Deep Neural Network FPGA Accelerator

深度神經網路 FPGA 加速器

專業領域: 系統組
組別: A186
指導教授: 鄭桂忠 教授
組員姓名: 廖一心、林書佑、陳信翰、吳定曉

Abstract

現今人工智慧領域中，深度學習是相當重要的一環，其中為深度神經網路(deep neural network)所造成重大突破的，非卷積神經網路(CNN, Convolutional Neural Networks)莫屬。CNN 模型同時也是參考人大腦的視覺組織來建立的，此外在影像辨識上或者語音辨識上在近年來更有十分顯著的發展。然而，CNN 架構中需要用到大量的卷積運算，若是只使用 CPU 進行運算將十分耗能且延遲極高。

本專題將在 CNN 架構的基礎下，討論幾種不同資料流(dataflow)，像是權重固定(Weight stationary, WS)、輸出固定(Output Stationary, OS)等，以及運算單元(Processing Element, PE)的擺放位置，還有不同卷積技巧所形成的架構。從討論中可以比較出各架 構定性化的優缺點，而後以定量化的方法，可以得到各架構的表現，再實際量測其運算 所需的面積以及功耗。

FPGA 包含 GPU 處理器，較適合處理簡單、重複性高的工作，並搭配平行處理來大幅度的降低卷積(convolution)的運算時間。在應用端，我們將討論後最合適的架構應用至預測肺癌的 CNN model，並利用 FPGA 板來進行此模型中卷積的加速，以提高整 個 CNN model 的運算速度，如此一來才能更添加用來預測肺癌之卷積模型(CNN model)的即時性(Real time)。

Introduction

一、原理分析與名詞解釋

1.1 卷積(Convolution)

卷積神經網絡(Convolutional Neural Network)，為輸入層(input data)與特定的預訓練的權重核(filter)做卷積，擁有同樣大小的輸入層而不同深度的 input 稱為往深度方向的張數(channel)，同理對於 filter 同樣有往深度方向的張數(channel)，此外 filter 也會有一組以上去增加學習量，使輸出的 feature map 也同樣產生 channel 深度。**1.2 Unroll Loop 1~4**

在卷積的過程中我們可以將步驟拆為四種 Unroll Loop(四種不同的迴圈)，分別為以下(如圖 1)

(1) Unroll Loop 1:

計算不同位置、相同 channel 的多個 Input pixel，與不同位置、相同 channel 的多個 weight 在一個 cycle 中內積以及一個累加。

(2) Unroll Loop 2:

計算相同位置、不同 channel 的多個 Input pixel，與相同位置、不同 channel 的多個 weight 在一個 cycle 中內積以及一個累加。

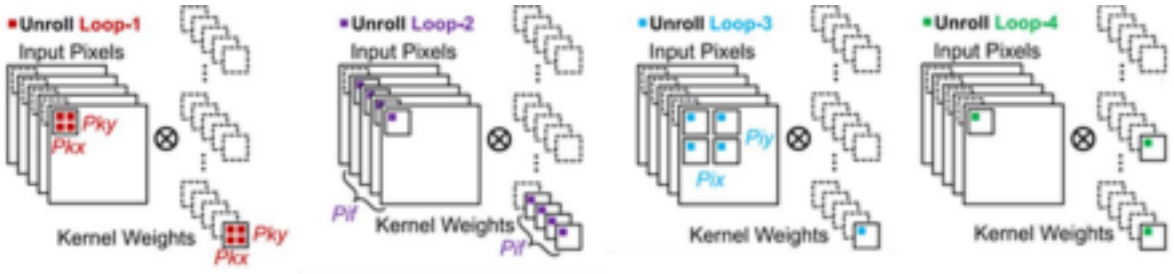
(3) Unroll Loop 3:

計算不同位置、相同 channel 的多個 Input pixel，與相同位置、相同 channel 的一個 weight 在一個 cycle 中內積以及多個累加。

(4) Unroll Loop 4:

計算相同位置、相同 channel 的一個 Input pixel，與相同位置、不同 channel 的多個 weight 在一個 cycle 中內積以及多個累加。

(圖 1 Unroll Loop 1 ~ 4)



1.3 架構比較方法

在以下架構當中，我們皆以 3 × 3 × d × k 的權重(weight)以及 n × n × d 的輸入(input)作為計算參考，其中權重與輸入皆為 4 bits，並且採取 5 種指標來比較架構差異與優缺，分別是: 平均吞吐量(Throughput)、面積效率(Area efficiency：speed/area)、讀 取資料總量、計算能耗、面積。

二、架構比較

2.1.1 Unroll Loop3

◎ 硬體架構&設計

● 架構發想

以 Unroll Loop3 的方法為出發，也就是先做完整一個深度(channel)的輸入(input)之後再往深度的方向做，但考慮到 data reuse 的部分，因此加上權重固定(Weight stationary)實作。

● 優點

Zero Padding Storage:由於架構的安排可以不用將最左與最右皆為 0 的資料存到 RAM 當中，仍可正常做運算，僅需將 input 直接給 0 即可，減少了 20% RAM 的儲存量。

Flexibility：此架構對於輸入大小的變化很彈性(flexible)，因為只需讓運算單元多算幾個週期即可完成與更大輸入大小的卷積，不用特意調整架構以及資料從 RAM 進

來的排法。

● 缺點

Partial Sum 過大：由於是先作單一深度之後，再往深度方向做，因此會有 Partial Sum 很大的缺點，因為必須先將這些值存起來後，與後面深度做完得到的結果相加才會是最終輸出。

2.1.2 Unroll Loop3 with weight stationary and input reuse

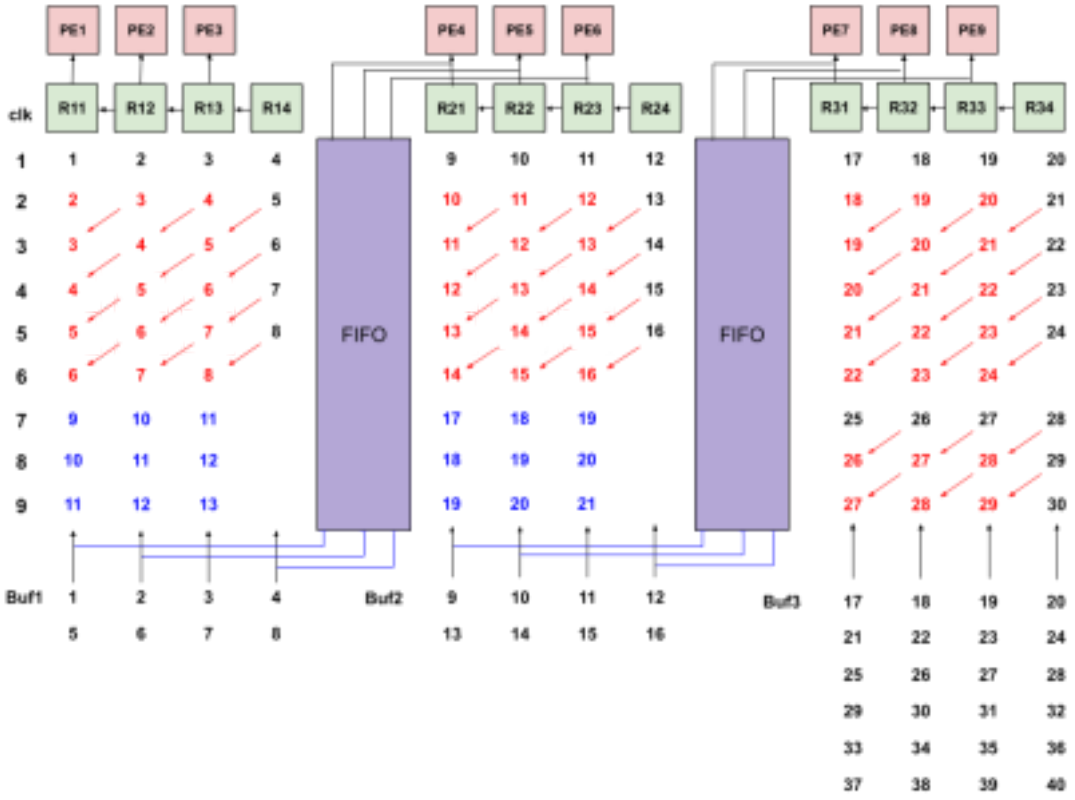
◎ 硬體架構&設計

● 架構發想

基於 2.1.1 設計，為了節省 RAM access，利用 FIFO 加上 input data reuse 的設計

● Data Flow

(圖 2 Data flow 設計圖)



除了 2.1.1 提到的優缺點外，此架構更增加了以下優缺點

● 優點

data reuse:利用 FIFO 與暫存器(register)的設計，達到 input data reuse，可以大量減少記憶體的儲存空間與讀取記憶體的次數。

● 缺點

餘數問題:當輸入層(input feature map)大小不是 4 的倍數時，緩衝器(buffer)必需額外補 0 才能維持硬體的正常運作，因此需要額外的控制訊號來解決這個問題。

2.1.3 比較

我們針對 2.1.1 與 2.1.2 以 input 16x16x14 weight 3x3x14x16 為例去做比較，數據資料如表 1。

(表 1)

	Area Efficiency$1/(s*um^2)$	讀取資料量(KB)
架構 2.1.1	0.0000130	74.484
架構 2.1.2	0.0000145	7.984

對於架構 2.1.1 與 2.1.2，雖然 Area Efficiency 是差不多的，但是由於 2.1.2 有做輸入資料重複使用的處理，因此與 RAM 拿取的資料量相對於 2.1.1 小了許多，在 RAM 的能耗上就會減少很多，不過 2.1.2 同時需要控制兩個 FIFO 與一個 partial sum buffer，因此實測出來能耗並沒有減少想像的多，且由於 2.1.2 做了資料重複使用，因此在面對 不同大小的輸入時，無法像 2.1.1 那麼彈性。

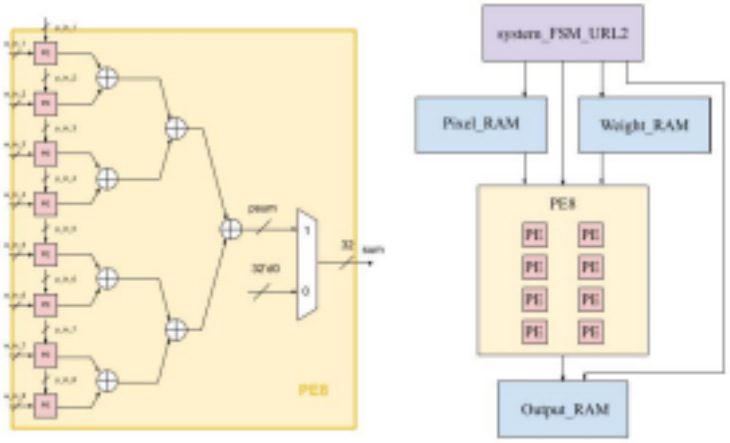
2.2.1 Unroll Loop 2

◎ 硬體架構 & 設計

● 架構發想

以 Unroll Loop 2 的方法為出發點，以「深度」方向(channel wise)作為設計的主要構想。Unroll Loop 2 的概念為，先以深度為運算的方向，從一個輸入位置(input pixel) 開始，依序往下對不同深度(channel)做運算，在計算完一個輸入位置後，接續進行下 一個輸入位置的運算，計算完整個 filter 所有的權重(input weight)後，即可得出一個輸 出位置(output pixel)的值。

(圖 3)



● 優點:

就加速器的部分來說，PE8 架構因為本身需要的資源少，使加速器的面積減少，且擁有快速的運算效率；本架構也因為 output pixel 可以很快地被輸出，不需要使用額 外

的緩衝器(buffer)來存取 partial sum。

● 缺點:

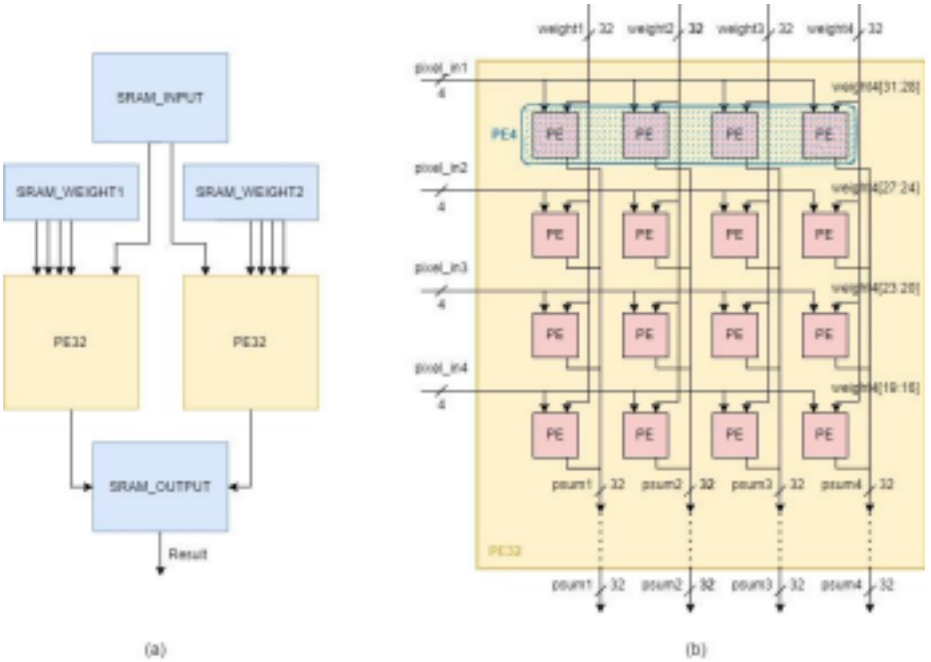
本架構沒有 data reuse，在運算時會有更多的 RAM 讀寫，造成更多的延遲(latency)與能耗；此外，因為架構太小，會造成吞吐量較低。若是輸入的層數並非為 8 的倍數 時，也會有閒置的運算單元產生。

2.2.2 Unroll Loop 2 & 4

◎ 硬體架構&設計

● 架構發想

為了克服 4.2.1 架構吞吐量(Throughput)過小的問題，以其架構為基礎，增加了 Unroll Loop 4 的技巧，同時以深度方向+卷積核(filter)方向做運算。(圖 4)



● 優點:

吞吐量(throughput)大，不需要 FIFO 減少面積的使用，加大了 PE 數量增加運算平行度(parallelism)，架構使用在任何形式的資料都很彈性(flexible)。● 缺點:

整體面積由於使用 PE 數較多所以會較大，input pixel 跟 weight 沒有重複使用，Memory Access 次數會較多，且會重複拿取相同的值。另外，若深度方向或 filter 組數沒有整除於 8，會導致閒置的 PE 產生。

2.2.3 比較

我們針對 2.2.1 與 2.2.2 去做比較，數據資料如表 2。

(表 2)

	PE(單顆)	PE_array(多顆)

	area(um^2)	power(mW)	PE 數量	area(um^2)	power(mW)
架構 2.2.1	366.2820	0.0166	8	3948.1343	0.1541
架構 2.2.2	86.8644	0.0105	32	4617.6479	1.0879

可以發現，雖然架構 2.2.2 為架構 2.2.1 的平行加大，但由於架構 2.2.1 在單顆 PE 中有架設加法器做 Output Stationary 的累加以及 MUX 去做讀寫的判斷，而架構 2.2.2 則在 PE_array(32 顆 PE)中設置 4 個累加器，一個 filter 與 input pixel 的卷積加在同一個累加器暫存器中，所以造成就算架構 2.2.1 的 PE 數較少卻沒有造成 area 以及 power 有明顯下降，或者為架構 2.2.2 的 1/4。

2.3 所有架構的比較(以運算 input 16x16x14 weight 3x3x14x16 所需之功率、面積、讀取資料量)

(表 3)

	計算能耗 (Energy)(uJ)	面積 (Area)(um^2)	讀取資料量 (KB)	面積效率 (Area Efficiency) $1/(s*um^2)$	平均吞吐量 (unit:1/cycles)
2.1.1	0.295994496	170308	74.484	0.0000130	2.214u
2.1.2	0.20737472	174242	7.984	0.0000145	2.531u
2.2.1	0.700011648	149355	441.000	0.0000148	2.215u
2.2.2	0.362389104	150025	248.063	0.0000147	2.214u

(1) 計算能耗：計算能耗由讀取資料的能耗、PE 的能耗以及 FIFO 的能耗所組成。由於 2.1.1 與 2.1.2 架構相似，但 2.1.1 沒有做 data reuse，讀取資料量多，因此計算能耗 較 2.1.2 大。然而，2.1 部分雖然有以暫存器陣列(Register File)所組成的 FIFO，但 因為讀

取資料量甚少，因此計算能耗仍比 2.2 部分的少。

(2) 面積：在 2.1 的部分，因為有 Partial sum buffer，因此面積較 2.2 的大，而 2.1.2 又 比 2.1.1 再多兩個 FIFO，所以面積略大。在 2.2 的部分，由於 2.2.1 在單顆運算單 元內，做較多運算，因此若與 2.2.2 有相同 PE 數的話(64 顆)，面積會比 2.2 還大。

(3) 讀取資料量：在 2.1 的部分，由於 2.1.2 有做重複使用，較 2.1.1 少讀了很多資料(10 倍)。2.2 的部分，由於 2.2.2 有做平行處理，等效上讀取的資料量與 2.2.1 相比少了 2 倍。而若拿 2.1 與 2.2 比較，由於 2.1 部分的都有採取權重固定(Weight Stationary) 的手法，在讀取資料量上相差至少 3 倍以上。

(4) 面積效率：最後使用平均吞吐量來計算，因此面積效率差不多。

(5) 平均吞吐量：由於計算平均吞吐量時，有除上各別有的 PE 數量，以示公平。2.1.2 僅運用 9 個 PE，但計算 cycle 最少，因此平均吞吐量高。

Feedback

大力感謝鄭桂忠教授給我們這個機會接觸到有關於深度學習加速器的技巧，並提供實驗室的資源給我們做研究。感謝助教揚翰學長在一開始，帶著青澀懵懂的我們看著各種不同深度神經網路並且思考其中的原理，以及助教禹樵學長在之後持續給了我們很多在實作方面的建議，不停協助我們架構方面的問題。在這其中我們不斷思考，我真的要在 FPGA 上做出卷積這個運算嗎？怎麼做……等問題。

一路到下學期，變更加老練成熟的我們將這些原理，打成硬體描述語言，再透過 不同的模擬工具，De 了千千萬萬的 BUG 之後，雖然可能還要再 De 千千萬萬個 BUG，但有了這股研究精神，每次看波形圖跑出來的值都像在對獎一樣，希望自己的答案會是對的。百般波折後，最終我們得到了正確的答案，那一刻可謂欣喜若狂呀！ 很興奮的打開了工具給我們的報告，看看自己設計出來的架構究竟有多少的能耗與面 積，之後苦思著如何改善、如何畫出更好的運算單元，與夥伴們熱烈的討論到三更半 夜早已司空見慣。大家做出來的加速器架構透過比較之後，並實作到 CNN 模型上，才不會有只是看著波形圖紙上談兵而已。在實作至 FPGA 時，對於大一邏輯設計不懂的地方有更進一步的了解，像是 IP、PS 與 PL 等的功能，讓我們可以將原本以為只有 軟體用的 jupyter notebook，拿去控制硬體，達到加速的效果。

同時這個過程，讓我們都了解到何謂數位 IC 設計的流程，其實不是只有波形圖看一看，行為對了就好，還必須考慮到能否合成，而其中也有很重要的一件事，即是 coding style，不同的打法會合成出不同的東西，雖然表示的邏輯可能一樣，但面積與 能耗可能有所不同。最後再次感謝鄭桂忠教授與禹樵學長的幫助！