實驗:簡單神經網路使用 Vitis HLS 合成，並放進 IP 驗證成功

本次實驗與上次不同之處在於本次傳輸介面使用 Stream，並且不需要輸入 weight、bias 等等，只需輸入圖片資料，輸出向量就是判斷結果。

Python 軟體

與上次架構相同。

Dataset :MNIST

Accuracy:96.9

Neural network structure:

```
class NeuralNetwork(nn.Module):
  def __init__(self):
    super(NeuralNetwork, self).__init__()
    self.flatten = nn.Flatten()
    self.linear_relu_stack = nn.Sequential(
      nn.Linear(28*28, 64),
      nn.ReLU(inplace=True),
      nn.Linear(64,32),
      nn.ReLU(inplace=True),
      nn.Linear(32, 10),
    )
```

Vitis HLS

本次使用 stream 傳輸介面，有兩種使用 AXI4-Stream 的方式

1. AXI4-Stream without side channel
2. AXI4-Stream with side channel

Axit4 stream 傳輸介面被定義在 ap_axi_sdata.h 中，包含 T,WUser,WId,WDest。

T	Stream data type
WUser	Width of the TUSER signal
WId	Width of the TID signal
WDest	Width of the TDest signal

Table 1 axis interface

當沒有使用 side channel 時，ap_axiu<32,0,0,0>資料不會包含 WUser, WId, and WDest 等等。

WUser,WId,WDest 這些訊號是能被使用在 RTL level 中的 stream 傳輸訊號使用。

在本次使用時，我只需要注意 T(stream data type) ,就能完成本次實驗，無須用到其他 side channel 訊號，所以本次實驗使用的是 1.stream without side channel。

Stream data type 若傳輸的是整數，則會有以下格式可以做選擇

1. Singned :hls::axis<ap_int<WData>, WUser, WId, WDest>
2. Unsigned: hls::axis<ap_uint<WData>, WUser, WId, WDest>

本次使用 2. unsigned 的資料傳輸格式。

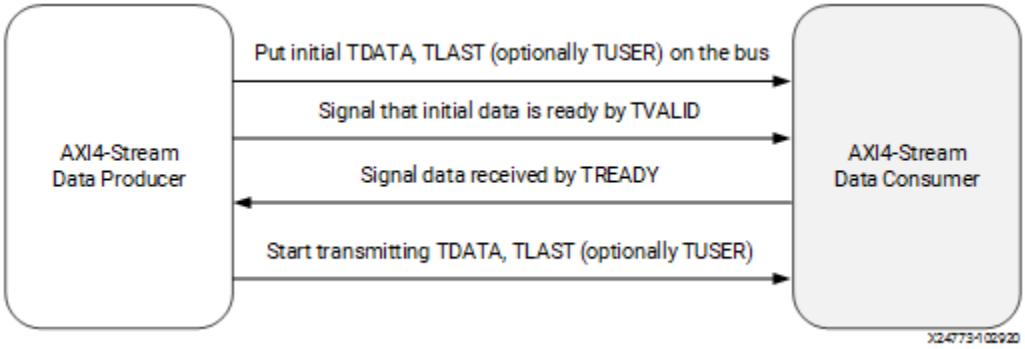

Figure 1 Stream handshake protocol

Stream 傳輸資料時，一定包含 Data producer 傳輸的 TVAID,TDATA,TLAST 以及 consumer 回傳的 TREADY。

在實作中，傳輸資料時 TVALID,TDATA 等資料會由 host 幫忙發送與傳遞，若使用 without side channel 時，則必須要自己加上 TLAST 在傳遞訊息中，否則硬體沒有接受到 TLAST 永遠不會完成資料傳輸。

硬體架構

由於 float point 尚未明白如何傳輸，本次實驗所有資料為 32bit int。

並且由於硬體計算時會遇到 overflow 問題，在做完第一層 layer 運算時，我將所有 bit 向右移 14bit，除以 16384。對準確度有一點影響。

```
        }
        else{
            //// dealing with overflow or underflow
            out1[i]=out1[i]/16384;
        }
    }
```

Figure 2 dealing with overflow

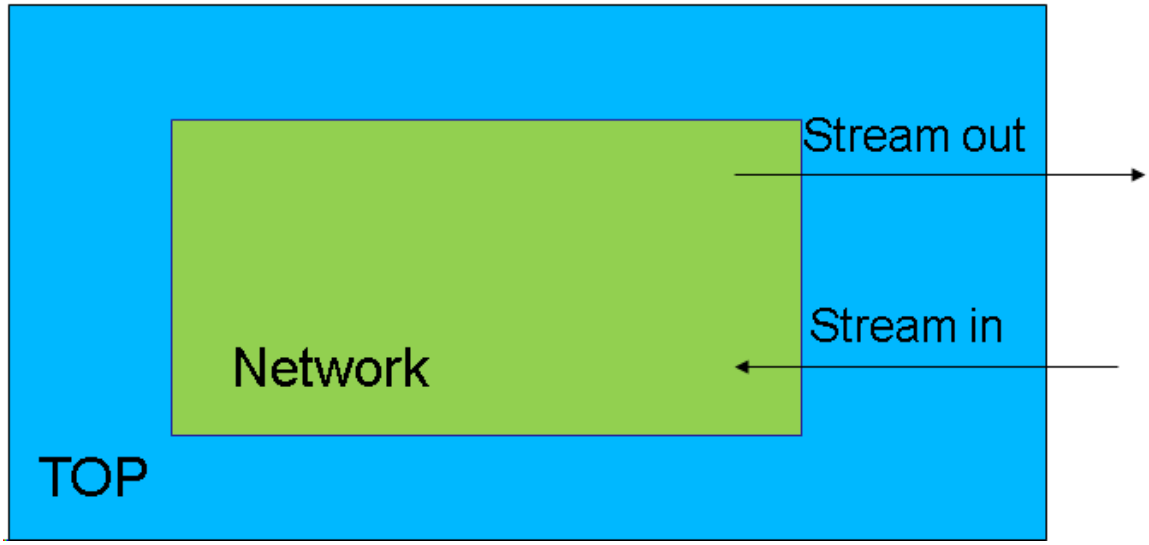

Figure 3 Hardware Block Diagram

僅有 Stream in,stream out 兩個 port。

```
void top(hls::stream<axis_t> &in, hls::stream<axis_t> &out){
#pragma HLS INTERFACE s_axilite port = return bundle = control
#pragma HLS INTERFACE axis port = in
#pragma HLS INTERFACE axis port = out
```

Figure 4 strea, in/out port

在硬體合成時，wieght,bias 等等已知的資料被合成為 ROM，而在硬體中的累加器被合成為 RAM。

```
Implementing memory 'top_nn_Pipeline_VITIS_LOOP_19_2_lrsw0_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_Pipeline_nn_label2_lrsw2_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_Pipeline_nn_label2_lrsb2_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_Pipeline_VITIS_LOOP_55_7_lrsw4_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_Pipeline_VITIS_LOOP_55_7_lrsb4_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_lrsb0_ROM_AUTO_1R' using auto ROMs.
Implementing memory 'top_nn_out1_RAM_AUTO_1R1W_ram (RAM)' using auto RAMs.
Implementing memory 'top_nn_out2_RAM_AUTO_1R1W_ram (RAM)' using auto RAMs.
Implementing memory 'top_b_img_RAM_AUTO_1R1W_ram (RAM)' using auto RAMs.
Implementing memory 'top_b_out_RAM_AUTO_1R1W_ram (RAM)' using auto RAMs.
```

Figure 5 Implementation ROM/RAM

接著合成為 IP。使用了一個 DMA read/write 處理資料 in/out。

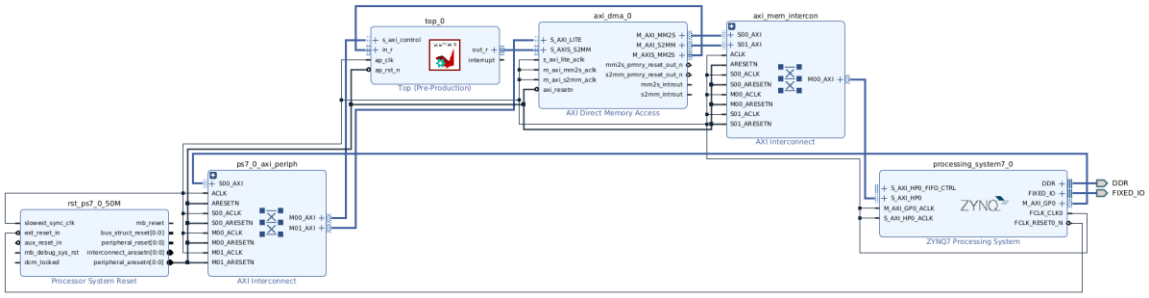

Figure 6 Vivado Diagram

Pynq 驗證

拿 1000 張 test dataset 中的圖片輸入觀察準確度 96.6%

花費 14.21 秒。

```
hit=0
start_time=time.time()
for i in range(1000):
    inBuffer0[:]=img[i]
    dma.sendchannel.transfer(inBuffer0)
    dma.recvchannel.transfer(outBuffer)
    ol.top_0.write(0x00, 0x01)
    while(ol.top_0.register_map.CTRL.AP_DONE==0):
        pass
    if(outBuffer.argmax()==label[i]):
        hit+=1
    #print("times: ",i+1,"hit: ",hit)
print("acc",hit/1000)
print("spend time",time.time()-start_time)
```

```
acc 0.966
spend time 14.211108684539795
```

Figure 7Pynq outcome

問題與討論

1. float point data type
   目前尚未了解 float point 如何再 interface 介面傳輸。
   一開始做實驗時，我本來打算使用 float point 做為 data type，但是合成為硬體後發現輸出資料是錯誤結果。

我後來將 float point 做成 ROM 如下圖。

```
const static float  arr0[]={0.5,-0.5,0.1,-0.1,1,-1,2, -2,-10,10};
const static float  arr1[]={-20.17459297,   -8.31953716 ,  18.02685547 ,
                            -0.5553475 , -102.96446991,  17.36007881,
                            -41.6920433  ,   1.72449279  ,-38.64471054 ,
                             -56.7712326};
```

Figure 8 C++ ROM

將 arr0 中 10 筆資料讀取，並加 1 得到 1.e+00 結果。小數點等資料都不見了。
將 arr1 中 10 筆資料單純讀取，僅得到 nan。目前尚未明白 float 如何傳輸。

```
dma.sendchannel.transfer(inBuffer0)
dma.recvchannel.transfer(outBuffer)
ol.top_0.write(0x00, 0x01)
while(ol.top_0.register_map.CTRL.AP_DONE==0):
    pass
print(outBuffer)
```

```
[1.0e+00 1.0e+00 1.0e+00 1.0e+00 1.0e+00 1.0e+00 1.0e+00 1.0e+00 1.0e+00
 1.0e+00     nan     nan     nan     nan     nan     nan     nan     nan
 2.5e-44 0.0e+00]
```

Figure 9 Error outcome

結論

目前能夠使用 stream 傳輸介面傳輸整數 32bit 資料，並實作簡單神經網路。

接下來:

a. 研究 float point 傳輸資料方式
b. 嘗試 AN code 加入神經網路做訓練
c. TCB 未提代替乘法以 C++方式硬體實現