Final Project

0416308 林正偉

I. 簡介

本次 LAB 的目的是要加速 find face。利用 master dma 抓資料，減少讀取資料的時間，還有使用大量的平行化，同時算好幾個 sad 和 face，使速度顯著提升。

II. 架構

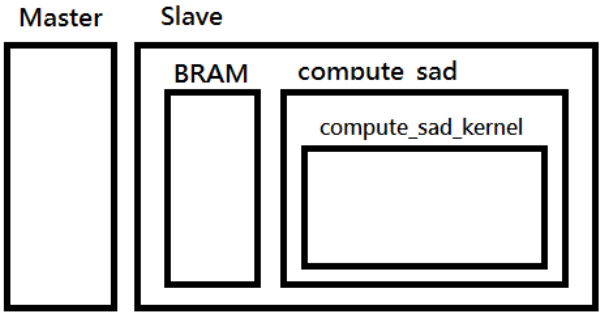

Fig. 1. 所有用到的模組。

A. *Master*

用 data burst 的方式讀取圖片的資料，當讀取 face 時，burst length 為 8 個 words，讀取 group 時，burst length 為 9 個 words。

用 9 個 words 的原因是當 master 讀取圖片時，使用的 source address 必為 4 的倍數，而 9 個 words 的長度可以一次做 4 個 sad，因此下一個 source address 為現在的 source address + 4，故符合要求，如 Fig. 2. 所示。

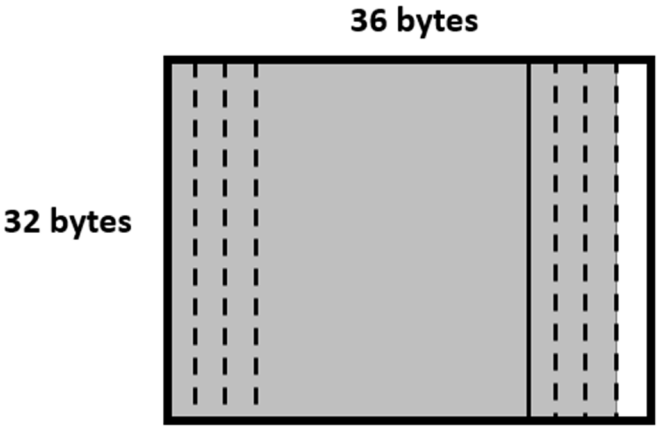

Fig. 2. 灰色部分為 1 次 data burst 所計算的 sad，共有 4 個。

接著是 4K boundary 的問題，我用一個 finite state machine 解決，它會把跨過 4K boundary 的 burst 切成兩段，如 Fig. 3. 所示。

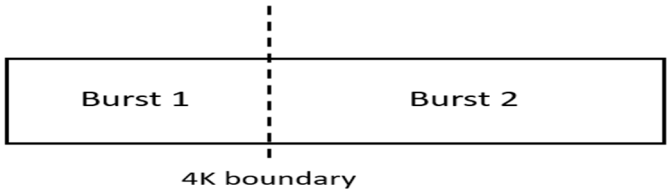

Fig. 3. 4K boundary problem，把一個 burst 切成 burst1 和 burst2。

B. *Slave*

控制什麼時候計算 sad 和什麼時候抓資料，各用一個 finite state machine 控制。

如 Fig. 4. (i)所示，在 FIRST_ROW 時抓完 group 最上面的 32 個 row，接著到 EXEC_SAD，這時計算 sad 和抓資料會同步進行，等到算到最後一個 row 的 sad 時，來到 LAST_SAD，一直等到 hardware done 才回到 INIT_SAD。

如 Fig. 4. (ii)所示，每抓完一個 row 的資料就要等一次 sad，抓完所有 row 後等到 hardware done 才結束。

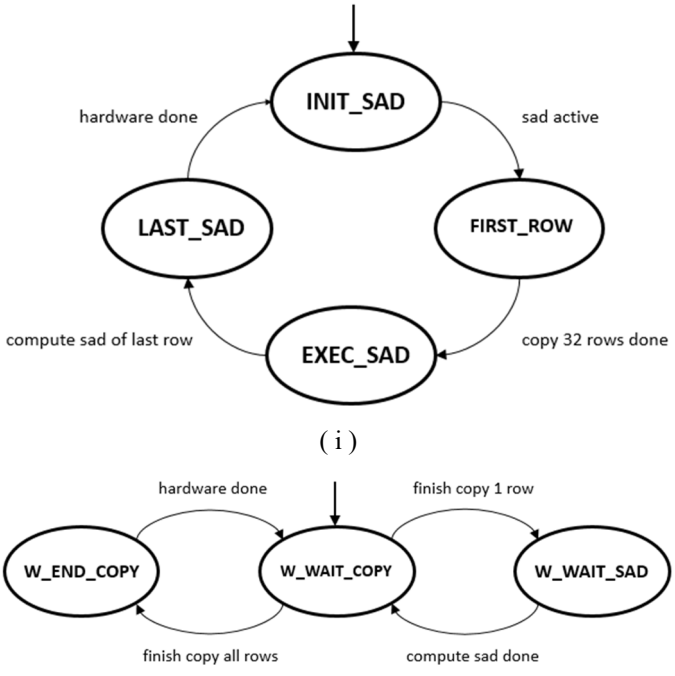

Fig. 4. (i) 控制何時計算 sad 的 FSM。(ii) 控制何時 copy data 的 FSM。

C. BRAM

我一共用了 5 個 BRAM 儲存 group 和 4 個 face 的資料。BRAM 雖然讀取跟寫入都要花一個 cycle，但可以儲存 4 張臉，也就是可以平行計算 4 張臉的 sad，故用此設計。

D. Compute sad

這個模組裡包含了 4 個 compute sad kernel，也就是同時算 4 個 sad。它是 slave 跟 compute sad kernel 的介面，會找出 4 個 sad 中最小的 sad。

E. Compute sad kernel

單純的 absolute difference 和 adder tree，輸出 sad 的值。這個模組設計的目的，是為了能快速嘗試平行計算 n 個 sad 的效率和資源使用率，並在速度和資源中取得平衡。

III. 分析和嘗試

A. Finite state machine 設計

在 Fig. 4. 中，控制抓資料的 FSM 有 W_WAIT_SAD 等待 sad 完成，而計算 sad 的 FSM 卻沒有等待抓資料完成的 state，是因為我想讓計算的過程少一個 cycle。原本設計有多一個等待抓資料完成的 state，但這樣 1 張臉跑完會多花 (1080-32+1)*((1920-32)/4+1) = 496117 個 cycles，大約是 5 毫秒。在這裡 5 毫秒看起來微不足道，但我嘗試做 32 個 words 的 burst 時，1 張臉只花 24 毫秒，此時 5 毫秒就佔大約 20%的時間了，故這樣設計。

B. Burst length 設計和嘗試

- Burst length 設為 9 個 words 時，同時計算 4 個 sad 需要 37 個 cycles，所以 1 個 sad 需要 37/4 = 9. 25 個 cycles。我試著把 burst length 調到 16 個 words，發現一張臉只要 35 毫秒就算完了，32 個 sad 花了 40 個 cycles，所以 1 個 sad 花了 40/32 = 1. 25 個 cycles，只是資源使用量增加了將近 3 倍。接著把 burst length 調到 32 個 words，計算 3*32 個 sad 需要 3*40 個 cycles，平均 1 個 sad 要花 1. 25 個 cycles，跟 16 個 words 的時候一樣，但 1 張臉花了 24 毫秒，是前者的 1. 458 倍，資源使用率跟前者差不多。
- 根據先前的嚐試，發現在 IP 要塞入 2 張臉的前提下，同時計算 32 個 sad 幾乎是極限，不然資源不夠。我選 16 和 32 個 words 的 burst，是因為前者要同時計算 32 個 sad，而後者要計算 3 組 32 個 sad。它們 1 個 sad 平均都要花 1. 25 個 cycles，理論上需要 1.25*(1080-32+1)*(1920-32+1) = 2476951.25 個 cycles，所需時間為 24.7695125 毫秒。要達到理論值的話，burst 所花的時間要小於計算 sad 的時間，這也驗證了我在 Fig. 4. 沒有 state 等 burst 完成的設計。而根據 Fig. 5. 的結果，發現要使 burst 足夠短的話，burst length 需要 32 個 words。
- 我最後選用 9 個 words 的 burst 是因為 16 和 32 個 words 的 burst 有 bug 還沒修好，32 個 words 的還有 critical path 的問題。Fig. 6. 為資源使用量，從表中也可以看到 burst length 越長，critical path 越長。

TABLE I. 算 1 張臉所需的時間

Burst length (words)	Time (ms)
9	239
16	35
32	24

Fig. 5. Burst length 和計算時間的關係。

TABLE II. 資源使用量

Burst length	WNS	LUT	FF	BRAMs
9 (1 face)	0.506	5507	4591	8.00
9 (4 face)	0.202	20262	11036	20.00
16	0.089	22969	15093	11.00

Fig. 6. 資源使用量。表中 9 (1 face) 代表 burst length 為 9 個 words，而且 IP 裡只塞 1 張臉，而 9 (4 face) 代表 IP 裡塞了 4 張臉，可以同時算 4 張臉的 sad。

IV. 總結

用 dma 抓資料使速度從 LAB3 的 1 張臉 5 秒進步到現在的 4 張臉 239 毫秒，可見讀取資料的速度是 bottleneck。在這次 LAB 中可以發現計算 1 張圖在 33 毫秒以內是可能的，所以可以用來做 30 fps 影片的人臉辨識。因為現在的速度已經達到理論值，所以未來如果要再加速的話，需要增加開發版的資源量，說不定可以做到 60 fps 影片的人臉辨識。

REFERENCES

[1] https://timetoexplore.net/blog/block-ram-in-verilog-with-vivado

[2] https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf