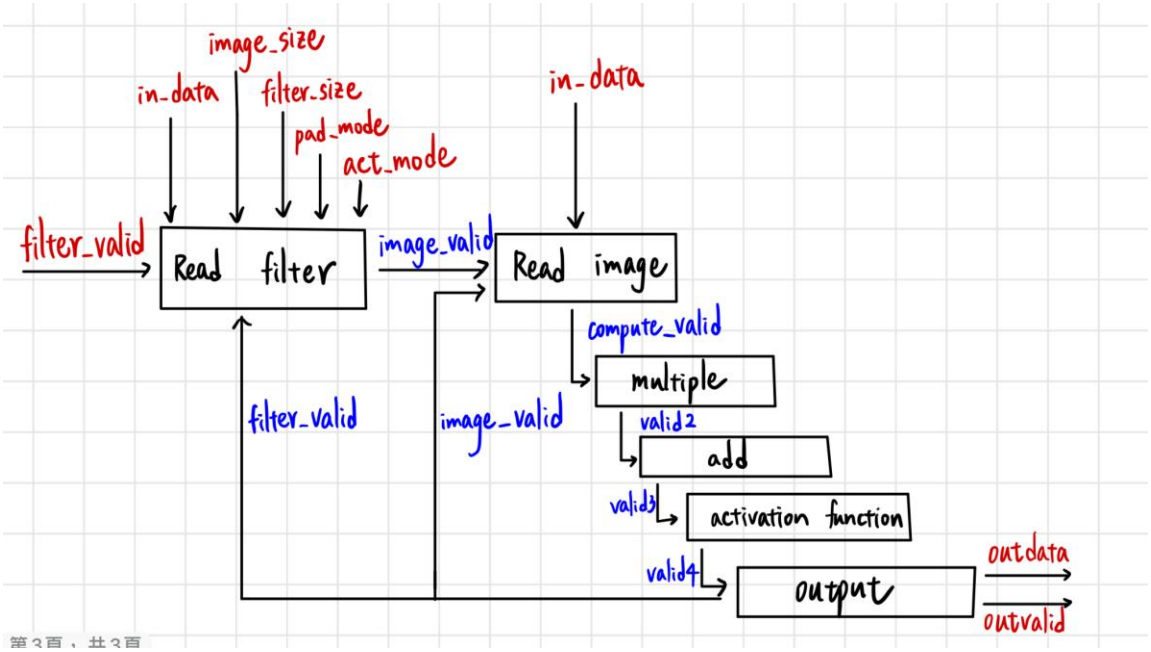
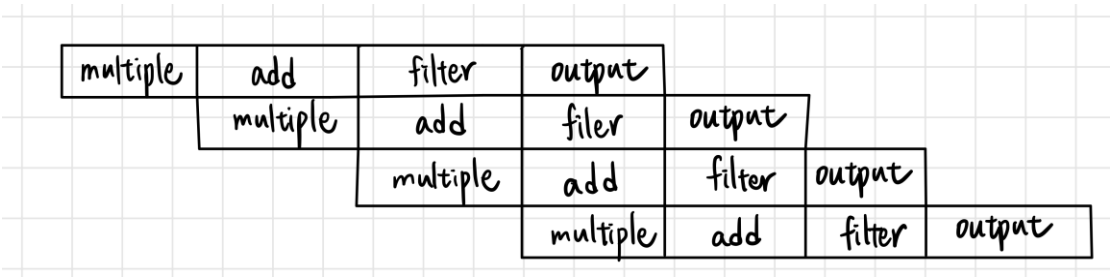
Report_dcs193 Final project 蔡東宏 110511277

1. 程式說明:

這次的 final project 主要是做 convolution，一開始先等 filter_valid 拉起來，當 filter_valid 拉起來時，讀取 filter_size, padding_mode, acting_mode, image_mode 以及 filter 的 data，利用 cntx, cnty 去數現在存到第幾個位置，接下來等 image_valid 拉起來時，讀取 image 的值，在讀取的同時判斷那個位置需不需要 padding，要的話就將該位置進行 padding(因為每次皆有歸 0，所以不需要特別執行 zero padding)，當讀了一定數量的值(可以計算 convolution)時，便把 compute_vailid 拉起來，當 compute_valid 拉起來時，進行 convolution 的乘法，同時將 compute_valid 傳給 valid2，如此一來，下個 cycle 便可以進行加法，接著將加完的結果傳給下一個 cycle 進行 activation function 的處理，最後傳給下一個 cycle 判斷有沒有大於 32767 或有沒有小於-32768，同時將最後的結果輸出。

2. 架構圖:





(pipeline 示意圖)

3. 優化過程:

A:共用乘法器和加法器:一開始在做 convolution(包括乘法和加法的時候)，我將 filter 是 3*3 和 5*5 的 case 分開討論，但後來發現這樣需要用到 9+25 個乘法器以及加法器，後來我試著將兩個 case 變成同一個 case，我先將 filter 全部歸 0，如此一來，便可將 3*3 的 case 變得跟 5*5 一樣，省下 9 個乘法器以及 9 個加法器的面積。

B:減少存 image 的 DFF:一開始我使用 12*12 個 8bit 的 DFF 去存 image 以及 padding 後的值(因為最大的情況是 image_size 為 8*8 且 filter_size 是 5*5)，但後來我發現只需要 7*12 個 8bit 的 DFF，因為在算後面的時候，前面有些的 image 值已經用不到了，所以我可以當某一排的值已經用不到時，全部往上平移一格，如此一直重複，最多只需要用到 7*12 的 DFF 去存 image 的值，如此可以大幅減少 DFF 的數量以及在做乘法的時候，二維陣列裡面的 column 不需要有變數去控制它，面積減少了許多。

```
store_comb[0][0]=image_reg[0][cnt_xx_reg]*filter_reg[0][0];
store_comb[0][1]=image_reg[0][1+cnt_xx_reg]*filter_reg[0][1];
store_comb[0][2]=image_reg[0][2+cnt_xx_reg]*filter_reg[0][2];
store_comb[0][3]=image_reg[0][3+cnt_xx_reg]*filter_reg[0][3];
store_comb[0][4]=image_reg[0][4+cnt_xx_reg]*filter_reg[0][4];
store_comb[1][0]=image_reg[1][cnt_xx_reg]*filter_reg[1][0];
store_comb[1][1]=image_reg[1][1+cnt_xx_reg]*filter_reg[1][1];
store_comb[1][2]=image_reg[1][2+cnt_xx_reg]*filter_reg[1][2];
store_comb[1][3]=image_reg[1][3+cnt_xx_reg]*filter_reg[1][3];
store_comb[1][4]=image_reg[1][4+cnt_xx_reg]*filter_reg[1][4];
store_comb[2][0]=image_reg[2][cnt_xx_reg]*filter_reg[2][0];
store_comb[2][1]=image_reg[2][1+cnt_xx_reg]*filter_reg[2][1];
store_comb[2][2]=image_reg[2][2+cnt_xx_reg]*filter_reg[2][2];
store_comb[2][3]=image_reg[2][3+cnt_xx_reg]*filter_reg[2][3];
store_comb[2][4]=image_reg[2][4+cnt_xx_reg]*filter_reg[2][4];
store_comb[3][0]=image_reg[3][cnt_xx_reg]*filter_reg[3][0];
store_comb[3][1]=image_reg[3][1+cnt_xx_reg]*filter_reg[3][1];
store_comb[3][2]=image_reg[3][2+cnt_xx_reg]*filter_reg[3][2];
store_comb[3][3]=image_reg[3][3+cnt_xx_reg]*filter_reg[3][3];
store_comb[3][4]=image_reg[3][4+cnt_xx_reg]*filter_reg[3][4];
store_comb[4][0]=image_reg[4][cnt_xx_reg]*filter_reg[4][0];
store_comb[4][1]=image_reg[4][1+cnt_xx_reg]*filter_reg[4][1];
store_comb[4][2]=image_reg[4][2+cnt_xx_reg]*filter_reg[4][2];
store_comb[4][3]=image_reg[4][3+cnt_xx_reg]*filter_reg[4][3];
store_comb[4][4]=image_reg[4][4+cnt_xx_reg]*filter_reg[4][4];
```

C:切 pipeline:我這次 final project 切 pipeline 的方式是我讓乘法一個 cycle，加法一個 cycle，過 activation 一個 cycle，最後判斷有沒有超出邊界並輸出。如此切 pipeline 可以壓 cycle time，讓一個 cycle 做的事情變少。

4. 結論:

這次的 final project 主要為 convolution，我嘗試了很多方法壓面積以及 latency 以及 cycle time，透過共用硬體來減少面積，重複利用 DFF 來儲存 image 來減少 DFF 的數量，以及透過切 pipeline 來降低 cycle time。