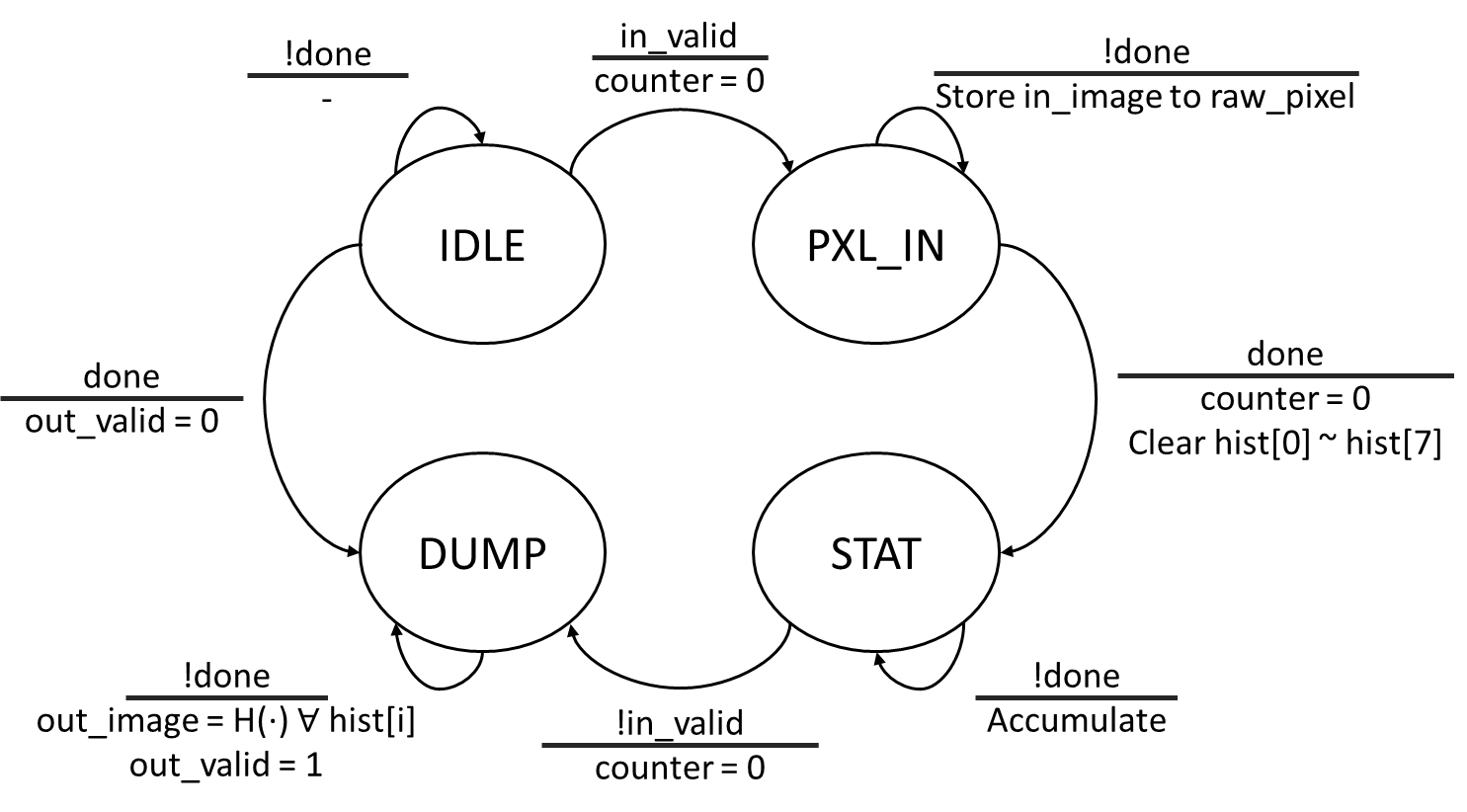
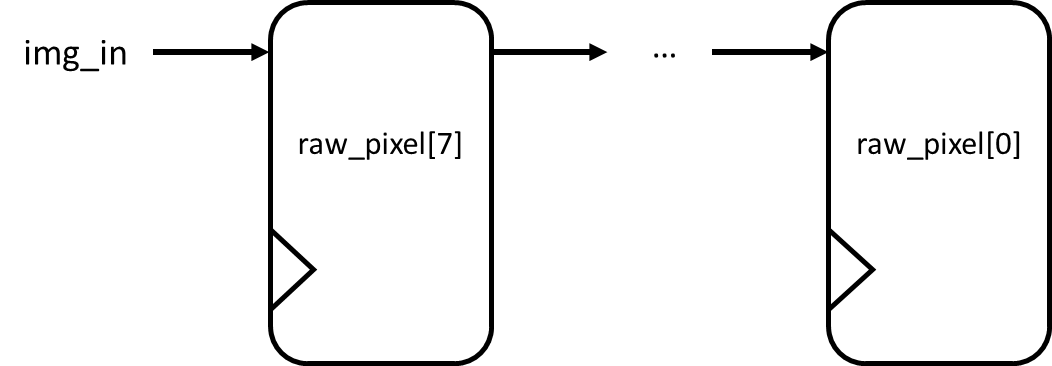
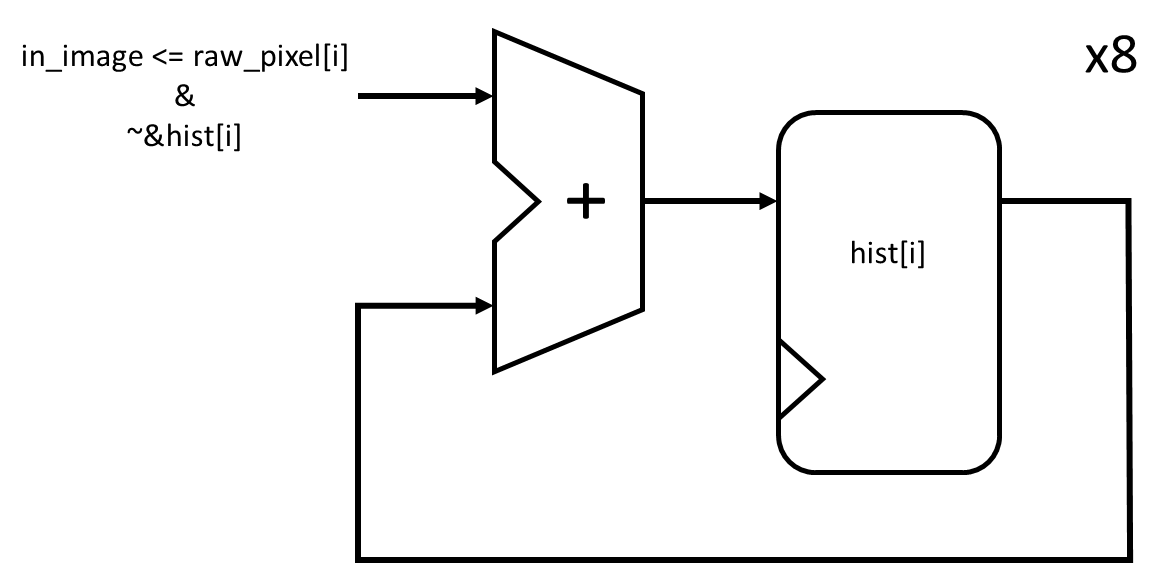
數位電路與系統 HW4 Report 110511233 李承宗

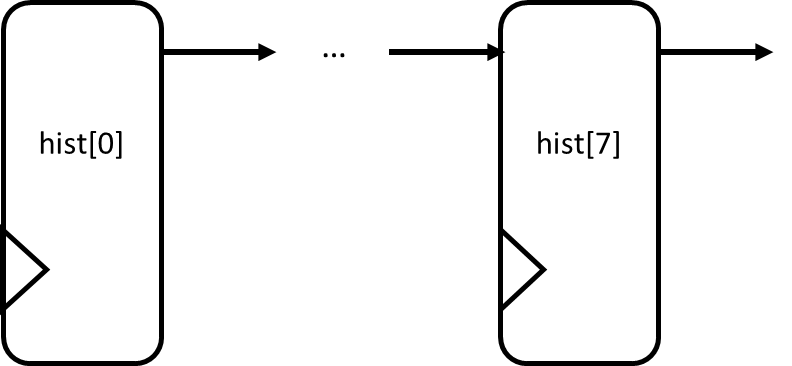
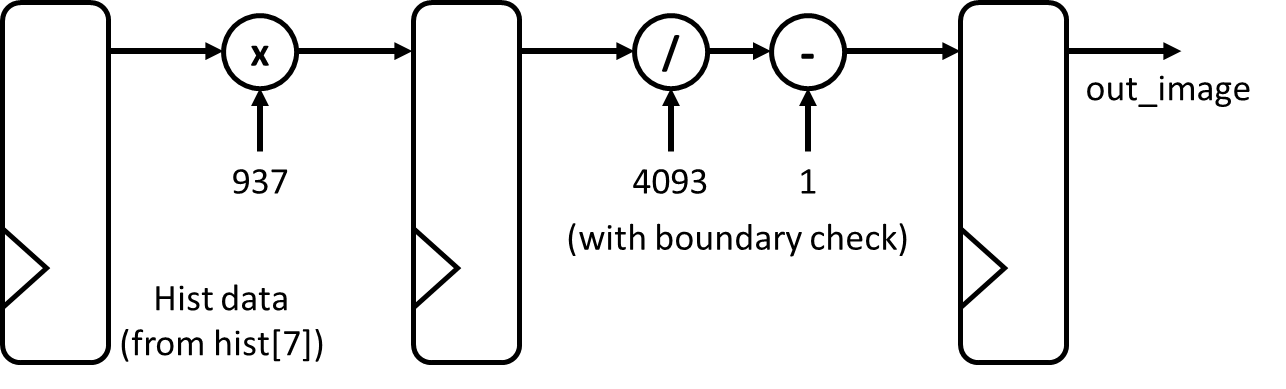
第四次作業要做的是Histogram Equalizer。在分析完題目之後，我大概拆解出了幾個流程：對圖片做累計分佈統計，然後取出輸入的資料對應到的值，接下來用transform function對該值做轉換再輸出。其中transform function我一開始估計會拆成乘法 + 除法&減法。就在我開始煩惱要怎麼不用for loop做256個點的Cumulative Histogram時，我突然發現第五點寫到pattern會先給要修正的pixel值，這麼一來就輕鬆多了，只要做8個點的累加即可。



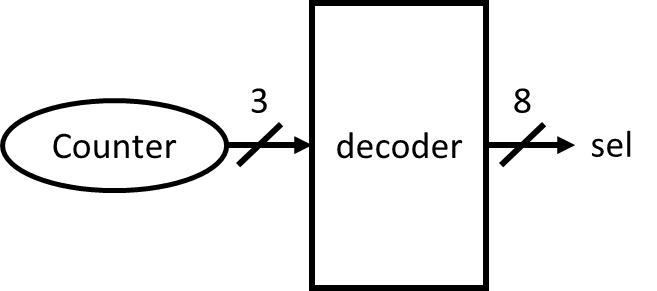
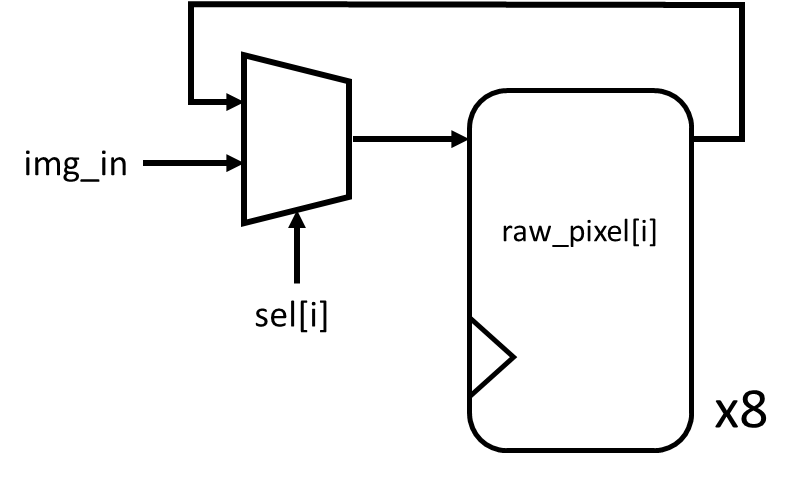
State diagram如上圖所示，由於這次的電路要跑的是一個很簡單的流程，因此設計出來的狀態機也只要正常的繞過一圈即可。其中都是由一個會持續上數的變數來決定的。

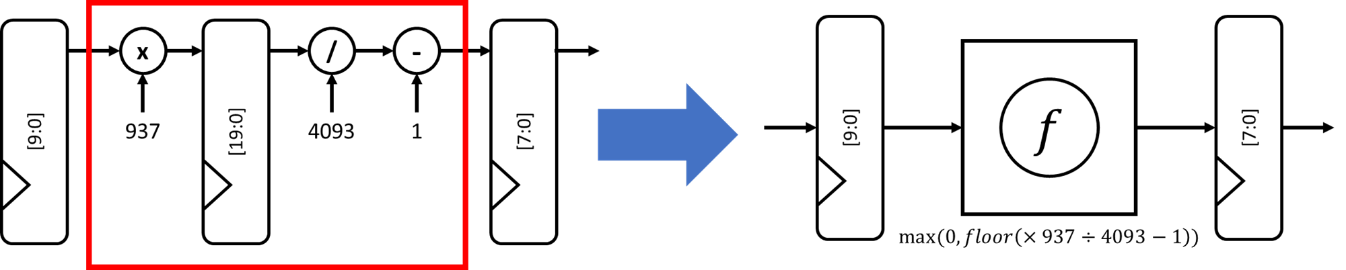
接下來就是處理各個流程的硬體。首先是儲存需要修正的pixel值，我使用了8組8bits的shift register來儲存。記錄Cumulative Histogram的部分則是使用了8組10bits的暫存器來兜出累加器。其中加法器輸入的部分因為我暫存器只有開10bits，而輸入情況最高會有可能到1024，但是1024跟1023經過transfer function出來的結果一樣，因此我想到可以在累加的條件順便檢查是否為1023，如果不是的話再累加。

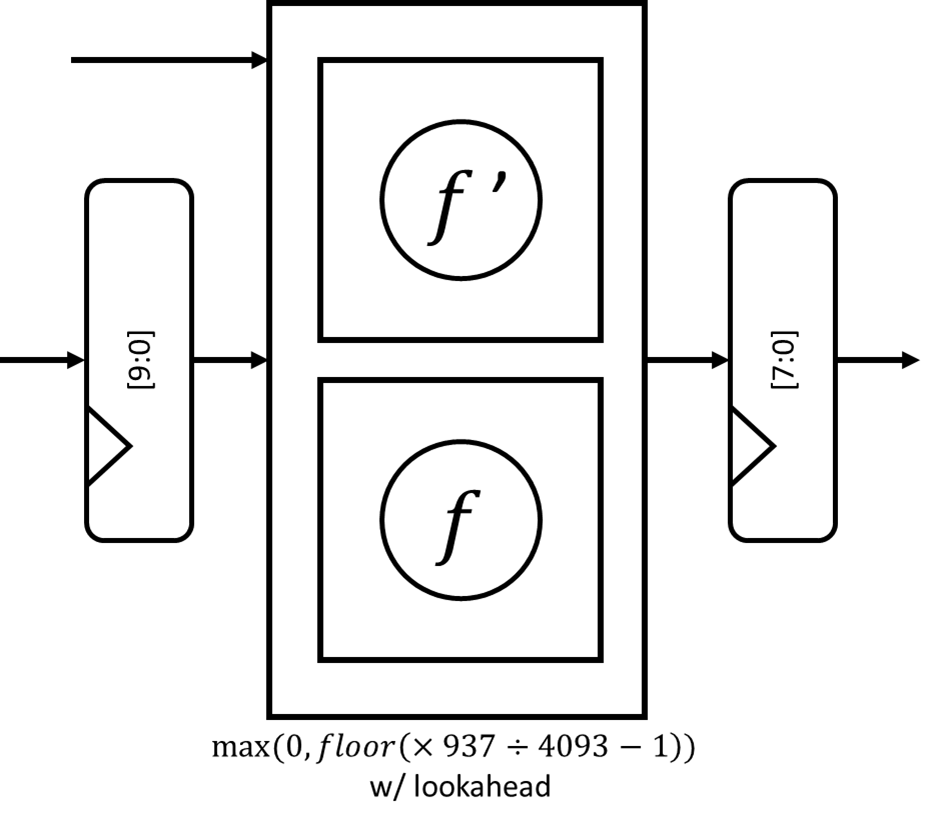
接下來是輸出的部分，我設計的是直接讓變成pipeline的其中一個stage，接下來運算的部分拆成兩個stage，分別是計算乘法以及除法&減法的電路。由於減法前可能計算出來的值就已經是0了，所以還得另外寫判斷條件。

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的方式，用類似讀取memory的方式來實現。

一張含有 圖表, 圖解 的圖片

自動產生的描述

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。由於時間只有的一半，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，也就是一次計算出跟的結果，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。