Homework 4 Report

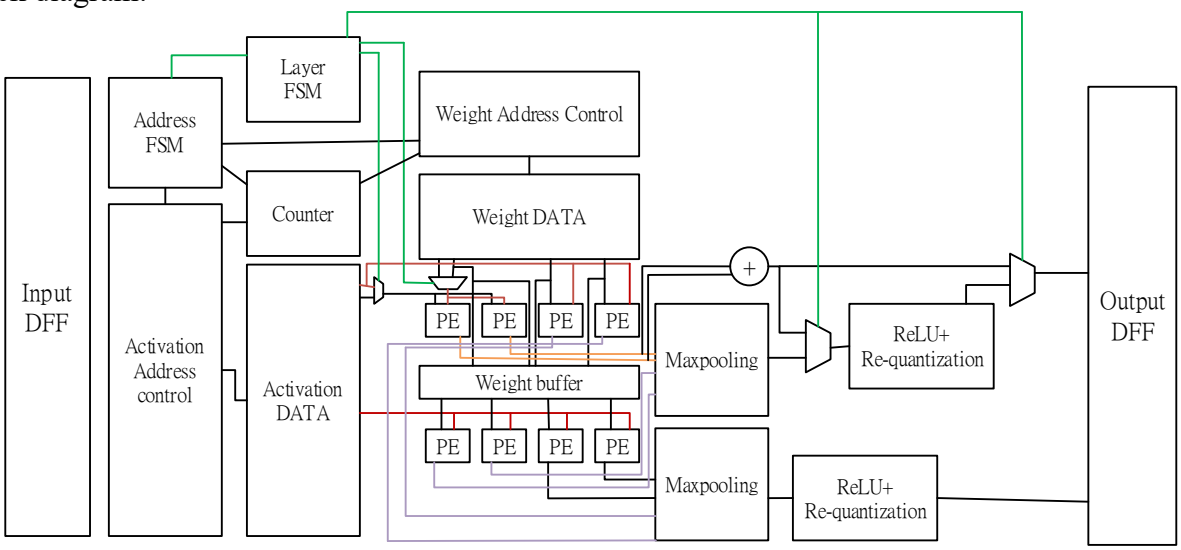
Student ID: 111061642 Name: 王煒翔

1. Design concept:
   - Explanation of the overall hardware architecture, and block diagram of each component.

我本次作業，使用了 8 個 pe，也就是 40 個 mac，這樣在 conv1 時，6 個 cycle 可以做出 8 筆 output，經過 maxpool 以及 actquantization，可以產出 2 筆 output，達成 data reuse，由於會產出兩筆 output，所以我的 maxpool 以及 actq module 各開兩個，分別給上排的 pe 以及下排的 pe 去做使用；由於 maxpool 跟 relu 彼此是獨立的，所以我 relu 放在 actq module 裡面，這樣 relu 只需做一次。

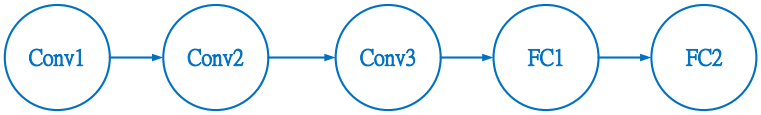
1. Pe module:有 5 個乘法以及 6 個加法，中間切 pipeline，5 個乘法做完，下個 cycle 再做加法。
2. Maxpool module:有 4 筆 input 進來去做比較，最終會輸出 1 筆 output
3. Actq module:這是一個純 combinatorial circuit，後面不需要做任何運算，所以我出去可以直接給 wdata，寫入 sram，這裡的 relu，我把它當成是判斷 output 是否要輸出為 0，如果我的 in_data<0，代表他不用做乘 scale、右移，以及 cliamp，直接給 output data 為 0 就好了。
4. Layer fsm: 這是拿來控制我的 scale 目前的值要為多少，以及去告訴 address fsm 現在在哪一層，由於我 fully connected ，使用其中兩個 pe 去做運算，節省面積，所以我在第一個 pe 以及第二個 pe 加了 mux，去判斷說現在是要做 conv 還是 fully connected，然後給相對應的值進去 pe，後面的 actq 也是，因為第三層以後就不用做 maxpool，直接送進去 axtq 去做處理，則最後一層不需要做 actq，pe 算完直接出去就好了，所以我 actq output 那裏也有 mux 判斷要用哪一個去做輸出，然而第三層以後只需要第一排的兩個 pe，所以第二排的 module 在第三層以後就沒再用了，所以可以不用去做處理
5. Address fsm:這是本次作業佔最大的一部份，其中 weight 的 address 相對簡單，每一層都是+2 的累加，除了最後一層，由於只有 84 筆 input，所以地址只有 21 個，這樣在最後一筆需要做額外的處理，在 activation address control，就沒有這麼簡單，第一層 conv，地址需要以+8 的形式去做累加，第二層改成+4，第三層以後都是 fully connected，所以地址改成+2 的形式去累加
6. Counter:這次主要有 input channel counter、output channel counter、row counter、col counter、writecounter，都很直觀，其中 writecounter 是來跟 address 現在要寫哪個地址。

Block diagram:

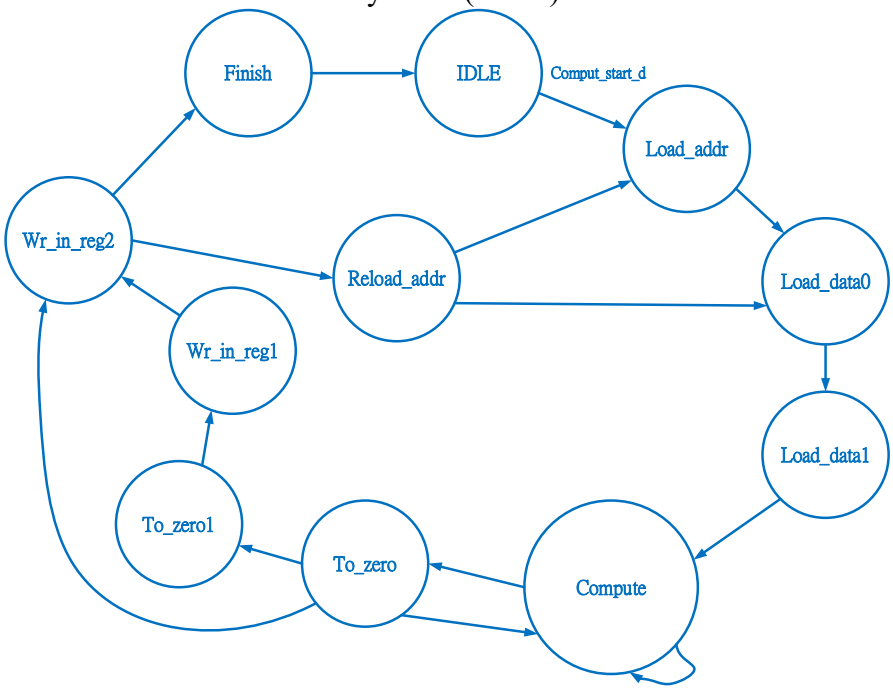


- State diagram and its detailed description (if any).

我使用了兩個 fsm，一個是 layer fsm，一個是 address fsm



Layer fsm(state1)



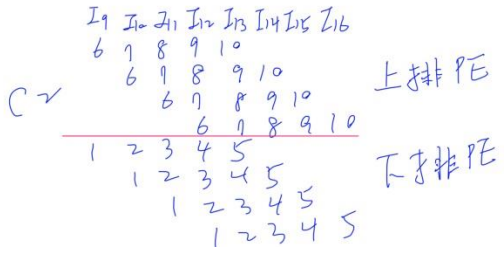
Address fsm(state)

Layer fsm： 由於 SRAM data 擺放為固定，所以每一層都有不同的 offset。

Address fsm： 由於有加 I/O DFF，所以從 SRAM 取 data 到 lenet 需要 2 個 cycle，所以有了 Load_data0 以及 Load_data1 來等待這段期間，好讓在 Compute 的時候能夠拿到正確的 data，To_zero 以及 To_zero1 指的是說當我完成 4 筆 data 時需要寫進去 SRAM，此時的 PE 會停止運作，To_zero 指的是上排的 PE 運算完要歸零，To_zero1 指的是下排的 PE 運算完要歸零，Wr_in_reg1 指的是所有的 pe 都願算完成，接下來會進行 maxpooling，Wr_in_reg2 指的是當完成 maxpooling 時要做 re-quantization，做完就直接寫進去 SRAM，由於 SRAM 是同步讀寫，所以為了不要讓 PE 拿到不正確的值，所以要重新 Reload_addr，但可以發現有兩條路可以走，當走到 Load_addr 時因為層與層之間需要轉換 scale，此時 Reload_addr 是用來叫 Layer scale 的 addr，Load_addr 則是叫 data 的 addr，另一條走到 Load_data0 指的是說這層還沒有結束，不用 reload layer scale，所以 reload_addr 這時候就是直接 reoload data 的 addr，compute 有繞回自己，指的是說現在還沒有 4 筆 output，也就是說 pe 在運算的同時，output 在持續在做後面的 maxpooling、re_quantization。

- Explanation of the dataflow.

Data reuse 的部份，activation 採用 spatial reuse 的方式，把進來的資料給全部的 pe 去做使用，weight 則採用 temporal reuse，加入 weight buffer，給下排的 pe 去做使用，以下是示意圖，不過 fully connected 的 data reuse 只有 input，一行算完再重新讀一次 input，weight 則一直往下讀。



Spyglass rpt 討論:

Warning ./hdl/lenet.v :Asynchronous reset signal'lenet.rst_n_d'(flop:'lenet.sram_weight_addr0_d[0]') used as non-reset/synchronous-reset at instance 'lenet.maxpool0.\out_data_reg[0] .D' (File Name: './hdl/lenet.v' ,Line no.: '1564')

我原本設計全都採用 sync reset，但是在 gate-level 模擬的時候，發現 sram address_1 的地址會讀不到，這點其實也不太清楚為甚麼，我初始值地址是給 1，但出來的值是 unknown，但我在討論區發現改成 async reset 即可以解決，所以我把內部全部改成 async reset，模擬就可以過了，外部 synce reset，由於助教說內部 reset 沒有硬性規定，所以我覺得這 warning 可以忽略。

Warning ./hdl/lenet.v:Rhs width '25' with shift (Expr: '((M[24:0] * in_data) >> 16)') is more than lhs width '8' (Expr: 'temp'), this may cause overflow [Hierarchy: ':lenet:actq0@actq']

這個是在指這裡

```
assign temp =  (M[24:0] * in_data) >> 16;
```

我的 temp 是 8bit，這 warning，會 overflow，temp 給 8bit 是因為，到時候 cliamp 的時候也是鎖在 8bit 裡面，所以這警告可以忽略。

Warning ./hdl/lenet.v:Variable 'scale_FC2_d[31:0]' set but not read.[Hierarchy: ':lenet']

這個是在指說我的 FC2 scale 沒用到，但本次作業確實不會用到，所以這個 warning 可以忽略

Fatal ./hdl/lenet.v: Could not find clocks for all the flops. Please add clock SGDC constraint to the design

這個就是我們在跑 spyglass 時沒有給 SDGC file，沒有設置 clock 所導致的，所以我覺得這 fatal 可以忽略

2. Result

Item	Description	Unit
RTL simulation	PASS	---
Gate-level simulation	PASS	---
Gate-level simulation clock period	1.66	ns
Gate-level simulation latency	25187	cycles
Total cell area	44856.1396	um^2

3. Others (optional)
   - Suggestions or comments about this class to teacher or TA.