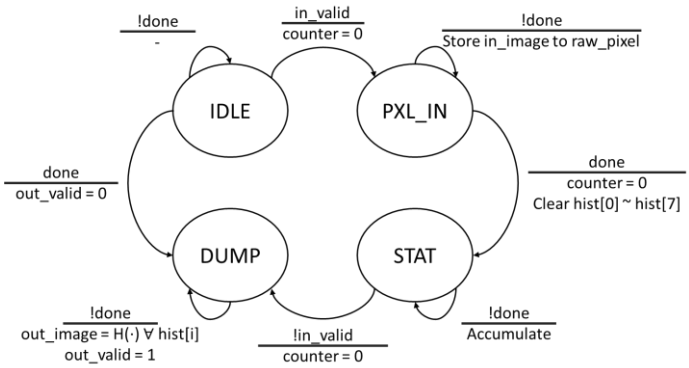
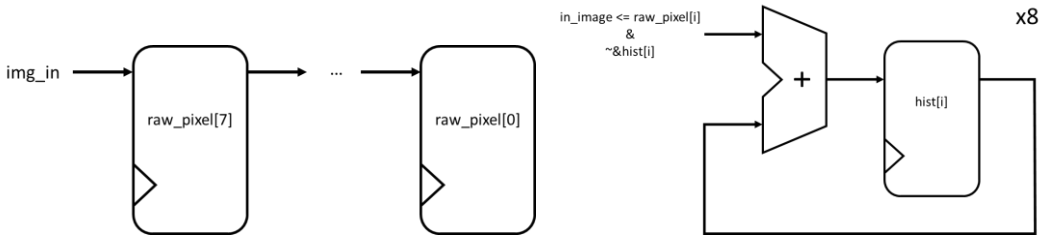
數位電路與系統 HW4 Report

110511233 李承宗

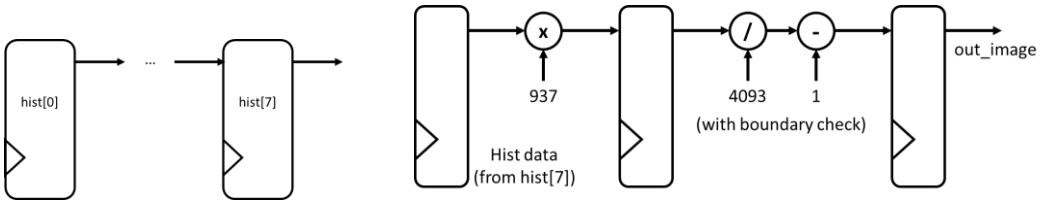
第四次作業要做的是 Histogram Equalizer。在分析完題目之後，我大概拆解出了幾個流程：對圖片做累計分佈統計，然後取出輸入的資料對應到的值，接下來用 transform function 對該值做轉換再輸出。其中 transform function 我一開始估計會拆成乘法 + 除法&減法。就在我開始煩惱要怎麼不用 for loop 做 256 個點的 Cumulative Histogram 時，我突然發現第五點寫到 pattern 會先給要修正的 pixel 值，這麼一來就輕鬆多了，只要做 8 個點的累加即可。



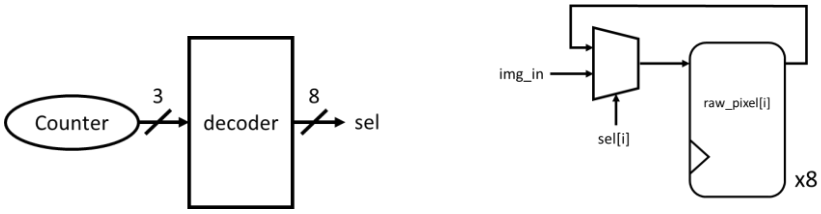
State diagram 如上圖所示，由於這次的電路要跑的是一個很簡單的流程，因此設計出來的狀態機也只要正常的繞過一圈即可。其中*done*都是由一個會持續上數的*counter*變數來決定的。



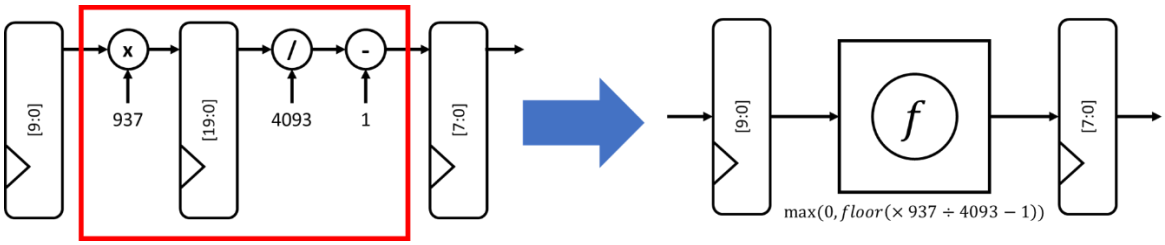
接下來就是處理各個流程的硬體。首先是儲存需要修正的 pixel 值，我使用了 8 組 8bits 的 shift register 來儲存。記錄 Cumulative Histogram 的部分則是使用了 8 組 10bits 的暫存器來兜出累加器。其中加法器輸入的部分因為我暫存器只有開 10bits，而輸入情況最高會有可能到 1024，但是 1024 跟 1023 經過 transfer function 出來的結果一樣，因此我想到可以在累加的條件順便檢查 $hist[i]$是否為 1023，如果不是的話再累加。



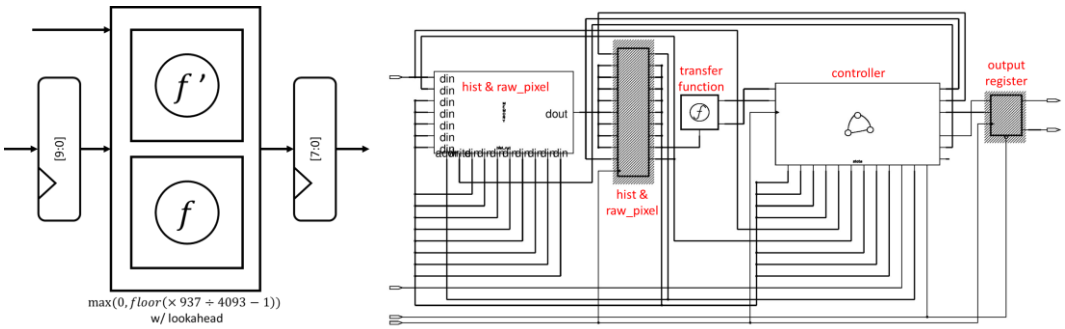
接下來是輸出的部分，我設計的是直接讓$hist[7]$變成 pipeline 的其中一個 stage，接下來運算的部分拆成兩個 stage，分別是計算乘法以及除法&減法的電路。由於減法前可能計算出來的值就已經是 0 了，所以還得另外寫判斷條件。



在我寫完後的幾天，我想到了一些可以改進的方法。在儲存需要修正的 pixel 值的時候，我改用*counter*的值來決定要存在哪一個地方，同時我也把原本是二維的 register 改成了一維，結果我在 RTL 裡面看到這些東西被 model 成了 memory，資源也省了不少，應該算是一個小的改進吧。



接下來是計算的部分，由於當初設計的時候我就有發現乘法跟除法，以及中間的暫存器很浪費資源，而且輸入跟輸出其實也只有 10bits 對 8bits 的關係，完全沒有必要存 20bits 的資料。因此我想到最簡單的方式就是直接將中間的電路換成一個 10bits 對 8bits 的 ROM，同時也省去了中間 20bits 的暫存器跟一個 stage 的 delay。另外，pipeline 輸入的地方我也改用類似前面調用*raw_pixel*的方式，用類似讀取 memory 的方式來實現。



但這樣輸出還是會有 50ns 的 delay，如果要做到 0ns 的話可能還得再將輸出往前推一級。但是這樣做的話就會導致最後一筆資料還沒進到暫存器的時候就要算出輸出的值。因此從資料進來到出去只能佔用一半的*clock cycle*。由於時間只有*clk*的一半，所以要能夠計算+1 之後再經過 transfer function 基本上是不可能的。因此我就想到使用平行二套的做法，也就是一次計算出$H(hist)$跟$H(hist+1)$的結果，再用最後一個輸入的值來判斷要輸出哪一個結果。如此一來就可以在輸入來的當下計算出結果。至此整個電路設計完畢。

經過這次的作業我發現：即便做出來的功能一樣，不同的 model 語法也可以對做出來的電路產生很大的差異。如果 coding style 太差的話，即便是使用新思的 EDA Tool 也拯救不了。