為了將來能夠將整個TCBNN實作成硬體並放入pynq板子，先嘗試將簡單的神經網路做成硬體放入pynq中。

實驗:簡單神經網路以vitis HLS 合成，並且做成IP放進pynq驗證成功。

首先要建構一個簡單的神經網路，我以MNIST為資料集，訓練三層神經網路，準確度為97.9%。

# Python軟體

Dataset :MNIST

Accuracy:96.9

Neural network structure:



# Vitis HLS

為了要將Python 合成為硬體並放入pynq中交給VITIS HLS合成，我將神經網路架構寫成C++格式。(如附檔)

在pynq當中，設計IP與Host的傳輸介面有Axi-lite,Axi-master,Axi-stream等等。

由於Axi-lite一次傳輸的資料量太少，Stream傳輸方式尚未實驗成功，所以這次實作內容是以Axi-Master的傳輸方式來實現。

Axi-master 傳輸介面可以以指標或者陣列的資料格式傳輸，在此有兩種傳輸模式[1]:

1.individual data transfer: 以傳輸的地址讀或寫單一筆資料。

2.burst mode: 以傳輸的地址讀或寫多筆連續的資料，連續的長度(burst-length)由使用者定義，以C的memcpy的方式實現。在vivado中，由於硬體規格的限制，單次傳輸資料的上限為512bit。

本次實驗方式是以burst mode傳輸神經網路內的權重,偏差和輸出入資料。

將權重,偏差,以及輸入圖片先放進pynq板內的global memory,再藉由Host(pynq) 的axi-lite控制資料輸入，將圖片、權重傳入設計的IP內，當IP運算完成後，會將結果output到global memory，最後將output結果再Host控制取出運算結果。

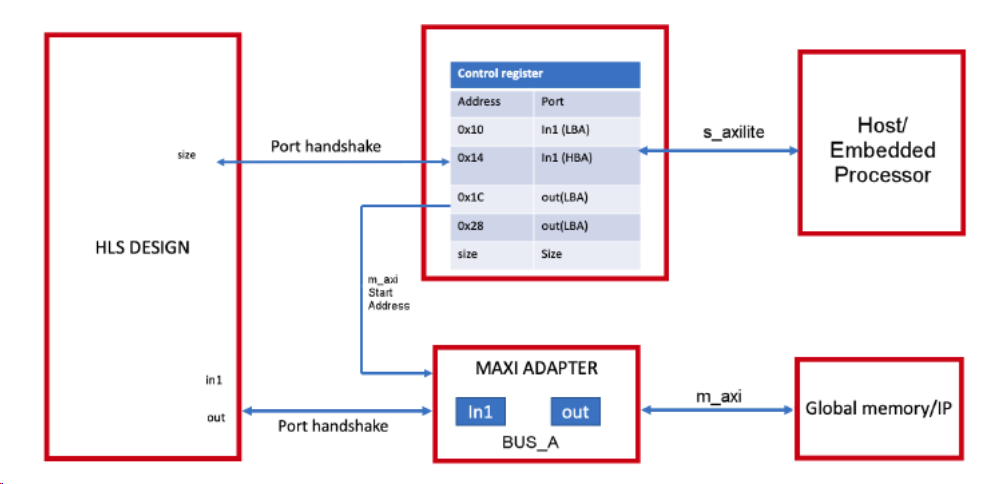


Figure 1 m\_axi interface[1]

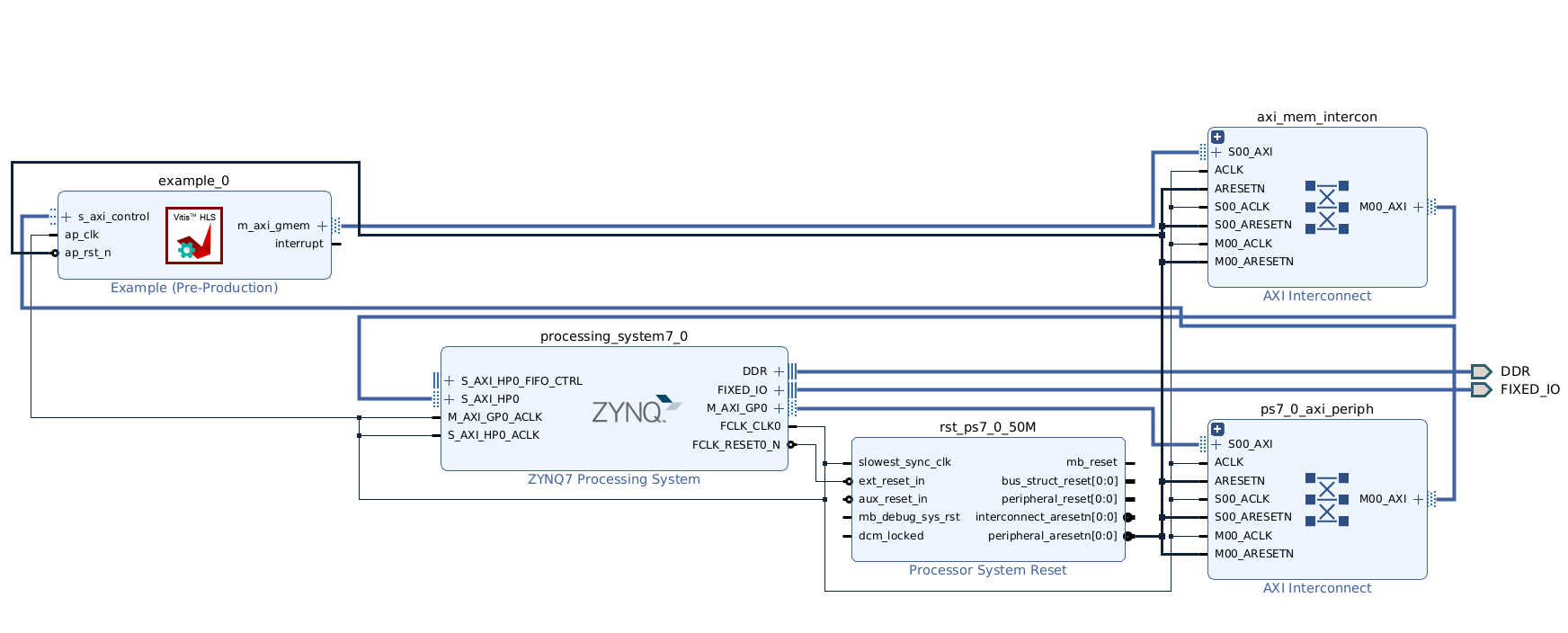


Figure 2 vivado diagram

# Pynq 驗證

最後將C++所設計的kernel 包成IP丟進vivado做成bi t stream和hwh檔案傳到pynq板子上做驗證，看看實作結果是否與軟體算出來的結果是否一致，和比較速度。

我將資料以np.save的方式存成npy放入pynq中，比較兩種方式運算10張圖片所需要的時間。

|  |  |  |
| --- | --- | --- |
|  | pynq軟體 | IP |
| Time(s) | 39.0618462562561 | 0.13218092918395996 |
| Accuracy(%) | 96.9 | 96.8 |

加速295.51倍。

IP計算1000張圖片所需要的時間: 12.220770597457886(s)

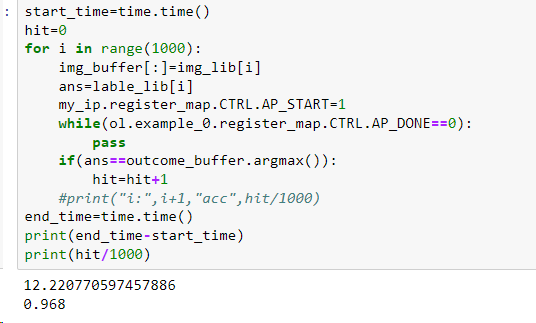


Figure 3 pynq outcome

# 結論

本次實驗成功將簡單神經網路以硬體方式實現，並且加速295倍。

接下來:

1. 繼續實驗使用stream傳輸資料格式
2. 開始嘗試將AN code加入神經網路做訓練
3. TCB乘法代替位移以C++方式硬體實現

Reference:

1. https://www.boledu.org/textbooks/hls-textbook/io-interface/axi-master