HW5: DSA

楊永琪, 109654020

***Abstract*— 此為作業五報告, 嘗試結合 floating point IP 加速 MLP 推論計算。**

I. Introduction

這次作業為在 Aquila 上跑一個辨識來自 MNIST dataset 中手寫數字的 MLP 模型。並除了用純軟體跑外, 也嘗試結合 Vivado 內提供的 floating point IP , 客製一個在 Aquila 外的加速電路, 並進行資料傳輸與計算時間的分析。

以下將先介紹這次嘗試的幾個方式, 後續會進行資料讀取以及計算時間的分析與最後的總結。

II. Implementation

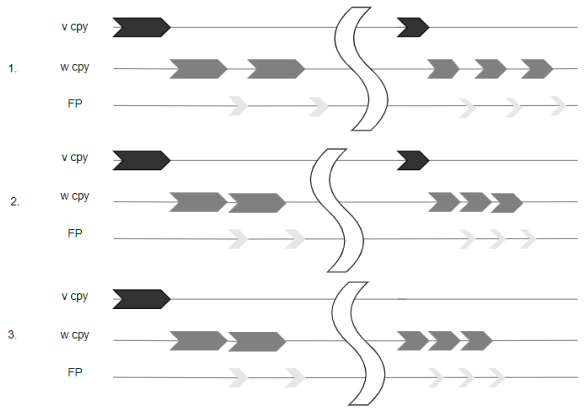

Fig. 1. Flow Chart of MLP Inference Using DSA

上圖描述了這次作業嘗試實作與改良的方法, 主要分別有以下三種與其他的嘗試。

A. One Buffer

第一種為單純把 vector 與 weight 的 data 用 while 迴圈複製到加速電路的 Block RAM, 資料複製完後將觸發電路做計算, Aquila 則會 busy waiting 等待 FP IP 計算結果。

B. Ping Pong Buffer

第二種為嘗試在 weight 上使用 ping-pong buffer, 觸發加速電路做計算後, 也開始傳輸下一筆 weight 的資料, 等資料傳輸完並電路計算完後, 便直接開始下一回的計算, 使得在做計算的時間可以充分被利用。

然而, 用 ILA 分析後發現, 由於每次資料複製的時間與計算的時間相差許多, 計算下來平均約差 20 倍左右, 導致後續分析時發現在整體的加速效果上沒有較顯著的提升。

C. Store Calculated Weight to Vector Buffer

第三種則是在每次計算完時, 將計算結果直接存在加速電路的 vector buffer 內 (如圖二), 目的是希望能減少一些 DRAM access 造成的延遲。在 DSA 電路內則增加 base address 與 top address 的設定, 告知電路該從何讀取資料, 減少第二層的 vector copy。然而因為後續兩層的 neuron 個數比第一層少許多, 此種作法也沒有對於速度的提升也沒有顯著的效果。

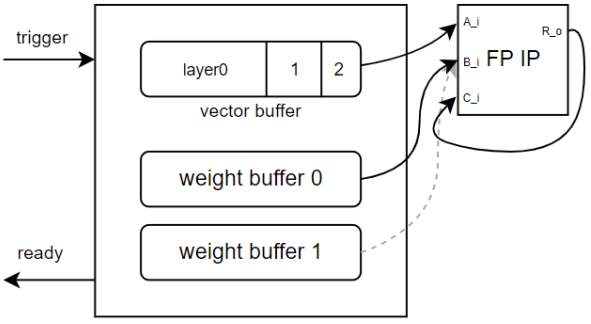

Fig. 2. Structure of DSA

D. Reduce write latency & FP IP latency

原先在使用 FP IP 時因為一直遇到 exceed timing violation 的問題困擾了很久, 後來使 Blocking Mode 的 FP IP 並將 latency 改成 3 倍才解決這個問題, 並會依序等每次計算完後在把 IP 的 result 傳到 C channel 並進行下一次的計算。

後續與老師討論後發現其實為進去的 c data 不用按照順序, 最後改成 NonBlocking Mode 並使用 latency = 3, 在每次計算時會額外花 8 個 clock cycles 累加最後的三個 output, 整體計算的時間如表一所示, 計算 100 張圖片的總時間約為 93 毫秒。

由於後續分析發現花最多時間的其實是在做資料搬移的動作, 最後也嘗試排除在寫入 DSA (exe_we = 1) 時的資料時要 stall 一個 clk cycle 的狀況。

TABLE I. Total FP IP Latency

Latency	***Original***	***After***
clk cycles	11,451,000	3,857,600
ms	274	93

III. Analysis

表二為原先使用軟體做模型推論的分析, 其中總 latency 為 22 秒左右, 其中關於 load, store 指令約佔 6.1 秒, 因計算乘法延遲時間約佔 1.6 秒, 但由於程式較為複雜, 還有一些流程的控制等較難進行進一步的分析。

TABLE II. ORIGINAL ANALYSIS (MS)

total latency	*load / store*	*exe stall*
22,038	6,129	1,635

表三則為針對客製化加速電路的分析, 在 DSA 內 Vector 與 Weight 的大小皆為 4KB, data copy 的計算為在進入自訂 dsa_cpy (while copy word) 與 return 之間的 clock cycle 數。

其中表格裡的 pure copy 為使用按照原先程式的流程 (one buffer) 單純將 vector 與 weight 複製到 DSA buffer 裡, 其總時長約 1,916 ms。

One buffer 則是先前提到使用一個 vector 與 一個 weight buffer 去存資料並計算的方式, 在還未修改 FP IP 之前 latency 為 2,223 ms, 修改後為 2,039 ms, 其中 data copy 約占總延遲的 93.03%, 真正計算的時間只占 4.54%。

使用 ping-pong buffer 後, 由於資料傳輸與計算會同時進行, 而資料傳輸時間較長, 故在修改 FP IP 後加快的速度提升較不明顯, 最後延遲為 1,953 ms。

在嘗試 ping-pong buffer 後, 我也嘗試將計算後的結果直接存在 vector buffer 裡, 但效果也沒有明顯進步, 約快 5 ms。最後嘗試在 aquila 中寫入 DSA 時 stall 1 cycle 的延遲去掉, 總延遲約 1,879 ms。

TABLE III. DSA ANALYSIS

	Latency			
	total(ms)	*FP (ms,%)*	*data copy (ms)*	*data copy (%)*
pure copy	1,916	x	x	x
One Buffer	2,223	12.37%	1,901 ms	85.52%
One Buffer FP	2,039	4.54%	1,897 ms	93.03%
Ping Pong	1,955	274 ms	1,899 ms	97.14%
Ping Pong FP	1,953	93 ms	1,899 ms	97.23%
Ping Pong (store to vector)	1,948	93 ms	1,897 ms	97.37%
Ping Pong v (reduce w lat)	1,879	93 ms	1,829 ms	97.34%

IV. CONCLUSION

在分析完後, 發現造成延遲最關鍵的因素還是資料讀取跟搬移的時間, 使用 ping-pong buffer 後單純計算時間應該只有在每個 layer 最後一次的計算, 估計約為 1.9 ms, 占不到總時間的 1%。