人工智慧晶片設計與應用

AI-ON-CHIP FOR MACHINE LEARNING AND INFERENCE

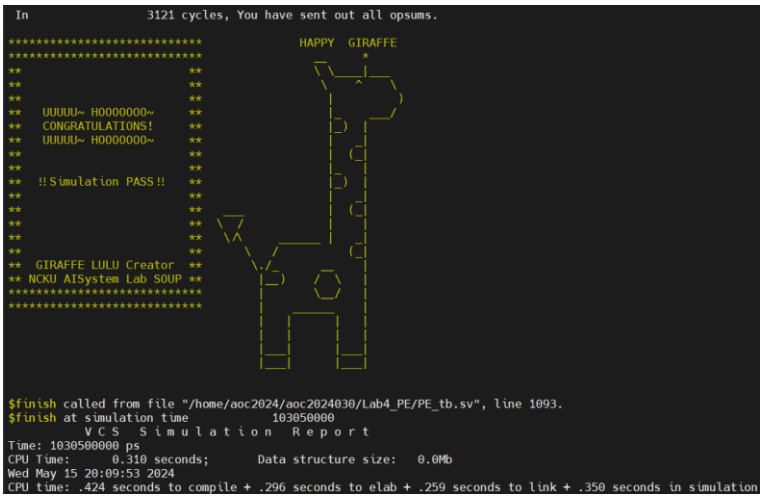
Lab 4

E14096724

鄭喆嚴

1. Pass all testbenches. Screenshots of passing 3 tbs.

   (1) tb0



   (2) tb1

   (3) tb2

2. Cycle comparison of each tb. Each tb contains 3 points.

(1) tb0 : 3121cycles

```
In                3121 cycles, You have sent out all opsums.
```

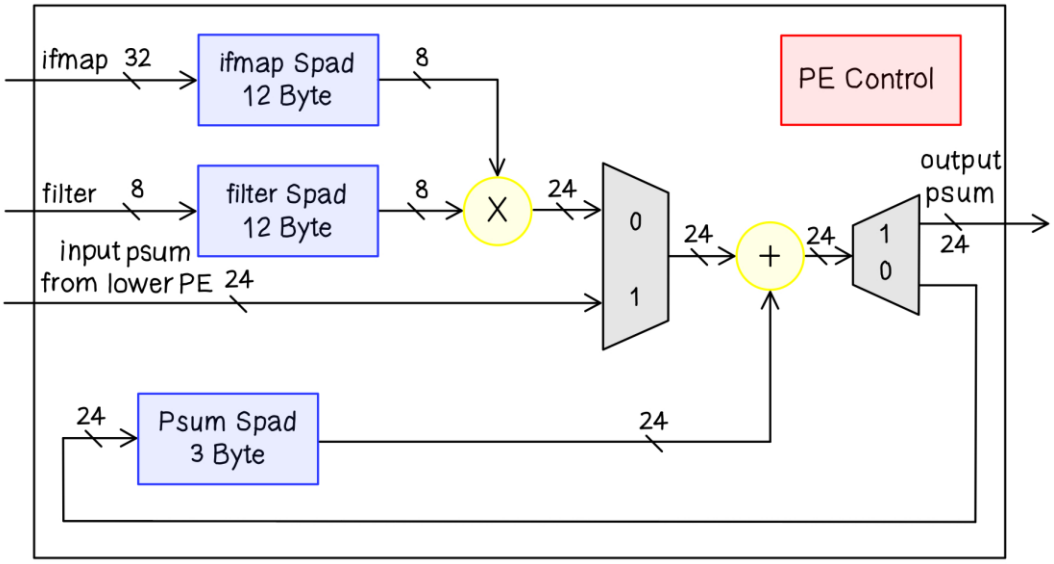
(2) tb1 : 103044cycles

```
In              103044 cycles, You have sent out all opsums.
```

(3) tb2 : 33283cycles

```
In                33283 cycles, You have sent out all opsums.
```

3. Draw your own PE architecture. Write down your total spad storage space and the corresponding functions. Explain how your PE works by FSM state diagram with waveform clearly. Mark down the values you mention.



以下是 Controller 的訊號：

ifmap_Quant_size_store: controller 儲存的 ifmap Quant size 要給 multiplier 的訊號。

filter_Quant_size_store: controller 儲存的 filter Quant size 要給 multiplier 的訊號。

filter_ready: 輸出準備好接收 filter 資料。

ifmap_ready: 輸出準備好接收 ifmap 資料。

ipsum_ready: 輸出準備好接收 ipsum 資料。

opsum_enable: 是否輸出 opsum 資料。

ifmap_spad_addr: 控制 ifmap spad 的地址。

ifmap_spad_wen: 控制 ifmap spad 的寫入。

ifmap_spad_ren: 控制 ifmap spad 的讀取。

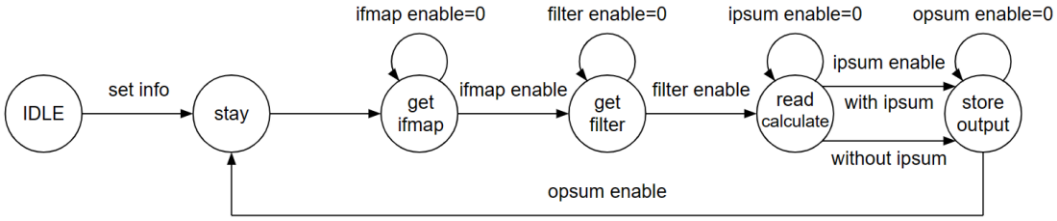
shift: 控制 ifmap 進行 shift 來讀取新的資料。

filter_spad_addr: 控制 filter spad 的地址。

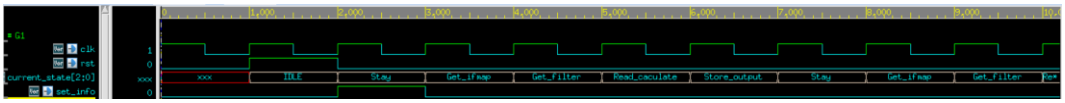
filter_spad_wen: 控制 filter spad 的寫入。

filter_spad_ren: 控制 filter spad 的讀取。

psum_spad_wen: 控制 psum spad 的寫入。
psum_spad_ren: 控制 psum spad 的讀取。
buffer_wen: 控制 buffer spad 的寫入。
buffer_ren: 控制 buffer spad 的讀取。
buffer_sel: 控制 buffer 要輸出 PE 或是輸出到 psum spad。
以下是 PE 整體的 spad 儲存空間：
Ifmap Spad: 12 Byte
Filter Spad: 12 Byte
Psum Spad: 3 Byte
Buffer: 3 Byte
Controller: 36bit = 4.5 Byte
Ch_size_store = 3 bit,
ifmap_column_store = 6 bit,
ofmap_column_store = 6 bit,
ifmap_Quant_size_store = 4 bit,
filter_Quant_size_store = 4 bit,
filter_counter = 4 bit,
ifmap_window_counter = 6 bit,
current_state = 3 bit
Total: 34.5 Byte

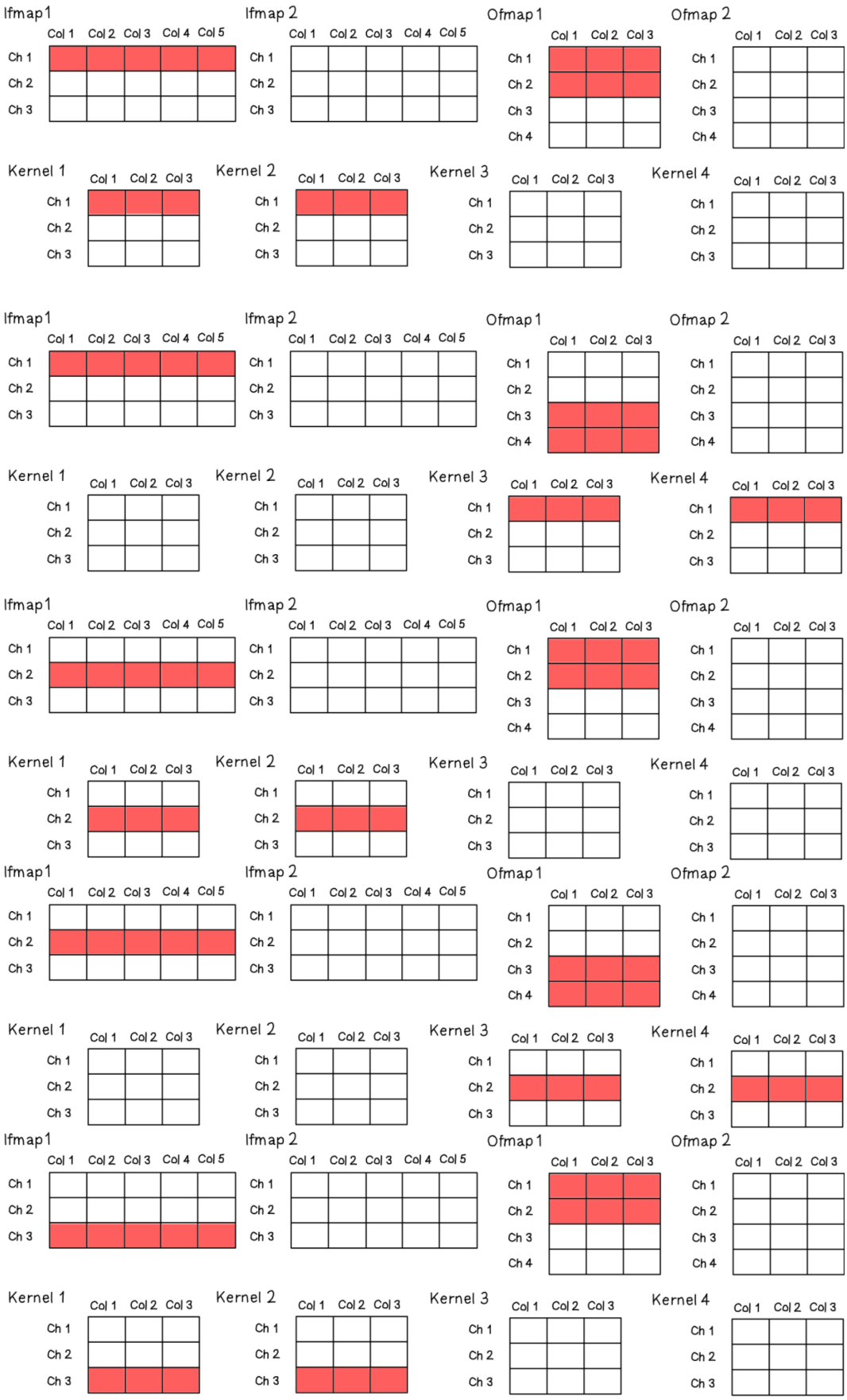


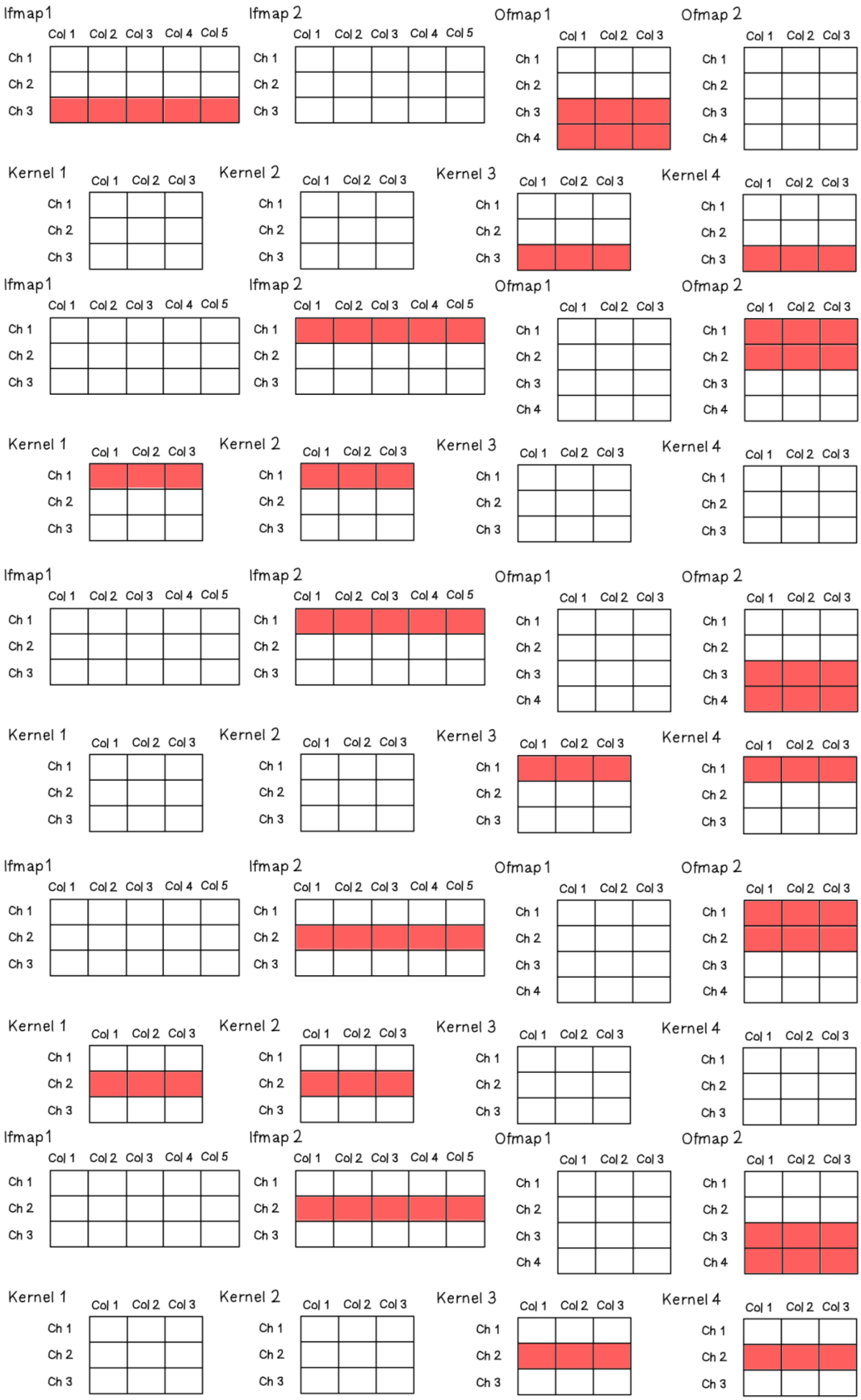
一開始在 IDLE 收到 set info 後開始運算，為了避免輸入資訊太緊湊所以先 stay，接著拿取 ifmap，如果不需要就不拿，得到 ifmap 後接著拿取 filter，如果不需要就不拿，得到 filter 後接著拿取 ipsum，如果不需要就不拿，同時進行乘法加法的計算，計算完後接著進行儲存或輸出。

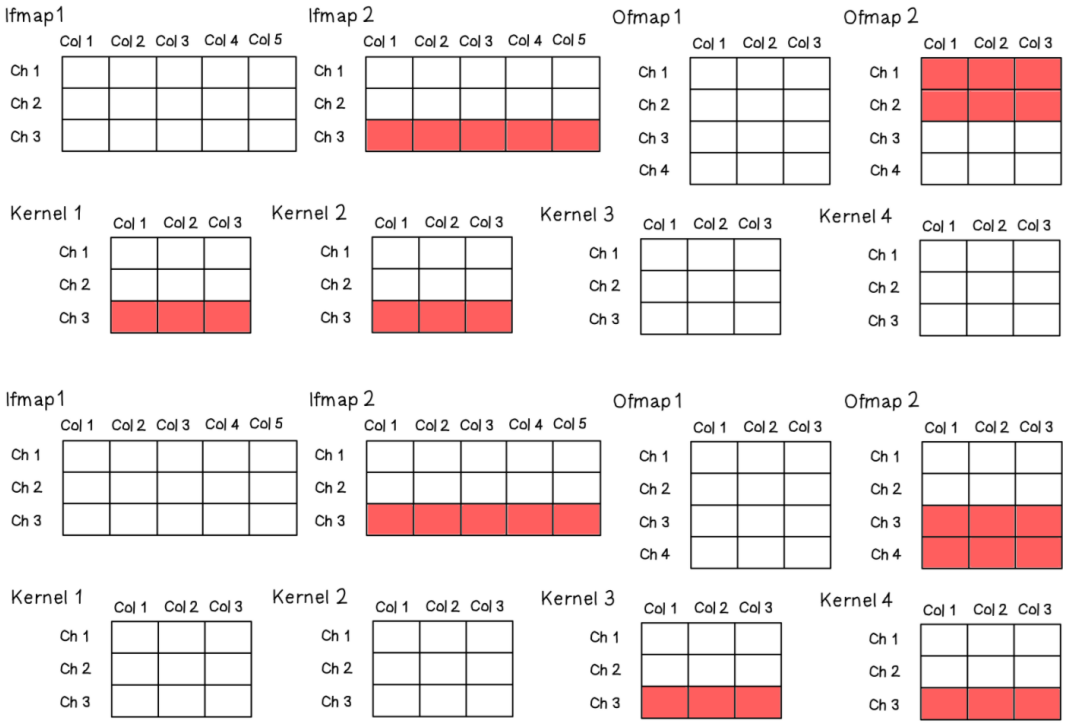


4. Demonstrate Scenario B dataflow by completing 2 3row x 3col x 2ch ofmaps by accumulating 2-channel of 2 3x3 filters(kernel) and 2 5x5 ifmaps under parameters n=1, p=2(kernel),q=1(channel).

下列 dataflow 的圖片中都是 PE1 的 dataflow，都只進行 ifmap 和 filter 中 row 1 的運算，剩下的 row 2 和 row 3 都會來自 PE2 和 PE3 運算完後組合成完整的 Ofmap。


Ifmap 1
Ifmap 2
Ofmap 1
Ofmap 2
Kernel 1
Kernel 2
Kernel 3
Kernel 4
Col 1
Col 2
Col 3
Col 4
Col 5
Ch 1
Ch 2
Ch 3
Ch 4


Ifmap1
Ifmap 2
Ofmap 1
Ofmap 2
Kernel 1
Kernel 2
Kernel 3
Kernel 4
Col 1
Col 2
Col 3
Col 4
Col 5
Ch 1
Ch 2
Ch 3
Ch 4



5. Compare scenario A, B, C from different aspects organized into a table. Analyze under scenario A (n=2, p=1, q=1), scenario B (n=1, p=2, q=1), scenario C (n=1, p=1, q=2) within a processing pass and across multiple passes to complete 2 3row x 3col x 2ch ofmaps by accumulating 2-channel of 2 3x3 filters(kernel) and 2 5x5 ifmaps.

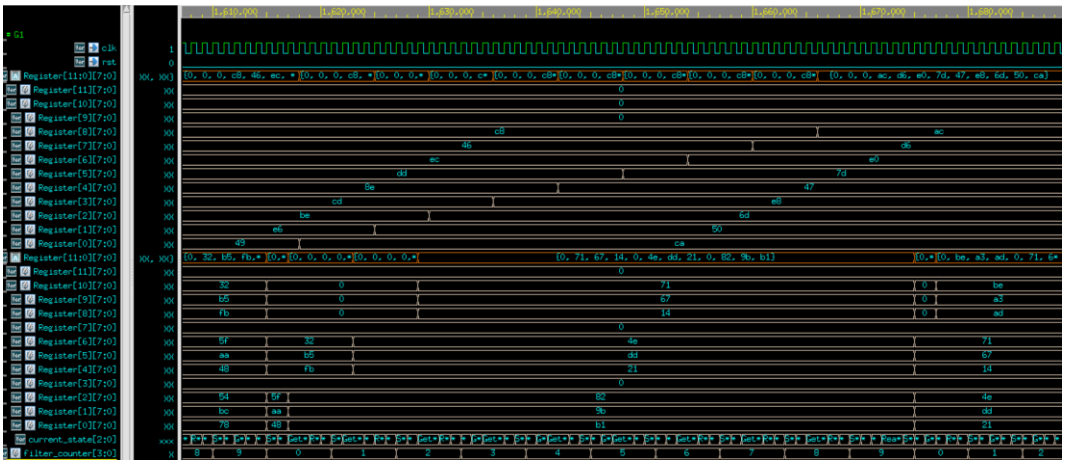
	Scenario A	Scenario B	Scenario C
重複使用	filter	ifmap	opsum
性能	在同時有兩個 ifmap 在 PE 中進行重複使用 filter 可以達到性能的提升，但是需要多次重複拿取 ifmap。	在同時有兩個 filter 在 PE 中進行 ifmap 重複使用可以達到性能的提升，但是需要多次重複拿取 filter。	在同時進行兩個 opsum 的累加可以達到性能的提升，但是需要重複拿取 filter 和 ifmap。
硬體	需要有可以儲存重複使用的 filter 的空間還有儲存 psum 累加器的空間。	需要有可以儲存重複使用的 ifmap 的空間，還有儲存 psum 累加器的空間。	需要有可以儲存重複使用的 ifmap 和 filter 的空間，但 psum 累加器的空間可以比較小。

6. Share your thoughts on this lab. Any takeaways or advice?

我覺得這次的 Lab 蠻有難度的，在拿取資料的方法之前沒有使用過，所以在實作上有遇到許多問題，而且不知道為什麼在使用 verilog 做作業時，使用的 sequential 電路部分有許多問題，clock 上升時應該要拿取 clock 上升前的資料，但不知道為什麼拿取得到 clock 上升後的資料，導致我需要加上一個 stay 的 state 來避免錯誤，不知道是不是 system verilog 和 verilog 混合執行會有什麼問題。

7. Modify your PE to a pipelined architecture or with zero-gating / zero skipping function. You can choose one from 3 functions to implement.

我將我的 PE 實現 zero skipping 當有 0 出現時，不會進行運算並且直接掉過，下圖中的 opsum 會有 3 個 0，所以跳過這些運算後只會計算 9 次，途中的 filter counter 只計算了 9 次。



Theoretically or Practically compare with basic PE architecture on page 33, 58 by required cycles, effectual and ineffectual operations(lec4 p.16-19、25), etc.

我的 PE 是透過 controller 實現 zero skipping，所以在整體的 PE 架構上不會有太多不同，並且在 channel 不是 4 時，ifmap 和 filter 中會有許多個 0 會節省蠻多 cycles。