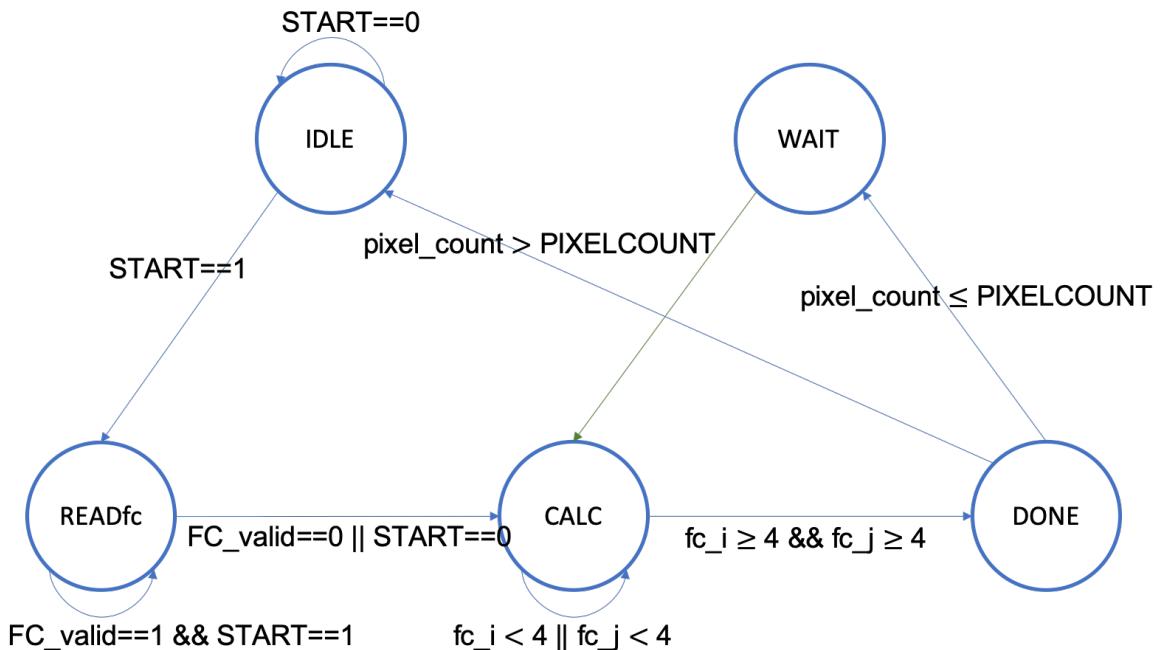


CS5120 Homework Assignment 03

Student ID: 107062612 Name: 熊祖玲

1. Design Concept

A. Finite state machine (如圖一所示)



圖一、Finite State Machine圖

當RST_N訊號由1變0時，將所有訊號做初始化，並且進入IDLE狀態。

I. IDLE狀態

- START為0：停留在IDLE階段，並將所有訊號設為初始值。
- START為1：進入READfc狀態。

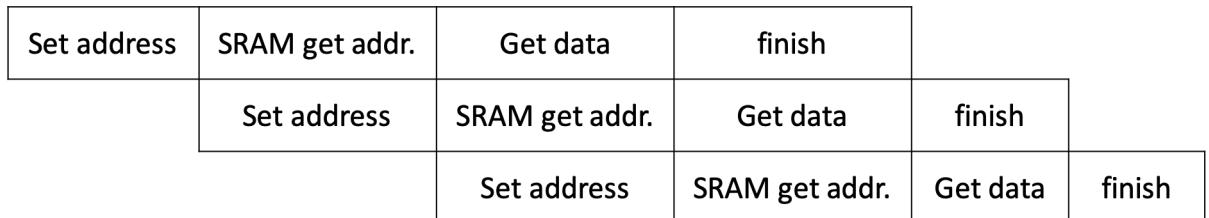
II. READfc狀態

- START和FC_valid皆為1：讀入fc將其存到FC的二維陣列中作為mask，並且更新fc_i, fc_j使得fc存到FC中正確的位置。
- 其他：進入CALC狀態。

III. CALC狀態

透過pipeline的技巧計算pixel，計算一個pixel需要四個步驟或cycles（如圖二所示）：計算pixel位於sram的位址，將此位址送進sram中，取得此位址的資料並進行計算，完成計算並累加outPixel的值。

- fc_i, fc_j皆大於4：完成當前pixel計算，進入DONE狀態。
- calc_count < 2：從pipeline圖得知，前兩個cycles尚未取得資料，無法進行運算，因此用calc_count作為開始計算pixel的依據，留在CALC階段。
- 其它：取得資料進行運算及累加，並停留在CALC階段繼續計算下個pixel和FC[fc_i][fc_j]的乘積。



圖二、Pipeline示意圖

IV. DONE狀態

- i. $\text{pixel_count} > \text{PIXELCOUNT}(65535)$ ：完成所有的計算，進入IDLE階段等待下一張圖的計算。
- ii. 其他：取得outPixel值，將小於0的值設為0，大於255的值設為255，並且更新下一個pixel的座標，進入WAIT狀態。

V. WAIT狀態

將out_valid訊號設為1，使當前計算好的outPixel值輸出，並進入CALC階段，計算下一個outPixel值。

B. 詳細之計算方法：

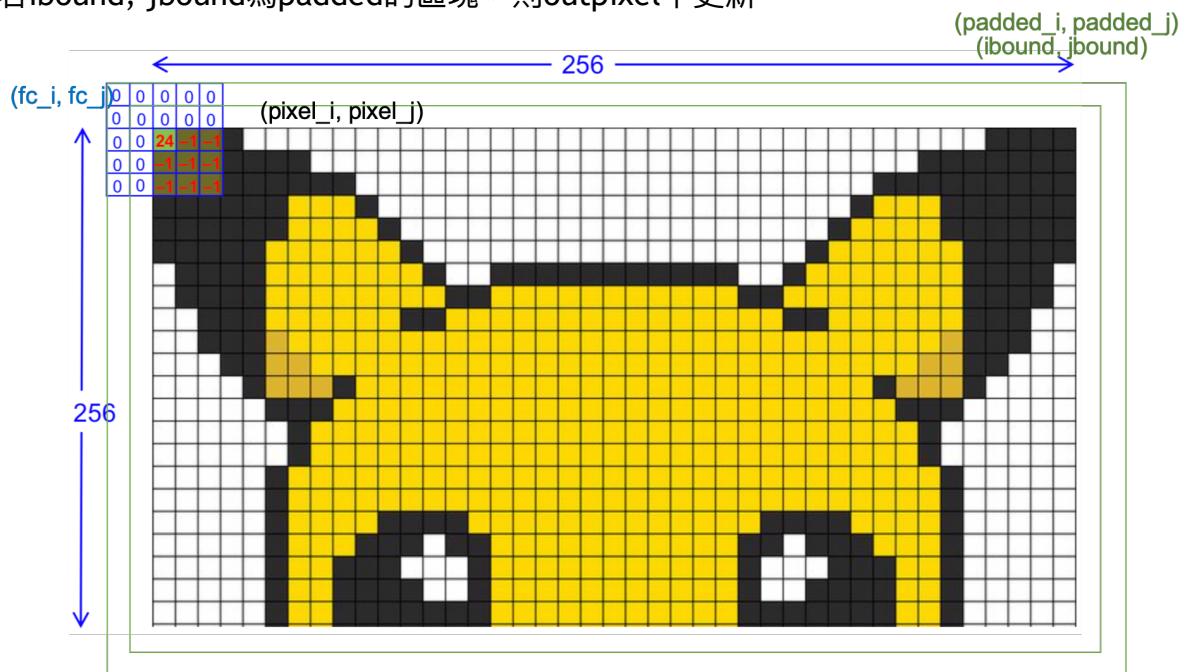
如圖三所示，計算時需要建立4組座標：

- I. filter coefficients (fc_i, fc_j) 紀錄計算的filter為何。
- II. 原圖 ($pixel_i, pixel_j$) 記錄目前計算的pixel為何。
- III. Padded後的圖 ($padded_i, padded_j$) 用於取得sram address (比計算早兩個cycle)
- IV. Padded後的圖 ($ibound, jbound$) 用於記錄兩個cycle前的 $padded_i, padded_j$ ，檢查目前計算的pixel是否為padded後的座標。

計算公式: $\text{outpixel} += FC[fc_i][fc_j] \times \text{sram}[ibound \times 256 + jbound]$

然而， $\text{sram}[ibound \times 256 + jbound]$ 為兩個cycles前藉由 $\text{sram}[padded_i \times 256 + padded_j]$ 取得的

若 $ibound, jbound$ 為padded的區塊，則 outpixel 不更新

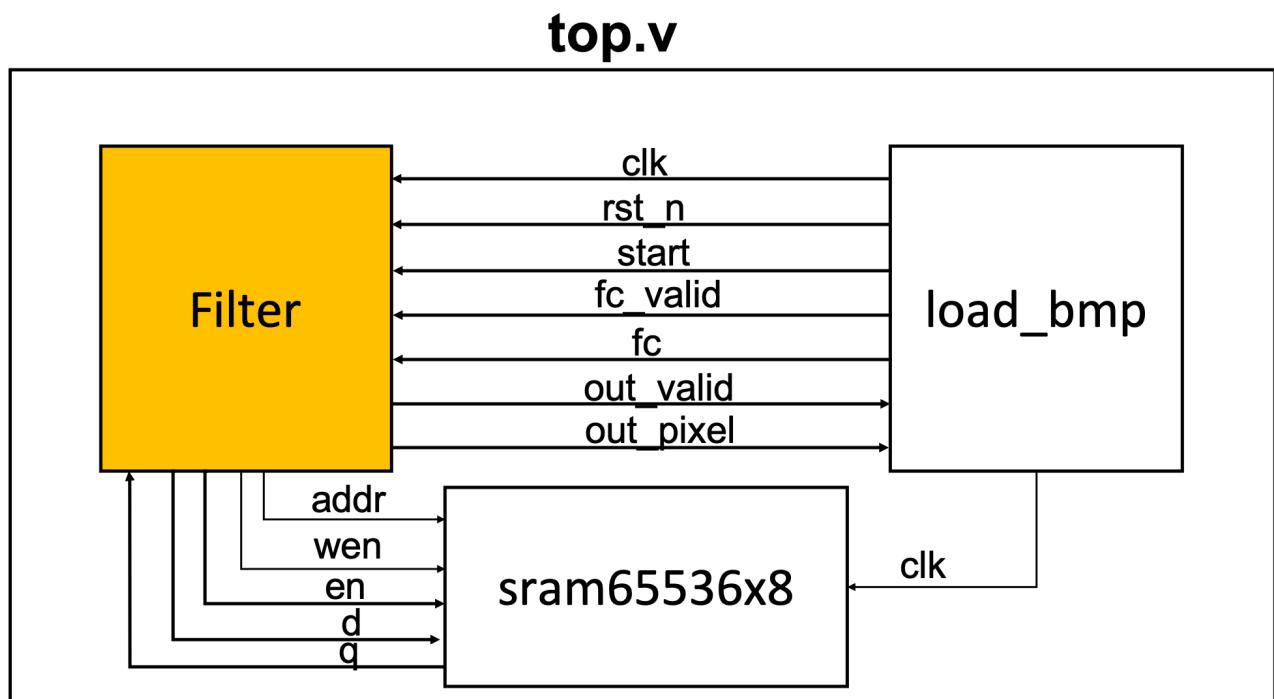


圖三、座標示意圖（擷取自本次作業資訊中的hw03-2.pdf）

C. Block diagram

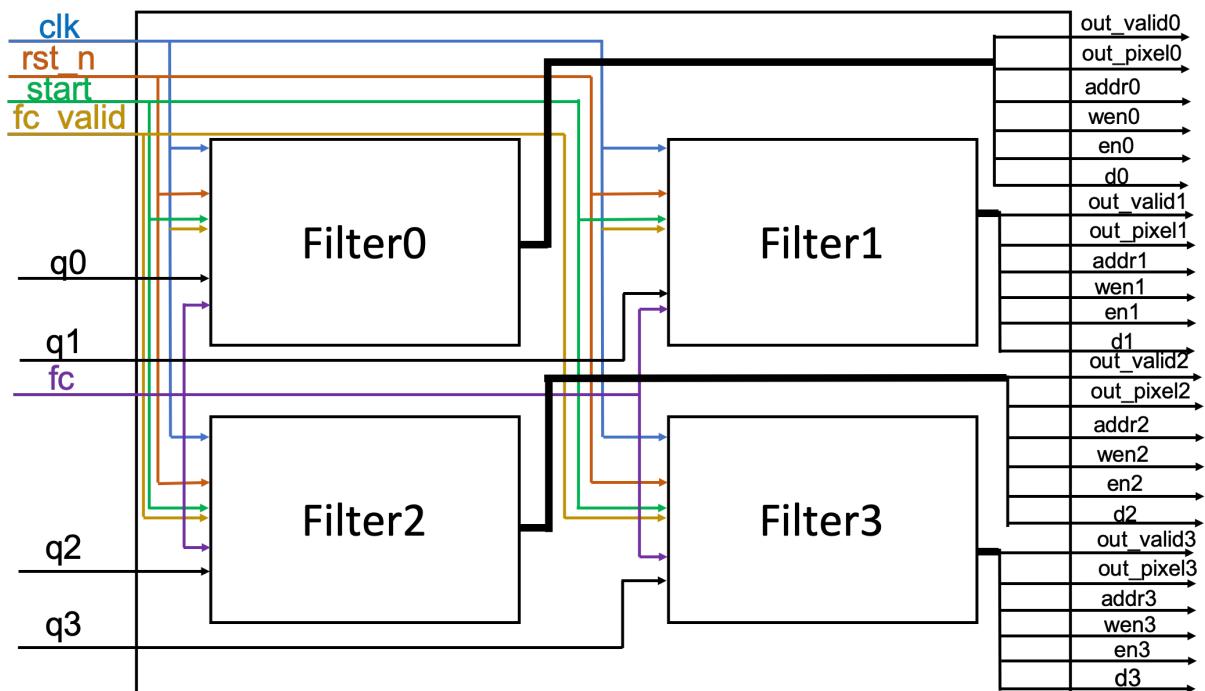
I. hw03a

a小題中的做法如前面AB所述，圖四為其block diagram。



圖四、hw03a block diagram

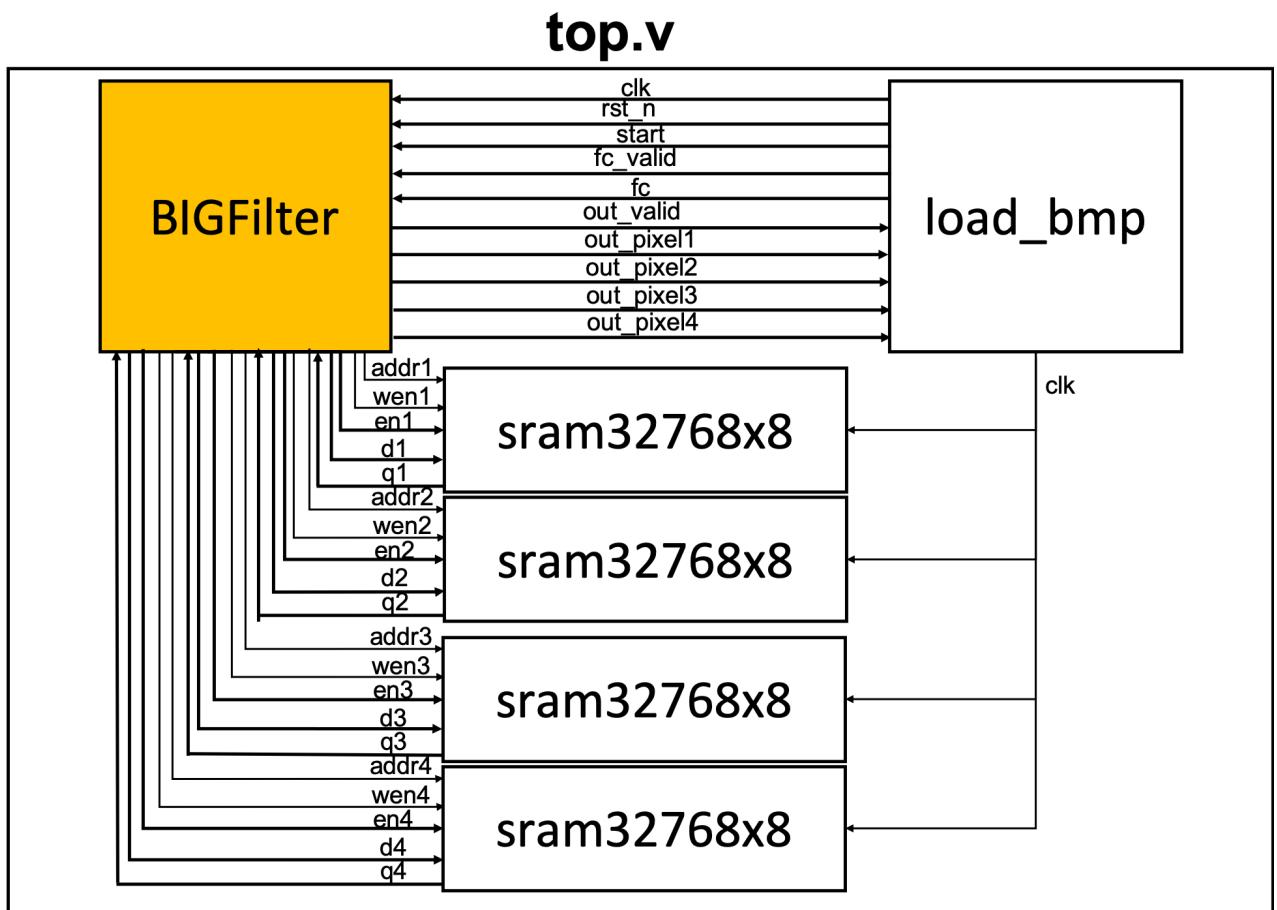
II. hw03b

BIGFilter.v

圖五、hw03b block diagram—module **BIGFilter**

b小題中，我使用4個32768大小的sram分別存1/4個圖的pixel，並且建立4個filters讓每個filter接到一個sram，藉此加快速度；除此之外，為了增加程式碼的易讀性，將4個filters包成一個BIGFilter (如圖五所示)，其餘作法與a小題相似。

若每個sram只存1/4張圖的pixel，在計算時會出現問題，每個filter在計算最前面兩個和最後兩個row的pixel時，會無法取得pixel，所以每個sram都必須多存取前後各兩個rows的pixel，並且將剩餘空間的部分補零。也因為每個sram都多存取前後各兩個rows的pixel，在計算時就可以不用再檢查row的boundary。

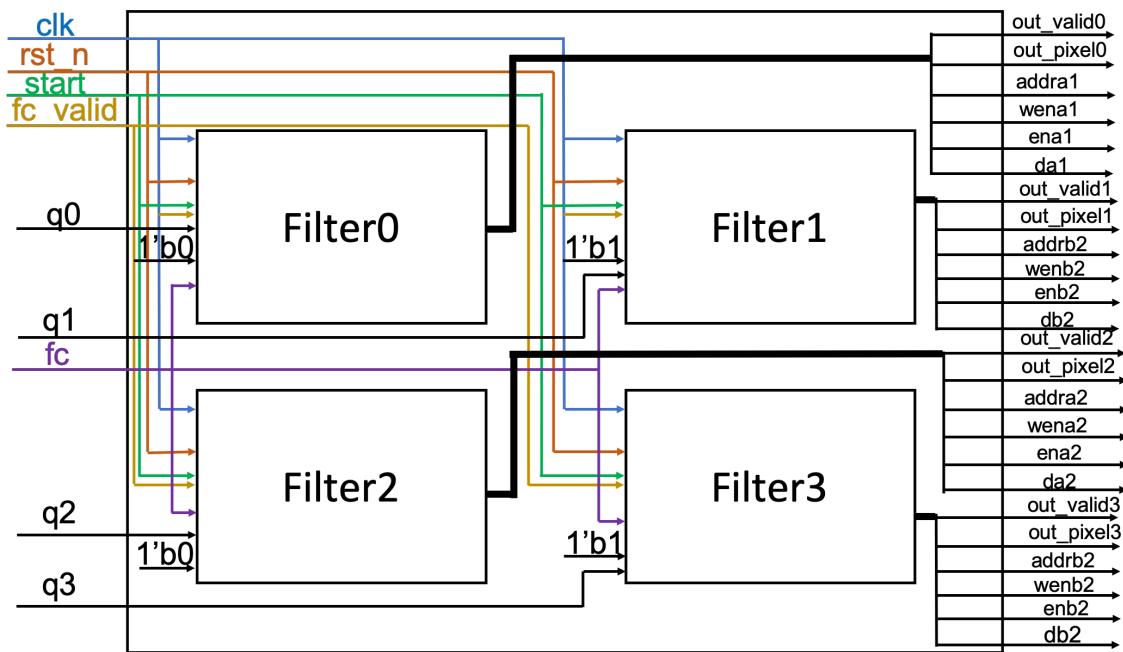
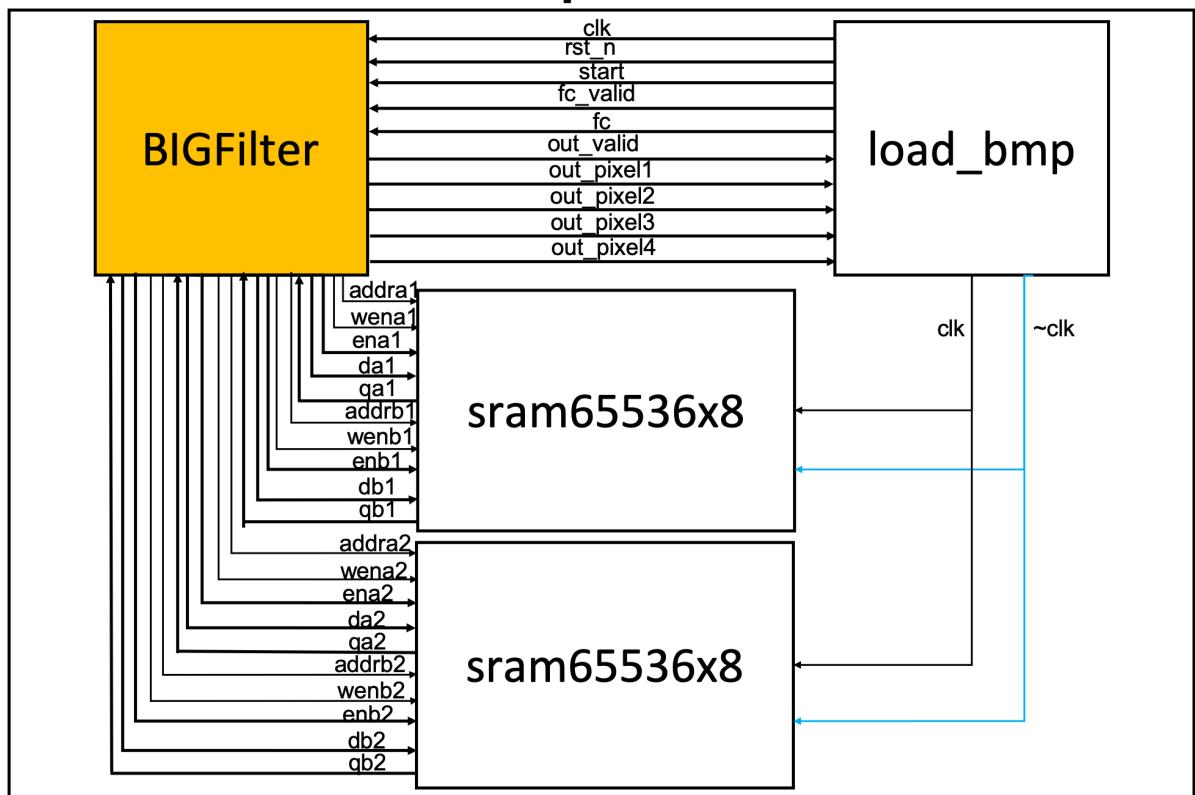


圖六、hw03b block diagram—module top

III. Hw03c

c小題中，我使用2個65536大小的dual port sram分別存1/2個圖的pixel，並且建立4個filters讓2個filters接到一個dual port sram；除此之外，為了增加程式碼的易讀性，將4個filters包成一個BIGFilter (如圖七所示)，並且在每個filter都增加一個輸入訊號，藉此作為這些filters讀取sram時的offset，其餘作法與b小題相似。

欲使用dual port sram需要處理clock的問題，因為dual port sram是用兩種clock控制資料輸出，在這裡我使用與clka相差180度的訊號作為clkb，然而這會使qb的資料提早一個clk取得，因此需要增加一個register儲存qb的資料，也就是有offset的filter必須使用register的資料進行計算。

BIGFilter.v圖七、hw03c block diagram—module **BIGFilter****top.v**

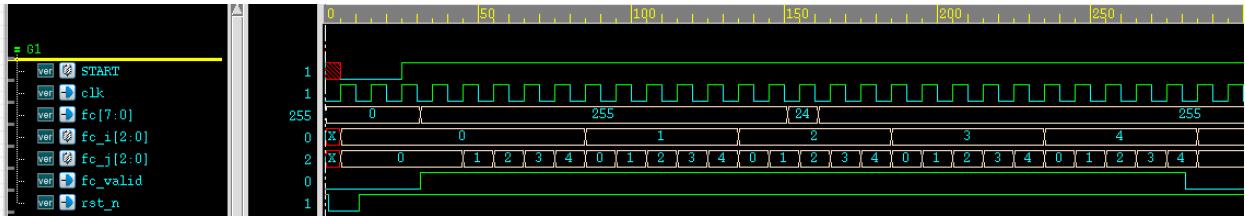
圖八、hw03c block diagram

2. Simulation and Discussion

A. Simulation

I. 讀取filter coefficient

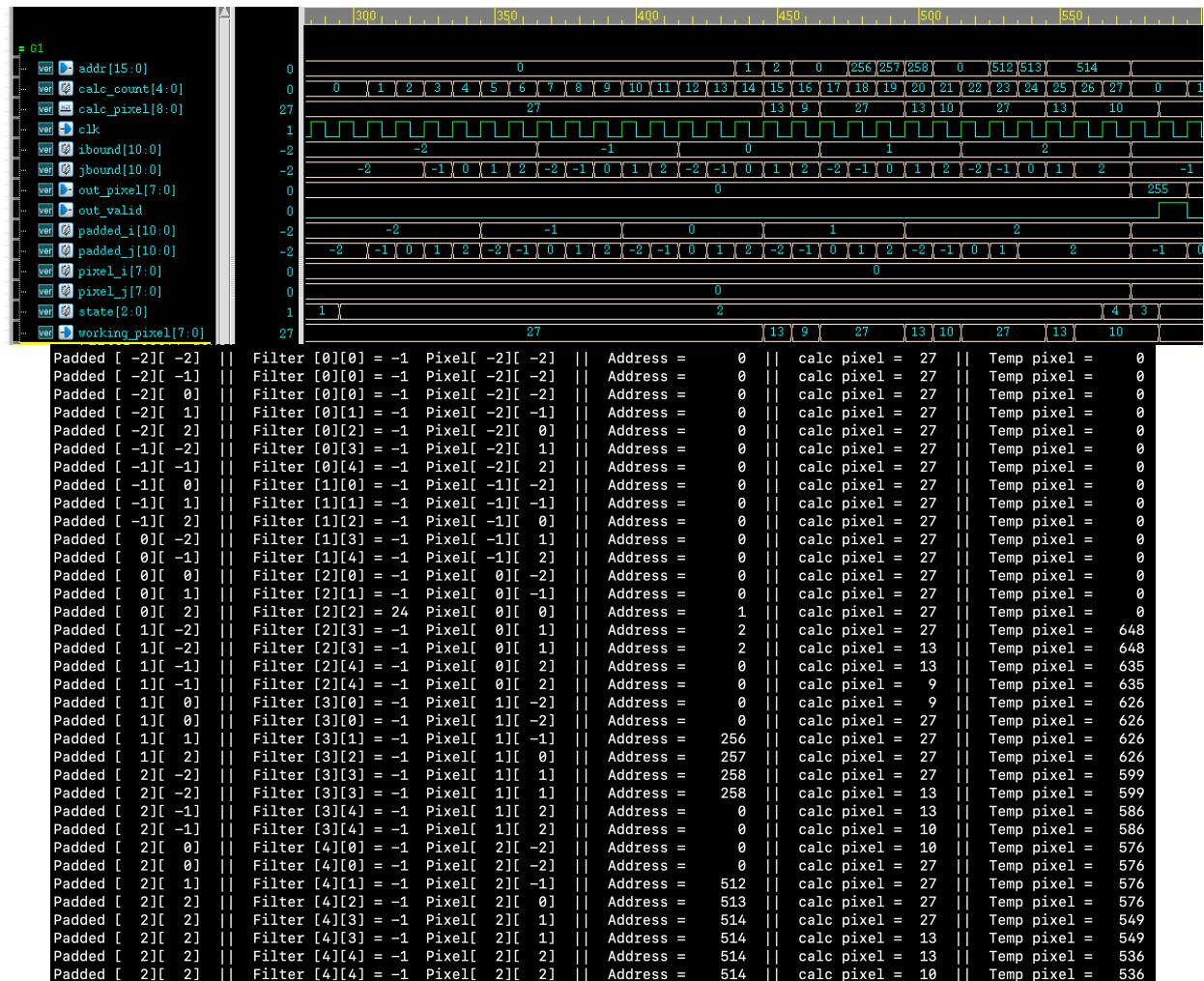
當START和fc_valid皆為1時，將filter coefficient讀到二維陣列FC中，從波形圖（如屠九所示）可以得知訊號無不正常的現象發生。



圖九、讀取fc之波形圖

II. 計算第一個pixel (single sram)

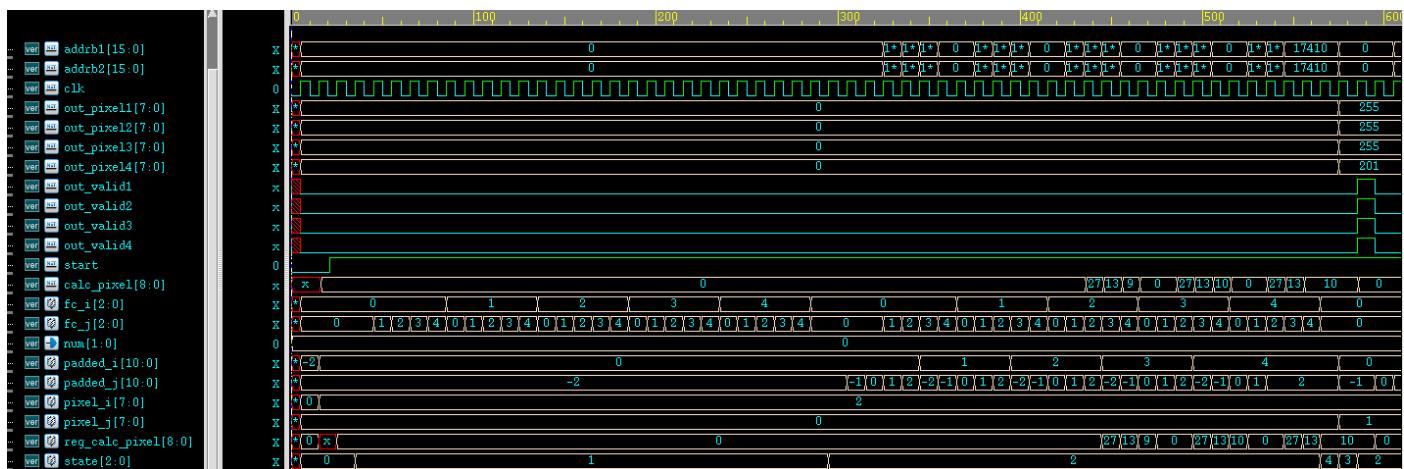
前兩個cycle不做convolution計算，只計算address並取得sram的資料，第三個cycle開始用前兩個cycle的address取得的sram資料進行計算，並且不計算超出pixel的座標的pixel值，計算完畢後進入DONE狀態取得下一個pixel座標，在進入WAIT狀態等outpixel穩定之後再將out_valid設為1，將outpixel輸出。



圖十、 計算第一個pixel之示意圖 (single sram)
(上圖為波形圖、下圖為log值截圖)

III. 計算第一個pixel (dual port sram)

前兩個cycle不做convolution計算，只計算address並取得sram的資料，第三個cycle開始用前兩個cycle的address取得的sram資料進行計算，並且不計算超出pixel的座標的pixel值，計算完畢後進入DONE狀態取得下一個pixel座標，在進入WAIT狀態等outpixel穩定之後再將out_valid設為1，將outpixel輸出。filter2, filter4在計算sram位址時都有加上offset，取得dual port sram的後半部的資料，又因為抓取後半段位址時是使用第二個clock觸發，而取得的資料會提早半個clock cycle得到，存在暫存器中以便在下個clock cycle使用。



圖十一、計算第一個pixel (dual port sram) 之波形圖

B. Speedup

I. Compare between hw03a and hw03b

i. Multiplier

我所使用之sram的位址都是 2^n ，因此在計算位址的時候可以透過shift n bit(s)達到乘法的效果，所以增加乘法器對我的design來說沒有可以加速的地方，或者是有其它部分可以加速，但我可能沒有想到。

ii. improve engine

b小題中，我使用4個32768大小的sram分別存1/4個圖的pixel，並且建立4個filters讓每個filter接到一個sram，藉此加快速度。如表一所示，在使用4個filters和sram且其它條件完全相同的情況下，可以將hw03a的時間加速將近四倍，跟預期中的效果一致，之所以沒有加速到4倍是因為讀寫檔的時間是無法被加速的。

II. Compare between hw03a and hw03c

i. Multiplier

我所使用之sram的位址都是 2^n ，因此在計算位址的時候可以透過shift n bit(s)達到乘法的效果，所以增加乘法器對我的design來說沒有可以加速的地方，或者是有其它部分可以加速，但我可能沒有想到。

ii. improve engine

c小題中，我使用2個65536大小的dual port sram分別存1/2個圖的pixel，並且建立4個filters讓2個filters接到一個dual port sram。如表一所示，在使用4個filters和2個dual port sram且其它條件完全相同的情況下，可以將hw03a的時間加速將近四倍，跟預期中的效果一致，之所以沒有加速到4倍是因為讀寫檔的時間是無法被加速的。

	hw03a	hw03b	hw03c
Simulation Time	19006735 NS	4752655 NS	4752655 NS
Post-Synthesis Time	19006735303 PS	4752655376 PS	4752655376 PS
Speedup	1	3.9992	3.9992

表一、hw03a、hw03b和hw03c的時間比較表

III. Dual port sram & sram which is easier

在實作的過程中，我認為使用sram比較容易一些，在sram中不需要處理兩種clock的問題，在實作上也比較直覺一些；另外，透過表一及表二的時間及面積比較可以知道使用dual port sram和一般的sram的執行時間一致，但是使用dual port sram所需的面積比較大，因此，我認為使用一般的sram也比較佔優勢。

C. Area

如表二所示，在b、c小題中所使用的DFF數量分別為a小題的3、3.13倍，因為在b、c小題中都使用了4個filters，另外在c小題為了暫存由反相clock觸發所得的資料所以DFF的數量比b小提的多一些。而為了要加快計算的時間，因此使用了更多的filter和sram，讓資料可以平行的處理，因此在面積上有很大的犧牲，但我所設計之b、c小題的執行時間是相同的，所以對於b、c小題面積來說，b小題的結果會比較好一點。

	hw03a	hw03b	hw03c
Combination area	11272.433504	46780.344338	57032.640553
Buf/Inv area	478.666793	3354.062348	3584.908745
Noncombination area	11798.627722	35801.561165	36894.68760
Total area	23071.061226	82581.905503	93927.327312
Number of DFF	366	1110	1144
Number of SRAM	1個65536x8 sram	4個32768x8 sram	2個65536x8 dualport sram
SRAM area	2777600	5555200	12265600

表二、hw03a、hw03b和hw03c的面積比較表

D. Problem

在模擬合成之後的design時，出現了Glitch suppression的警告（如圖十一所示），但目前仍然不知道為什麼會出現這個問題，不過可以知道的是這個警告不會影響結果。

```
Warning! Glitch suppression
Scheduled event for delayed signal of net "D" at time 7019409251 PS was canceled!
  File: /theda21_2/CBDK_IC_Contest/cur/Verilog/tsmc13.v, line = 19639
  Scope: top.filt.\temp_pixel_reg[2]
  Time: 7019409035 PS
```

圖十二、Glitch suppression警告截圖

3. Summary

這次的作業有點複雜，最需要花時間想的部分是如何與sram溝通，以及pipeline的部分，在寫作業的過程中，只要座標沒有想清楚，就會花很多時間在找問題，果然寫作業的時候不可以操之過急，有縝密的規劃再開始做才會比較順利。另外，這次作業實在要感謝實驗室的同學的幫忙與討論，我才可以如期地完成，多虧同學在我遇到問題時幫我想可能問題可能出現在哪裡，不然我一個人可能會花更多的時間debug。