國立清華大學 電機工程學系

實作專題研究摘要

Spiking Neuron Network hardware accelerator available for visual obstacle avoidance in autonomous vehicles

可應用於自駕車視覺避障的脈衝神經網路硬體加速器

專題領域：系統領域專題

組　　別： B469

指導教授：鄭桂忠 教授

組員姓名：郭霆宇、黃政倫

研究期間：113年 1月 30日 至 113年 12月 1日止，共 10 個月

Department of Electrical Engineering, National Tsing Hua University
Special Topic on Implementation Summary

Spiking Neuron Network hardware accelerator for visual obstacle avoidance in autonomous vehicles

可應用於自駕車視覺避障的脈衝神經網路硬體加速器

Major Category: System design

Group Number: B469

Advisor: Professor Kea-Tiong Tang

Members: Ting-Yu Kuo, Jeng-Luen Huang

Research Period: From 2024-01/30 to 2024/12/01.

Abstract

Spiking Neural Networks (SNNs) are characterized by their use of bio-inspired neurons, replacing the nonlinear layers in traditional neural networks. This feature simplifies hardware implementation compared to traditional nonlinear equations and provides superior energy efficiency. This project implements an existing SNN as hardware, targeting its application in a hardware accelerator for visual obstacle avoidance in autonomous vehicles. The neural network comprises five layers, each consisting of multiple integrate-and-fire (I-IQIF) neurons that simulate basic neuronal behavior through linear equations. Membrane potentials and other parameters are quantized to reduce computation overhead while preserving essential characteristics. Connections within the neural network include fully connected layers and residual connections. Visual input signals, such as speed, angular velocity, and optical flow, are processed to predict and generate spike patterns as outputs.

The network was pre-trained in Python to obtain parameters like synaptic weights, which were subsequently implemented in hardware using Verilog and validated through software simulations. To manage the five-layer structure with varying neuron counts, neurons within each layer were grouped. During inference, a pipeline approach was employed, with computation units and SRAM channels serving one neuron group at a time. This reduced idle time and maximized time efficiency. Read-write conflicts were avoided using ping-pong SRAM operations. Additionally, during each FC layer inference cycle, weight pruning was applied in advance to adapt to the current conditions, reducing power consumption and processing time. This approach enabled the concurrent invocation of multiple circuits and extensive reuse of circuits over time intervals, significantly reducing the hardware area. Ultimately, the implemented hardware supports real-time offline inference, successfully serving as an accelerator for obstacle avoidance.

摘要

脈衝神經網路(Spiking Neuron Network, 以下簡稱 SNN)的特點在於仿生神經元的使用，並取代傳統神經網路中的非線性層。這樣的優勢在於神經元的硬體實作相較於傳統的非線性方程更簡易，且有更高的能耗表現。本專題將既存的 SNN，進行硬體的建立。該神經網路是用於自走車視覺避障的加速器，一共有五層 layer，每層為多個二次方程累積放電脈衝(I-IQIF)神經元，設計其線性方程式以模擬神經元基本行為，再量化膜電位等參數，降低計算消耗卻不失其特性。神經群的連結包含全連接層(Fully connected)、殘差連接（Residual Connection）等。電腦輸入的視覺訊號有速度、角速度、光流等資訊，以此進行預測，生成脈衝 spike pattern 作為輸出。

神經網路的訓練是預先利用 python 進行，獲得神經權重等參數，最後透過 verilog 來實作神經網路的硬體架構，並以軟體驗證輸出。對於五層不同神經元數量的 layers，我們決定以神經元組來處理，將同一層間的神經元分組。在推論的過程中以 pipeline 來優化，計算單位和 SRAM 通道在同一時間為一組神經元服務，減少單元互相等待的時間，使時間利用效率最大化；在過程中產生的讀寫衝突，使用了乒乓操作(ping-pong SRAM) 進行避免。每一輪推論 FC layer 時，為降低功耗和時間，會提前對於當下的情況進行權重枝剪。因此不僅能同時調用多種電路，也利用時間區隔大幅重複使用電路，減少面積。最終硬體能負責離線實時推演的部分，完成避障功能的加速器。

1. Introduction

1-1background of SNN

脈衝神經網路的特點在於仿生神經元的使用。特點在於仿生神經元的輸出不單與當前的突觸輸入值相關，其輸出的值是依據在於神經元當前的膜電位(membrane potential)與閥值電位(threshold voltage)的相對關係進行計算，並且因此能充當了神經網路中非線性的提供者。其中膜電位的更新主要由突觸輸入充電、隨著時間的漏電、發射與重置(fire-and-reset)等仿生行為主導，具體設定和參數隨著架構不同而改變；而傳統神經網路(Additional Neuron Network, 簡稱 ANN)的輸出只和當前的突觸輸入值有關，換句話說在神經元的輸出對時間是否具有相依性具有差別，同時 ANN 需要加入額外的非線性函數提供非線性關係。。

脈衝神經網路的研發主要呼應著兩個訴求。其一，在生物大腦相對神經網路的巨大優勢的前提下，SNN 利用更擬真的仿生手段提升架構的精確度和擴大複雜度。其二，發揮仿生神經元的易於硬體時做的優勢，製作低功耗的神經網路硬體加速器。在前者的研究中，包含根據神經學建立更仔細的放電模型分類，或是利用神經的記憶特性，在神經網路製造回授路徑。除了完全模仿生物大腦，在一些研究中脈衝神經網路的線性連接架構也會借鑑深度神經網路的研究成果，例如殘差連接、剪枝等技巧提升架構效能。後者研究啟發的在於，隨著物聯網等應用的興起，神經網路也逐漸普及在邊緣系統上，因此神經網路在硬體實作的成本，包含晶片面積和能耗，受到更多重視。而在計算神經元的輸出時，因為仿生神經元自帶非線性特質，可以用簡單多項式的代價計算免去傳統神經元的非線性層的複雜計算，例如常見的 softmax 函數使用的指數和除法計算，都會浪費大量的硬體資源和計算力。特別是在一些較小型的模型中，將既有的神經網路的神經元替換成仿生神經元，就能獲得具有低功耗且易於硬體時做的 SNN，並只犧牲些微的準確率。這類的 SNN 的網路訓練，通常是藉由數學模型模擬神經元的非線性行為，在使用傳統的反向傳播法進行訓練，獲得神經元之間的突觸的權重，神經元的權重等數值，在實做到硬體電路上。本專題就是此例的應用。

1-2motivation of the accelerator

傳統基於立體視覺的深度估計法，需要兩個精確校準的相機，並且基於視差計算深度資訊，但該法受到較的成本（兩個相機）和物理的限制。在小型自走車等機具上，兩相機的間距受限，造成的視差有限。因此，使用 2D 的輸入影像搭配演算法進行空間分析是個可行方案。本專題啟發自 Chen-Fu Yeh, Chao-Yang Tang 等人的研究[1]，該研究專注於解決單影像的深度估計問題，透過開發加速的

神經網路，最小化計算複雜性，同時保持適合微型車輛障礙物檢測的適當精度。合作實驗室提供了將此 ANN 加速器轉換並訓練完成的 SNN 版本。本專題的實驗內容僅集中於將改編後的 SNN 加速器從軟體實作至硬體描述語言，並使用 synopsys 設計編譯器(Design Compiler)合成出，並測試結果是否有達到我們在設計時期望的目標。

2. Research Methodology

2-1 data preprocessing

權重、神經元資訊等數值在軟體學習得到後，放大$\sqrt{127}$倍後，將上下限限縮在-128 到 127 之間後取整數，在硬體中以 8 位元二補數表示，並儲存在相應的記憶體位置等待讀寫。取整數的目的在於讓所有數值都是以整數進行運算，因此能夠簡化計算複雜度，並且確定資料長度。位元數固定後，記憶體位置也能清楚規劃計算。事先放大$\sqrt{127}$倍的目的為保留數字精準度，避免取整影響計算的精準度。

2-2 layer introduction

- 速度層（Velocity Layer）、旋轉層（Rotation Layer）：

  前者接收 6 位元速度數據，並轉化為 64 位元作為輸入，包含 64 個神經元。後者接收 6 位元的角速度數據，擴展至 256 位元，包含 256 個神經元。由於不牽涉上一級 spike 發射，因此可以以資料預處理的方式計算完給該層的神經群，減少硬體計算。

- 去旋轉層（De-Rotation Layer）：

  擁有兩類輸入，前一級神經群的 spike 以及外部的光流（Optical Flow）輸入。外部輸入因同上理由，其對神經群輸入以預處理方式完成。而 spike 的輸入則需等到旋轉層的神經群完成膜電位計算後才能進行。

- 縮放層（Scalar Layer）：

  只包含前一級神經群的 256 個 spike，然而一個神經元接收 4 個來源的 spike 進行統合，因此只有 64 個神經元。

- 深度層（Depth Layer）：

  輸入為 128 個 spike，依序為速度層和縮放層提供各 64 個。此外對神經群的輸入是全連接的方式(Fully connected)。

2-3 output pattern

最終輸出是經過深度層（Depth Layer）的預測，生成 64 組 8bits 自走車避障可用的深度信息。SNN 因膜電位的記憶特性，對多次、連續的輸入以及其順序有很大的相依性。因此每一批的輸入的資訊會重複輸入 100 次，用來累積膜電位等訊息，強化對訊息的萃取，而最後的輸出是一百次循環的 Depth Layer 輸出脈衝之加總。

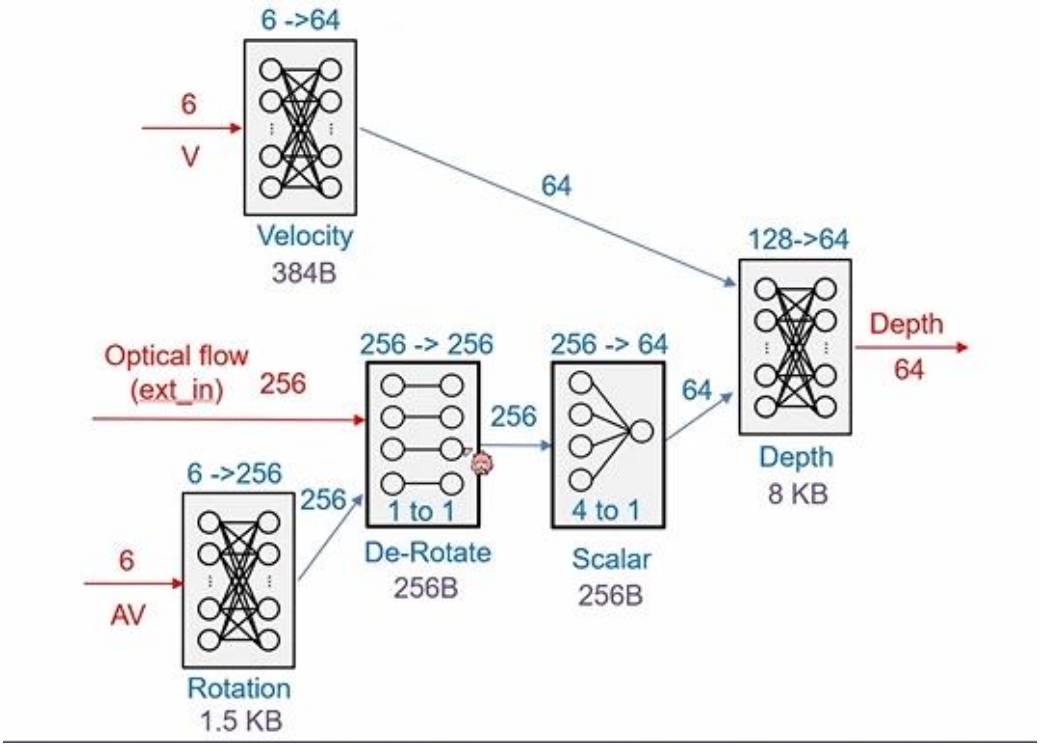

Figure 1: The Schema of the SNN

2-4 Design overview

在設計時因為考慮到 depth layer 的獨特性和分工，我們將整個架構分為兩個部分討論。前面的四個 layer 不包括全連接層的運算，速度層和旋轉層的結構雖然在圖片中是全連接層結構，但在前處理時，軟體已經負責將外部輸入和權重處理後得出神經元的純輸入，所以實作的過程會和去旋轉、縮放層等偏稀疏的 layer 處理方法相似。在 depth 的部分，因為關係到實際的全連接層處理，我們將原先架構加入額外階段來實作。

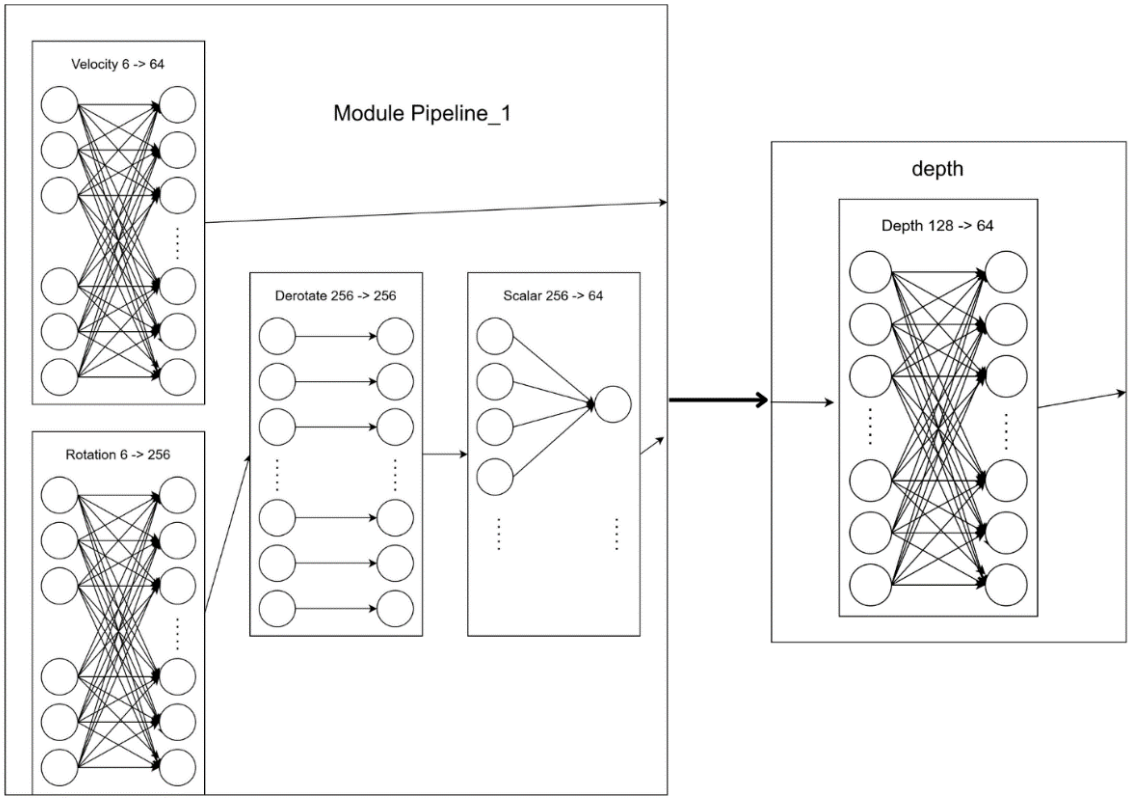

Figure 2: RTL module separation

2-5 data flow of computation

2-5.1 memory access

根據靜態隨機存取記憶體(SRAM)讀寫速度限制，在每一個 cycle 能讀取 16 個權重和 16 組神經元的資訊。此外，記憶體的讀取需要向記憶體硬體模塊提供目的位址，並且在 2 個 cycles 之後才會拿到記憶體該位址的資料。同理，寫入的過程需要提供地址以及寫入資訊，接著在 2 個 cycles 後才會完成寫入。在以上情況下，為了更好說明我們設計的演算法，會先省略這兩個 cycle 的等待時間的討論。在說明完成後，會在時間分析時，算出多了這兩個 sram delay cycle 的結果。

2-5.2 pipeline phase

將電路行為分成 4 階段，每階段代表使用不同的電路硬體模塊，資料間彼此用 flip-flop 間隔。在 pipeline 劃分使用時間後，在不增加硬體數量下，運算時間效率理論上達到 4 倍，以下統稱四個階段總合為一個操作(以下稱 instruction)。在一個 instruction 中有 4 個階段：

- 讀取，計算並發布下一個需要的記憶體位址
- 進行權重與脈衝的計算，調用計算神經元輸入的單元
- 神經元的數值計算，調用計算神經元的單元
- 寫入

再根據 16 格資料一組的運算規則，可以得到以下流程圖。

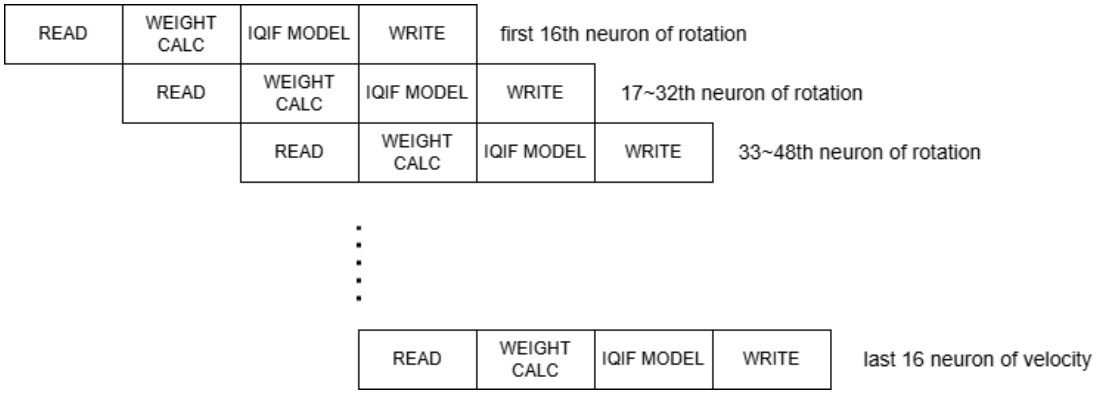

Figure 3: pipeline stage explanation

對於 rotation、de-rotate、scalar、velocity(依序)的處理。雖然 scalar 也可以直接放進這個相同格式，但由於 scalar 為 4 to 1 的架構，需要稍微修改流程。

如下圖，在其他的 layer 中，pipeline 中的 instruction 是以接收輸入的 16 個神

經元為單位依序執行，但在 scalar 的 4 to 1 中，我們設計會以四個 instruction 作為每 16 個接受輸入的神經元計算的週期。請參考下兩張圖，圖(左)第一行指令會讀入圖()上層神經元 1~16 並進行累加，第二行指令會讀入 65~80，第三行讀入 129~145，第四行讀入 193~209 後，會用前面四個 instruction 累加出來的輸入開始 IQIF model 的電位累加行為，最後更新下層前 16 個的神經元電位，這樣的一組處理需要重複 4 次來完成總共 64 個神經元的 scalar layer。

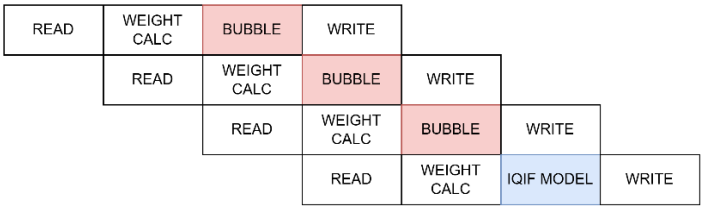

Figure 4: actual pipeline stage

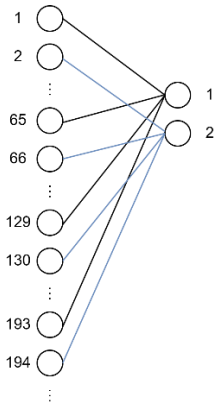

Figure 5: neuron and synapse of depth layer

最後，在深度層中，因其連接方式是全連接(Fully connected)，在相同的 SRAM 頻寬下，時間需求較大。特別是在 input spike = 0 的情況，其調用的權重對膜電位沒有影響，會浪費時間、能耗等資源。因此對於每一次計算，額外花費資源評估其必須性具有較大的效益。

2-5.3 pruning and simplifying of depth layer

前面提過我們可以在一個 cycle 內做 16 個權重乘法計算，depth layer 由於單一 spike 的輸入會動用 64 個權重，代表 depth layer 中這 128 個 spike input，每個 spike 的權重處理需要 4 個 cycles 進行計算，相當費時。因此在準備權重位址階段，事先檢查總共接收到的 spike 不為零的位置與數量，在依此動態發送權重讀取要求。

檢查 128 個輸入 spike 時，需先等待速度層、旋轉層的 spike 輸出完成。spike 儲存在一個 128bits 的訊號中，此時花費 1clk 將所有非零 spike 找出，記錄其數量和編號，總計花費 6 個 cycles。接下來，每 4 個 cycles 完成一個 spike 相連的權重計算與調用，即這個 spike 對 64 個 neuron 權重的影響量。然而，因為深度層是 Fully connected 的結構，必須等待所有 spike 對 neuron 的影響加總完畢後，也就是 4n 個 cycles 後，才能對 neuron 的電位進行更新。又其 64 個 neuron 需花費 4 個 cycles 更新，因此總共花費 4n+4 個 cycles。比起原本的算法，4*128 + 4，可以省下大量 cycle 數。

2-5.4 confliction of memory read/write

神經元的計算完成會得到 spike 的值以及更新後的膜電位，需要再寫入記憶體，供下一輪該神經元之計算用。因此在規劃上，儲存神經元膜電位的記憶體同一時間點會有讀和寫的需求。解決方式是用 ping-pong SRAM，亦即使用兩個「完全相同」的記憶體來分割存取資料。因為本 pipeline 階段數是偶數，因此只要將奇數、偶數神經元膜電位分開在兩個記憶體，就能避免衝突，如圖，可以看見在同一時間(同一鉛直線)上沒有衝突。

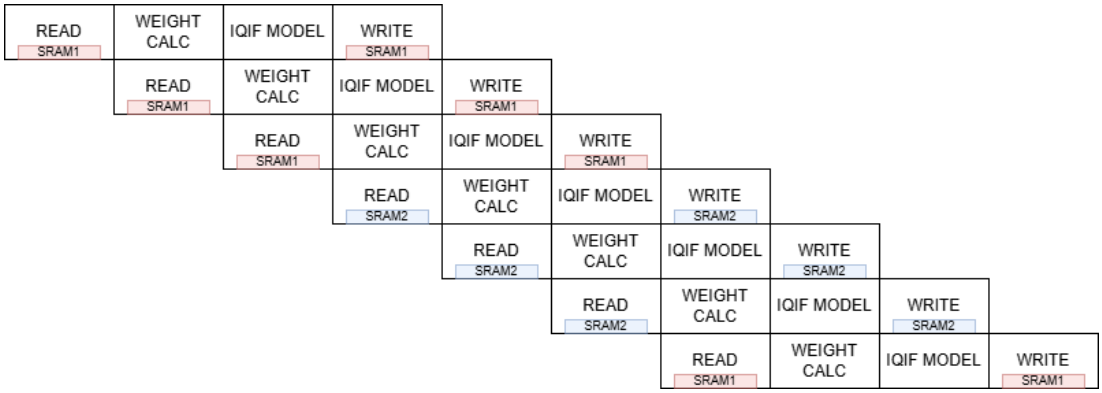

Figure 6: ping-pong SRAM(red for SRAM1; blue for SRAM2)

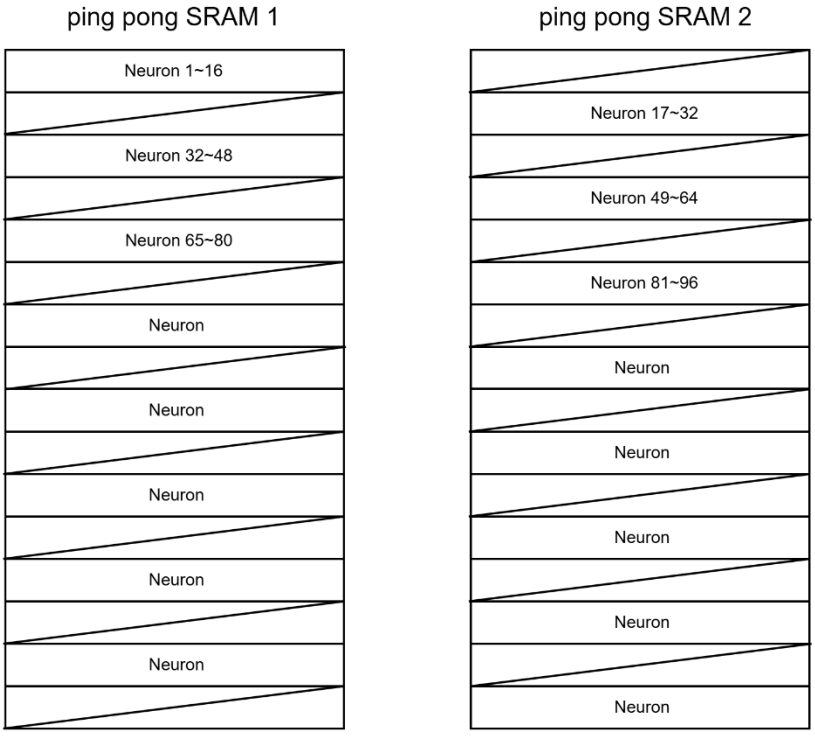

Figure 7: the location of neurons of ping-pong SRAM

2-5.5 worse case analyze

在 pipeline 的堆疊下，以下列出深度層以外的每個神經層進行計算需要的 cycle 數：

Rotation: 256 個神經元 ➞ 16 個 instructions

De-rotate: 256 個神經元 ➞ 16 個 instructions

Scalar: 實際處理 16 個 instructions

Velocity: 64 個神經元 ➞ 4 個 instructions

→前半部分總共需要 52 個 instructions。

後半部分的 depth layer 中，如上所述，統整如下：

spikes 的等待與統計➞ 6 個 cycles

weight 的調用和計算➞ 4n+4 個 cycles(n 為最後層偵測到的 spike 數量)

輸入結果傳送到 pipeline_1 進行運算➞64 個神經元/16 = 4instructions

→後半部分總共需要 4 instructions + 4n + 10 cycles。

最後基於 2-5.1 memory access 提及過的 2 cycle delay 加上後，我們在 pipeline 中的 instruction 加入了 bubble，如下圖所示

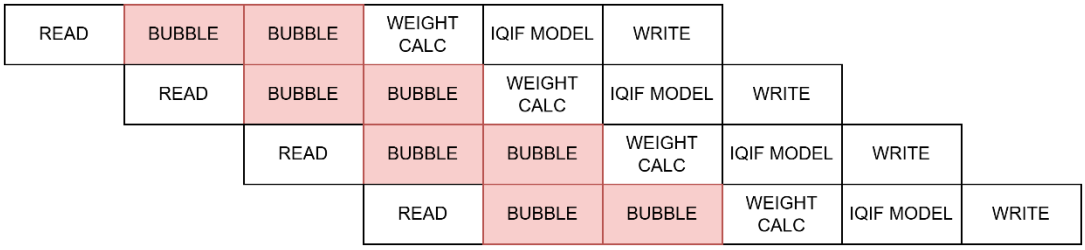

Figure 8: pipeline stage adding bubbles

因此每一輪完整的推論過程消耗 64+4n 個 cycles，$0 \leq n \leq 128$。在實作上，cycles 頻率大約是 100MHz 的數量級，因此每筆資料，在 100 輪運算後，最差狀況(worse case)費時大約在 1 毫秒之內，具有實際運用的可能性。

2-6 design detail

2-6.1 weight processing unit

權重計算單元(以下簡稱 WPU)用於計算前一層的輸入 spike 與權重之互動，最後輸出神經元的輸入。因為每層每個 WPU 中，spike 與權重互動方式皆不同，因此若要在所有層之中使用同一副電路計算，將會需要 256*256 的全連接電路才能兼容。計算下，除去前處理取代實時計算的速度層和轉動層，為剩下三個層客製化製作 WPU 將會更節省 flip-flop 數量。此外，WPU 閒置時可以利用 enable 訊號關閉其功能，減少耗能。

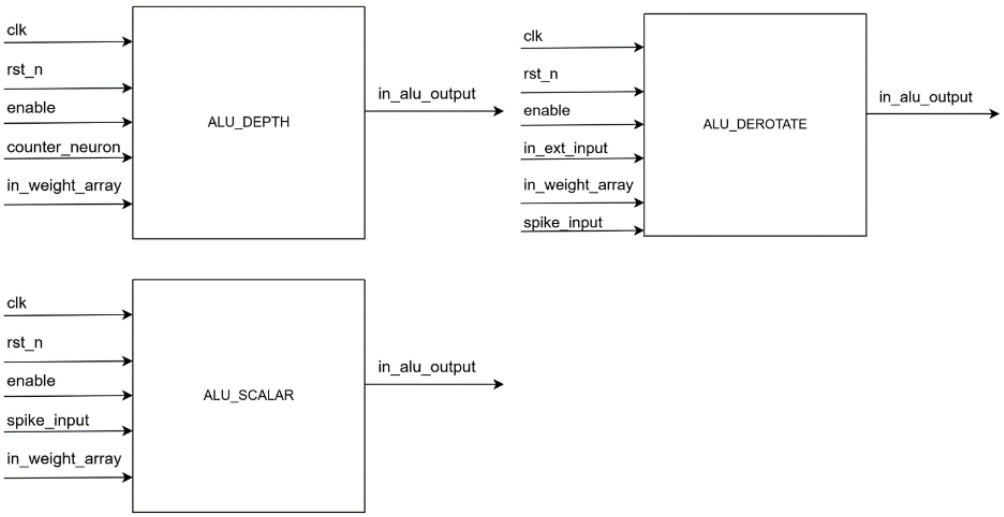

Figure 9: input and output of WPU(ALU)

2-6.2 neuron processing unit

神經元處理單元 NPU(以下簡稱 iqif)，每次計算 16 組神經元資料，而所有層中神經群的神經元數量都是 16 的倍數，所以透過 pipeline 安排只要有一副電路就能分時為所有層計算神經元。在神經元的設計上，採用二次方程累積放電脈衝(I-QIF)為基本架構，根據當前膜電位和輸入電壓，計算更新後的膜電位和發射的 spike 值。膜電位的單位時間變化有兩種行為：在當前電位小於預測閥值(predict-threshold)，膜電位會有漏電。而當前膜電位超越預測閥值時，神經元開始充電，準備發射 spike。在兩個情況下，輸入電位都會累積在膜電位上。以下是膜電位計算公式：

$$\begin{cases} \Delta V_{mem} = -\dfrac{a(V_{mem} - rest)}{8} + input\,, for\ V_{mem} < pde\_th \\ \Delta V_{mem} = \dfrac{b(V_{mem} - threshold)}{8} + input\,, for\ V_{mem} \geq pde\_th \end{cases}$$

其中

$$pde_{th} = \frac{a \times rest + b \times threshold}{a + b}$$

Spike 的發射條件為下，其值為一位元的 0 或 1，同時神經元發射後會重製膜電位，模擬生物神經元的 integrate-and-fire 行為。

$if\ mem < V_{peak}$

$$\begin{cases} spike = 0 \\ V_{mem}(t) = V_{mem}(t-1) + \Delta V_{mem} \end{cases}$$

$if\ mem \geq V_{peak}$

$$\begin{cases} spike = 1 \\ V_{mem}(t) = rest \end{cases}$$

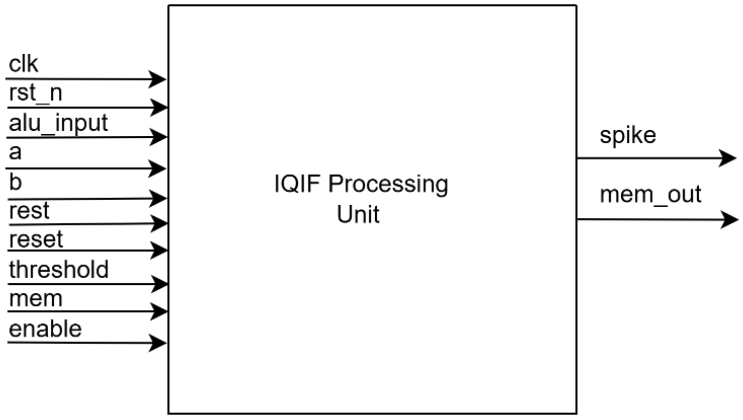

Figure 10: input and output of IQIF unit

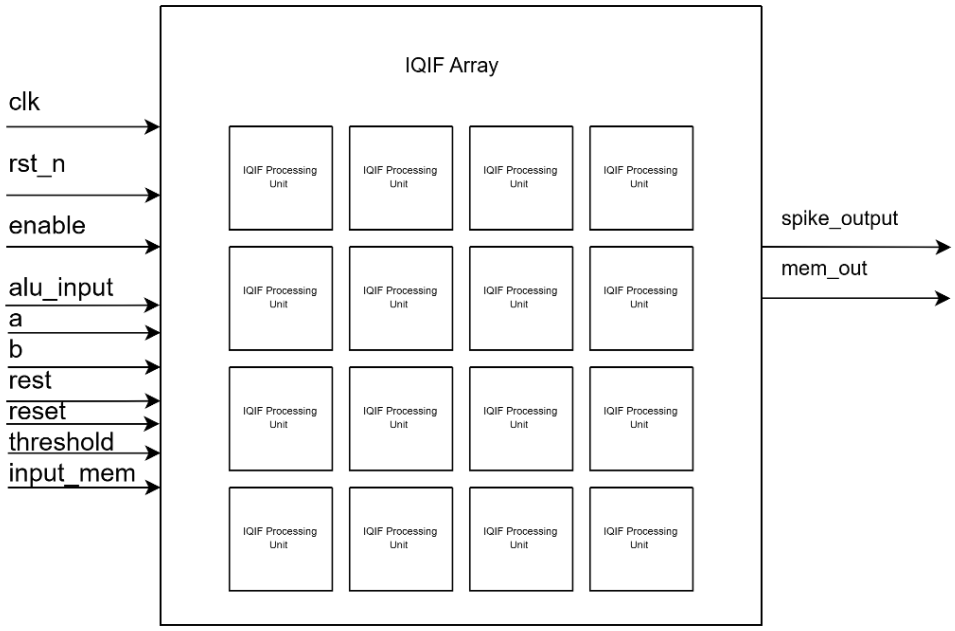

Figure 11: input and output of NPU(IQIF Array)

2-6.3 overall design of first part

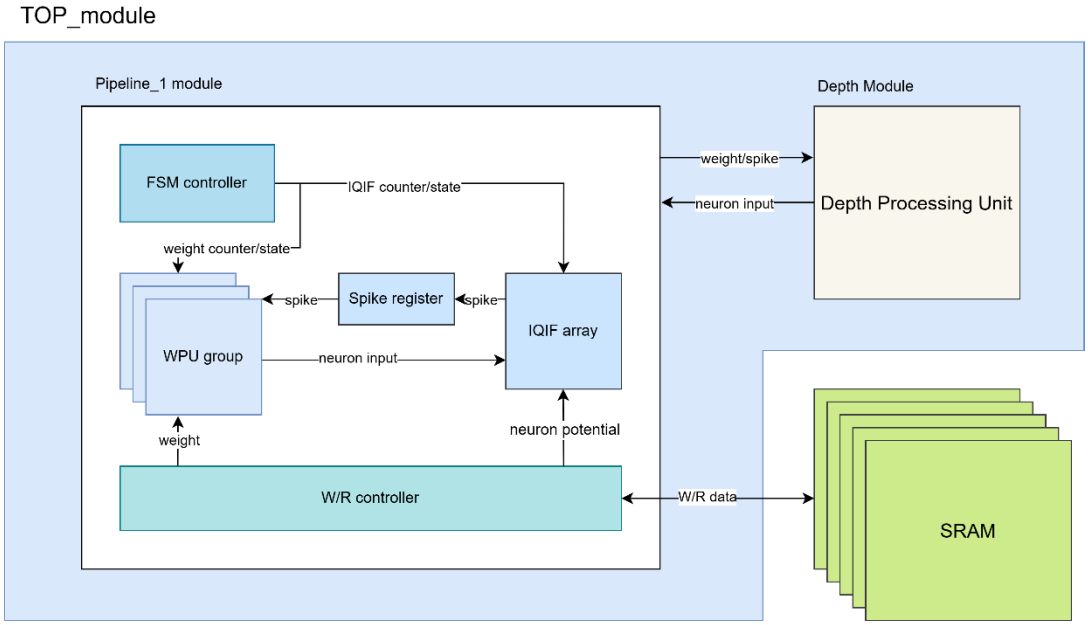

Figure 12: schematic of dataflow

3. Experimental Result

3-1 output verification

為檢查硬體是否被正確建立，我們使用軟體版本在 100 輪中不同神經層的輸出，對應硬體在各個時刻的輸出波形圖，確認過程中輸出一致。最後確認經過 100 輪後的加速器給出的 spike pattern，硬體輸出與軟體一致，顯示計算功能正常。

3-2 cycles optimization

本次實作著重在對於 cycle 數的加速實作。以下是我們在各個階段的 100 timesteps 所需要的 cycle 數

a. single cycle architecture:
   對於每組十六個神經元的計算都分配獨立 6 個 cycle 來執行，總共會花費 100(526 + 1284 + 4) = 82800 cycles。
b. pipeline architecture:
   在使用了對於不同 layer 去設計出的 WPU 後，實作出了前四個 layers 可以一起使用的 pipeline 架構後，此時總共花費 100（5+52 + 1284 + 4）= 57300 cycles。
c. Pre depth processing:
   對於最後層 depth 的輸入 spike 檢查後，對於前面出現的 128 4 cycles 可以以這個方法減少沒出現 spike 而浪費的 cycles，最後的實驗結果，在 100timesteps 只花費了 12754 cycles。

3-3 performance on Cadence GPDK 45 nm

我們將完成的 RTL 以 cadence 的合成工具，以 slow, vdd1v2, GPDK 45 nm 的製程進行合成。合成的結果分別以整體、Figure.12 中展示的 module pipeline.1 和 module depth 進行討論，獲得結果如下：

	Area (um^2)	Timing (ns)
total	140,597	22
module pipeline.1	37,346	4
module depth	103.251	22

Table.1 synthesis result of area and timing

從 Figure.2 中可以看到，Depth Layer 因為是全連接層，本身突觸數量高於其他層的總和，因此相對其他層所需要的加法乘法器和暫存器數量會多不少，面積會佔整體較大的部分。在時間的部分，在一開始規劃使用 pipeline 欲將 critical path 切開，並且最大化同時運用硬體資源降低 cycle 花費數。結果顯示出我們在優化 cycle 有達成預先的目的，然而在 timing 的部分因為 module depth 中加入的額外的 spike 判斷程序，其行為上在一個 clock 中做完所有判斷並且計算其位址，因此花費大量的邏輯計算時間，導致在 module pipeline 所帶來的時間優勢有被浪費的情形產生。module Depth 在 spike 判斷程序的其他行為在設計上和 module pipeline 相同，所以可以預測針對 spike 判斷程序優化就能獲得巨大的進展。優化方式可以將 spike 判斷程序拆分成在二個或更多 clock 中進行，降低單個 clock 中的工作量。

若套用 3-2 的 cycle 計算結果，展示在經過 100 次推演後獲得 1 次預測結果所花費的時間。可以發現每次預測花費時間在 1mm sec 左右，相對於目標的小型車輛的機電系統應足夠快速，能滿足視覺處理的要求。考慮到電力消耗，實際應用上應該會降頻至合適的頻率以減少功耗，因此可見這樣的加速晶片確實足夠迅速。

	Amont of cycle	Time per prediction(ms)
Worst case	57300	1.26
Average case	12754	0.28

Table.2 Result of inference time per prediction(100 timesteps)

4. Conclusion

本專案基於脈衝神經網路（Spiking Neural Network, SNN）的設計，實現了一個自走車避障的硬體加速器。透過 Verilog 實現硬體架構，並進行一系列的優化，我們成功地將該系統應用於實時視覺數據的處理。資料流的設計與控制訊號的協調是系統高效運作的核心，而 Pipeline 技術與動態枝剪技術進一步提高了硬體的效能與功耗表現。

本次的硬體實現展現了 SNN 在嵌入式系統中的潛力，尤其是在低功耗、低延遲應用中的優勢。未來的發展方向可以集中於進一步優化網路結構和計算單元，使該系統更具擴展性與應用範圍。

5. Review and reflections

在兩學期的專題中，我們從閱讀文獻瞭解脈衝神經網路的基本原理以及其應用上的潛力。我們也在每次的報告中提升了重點整理和口頭發表的能力。第二學期開始時線電路的過程中，我們最大的收穫是學習自行規劃整體的架構，並逐步解決各硬體部件整合的問題。過程中也能反思當初的規劃是否得當，或是需要根據現況進行計畫的更改，因次我們獲益良多。

感謝鄭桂忠教授和實驗室的學長們讓我們參與此專題，提供我們學習資源並且細心的解決我們的問題。學長們也在每周的開會中督促我們，更在我們的規畫和安排上給出建議，使研究模擬過程能順利完成。

6.References

[1] Chen-Fu Yeh, Chao-Yang Tang, Tsu-Chiao Chen, et al. FlowDep - An efficient and optical-flow-based algorithm of obstacle detection for autonomous mini-vehicles. *TechRxiv.* April 03, 2024.