# 以浮點數為資料型態的神經網路

# 實驗來由

為了比較普通神經網路與有使用TCB的網路運算速度,硬體資源等等差異，有必要將floating point的神經網路實作出來，當作TCB的對照組。

# 解決floating point無法傳輸的問題

上週遇到一個在stream中傳輸浮點數無法正常讀取的問題，無論寫入資料為何，讀取時都是亂數。本周找到了解決方案。

# 如何傳輸float data使用axis stream

根據這篇文章: <https://support.xilinx.com/s/question/0D52E00007DnHxuSAF/streaming-floats-with-tlast?language=en_US>

在stream介面中，只要使用除了ap\_axis, ap\_axiu外的資料型態，HLS就不會再合成硬體時加上TLAST訊號。這會導致傳輸協定永遠無法達成。

所以在傳輸資料時，要將stream傳輸介面的資料當成int或者uint傳輸，在內部運算時用converter將int轉換成float。如Figure 1 converter所示。

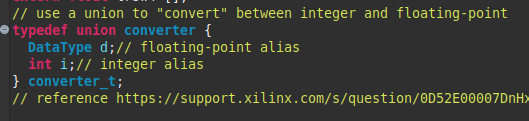


Figure converter

內部運算要將int 轉回float如Figure 2 int to float 所示。

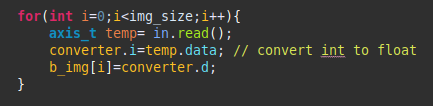


Figure int to float

# 硬體實作

軟體架構與先前相同，784x64x32x10。

硬體部分input img和內部ROM 中的weight,bias等等都是floating point。

# DSP最大數量限制

若完全不對TOOL加上任何pragma或directive，TOOL會以他自己的方式自動做硬體最佳化，盡可能將所有LOOP做PIPELINE，如此會導致硬體資源大幅增加。

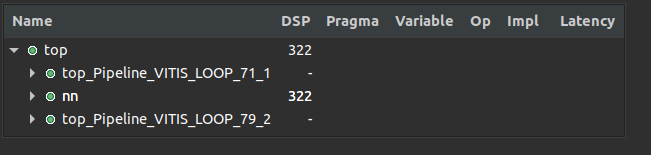
在不加上任何pragma的情況下，合成一個floating point的硬體會需要322個DSP。

Figure Without pragma outcome

但由於pynq z1的硬體限制，pynq z1中，DSP可使用最大數目為220個，為了要能夠將硬體順利合成，有必要在source code中加上pragma 使得硬體DSP硬體使用數量下降。

如所示，觀察合成結果，在LOOP30\_4這邊使用了大量的DSP。

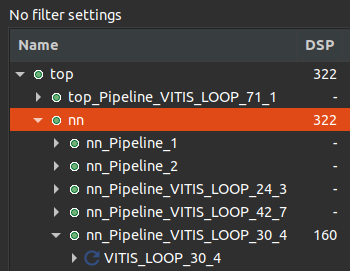


Figure LOOP\_30\_ 4

為了解決這個問題，在directive中添加PIPE\_LINE OFF的命令。

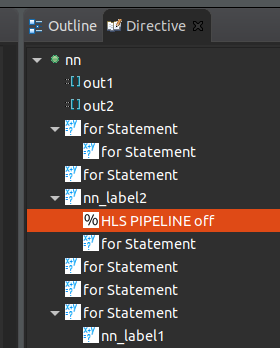


Figure PIPELINE off

如此再去做合成就能夠限制DSP數量降到162個並且順利合成硬體。

Figure With Pragma outcome

# PYNQ驗證

將硬體合成IP放到pynq上做準確度測試。

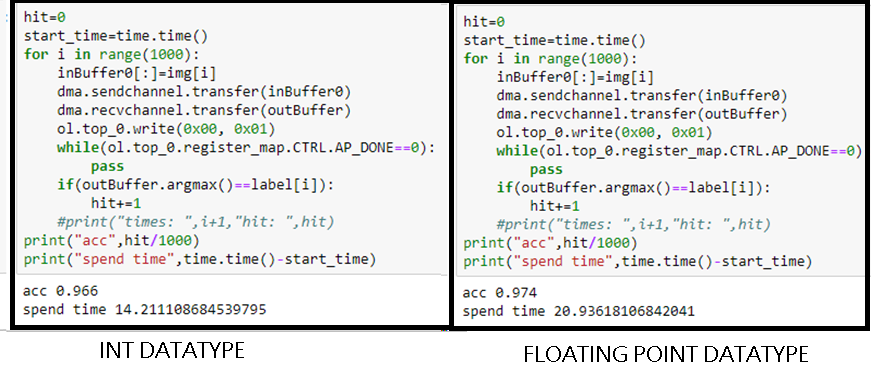


Figure 7 Comparison fig

Floating poiint所花時間比int慢6.72秒。但準確度高了0.8%。