第三章　PDS 架構與硬體實現

本章將將詳細說明本論文提出的硬體架構。

3.1 PDS 架構的硬體實現

本論文提出的 PDS 架構具有以下三樣特性：子空間搜尋、位元平面縮減以及多係數部分距離累積，適合用來做硬體實現，以下三個小節將分別說明這些特性及其對硬體實現所附加的優點。

3.1.1 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，所以在小波領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向量量化器編碼端的部分距離搜尋處理程序，可大量的降低編碼端的運算複雜度。同時子空間搜尋的向量量化器碼簿僅需儲存 Y_{Lm}^{j}，也就是原始碼簿 y^{j} 經由離散小波轉換後的部分低頻係數，其中 $j=1,2,\cdots,N$。這一點使得碼簿的儲存空間明顯地大為降低，這亦是搭配子空間搜尋特性的向量量化器在 VLSI 硬體實現的優點。

雖然碼字 Y_{Lm}^{j}，$j=1,2,\cdots N$，可以事先由原始碼簿經過離散小波轉換後取部份係數再儲存於硬體的碼簿中，但是輸入向量 x 的離散小波係數 X_{Lm} 卻需於向量量化器編碼端硬體中運算求得，所以在子空間搜

尋向量量化器硬體中仍需具備 DWT 運算單元。我們使用 Haar 小波轉換來做向量量化器的 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 8x8 以及 4x4，也就是說在這個架構中 $n=3$，$m=2$。其中 X_{L3} 為輸入向量 x 的 64 個係數，排列順序如圖 3-2 所示；而 X_{Lm} 則是由 X_{L0} 和 X_{Vi}、X_{Hi}、X_{Di} 所組成，$i=0,1$，如圖 3-3 所示。

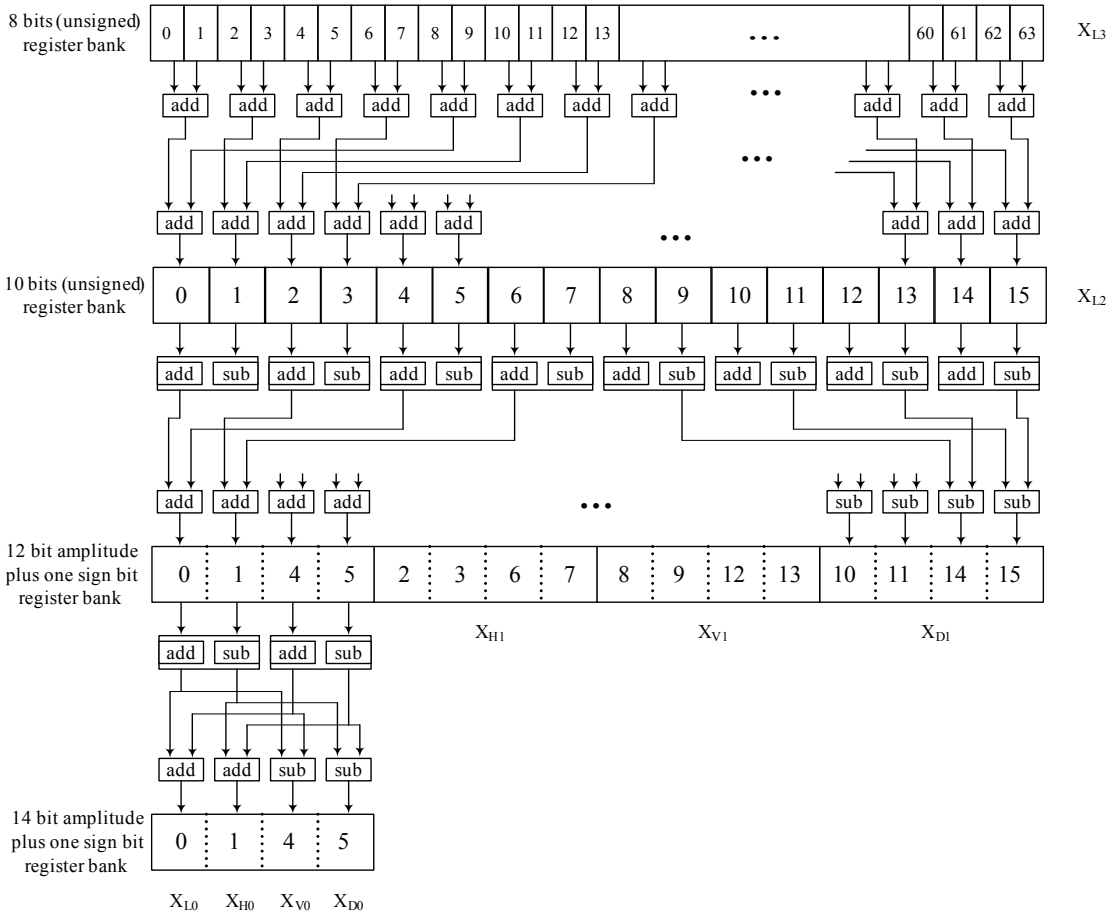

圖 3-1 DWT 運算單元硬體架構圖

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 X_{V2}、X_{H2}、X_{D2} 等高頻部份，保留低頻的 X_{L2}。然後對 X_{L2} 繼續做兩階的 Haar DWT 後求得 X_{L0} 和 X_{Vi}、X_{Hi}、X_{Di}，其中 $i=0,1$。

X_{L3}

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

圖 3-2 輸入向量排列順序

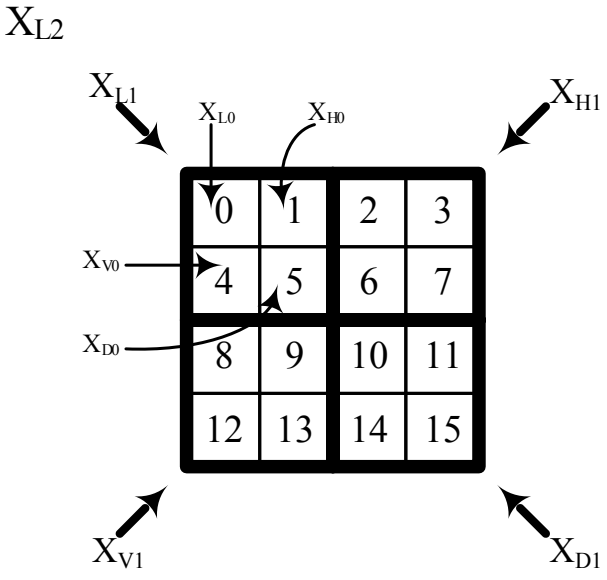

圖 3-3 三階小波係數順序對應圖

3.1.2 位元平面縮減（Bitplane reduction）

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 ROM 裡用來做部分距離搜尋運算。

我們可以發現移除係數的部分 LSB（least significant bit）位元平面對於 PDS 的效能只會造成極小的影響，卻可以大幅度的降低硬體實現時碼簿儲存空間的面積複雜度，所以我們並不需要精確的儲存小波係數 X_{Lm} 的所有位元平面。從表 3-1 列出的數據可以看出搭配子空間部分距離搜尋技巧的向量量化器，在不同程度位元平面縮減情況下進行編解碼後還原影像的峰值訊雜比（Peak Signal to Noise Ratio，PSNR）。同時表 3-1 也包含了在這些條件下硬體實現時碼簿儲存空間（Codebook ROM）所需的面積複雜度。在這裡 PSNR 值是由 $10\log\frac{255^2}{D}$ 所求得，其中 D 代表的是輸入向量 x 與重建向量 $\hat{x}$ 的均方誤差。而儲存向量量化器碼字係數的 ROM 在 FPGA 硬體實現時所需使用的邏輯元件數量（logic elements，LEs）在這邊用來當做衡量面積複雜度的尺度。表 3-1 中，用來合成碼簿硬體的 FPGA 是 Altera

公司所生產的 Cyclone FPGA 發展板，而位元平面縮減程度是以 l 來代表移除碼字係數多少 LSB 位元平面。

Subspace		**No**	**Yes**			
l		**0**	**0**	**2**	**4**	**6**
PSNR	Girl512	*30.2219*	*30.0063*	*30.0036*	*29.9647*	*29.8685*
	Baby512	*26.9533*	*26.6825*	*26.6782*	*26.6729*	*26.5793*
	House512	*25.9510*	*25.6674*	*25.6667*	*25.6648*	*25.6115*
CodeBook LEs		*11645*	*5156*	*4229*	*2973*	*1891*

表 3-1 位元平面縮減對影像品質與碼簿大小的影響

給定碼字數目 $N=256$，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., $m=2$）。其中碼簿中的碼字是由 LBG 演算法[2]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 “Girl512”、“Baby512” 和 “House512” 三張測試影像做編解碼後還原求得。從表 3-1 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，但是碼簿的面積複雜度卻可以大幅度的降低（從 11645 LEs 到 5156 LEs）。另外採用位元平面縮減的技巧可以更進一步的降低儲存空間，當碼字係數移除 6 個 LSB 位元平面時，只要付出 PSNR 值降低約 0.5 dB 的代價便可以降低面積複雜度約 65%（從 5156 LEs 到 1891 LEs）。

3.1.3 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（2.8）式所示，因此在硬體實現的時候僅需要一個乘法器。為了要提高向量量化器編碼端的吞吐量，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，也就是在我們的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：

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

，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 2-4 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。

我們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 3-4 所示。

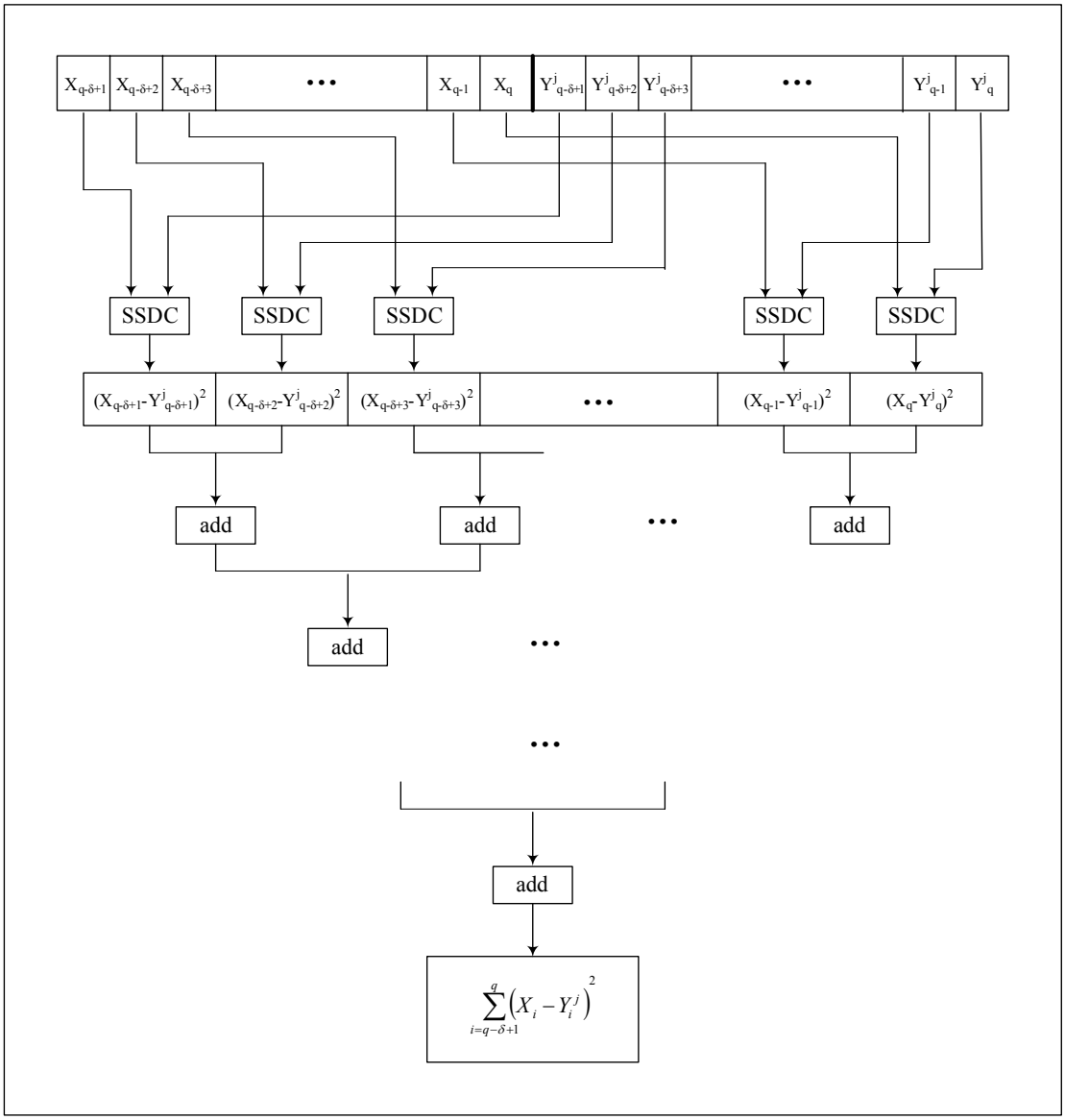

圖 3-4 向量平方距離計算單元 VLSI 架構示意圖

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 3-5 所示。

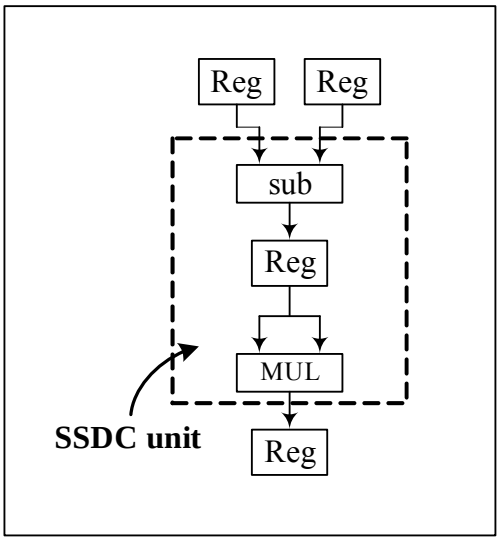

圖 3-5 純量平方距離計算單元 VLSI 架構示意圖

經過實際測試後，我們可以很清楚的觀察到 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，隨著位元平面縮減程度l的增加可以不斷的降低面積複雜度。表 3-2 顯示了不同的δ與l對 VSDC 單元面積複雜度的影響。

LEs		l			
		0	**2**	**4**	**6**
δ	**1**	*396*	*286*	*229*	*159*
	2	*833*	*611*	*493*	*349*
	4	*1712*	*1275*	*1032*	*740*
	8	*3463*	*2579*	*2095*	*1507*
	16	*6967*	*5195*	*4223*	*3043*

表 3-2 不同參數對 VSDC 單元面積複雜度的關係表

在這裡我們量測 VSDC 單元硬體電路合成時所需花費的面積複雜度一樣是採用 Altera Cyclone FPGA 發展板。從表 3-2 中我們可以觀察到，雖然δ值較高的 VSDC 單元可以有較高的編碼吞吐量，但是它所需的面積複雜度卻是非常的大。尤其是當$\delta=16$、$l=0$時，硬體實現所需的面積複雜度為 6967 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，其中$m=2$，但是它卻需花費 Cyclone FPGA 高達 34%的 LEs，也因此δ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., $\delta=4$）需要的面積複雜度也較低。在我們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 740 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 10%左右的比重。

3.1.4 subspace PDS 硬體實現的概述

我們提出用來硬體實現的部分距離搜尋架構的整個概要流程如下所列：

假定條件：給定m、l、δ的值

碼字 $Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$ 已預先訓練好，並以位元平面縮減程度 l 處理過後儲存於碼簿記憶體（Codebook ROM）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 $j=1$

指定初始目前最近距離碼字 y^{p} 為 $y^{p}=y^{1}$

則初始目前最小量化誤差 $D_{\min}$ 為 $D_{\min}=D\left(X_{Lm},Y_{Lm}^{1}\right)$

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 $D_{\min}$

如果 $D^{q}\left(X_{Lm},Y_{Lm}^{j}\right)>D_{\min}$，則到 Step3 進行下一個碼字的搜尋，否則就繼續往下一個步驟執行

Step5：如果 $q=2^{m}\times 2^{m}$

則設定 $y^{p}=y^{j}$ ， $D_{\min}=D\left(X_{Lm},Y_{Lm}^{j}\right)$

然後到 Step3 進行下一個碼字的搜尋

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，系統保留的目前最近距離碼字就是相對於輸入向量 x 的最近距離碼字（final closest codeword to x）。

從上面的流程我們可以觀察到碼字的搜尋是在小波領域上面進行（X_{Lm} 是 x 的小波低頻係數，$Y_{Lm}^1,\cdots,Y_{Lm}^N$ 是所有碼字的小波低頻係數），所以這是一個子空間部分距離搜尋演算法的架構。這個架構使用了位元平面縮減的技巧，在部分距離搜尋處理程序之前會先移除 X_{Lm} 以及 $Y_{Lm}^1,\cdots,Y_{Lm}^N$ 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，本文針對向量量化器編碼程序的硬體實現提出了三個極佳的技巧，包括子空間搜尋、位元平面縮減以及多係數累積，這三個技巧能夠大幅度的降低硬體實現所需的面積複雜度，提高吞吐量。

3.2 單一模組 VLSI 硬體架構

本論文提出了適合硬體實現的子空間部分距離搜尋演算法則，在圖 3-6 中顯示了這個法則的基本 VLSI 硬體架構，包含了一個 CodeBook ROM、一個 DWT 運算單元、一個計算向量平方距離的 VSDC 運算單元、累積器、比較器、PDS 模組控制單元，以及一些儲存中間值、暫時結果、暫存資料的暫存器。CodeBook ROM 儲存了碼字 $Y_{Lm}^{1},\cdots,Y_{Lm}^{N}$ 以用來進行快速搜尋比對，DWT 運算單元對輸入向量 x 進行 Haar 小波轉換獲得低階的小波係數 X_{Lm}，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 $D^{q}\left(X_{Lm},Y_{Lm}^{j}\right)$，然後比較器用來比較部分距離 $D^{q}\left(X_{Lm},Y_{Lm}^{j}\right)$ 跟目前最小量化誤差 $D_{\min}$。比較器的結果會傳送到 PDS 模組控制單元，PDS 模組控制單元根據這個結果來決定是否要重置累積器、更新目前最近距離碼字 y^{p} 以及更新目前最小量化誤差 $D_{\min}$。各單元間的運作由 PDS 模組控制單元和整體控制單元協調控制。

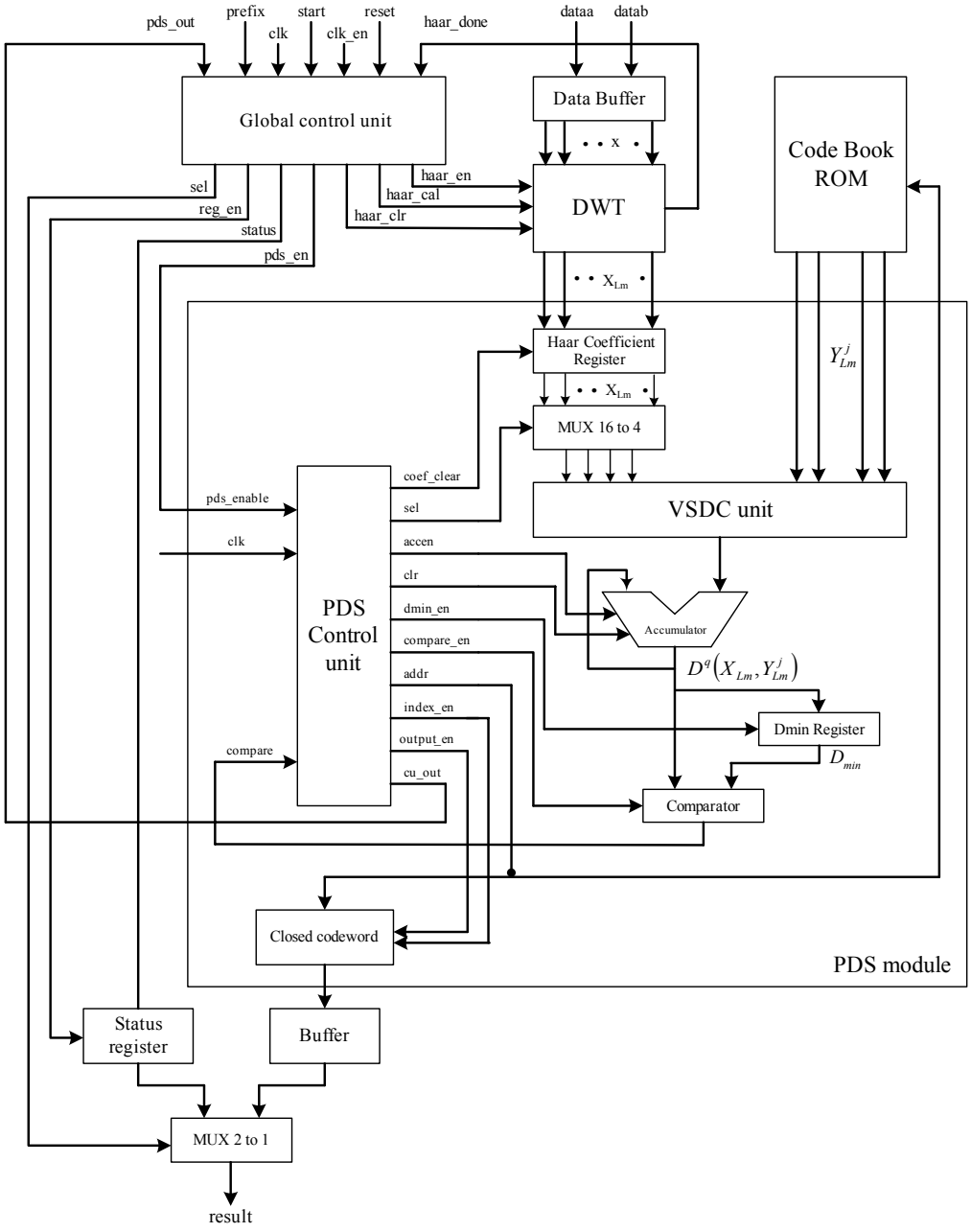

圖 3-6 部分距離搜尋單模組 VLSI 架構

在表 3-2 中，我們研究過不同δ值在 VSDC 運算單元硬體實作時，對面積複雜度所造成的影響。我們知道δ值越小越能夠降低面積

複雜度，但在此之前尚未討論δ值對系統效能的影響。我們利用表 3-3 所提出的實驗數據來觀察不同的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 3.1.4 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（3.2）式所示：

$$\mathcal{L} = \frac{1}{tN}\sum_{j=1}^{t} \mathcal{T}(x^j) \qquad (3.2)$$

其中t代表的是輸入的向量個數，N代表的是碼簿中的碼字數目。對於每一個輸入向量而言，比較每一個碼字所花費的平均時脈週期數記做$\mathcal{L}$。實驗環境的參數條件與碼簿訓練方式與表 3-1 相同，$N=256$、$n=3$、$m=2$，輸入向量個數$t=12288$來自於 “Girl512”、“Baby512” 以及 “House” 這三張512×512的影像。

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

表 3-3 Latency：比較單一碼字平均時脈週期數

在表 3-3 中我們可以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，判斷一個碼字是否為非最近距離碼字（undesired codeword）在參數δ值些微大於 1 時所花費的 Latency 比 1 大但接近於 1。另外我們從表 3-1 和表 3-2 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。因此我們提出具有低面積複雜度和高吞吐量產出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碼字是否為非最近距離碼字（undesired codeword）所需耗費的 Latency 接近於 1。

3.3 多模組 VLSI 硬體架構

由於不同的輸入向量在進行向量量化器編碼的處理程序時可以各自獨立的運作，因此利用多個部分距離搜尋模組同時對多個輸入向量平行做碼字搜尋可以大量的增加向量量化器編碼端的吞吐量。如圖 3-6 所示的基本部分距離搜尋單模組 VLSI 架構可以針對這個需求做延伸，提出一個系統具有多個基本單模組可平行獨立運作的架構，如圖 3-7 所示。這個多模組的架構包含了 M 個模組，每一個模組負責針對一個輸入向量做部分距離搜尋的運算，也就是說在這個架構下同時間可以平行處理最多 M 個輸入向量。另外，在這個架構裡我們每一個時間點只從 host processor 抓取一筆輸入向量，也因此隨著模組數 M 的增加，我們的接腳數目仍然可以保持在低水平。在這個輸入向量單一擷取的結構下不需同時進行多筆輸入向量的 DWT 轉換運算，因此所有的模組可以共用一個 DWT 運算單元。從圖 3-6 中我們也可以發現所有的模組是共同存取一個碼簿的內容，這些硬體單元共享的機制在模組數目變多時可以很明顯的降低晶片的面積複雜度。

每一個模組都有自己專用的 VSDC 運算單元、累積器、比較器和 PDS 控制單元，所以每一個模組都可以獨立的進行部分距離搜尋的運算。輸入到每一個模組的資料是做完 DWT 運算後的低階小波係

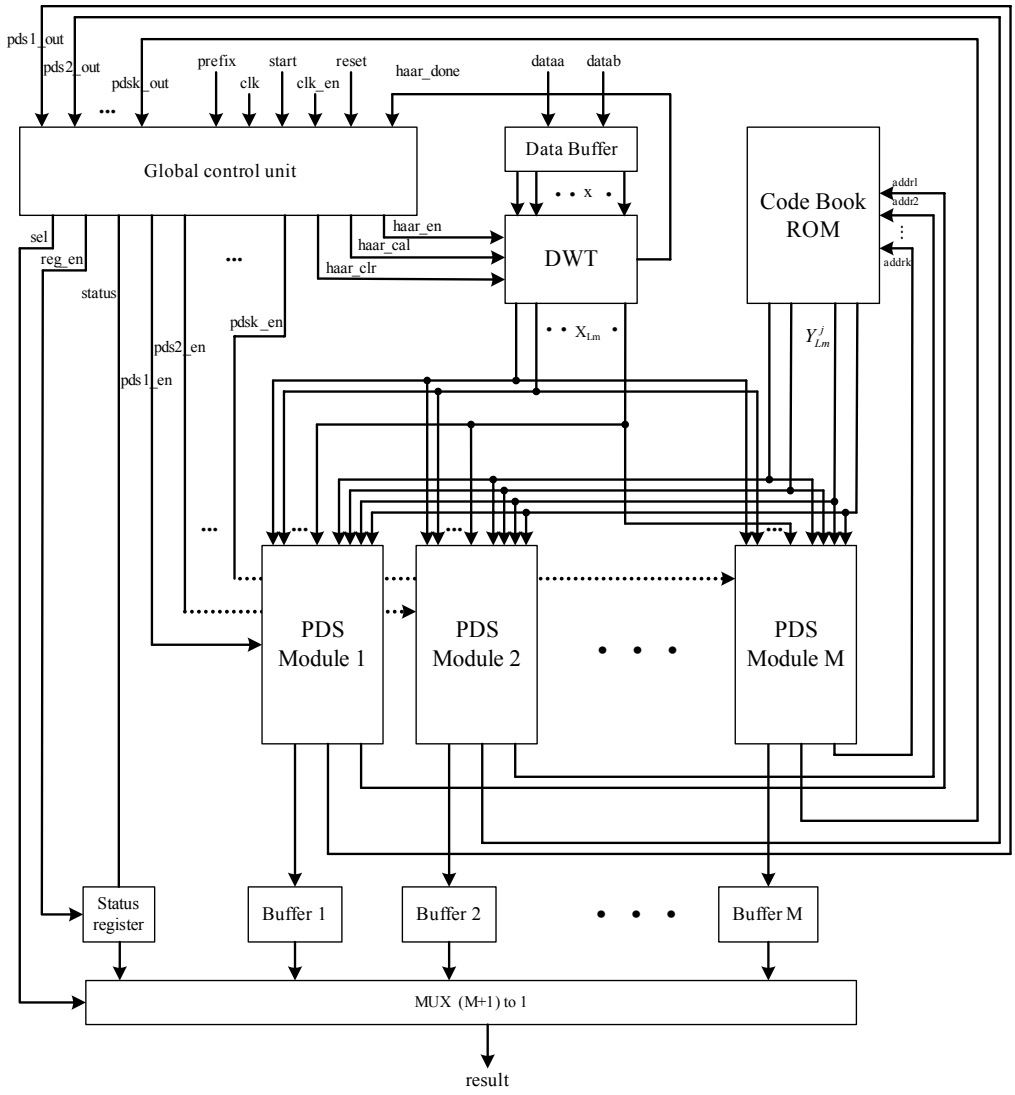

圖 3-7 部分距離搜尋多模組 VLSI 架構

數 X_{Lm}，所以在 3.1.4 節所討論的部分距離搜尋演算法則各個步驟中，每個模組只要負責 Step2 ~ Step6 的運算步驟。表 3-3 的討論告訴我們本論文提出的部分距離搜尋演算法所需花費的平均 Latency 很低，雖然輸入向量 x 都在小波領域經過相同程度的能量壓縮，但實際

上的 Latency 卻不盡然相同，所以在多模組的 VLSI 架構裡，對於多個依序的輸入向量，計算完成產生最接近碼字的時間不一定會依照先進先出（first-in first-out，FIFO）的順序。於是在多模組的架構中，我們必須用一個狀態暫存器配合全域控制單元來協調各個模組的運作，這個狀態暫存器裡面記錄了以下幾種狀態：

1.Input_Data_Available：輸入向量是否已完整送進 data_buffer。

2.Write_Back_Result：模組算完結果是否已傳出。

3.Ready_for_Input：是否有模組有空且 data_buffer 是可以使用的。

（Ready_for_Input = Module_Available & Input_Buffer_Available）

（Module_Available = $\bigcup_{i=1}^{M}$Module_Available_of_A_i）

4.Result_Available：是否有運算結果等待傳出。

（Result_Available = $\bigcup_{i=1}^{M}$Result_Available_of_A_i）

5.Input_Buffer_Available：是否可使用 data_buffer 存放輸入向量。

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

模組的運作，完整的狀態轉換流程如圖 3-8 和圖 3-8-1 所示：

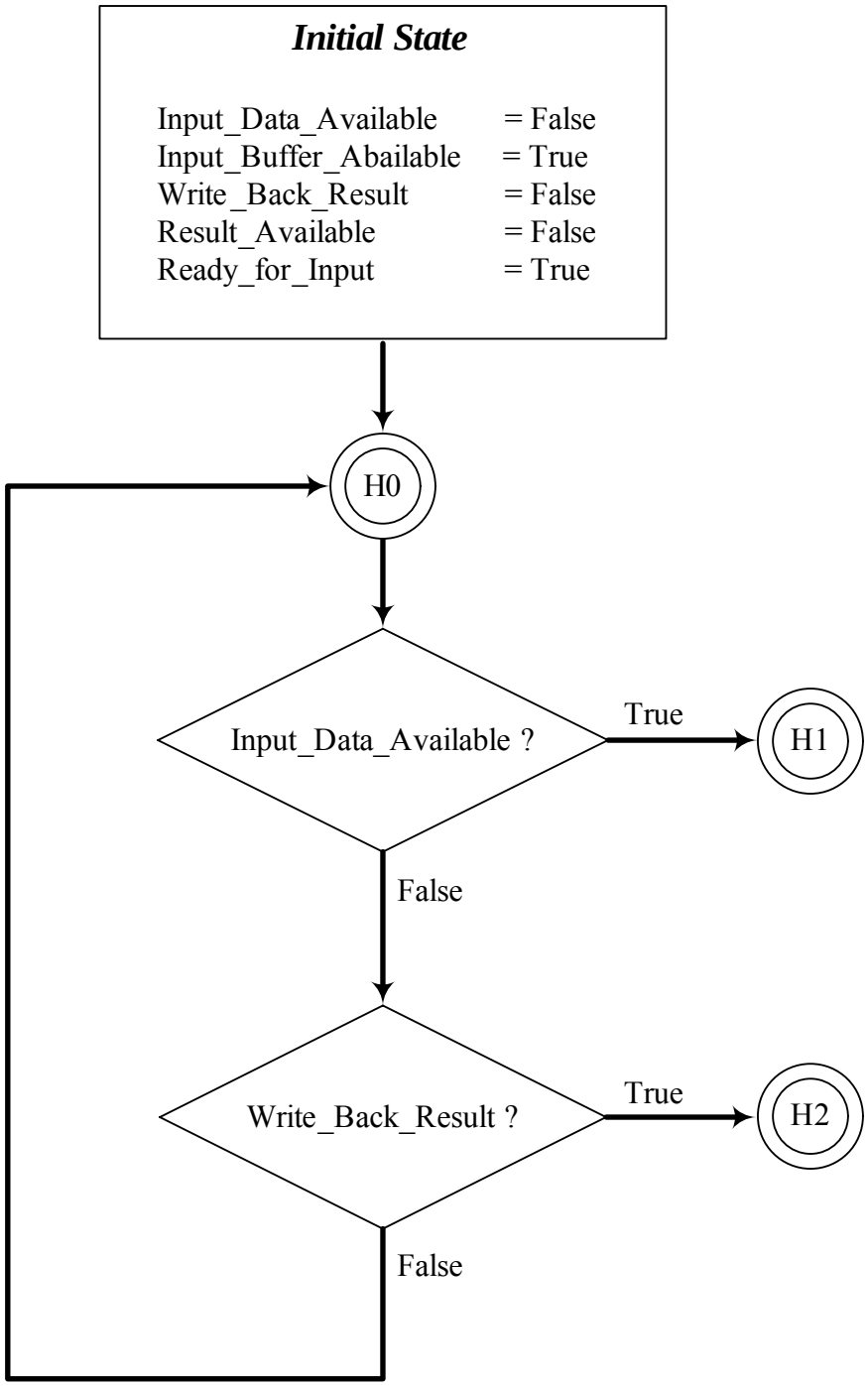

圖 3-8 全域控制單元的狀態流程

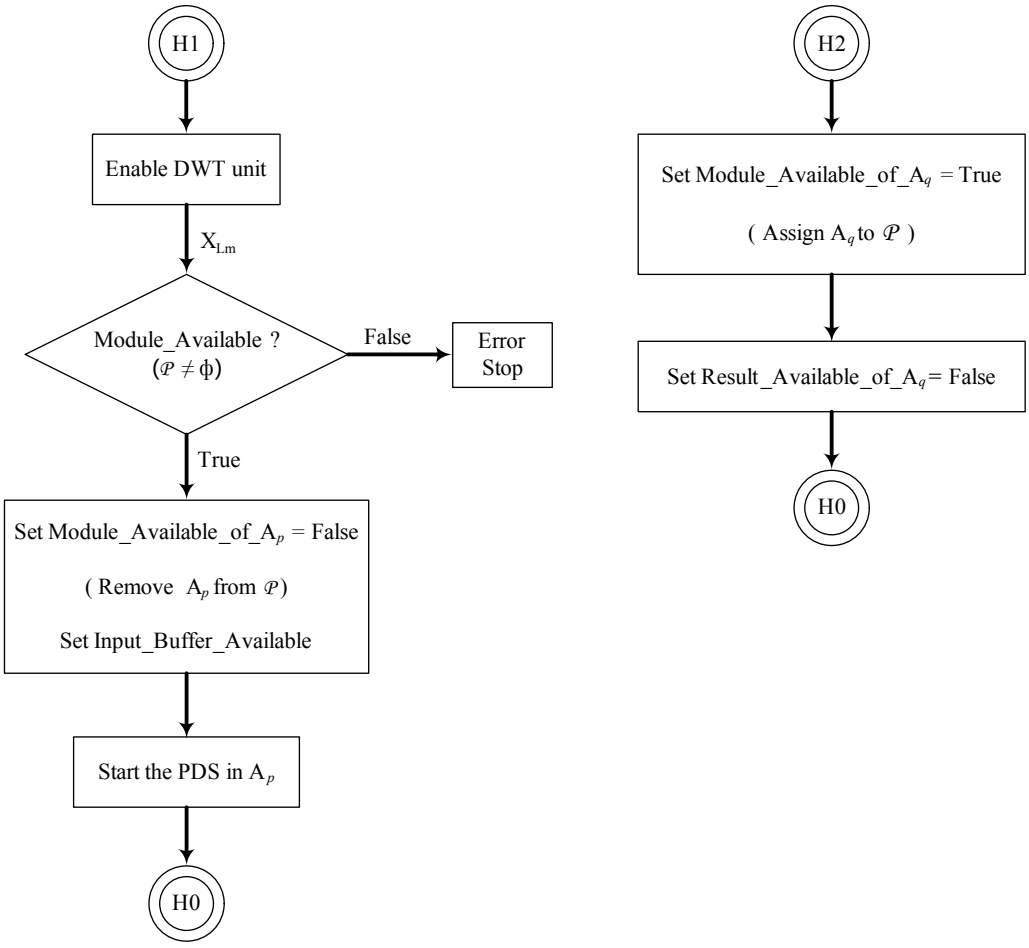

圖 3-8-1 全域控制單元的狀態流程之二

系統的初始狀態定義在圖 3-8 的一開始 Initial State，系統啟動之初 Input_Buffer 是空的，允許寫入資料（Input_Buffer_Available = True），未有資料放入 Buffer 等待運算（Input_data_Available = False），當然也沒有任何一個模組會有結果完成（Result_Available = False），軟體端也不會有抓取結果的回傳訊息（Write_Back_Result = False），有模組有空且允許寫入資料（Ready_for_Input = True）。

當軟體端已經輸入向量完全傳送至 input_buffer 時，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，選擇任一有空的模組開始進行部分距離搜尋的運算。如果有模組已完成編碼運算且軟體端已經收結果，則 Write_Back_Result 會被軟體端設為 True，此時若 Input_Data_Available = False，硬體端便會進入 H2 的流程將已取走編碼結果的模組設為有空，可準備繼續進行下一筆輸入向量的編碼程序。

同時有多個模組有空或是多個模組編碼結果等待接收時，全域控制單元選擇的機制如下：令 $\mathcal{P}$ 表示目前有空的模組集合，$A_i, i = 1, \cdots, M$ 代表架構中的第 i 個模組，當要選擇模組進行編碼運算時，我們會選擇模組 A_p，$p = \min\{\ i : A_i \in \mathcal{P}\ \}$。令 Q 表示忙碌狀態且已完成編碼運算的模組集合，當要選擇編碼結果輸出到軟體端時，我們會選擇模組 A_q 的最接近碼字編碼結果輸出，q=min $\{\ i : A_i \in Q\}$，當編碼結果送出後，A_q 變為有空模組設定回集合 $\mathcal{P}$ 等候下次進行編碼運算。

而軟體端的流程我們留待下一小節再做介紹。

3.4 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。

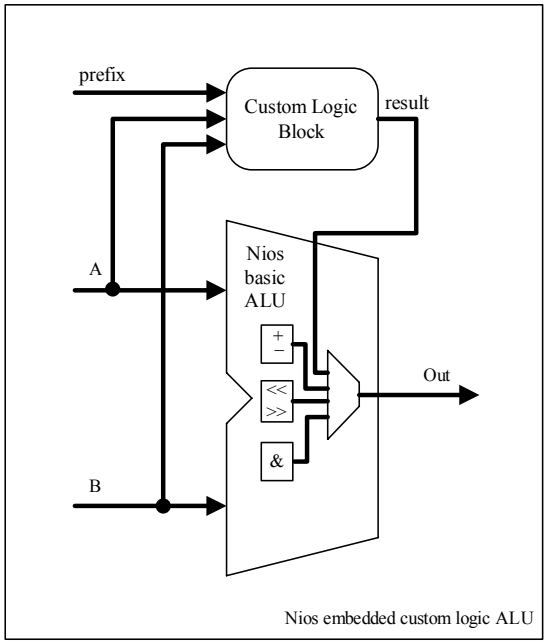

圖 3-9 使用者專用邏輯區塊在 Nios CPU 中的位置

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 3-9 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。在圖中我們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。這個架構以我們的部分距離搜尋專用硬體電路為例，其介面如圖 3-10 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”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，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“prefix”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輸入向量 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 到我們的專用硬體電路需要花費 8 個時脈週期，因此在使用

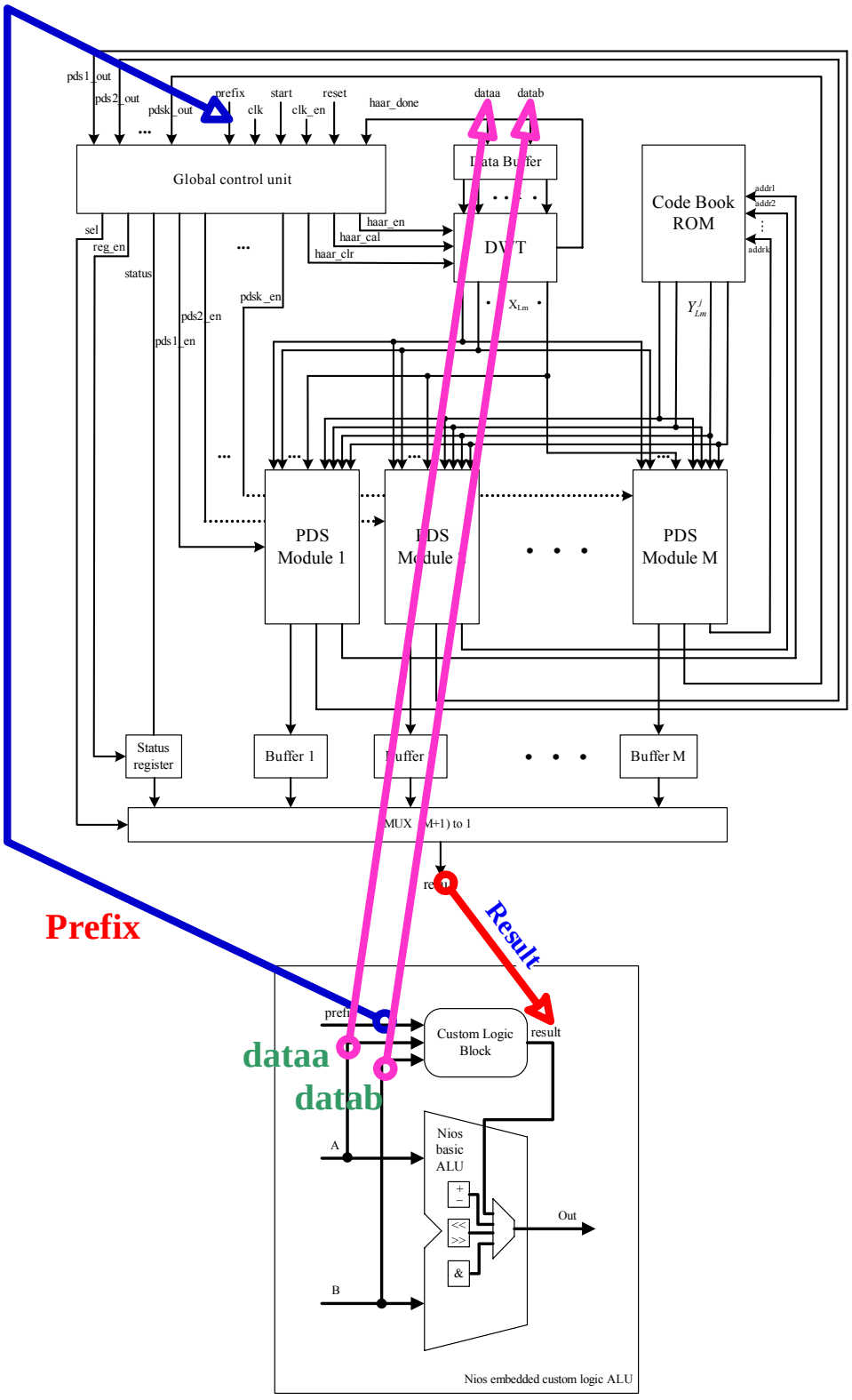

圖 3-10　PDS 硬體架構介面

DWT 單元進行小波轉換之前，我們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。

軟體端可以透過呼叫軟體巨集的方式，使用客製指令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，可以傳送資料區塊到部分距離搜尋專用硬體中，也可以接收專用硬體電路編碼後的結果。軟體端的狀態流程如圖 3-11 以及圖 3-11-1 所示，有模組有空且 data_buffer 可供暫存資料，此時的狀態暫存器顯示 Ready_for_Input，各種狀態成立的條件已在 3.3 節中討論定義，軟體可進入 S1 流程輸入資料到專用硬體電路中。若所有模組皆沒空但有編碼結果等候軟體端接收，此時狀態暫存器顯示 Result Available，軟體可進入 S2 流程從專用硬體電路擷取編碼結果。

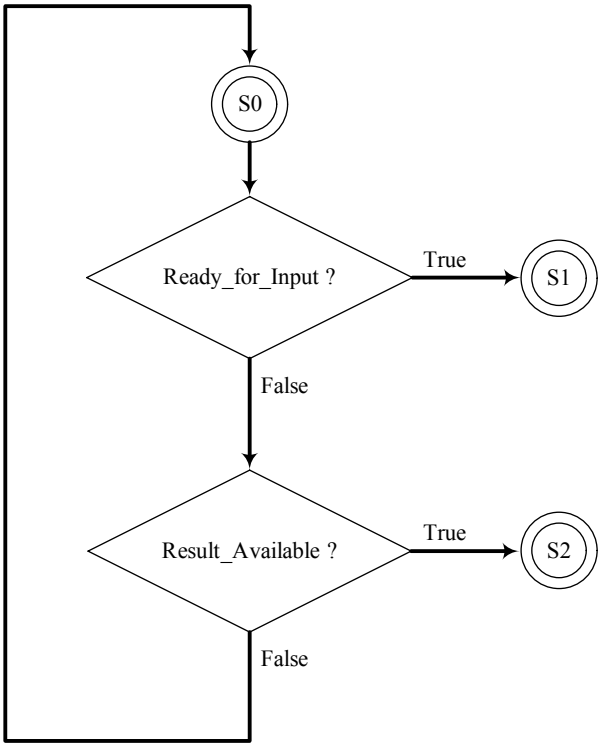

圖 3-11 軟體端的狀態流程

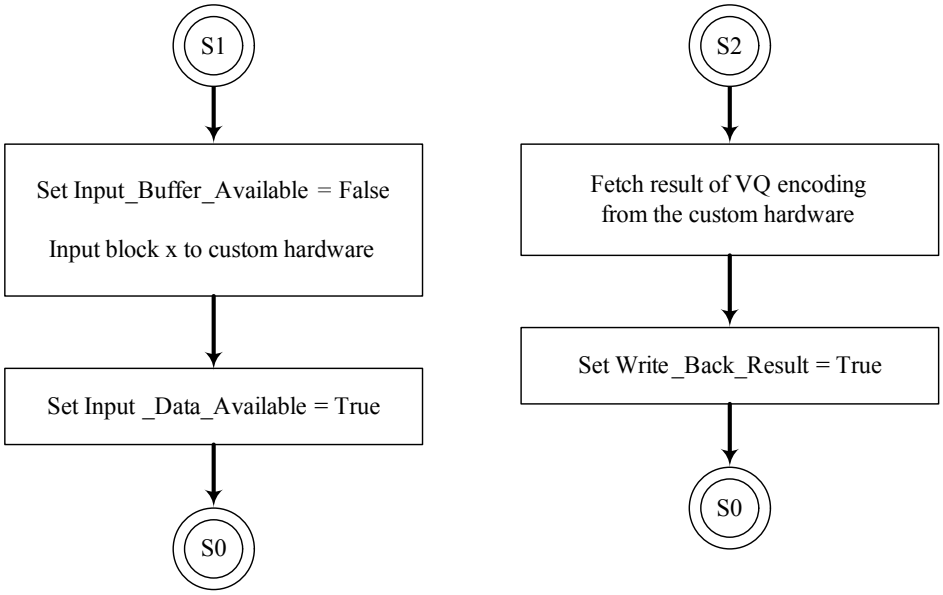

圖 3-11-1 軟體端的狀態流程之二