

# AI on Chip 2024

## LAB IV Processing Elements REPORT

Student name: \_\_胡家豪\_\_

Student ID: \_\_N26122246\_\_

# 目錄

一. Implementation Result.....	3
1. TB.0 .....	3
2. TB.1 .....	3
3. TB.2 .....	4
二.Implementation Cycle Result .....	4
三.PE architecture and Design ideas .....	4
1. PE architecture .....	4
2. Spad storage space and functionality .....	5
3. FSM state and functionality .....	5
4. Waveform Result.....	6
四.Demonstrate Scenario B dataflow .....	7
五.Compare scenario A,B,C from different aspects.....	9
六.Share your thoughts.....	9

# 一.Implementation Result

## 1. TB.0

```
In 922 cycles, You have sent out all opsums.

*****
***** HAPPY GIRAFFE *****
*****
**
**
**
** UUUUU~ H0000000~
** CONGRATULATIONS!
** UUUUU~ H0000000~
**
** !!Simulation PASS!!
**
**
** GIRAFFE LULU Creator
** NCKU AISystem Lab SOUP
*****
*****

$finish called from file "/home/kevin199907/AOC/LAB4/PE_tb.sv", line 1093.
$finish at simulation time 100000000
VCS Simulation Report
Time: 1000000000 ps
CPU Time: 0.980 seconds; Data structure size: 0.0Mb
Sun May 12 21:59:05 2024
CPU time: .950 seconds to compile + .646 seconds to elab + .515 seconds to link
+ 1.037 seconds in simulation
```

## 2. TB.1

```
In 28163 cycles, You have sent out all opsums.

~ ~ ~ Simulation PASS ~ ~ ~
* * * * *
* TWO HAPPY Giraffes *
* * * * *

$finish called from file "/home/kevin199907/AOC/LAB4/PE_tb.sv", line 1093.
$finish at simulation time 100000000
VCS Simulation Report
Time: 1000000000 ps
CPU Time: 1.520 seconds; Data structure size: 0.1Mb
Sun May 12 21:59:37 2024
CPU time: .895 seconds to compile + .687 seconds to elab + .525 seconds to link + 1.568 seconds in simulation
```

### 3. TB.2

```

In      8109 cycles, You have sent out all opsums.

      ~~ Simulation PASS ~~
      * * * * *
      * TWO HAPPY Giraffes & A baby Giraffe *
      * * * * *



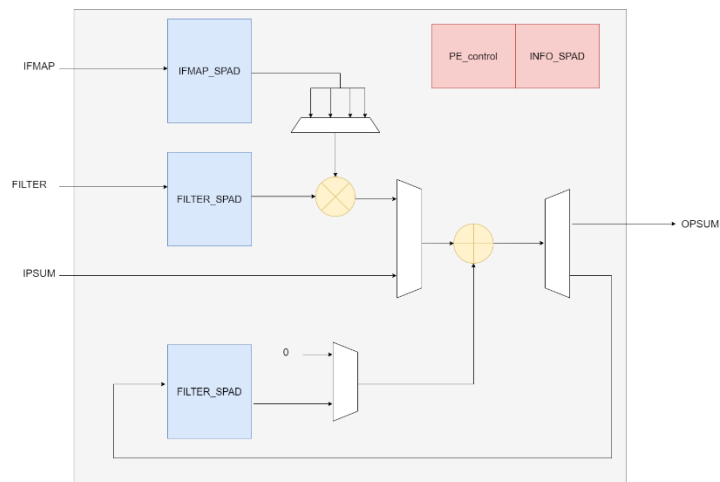
$finish called from file "/home/kevin199907/A0C/LAB4/PE_tb.sv", line 1093.
$finish at simulation time      100000000
      VCS Simulation Report
Time: 1000000000 ps
CPU Time:      1.060 seconds;      Data structure size:      0.0Mb
Sun May 12 22:00:06 2024
CPU time: .912 seconds to compile + .682 seconds to elab + .528 seconds to link + 1.116 seconds in simulation
  
```

## 二.Implementation Cycle Result

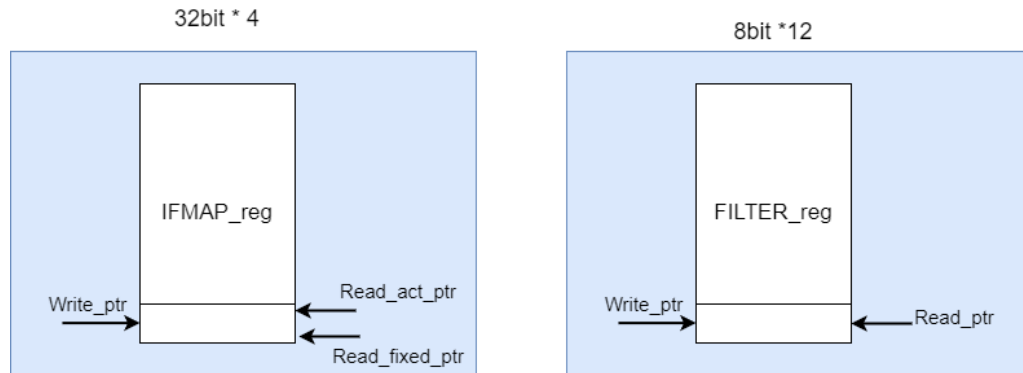
Testbench	Tb0	Tb1	Tb2	Tb3
Clock Cycle	922	28163	8109	-

## 三.PE architecture and Design ideas

### 1. PE architecture



## 2. Spad storage space and functionality



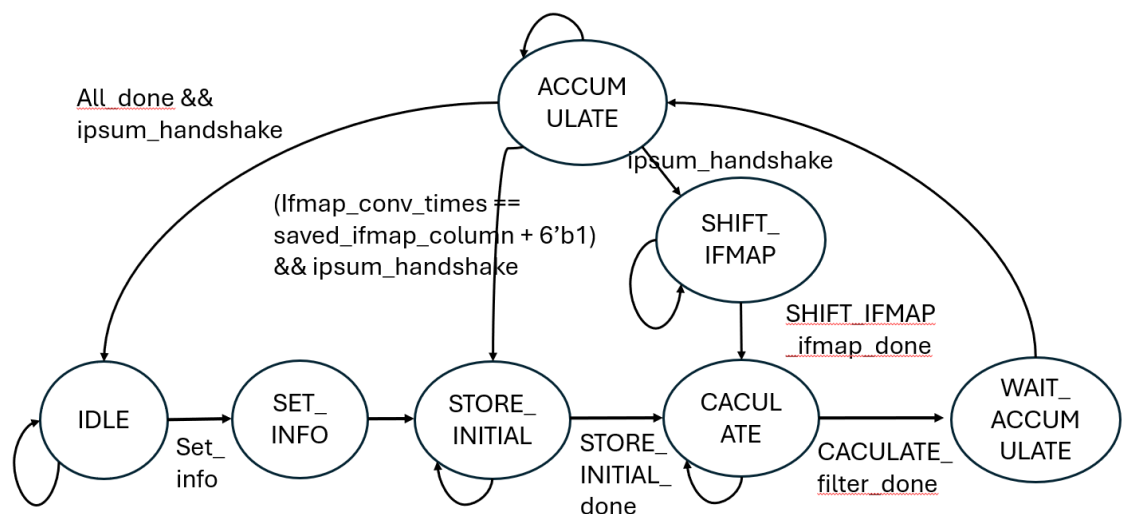
我使用的 spad 大小為 ifmap 的 16 Byte 與 Filter 的 12 Byte。

Ifmap 總共使用到三個 pointer 來進行讀寫，分別是 Write\_pointer 負責資料寫入的地址、Read\_fixed\_pointer 負責固定 filter 框選到的大小與 Read\_act\_pointer 負責實際將資料讀出。

一開始 Write pointer 先寫入三筆資料，之後如果 sliding window 需要往右，就 write pointer 與 read fixed pointer 一起+1，然後 Read act pointer 會在 caculate 時對齊 Read fixed pointer 的位置，直到加到 Write pointer 的位置後告訴 control 完成運算

Filter 則只用到兩個 pointer，分別為 Write pointer 負責資料寫入的地址、Read pointer 負責資料讀取，每次計算完一次 convolution，Read pointer 就跳回原本的位置，直到要讀取新的 filter 進來。

## 3. FSM state and functionality



IDLE：閒置狀態

SET\_INFO：吃到「set\_info」後進入，此 stage 在儲存 Layer 的狀態。

STORE\_INITIAL：存完 layer 後直接進入，在此狀態會將要儲存的 ifmap 以及 filter 讀入，之後每次要拿新的 ifmap 就會進入這個狀態重新拿取 ifmap 與 filter，並且會有 counter 計算要讀取幾個 ifmap 與 filter，都讀取後將 STORE\_INITIAL\_done 拉起

CACULATE：在這個 stage 會將 ifmap 與 filter 進行相乘相加，利用 counter 計算要相乘相加多少筆資料，完成後將 CACULATE\_filter\_done 拉起。

WAIT\_ACCUMULATE：用於緩衝

ACCUMULATE：這邊將讀入的 ipsum 與算出來的結果相加乘 opsum，並把結果輸出。並且分為兩個情況。

情況 1：該次計算還沒有算完這個 ifmap，那就等待 ipsum Handshake 後將 ifmap 右移一格。

情況 2：該次計算已經算完這個 ifmap，那就等待 ipsum Handshake 後重新讀入 filter 與 ifmap。

SHIFT\_IFMAP：這個狀態是一個 ifmap 還沒算完，所以進行右移，移完後就直接回到 CACULATE 進行下一次的運算。

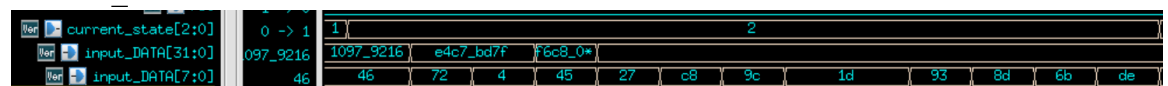
#### 4. Waveform Result

SET\_INFO：



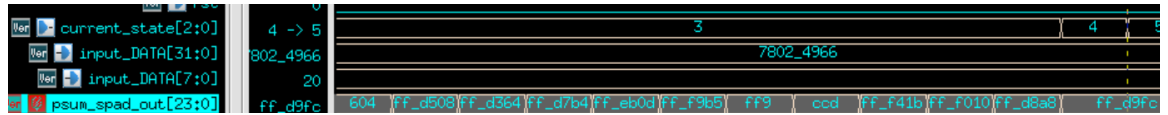
由 IDLE 轉到 set\_info，並且讀取 layer 資料

STORE\_INITIAL：



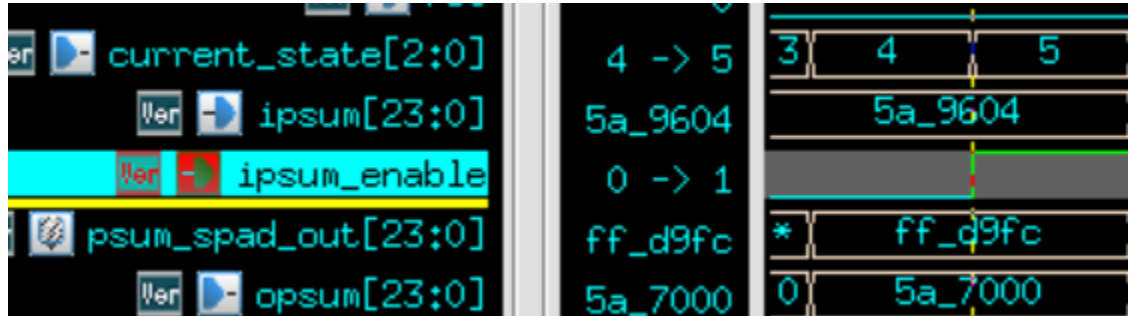
在這邊儲存四筆 ifmap 與 12 筆 filter (ch\_size=4)

CACULATE：



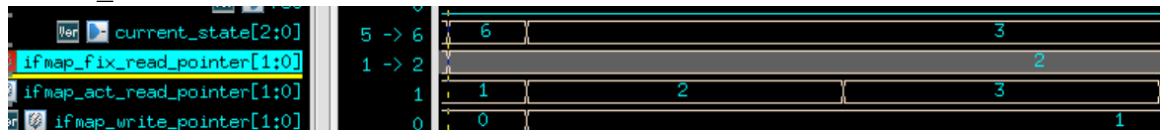
在這個階段進行 mac 運算並且存到 PSUM\_SPAD 中

ACCUMULATE :



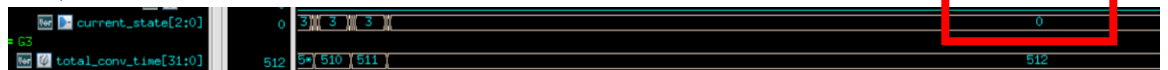
等待 Ipsum 將資料送進後，與 psum 相加後成為 opsum 送出。

SHIFT\_MAP :



移動將 wrtie pointer 與 fix\_read pointer 都加一，並且讓 act read pointer 與 fix read pointer 對齊。之後繼續進行 ACCUMULATE 的動作。

由於我使用的是 tb0，並沒有結束資訊的 wire，所以我計算我總共需要幾次 convolution 以及 ifmap\_quant\_size 的資訊來回到 IDLE，以 TB3 而言，總共進行 512 次



#### 四. Demonstrate Scenario B dataflow

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

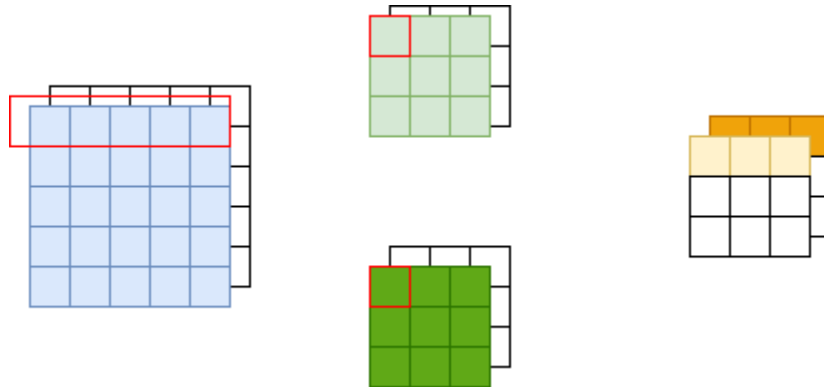
N：一個 PE 一次執行幾個 ifmap

p：PE 執行一次有多少 kernal 存在 PE

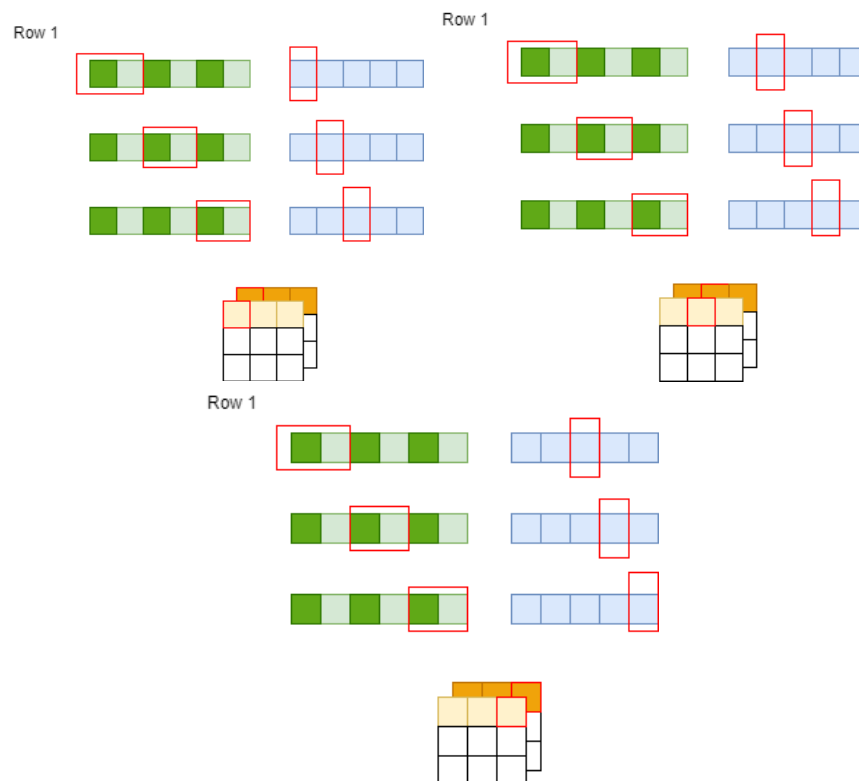
q：一次算多少個 channel

Scenario B： ifmap reuse

根據上面的要求，我們可以將要進行的運算畫成如下所示，下面藍色為 ifmap、綠色為 kernal、黃色為 opsum 前三次運算會得到的部分。



假設我們用 PE1,1 當作示例，ifmap 會直接存入 spad 之後在完成這個 row 之前都不再另外讀取其他的 data，而 filter 的 kernel 1 channel 1 與 kernel 2 channel 1 的一個元素則會由左至右不斷讀取進 PE 內進行運算，一次讀取兩個，在重複使用 ifmap 的情況下直接將兩個 kernel 完成運算，並且將輸出與其他的 PE 結果相加，完成一個 opsum，如下所示：





## 五. Compare scenario A,B,C from different aspects

	scenario A	scenario B	scenario C
Data reuse	Reuse filter，與不同的 ifmap 進行運算	Reuse ifmap，與不同的 filter(但相同 channel)進行運算	將不同 channel 但相同位置的部分同時運算，看要 reuse ifmap 或是 kernal
Spad size	Ifmap 每個 row 可能有較大 size 的 spad	Kernal 的 row 通常為 5 個或 3 個，較小 spad	中等
Memory R/W	較少	較多	中等
Energy consumption	較小	較大	中等

## 六. Share your thoughts

本次作業比起前幾次作業而言來說規模比較大，也非常具有挑戰性。我覺得最難的部分在於理解作業要我們做甚麼。由於 Eyeriss 的架構比較複雜，而且又有許多 dataflow 的方式，所以光是看完助教的投影片與 youtube 影片就花了不少時間。

實作時我一開始打算使用 FIFO 當作 SPAD 的儲存，但是後來做出來在看波型時才發現 FIFO 會將 data 丟棄，而我們必須要將 data reuse，所以行不太通，浪費了不少時間才做出現在這個版本。

然而即便畫好 FSM，怎麼對 clock 也是一門學問，後面我常常在處理某些地方沒有考慮 handshake 的問題。

總體而言，雖然這次的作業相對困難，但是收穫也很大！