第四章 實驗數據與效能比較

本章將呈現本論文所提出的架構實際效能量測的實驗數據與效能比較。首先要看的是，一個向量量化器編碼系統採用我們提出內嵌於 Nios 處理器的部分距離搜尋專用硬體電路，在不同的模組參數 *M* 和碼字數量 *N* 的情況下實現時所需花費的面積複雜度，量測結果如表 4-1 所示。

LEs		**N**		
		64	**128**	**256**
M	**1**	*7804*	*8240*	*9277*
	2	*9337*	*10266*	*12340*
	3	*10870*	*12256*	*15362*
Basic PDS		*11427*	*15726*	*24167*

表 4-1 向量量化器編碼系統面積複雜度

面積複雜度的量測是在 Altera 的 Cyclone EP1C20F4OOC7 FPGA 發展板上合成硬體電路，該發展板提供的邏輯單元數（logic elements，LEs）最大值為 20060 LEs。至於選擇在 Cyclone 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）可以快速的將硬體設計實現驗證，具備快速上市與系統再修改等特性，非常適合做雛型設計。

硬體實現時，碼簿內儲存預先設計好的碼字，碼字是由 LBG 演算法[2]經由兩張512×512的影像“Lena” , “Zelda”訓練得來，碼字的維度是8×8（i.e., $n=3$）。碼簿中的碼字只儲存$Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$，其中$m=2$，也就是說$Y_{Lm}^{j}$的維度是$4\times4$。位元平面縮減參數$l=6$，VSDC 運算單元一次計算四個係數（i.e., $\delta=4$）。表中同時實現了基本部分距離搜尋的架構來做比較，這個基本的部分距離搜尋架構是在小波領域上進行全部小波係數的部分距離搜尋，並未採用子空間搜尋、位元平面縮減以及多係數累積等技巧。

雖然說實現子空間部分距離搜尋架構增加模組數量時，需要較多的 VSDC 運算單元以及存取 CodeBook ROM 會有較高的繞線額外負擔，但從表 4-1 中可以觀察到，相同碼字數量N的情況下，多模組的子空間部分距離搜尋架構所花費的面積複雜度仍小於單一模組的基本部分距離搜尋架構。這是因為 DWT 運算單元做完小波轉換後儲存的碼字位元平面數量大增後，若沒有採用子空間搜尋和位元平面縮減等技巧，那麼硬體實現時儲存空間的額外負擔會是非常巨大的。特別是當$N=256$時，Nios 處理器內嵌小波領域上全係數部分距離搜尋硬

體架構的面積複雜度甚至超過了 Cycloe FPGA 邏輯單元數量的上限。相較之下，$N=256$、$M=3$ 的 Nios 處理器內嵌子空間部分距離搜尋硬體架構的面積複雜度還比 $N=128$、$M=1$ 的 Nios 處理器內嵌小波領域上全係數部分距離搜尋硬體架構來的低。可見本論文所提出的架構對於向量量化器編碼端部分距離搜尋演算法則的 VLSI 硬體實現有很大的幫助。

更進一步地來看本論文提出架構的效能，表 4-2 比較了部分距離搜尋在 Pentium4 處理器上以軟體實現和本論文提出的架構在 Nios 處理器上硬體實現所需的執行時間數據。

	Pentium4 1.4 GHz		Pentium4 1.8 GHz	
	PDS+Wavelet	**Subspace PDS**	**PDS+Wavelet**	**Subspace PDS**
uS	*59.1751*	*38.3671*	*40.3565*	*29.8685*

	Nios 50 MHz			
	M = 3	**M = 2**	**M = 1**	**Software**
uS	*11.998*	*15.726*	*29.3918*	*3189*

表 4-2 軟硬體向量量化器效能比較

在這邊執行時間定義為對每個輸入向量找到最接近距離碼字所需花費的平均 CPU 時間（uS）。在表 4-2 中，Nios 處理器相關呈現的數據包含本論文提出的架構在一個模組到三個模組的執行時間量測外，還測試了相同的法則以純軟體的方式在 Nios 50MHz 處理器上執行所需花費的平均 CPU 時間；Pentium4 處理器相關呈現的數據包含了兩種純軟體搜尋的方式，一種是在小波領域上對所有的小波係數做部分距離搜尋，另一種則是在小波領域上做子空間的部分距離搜尋。在這個實驗中，向量量化器的碼簿有 256 個8×8的碼字（i.e., $N=8$、$n=3$），訓練的方式及影像與表 4-1 相同。子空間部分距離搜尋的各個參數分別為$m=2$、$l=6$以及$\delta=4$。

表 4-2 中顯示了 Nios 處理器內嵌單一模組硬體電路執行搜尋的平均時間和 Nios 處理器以純軟體的方式執行相同演算法則的搜尋時間分別為 29uS和 3189uS，也就是說有單一模組部分距離搜尋硬體電路輔助的架構比純軟體的方式快了 110 倍左右。

另外 Nios 處理器內嵌部分距離搜尋硬體電路可以透過增加模組個數的方式來達到進一步降低執行時間的目的，特別是模組數從一增加到二的時候降低的效果最為顯著，平均的執行時間從 29uS降到 15uS，降了將近 50%的時間。另外，在表 4-2 中我們也可以看到，

在 Nios 處理器內嵌部分距離搜尋硬體電路上執行效果也比在 Pentium4 處理器上以純軟體的方式實現相同演算法則來的好。

另外，一般基本的全搜尋演算法則在 Pentium4 處理器上以純晚體的方式實現平均執行時間要 350*uS* 左右，我們提出的架構以兩個模組的來看僅需 15.726*uS*，約快了 22 倍左右。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。