# 第五章 影像處理相關硬體單元設計

影像處理單元(IPU)為整體系統的核心，IPU 可將 ICU 所擷取的影像進行第二章所談的高通、低通、高斯平滑、最大值、最小值與中間值濾波六種影像濾波以得到目標影像。其中高通、低通、高斯平滑濾波必須透過迴旋運算得到，而最大值、最小值與中間值濾波，則經由排序運算得到。

對於迴旋運算器的設計為本研究的重點，因此以下將先針對前人如何設計迴旋運算器進行說明，之後對其架構進行缺點分析，並提出新型的迴旋運算器設計方式。最後一節再說明排序濾波硬體，如何以最精簡的方式，整合至迴旋運算器硬體。

## 第一節 傳統之二維影像空間濾波器硬體架構

迴旋運算硬體廣泛運用於 DSP 處理晶片，至今已有相當的文獻討論其硬體實做方式，但硬體架構多大同小異，經統整提出作者 Crookes 於文獻[2][4][10][11]採用的架構，如圖 5-1 所示，此架構又稱為二維迴旋運算器(2-D Convolver)[3] [13][16] [17]。

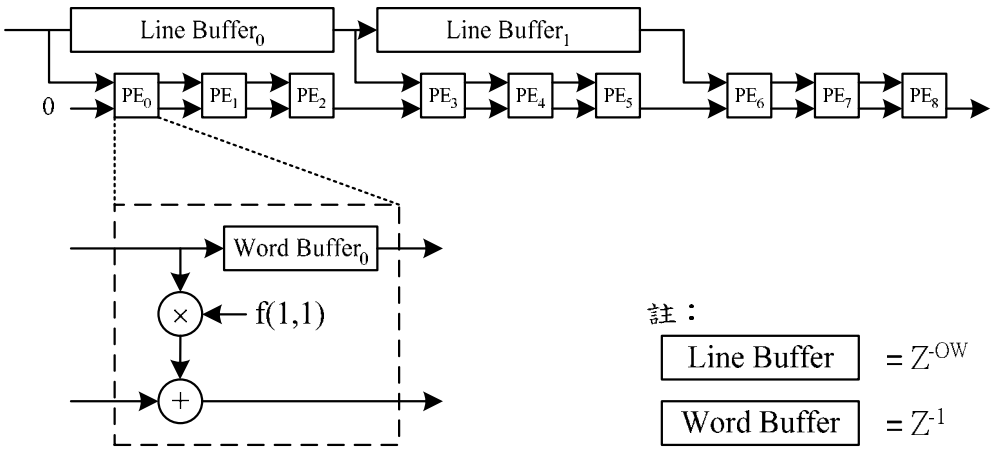



圖 5-1. Crookes 提出之 2-D Convolver 硬體架構

圖 5-1 是預設使用 3X3 大小的 Filter 與 BImg(FW=FH=BW=BH=3)。圖 5-1 的每個 Line Buffer，會將輸入訊號延遲 OW 個週期再輸出，在數位電路可視為 OW 大小的 FIFO 或移位器，內部將暫存最新輸入的 OW 筆資料，亦可將其表示為 $Z^{-OW}$，而 Word Buffer 表示 1 個延遲，亦可表示為 $Z^{-1}$。為方便之後的解說，將圖 5-1 重新表示為圖 5-2 的訊號流程圖。

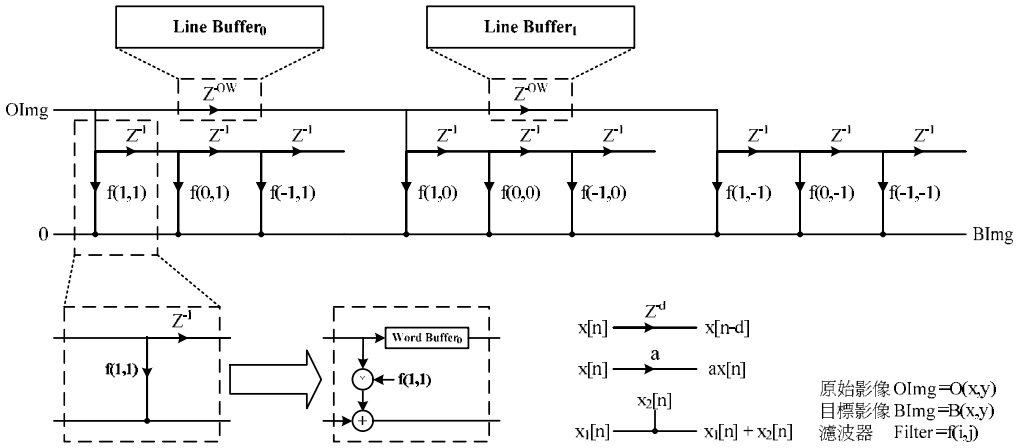



圖 5-2. Crookes 提出之 2-D Convolver 之訊號流程圖

令 ICU 所擷取的原始影像長、寬各為 10，當 OImg 送至 O(5,5)時，Line Buffer 與 Word Buffer 所暫存的像素，如圖 5-3 所示。

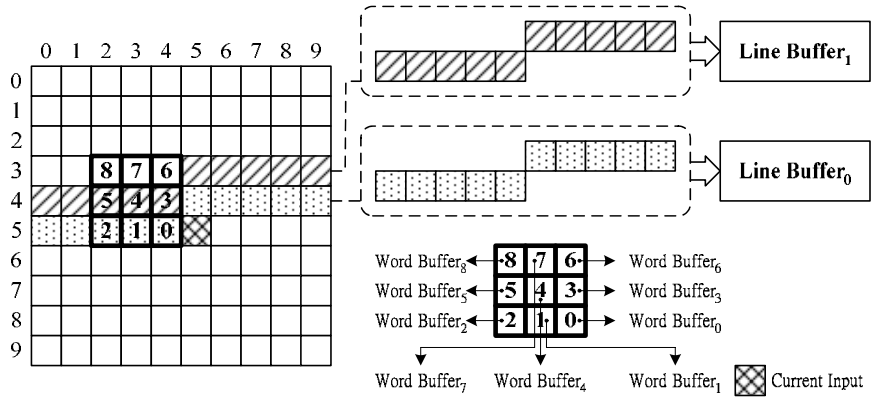



圖 5-3. OImg 為 O(5,5), Crookes 之 Line Buffer,Word Buffer 存放像素

此時圖 5-3 被粗黑框選取的像素，皆被暫存於 Word Buffer，每個 Word Buffer 負責暫存一個像素，圖 5-3 含有左斜線的像素，存放於 Line Buffer$_1$，含有網點的像素，則存放在 Line Buffer$_0$，若將圖 5-3 再對應至訊號流程圖，則詳細情形如圖 5-4 所示。

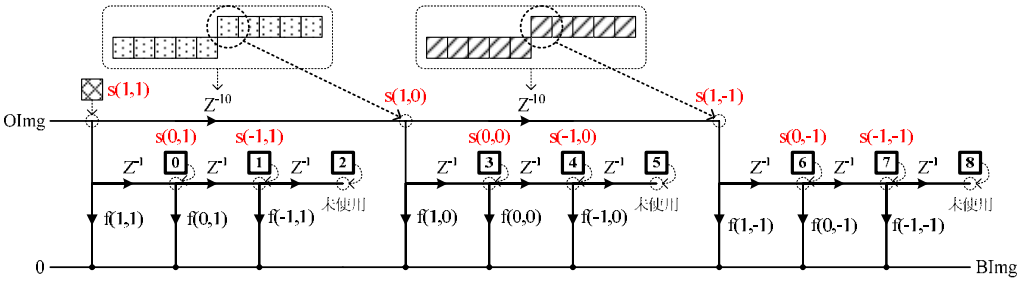



圖 5-4. OImg 為 O(5,5), Line Buffer 與 Word Buffer 所存放之像素

由圖 5-4 可知，Word Buffer$_2$、Word Buffer$_5$ 與 Word Buffer$_8$ 所存放的像素，未參與迴旋運算，反而是 Current Input、Line Buffer$_0$ 與 Line Buffer$_1$ 參與運算。因此實際參與運算的影像範圍，如圖 5-5 粗黑虛線框內的像素，這群像素也就是第二章所談，由原始影像抽取出來的子影像 SImg。

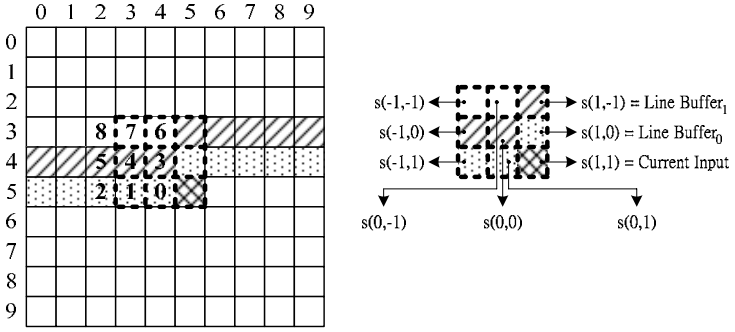



圖 5-5. 使用 Line Buffer 與 Word Buffer 取出之 SImg

依照 Crookes 的訊號流程圖，這群抽取出來的 SImg 再跟 filter 的係數進行運算並累加總和，計算得到之總和，即為濾波之後的 BImg。

## 第二節 改良之二維影像空間濾波器硬體架構

壹、精簡暫存器

若將圖 5-4 的 Line Buffer 的 $Z^{-10}$，拆解為 3 個 $Z^{-1}$ 與 $Z^{-7}$，並以圖 5-6 呈現，觀察圖 5-6 的資料流動方向即虛線部分。可發現 Word $Buffer_0$ ~Word $Buffer_2$ 的 3 個 $Z^{-1}$ 與 Line $Buffer_0$ 拆解出來的 3 個 $Z^{-1}$ 為重覆存放相同的像素，類似的情形也出現在 Word $Buffer_3$~Word $Buffer_5$ 及 Line $Buffer_1$，因此將之合併以減少硬體資源的浪費。

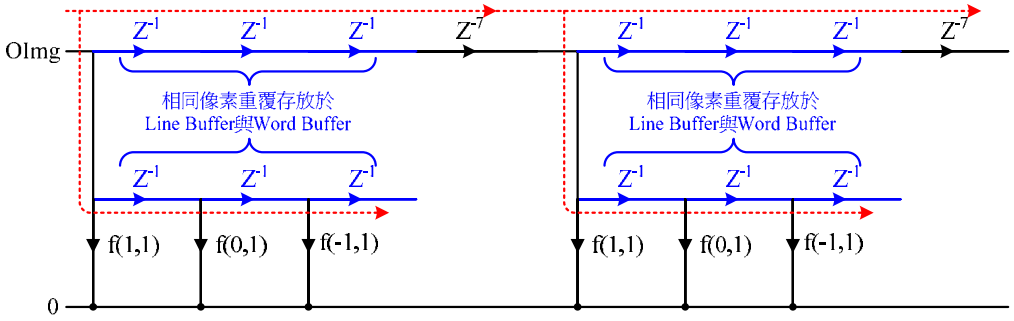



圖 5-6. Line Buffer 與 Word Buffer 重複存放像素示意圖

貳、調整延遲與運算元件次序，以確保 Convolver 運算同步

OImg 為 PAS109 所送出，為避免 PAS109 送出像素與 Convolver 接收像素的頻率不同而產生錯誤，所以將 Filter 的運算皆往後挪一級，此時 OImg 會先經過 $Z^{-1}$ 才進行運算，電路的意義即於輸入端加入一正反器，此正反器唯有於系統特定的條件成立時(例如：正邊緣、負邊緣...等等)，才會擷取 PAS109 送出的像素；若條件不成立，PAS109 訊號便不會影響 Convolver 內部訊號的運算，如此便可確保 Convolver 內部能運算同步。

綜合上述兩點所談，提出修改後的硬體迴旋運算器，如下圖所示。

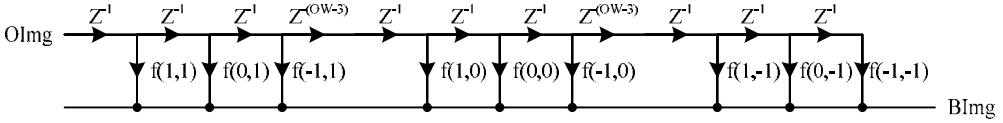



圖 5-7. 合併部分 Line Buffer 與 Word Buffer 的 2-D Convolver

就圖 5-7 與 Crooks 的 Convolver 比較，圖 5-7 仍保有 9 個 Word Buffer，但每個 Line Buffer 均省掉 3 個暫存器，因此共省去 6 個暫存器，更詳細的說，即 Crooks 需要(2OW+9)個延遲元件來實現，現在可精簡為(2OW+3)即可完成，並且可確保 Convolver 運算同步。如同樣以圖 5-3 的例子，當 OImg 送至 O(5,5)時，Line Buffer 與 Word Buffer 所存放的像素，如圖 5-8 所示。

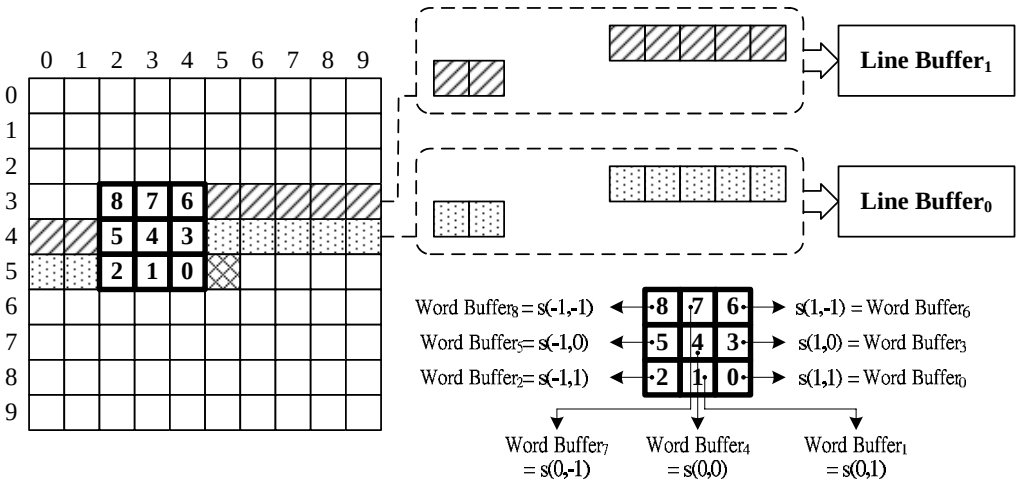
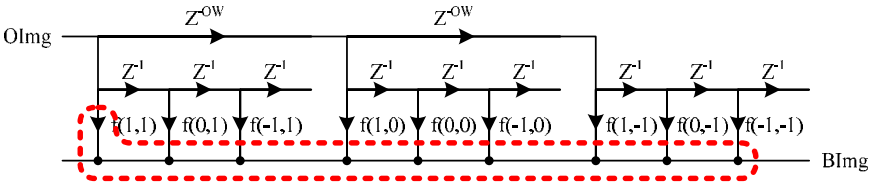



圖 5-8. OImg 為 O(5,5)，圖 5-7 之 Line Buffer,Word Buffer 存放像素

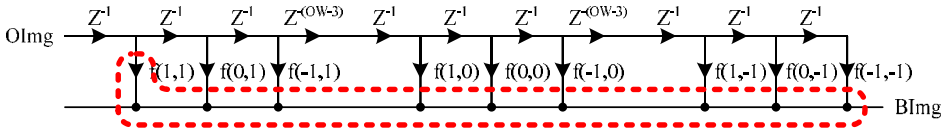
由圖 5-8 與圖 5-3 比較，可發現粗黑框所選的像素並沒與網點或左斜的像素重疊，亦即同位址的像素，不會同時存在 Line Buffer，又存在 Word Buffer，由此可見，圖 5-7 架構確實可節省暫存器的使用。

參、減少連加路徑的延遲時間

不論 Crookes 或圖 5-7 的架構，都有共同長路徑的問題，其最長路徑為 9 個加法器及 1 個乘法器造成的延遲，此長路徑將造成系統運算資料傳遞相當慢，進而影響工作效能變差，如下圖粗虛線所選之區域。




(a) Crookes 架構隱含之加法器長路徑




(b) 圖 5-7 架構隱含之加法器長路徑

圖 5-9. Crookes 及圖 5-7 Convolver 之延遲長路徑(粗虛線圈選區)

為改善其延遲路徑，因此將圖 5-7 的資料延遲路徑稍做修改，如圖 5-8 所示。此改善方式又稱為 Data Broadcast Structures，其濾波功能等效於圖 5-7，且可發現其長路徑已由 9 個加法器減少為 1 個加法器及 1 個乘法器。

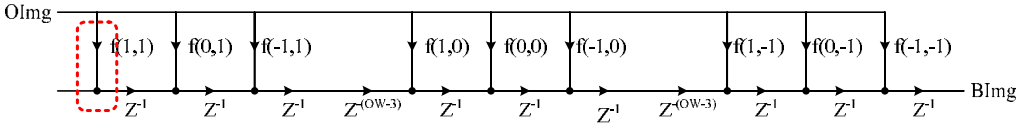



圖 5-10. 經由 Data Broadcast Structures 方式改善的 Convolver

肆、用 FPGA 內建的 Block Memory 做動態佇列，以達動態調整影像大小

ICU 可經由 ICU_BV、ICU_EV、ICU_BH、ICU_EH 設定取像的大小，但若硬體 Convolver 無法隨影像大小調整而適應調動，則 ICU 可隨時設定影像大小的功能亦為空談。

由圖 5-10 可知架構中，唯有原始影像寬度 OW 參數與 Convolver 的設計有關，因此不需變動任何硬體，即可適應動態影像高度。因此以下將問題縮小，鎖定於由 MAU 得知 ICU_BH、ICU_EH 並算出 OW，如何去變動圖 5-10 的 $Z^{-(OW-3)}$延遲元件。由於 Crooks 架構假設影像大小為固定，因此不對此問題研討，所以之後討論，將對華春和[28]之動態調整影像寬度的方式進行比較。

一、華春和之動態調整影像寬度方式

圖 5-11 為華春和所採用的影像寬度調整方式，於其論文又稱為 Pipeline FIFO，亦可視為取代 Crooks 的 Line Buffer。Pipeline FIFO 可透過多工器的選擇線 IMGD，決定延遲為 $Z^{-128}$ 或 $Z^{-256}$，因此以華春和的方式而言，僅可適應兩種不同的影像寬度。

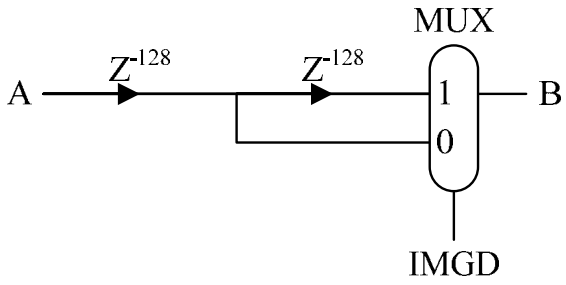



圖 5-11. 華春和提出之動態調整影像寬度方式

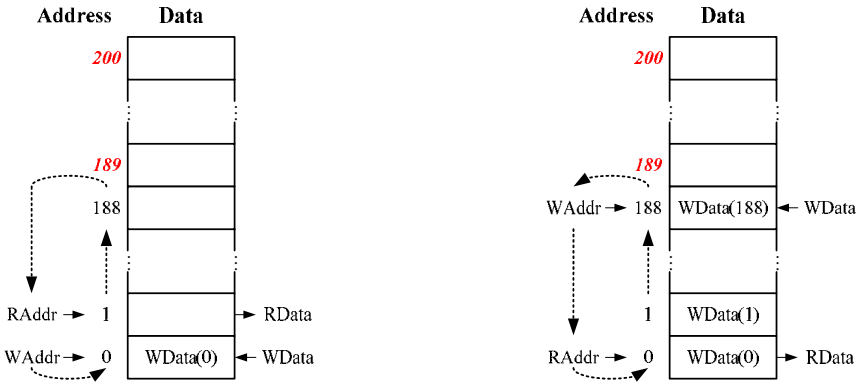
三、改良式之動態調整影像寬度方式

早期 FPGA 使用者若要在晶片暫存資料，必須使用晶片內部的閘數組合出記憶元件，但這樣將耗去相當的閘數。因此目前 FPGA 多內建記憶體，獨立於一般可規劃閘數之外，使用這些記憶體將不耗掉閘數，這些內建記憶體又稱為 Block Memory。

利用此利基，提出以雙埠 Block Memory 為基礎之動態可調大小的 FIFO，以下簡稱為 DFIFO，雙埠的 Block Memory 具有兩組可讀寫的位址、資料匯流排，但實際使用時，一組只當寫入、另一組只供讀取。寫入組的位址匯流排稱為 WAddr，資料匯流排為 WData；讀取組的位址匯流排稱為 RAddr，資料匯流排為 RData。為使此雙埠的 Block Memory 具 DFIFO 的功能，以下必須進行三項定義：

1. 雙埠 Block Memory 之大小為 Maximum Range。
2. WAddr 與 RAddr 皆為兩組除 Current Range 的計數器，且 RAddr 永遠領先 WAddr 一週期。
3. 系統重置時，WAddr 為 0，RAddr 為 1。

假設 Maximum Range 為 200，Current Range 為 189，系統重置時，WAddr 為 0，RAddr 為 1，如圖 5-12(a)。經過 189 個週期後，WAddr 由 0 遞增至 188，此時間 WData 也依次序寫入資料 WData(0)~WData(188)，而 RAddr 由 1 遞增至 188 又折回 0，此時 RData 開始讀出 189 個週期前，所存放至 Block Memory 的 WData(0)，如圖 5-12(b)所示。依此動作類推往後的週期，RData 讀取的資料，皆會是 189 個週期前，WData 所存入的資料，此動作即可視為大小 189 的 FIFO，於 DSP 上的意義即為 $Z^{-189}$。因此若能調整 Current Range，便能得到 $Z^{-\text{Current Range}}$ 大小的 FIFO。




(a)週期 0 之記憶體寫入與讀出狀態 (b)週期 189 之記憶體寫入與讀出狀態

圖 5-12. Current Range 為 189，記憶體之寫入與讀出狀態

簡而言之，只要將 RData 當為 DFIFO 的資料推出口，WData 作為 DFIFO 的資料吸入口，經由調整 Current Range，即可把此架構視為 DFIFO，為方便表示，使用以下符號來代表 DFIFO。

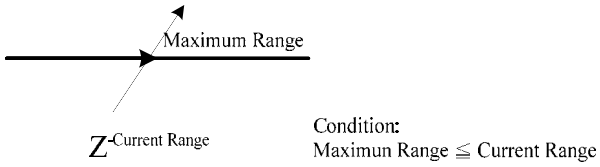



圖 5-13. 動態可調大小 FIFO(DFIFO)符號

實做時根據 PAS109 的大小，將上圖的 Maximum Range 訂為 200 即足夠使用。綜合上述 DFIFO 的改進，可得到以下最終版的濾波訊號流程圖。

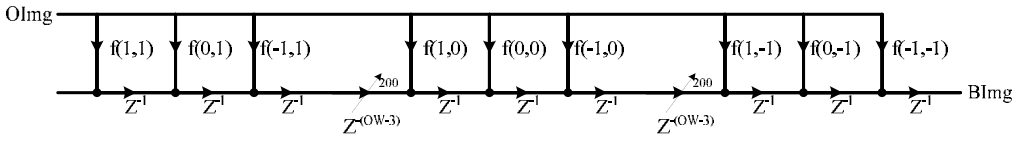



圖 5-14. 最終版的迴旋運算器架構

如本節開始所談，為因應 ICU 可動態選擇取像的影像範圍，須設法可動態改變圖 5-10 的 $Z^{-(OW-3)}$的大小，但如華春和的架構，所能變動的影像範圍實為有限，反觀本研究所採用的 Block Memory 為基礎的 DFIFO，將原本複雜的問題，簡化至一動態計數器的設計，不但加強可調性，亦利用 FPGA 本身特點，節省閘數的使用量。

將圖 5-14 所有的 $Z^{-1}$ 與 DFIFO 所建構的 $Z^{-(OW-3)}$串接起來，可視為 2OW+3 大小的 FIFO，在 Convolver 必須依賴這些 FIFO 來暫存運算過程的資料，但同時也造成像素由 OImg 輸入，一直到 BImg 輸出有效像素，將因這些 FIFO 延遲 2OW+3 個週期，此延遲期間稱之為 Latency。由於採用圖 5-14 來設計，因此 Latency 將隨影像大小而不同，但固定可用 2OW+3 來表示。

## 第三節 二維影像排序濾波器硬體架構

近幾年提出之硬體排序演算法，可以分為*位元式排序(Bit Sort)* [15][25]、*字組排序(Word Sort)* [9][19][23][26]，位元式排序是利用位元間，權重值來互相進行比較，對於需要快速排序需求有較佳的效果，但是每次運算完，只能得到最大值、最小值或中間值其中一種，因此若要同時得到三種需求硬體將需要大幅擴增。位元式排序另一缺點為內部連線、排序流程複雜，並需要位元流入出的硬體架構。

字組排序顧名思義，以字組為單位進行排序，此類論文與位元式排序相似，多針對最大值、最小值與中間值各有不同的設計，若依本研究需求，需要最大值、最小值與中間值三種皆要達成，則需要三套不同的硬體，在硬體資源上並未比位元式排序節省。

經搜尋相關文獻，發現作者 Maheshwari 於文獻[19]提出的字組式排序方式恰好符合本研究需求，其演算法可用同一硬體、流程排序後，同時取得最大值、中間值與最小值，且分析其架構也可以配合 Convolver 的部分架構，因此以下針對此演算法提出說明。

壹、Maheshwari 排序演算法

作者 Maheshwari 提出的演算法，基礎建立於*三輸入排序器(Triple Input Sorter)*，其功能如圖 5-15 所示。在圖 5-15 箭頭所指處為數值較大的位置，反之則是數值較小的擺放位置，假設 136,40,79 三像素進行排序，則經過排序後，箭頭所指將為最大值 136，其次為 79 與 40。

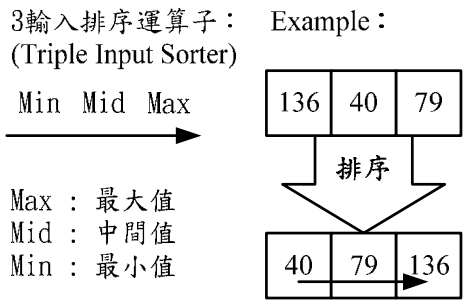
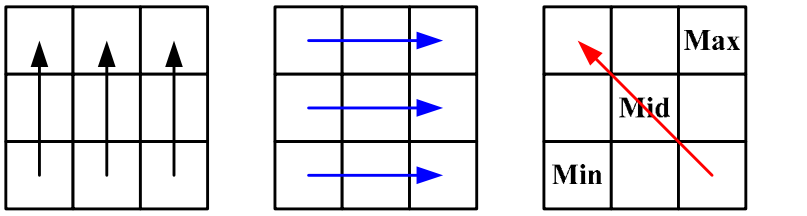



圖 5-15. 三輸入排序器符號定義

接下來就是使用三輸入排序器，取出 SImg 的九個像素之最大值、中間值與最小值，排序的動作需要三步驟來完成，第一步驟稱為”垂直排序”，即對分別的垂直三元素進行排序，工作示意如圖 5-16(a)。將垂直排序的結果再進行”水平排序”，工作示意如圖 5-16(b)，最後進行”對角排序”，工作示意如圖 5-16(c)。




(a)垂直排序　　(b)水平排序　　(c)對角排序

圖 5-16. Maheshwari 提出之三步驟排序演算法

經過上述排序過程，可在排序完畢之影像左下至右上的對角線找出最小值(Min)、中間值(Mid)與最大值(Max)，此演算法之優點為可同時找出三個排序值，而不需為求得其中一值而發展專用的演算法。根據初步統計，此排序方式只需要 15 個以下的雙輸入排序器，而以傳統的氣泡排序法，則需要 32 個雙輸入的排序器，因此在硬體資源的耗費下可以大大節省。

## 貳、Maheshwari 硬體排序架構

### 一、排序器硬體化

Maheshwari 排序演算法，須以三輸入排序器為基礎，而三輸入排序器為利用氣泡排序的觀念所設計，即兩兩變數互相進行比較。因此每個三輸入排序器包含三個*二輸入排序器(Dual Input Sorter)*，如圖 5-17 所示。

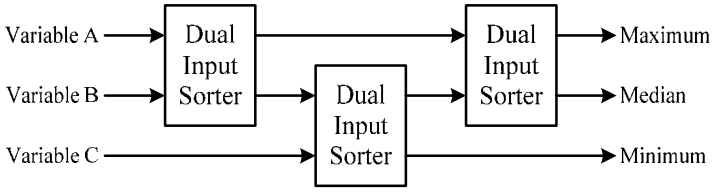



圖 5-17. 三輸入排序器之硬體方塊圖

每個兩輸入排序器將其展開，如下圖所示。

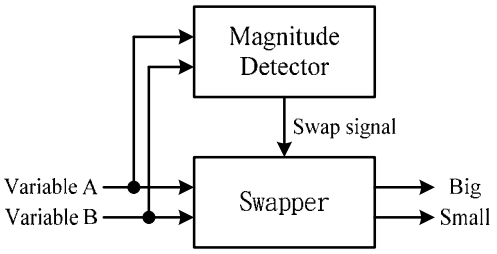
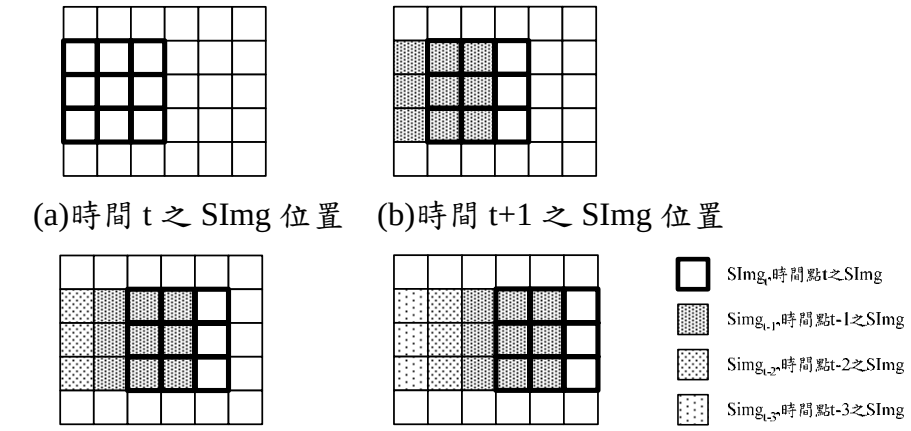



圖 5-18. 二輸入排序器之硬體方塊圖

### 二、簡化排序器數量

根據圖 5-16 共需要 7 個三輸入排序器，所以需要 21 個二輸入排序器，比起傳統的氣泡排序法，需要 32 個排序器已經少了很多。但若再分析 SImg 的取出方式，可再予以精簡，圖 5-16 列出相鄰 4 個時間取出 SImg 的相互關係，可發現兩個連續時間取出的 SImg，必有 6 個像素重覆，且這 6 個像素恰好是 2 組垂直排序的結果，因此可將這兩組垂直排序的結果，保留給下一時間作為垂直排序的部分結果，如此一來便可再精簡 6 個二輸入排序器。




(a)時間 t 之 SImg 位置　(b)時間 t+1 之 SImg 位置

(c)時間 t+2 之 SImg 位置　(d)時間 t+3 之 SImg 位置

圖 5-19. 相鄰四個時間點之 SImg 的運作位置

三、硬體方塊設計

如上說明，由於垂直排序可拿前一時間，已經垂直排序過的 6 個像素繼續使用，所以要進行新的一次排序，只需讀入新的 3 個像素便可排序，即圖 5-19 粗黑框的右邊 3 個白底的像素，將上述演算步驟與精簡方法整合，規劃出以下的硬體濾波器方塊圖。

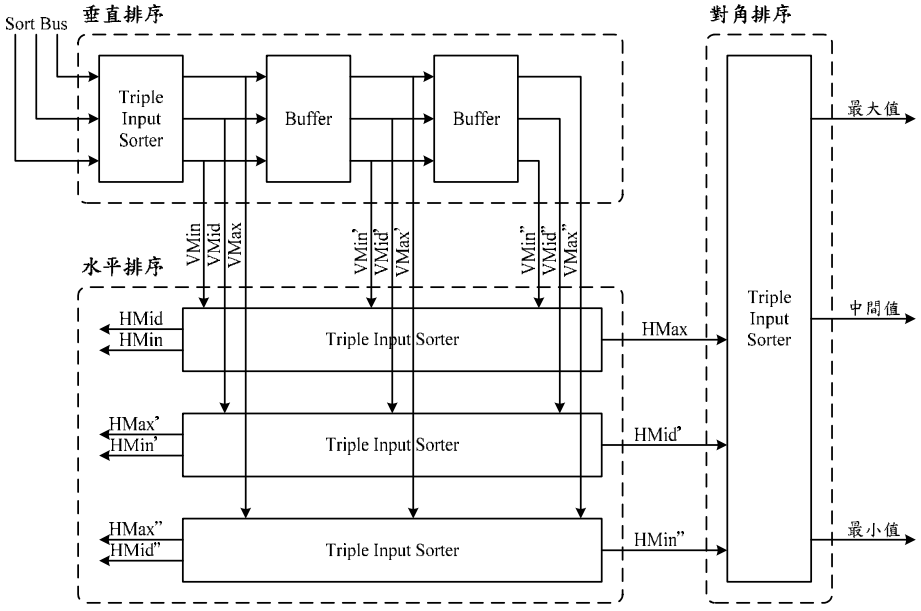
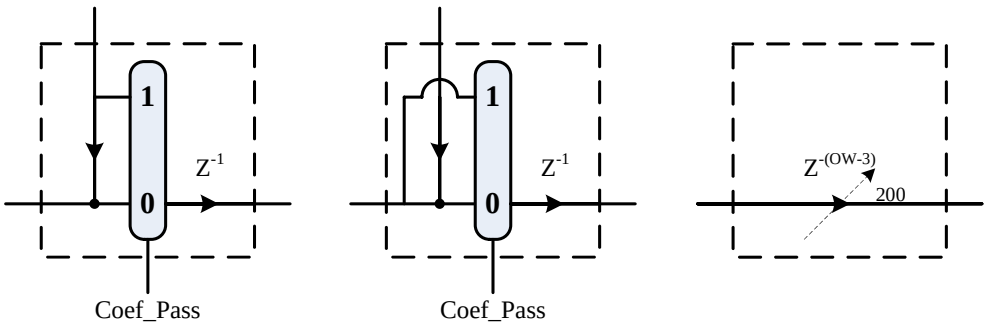



圖 5-20. 依據 Maheshwari 排序演算法發展之硬體方塊圖

## 第四節 IPU 硬體架構設計

在上兩節的內容，已經對硬體迴旋運算器及排序器的硬體設計進行說明，本節主要任務，則是探討如何將兩種濾波器整合為一。為後續講解方便，以下將對圖 5-14 的架構做些調整，以順利讓排序硬體與迴旋濾波器部分的資源整合使用。但進行調整前，必須先定義下列基本的元件。




(a)Start Element(SE)　(b)Process Element(PE)　(c)Delay Element(DE)

圖 5-21. IPU 基本元件訂定

接著利用 SE、PE 與 DE，將圖 5-14 架構調整如圖 5-22 所示。SE 必須作為像素輸入端的第一個元件，之後便以使用 PE 來進行串接，DE 則用來取代圖 5-14 的 DFIFO。比較圖 5-22 與圖 5-14，可發現主要差別為加入多工器，當多工器選擇線 Coef_PASS 為低準位，此時架構等效圖 5-14 的迴旋運算器，表示要進行線性濾波。若 Coef_PASS 為高準位，表示要進行非線性濾波之排序運算，此時只有 $Z^{-1}$ 與 $Z^{-(OW-3)}$作用，簡化後的等效路徑如圖 5-23，與圖 5-19 比較可發現圖 5-23 選出 Sort Bus 的 3 個像素，即第三節談到每次濾波所需要的 3 個像素，只要將這 3 個像素，連接圖 5-20 的 Sort Bus，便可使排序濾波的硬體正常工作。

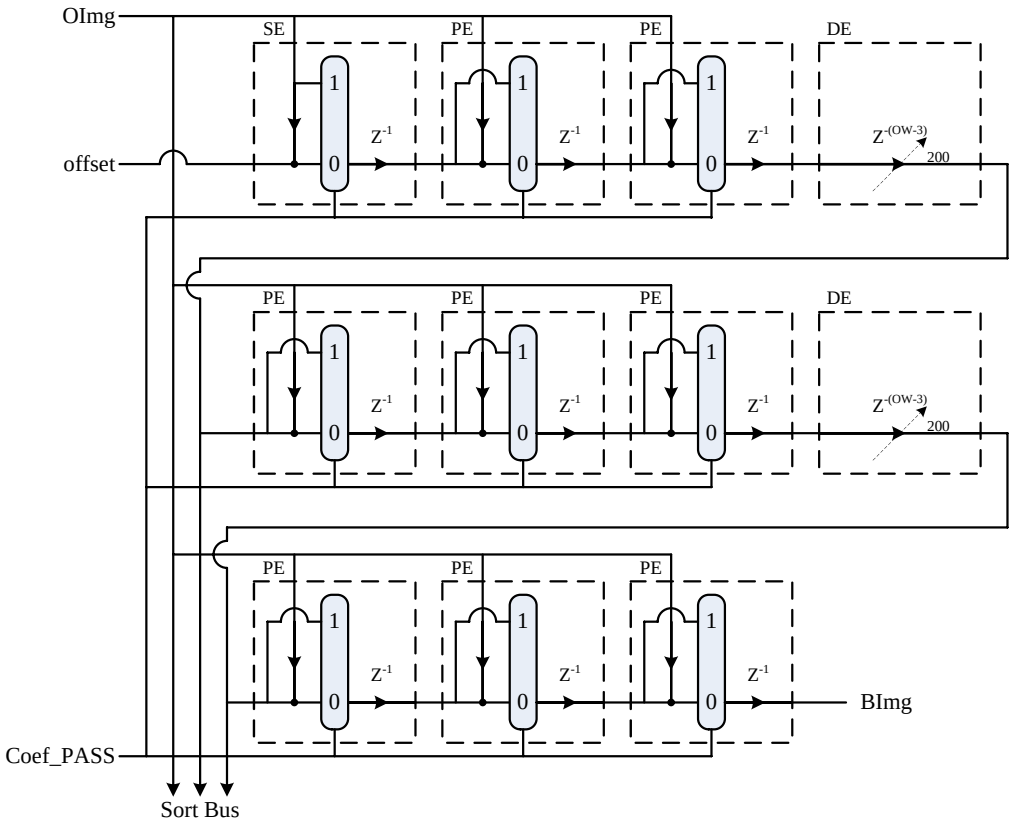



圖 5-22. 以 SE、PE 與 DE 所組成之迴旋運算器

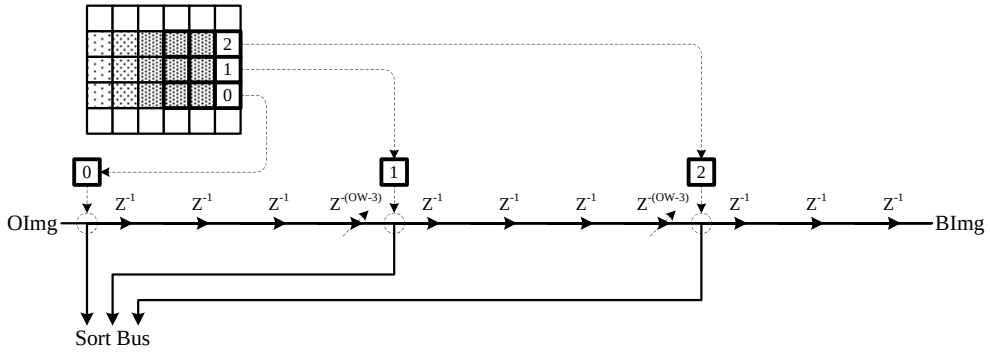



圖 5-23. Coef_PASS 為高準位的等效路徑

圖 5-22 的 offset 可用於疊加偏移值至影像，如藉由疊加常數偏移值，可用於調整影像亮度。但本研究中，則將 offset 作另一種應用，當進行高通濾波後，影像僅剩高頻的部分，但以人眼觀察對於僅有高頻的影像較不

敏感，因此一般會將高頻部分疊加至原始影像再顯示，如第二章於圖 2-8 所討論，而圖 5-22 的 offset 就是需要疊加原始影像的實現方式。

根據前幾節說明，最後將 IPU 迴旋運算器及排序器整合為以下硬體方塊：

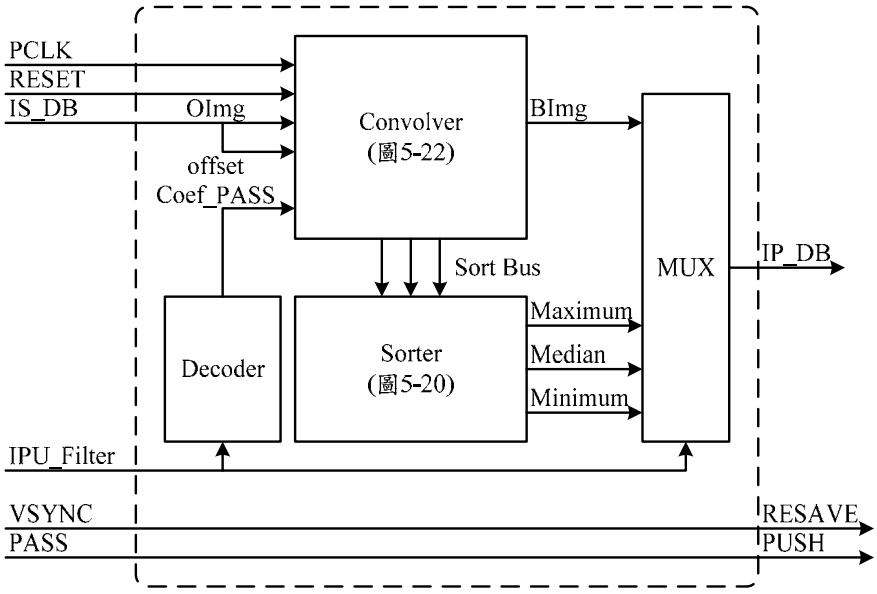



圖 5-24. IPU 之內部硬體方塊設計