高效能線性拓展之影像解析度縮放硬體實現

The Efficient VLSI Design of Extended-Linear Scaling for Digital Image

吳曾傳[1]，陳嘉宏[1]，陳思穎[1]
[1]國立雲林科技大學電子工程所
Email:{g9310806,g9613704,g9613712}@yuntech.edu.tw

林正基[2]，蔡文凱[2]，許明華[2]
[2]雲林科技大學工程科技所
Email:{g9413721,sheumh}@yuntech.edu.tw

摘要

由於多媒體應用的普及性與顯示器製造技術不斷的進步，數位顯示器的解析度規格也愈趨多樣性，因此如何將數位影像於不同解析度規格之間做轉換並且在維持其高影像品質的同時兼顧低運算成本便是一個重要的課題。本篇論文將針對影像解析度縮放技術設計出線性拓展(Extended-Linear Scaling)的演算法，此演算法具有高效能、低硬體成本與高使用彈性的電路架構。透過撰寫Verilog HDL與VLSI cellbase 晶片製作流程，將此硬體架構實現，此晶片使用的製程為TSMC 0.13μm，其使用面積為 452×452μm^2，其Gate Count為26200 gates，而工作速度可達 267Mhz，如此將可達到即時運算之能力。

關鍵詞：影像縮放、影像插補、Extended-Linear、VLSI

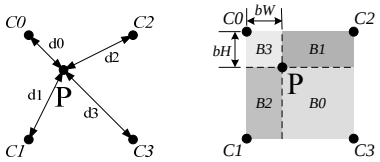
一、簡介

顯示器製造技術愈來越發達，螢幕的尺寸大小也與日俱增，使得畫面上呈現之影像解析度亦逐漸提升，為了讓影像能夠在各種不同尺寸大小的螢幕上完美的顯示圖片而不產生失真，影像縮放(image scaling)技術就顯得格外重要。影像縮放技術主要就是降低原始高解析度影像之像素個數，或者提高低解析度影像之像素個數。然而影像縮放所需要的運算量相當龐大，若是單純透過處理器以軟體的方式進行處理，將造成系統繁重的負擔。因此可以利用硬體實現的方式完成影像縮放演算法，加速整體影像系統之影像品質與效能。

目前普遍使用中的影像插補技術有三個種類：(1)最鄰近內插法(Nearest - neighborhood interpolation)、(2)雙線性內插法(Bilinear interporlation)與(3)雙立方內插法(Bicubic interpolation)[1-5]。以上三種插補技術中最簡單的就是最鄰近內插法，圖一(a)中 *C0*、*C1*、*C2* 與 *C3* 為原始影像的像素點，P 則是即將要插補的像素點。判斷 P 與周圍各點的距離後，選擇距離最短的像素值並指定給 P。此演算法的運算複雜度低。利用這種方式完成的影像放大會造成方塊效應(Block effect)，使得影像的品質大幅降低。

由於最鄰近內插法會造成嚴重的方塊

效應，因此為了達到品質較佳的插補影像，於是發展出雙線性內插法。使用雙線性內插法首先需計算出 P 與鄰近四個原始影像像素點的間距，而這些距離的資訊代表 P 與它們之間的關聯性，間距越短代表關聯性越高，反之關聯性則越低。因此可以藉由 P 與四個原始影像點劃分出四塊對應的區域，如圖一(b)所示。利用劃分出來的四塊區域面積當成權重值，透過式(1)得到 P 點的像素值。



(a) (b)

圖(一) (a)最鄰近內插法，(b)雙線性內插法

$$P = (B0 \cdot C0) + (B1 \cdot C1) + (B2 \cdot C2) + (B3 \cdot C3) \quad (1)$$

雙立方內插法也是目前被廣為使用的一種插補技術。完成雙立方內插法必需藉由較複雜的運算條件，如式(2)所示，因此其補插效果比上述兩種方式佳。由於複雜運算條件的影響，導致處理時的複雜度也跟著提高。

$$h_c(x) = \begin{cases} 1 - 2|x|^2 + |x|^3, & for\ 0 \le |x| < 1 \\ 4 - 8|x| + 5|x|^2 - |x|^3, & for\ 1 \le |x| < 2 \\ 0, & otherwise \end{cases} \quad (2)$$

二、線性拓展插補法

雙立方插補法之影像插補品質相當優良，但是其運算複雜度過於龐大，因此我們提出線性拓展插補演算法，此演算法具有接近雙立方插補法之影像插補品質，且以低運算複雜度來降低硬體電路現實的電路成本，提升其使用效能。

(一) 線性拓展插補演算法

於雙線性插補法的線性方程式僅使用鄰近的兩個像素作為取樣點，而雙立方插補法的取樣點則是四點，所以雙立方插補法之影像插補品質相當優良，這與趨近無限函數Sinc函數有關，因此我們將線性插補方程式拓展成取樣四點。透過圖(2)Sinc函數的時域表現可以發現，當時間軸介於一至二之間時，Sinc函數將會是負值，因此對照至影像系統的空間軸時可以推論當與插補點距離為一至二之間時，其插補係數若為負值則對影像品質的提升會有正面的幫助。所以我們定義線性拓展插補法的插補係數方程式，如(3)式。(3)式中的變數d為四個像素各自距離插補點的間距值如圖(3)所示，而β則是銳化係數(Sharp Factor)。所以插補方程式如(4)式，將A至D此四點與P點的距離代入(3)式中所得的插補係數k_0至k_3再進行迴旋積分。

$$k(d) = \begin{cases} (\beta - 1) \times |d| + 1 & ,0 \le |d| < 1 \\ -\beta \times |d| + \beta & ,1 \le |d| < 2 \\ 0 & ,2 \le |d| \end{cases} \quad (3)$$

$$P = (A \times k_0) + (B \times k_1) + (C \times k_2) + (D \times k_3) \quad (4)$$

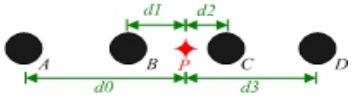
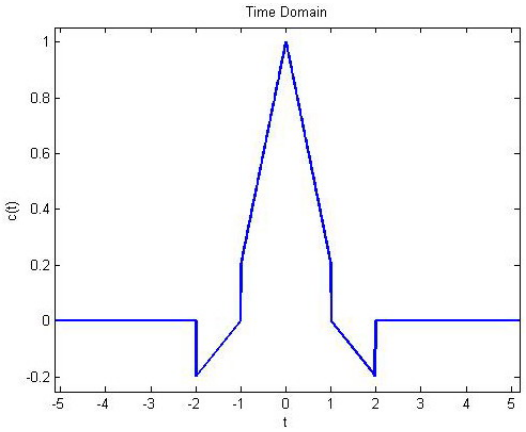


圖 3 四取樣點之線性插補方程式示意圖

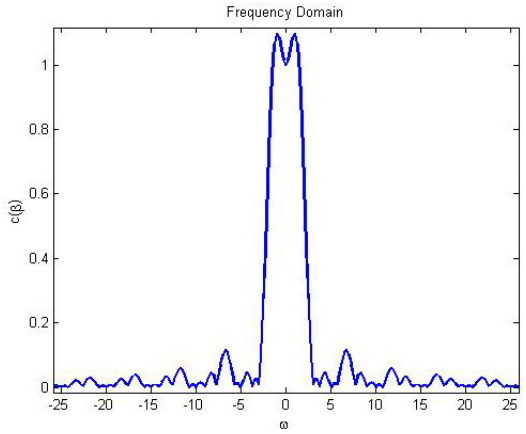
當(3)式中的β值介於 0 至 1 之間時，其時域之表現將可以得到趨近無限函數 Sinc 函數的結果。如圖(4)的各時域圖所示當距離介於 0 與 1 之間時插補係數將會是正數，而距離介於 1 至 2 之間時會得到負數的插補係數，使得插補係數於時域的表

現類似 Sinc 函數，藉以提升插補影像品質。而此線性插補方程式可維持插補係數的總和為 1，避免整體影像的像素值遭到同步提升或降低的現象發生。

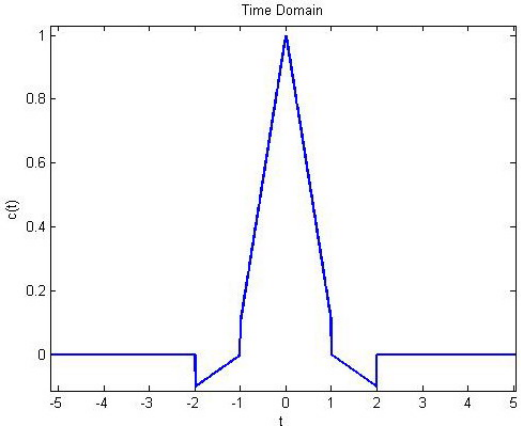
同時β值將決定插補影像的銳化程度，如圖(4)頻域圖所示，當β值等於 0.2 時其頻域的表現可明顯看出高頻響應偏高，降低低通濾波的效果，因此將會造成插補影像的對比升高。當β值等於 0.1 時，可發現其高頻響應已明顯變低，並且其低頻響應的部份也逐漸修正至接近理想的低通濾波器，低通的頻寬也較原四取樣點線性方程式較高，因此可得知β值與插補影像的銳利程度成正比。



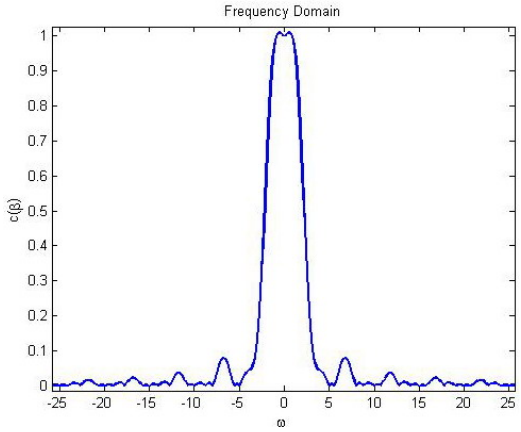
(a) $\beta = 0.2$ 時域圖



(b) $\beta = 0.2$ 頻域圖



(c) $\beta = 0.1$ 時域圖



(d) $\beta = 0.1$ 頻域圖

圖 4 銳化係數β=0.2 及 0.1 之時/頻域表現圖

(二) 最佳銳化係數β值之決定

為了求得線性拓展插補法的插補係數方程式影像品質為最佳之β值，我們將透過標準差(Standard Deviation)如(5)式，比較各β值與 Sinc 函數於頻譜中的相似程度，因為當某一β值其頻譜與 Sinc 函數之頻譜越為相近時，代表其影像品質將越顯優秀。圖(5)便是計算介於 0 至 0.5 之間各β值所呈現之頻譜與 Sinc 函數之頻譜所求得標準差之曲線圖，可發現當β介於 0.1 與 0.15 之間時其標準差都相對微小且接近，而頻譜標準差最小之β值為 0.1216。

$$\sum_{i=-n}^{n}\{\{[a_i-(a_i+b_i)/2]^2+[b_i-(a_i+b_i)/2]^2\}/2\}^{0.5} \quad (5)$$

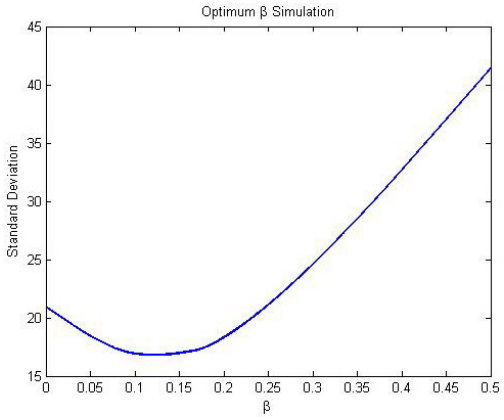


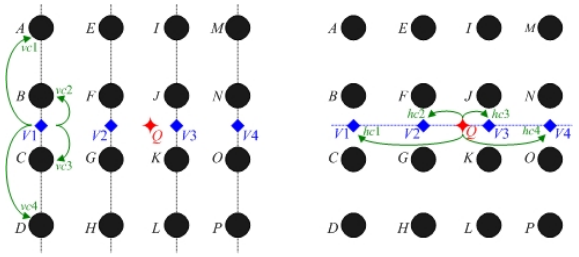
圖 5 $\beta=0\sim0.5$ 之標準差曲線圖

依線性拓展插補法的定義所求得影像品質最佳之β值為 0.1216，接著選擇β值為 0.125，其方程式如(6)式。選擇β值為 0.125 的原因，因為式中的距離值d乘上 0.125 這常數時，而對電路運算特性而言僅需要將距離值d執行右移三位之二的冪次方除法動作即可，再搭配運算複雜度低的加法器便可以完成係數產生的動作。同時 0.125 也是最接近最佳β值 0.1216 且擁有上述效果之數值。

$$C_{0.125}(d)=\begin{cases}-0.875\times|d|+1 & ,0\le|d|<1\\ 0.125\times|d|+0.125 & ,1\le|d|<2\\ 0 & ,2\le|d|\end{cases} \quad (6)$$

(三) 二維插補之一維分解架構

線性拓展插補法是使用二維的插補運算觀念，每個插補點必須一次完成 16 點的迴旋積分，如此其運算複雜度過於龐大。因此將一次的二維插補拆解成一維插補架構(One-Dimension Decomposition of Two-Dimension Interpolation)，使用此架構不但可以減少係數產生之運算量亦可以減少記憶體存取之次數。如圖(6)所示，Q點為目前的插補點，A至P點則是插補點鄰近之原始影像像素，透過式(3)四次的垂直插補可產生其中的$V1$、$V2$、$V3$、V4，而此四點是暫時設置的虛擬像素(Virtual Pixel)，虛擬像素並不屬於實際影像的一部分，而是為了進行水平插補而暫時產生，當插補Q點所需的四個虛擬像素產生後，水平插補會透過執行(4)式完成插補出Q點的所有運算。



(a) 垂直插補　　(b) 水平插補

圖(6) 二維插補之一維分解運算示意圖

$$\begin{aligned}V1&=(A\times vc1)+(B\times vc2)+(C\times vc3)+(D\times vc4)\\ V2&=(E\times vc1)+(F\times vc2)+(G\times vc3)+(H\times vc4)\\ V3&=(I\times vc1)+(J\times vc2)+(K\times vc3)+(L\times vc4)\\ V4&=(M\times vc1)+(N\times vc2)+(O\times vc3)+(P\times vc4)\end{aligned} \quad (7)$$

$$Q=(V1\times hc1)+(V2\times hc2)+(V3\times hc3)+(V4\times hc4) \quad (8)$$

由於對一張二維影像進行影像插補時，是以row by row的方式依序計算每個像素，而透過觀察(7)與(8)式可得知與Q點同處相同row上所有的待插補點執行垂直插補所需的垂直係數(Vertical Coefficient)$vc1$、$vc2$、$vc3$與$vc4$是相同的，所以對同一個row上所有的待插補點而言只須計算一次的垂直係數便可套用至此row上的所有待插補像素，之後僅需要計算各插補點的水平係數(Horizontal Coefficient)$hc1$、$hc2$、$hc3$與$hc4$即可，如此一來便可以大幅降低係數計算時所需之運算量。

記憶體讀取方面，在一維分解架構中由於垂直插補僅需要讀取待插補row上之虛擬像素(Virtual Pixel)的垂直方向上下各兩個原始像素，如圖(6)*V1*的上下兩個像素*A*、*B*、*C*、*D*，而水平插補則讀取水平方向之虛擬像素，如如圖(6)的*V1*、*V2*、*V3*、*V4*。因此當垂直插補對外部記憶體讀取像素進行一個虛擬像素的插補時，每一組被垂直插補所使用的原始像素都僅需要被讀取一次，如圖(6)在進行*Q*點所屬row的插補時，以*A*、*B*、*C*、*D*或*E*、*F*、*G*、*H*與其它四個像素為一組的原始像素都僅需要被讀取一次，相較於傳統二維插補，每組像素被讀取的次數不定且多次而言，二維插補之一維分解架構將可大幅降低對記憶體讀取次數。

(四) 簡化線性拓展插補法之插補係數

當線性拓展插補法的$\beta = 0.125$時，是影像插補效果與硬體電路成本間可取得的最佳設定值，而透過圖(7)我們可以得知在進行一維插補時，各原始像素與插補點間的距離關係，進而將各距離帶入(6)式中可得到(9)式，再經過數學式的展開步驟後可得到(10)式，至此(10)式便符合二維插補之一維分解架構中Coefficient Generator的輸入為一距離值，輸出為四個插補係數之硬體規格。

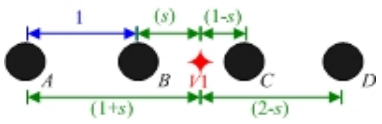


圖 7 一維插補距離示意圖

$$\begin{aligned} c_0 &= 0.125\times(1+s)-0.25 \\ c_1 &= (0.125-1)\times s+1 \\ c_2 &= (0.125-1)\times(1-s)+1 \\ c_3 &= 0.125\times(2-s)-0.25 \end{aligned} \quad (9)$$

$$\begin{aligned} c_0 &= 0.125\times s-0.125 \\ c_1 &= -0.875\times s+1 \\ c_2 &= 0.875\times s+0.125 \\ c_3 &= -0.125\times s \end{aligned} \quad (10)$$

(五) 插補座標定位系統

在影像進行插補計算時，必須將原始影像以及插補影像放置到同一插補座標平面上進行計算，而且使用的插補座標系統必須同時包含整數以及小數，因此在影像進行插補時首先便是根據scale factor逐一計算出目前插補座標系統中插補點之座標，然而原始影像在off-chip memory之位置則是另一種座標系統，例如，原始影像之原點在插補座標系統之座標為(0.5, 0.5)，而其在off-chip memory之位置則是(1, 1)，因此我們必須先定位插補點位於插補座標系統之座標，而後根據此座標作後續所有之運算。當系統要定位插補點*Q*(*i*,*j*)的座標時，便先計算出目前vertical interpolation所需插補之虛擬像素點的垂直座標(*j*)，定位時僅需定位出vertical interpolation中最上方原始像素之row address (*coord*1)即可，其餘的*coord*2、*coord*3與*coord*4因為與*coord*1必為相鄰座標，所以只需各別加上一常數便可求得，如式(11)，而*coord*1，其餘的*coord*2、*coord*3與*coord*4亦是相關之原始像素點在off-chip memory之row address。相對的，當要定位horizontal interpolation所需的座標時，亦先計算出目前插補點座標的水平座標，其使用公式與式(11)相同，只是變數(*j*)改為(*i*)。

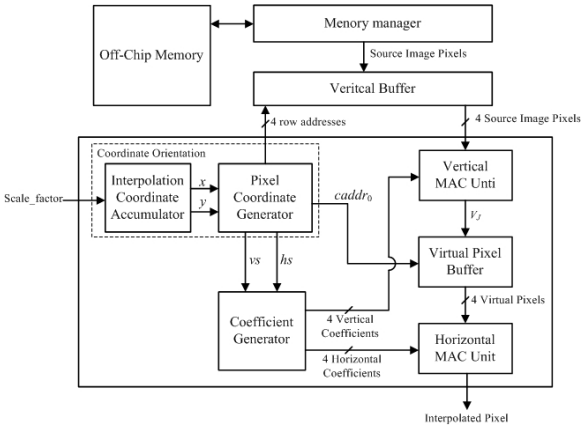
$$\begin{aligned} coord\ 1 &= \begin{cases} (integer\ \ of\ \ j) & ,fraction\ \ of\ \ j \geq 0.5 \\ (integer\ \ of\ \ j)-1 & ,fraction\ \ of\ \ j < 0.5 \end{cases} \\ coord\ 2 &= coord\ 1+1 \\ coord\ 3 &= coord\ 1+2 \\ coord\ 4 &= coord\ 1+3 \end{aligned} \quad (11)$$

經過我們化簡的係數產生公式(10)，若為產生 vertical coefficients (*vc*1、*vc*2、*vc*3、*vc*4)時，則必須先計算出 vertical 方向間距值 *vs*，式(12)便是產生此間距值所使用之公式，至於 horizontal 方向間距值 *hs* 所產生之公式與式(12)相同，只是變數(*j*)改為(*i*)。

$$vs = j - (coord1 + 0.5) \qquad (12)$$

三、高效能線性拓展插補法之硬體架構

圖(8)為線性拓展插補法之一維分解架構的電路方塊圖，分別由下列各方塊所組成 Coordinate Orientation Unit 、 Coefficient Generator、Vertical Multiply Accumulate (MAC) Unit、Virtual Pixel Buffer、Horizontal Multiply Accumulate (MAC) Unit。

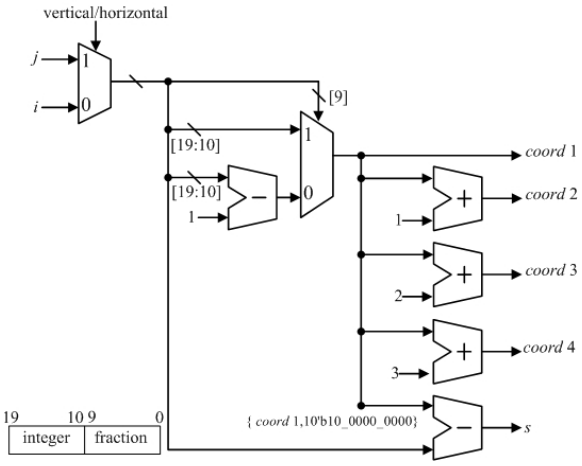


圖(8) 二維插補之一維分解架構電路方塊圖

(一) Coordinate Orientation Unit 電路架構

線性拓展插補法進行插補時需要使用插補點鄰近 16 個原始影像的像素，依我們的方法，其定位原始像素可獨立成兩個動作，第一便是在每個 row 的插補初始時定位出此 row 的垂直方向像素範圍，以便進行 vertical interpolation 時可以正確的從記憶體中讀取原始像素，以圖一為例，當開始插補 *Q* 點所在的 row 時必須先定位出 *A*、*B*、*C*、*D* 這四個像素所屬的四個 row address；第二則是於個別插補點執行插補運算時定位出其水平方向像素範圍，也就是四個虛擬像素所屬之 column address，以便進行 horizontal interpolation 時可以正確的從 Virtual Pixel Buffer 中讀取虛擬像素，以圖一為例，當進行到 *Q* 點的插補運算時必須定位出 *V*1、*V*2、*V*3、*V*4 這四個虛擬像素所屬的四個 column address，而上述兩個動作於電路運算時並不需要同時執行，因此僅需一套的 Coordinate Orientation 電路便可以符合所需。

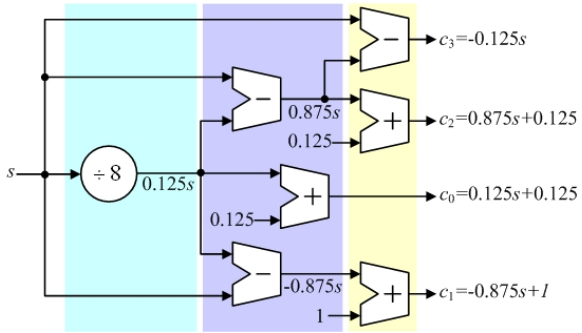
圖 (9) 為所設計的 Coordinate Orientation 電路，透過控制電路所發送的 vertical/horziontal 訊號判斷目前將進行垂直或水平的座標定位，以選取輸入目前插補點的垂直座標(*j*)或水平座標(*i*)進行定位，由於座標的資料格式為 20-bit，其中低位元 10-bit 為小數部分，而高位元 10-bit 為整數部分，因此座標中的第九位元為小數的 most significant bit，因此透過此位元便可以判斷此座標的小數部分是否大於等於 0.5，接著便可接續完成式(11)與式(12)的動作求得座標 coord1、coord2、coord3、coord4 與間距 *s*。若 vertical/horziontal 訊號為 vertical 時 *s* 為 *sv*，反之若 vertical/horziontal 訊號為 horziontal 時 *s* 則為 *sh*。



圖(9) Coordinate Orientation 電路架構圖

(二) Coefficient Generator 電路架構

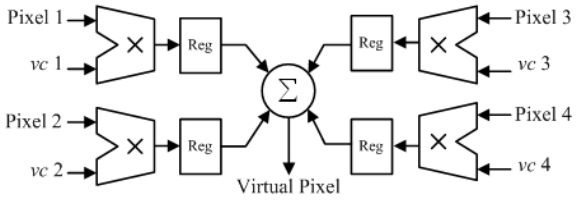
(10)式為經過我們的化簡的線性拓展插補係數方程式，由於所提出的架構中垂直係數與水平係數在插補時不需同時進行運算，因此僅需設計一組係數產生電路便可以同時產生垂直係數以及水平係數，圖(10)則是所設計的 Coefficient Generator 電路架構圖，當中透過控制單元所發出的 vertical/horizontal 訊號決定目前將產生垂直係數或是水平係數，經由圖(10)可以發現我們已將整個雙立方插補法中運算量最龐大的係數產生電路簡化至僅需使用兩個乘法器以及四個加法器便可產生完成所有插補所需之所有插補係數。



圖(10) Coefficient Generator 電路架構圖

(三) Vertical MAC Unit 與 Horizontal MAC Unit 電路架構

Vertical MAC Unit為執行垂直插補此的運算步驟，將從外部記憶體取得的四個原始影像像素Pixel1、Pixel2、Pixel3、Pixel4以及Coefficient Generator產生的四個垂直係數*vc1*、*vc2*、*vc3*、*vc4*進行迴旋積分，也就是執行個別相乘以及加總的動作，如(7)式。接著便可以得到一個虛擬像素(Virtual Pixel)並且存入Virtual Pixel Buffe中以供進行水平插補時使用，圖(11)為Vertical MAC Unit的電路架構圖。



圖(11)Vertical MAC Unit 電路架構圖

Horizontal MAC Unit是水平插補的最後一個步驟，也是單一插補點的插補運算最後一步驟，將從Virtual Pixel Buffer中取得的四個虛擬像素Virture Pixel1、Virture Pixel2、Virture Pixel3、Virture Pixel4 以及Coefficient Generator產生的四個水平係數*hc1*、*hc2*、*hc3*、*hc4*、進行迴旋積分，也就是執行個別相乘以及加總的動作，如 (8)式。接著便可以得到最終插補結果，其電路架構與Vertical MAC Unit的電路架構圖相同。

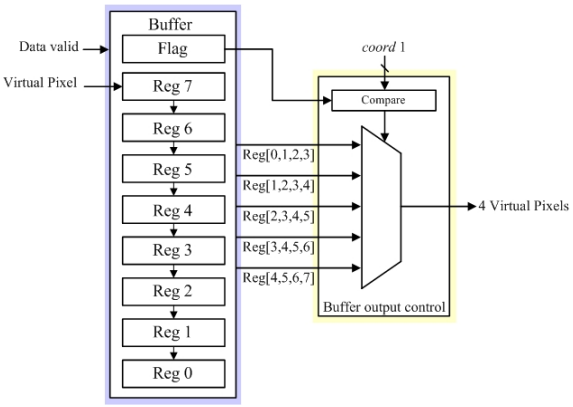
(四) Virtual Pixel Buffer 電路架構

當虛擬像素(Virtual Pixel)經由垂直插補產生後，將直接送入 Virtual Pixel Buffer 儲存，等待水平插補運算時進行提取使用。但當執行影像放大(Scale Up)的插補動作時 Virtual Pixel Buffer 的輸入 Pixel Rate 大於輸出的 Pixel Rate，也就是說垂直插補產生虛擬像素的速度比水平插補所需要的還要快，因此將導致 Virtual Pixel

Buffer 需要較大的記憶體空間進行虛擬像素的儲存；相反的，當執行影像縮小(Scale down)的插補動作時 Virtual Pixel Buffer 的輸入 Pixel Rate 小於輸出的 Pixel Rate，也就是說垂直插補產生虛擬像素的速度比水平插補所需要的還要慢，將使得水平插補無法使用有效的虛擬像素進行運算，導致錯誤發生。

因此我們將在垂直插補與水平插補之間加入了運算協調機制，在Scale up的插補運算時，當系統發現所產生的虛擬像素已經超越水平插補所需要的達一定幅度時，便暫停垂直插補的運算，以停止虛擬像素的繼續產生。暫停垂直插補運算時，則僅需暫停Address Generator中對垂直方向座標繼續累加之執行即可，直到領先幅度縮小時再繼續進行累加，藉此對Virtual Pixel Buffer而言便減少了大量存放的有效虛擬像素之空間，並搭配判斷所儲存的資料的有效性，將無效資料的記憶體空間由有效資料所取代，因此便可以大幅降低所需使用的記憶體空間，以達到降低硬體成本之效果；在Scale down的插補運算時，當系統判斷Virtual Pixel Buffer內有效的虛擬像素將要落到水平插補所需範圍外時，也就是垂直插補將要開始落後，此時便暫停水平插補的運算，等待垂直插補將水平插補所需的虛擬像素補齊後再開始運算，而暫停水平插補僅需暫停 Coordinate Accumulator中對插補點之水平座標(i)位址的繼續計數，便可以連帶停止Coordinate Orientation 與 Coefficient Generator 的動作，所以水平插補的動作將完全暫停，藉此將避免水平插補讀取到所需範圍外的虛擬像素導致計算出錯誤的結果像素。

圖(12)為 Virtual Pixel Buffer 的電路圖，當中僅需使用八個暫存器做為記憶單元，便可以同時滿足 Scale Up 與 Scale Down 的需求。在影像放大時，垂直插補才有可能處在暫停狀態，當垂直插補處在暫停狀態時，輸入訊號 Data valid 將為除能(Disable)，告知 Virtual Pixel Buffer 此時的虛擬像素將是無效的，不需儲存入 Virtual Pixel Buffer 中，接著使用一個旗標(Flag)紀錄 Reg4 是儲存的虛擬像素是經由哪一個 column 的垂直插補所產生，並搭配一個比較器與多工器便可以相對位址判斷目前水平插補所需的四個虛擬像素為哪一組，並將其輸出，可明顯看出搭配運算協調機制下 Virtual Pixel Buffer 設計相當精簡。



圖(12) Virtual Pixel Buffer 電路架構圖

四、實驗結果及 VLSI 硬體實現

在本論文中，我們直接對電路的功能進行驗證，且以 Peak Signal-to-Noise Rate (PSNR)作為我們評斷影像品質的基準，並使用圖(13)共八張測試影像對所設計的電路進行測試，且本篇論文所提出的方法將與雙立方插補法方法作比較。

(a)Tank (b)Sailboat


(c)Boat (d)Bridge


(e)Goldhill (f)Lena


(g)Peppers (h)Airplane

圖 13 測試影像之原始圖片

我們測試的方法分別為將原始影像之解析度以 2/3 比例先縮小後再放大，然後再使用相同的插補法將解析度以 2/3 比例先放大後再縮小，最後將結果影像(Scaled Image)與原始影像進行 PSNR 的評比所得之數據紀錄於表(1)。

表 1 硬體影像插補品質 PSNR 比較表

	以 2/3 比例先縮小後再放大		以 2/3 比例先放大後再縮小	
	Extend Linear	Bi-cubic	Extend Linear	Bi-cubic
Tank	33.33	33.06	35.29	35.63
Sailboat	30.75	30.39	32.12	32.29
Boat	31.03	30.76	32.11	31.95
Bridge	27.27	26.91	29.14	29.17
Goldhill	31.95	31.75	34.00	34.20
Lena	33.97	33.73	35.21	35.17
Peppers	33.22	32.88	34.41	34.62
Airplane	33.12	33.05	34.21	34.06

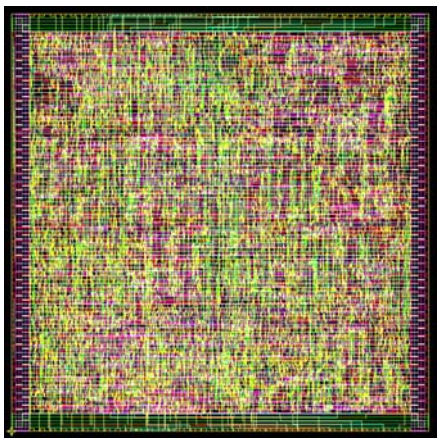
在所提出的架構中，藉插補係數函式(10)之化簡，可以發現於產生插補係數所需使用的乘法器與加法器較原始雙立方插補法有相當大幅度的縮減，如表(2)。

表(2) 運算複雜度比較表

Algorithm	Bilinear	Winscale	Bicubic	Proposed
Pixel Moving	2 add	6 add	2 add	2 add
Weighting Factor Calculation	3 mul 2 add	6 mul 2 add	16 mul 36 add	0 mul 6 add
Pure Filter Operations	4 mul 3 add	4 mul 3 add	16 mul 15 add	8 mul 6 add

我們採用Verilog硬體描述語言撰寫此硬體架構，並且使用Design compiler進行電路的合成，透過Astro完成放置與繞線的動作後，得到整個VLSI晶片的實現。最後，我們進行Post-layout的Verilog模擬，得到的模擬結果與Gate-level模擬是相同的。圖(14)為透過Astro所繞出的實體佈局圖(Layout)。使用TSMC 0.13μm製程下，Gate counts 為 26200，整個晶片面積為 452×452 μm^2，並且在不考量輸出腳位的情

況下的速度可達 267MHz。表(3)為此晶片之其他面積以及功率的相關參數值。



圖(14) VLSI 實體佈局圖

表(3) 晶片參數

Processing	TSMC 0.13μm
Core area	452×452 μm^2
Gate Count	26200
Power	18.14mW
Frequency	267MHz

五、參考文獻

[1] C.H. Kim, S.M. Seong, J.A. Lee, L.S Kim, "Winscale: an image-scaling algorithm using an area pixel model," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 549-553, June 2003.

[2] E. Meijering, M. Unser, "A note on cubic convolution interpolation," *IEEE Trans. Image Processing*, vol. 12, no. 4, pp. 477-479, April 2003.

[3] M.A. Nuno-Maganda, M.O. Arias-Estrada, "Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling," *International Conference on Reconfigurable Computing and FPGAs*, Sept. 2005.

[4] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Signal Processing*, vol. 29, no. 6, pp. 1153-1160, Dec. 1981.

[5] T.M. Lehmann, C. Gonner, K. Spitzer, "Survey: interpolation methods in medical image processing," *IEEE Trans. Medical Imaging*, vol. 18, no. 11, pp. 1049-1075, Nov. 1999.