AI on Chip 2024

LAB Ⅳ Processing Elements

REPORT

Student name: \_\_胡家豪\_\_\_\_\_\_\_\_\_

Student ID: \_\_N26122246\_\_\_\_\_\_\_\_

目錄

[一. Implementation Result 3](#_Toc166444781)

[1. TB.0 3](#_Toc166444782)

[2. TB.1 3](#_Toc166444783)

[3. TB.2 4](#_Toc166444784)

[二. Implementation Cycle Result 4](#_Toc166444785)

[三. PE architecture and Design ideas 4](#_Toc166444786)

[1. PE architecture 4](#_Toc166444787)

[2. Spad storage space and functionality 5](#_Toc166444788)

[3. FSM state and functionality 5](#_Toc166444789)

[4. Waveform Result 6](#_Toc166444790)

[四. Demonstrate Scenario B dataflow 7](#_Toc166444791)

[五. Compare scenario A,B,C from different aspects 9](#_Toc166444792)

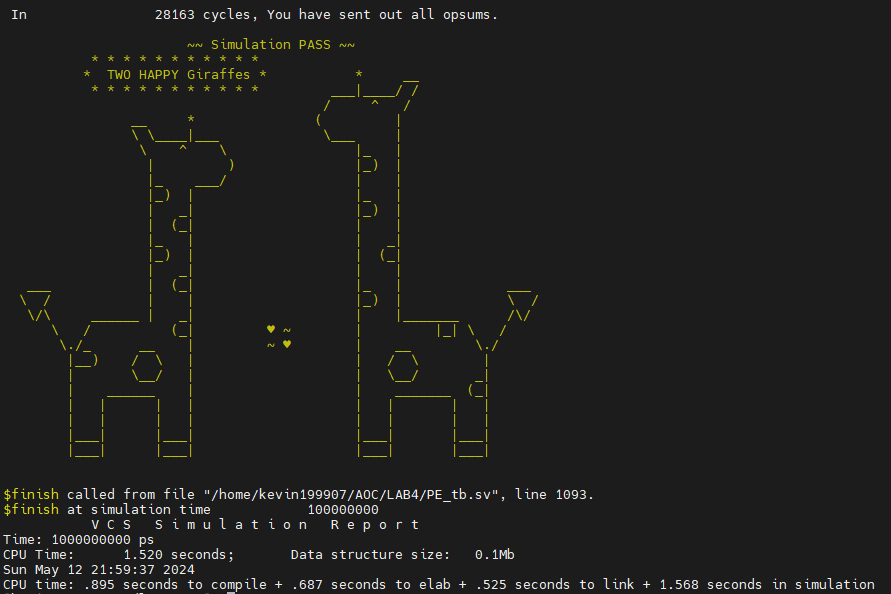
[六. Share your thoughts 9](#_Toc166444793)

1. Implementation Result
2. TB.0

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

自動產生的描述

1. TB.1



1. TB.2

一張含有 文字, 螢幕擷取畫面, 字型, 圖表 的圖片

自動產生的描述

1. Implementation Cycle Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testbench | Tb0 | Tb1 | Tb2 | Tb3 |
| Clock Cycle | 922 | 28163 | 8109 | - |

1. PE architecture and Design ideas
2. PE architecture一張含有 文字, 圖表, 螢幕擷取畫面, Rectangle 的圖片

   自動產生的描述
3. Spad storage space and functionality

一張含有 文字, 螢幕擷取畫面, 圖表, Rectangle 的圖片

自動產生的描述

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。

1. FSM state and functionality

一張含有 文字, 圖表, 字型, 圓形 的圖片

自動產生的描述

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。

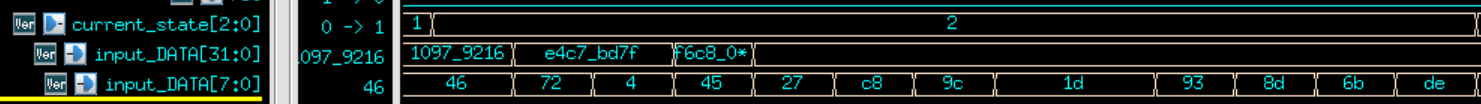
1. Waveform Result

SET\_INFO：一張含有 螢幕擷取畫面, 多媒體軟體, 多媒體, 軟體 的圖片

自動產生的描述

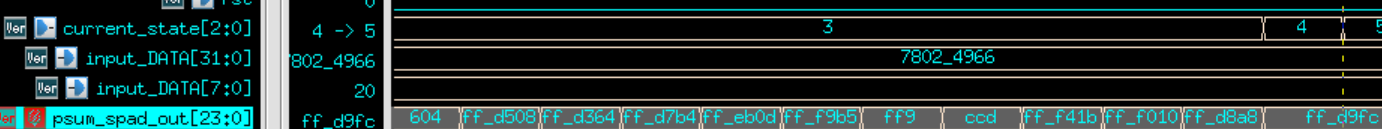
由IDLE轉到set\_info，並且讀取layer資料

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

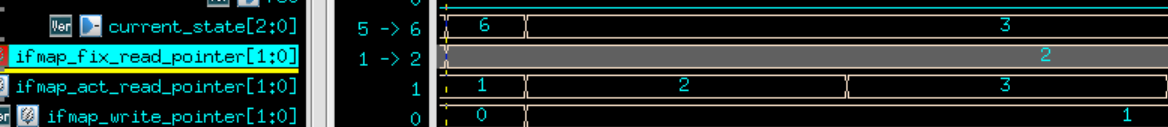
ACCUMULATE：

一張含有 文字, 螢幕擷取畫面, 字型, 設計 的圖片

自動產生的描述

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



1. 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。

一張含有 Rectangle, 正方形, 圖表, 行 的圖片

自動產生的描述

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kernal 1 channel 1 與 kernal 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kernal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：

一張含有 Rectangle, 正方形, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述一張含有 Rectangle, 正方形, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

一張含有 Rectangle, 正方形, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

1. 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 comsumption | 較小 | 較大 | 中等 |

1. 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，但是收獲也很大！