

軟硬 Lab6 Report

張或豪, 0416005

Abstract—利用軟硬協同設計來加速 findface 這個程式的執行速度。

Keywords—臉部辨識、軟硬協同

I. INTRODUCTION

Find face 是一個利用”template matching”來做臉部辨識的 C 程式，辨識的方法是找出小圖 face 與大圖 group 中哪個區域的 sum of absolute difference 最小。在 lab1 已經知道這個程式要花最久時間的地方是比對部分，這次作業目的就是要利用硬體加上軟體的共同合作使這個 find face 程式的執行速度加快。

II. RELATED SEARCH

這次作業主要做的是臉部辨識，準確的來說為”template matching”，希望可以利用硬體來加速”template matching”的速度。

III. DEVELOPING THE PROGRAM

A. 設計理念

這次作業的架構設計我是採用全硬體的架構，這個架構是我跟王柏堯(0416305)同學討論出來的，會這樣設計的原因在於在前五次的 lab 中，如果要讓 find face 程式有最大程度的加速，使用 lab3 加上 lab4 所學到的東西應該可以讓 find face 有最大程度的加速，lab3 先計算一個 column 後再右移可以減少資料傳輸的次數，修正 lab2 的缺點，但使用軟體進行資料傳輸還是較硬體慢，所以我們又將 lab4 所學到的 DMA 技術加進我們的設計中，期望這樣可以有最大程度的加速。

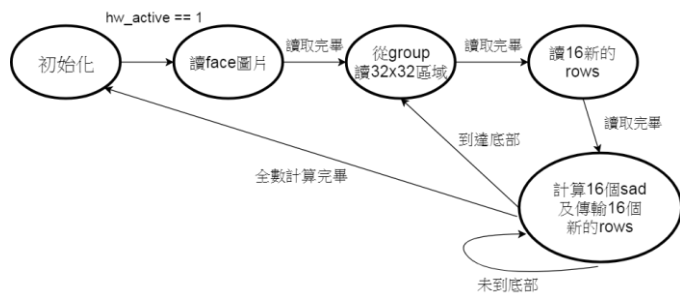


圖 1. 此次 lab 狀態有限機

B. 程式架構—硬體部分

在我們的設計中，硬體必須實作出像 lab3 那樣的流程控制，由於資源消耗的問題，我將 Lab3 中的 reg_bank 換成 shift registers 設計，這樣可以讓硬體用 shift 的方式來準備好下一次要計算的 group 區域，這樣計算絕對值也不用還要看 reg_bank 的值是多少後再進行計算，shift register 會讓每次要進行計算的區域都在最上面。我們也有將 lab3

的加法樹改進，在 lab3 中我採用的是一次處理 256 個元素，每個加法元件相加兩個元素，需要八個 cycles，搭配上 pipeline，加完 1024 個元素需要 $11+1=12$ 個 cycles(最後一個 cycles 是將四個 256 個元素輸入的加法樹結果相加)；在這次的 lab 中，我採用的加法樹模型如圖 2 所示，採用的是更貪心的做法，每個加法元件相加 4 個元素，所以只要 5 個 cycles 就可以算出該區域的 sad 值，之後再用一個 cycles 比較，在計算單一個 32x32 區域的行為上比 lab3 少了將近一半的 cycles 數。

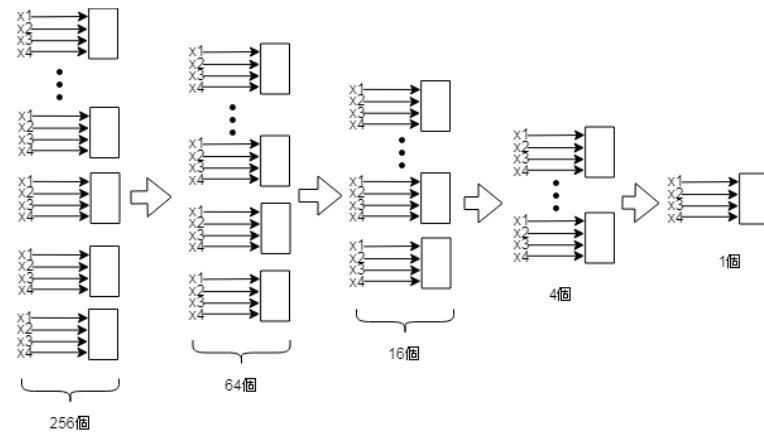


圖 2. Lab6 使用的加法樹圖示

由於還要將軟體寫值這個行為轉為硬體實作，所以要把 lab4 的設計也加進來，基本上使用硬體直接跟記憶體拿值的行為跟 lab4 完全一樣，我們採用固定的 burst length，方便除錯跟撰寫程式碼。

由於 lab3 的計算模式為硬體算完一 32x32 區域後，等軟體將一新的 row 傳進才繼續開始算，我們認為這樣的方式可以與 lab4 的傳輸資料平行處理，因為 lab3 與 lab4 是獨立的計算行為，所以可以平行化處理。這邊的平行化並不是算一行的同時去抓取另外一行，而是先預備好多的 16 個 rows，接著開始 pipeline 計算 16 個 32x32 的區域 sad 以及傳輸新的 16 個 rows，等到傳輸及計算皆完成後就可以將再去將新的 16 行搬進來後就可以再開始計算 sad 值，可參考圖 3。雖然說這樣蠻方便的，可以一次算出 16 個 sad 值，但這種平行化方法效率可能沒有算一行同時讀一行這種效率來的好，原因在於傳輸時間可能會比計算時間還長，那我們會採用的原因是這樣寫較好除錯且時序問題就不會是另一個很惱人的問題。

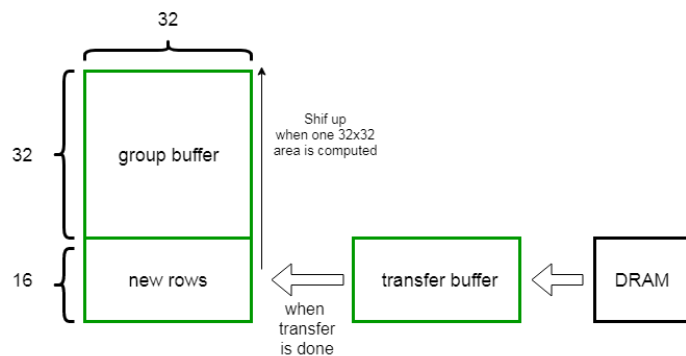


圖 3.硬體平行架構

C. 程式架構－軟體部分

因為我採用全硬體的方式來實作取值、計算並比較找出最小的 sad 值，軟體的事情幾乎都是硬體在做，軟體只要先把 face 跟 group 的起始位址藉由 slave registers 傳入硬體並傳入 hardware active 的訊號給硬體請它開始計算，最後軟體利用 busy waiting 的方式等硬體將結果經由 slave registers 送回軟體即可。我使用的是一次只計算一張 face，因為我是採用 register array 實作，若同時計算兩張 face 會導致電路面積過大無法合成出來。

D. 記憶體位址問題

在這次作業中，我遇到最麻煩的問題就是記憶體位址的問題，一開始我的 burst length 設為 8，這樣可以一次把 32-bytes 長的 row 直接讀進來，但由於圖片在記憶體中是一 byte 一 byte 存的且 master IP 發出的位址卻必須間隔 4 bytes，這樣導致使用硬體實作出 lab3 中往右移動一個 byte 的行為變得困難。

我的第一個解決方案是參考黃右萱同學提出的一次讀取 36 個 bytes。

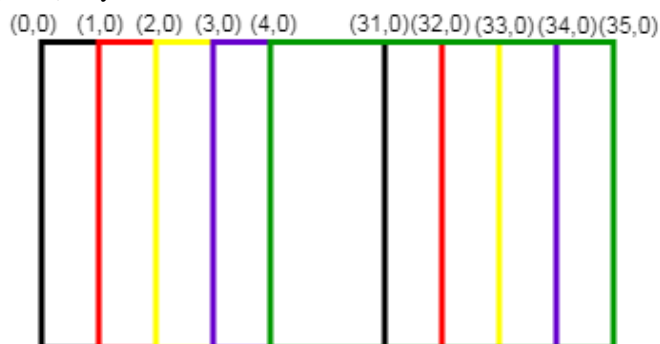


圖 3. 36-bytes 示意圖

如圖 3 所示，當一次讀進一個 row 且是 36bytes 時，下一次要再讀的起始位址就是原本的加 4，且 lab3 中的往右移動一個 byte 後再開始計算的行為也可以被實作出來，且還是可以使用我前面提到的先預先傳 16 個新的 rows 後再開始計算並讀取下一輪的 16 個新的 rows。如果這樣設計，運算的流程會變成算完一個 32x32 區域後，先往右再算，直到算到底後再移回原本的開頭往下再開始新的一輪，這樣的設計看似很好寫，但我在實作的過程中出了一點問題，我遇到的第一個問題是控制訊號不好寫，一直配不出適合

的控制訊號，加上我使用 register array 實作這次 lab，導致電路面積偏大，不好使用 ILA debug，經過好幾天掙扎決定放棄這種方法。

我的第二種解決辦法是在軟體讀取照片後，將照片的像素值存進以 unsigned int 為型態的陣列中，這樣每個 pixel 就是存在一個 4bytes 的空間內，lab3 的往右移動一個 byte 的行為就可以輕易的被實作出來，只是在 lab6 中位址是要移動 4bytes，所以我將我的 burst length 設為 32，這樣才能一次傳完一個原本 32-bytes 的 row，當然這樣的實作方法讓導致傳輸時間是原本 burst length 設為 8 時候 2~3 倍時間，算是一種犧牲吧，拿時間來換取降低實作的複雜度。

E. 流程

1. 首先先由軟體將第一張 face、group 的起始位址藉由 slave registers 傳給 master IP。
2. 軟體使用 slave register 驅動 master IP 請其開始計算，軟體同時進入 busy waiting 等待最終結果由 master IP 經由 slave registers 傳回。
3. 硬體收到啟動訊號會先讀取 face 照片。
4. 接著讀取 group 中指定的 32x32 區域。
5. 再來讀取 16 個新的 rows 進來 master IP。
6. 開始計算及比較 16 個 sad 及讀取新的 16 個 rows。
7. 當讀取完畢後，重複第 6 步直到 face 的位置跑到 group 圖片的最底部，這時將 x 座標右移 1 後回到第三步開始。
8. 當已經算到最後一個 32x32 區域後，將 hw_active 訊號拉下，並將算出的結果藉由 slave registers 傳回軟體。
9. 軟體將結果印出來，若還有 face 圖片需要比對，則回到第一步重新開始動作。

IV. RESULE AND DISCUSSION

```
1-1. Reading group image ... done in 1537 msec.  
1-2. Median filtering ... done in 981 msec.
```

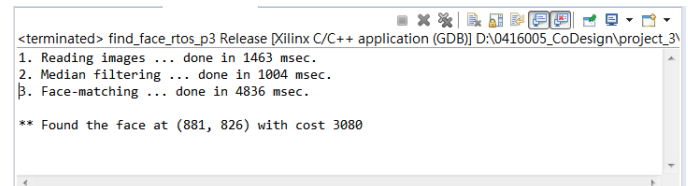
```
1.Face:  
1-3. Reading face image ... done in 4 msec.  
3. Face-matching ...  
miniSAD, 4729  
xIndexAns, 420  
yIndexAns, 798  
done in 1436 msec.
```

```
2.Face:  
1-3. Reading face image ... done in 4 msec.  
3. Face-matching ...  
miniSAD, 10333  
xIndexAns, 1527  
yIndexAns, 439  
done in 1436 msec.
```

```
3.Face:  
1-3. Reading face image ... done in 4 msec.  
3. Face-matching ...  
miniSAD, 5881  
xIndexAns, 1832  
yIndexAns, 645  
done in 1436 msec.
```

```
4.Face:  
1-3. Reading face image ... done in 4 msec.  
3. Face-matching ...  
miniSAD, 4250  
xIndexAns, 1297  
yIndexAns, 489  
done in 1436 msec.
```

圖 4.lab6 執行結果



```
<terminated> find_face_rtos_p3 Release [Xilinx C/C++ application (GDB)] D:\0416005_CoDesign\project_3  
1. Reading images ... done in 1463 msec.  
2. Median filtering ... done in 1004 msec.  
3. Face-matching ... done in 4836 msec.  
  
** Found the face at (881, 826) with cost 3080
```

圖.5 lab3 執行結果

由圖上可以看到，算出一張圖片的時間大約是 1436msec，這個時間大約是 lab3 的三分之一左右(可由圖 4 及圖 5 看出)

V. CONCLUSION

在這次的 lab 中，可以看到若是改成全硬體，速度真的可以加快不少，但同時也有硬體實作上的限制，例如：記憶體位址問題，但若克服天生的限制後其實硬體實作上是比較有效率的。那從這次 lab 的 debug 過程我也發現到，其實計算部分並不會佔到多少時間，反正時間大部分都花在資料傳輸上，記憶體的傳輸才是整個運算過程中的瓶頸。那這個 lab 若有 block ram 來實作的話還可以更快，因為電路面積所以可以同時實作好幾個運算單元，但因為時間及能力不足，沒有辦法實作出來。