HW-SW Optimization

劉佳勳, 0211232

摘要—此次實驗的目的在於優化軟體演算法及硬體設計，讓臉部偵測的程式最佳化，並以最佳的執行速度執行程式

I. 簡介

此次實驗將會有一張 1920*1080 像素的群眾圖片和四張 32*32 像素的小型人臉圖片，程式必須透過計算人臉圖片與群眾圖片每個像素點的差值來計算人臉在群眾中的位置。

此程式的重點在於運用軟體與硬體平行化計算的優勢來加速程式的進行。

II. 程式架構

軟體採用單主程式方案，而主程式分為

1)初始化 SD 驅動。

2)將一張群眾圖片和四張人臉資料讀取到記憶體。

3)將群眾圖片和四張人臉資料從記憶體寫入 HW-IP。

4)彙整從 HW-IP 傳回來的四個結果，分別計算出四張圖片的最佳位置和最低成本。

硬體採用 ZYNQ 平台和 my_dma、compute_sad 兩個 IP

my_dma 的主要功用為取代軟體 mem_cpy 的功能，內部分為一個 Slave 及一個 Master，Slave 的功用主要是將來源記憶體位置和目標記憶體位置從軟體傳遞到 Master，而 Master 則負責記憶體的搬移。

compute_sad 的主要功用為取代軟體來計算兩個圖片像素點的差值，當接收到 hw_active=2ll3ll4ll5ll6 時則負責做圖片的儲存，儲存架構為二十個小型的 blockram，每四個 blockram 負責一張 32*32 像素照片的儲存，所以總共會有五張圖片儲存在這個 IP 當中，並且採用 lab3 的方案，每次移動群眾圖片掃描位置時，只進行了 32 個像素點的搬移，當接受到 hw_active=1 時會開始計算像素點差異的絕對值並開始加總，加總的架構為 10 層兩兩相加的結構，2^10 剛好為 1024，並且用計時器分時，分別輪流計算每張圖的總差值，並且將每張圖的結果回傳到個別的 Slave Register 上。

(圖一)為 HW-IP 架構圖 (圖二)為程式流程圖

III. 程式分析

軟體部分雖然搭配作業系統可以使用到第二個 Crotex A9 核心來進行優化加速，但是程式主要耗時部分已經移至 HW-IP 做運算，再來就是使用作業系統時常遇到 stack 大小不足，經過調整之後還是無法解決，最後使用單主程式而捨棄使用作業系統。

這次的 lab 以往不同的是需要處理四張人臉的資料，所以需要做一些特別的優化和避免使用過量的 LUT。

在 my_dma 我優化了一些程序，在 lab4 時搬移長度是可以使用者自訂的，因此會有比較多的處理程序，而這次的 lab 只會搬移 32byte 也就是 8word 的資料，所以就可以在 Master IP 進行了簡化，讓記憶體的搬移速度提升，以 8-word burst copy 速度 39.18mbps 來說大概比軟體 memcpy 的 31.2mbps 快了 25%。

在 compute_sad 也做了一些改變，在 lab3 時因為只需要處理一個 face 的資料，因此可以儲存在宣告的 reg 上，但是此次 lab 需要計算四張人臉如果採用一樣的儲存方式，會造成 LUT 資源不夠的問題，若將四個人臉資料輪流搬移到 IP 上也可以解決資源不夠的問題，但需要大量搬移資料，造成大量的耗時，因此我選擇了另一種叫 blockram 的解決方案，但 blockram 也不是沒有缺點的，一塊 blockram 同一時間只能寫入或是讀取一個資料，如果把 1024 個像素點都寫入同一塊 blockram 需要 1024 個 clock，同樣的讀取這些資料也需要 1024 個 clock，這造成的時間消耗也不輸上一個解決方案，測試之後時間是 lab3 的 2.78 倍，而最後採用的是將一塊大型的 blockram 分割成四塊小型的 blockram，優點是同一時間可以進行四個資料的讀取或是寫入，跟一塊大型的 blockram 比起來，它只需要 256 個 clock 就可以進行一張圖的讀寫，迅速了許多，也因為 LUT 的使用減少，原本 lab3 放不下的十層兩兩相加結構也放得下去，同時我也嘗試了使用八塊小型的 blockram 來加速，只有一張人臉圖片時相當成功，速度也比四塊 blockram 時快了一些，可惜若要儲存四張圖時，會造成資源過度使用而無法合成，使用小型 blockram 的方法雖然就跑單張人臉效能上還不及 lab3 的作法，但是若一次跑四張圖將時間除以 4，時間是比 lab3 還要短的。

IV. 可能的改進方式

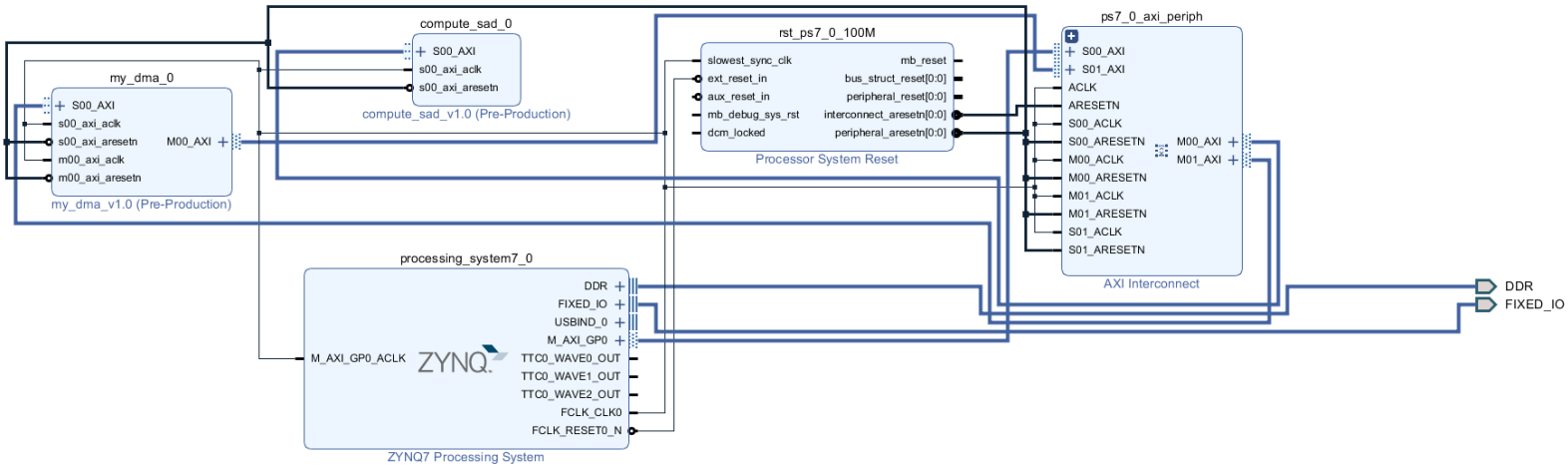
在分析了自己的程式之後，我覺得其實還是有些不足的地方，當軟體把資料傳給硬體之後，做的事只是空等硬體傳回來的結果，如果可以善加利用軟體等待的時間，或許可以再進行加速，若搭配作業系統多核心的優化，則可以讓第一個核心來等待結果，第二個核心做其他照片的計算處理，來達到類似平行運算的效果。

在硬體方面還有一種方法就是，簡化每個 clock 做事的複雜度，然後將時脈進行超頻，若能成功合成則可以一直將時脈往上加，加到 Worst Negative Slack 為最小正值為止，但是這方法會有點運氣成分，若晶片體質好則會成功，若晶片體質不好則有可能出現運算上的錯誤。

V. 總結

在發達的電腦科技中，光是一個好的軟體演算法或是好的硬體設計是不夠的，軟體的優點在於開法簡單成本低，硬體的優點在於平行化的運算，若能將兩者的優點結合，勢必能優化程式的進行，再加上 FPGA 這種可編程的晶片大大得降低了硬體的開發成本，讓軟體與硬體的協同設計越來越重要。

現今電腦的中央處理器大多為通用處理器，而這些通用處理為了處理各式各樣複雜的指令， 成本會比較高且處理核心數少，對於單一且大量的運算表現不盡理想，因此需要硬體來針對軟體進行加速，然而硬體加速也牽扯到很多技巧，像是硬體的設計、資料的傳輸和軟硬體的溝通都會影響到程式的運作速度，一個好的軟硬體設計就像天堂，一個不好的軟硬體設計就會像地獄，反而造成嚴重的負優化。



(圖一) ⤴

(圖二) ⤵