CS5120 Homework Assignment #4

Name: 王領崧 Student ID: 107062107

1. Design Concept

此次 lab 要我們利用軟硬體的方式分別實作出 fc & conv，並且硬體的部分會借助 PicoRV32 和 Main memory 做資料的運輸。底下會對軟體、硬體、PicoRV32(放到 discussion 的部分) 有更仔細的說明。

軟體

軟體的部分單純且簡單，就是將 python code 轉成 c code，shape & data 也都是參數可以直接 access，實作上就只要把 7 層 for loop 寫出來即可，唯一比較麻煩的是在加的過程中， c 沒有像 python 那麼方便可以一次傳多個數值到 array，需要用迴圈額外處理。

硬體

硬體的部分是由 4 個 module 牽動而成，大致分為兩個部分。

第一個部分是接收 parameter。PicoRV32 讀到初始的值後 (input, weight size)，會送 valid 的 signal 給 dnn_mmap，讓他能依據不同的 dnn_addr，將讀進來的 wdata 儲存到相對應的變數中，然後回傳一個 ready 的訊號給 PicoRV32，有點像是 ack (positive respond) 的感覺，然後這樣反覆做直到所有的 size 都輸入完畢，最後會送一個 write_start 的 dnn_addr signal，表示所有的 size 都送完了，可以開始進行運算。

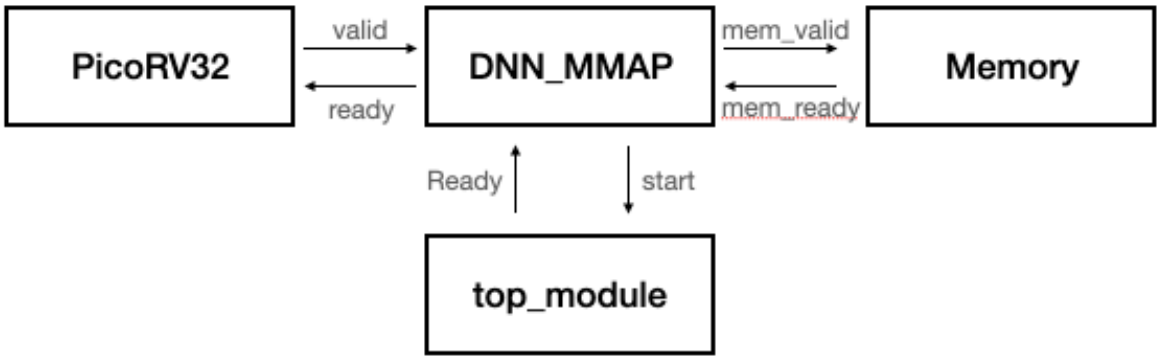
```
DNN_MMAP_WRITE_N: begin
    N <= wdata;
    ready <= 1;
end

DNN_MMAP_WRITE_C: begin
    C <= wdata;
    ready <= 1;
end
```

第二個部分就是進行 conv / fc 的運算，這邊會是 dnn_mmap, top_module, memory 三個一起做溝通，dnn_mmap 和 top_module 的互動與 hw3 是一樣的，傳送一個 valid (start) signal 通知 top_module 可以開始運算，當 top_module 把所有的計算完成後，會回傳 Ready signal (與 picoRV32 的互動機制相似)。同時間，dnn_mmap 也會跟 memory 做溝通，不斷地從 memory 拿取 input, weight，並將 output 結果寫入 memory，這邊溝通的方式也是一樣的，每種 data 有各自的 valid (要寫要讀的時候就會拉起來)，然後當 memory 做完寫入 / 輸出的動作後，就會傳送 ready signal 回來，那因為我們的 memory 只有一個，因此從 top_module 回傳的 addr 都必須加上各自的 offset，才會 access 到正確的位置，要特別注意的是由於 memory 一個 address 只能存放 8 bit，所以 output data 在存放的時候，每次的 addr * 4，一次會佔掉 4 個位置。

```
assign mem_valid_2 = output_wen;
assign mem_addr_2 = output_addr*4 + output_offset;
assign mem_write_2 = output_wen;
assign mem_wdata_2 = output_wdata;
assign output_rdata = mem_rdata_2;
```

另外有個小細節需要注意 (感謝討論區的提醒)，要將 top_module 設定為可以一直重複使用的，那因為 reset 只會有一次，因此這邊採用 valid 來進行後面 reset。

下方為一個簡單的架構圖



2. Simulation and Discussion

1. Software & Hardware profiling

FC layer

soft_fc

```
The software time usage: 12161301
```

hard_fc

```
The hardware time usage: 792
```

12161301 / 792 = 15355 多

CONV layer

soft_conv

```
The time usage: 843065300
```

hard_conv

```
The time usage: 17519
```

843065300 / 17519 = 48122 多

一開始看到結果的時候十分意外，因為沒想到差了萬倍之多，但後來仔細看了 PicoRV32 指令所需的 cycle 數量，平均的 CPI 是 4，就覺得會相差那麼多好像蠻合理的。基本來說，如果一次只能做一個 MAC，那速度就會相差 16 倍 (from HW3)，因此以 CPI = 4 來看，就會相差 64 倍了，加上

memory load & write (CPI = 5)、for loop 的變數計算累加 (這其中還牽涉 if 判斷式，branch 的 CPI 是 6)，速度會相差萬倍就十分合理了。CONV 又因為本身計算的 element 就比較多，維度又比 FC 還要大，自然而然計算的時間就會比 FC 還要更久。

2. Improvement

此次作業在 CPU 與 DNN_MMAP 的溝通，由於 c code 裡面寫的是利用 while loop 來等待 dnn 運算完成，但我覺得比起這種 polling 的方式，採用 interrupt 效果應該會更好，這樣我們就可以先執行其他的工作，提升執行的平行度，不需要 idle 在這裡進行等待，等到 dnn 完成後，自然會發一個 interrupt 提醒，然後 CPU 再把接下來的工作完成即可。

3. Software outperform hardware

硬體設計雖然能利用平行計算來減少 time usage，但同時也會額外產生一塊 area，不過像是這次的作業，在 fc, conv 都需要大量的矩陣運算，會很頻繁地使用到 dnn 加速器，因此比起加速的效果，額外產生面積的 drawback 是完全可以接受的。然而，如果我們今天只是進行一些很簡單的四則運算，或是一些 if-else statement 的判斷，那麼使用了硬體設計，hardware reuse 上就會十分有限，這樣就只會增加 design area，造成 design 要花比較久的時間，出來的 performance 也未必比 software 來得好。

4. PicoRV32

picoRV32 是一個執行 RISC-V 32 指令的 CPU，他會把 firmware 裡面的 c code 檔案 compile 成 RISC-V 指令來執行，我們這次使用的 memory interface 是最 simple 的，也有另外一些複雜的 memory interface。CPU & memory 溝通的方式很直觀，就是 valid & ready 的搭配，當 CPU 要寫入 / 讀出 value 的時候，就會送出 valid 的 request，然後 memory 看到 request 後，就會開始執行。

如果是讀取 value，就根據 addr 進行讀取 & 回傳，完成後把 ready 拉起來通知 CPU

如果是寫 value，會根據 wstrb，來決定要寫入的 data bit數量，完成後也是把 ready 拉起來通知 CPU

簡單來說就是一個 request (valid) -> respond (ready) 的機制，picoRV32 也有提供 look-ahead memory interface 讓你能早一個 cycle 了解接下來的動作 (read or write instruction)，對於一些邏輯的判斷更有幫助。

它也有一些參數來讓我們調整，以 lab4 為例，我們 dnn 和 memory 溝通的方式為 mmap，因此我們就會 ENABLE_MMAP，然後把 ENABLE_PCPI 關掉，另外也有初始值設定(REGS_INIT_ZERO)、指令壓縮(COMPRESSED_ISA) 等等的參數可以運用。

3. Summary

Lab4 分別使用軟硬體來 implement conv & fc layer 的計算，硬體計算方面，我們利用 c code 方式給予初始的參數，然後 dnn_mmap 會利用 valid-ready signal 跟 memory 做寫值傳值的動作，軟體計算方面，就是利用 picoRV32 將 c code compile 成 RISC-V 的指令來執行，並做 performance 上面的比較。以分析的結果來看，執行速度可說是天差地遠，已經是好幾個量級差距，但綜觀指

令耗費的 cycle、硬體 design 平行度，有如此大的差距也算是蠻合理的。經由這個 lab 也深刻體會到，為什麼越來越多的公司要發展自己的 hardware 來輔助，因為整體的效能真的能提升很多~

遇到的困難 & 心得: 這次的作業比起上一次來說容易很多，只是要先看的東西比較多 (RV32 的介紹，trace 已經寫好的 code)，助教也很好心的把 TODO 都標示上去，讓我們能照著步驟來完成。唯一卡住的地方就是 output addr 給值的部分，想說怎麼用輸出的值都感覺是 overflow，後來用 waveform 從頭檢查後才發現 output activation 因為是 32-bit，所以從 top module 輸出的 addr 應該要 *4 來做擺放，才順利地跑過 fc & conv。這次作業是我第一次看到軟硬體的整合，了解到如何用硬體來加速，同時使用軟體來進行一些控制，蠻有意思的，希望 final project 也能往這方面去發展和研究。