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

基於正規化關聯值與 OSort 演算法之棘波分類系統在 FPGA 之實現

Efficient VLSI Architecture for Spike Sorting System Based on Normalized Correlation and OSort Algorithm

研究生：賴柏佑　撰

中華民國　一百零四　年　七　月

中文摘要

本論文希望能在 FPGA(Field Programmable Gate Array) 開發平台上實現棘波分類硬體系統。棘波分類分為三大步驟，棘波偵測、特徵擷取以及分類。此類系統最大的困難點在於，如何正確的偵測到棘波序列中的棘波，以及正確的分類所得到之棘波。特別是在高雜訊的環境之下，很可能會因為雜訊而產生誤判的情形。

本論文提出以正規化關聯值 (Normalized Correlator) 與 OSort 演算法組合而成之棘波分類系統。棘波偵測選用正規化關聯值是因為這個法則在各種強度的雜訊環境之下都有很不錯的表現，較不易受到雜訊的影響。而後面特徵擷取與分類則是選用 OSort 演算法，這個演算法不僅可以一次完成兩個步驟，而且還不需要指定分類的群集數量，是非常具有彈性，且準確度高的演算法，甚至還可以用於即時分類。

本論文最後的成果與效能評估，可證明本系統具備正確偵測並且分類棘波的能力。以軟體 Matlab 程式於個人電腦上運算，並與本論文所實現之系統的結果做比對驗證，可以確保其正確性，並且驗證硬體效能比起軟體運算的效能還要好。

關鍵字：FPGA、棘波分類、棘波偵測、OSort

謝誌

首先，要感謝我的指導老師 黃文吉教授，在研究所的課程中，若沒有他在背後不斷的推動，很多連自己都沒發現的潛能都無法激發出來，並且適時的給予指導、鼓勵，使我在這段時間有明顯的進步。同時也感謝兩位抽空前來參加學位考試的教授們，國立臺北大學資訊工程學系 黃俊堯教授，以及國立臺北科技大學資訊工程學系 楊士萱教授，並且感謝他們所給予的論文指導以及建議。

接著，也要感謝實驗室的學長姐以及同學們提供的資源以及協助，大家一同在這個實驗室中學習與成長。感謝已經畢業的建廷、思淮、一修、奇恩、烝祺、皓棠、光耀、雅姿、煥元，在我學藝不精的時候給予耐心且細心的指導。也要感謝我的同學們，映綸、元品、信豪、建旻、承祐，分享各種知識與技術，並且互相鼓勵支持。最後，感謝下一屆的學弟妹們協助學位口試的事務。

最後，更要感謝我的家人們支持我到台北讀書進修，沒有他們各方面的支持，就沒有今天的成就。謹將此論文的研究成果獻給所有支持我的人們，也希望這份研究能對社會進步有些許微薄的貢獻，也希望大家與我分享這份喜悅與榮耀。

目錄

附表目錄

附圖目錄

第一章　緒論

第一節　研究背景與動機

大腦是人最精密的器官，由神經元、膠質細胞、上皮細胞與血管所組成，負責處理意識、思考、情感、行動等事務的器官，而神經元 (Neuron) 被認為是大腦中的重要成份 [1]，腦部擁有上億個神經元，神經元之間會產生上千種不同的連結排列。神經元之間會發出動作電位 (Action Potential) 來彼此溝通，傳遞想做的動作之控制訊號給各部位的細胞，動作電位又被稱為棘波 (Spike)，為一連串的離子通過細胞膜產生電位差後於神經元細胞間串連成一路徑傳遞訊號，棘波是一段特殊的波形，持續時間約為 1 到 4 毫秒 [2]。分析以及區分不同的動作電位，也就是所謂的棘波分類 (Spike Sorting)，便能解析人類大腦如何運作。

棘波分類是設計腦機介面 (BCI,Brain Machine Interface)[3] 相當重要的一部分，我們需要能解讀棘波所代表的含意，才能將腦波所表示的目的轉換為控制機器的訊號。早從 1970 年代便有腦波的發現，從動物到人體的實驗，應用領域包含醫療、運動訓練、遊戲、義肢輔助等等，但是生物的大腦是一個相當複雜的系統，腦機介面仍未能達到準確的轉換相應的訊號供人類使用，還有相當的改善空間。

在進行棘波分類之前，首先要考慮的是如何取得腦部的訊號，分為侵入式、

半侵入式和非侵入式 (如表1.1)。侵入式的作法是，直接在大腦的灰質中植入電極或是微電極陣列，直接蒐集神經元所發出的訊號，可以取得較純淨的棘波訊號，但是容易引起感染，與免疫排斥等問題。半侵入式則是將電極植於頭骨與灰質之間，雖然訊號上干擾較侵入式多，但較不容易引起感染與排斥問題等等，這兩種方式都必須透過手術才能達成，目前只用在生物還有一些醫學研究之用，還有幫助少數肢體癱瘓的人士，讓這些人們可以利用腦機介面來操作機械義肢，改善日常生活。而非侵入式的方法則是以導電膠塗抹於頭皮，經過電極測量出頭皮的電流反應，可以得到腦電波 EEG(Electroencephalography) 訊號，在 2007 年由美國 Neurosky 公司 [4] 開發出第一款提供一般消費者使用的乾電極腦波偵測器，並且具備相當的應用準確度，如專注值、冥想值的演算，還有各腦波頻帶的紀錄，於台灣電腦應用展上有能控制四軸飛行器飛行的應用。

不過以任何方式取得棘波序列訊號都免不了會有雜訊的干擾，這會嚴重影響到棘波分類的正確率，而且訊號的收集上得到的資料量相當的龐大，根據採樣定理，要還原原始的訊號，採樣頻率要為目標訊號最高頻率之兩倍，而棘波訊號的頻率最高為 12000Hz，所以採樣頻率至少要高於 24000Hz，也就是一秒鐘有 24000 個採樣點數據。若想要即時的分類棘波，不可能以個人電腦或是嵌入式微電腦來進行，結果運算速度上遠不及資料收集的速度，所以本論文希望能設計並實做驗證棘波分類系統的可能性。

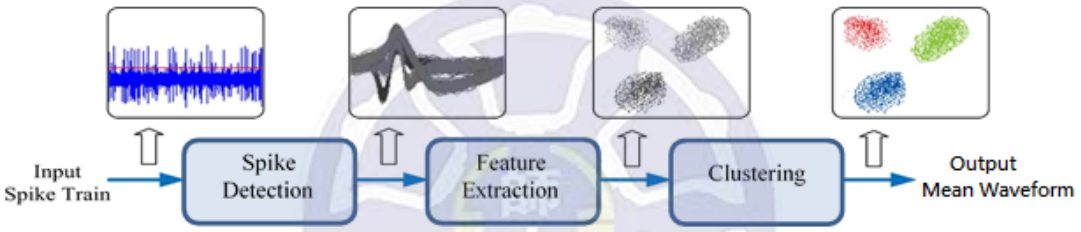
表 1.1: 獲得棘波序列之方法優劣比較

	侵入式	**半侵入式**	**非侵入式**
優點	偵測正確率最高	偵測正確率次高	安全性高，無感染排斥風險
缺點	在大腦灰質植入電極，危險性最高	植入電極於頭骨灰質之間，危險性相對侵入式低	正確率低

第二節　研究目的與方法

棘波分類 (Spike Sorting) 系統主要分為棘波偵測 (Spike Detection)、特徵擷取 (Feature Extraction)、棘波分類 (Clustering)，這三個部份 (如圖1.1)。上一節中有提到過棘波序列的取得方式，配合棘波分類系統後，可以得到分類後的平均波形，了解神經元所控制之動作為何，做各種生體訊號的判斷。接下來將會說明，這三個部份選用哪種法則或是演算法來設計系統架構。

圖 1.1: 棘波分類流程圖



上節有提到各種棘波訊號取得的方式，都免不了會受到雜訊的干擾，所以一個優良的棘波分類系統，第一個步驟，棘波偵測的準確率相當的重要，若棘波偵測時便能將受到雜訊干擾之棘波序列做妥善的處理的話，在後面的分類階段就能更準確的分出該棘波是不是屬於同一個神經元細胞所發出的棘波訊號。現行的棘波偵測方式當中 [5]，約可分為兩種方式 (1) 振幅偵測 (Amplitude) 與 (2) 能量計算 (Energy)，其比較如表1.2所示。

表 1.2: 棘波偵測的兩大種類

方法		**Amplitude**	**Energy**
運算複雜度		低	略高
偵測效果	**雜訊高**	弱	弱
	雜訊低	弱	好

(1) 振幅偵測 (Amplitude)

振幅式棘波偵測方法是一種相當簡單的方式，訂定一個閥值，然後對棘波序列取絕對值，棘波序列的絕對值上有超過該閥值的部份，就將那部份認定為棘波。但這類型偵測法則的缺點則是準確率偏低。

(2) 能量計算 (Energy)

能量計算式的棘波偵測主要是計算各點的關聯值，有一種是計算棘波序列中各點的平方，與相鄰此點的乘積之能量差，最後再判斷這個能量值是否有大於閥值，大於閥值者便是棘波，優點在於偵測的結果比較不會被突然出現的雜訊所影響，像 NEO[6] 便屬於這類的法則。除了 NEO 之外，還有一種是 SWT(Stationary Wavelet Transform)[7][8] 也都是能量計算式的棘波偵測法則。上述之棘波偵測法則的複雜度都不高，效能也具備一定的程度，但是閥值難以訂定，就算能以其他的自動定義閥值演算法 [6][7][9]，也難以提高準確度，還有雜訊越多，偵測準確率越低。

比較上述兩種類型的偵測方法，本論文所設計的方法便使用了能量計算式的棘波偵測法，並且導入 Template 的比較計算，來改善能量計算式的演算法的缺點，這種利用 Template 來計算的演算法中，最典型的就是 Match Filter[10] 演算法。這種演算法利用特定的棘波產生一組 Template，並且利用此 Template 與棘波序列做摺積運算後的值來比對偵測，若該值大於閥值，則代表偵測到的是棘波，這種方法是一種 Correlator(正規化關聯值) 的計算方式。並且在 Correlator 前預先進行棘波序列中的 Segment(區段) 的 Normalized(單位化) 計算，先進行這個運算的話，可以使後續的 Correlator 結果不受棘波序列的訊號強度以及雜訊強度的干擾，皆會維持在一定的範圍之內。如此在 SNR 變低 (雜訊強度高) 的環境之下，也容易選定一個閥值來進行 Template 的比對。所以本論文稱之為 Normalized Correlator，用來作為棘波偵測用法則。

在棘波特徵擷取與分類 [11] 的部份，有些是利用 Principal Component Analysis(PCA)[12] 或是其他類似的法則來做特徵擷取，像是 PCA 與它的變形 GHA(Generalized Hebbian Algorithm)[13, 14]。而分類的方法，則有 K-means、Fuzzy C-means(FCM)[15] 這兩種常見的演算法較為廣泛的應用。上述這些方法可以組合起來完成棘波分類後半部的動作，但本論文所採用的 OSort 演算法則可以一次完成特徵擷取與分類的部份，OSort 演算法可以自動決定 Cluster 的數目，不需要時間做離線訓練 (Offline Training)，即可立即獲得棘波分類的結果，而且 OSort 演算法為一非監督式演算法，可以去除監督式演算法需要設定參數所花費的時間，運算的速度上相較於其他的演算法快，且能夠即時的分類棘波，為其他演算法所不能及的主要優勢，表1.3對各種特徵擷取及分類演算法組合做比較。

根據上面所提到的棘波偵測、特徵擷取、以及分類，這些部份都各有不同的演算法，以及各種參數上的設定，在目前有許多新的棘波分類系統持續被提出的狀況下，在驗證的平台上的選擇就極為重要，考量到快速實現、可擴充性、可修改性，這幾點上，使用 FPGA 開發平台會比起 ASIC 產生客製化晶片而言，來得方便且快速，並且可以根據各種不同的環境參數還有驗證要求來快速的改變電路的布局。所以在這種條件之下，本論文認為選擇用 FPGA 開發平台來當作硬體架構驗證與實做，還有測試平台是個明智的選擇。

表 1.3: 棘波特徵擷取以及分類演算法組合之比較

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

第三節　論文架構

本論文分為五個章節，各章節內容如下列說明。

第一章　緒論（本章）

說明此研究的研究背景、動機以及目的，協助讀者瞭解整個研究之發展過程。

第二章　演算法介紹

介紹本論文中所使用之演算法（包含前文所提及之 Normalized Correlation 與 OSort）概念。

第三章　電路架構與設計

解釋本論文如何以第二章所提及的演算法概念，設計成電路架構，並實做整合成一完整可運作的系統。

第四章　實驗結果與數據分析

介紹本論文所使用之 FPGA 平台，並且分析第三章所提出之電路系統之效能，呈現實際的電路資源消耗數據。此章亦對於棘波分類之產出量做一些評估分析。

第五章　結論

論文總結。

第二章　演算法介紹

根據上一章的內容，本論文介紹了棘波分類 (Spike Sorting) 分為三個部份，並且分析三個部份有什麼樣的演算法適合處理這些部份，從而選出兩個演算法來組合成為本論文欲實現之硬體系統架構。棘波偵測演算法的部份，本論文選擇的是 Normalized Correlator，而特徵擷取與分類這兩個部份，本論文選擇兩部份兼具的 OSort 演算法，這兩個部份組合起來便是一個完整的棘波分類系統，並且在本章來分別且深入的介紹這兩個演算法的原理與細節。

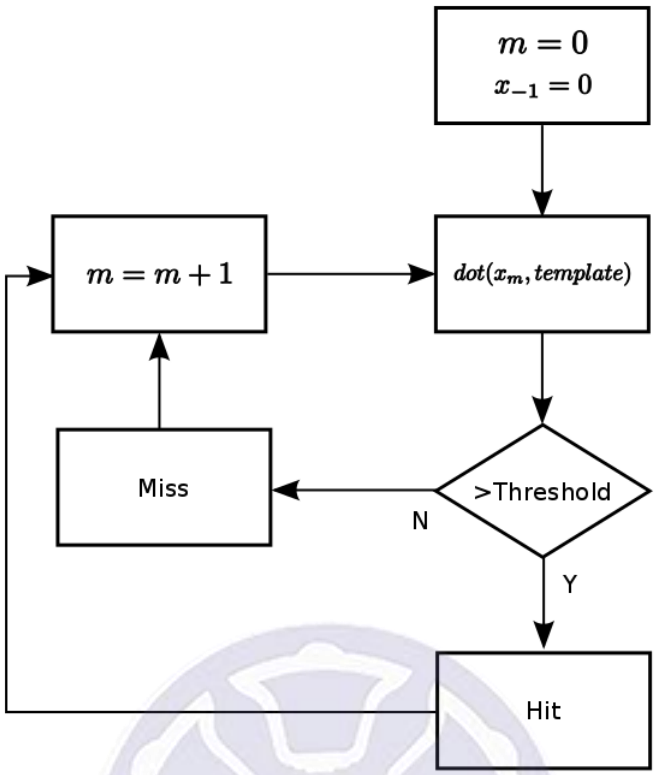
第一節　棘波偵測演算法

(a) Matched Filter

這一節我們會從 Matched Filter 的部份開始討論，Matched Filter 是一個用來進行棘波偵測的簡單架構，其核心的技術是利用多組事先儲存好的 Template 作為循序的與棘波序列 (Spike Train) 的區段 ($x[m]$) 做摺積運算。其運算的流程大致上如圖2.1。

這邊先假設只有一組 Template，方便解說 Match Filter 的作法，以下先定義一些符號。$x[m]$ 為棘波序列中的第 m 個樣本 (Sample)。$x_m = [x[m], x[m-1], ..., x[m-N+1]]^T$ 為棘波序列中第 m 個區段 (Segment)，其中 N 為區段長度，

圖 2.1: Matched Filter 運作流程圖



即是所謂的棘波長度 64 個 Sample。而用於 Matched Filter 的模板同樣擁有 N 個元素 (element)，並將其定義為：$t = [t[1], t[2], ..., t[N-1]]^T$。$y[m]$ 定義為 Matched Filter 在棘波序列中第 m 點的摺積輸出值。

$$y[m] = \sum_{k=1}^{N-1} x[m-k]t[k] = x_m^T t \tag{2.1}$$

在這邊可以觀察到，摺積的計算相當於區段 X_m 與 Template t 做內積的計算，這也相當於摺積所計算出的結果代表區段 x_m 與 Template t 這兩個向量之間的相關性。當 $y[m]$ 比事先訂定的閥值 η 還大的時候 ($y[m] > \eta$)，區段 x_m 就會被判定為棘波。不過 Matched Filter 的缺點也很明顯的是，單一不變的閥值 η 無法在不同 SNR 環境下做出有效的偵測。

(b) Normalized Correlator

前面提到的 Matched Filter 有分析過其缺點，不容易制定理想的閥值是一個極大的挑戰。因此，本論文選擇以另一種方式來制定閥值 η，透過量測 Template t 與區段 x_m 的誤差值上限來決定 η。透過事先訂定其誤差上限，當輸入的區段 x_m 小於誤差上限就判斷其為棘波。這是一種設計 Correlator 的方式，可對應另一種僅用平方誤差的簡單比對方式。

透過下列的平方誤差方程式2.2，運用平方誤差 (Squared Distance) 來觀察區段 x_m 與 Template t 之間的關係:

$$d(x_m, t) = \|x_m\|^2 + \|t\|^2 - 2x_m^T t \tag{2.2}$$

因此，當 $2x_m^T t > \eta$ 時，可得出以下的關係式：

$$d(x_m, t) \leq \|x_m\|^2 + \|t\|^2 - \eta \tag{2.3}$$

觀察方程式2.3，當偵測到 x_m 為棘波時 (i.e., $x_m^T > \eta$)，$d(x_m, t)$ 的上限取決於 $\|x_m\|^2$、$\|t\|^2$ 以及 η，而 $\|x_m\|^2$ 的大小取決於輸入之棘波序列。所以當 $\|x_m\|^2$ 變大時，$d(x_m, t)$ 也會隨之變大，在這種情況下，即使 $x_m^T > \eta$，也會很容易就把雜訊誤判成為棘波，從而產生 False Alarm 的情況。

那麼在本論文取此方法作為應用之前，必須先修改一些運算的過程，將 x_m 以及 t 在進行 Correlation 運算之前就先做 Normalization(正規化) 的動作，這樣的修改可以讓此法則更有效率，如下式所列：

$$\overline{x}_m = \frac{x_m}{\|x_m\|}, \overline{t} = \frac{t}{\|t\|} \tag{2.4}$$

改變之後的 Normalized Correlator 輸出為：

$$\overline{y}[m] = \sum_{k=1}^{N-1} \overline{x}_m[k]\overline{t}[k] = \overline{x}_m^T \overline{t} \tag{2.5}$$

$\overline{x}_m[k]$ 及 $\overline{t}[k]$ 分別為 $\overline{x}_m$ 以及 $\overline{t}$ 之第 k 個 element，所以可以將式2.3改寫為：

$$d(\overline{x}_m, \overline{t}) = 2 - 2\overline{x}_m^T \overline{t} \tag{2.6}$$

在 $d(x_m, t) \leq 1$ 的時候，可得此式：

$$\overline{x}_m^T \overline{t} \leq 1 \tag{2.7}$$

所以 Normalized Correlator 是指 $\overline{x}_m$ 與 $\overline{t}$ 這兩個向量上的計算。當 $\overline{x}_m^T \overline{t} \geq \eta$ 時，就可以得知此時對應到的 x_m 就是被偵測到的棘波，故從2.7可以推出 $\eta \leq 1$。另外在 $x_m^T t > \eta$ 時，根據2.6可以得出下式：

$$d(\overline{x}_m, \overline{t}) \leq 2(1 - \eta) \equiv D \tag{2.8}$$

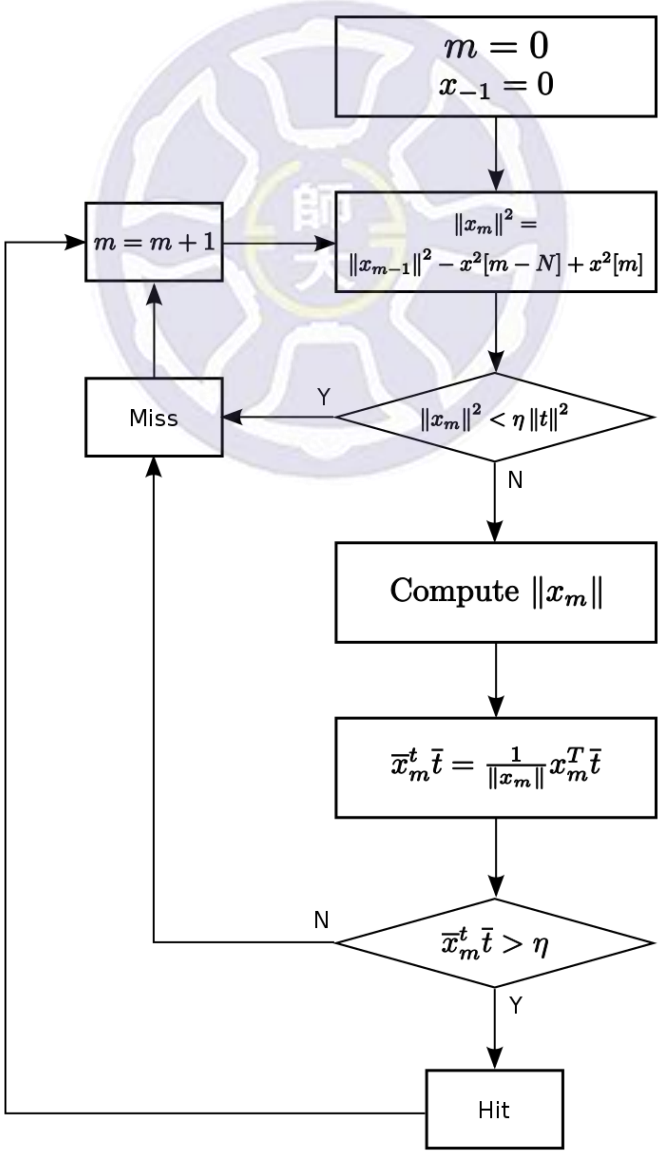
D 可以表示為 Hit 時的平方誤差上界，經由 Normalized Correlator 後可以得到唯一 η，當作 Template Matching 用的平方誤差上界。除此之外 $d(\overline{x}_m, \overline{t})$ 之值取決於 η 的大小，故當 η 變大時則 $d(\overline{x}_m, \overline{t})$ 變小。當閥值 η 為最大值 1 時，且並不會受到輸入的棘波序列所影響。

實際上，在 $\eta = 1$ 時，作為判斷輸入的棘波區段 x_m 還有 Template t 是否吻合的平方誤差最大值為 $D = 2(1 - \eta)$。這種 Normalized Correlator 的方法對於 η 有了新的解釋。如果在這邊將 η 設定為，也就是當區段 x_m 與 Template t 做運算之後，其值超過 η ($\overline{x}_m^T \overline{t} \geq 1$) 才算是偵測到棘波的話，代表該區段 x_m 與 Template t 有完全相關性 (Full Correlation) 時才會被視為偵測到一個棘波，這時兩者之間的平方誤差為 0。換一個閥值而言，當 $\eta = 0.5$ 時，就表示區段 x_m 與 Template t

運算後之值大於 η ($\overline{x}_m^T \bar{t} \geq 0.5$) 就被判定為偵測到棘波，此時兩者之間的平方誤差上限為 1。那把 η 設定為 0 的話，就表示區段 x_m 與 Template t 毫無相關性 (No Correlation)，表示只要有區段 x_m 進來就算是偵測到了棘波，此時它們之間的平方誤差上限值為 2。

但是在真實的測量環境之下，一定會有雜訊的干擾，所以不能採取完全相關性的條件 $\eta = 1$ 來作為偵測標準，根據實驗所得的結果，即使在雜訊很強的環境之下，亦只需要 70% 的相關性 (i.e. $\eta = 0.7$)，就可以讓本論文提出的 Normalized Correlator 偵測效能達成高 True Positive，低 False Positive 的效果。

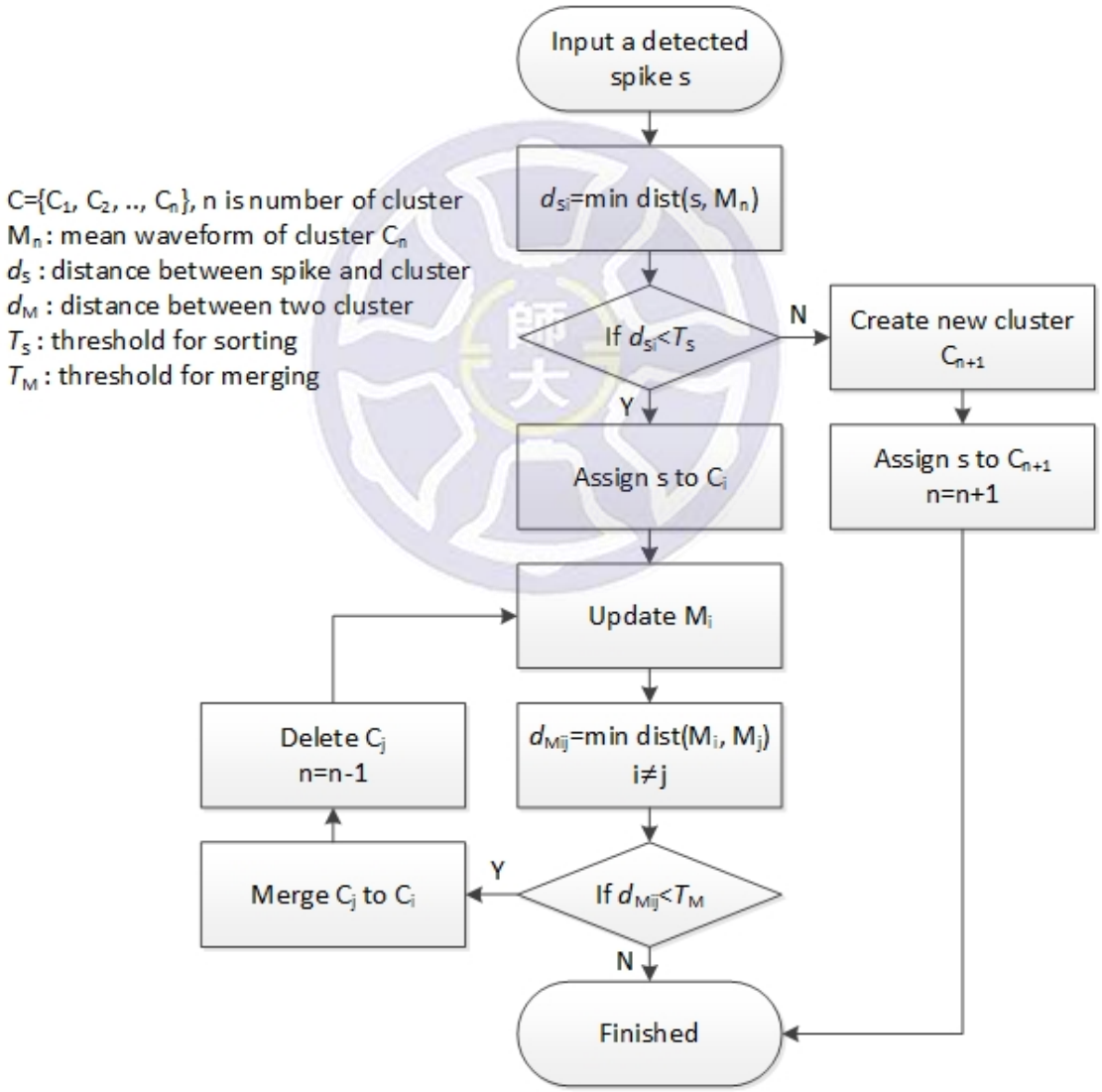
圖 2.2: Normalized Correlator 流程圖



第二節　OSort 演算法

OSort 是一個為了做即時棘波分類並能立即得知分類結果所設計之演算法，由 Rutishauser 在 2006 年提出 [16]。不同於以往需要分為特徵擷取與分類兩個部份分別運算，以模板比對 (Template Matching) 的方式來達成目的，沒有複雜的計算，以簡單的運算來實做棘波分類的能力，流程如圖2.3所示。

圖 2.3: OSort 演算法流程



一開始先做閥值得運算，T_S 作為分類 (Sorting) 的用途，而 T_M 則作為合併 (Merging) 的用途，Rutishauser 提出了兩種計算閥值的方法，分別為近似法，以及估算法：

(1) 近似法 (Aproximated Method)：$T_S = T_M = T$，T 的計算方式如公式2.9，σ_r 為通過濾波器後之訊號的平均標準差，N 為一個棘波的 Sample 個數。

$$T = N(\sigma_r)^2 \tag{2.9}$$

(2) 估算法 (Estimation Method)：需要使用到卡方分配來計算 T_S 與 T_M，擁有比較高的正確率，但是公式較前者複雜許多，故不在此列舉，有興趣之讀者可以參考 [16]。

C 為群集 (Cluster) 之集合，假設目前存在 n 個群集，$C = C_1, C_2, ..., C_n$，每個群集的平均波形值為 $M_i, i = 1, 2, ..., n$，輸入一個被判定為棘波者 s 時，計算 s 與所有群集 C_i 之平均波形 M_i 的距離 (相似度)，如式**??**，找出其中被計算出擁有最小距離者 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 與目前所分類出來的群集相似度較低，則需要建立一個新的群集來存放，也就是棘波 s 分類到新的群集中。反之，若最小的距離 d_{Si} 小於 T_S，表示此棘波 s 與群集 C_i 相似度高，則將棘波 s 分類到被計算出與之距離最短的群集 C_i。

將 s 與 C_i 中的數據重新計算來更新群集的平均波形 M_i 後，換計算此群集 C_i 與其於群集 C_j 的平均波形 M_j 的距離，由公式2.11中可以得知這樣算出來的距離結果提供了兩個群集之間的相似度依據。若最小的距離 d_{Mij} 小於 T_M，則代表這兩個群集相似度極高，則將這兩個群集做合併的動作，重複上述的步驟，持續計算與合併，直到此群集 C_i 與其餘的群 C_j 之距離皆大於 T_M 之後棘波分類結

束，即為 OSort 演算法運算結束。

上述的流程中可得知 OSort 演算法並不需要設定群集的個數有幾個，也就是不需要得知偵測到的棘波該分為幾群，它可以自動的將棘波分為若干群。由於棘波偵測法則的偵測正確率不可能達到百分之百正確，故在這邊會需要過濾掉一些偵測錯誤的棘波還有 Outlier，只要將存在較少棘波數量的群集刪除，便可以做到濾除的功能。

$$d_s(s, M_i) = \sum_{k=1}^{N} (s(k) - M_i(k))^2 \tag{2.10}$$

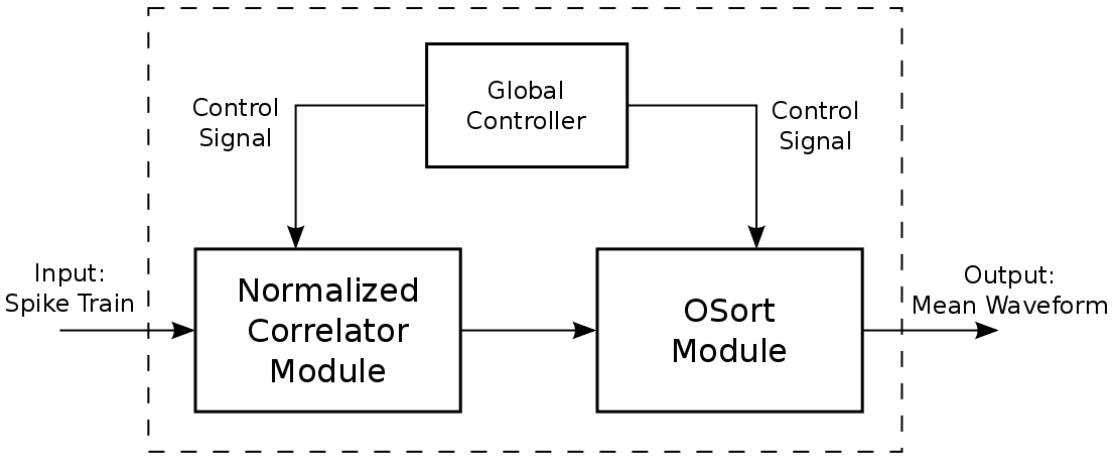
$$d_M(M_i, M_j) = \sum_{k=1}^{N} (M_i(k) - M_j(k))^2 \tag{2.11}$$

第三章　電路架構與設計

第一節　電路架構概觀

本論文所實做之系統分為三個部份，第一個部份是棘波偵測電路，這邊採用 Normalized Correlator 法則作為棘波偵測電路實現的基礎。第二個部份是 OSort 演算法為基礎實現的電路，如第二章第二節所提到的，OSort 演算法可取代傳統的棘波分類方式中的特徵擷取以及分類的部份。最後，第三個部份是一個整合用的控制電路 (Global Controller)，可以控制棘波偵測電路部份的輸出，與 OSort 演算法電路部份的輸入，這些控制訊號可以有效整合兩者間的訊號傳遞問題，以達到使系統效能提昇之目的。

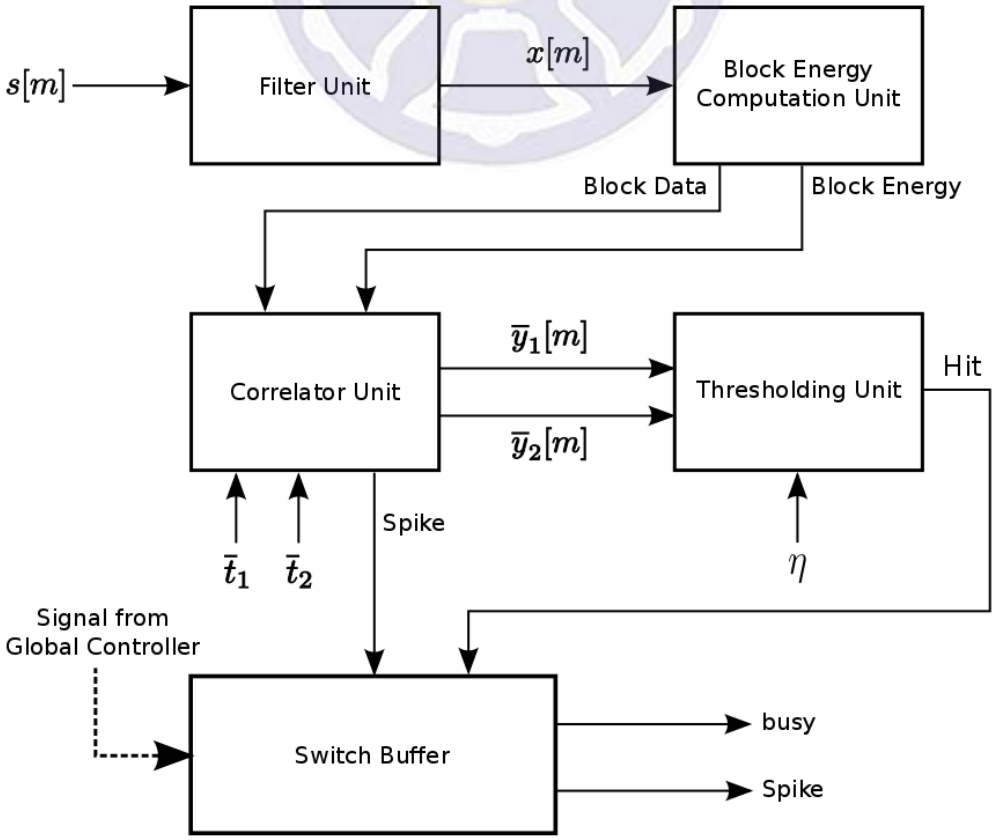
圖 3.1: 本論文所實做之電路系統概念圖



第二節　棘波偵測電路

本節介紹的是本論文所採用 [17] 之棘波偵測系統的電路架構，本架構中包含四大單元，(如圖3.2) 棘波偵測系統有 Filter Unit、Block Energy Computation Unit、Correlator Unit、Thresholding Unit。Filter Unit 的部份會去除棘波序列 $s[m]$ 中的直流成份與高頻雜訊，棘波序列 $s[m]$ 通過 Filter Unit 後之輸出 $x[m]$ 將作為 Block Energy Computation Unit 之輸入，做平方以及累加的運算，得到 $\|x_m\|^2$。接著 Correlator Unit 會將前兩個 Unit 的輸出 $x[m]$ 以及 $\|x_m\|^2$ 與事先準備好的兩個 Template 分別是 $\overline{t_1}$ 和 $\overline{t_2}$ 做 Correlation 的計算，得到輸出為 $\overline{y_1}[m]$ 以及 $\overline{y_2}[m]$。最後，Thresholding 會將 $\overline{y_1}[m]$ 以及 $\overline{y_2}[m]$ 兩者與閥值 η 做比較，若任何一者大於閥值就判斷為棘波。

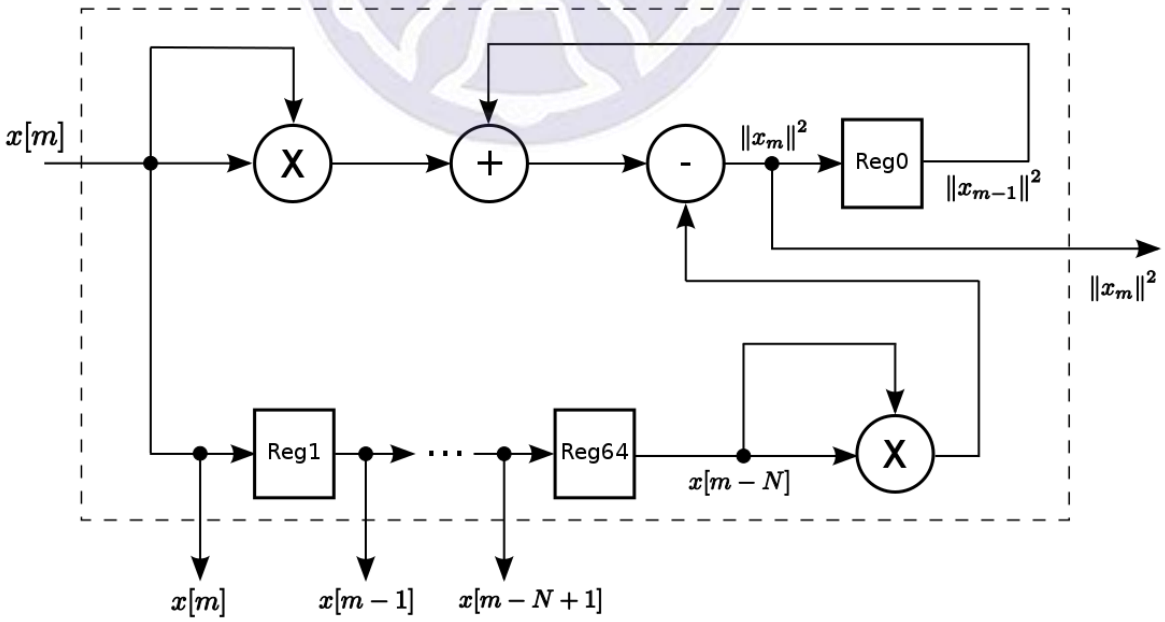
圖 3.2: Normalized Correlator 棘波偵測硬體架構



(a) Block Energy Computation Unit

Block Energy Computation Unit 的電路架構 (如圖3.3)，可以從中看出其運算過程，這個架構是以加速演算法 Fast Energy Computation 為基礎所設計，可以有效的降低 Area Cost。電路中 $x[m]$ 資料寬度為 8-bits 之棘波序列採樣點，Reg1 至 Reg64 是寬度為 8-bits 之暫存器，用以儲存棘波序列 64 個採樣點，而 Reg0 則是 16-bits 之暫存器。Block Energy Computation Unit 會先將 Filter Unit 輸出的訊號 $x[m]$ 存入 64 階的位移暫存器中，並將 $x[m]$ 的值作平方運算，交由後面的邏輯電路與 Reg0 中的值做累加的運算，再減掉 Reg64 中的值之平方，再次存入 Reg0，透過不斷的累加上次的 $\|x_m\|^2$ 結果並同時輸出 $\|x_m\|^2$ 給下一個 Unit 作為輸入，就可以達到 Sliding Window 的效果。

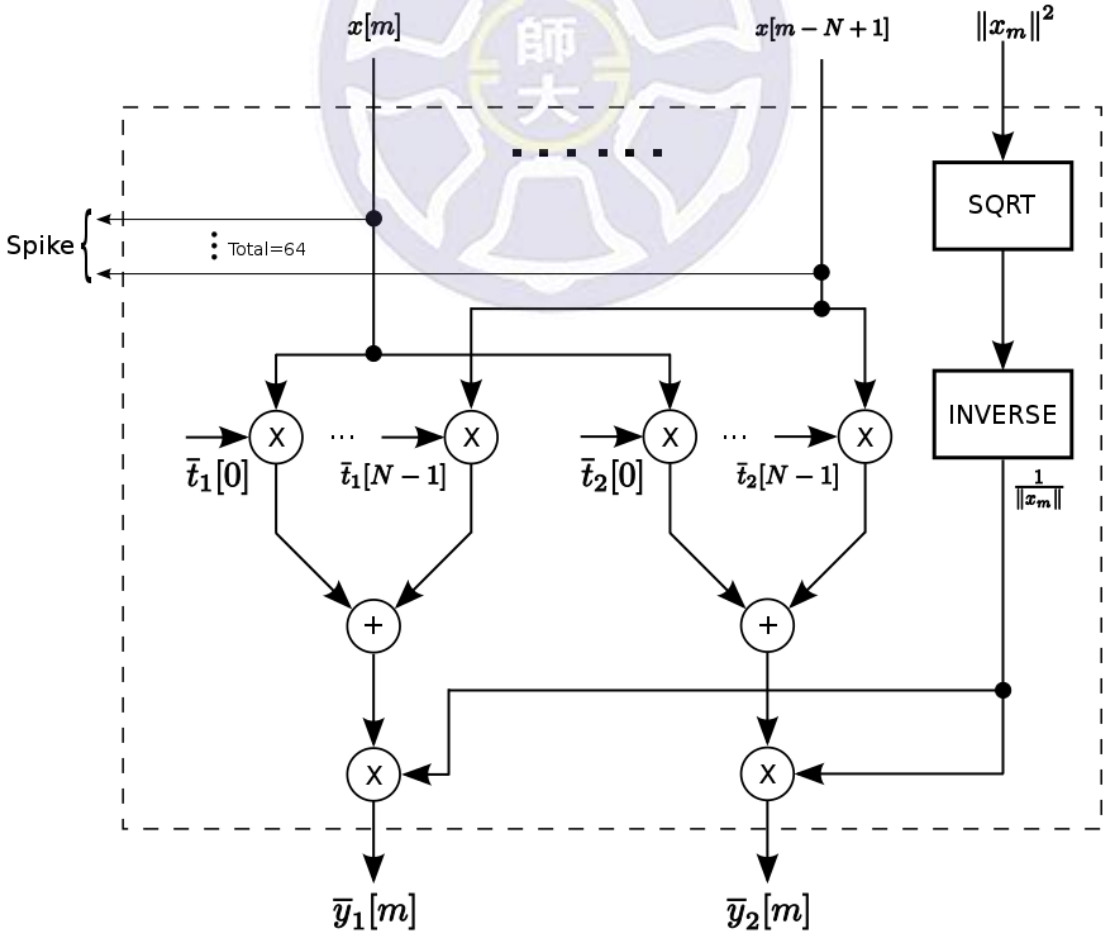
圖 3.3: Block Energy Computation Unit



(b) Correlator Unit

Correlator Unit(如圖3.4) 會將 Block Energy Computation Unit 的輸出 $\|x_m\|^2$ 經由 SQRT 電路以及 INVERSE 電路運算後得到結果為 $\frac{1}{\|x_m\|}$，同時也會將另一個輸入，棘波序列 $x[m]$ 與事先準備好的兩個 Template，$\overline{t_1} = [\overline{t_1}[1], ..., \overline{t_1}[64]]^T$ 和 $\overline{t_2} = [\overline{t_2}[1], ..., \overline{t_2}[64]]^T$ 做內積運算。最後會將兩個內積後的結果，分別乘以 $\frac{1}{\|x_m\|}$，得到 $\overline{y_1}[m]$ 與 $\overline{y_2}[m]$，這段運算是一個簡化架構的作法叫做 Post-correlation Normalization，相較於 Pre-correlation Normalization 作法是 64 個 $x[m]$ 採樣點先與 $\frac{1}{\|x_m\|}$ 相乘，減少了 62 個 8-bits∗8-bits 乘法器，亦可減少後面邏輯電路所需的位元數。

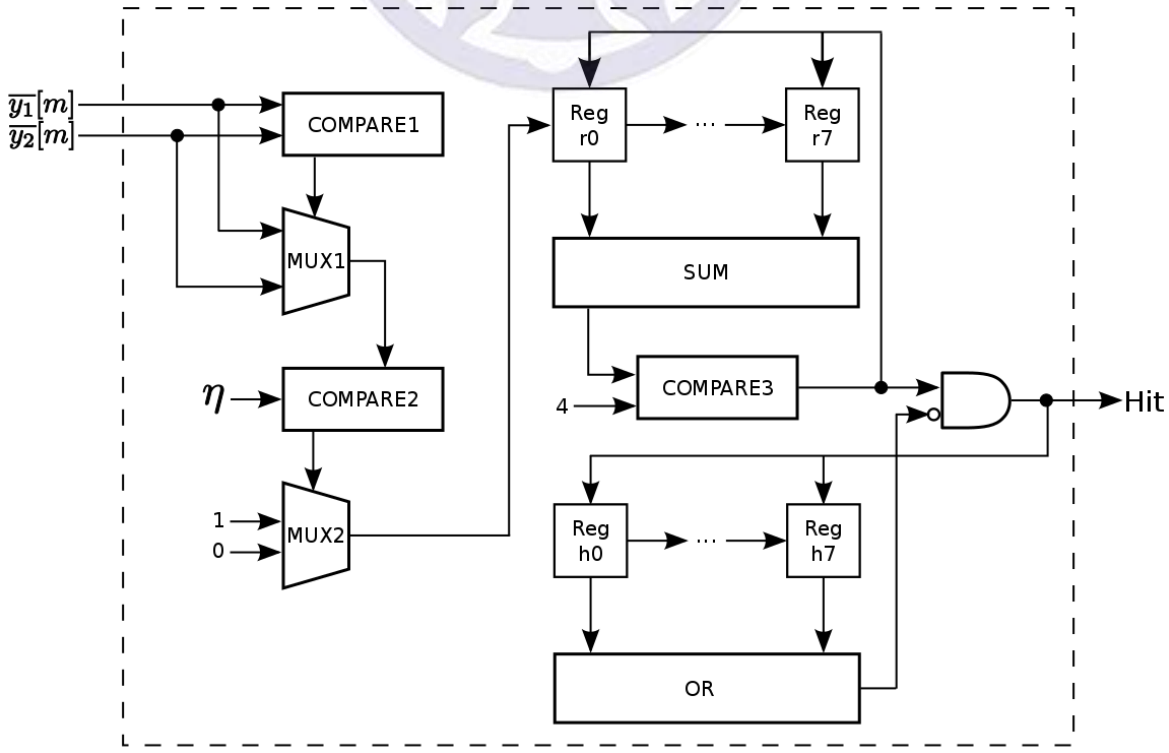
圖 3.4: Correlator Unit



(c) Thresholding Unit

Thresholding Unit(如圖3.5) 會以 Correlator Unit 的兩個輸出 $\overline{y_1}[m]$ 和 $\overline{y_2}[m]$ 為輸入，找出這兩者中，何者與模板 $\overline{t_1}$、$\overline{t_2}$ 相關性較高，先比較 $\overline{y_1}[m]$ 和 $\overline{y_2}[m]$，看哪一個比較大，數值較大者再與事先給定的閥值 $\eta = 0.7$ 做比較，大於該值的話便可判斷其為 Hit，即偵測到棘波。在偵測的過程中，可能會有沒完全消除之雜訊干擾，故產生不連續且大於閥值的 Correlation 值，這邊的電路架構上，利用了一組 8 階的位移暫存器 (Reg r0 到 Reg r7)、加法器與比較器 (COMPARE3) 來減少誤判的可能。當 MUX2 的輸出為 1 時，代表 Subhit 一次，將位移暫存器中的 Reg r0 設定為 1，之後的比較器 (COMPARE3) 會去檢查這組 8 階位移暫存器中有幾個 Reg 被設定為 1，如果有 4 個以上的 Reg 被設定為 1 時，代表有 4 次以上 Subhit 才認定為一次 Hit。最後還有另一組 8 階位移暫存器 (Reg h0 到 Reg h7) 加上 or 邏輯，用來排除相同的棘波在一段時間內被重複判定的情形。

圖 3.5: Thresholding Unit



(d) Switch Buffer

Switch Buffer 用來暫存在 Correlator Unit 中偵測到之棘波資料，由圖3.6可以發現，這邊有兩個暫存區塊 Buffer x 以及 Buffer y，大小相同用來交替接收棘波資料，當一者負責接收時，另一者就可以負責輸出棘波資料給後面的 OSort Algorithm 硬體電路做運算。雖然這樣做的電路面積會增加，但是考慮到兩個電路可以同時間運算，不需要長時間互相等待，處理棘波序列的速度將會大大提昇，減少需要中斷等待資料消化的時間。

這邊有幾條控制訊號來處理資料的輸入與輸出，如 x_write_en 以及 y_write_en 是用來控制哪一個暫存區塊目前是負責接收棘波的。而 Select 則是負責控制輸出棘波的。不過棘波偵測的速度上仍有可能快過 OSort Algorithm 硬體電路處理分類的速度，故設計了一個 busy 的訊號，負責通知外部輸入棘波序列的控制者 (如 Processor 或 DMA 之類的 Controller) 需要暫停棘波序列的輸入，這條 busy 訊號的訊號平常是輸出 0 的，但是當兩個暫存區塊皆透過 x avail 以及 y avail 通知已經存滿棘波資料，若再輸入會蓋過暫存起來的棘波資料。

最後是 switch 訊號會影響 x_write_en 跟 y_write_en，如果 switch 輸出為 0，表示將偵測到的棘波資料存入 Buffer x，反之當 switch 輸出為 1 則將棘波資料存入 Buffer y。而決定 switch 輸出的訊號是一個額外的小控制器，如圖3.7。

圖 3.6: Switch Buffer

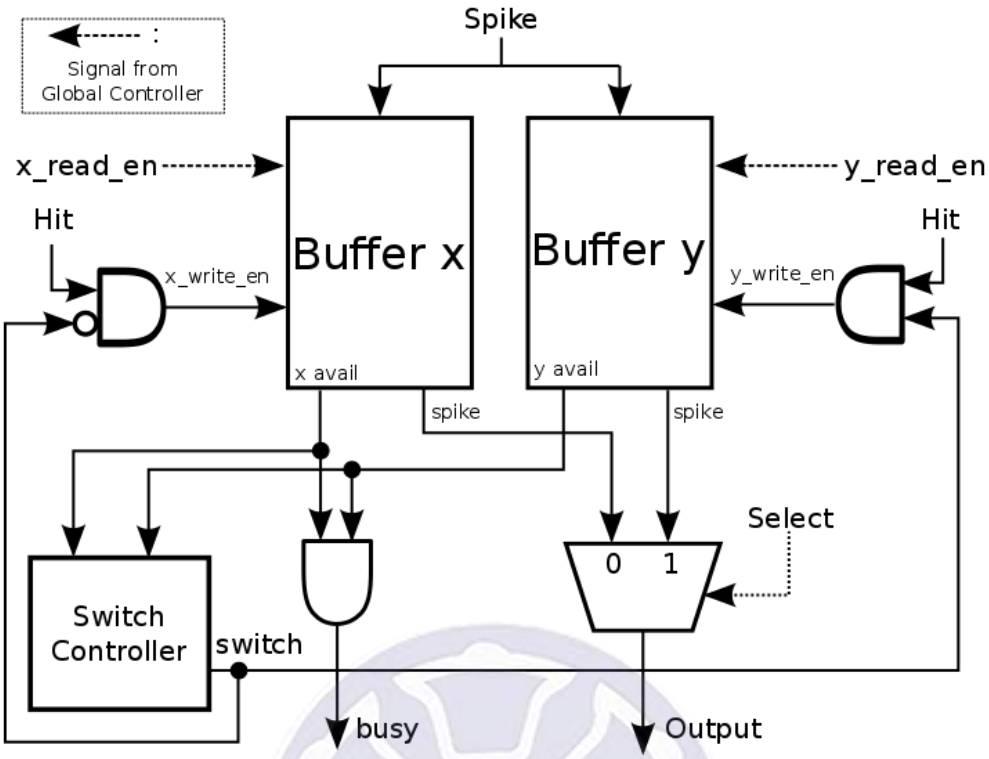
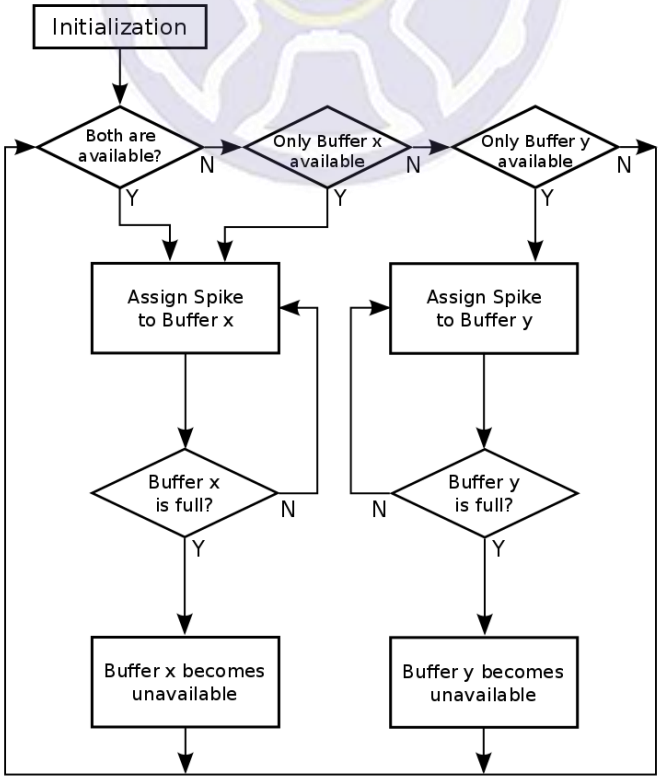


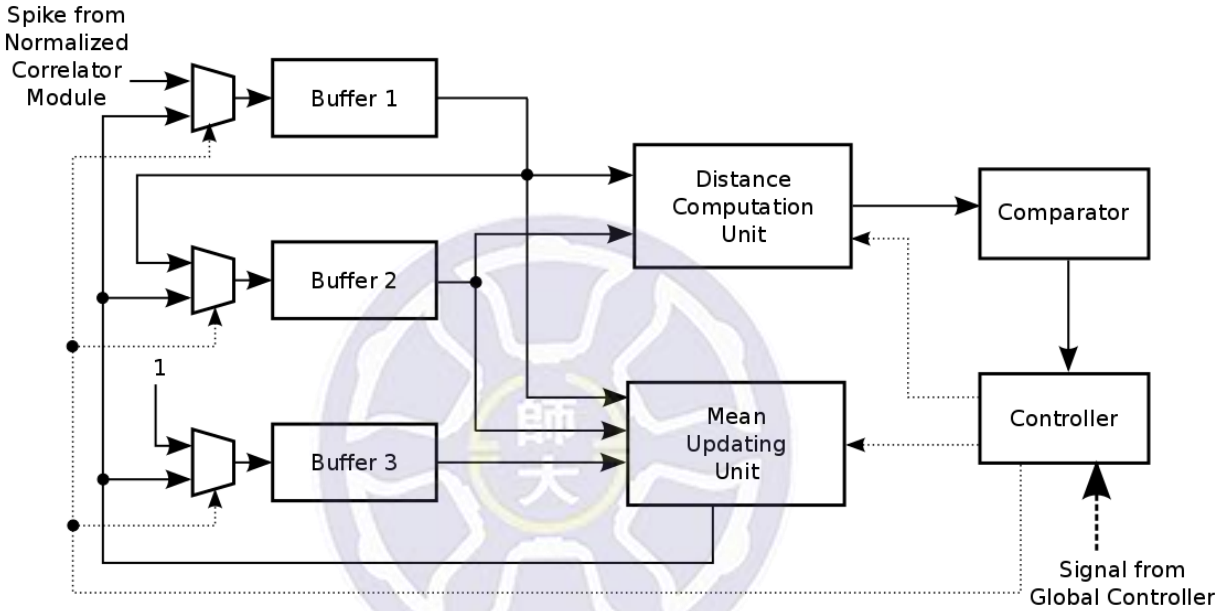
圖 3.7: Switch Controller 狀態圖



第三節　OSort 演算法電路

OSort 演算法的電路架構為本節的重點，採用 [18] 所設計的電路架構，分為三個緩衝區 (Buffer 1, Buffer 2, Buffer 3)、Distance Computation Unit、Mean Updating Unit、Comparator、Controller，這部份整體的電路架構如圖3.8所示。

圖 3.8: OSort 演算法硬體架構



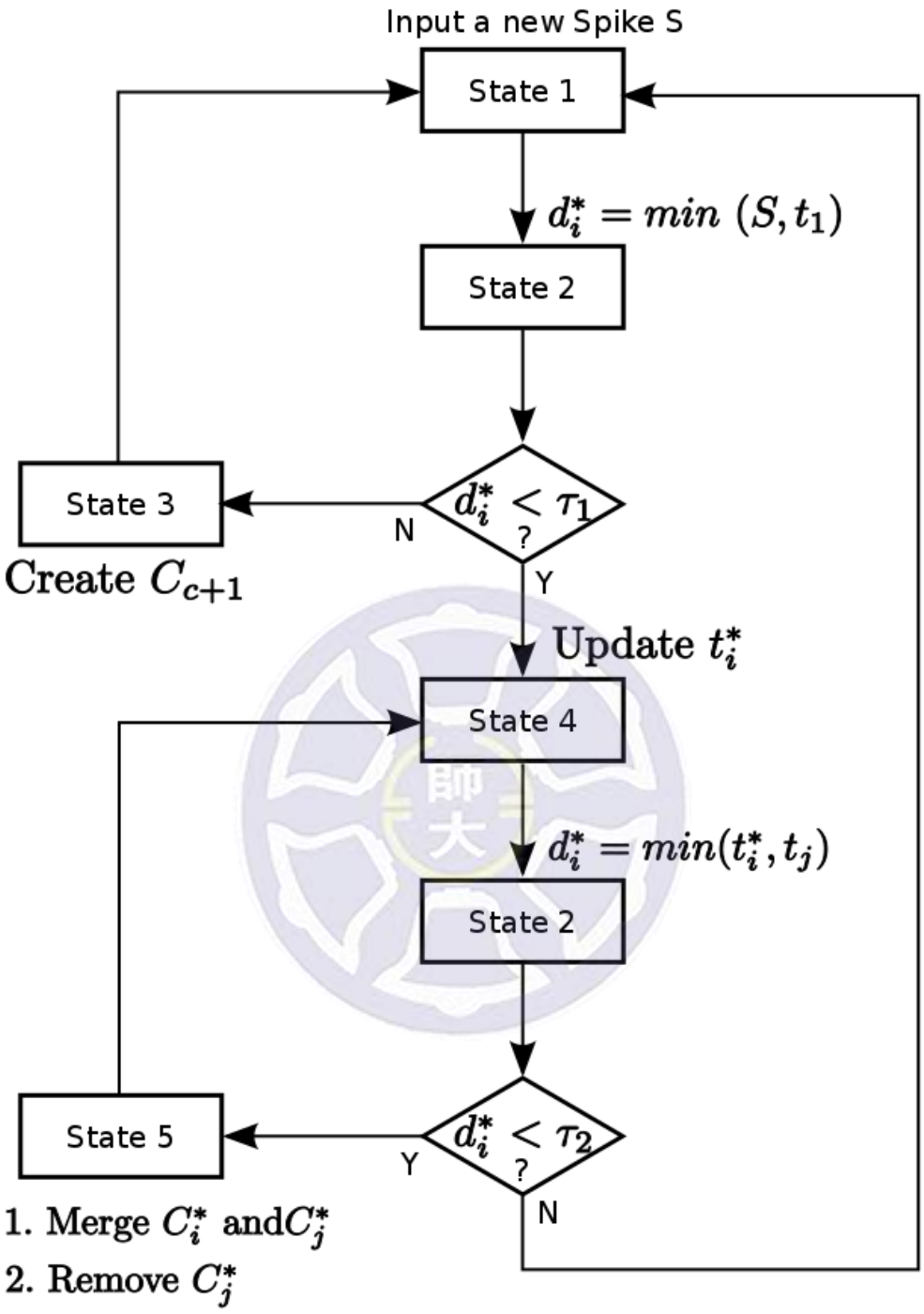
OSort 演算法電路架構中，由於運作流程有一定程度的變化，需要額外的控制電路來輔助各部份進行資料運算，各單元中的資料運算流程分為五個狀態，如表3.1所列。狀態一，電路會抓取棘波放入 Buffer 1 中暫存。狀態二，Distance Computation Unit 會計算，並且尋找與 Buffer 1 中的棘波最相似的群集 (C)。狀態三，建立一個新的群集來儲存找不到相似群集的棘波。狀態四，更新平均波形並且群集會累計棘波的數量。狀態五，當兩個群集平均波形過於接近時，會移除相似的群集。

表 3.1: OSort Controller 之狀態表

States	Activated Compenents	Operations
State 1	Buffer 1	Fetch Spike to Buffer 1.
State 2	Buffer 1, 2 Distance Computation Unit	Find the best matching Cluster to the Spike in Buffer 1.
State 3	Buffer 1, 2, 3	Creating a new Cluster.
State 4	Buffer 1, 2, 3 Mean Updating Unit	Update the mean and size of a Cluster.
State 5	Buffer 2, 3	Remove a Cluster.

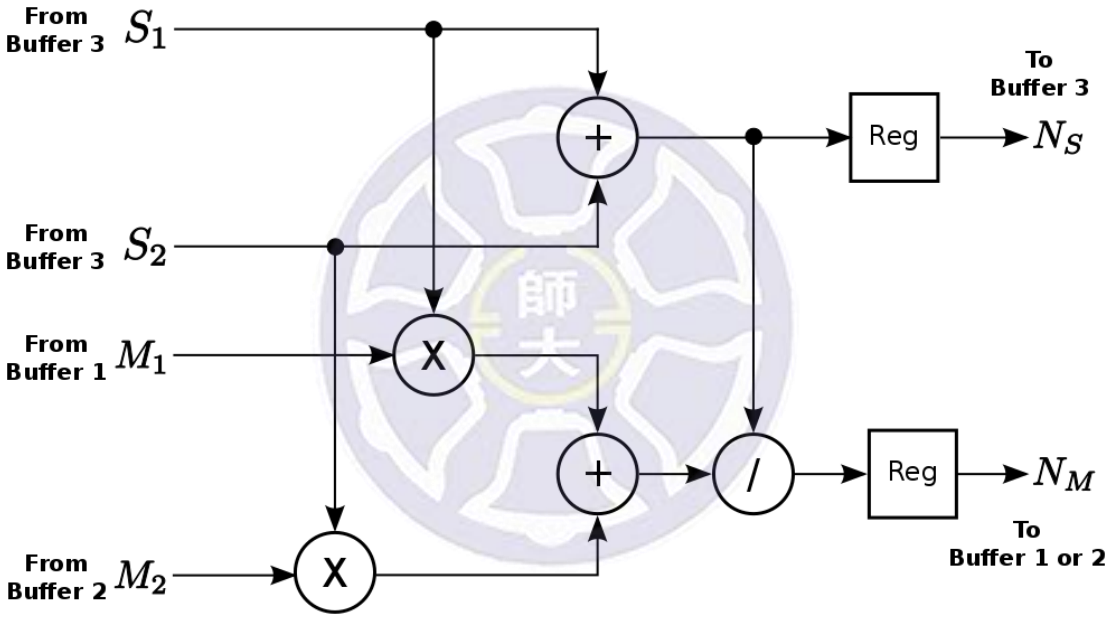
如圖3.9可以看到詳細的 OSort 硬體電路中 Controller 是如何運作的，在狀態一，輸入一筆新的棘波 S 到 Buffer 1 中。接下來進入狀態二，Distance Computation Unit 會將 Buffer 1 中的 Spike 與 Buffer 2 中的所有群集計算，得到最小平方距離，再與閥值 τ_1 做比較，如果最小平方距離比閥值 τ_1 還要大則表示為新的群集。進入狀態三，建立一個新的群集來儲存這筆棘波，之後回到狀態一再輸入下一筆棘波。但最小平方距離比閥值小時，則會被納入現有群集中，於是進入狀態四，更新現有的群集之平均波形以及累計群集中的棘波數量。更新完成之後，再到狀態二中做群集之間的相似度比對，如果群集相似度高者，則其最小距離平方會比閥值 τ_2 還要高，於是合併兩個群集 C_i^* 還有 C_j^*，並移除被合併者 C_j^*。反之群集相似度低的話，最小距離平方會比閥值 τ_2 還低，就不需要做群集的合併，回到狀態一，取下一筆棘波進來運算。

圖 3.9: OSort Controller 狀態流程



由於 Mean Updating Unit 在本節的 OSort 演算法電路中是相當重要的運算部份，故在這邊特別的提出來說明，如圖3.10所示。此單元為棘波被分類到某個群集時，或是兩個群集的平均波形極為相似，要做群集合併時所需的運算單元。負責重新計算群集的平均波形。S_1 及 S_2 代表群集中的棘波數量，M_1 及 M_2 代表群集的平均波形。首先分別將 S_1 與 M_1 相乘還有 S_2 與 M_2 相乘來還原數據，再將 S_1 還有 S_2 的值相加得到 N_S。之後將還原所得之波形數值除以 N_S，得到新的平均波形 N_M，完成群集的平均波形的更新。

圖 3.10: Mean Updating Unit

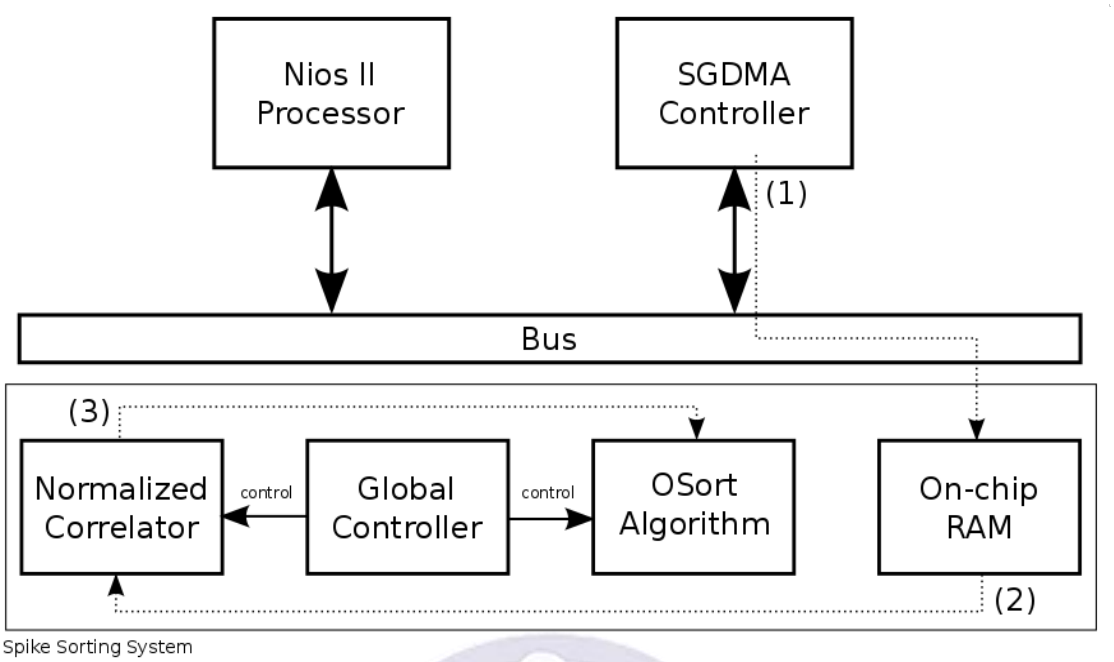


第四節　整合棘波分類系統

本節要討論的重點則是如何將前兩節所提到的 Normalized Correlator 硬體電路部份與 OSort 演算法硬體電路兩者相結合，並且搭配 Global Controller 與 FPGA 系統元件來將這兩個硬體電路整合成一個系統。

本論文所實現之系統搭配 DE4-230 FPGA 開發平台上所提供之各硬體單元，可以組成一個系統架構如圖3.11所示，棘波分類系統可分為 Normalized Correlator 單元、Global Controller 單元以及 OSort Algorithm 單元。這邊大略說明一下資料處理的流程。本論文利用 SGDMA Controller 單元來控制存放於 On-chip RAM 上的棘波序列的資料，能快速的傳輸棘波序列的資料給 Normalized Correlator 單元偵測棘波。接下來，由 Global Controller 來控制 Normalized Correlator 單元的輸出，使 Normalized Correlator 單元中由 Switch Buffer 暫存的棘波資料能輸出給 OSort Algorithm 單元，同時間也必須控制 OSort Algorithm 單元的輸入，才能完成棘波資料的傳遞。最後可以從 OSort Algorithm 單元的群集暫存區中讀出平均波形資料，即完成棘波分類。

圖 3.11: 棘波分類系統架構圖



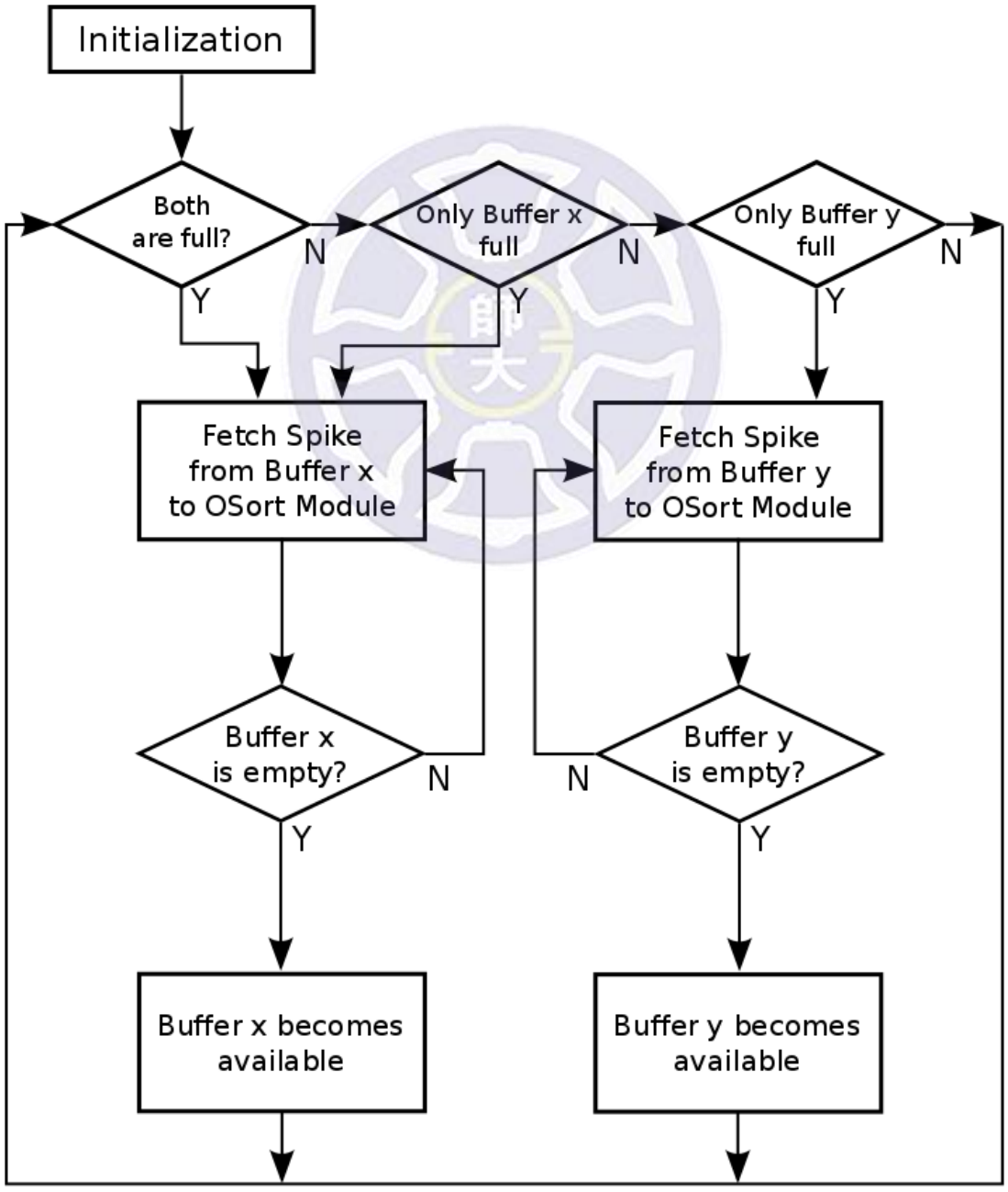
本論文所實現之系統在 Normalized Correlator 單元有一 Switch Buffer，如圖3.4所示，這部份是用來作為暫存 Normalized Correlator 單元計算後被偵測為棘波的資料暫存部份，在本章第二節中有詳細的說明 Switch Buffer 的電路架構。還有 OSort Algorithm 單元中的 Buffer 1 輸入棘波資料的控制訊號也需要由其他電路提供。所以，這些電路都由一個叫做 Global Controller 的單元發出訊號來控制它們，這個 Global Controller 單元的控制流程、狀態如圖3.12所示。

初始化之後，Global Controller 單元會先等待 Buffer x 的部份存滿棘波資料，待存滿之後會由 Global Controller 單元發出控制訊號，去使 Normalized Correlator 單元輸出暫存的棘波資料並令 OSort Algorithm 單元來輸入棘波資料，每傳輸一筆棘波資料都會檢查一次，是否已經將 Buffer x 中的棘波資料都讀出來運算過，如果已經取完 Buffer x 中所有的棘波資料就換等待 Buffer y 的部份存滿棘波資料。之後 Buffer y 的部份存滿了棘波資料就再一次重複傳送棘波資料的控制訊號，直到 Buffer y 中的棘波資料也都運算過，再回到等待 Buffer x 的部份存滿棘

波資料的狀態。

由這些單元的整合之後，可以發現本論文所實現之系統，控制相當的簡單，僅需事先輸入 Normalized Correlator 單元中要使用的 Template，在這之後便不需要額外的控制，一直輸入棘波序列來做運算即可，而且加上 SGDMA Controller 之後，棘波序列的輸入速度加快許多，比起由 Nios II Processor 來給予位址傳遞棘波序列還快。在下一章的討論中會提供實際測試的數據供讀者參考。

圖 3.12: Global Controller 單元狀態圖



第四章　實驗結果與數據分析

第四章呈現本論文的研究成果。第一節，介紹使用的開發平台與實驗環境，以及使用的驗證模擬工具。第二節，分析系統中消耗的硬體資源。第三節，分析系統效能，並測量產出效率等等。

第一節　開發平台與環境

本論文使用的開發平台為 Altera 公司所生產的 DE4-230 EP4SGX230KF40C25 來實現棘波分類系統之硬體架構，如圖4.1所示。這個 FPGA 平台是發展與教育 (DE) 系列，擁有許多的週邊應用元件，具有相當研究與應用價值，也因為其提供了大量的邏輯單元及記憶體資源，方便我們將棘波偵測電路與 OSort 電路整合在一起。另外，選擇 FPGA 開發平台來實現與驗證電路的理由是因為其擁有可程式化系統晶片 (System on Programmable Chip, SoPC)，具有可以重複修改電路的彈性，對於本論文的設計驗證相當合適，因此選擇這類型的平台。

詳細的規格說明，如表4.1說明本論文所使用的 FPGA 開發平台的規格，而表4.2說明的是本論文實驗與實做的開發環境，以及一些驗證工具的版本列表。

圖 4.1: Altera DE4-230 FPGA 平台

表 4.1: Altera DE4-230 EP4SGX230KF40C25 FPGA 平台規格表

DE4-230 FPGA 平台規格表	
Family	Stratix IV GX
Device	EP4SGX230KF40C25
Adaptive Logic Modules (ALMS)	91,200
Logic Elemnts(LEs)	228,000
Total Memory Kbits	17,133
Embedded 9x9 Multipliers	1,288
Total PLL	8
Total DLL	4
User I/Os	744

本論文使用 Altera Quartus II 13.1 作為 Verilog 硬體描述語言之 EDA Tool，這個工具提供了基礎的語法解析與錯誤偵測、時序分析、初步的邏輯元件繞線布局、產生規劃檔案、以及電路合成等功能，方便快速的建立系統，並且可以連結 ModelSim 做快速的 RTL-Level 以及 Gate-Level 圖形波型模擬，來驗證電路的正確性。當系統建立完成後，便可掛載到 SoPC 上，而在 Quartus II 13.1 中所使用的元件彙整平台是 Qsys，可以快速的選擇想要使用的系統元件，例如，CPU、Onchip Memory、DMA、客製化電路 (棘波偵測、OSort、整合元件)，最後將這些元件掛載到 SoPC 上便可以實際的輸入訊號，與擷取輸出結果，驗證整體的系統架構與正確性。

最後，由以 Eclipse 為基礎之 Nios II IDE 作為控制電路以及擷取電路輸出訊號的工具，它提供設計人員在 Nios II 嵌入式系統中所需之軟體工具、驅動程式及函式庫，令系統的開發更有效率，方便驗證。

表 4.2: 各項開發環境

硬體開發環境	
FPGA	Altera DE4-230 EP4SGX230KF40C25
軟體開發環境	
CPU	Intel I7-3770 @ 3.40GHz
Memory	DDR3 32GB
Operating System	Linux 3.18.12-gentoo
開發工具	
Synthesiser	Quartus II 64-bit version 13.1
Simulation Tool	ModelSim-Altera 10.1d
Software Build Tool	Nios II IDE
Verification Tool	Matlab R2014a

第二節　系統偵測效能分析

本論文使用 [19] 所提出的細胞外紀錄模擬器，可以用在模擬不同的棘波序列，提供各種不同參數給使用者作設定，如棘波種類、採樣頻率、神經元個數以及 SNR(訊噪比，Signal to Noise Ratio)，它除了可以產生近似於實際神經元所發出的訊號以外，更提供了標準數值 (Ground Truth)，確立基準來判斷棘波分類演算法之優劣，並能進一步的評估本論文所實做系統之效能。本論文實驗使用之採樣頻率為 24000Hz，也就是每一秒有 24000 個採樣點，而每個棘波訊號長度為 2.67ms，是為 64 個採樣點。由於實際的棘波採樣中容易受到環境中的雜訊干擾，故 SNR 如公式 (4.1) 及公式 (4.2) 所示，是一個相當重要的議題，這個數值表示訊號和雜訊的比值，單位為分貝 (dB)，SNR 越高代表雜訊越少，訊號品質越好，反之則越差，受到雜訊的干擾嚴重。

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

$$SNR_{dB} = 10log_{10}\frac{P_{signal}}{P_{noise}} \tag{4.2}$$

由於本論文所實現的系統中，Normalized Correlator 硬體電路部份，是根據 [17] 中所設計之硬體架構實現的，而在 [17] 第四章第四小節的數據內容上，可以更明確的評估本論文所提出之演算法的優劣，透過表4.3比較了現行各種棘波偵測法則，在各種 SNR 下的 TP Rate 還有 TA Rate 的表現。實驗用之棘波序列都是內含兩種棘波形狀的棘波序列，棘波之長度為 20 秒。

值得注意的是，[17] 提出的演算法在第一階段是為了產生模板，主要的偵測效能表現是觀察第二階段 (開始利用模板做偵測) 而來，所以表4.3顯示的結果是第二階段的效能。而 Match Filter[10] 則是假設已經先行準備好需要的模板才進行效能觀測。

從表4.3還可以觀察到，[17] 所提出的的回饋式演算法透過自行適應的模板產生跟簡化閥值選擇，在各個 SNR 的狀況下的表現都優於其他的演算法。就算是跟已經假設有準確模板之 Match Filter 相比較，準確率與失誤率都高出它一些。這是因為 Match Filter 的閥值需要隨 SNR 及棘波峰值的變化來更動，所以要選出一個理想的閥值更加的困難，但單一閥值更不可能適用於包含各種強度雜訊的棘波序列。

雖然能量偵測為基礎之 NEO[6] 還有 SWT[7] 法則在高 SNR 下有良好的表現，但是隨著 SNR 的下降，雜訊提高使得兩個演算法的效能急速的下降，在 SNR=-3 時，NEO 和 SWT 的 FA Rate 高達 50%。與此相對的，[17] 提出的演算法在 SNR=-3 時，FA Rate 卻只有 1% 而已，這個結果證明了 [17] 所提出的演算法在高雜訊的狀況下仍有不錯的表現。

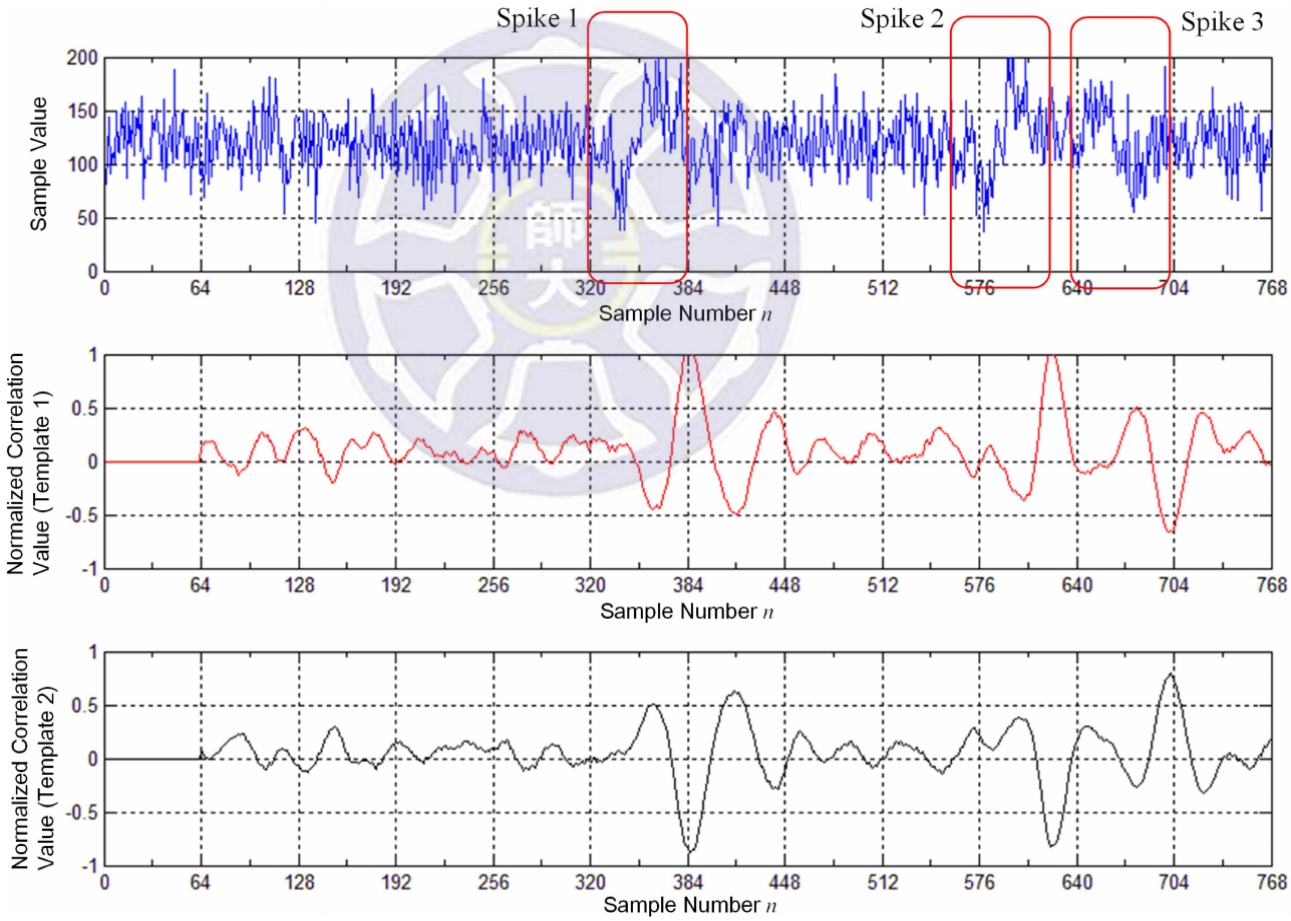
表 4.3: 各種棘波偵測法則在不同雜訊等級的棘波序列下之 TP Rate 與 FA Rate 表現

SNR(dB)		-3	-2	0	1	8	10
Proposed	TP Rate	82.71%	82.39%	88.22%	90.04%	93.04%	93.64%
Algorithm	FA Rate	1.06%	0.41%	0.59%	0.92%	0.36%	0.40%
NEO	TP Rate	80.53%	82.05%	83.47%	87.21%	92.24%	93.10%
Algorithm	FA Rate	57.87%	56.16%	39.62%	22.49%	7.75%	3.57%
SWT	TP Rate	86.66%	87.38%	91.37%	92.43%	94.26%	94.82%
Algorithm	FA Rate	82.43%	81.80%	79.00%	79.36%	19.00%	6.77%
Absolute	TP Rate	22.22%	51.08%	54.65%	67.81%	81.74%	90.90%
Value[9]	FA Rate	3.70%	0.84%	1.52%	1.25%	12.44%	17.96%
Matched	TP Rate	80.31%	80.83%	82.20%	82.90%	86.32%	89.65%
Algorithm	FA Rate	8.92%	7.69%	4.62%	3.02%	2.94%	2.80%

接下來在圖4.2中可見本論文提出的法則 Normalized Correlator，是如何由

棘波序列中偵測出棘波的，若單純觀察棘波序列，幾乎不太可能直接找出何者為棘波。但若加入已知的棘波 Template 1，便可以由 Sample Number=320 到 Sample Number=384 這一段中找到波形類似於 Template 1 中的一段，還有 Sample Number=576 到 Sample Number=640 這附近也有一個棘波，總共可以由 Template 1 比對出兩個棘波。再由 Template 2 中可以比對出第三個棘波約在 Number=640 到 Sample Number=704 附近。本論文提出的法則大致的運作方式便是如此，由 Template 當作對照組，並從其中比對出相似者，就能找出範例中這 768 個 Sample 中有哪些是棘波。

圖 4.2: SNR=-3dB 且使用兩個 Template 之 Normalized Correlator 範例



第三節　系統資源分析

本論文結合 [17] 的 Normalized Correlator 硬體電路以及 [18] 的 OSort 演算法硬體電路，並且以模擬器所產生之數據作為測試資料，將兩者以第三章第四節所述之控制電路整合控制，故產生了可以連續輸入棘波序列的整合電路。實做上棘波偵測選用 Normalized Correlator，相較於其他的棘波偵測法則，對於 SNR 相當的敏感，準確率與誤判率都須在高 SNR 為前提之下才行，但 Normalized Correlator 具有在不同 SNR 環境下能高準確率，低誤判率偵測的優點，配合能同時具備特徵擷取還有分類能力的 OSort 演算法，且此演算法還具有能夠即時分類 (online sorting) 的能力。

在這邊先分析兩個硬體電路的面積複雜度分析，Normalized Correlator 硬體電路分為 Filter Unit、Block Energy Computation Unit、Correlator Unit、Thresholding Unit 以及 Switch Buffer 幾個部份，而 OSort 演算法硬體電路則為 Buffer、Distance Computation Unit 以及 Mean Updating Unit 幾個部份。如表4.4所示，N 為棘波之長度 (64 個 sample)，亦可稱作 Block Dimension，而 q 為 Template 之個數，目前是設計為兩個 Template，還有 L 則是指 Switch Buffer 的大小，也就是 Switch Buffer 可以儲存幾個被 Normalized Correlator 電路偵測出來的棘波，根據這些面積複雜度的分析來看，L 的大小對整體面積的影響相當的大，並且也會影響到下一節要提的的效能，是本章的重要討論之一。

表 4.4: 各單元面積消耗分析

(a) Normalized Correlator Circuit

	Filter Unit	**Block Energy Comp. Unit**	**Correlator Unit**	**Thresh. Unit**	**Switch Buffer**
Adders	$O(1)$	$O(1)$	$O(qN)$	$O(1)$	0
Multipliers	$O(1)$	$O(1)$	$O(qN)$	$O(1)$	0
Dividers	0	0	$O(1)$	0	0
Comparators	0	0	0	$O(1)$	0
Registers	$O(1)$	$O(N)$	$O(qN)$	$O(1)$	$O(LN)$

(b) OSort Algorithm Circuit

	Buffer	**Distance Computation Unit**	**Mean Updating Unit**
Adders	0	$O(1)$	$O(1)$
Multipliers	0	$O(1)$	$O(1)$
Dividers	0	0	$O(1)$
Comparators	0	$O(1)$	0
Registers	$O(qN)$	$O(1)$	$O(1)$

緊接著實際的將這些電路繞線布局，由 EDA Tool 中觀察真實的硬體資源消耗，整理成表4.5，與上一張表格所分析預估之硬體資源消耗比例相符，且可以明顯的發現，本論文所實做之系統中消耗最多硬體資源的部份是 Normalized Correlator 電路中的 Switch Buffer 以及 OSort 演算法電路中的 Buffer，由於改變 Normalized Correlator 電路中的 Switch Buffer 會直接的造成系統的處理結果產出的變化，在之後有更深入的討論，將會列出各種不同的 Switch Buffer Size 的組合的面積表現，還有處理結果產出的量化表格，也會討論到為什麼 Switch Buffer Size L 會對電路的效能產生影響。而 OSort 演算法電路中的 Buffer 主要是儲存群

集 (Cluster) 之平均波形 (Mean Waveform) 所用，這直接的表示了 OSort 演算法電路的最大分群數量上限，在實現本系統時，可以考量到採樣目標的需求，來調整最大分群數量上限，這部份也可以減少一些電路面積與硬體資源消耗，本論文所實做之系統是以 32 個群集為例。

表 4.5: 各單元實際硬體資源消耗情形

(a) Normalized Correlator Circuit (Switch Buffer Size L = 40)

	Filter Unit	Block Energy Comp. Unit	Correlator Unit	Thresh. Unit	Switch Buffer
ALUT	552	637	3289	52	8560
Registers	196	237	1130	12	28366
Memory bits	0	0	0	0	0
DSP Blocks	0	2	530	0	0

(b) OSort Algorithm Circuit

	Buffer	Distance Computation Unit	Mean Updating Unit
ALUT	20682	1168	473
Registers	20842	1020	682
Memory bits	0	0	320
DSP Blocks	0	86	2

有別於以往兩個硬體電路之間缺乏有系統的控制，導致 OSort 演算法的部份必須等候 Normalized Correlator 的部份計算完畢，才能向其要求所偵測到的棘波訊號。整合後的系統只需要由外部輸入棘波序列即可，故整體產出大幅度的提昇。實現這個能力的關鍵在於，兩個硬體電路間的緩衝區設計，在第三章第四節中提到，兩者間須要一組 Switch Buffer，一者負責接收偵測到

的棘波，一者負責傳遞給 OSort 演算法的部份進行運算，在表4.6中，列出各種緩衝區大小所消耗的資源量，由此表可見，ALUT 與 Register 的使用量都會隨著緩衝區大小變化，緩衝區越大則使用量會越大，這部份的資料將會與下一節的處理時間來做交叉比較，由此可以選出一個適當的緩衝區大小。

表 4.6: 緩衝區資源消耗

	ALUT	**Registers**
Switch Buffer Size L **= 80**	17120	56720
Switch Buffer Size L **= 52**	11130	36868
Switch Buffer Size L **= 40**	8560	28366
Switch Buffer Size L **= 32**	6850	22685
Switch Buffer Size L **= 20**	4281	14178
Switch Buffer Size L **= 16**	3424	11345

如表4.6所示，本論文所實做之硬體電路擁有一組的 Switch Buffer 作為緩衝區，其資源消耗佔了 Normalized Correlator 硬體電路很大的一部分，可見 Switch Buffer 越大則消耗的 AUL 越多，而硬體面積是很重要的指標之一，應取面積小且處理速度快的方案來實做硬體電路，下一節的效能中將會量測處理時間與緩衝區大小的關係，有利於選擇適當的緩衝區大小方案。

如表4.7列舉出所有電路之資源消耗，從個別電路 Normalized Correlator 的硬

體電路以及 Sort 演算法硬體電路，還有整合兩者之控制電路、FPGA 平台上所提供之系統元件、兩個主要的處理硬體電路資源加總，從表中亦可觀察到整合用的控制電路面積相當的小，卻能使整體系統效能提昇，具有能連續長時間輸入棘波序列的能力。

表 4.7: 各電路的資源消耗 (with Switch Buffer Size L=40)

	ALUT	**Registers**	**Memory bits**	**DSP Blocks**
OSort Circuit (upper limit with 32 clusters)	22604	22562	320	88
Normalized Correlator Circuit	13966	29733	0	530
OSort and Normalized Correlator Circuit	39355	36517	320	642
Global Controller	40	20	0	0
Nios CPU +onchip RAM+JTAG	2638	819200	4	0

最後是表4.8，將本論文所實做之系統與其他論文比較，由於 [20] 中所提出之 OSort 演算法之硬體架構使用的是 PIPO(Parallrel-in-Parallel-out) 的資料傳輸方式，會增加消耗資源，而本論文所實做之系統是以 SISO(Serial-in-Serial-out)

的方式做資料傳輸，能減少資源消耗。[20] 所使用的 FPGA 開發平台的品牌為 Xilinx，與我們使用的 Altera 所開發之 FPGA 開發平台，它們所定義設計之邏輯電路不盡相同，故在此說明個別的差異處。在乘法器的部份 DSP 類似於 Multiplier，本論文所使用之 Altera FPGA 開發平台使用 18-bits 單位乘法器而 [20] 則是使用 25x18 乘法器，所消耗之資源較多一些。而 ALUT 的部份，Xilinx 是以 slice LUTs 為單位，約等於兩個 ALUT。而本論文所實做之系統在 Memory bits 的使用量上，比起 [20] 少了很多。在這些比較之下可知，各類型的硬體資源消耗綜合起來與 [20] 差不多。

本論文亦實做以 NEO 取代 Normalized Correlator 的部份，做一個面積的估算，由於 NEO 的乘除運算相較於 Normalized Correlator 少了非常多，故可以發現取代後，DPS Blocks 的使用量大幅度的降低到與 [20] 相近，而 LUT 消耗量也有小幅度的下降。

表 4.8: 本系統與其他論文比較

	LUT	**Memory bits**	**DSP Blocks**
Proposed (L=40)	39355	320	642
Hardware proposed by [20]	47134	225000	29
NEO-Proposed (L=40) (NEO replace Normalized Correlator)	38529	320	79

第四節　系統速度分析

本論文所實做之系統，可以正常運作於 50MHz、75MHz、100MHz 三種時脈之下，本論文輸入之測試資料長度為 1 秒的棘波序列，也就是 24000 個採樣點，並測量執行時間，之後彙整4.6所觀察到的 Logic Elements 消耗如表4.9所示(本表所測量之處理時間及產出是運作於 50MHz 時脈之下的結果)，可以清楚的觀察到 ALUT 的消耗與處理時間的關係，由此可知 Switch Buffer 越大則處理時間越短，但是也不能無限的增加，Logic Elements 使用過多的話功耗與面積都會變得太大。

另外，由於硬體架構上的關係，於 Switch Buffer 暫存之棘波數量佔滿之後，才會開始傳遞給 OSort 演算法的部份，這也是不能讓 Switch Buffer 過大的原因之一，過大會導致 OSort 有一段時間沒有棘波可以處理，閒置過久。還有表4.6中可觀察到，由 Switch Buffer Size L=16 擴大到 Switch Buffer Size L=20 時，處理速度大幅度的提昇了，原因是暫停外部棘波序列輸入的次數過多。在系統的硬體架構中，棘波偵測的速度可能稍快於 OSort 演算法電路處理的速度，故在緩衝區被佔滿的狀態下，系統會暫停外部的棘波序列輸入，避免之後所偵測到的棘波遺失，沒有被儲存到緩衝區內。

表 4.9: 系統執行時間彙整

(@ 50Mhz)	**Processing Time (Duration = 1 Sec)**	**Throughput**
Switch Buffer Size L **= 80**	1.803 ms	13.327 MBps
Switch Buffer Size L **= 52**	2.155 ms	11.145 MBps
Switch Buffer Size L **= 40**	2.341 ms	10.263 MBps
Switch Buffer Size L **= 32**	2.457 ms	9.56 MBps
Switch Buffer Size L **= 20**	2.603 ms	9.23 MBps
Switch Buffer Size L **= 16**	2.941 ms	8.17 MBps

表4.10是本論文所實做之系統，所有緩衝區大小在不同的時脈環境下的處理速度表現，未來修正一些硬體架構中資料路徑過長的部份，可以用更高的時脈環境來驅動硬體電路，可望再提昇硬體電路的處理結果產出，目前可以看到在 Switch Buffer Size L=80 且驅動時脈環境在 100MHz 的情況下，處理結果的產出量可以達到 25MBps，相較於 Switch Buffer Size L=16 且驅動時脈環境在 50MHz 的情況下，處理結果的產出量大上了三倍有餘，證明了緩衝區越大且時脈環境越高則本論文所實做之系統效能越好。

表 4.10: 不同運作時脈下的處理結果產出量

	50 MHz	**75 MHz**	**100 MHz**
Switch Buffer Size L = 80	13.311 Mbps	20.001 Mbps	25.035 Mbps
Switch Buffer Size L = 52	11.136 Mbps	16.701 Mbps	21.473 Mbps
Switch Buffer Size L = 40	10.252 Mbps	15.374 Mbps	20.255 Mbps
Switch Buffer Size L = 32	9.768 Mbps	14.652 Mbps	19.349 Mbps
Switch Buffer Size L = 20	9.219 Mbps	13.001 Mbps	17.795 Mbps
Switch Buffer Size L = 16	8.159 Mbps	12.238 Mbps	16.954 Mbps

在本節最後，表4.11中將本論文所收集的各種數據做個比較，尤其是針對與軟體運算的結果做比較。在本論文的研究中所使用的軟體為 Matlab 做本系統的軟體模擬與驗證，運行於四台硬體配備不同的電腦分別做效能評測 (與硬體比較產出量)，三台桌上型電腦，在表格中都有大略提到主要的配備規格，另一台則是筆記型電腦，也有提到配備規格，在軟體上運作的效能其實差距不大。然後硬體的部份則是取本論文所作之實驗中產出量結果最不差的硬體環境參數狀況，Switch Buffer L=20，Clock Rate=50MHz，以及最好的硬體環境參數狀況 Switch Buffer L=80，Clock Rate=100MHz，與先前提到的四台電腦上的軟體產出量結果做一個比較作結，可以看出各種狀況下的結果產出量差異。

表 4.11: 比較處理結果產出量

Software	
	Throughput
Intel I7-3770@3.4GHz, RAM 32GB	0.823 MBps
Intel I5-3570@3.4GHz, RAM 16GB	0.723 MBps
Intel I7-930 @2.8GHz, RAM 16GB	0.692 MBps
Intel I5-420M@2.1GHz, RAM 8GB	0.627 MBps
Hardware	
	Throughput
***L*=20 , Clock Rate=50MHz**	8.159 MBps
***L*=80 , Clock Rate=100MHz**	25.035 MBps

第五章　結論

本論文成功的在 FPGA(Field Programmable Gate Array) 開發平台上實現基於正規化關聯值與 OSort 演算法之棘波分類硬體系統，並且擁有一定程度的運算速度，可用於快速的將棘波序列做分類，而且經過驗證後具備一定的準確度。本論文所提出之系統架構，具備有在各種程度的雜訊強度下，都能有穩定的由棘波序列偵測出棘波並且分類群集數正確的優勢，而且不需要事先設定群集數，相較於其他種類硬體系統有更高的彈性。

然而，兩個不同功能的運算單元電路的整合也是本論文所實現之硬體系統中的一大挑戰，不僅需要設計一組緩衝區 Switch Buffer，還需要一個 Global Controller 來對兩個運算單元電路做同步的訊號控制。這樣的整合讓整體系統的控制非常簡單，僅需輸入棘波序列，兩個運算單元中間的控制訊號都會自動的處理，系統的使用者可以輕鬆的操作。

此外，本論文所實現之硬體系統也與在軟體上運作的系統做比較，證明了實現出來的硬體系統效能是遠高於軟體上運作的系統，也能證明硬體系統具有能快速棘波分類的能力。未來亦可能採用本論文所設計之架構改良，來實現腦機介面協助特定人士的需求。

第六章　參考著作

[1] J. H. Schwartz E. R. Kandel and T. M. Jessell. Principles of Neural Science. 2000.

[2] Sarah Gibson, Jack W. Judy, and Dejan Markovic. Spike Sorting: The first step in decoding the brain. *IEEE Signal Processing Magazine*, 29(1):124–143, January 2012.

[3] David Ewing Duncan. The Brain-Computer Interface That Let a Quadriplegic Woman Move a Cup. http://www.theatlantic.com/health/archive/2012/05/the-brain-computer-interface-that-let-a-quadriplegic-woman-move-a-cup/257275/, May 2012.

[4] NeuroSky Inc. http://neurosky.com/.

[5] M. S. Lewicki. A Review of Methods for Spike Sorting: The Detecion and Classification of Neural Action Potentials. *Netw. Comput. Neural Syst.*, 9:R53–R78, 1998.

[6] S. Mukhopadhyay and G.C. Ray. A new interpretation of nonlinear energy operator and its efficacy in spike detection. *Biomedical Engineering*, 45(2):180–187, February 1998.

[7] K. Kim and S. Kim. A Wavelet-based Method for Action Potential Detection from Extracellular Neural Signal Recording with Low Signal-to-Noise Ratio. *IEEE Trans. Biomed. Eng.*, 50:999–1011, 2003.

[8] M. Appalsamy R. J. Brychta, S. Tuntrakool. Wavelet Method for Spike Detection in Mouse Renal Sympathetic Nerve Activity. *IEEE Trans. Biomed. Eng.*, 54:82–93, 2007.

[9] R. Quian Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Computation*, 16(8):1661–1687, August 2004.

[10] L. S. Smith N. Mtetwa. Smoothing and Thresholding in Neural Spike Detection. *Neurocomputing*, 69:1366–1370, 2006.

[11] Wen-Jyi Hwang, Wei-Hao Lee, Shiow-Jyu Lin, and Sheng-Ying Lai. Efficient Architecture for Spike Sorting in Reconfigurable Hardware. *Sensors*, 13(11):14860–14887, November 2013.

[12] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag New York, 2nd edition, 2002.

[13] Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, April 1989.

[14] Simon O. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2008.

[15] H. Ichihashi S. Miyamoto and K. Honda. *Algorithm for Fuzzy Clustering*. Springer-Berlin Heidelberg, 2010.

[16] U. Rutishauser. Online detection and sorting of extracellulary recorded action potentials in human medial temporal lobe recordings. *J. Neurosci. Methods*, 154:204–224, 2006.

[17] 王思淮. Spike Detection Based on Normalized Correlation with Automatic Template Generation. Master's thesis, National Taiwan Normal University, 2014.

[18] 徐雅姿. Hardware Implementation for Spike Sorting Based on NoC with OSort Algorithm. Master's thesis, National Taiwan Normal University, 2014.

[19] Leslie S. Smith. A tool for synthesizing spike trains with realistic interference. *Journal of Neuroscience Methods*, 159(1):170–180, January 2007.

[20] J. W. Judy S. Gibson and D. Markovic. An FPGA-based platform for accelerated offline spike sorting. *Journal of Neuroscience Methods*, 215(1):1–11, January 2013.