晶片中容量最大的EP4CE115 FPGA，提供大量的邏輯單元和記憶體資源，其詳細規格如表 4.1 所示，其特點為低成本、高性能且低功率消耗，並在DSP效能方面提供最佳解決方案，相較於前一代的Cyclone FPGA，降低了 25%的功率消耗，可滿足各類型的應用需求。

DE2-115 開發板規格表	
Family	Cyclone IV E
Device	EP4CE115F29C7
Logic Elements(LEs)	114480
Total memory bits	3981312(3.9Mbits)
Embedded 9x9 multipliers	532
Total pins	529
Total PLLs	4

表 4.1　Altera DE2-115 EP4CE115F29C7 開發板規格表

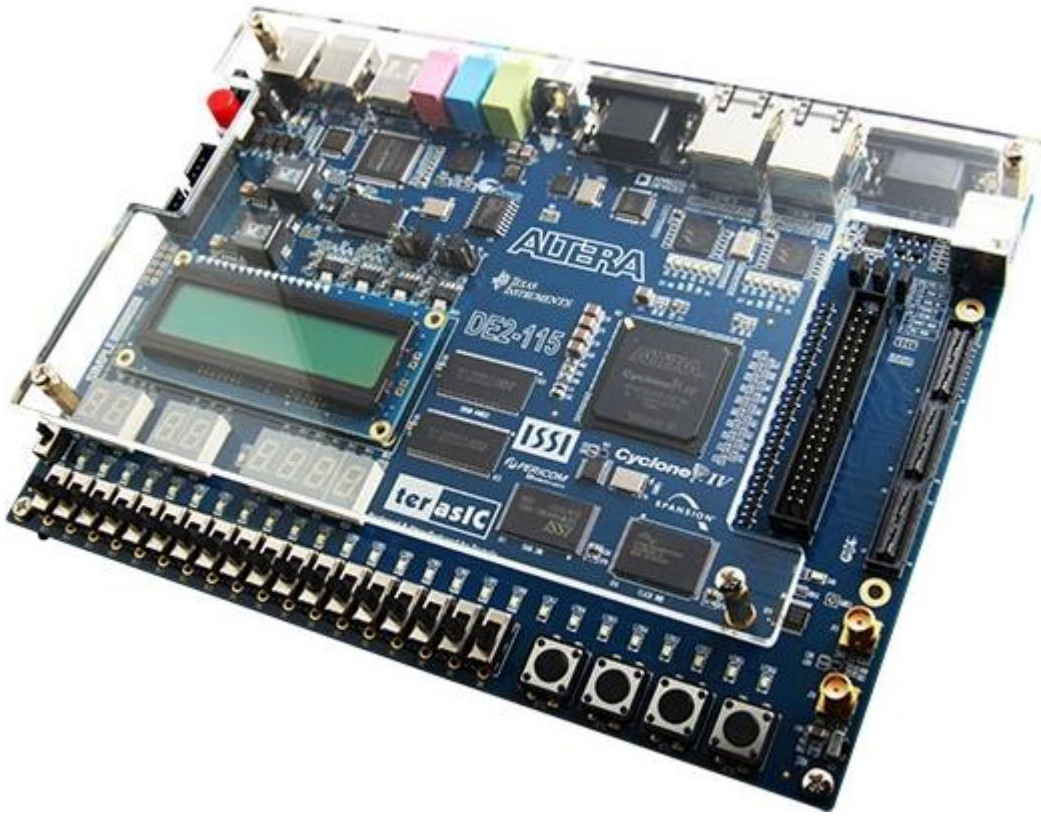


圖 4.1　Altera DE2-115 開發板

本論文使用Altera Quartus II 13.0.1 SP1 64-Bit Version作為撰寫硬體描述語言Verilog HDL的平台，進行設計與開發，透過Quartus II提供的語法檢查、時序分析、電路合成、佈局和繞線等功能，並以ModelSim-Altera 10.1d模擬電路，驗證其正確性，即可燒錄製FPGA上，再由以Eclipse為基礎的NIOS II IDE進行測試。

MATLAB是由美國MathWorks公司推出的商業數學軟體，採取直譯的方式，可以查取工作區（Workspace）的變數數值，利於程式除錯，提供許多不同領域的工具箱（Toolbox），如訊號處理、影像處理等等，大量的函式取代程式語言的副程式，節省大量的撰寫時間，鑒於以上優點，本論文選定MATLAB 2009a作為軟

體實作的工具。

綜合以上，可以整理如下：

※硬體開發環境

FPGA：Altera DE2-115

※軟體開發環境

CPU：Intel® Core™ i7-3770 CPU @ 3.40GHz

Memory：DDRIII 16.0 G

※開發工具

Altera Quartus II 13.0 SP1 64-bit Version

ModelSim-Altera 10.1d

Nios II 13.0 SP1

MATLAB 2009a

4.2　實驗數據與結果呈現

本論文使用L.S. Smith和N. Mtetwa在 2007 年提出的細胞外紀錄模擬器[17]，可用於模擬不同的棘波序列，提供棘波個數、棘波種類、採樣頻率和SNR（Signal to Noise Ratio）等等的變數給使用者做調整，除了產生近似於真實情況的原始數據外，它也提供了標準（Ground Truth），使我們可以有個準則來衡量棘波分類演算法的優劣，進一步的來評估我們的系統效能。在實驗的過程中，我們使用不同SNR值下所產生的棘波序列，採樣頻率設定為每一秒 24,000 個採樣點，模擬器產生的棘波訊號長度為 2.67ms，每個棘波包含 64 個採樣點。

在討論實驗數據前，我們必須先認識一個名詞——SNR，所謂的SNR（Signal-to-Noise Ratio），又稱為訊噪比，用來表示訊號和雜訊的強度比率，單位為分貝（dB），SNR值越高表示雜訊越少，代表訊號的品質越好，反之，SNR值越低，則表示此訊號受到雜訊干擾程度嚴重，SNR值的計算見公式(4.1)及(4.2)，P_{signal}為訊號功率，P_{noise}為雜訊功率，A_{signal}為訊號振幅，A_{noise}為訊號振幅。在我們的實驗中，SNR=100 為沒有雜訊干擾的棘波序列，圖 4-2 和 4.3 為我們在SNR=8和SNR=-2 下擷取的棘波序列，可以更清楚的了解到SNR值的高低對訊號的影響，圖中有標示的部分代表該區段有棘波產生，隨著SNR值的下降，我們越來越不容易分辨棘波與雜訊，使得棘波分類的難度也隨之增高。

$$\mathrm{SNR} = \frac{P_{\mathrm{singal}}}{P_{\mathrm{noise}}} = \left(\frac{A_{\mathrm{signal}}}{A_{\mathrm{noise}}}\right)^2 \tag{4.1}$$

$$\mathrm{SNR}_{dB} = 10\log_{10}\left(\frac{P_{\mathrm{singal}}}{P_{\mathrm{noise}}}\right) \tag{4.2}$$

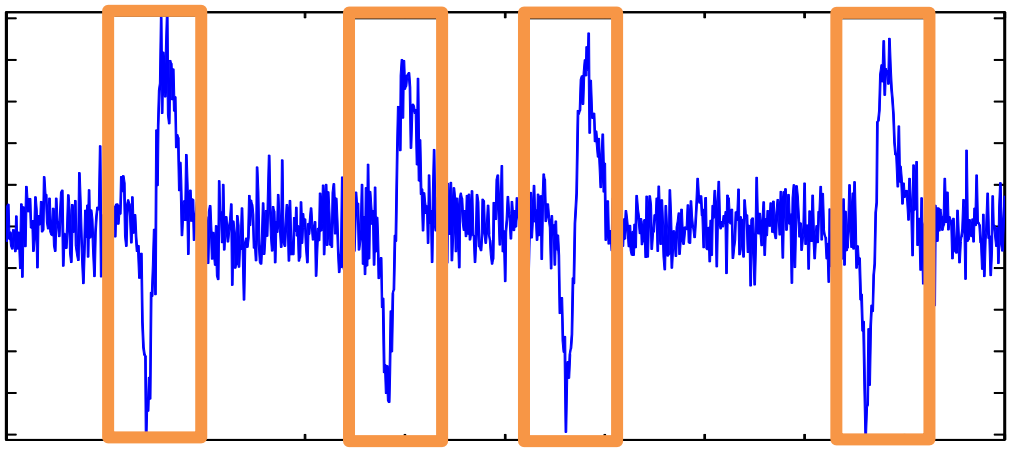


圖 4.2　SNR=8 下的棘波序列

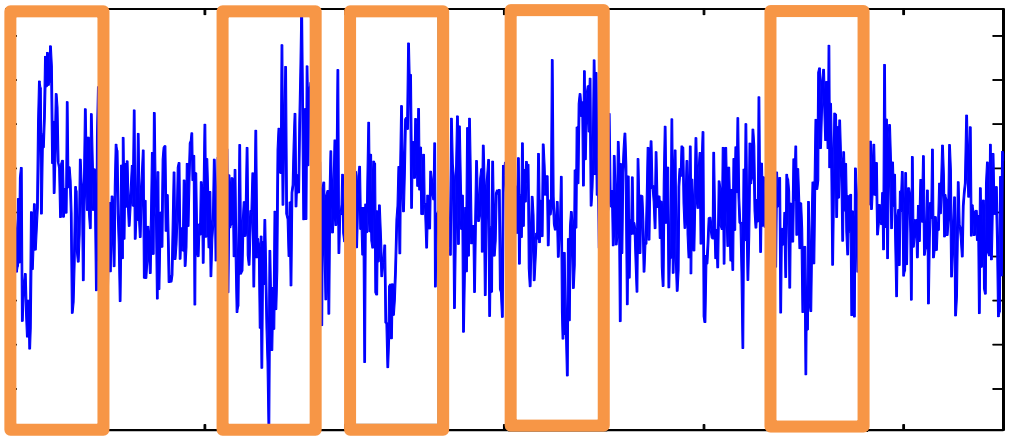


圖 4.3　SNR=-2 下的棘波序列

為了有效且完整的測試，本論文使用王思淮所提供的Normalized Correlator硬體架構和以模擬器產生的數據，作為棘波偵測硬體架構[16]和測試資料，與OSort硬體架構組合成為一套棘波排序硬體架構，兩者的運算資料皆採用 10bit定點數格式，再將結果進行軟體與硬體比對，表 4.2 為在不同SNR值下，以模擬器產生包含兩種不同棘波的棘波序列，經棘波排序運算後，由欄位 3、4 可發現，只要

Normalized Correlatort成功地偵測到棘波，我們的棘波分類正確率都維持在 99%的高正確率，而整體的正確率，即使在SNR值為-2，雜訊如此高的情況下還能維持高達 82.51%的棘波排序正確率，整體而言此架構抵抗雜訊的能力非常顯著。

SNR(dB)	Total number of spikes	Number of spikes not detected	Number of spikes detected & mis-classified	Number of spikes detected & correctly-classified
-2	2378	406 (17.07 %)	10 (0.42 %)	1962 (82.51 %)
0	2308	260 (11.27 %)	11 (0.48 %)	2035 (88.17 %)
2	2375	240 (10.10 %)	10 (0.42 %)	2115 (89.05 %)
4	2387	197 (8.25 %)	10 (0.42 %)	2180 (91.32 %)
6	2359	194 (8.22 %)	9 (0.38 %)	2156 (91.38 %)
8	2332	165(7.08%)	7 (0.30 %)	2160 (92.62 %)

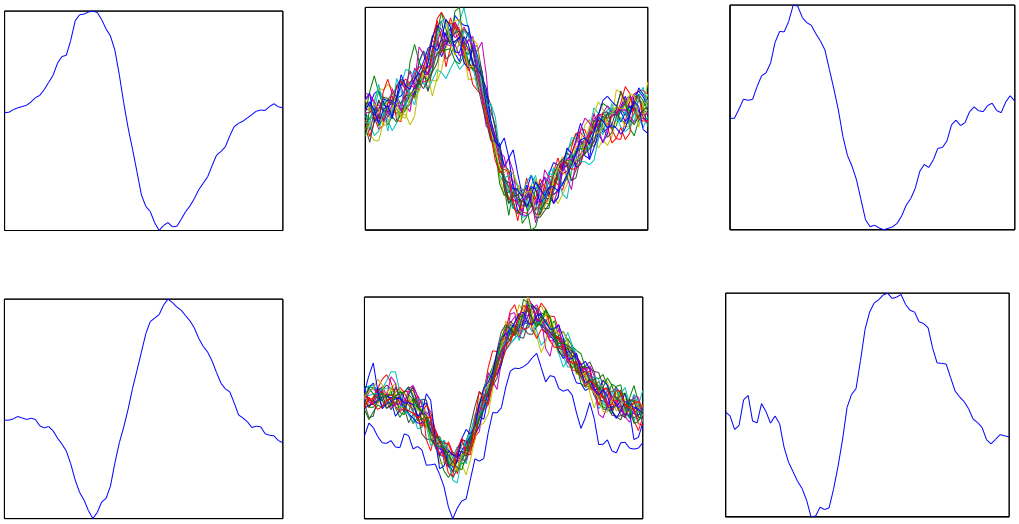
表 4.2　不同SNR值下棘波排序的正確率（兩群）

表 4.3 為在不同SNR值下，以模擬器產生包含三種不同棘波的棘波序列，經棘波排序運算後，由欄位 3、4 可發現，棘波偵測的正確率差別不大，由於棘波的種類增加，棘波分類的正確率些微下降，整體而言在SNR值大於 2 的情況下，

依舊維持 80%以上的棘波排序正確率，在SNR值為-2 高雜訊的情況下，棘波排序正確率也都還維持在七成左右。

SNR(dB)	Total number of spikes	Number of spikes not detected	Number of spikes detected & mis-classified	Number of spikes detected & correctly-classified
-2	2014	354 (17.57 %)	206 (10.23 %)	1454 (72.19 %)
0	1990	327 (16.43 %)	194 (9.75 %)	1469 (73.81 %)
2	1983	217 (10.94 %)	153 (7.71 %)	1613 (81.34 %)
4	1988	227 (11.42 %)	81 (4.07 %)	1680 (84.51 %)
6	2008	219 (10.91 %)	166 (8.27 %)	1623 (80.83 %)
8	1987	179 (9.00 %)	123 (6.34 %)	1682 (84.65 %)

表 4.3　不同SNR值下棘波排序的正確率（三群）



(A)　　(B)　　(C)

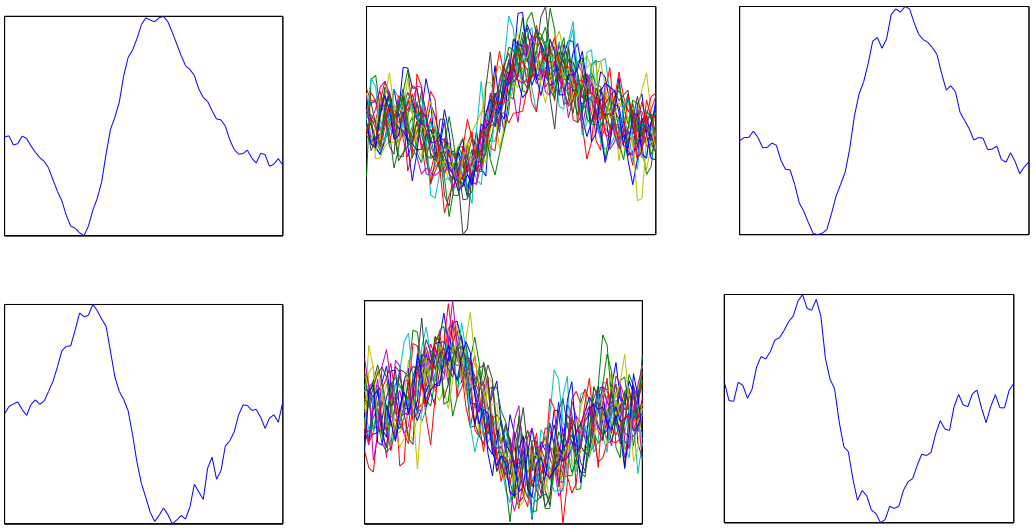
圖 4.4　SNR=8 時的棘波圖形（兩群）

(A) 同一群的棘波平均值。
(B) OSort演算法分類的結果。
(C) OSort電路回饋的模板。

在第 2.3 節提到可以將OSort電路的分群結果，做為回饋式的模板，提供給棘波偵測電路Normalized Correlatort所使用，以此增加偵測正確率。在SNR=8 情況下，我們實際將OSort電路的分群結果，透過軟體Matlab的輔助，先算出模擬器提供的數據中同一群的棘波平均，如圖 4.4（A）所示，(B)為OSort演算法分類的結果，並以軟體Matlab顯示每一個棘波的形狀，可看出雖屬於同一群棘波，但資料點起伏程度略有不同，（C）為OSort電路所回饋的模板，雖然與（A）稍有誤差，但整體上形狀相似，該模板依然可以有效地提供給棘波偵測電路。

圖 4.5 為在SNR=-2 下的實驗結果，在雜訊很高的情況下，偵測出的棘波已經

不是平滑的曲線，可以在（E）看到每個棘波的資訊，所記錄到的棘波資訊範圍幅度增大，但藉由（F）OSort電路所回饋的模板與（D）軟體計算的結果相比，結果並沒有相差太遠，同樣能將模板回饋給Normalized Correlatort電路，來提高棘波偵測正確率。



(D)　　(E)　　(F)

圖 4.5　SNR=-2 時的棘波圖形（兩群）

(D) 所有棘波的平均值。
(E) OSort演算法分類的結果。
(F) OSort電路回饋的模板。

本實驗設計於Altera Quartus II 的系統開發工具Qsys與NIOS II IDE平台，所使用的FPGA開發板為DE2-115，在FPGA開發板上，硬體資源主要可以分成三大部分，分別為Logic Elements、Memory bits和Embedded Multiplier，Logic Elements被用於實現暫存器以及運算單元，Memory bits為記憶體元件，Embedded Multiplier

則用來作為乘法器或除法器的元件。

從表 4.4 中可以觀察到OSort電路所消耗的資源，第一列為電路以 4 群為最大群集上限數，第二列為電路以 32 群為最大群集上限數，其餘參數皆無做更改，所以擁有相同的Embedded Multiplier，Memory bits的使用率趨近於 0，唯一的差別在於LEs的數量，可以看出OSort電路所使用的LEs數量與最大分群數量有直接的關係，未來若讀者有興趣實作此系統，可根據微電極的通道數量和FPGA開發板的資源上限，做最大群集數量的調整，來降低多餘的資源消耗，而本論文是以 32 群集為例。

	Logic Elements (LEs)	**Memory bits**	**Embedded Multiplier**
OSort Circuit (upper limit with 4 clusters)	9871/114480 (9%)	0/3.9Mbits (0%)	132/532 (25%)
OSort Circuit (upper limit with 32 clusters)	24426/114480 (21%)	320/3.9Mbits (<1%)	132/532 (25%)

表 4.4　本論文提出之棘波分類系統硬體資源消耗表（A）

表 4.5 為加入Normalized Correlator電路的硬體資源消耗表，第二列為Normalized Correlator電路的資源消耗，第三列為Nios CPU以及其他燒錄至FPGA

所需之資源消耗的資源量，我們宣告 1Mbits的on-chip RAM作為我們存放程式碼的空間，由於本論文提出的硬體架構只與Normalized Correlator電路初步整合，所以需要額外的空間來儲存資料，可以在第四列觀察到我們使用額外的Buffers來儲存棘波，OSort電路再透過NoC架構從此處抓取資料，所以若要提升系統效能、增進傳輸速率以及降低資源消耗，則需要再更進一步地討論。最後則為本電路架構與Normalized Correlator透過Qsys加入硬體元件後的整體的資源消耗。

	Logic Elements (LEs)	**Memory bits**	**Embedded Multiplier**
OSort Circuit	24426/114480 (21%)	320/3.9Mbits (<1%)	132/532 (25%)
Normalized Correlator Circuit	15618/114480 (13%)	0/3.9Mbits (0%)	532/532 (100%)
Nios CPU+ on-chip RAM+ JTAG	3102/114480 (3%)	882944/3.9Mbits (22%)	4/532 (<1%)
Buffers	39982/114480 (35%)	0/3.9Mbits (0%)	0/532 (0%)
Entire NoC	83128/114480 (73%)	882944/3.9Mbits (22%)	532/532 (100%)

表 4.5　本論文提出之棘波排序系統硬體資源消耗表（B）

從第三章圖 3.8 可知，OSort電路是一個FSM的結構，所以每個棘波需要的運

算時間並不相同，表 4.6 為控制器中的模式所需要的Clock Cycle數目，C為群集數目，N為資料點數量，由於Mode 2 使用的時機是在所有棘波分類完後，所以在此不討論其Clock　Cycle數目，表 4.7 為所有棘波以Mode來表示的資料路徑情況。

狀態	Clock Cycle數目（*C*=32, *N*=64）
Mode 1	*N*
Mode 2	--
Mode 3	3*C*+*C***N*+1
Mode 4	*N*
Mode 5	(*N*+1)+(3*C*+*C***N*+1)
Mode 6	2*N*+1
Mode 7	*C*+1

表 4.6　各狀態的Clock Cyle數目

花費最多的時間為編號 5 的情況，需要 6C+3N+2*C*N+4 共 4484 個Clock Cycle，最短則為編號 1 的情況，但編號 1 的情況僅會發生在第一個棘波輸入的時候，正常狀態下仍舊是編號 4 和 5 的情況較常發生，我們以Worst Case也就是編號 5 的情況，來估測一秒鐘可運算的棘波數目。

編號	順序	意義
1	Mode 1	運算第一筆資料
2	Mode 3→Mode 4	產生新的群集
3	Mode 3→Mode 7→Mode 4	產生新的群集（需清空記憶體）
4	Mode 3→Mode 5	將棘波分類到某群集
5	Mode 3→Mode 5→Mode 6	將棘波分類到某群集，此群集又與其他群集合併

表 4.7　資料路徑表

Clock Rate	**50MHz**	**100MHz**	**500MHz**	**[11]**	**[18] 100MHz**
One Spike Computation Time	89.7μs	44.8μs	8.9μs	833.3μs	3.8μs
Spikes Computed in One Second	11150	22301	111507	1100	263157

表 4.8　OSort電路Throughput分析

如表 4.8 所示，與軟體相比[11]，OSort演算法平均每秒可完成 1100 個棘波分類，對比於OSort電路在Clock Rate為 500MHz的情況下，本論文提出的硬體架構，可提升 101 倍以上的運算量。[18]所提出之OSort演算法硬體架構，由於使用PIPO

（Parallel-in-parallel-out）的資料傳輸方式，減少了大量的Clock Cycle時間，所以單位運算量較高，但所耗費的成本就是面積消耗高，如表 4.9 所示，[18]使用的是Xilinx的FPGA開發板，與Altera相比，兩家公司所定義設計的邏輯電路不盡相同，所以我們以盡可能合理的方式來比較。

在乘法器的部分DSP類似於Multiplier，我們使用的是 9x9 的單位而[18]則使用 25x18 的乘法器，就使用的資源上來看是差不多的；在使用的Memory bits部分，[18]是遠遠高出我們許多，在Logic Elements的部分，Xilinx是以slice LUTs為單位，約等於 2 個Logic Elements，本研究所提出的硬體架構所使用的資源量約為[18]的一半，在這些比較之下，我們的電路擁有較低的資源消耗和面積消耗。

	Logic Elements (LEs)	**Memory bits**	**Embedded Multiplier (9x9)**	**DSP48E (25x18)**
This work	24426	320	132	0
[18]	47134	225000	0	29

表 4.9　OSort電路與其他研究比較

由於現今植入於人腦的微電極通道數量，約在 64~128 通道，而一個棘波產

生的時間為 1~4 毫秒，以本研究所提出的硬體架構來說，在Clock Rate 500MHz 的情況下在 4 毫秒內可運算的棘波數目為 449 個，已經遠大於平均的通道數目，在使用上已經相當夠用，節省下來的硬體資源，也可拿來設計其他的應用電路與OSort演算法做結合，提供給讀者作為參考。

第五章 結論

本研究實現OSort演算法之硬體架構,透過第四章的實驗結果,本論文在兩群、三群的模擬資料中，在不同的SNR值下，擁有高的分群正確率，亦能將分類結果作為模板回饋給棘波偵測演算法，來提高偵測的正確率。雖然本論文提出的硬體架構運算速度與[17]相比較慢，但在資源使用量方面本論文較占優勢，且單位時間能處理的資料量已經遠大於不管是單通道或多通道所能採集到的資料量，在Clock Rate為 1GHz的情況下與軟體相比減少 185 倍的運算時間，大幅地縮減運算時間，且本電路亦可根據不同數據進行參數調整來提高正確率，由於FPGA開發板還有許多資源，未來可視使用者需求擴充系統，亦可提供腦機介面設計參考，因此，本論文所提出之棘波分類硬體架構是一項符合實際需求且運算快速的設計。

參考文獻

[1] E. R. Kandel, J. H. Schwartz and T. M. Jessell, Principles of neural science, New York: McGraw-Hill, 2000.

[2] A. L. Hodgkin and A. F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol., Vol. 117, pp. 500–544, 1952.

[3] M. A. Lebedev and M. A. L. Nicolelis, Brain machine interfaces: past, present and future, Trends in Neurosciences, Vol. 29, pp. 536-546, 2006.

[4] S. Gibson, J. W. Judy and D. Markovic, Spike sorting: the first step in decoding the brain, IEEE Signal Processing Magazine, pp. 124-143, 2012.

[5] http://neurosky.com/

[6] M. S. Lewicki, A review of methods for spike sorting: The detection and classification of neural action potentials, Netw. Comput. Neural Syst., Vol. 9, pp. R53–R78, 1998.

[7] K. Pearson, On lines and planes of closest fit to systems of points in space, Phil. Mag., Vol. 2, pp. 559–572, 1901.

[8] S. J. Lin, Y. T. Hung and W. J. Hwang, Efficient hardware architecture based on generalized Hebbian algorithm for texture classification, Neurocomputing, pp. 3248-3256, 2011.

[9] R. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering, Neural Comput., Vol. 16, pp. 1661-1687, 2004.

[10] K. L. Wu and M. S. Yang, A cluster validity index for fuzzy clustering, Pattern Recognit. Lett., Vol. 26, pp. 1275-1291, 2005.

[11] J. Wild, Z. Prekopcsak, T. Sieger, D. Novak and R. Jech, Performance comparison of extracellular spike sorting algorithms for single-channel recordings, J. Neurosci. Methods, Vol. 203, pp. 369-376, 2012.

[12] S. Mukhopadhyay and G. C. Ray, A new interpretation of nonlinear energy operator and its efficacy in spike detection, IEEE Trans. Biomed. Eng., Vol. 45, pp. 180-187, 1998.

[13] T. Sato, T. Suzuki, and K. Mabuchi, Fast Template Matching for Spike Sorting, Electronics and Communications in Japan, Vol. 92, pp. 57-63, 2009.

[14] U. Rutishauser, Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo, J. Neurosci. Methods, Vol. 154, pp. 204-224, 2006.

[15] W. J. Hwang, S. H. Wang and Y. T. Hsu, Spike Detection Based on Normalized Correlation with Automatic Template Generation, Sensors, Vol. 14, 2014.

[16] 王思淮, 以回饋式自動模板生成為基礎之正規化關聯值棘波偵測系統之設計及實現, 2014.

[17] L.S. Smith and N. Mtetwa, A tool for synthesizing spike trains with realistic interference. J. Neurosci. Methods, Vol. 159, pp. 170-180, 2007.

[18] S. Gibson, J. W. Judy and D. Markovic, An FPGA-based platform for accelerated offline spike sorting, J. Neurosci. Methods, Vol. 215, pp. 1-11, 2013.