# Report_dcs193 HW04 蔡東宏 110511277

## 1. 程式說明:

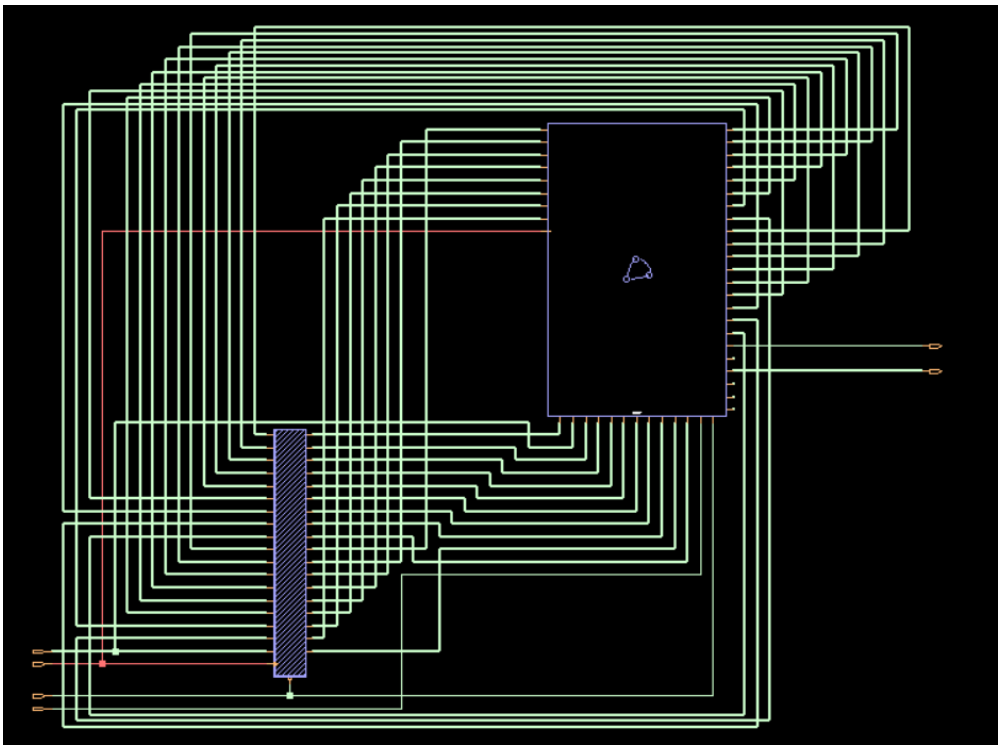
此題作業利用 pipeline 設計 Histogram Equalizer 硬體，輸入的前 8 筆資料為需要被轉換的 pixel 資料，接下來的 1024 筆為圖片資料，輸出為連續輸出 8 個 cycle 的已被轉換的 pixel 資料。

此題我用了 3 個 state 去處理此問題，第一個 state 為 IDLE，處了在這個 state 等待 in_valid 拉起來以外，我還在此 state 將前 8 個 input 資料存入 register；第二個 state 為 STORE，前 1023 筆資料分別判斷是否小於前 8 筆 input data，若有的話將變數+1，沒有的話則維持原狀，在第 1024 筆資料時，只判斷有沒有小於第一筆 input data，有的話+1 後*937，沒有的話直接*937；最後一個 state 為 OUT，在第一個 cycle，判斷第 1024 筆資料有沒有小於 2~8 筆(因為此時 in_image 只會拉起半個 cycle，所以我們用 register 將它存起來)，且第一個 cycle 因為已經*937 了，所以直接除 4093，即可得到 output，並同時將下一個 cycle 要輸出的資料*937，連續輸出 8 個 cycle 後回到 IDLE，在輸出的時候還需考慮邊界條件，當*937/4093 後若是 0，則不用-1，反之則要減 1。

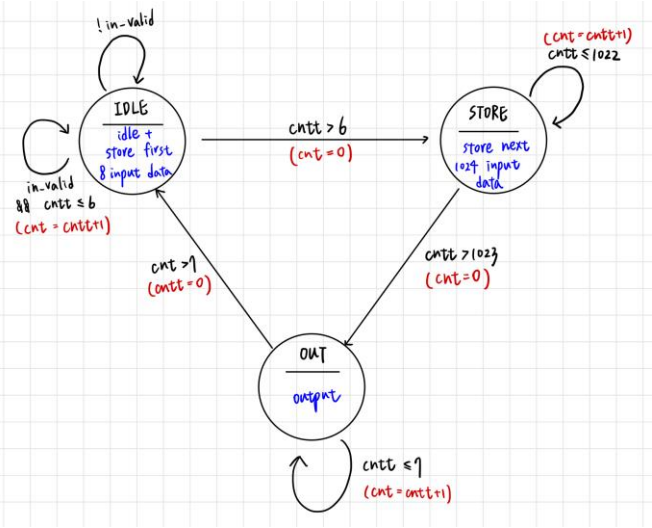
```
out_image_temp=out8[19:12]+(((3*out8[19:12]+out8[11:0])>4093)?1:0);
```

(除法利用一點變換讓除法能夠更快)

## 2. 架構圖:



FSM:




## 3. 優化過程:

1. **將 latency 變成 0:**一開始我是先讓 1024 筆資料全部判斷完再*937 並進入 OUTPUT 的 state，用此方法的 latency 會等於 50，因此我讓*937 這個步驟放在第 1024 筆輸入時，所以那個 cycle 會判斷需不需要+1 以外還要*937，讓 output 能在 invalid 拉下時馬上拉起，如此 latency 便能減少為 0。
2. **減少 design area:**在輸出時，我一開始在剛進 OUT state 的時候，除了第一筆資料，我其他筆都*937，如此一來，存那些資料的變數就需要 20bit，所以就會增加 DFF 的面積，且同時需要多個乘法器。因此，我將 design 改成在輸出的前一個 cycle 才*937，如此能夠減少 DFF 的面積，也只需要一個乘法器。
3. **Pipeline:**因為在同一個 cycle 做乘法和除法，critical path 會太長，導致 02 的 timing 會 violation，因此我將乘法和除法分別在 2 個 cycle 做，所以我在要進 OUT state 之前，我就將第一筆要輸出的資料*937，一進 OUT 時，就將它除 4093 並將第二筆要輸出的資料*937。

## 4. 結論:

這次的作業利用 pipeline 設計 Histogram Equalizer 硬體，用了 pipeline 去處理 *937/4093 的問題，因為要盡可能讓 latency 變小，所以要盡量讓 output 在 invalid 剛往下拉的時候就拉起來，除此之外，我利用減少 DFF 的 bit 數以及乘法器的數量減少面積。

## 5. 小建議:

關於 performance 的計算，我覺得在比 1de 排名的時候，須將 2de 的人排在下面(目前計算方式好像沒有)，因為若只有 3 個人 1de 交出來，那麼第三名只有 33 分且他們之間的差距會非常大，感覺有點不太公平。