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。

```
// use a union to "convert" between integer and floating-point
typedef union converter {
  DataType d;// floating-point alias
  int i;// integer alias
} converter_t;
// reference https://support.xilinx.com/s/question/0D52E00007DnHx
```

Figure 1 converter

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

```
for(int i=0;i<img_size;i++){
    axis_t temp= in.read();
    converter.i=temp.data; // convert int to float
    b_img[i]=converter.d;
}
```

Figure 2 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。

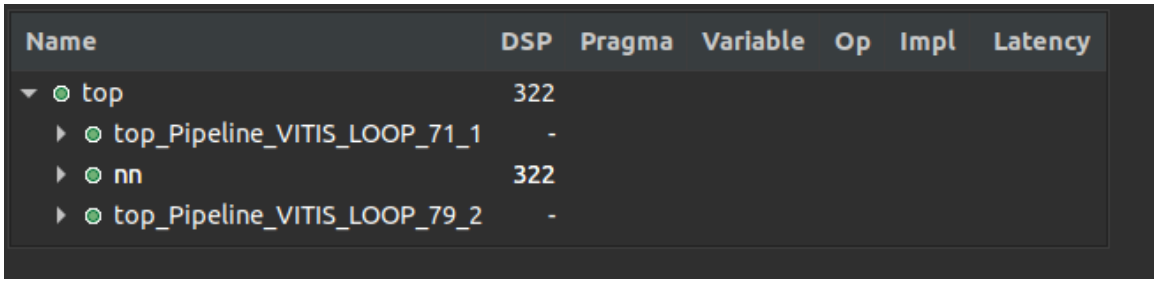

Figure 3Without pragma outcome

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。

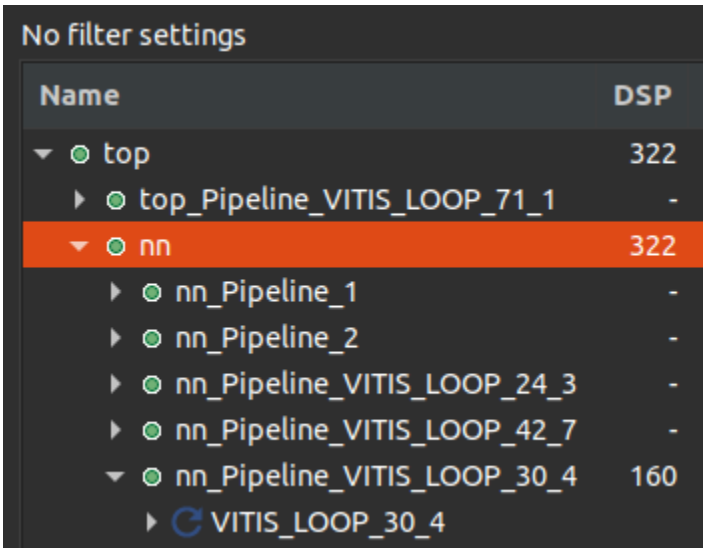

Figure 4 LOOP_30_ 4

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

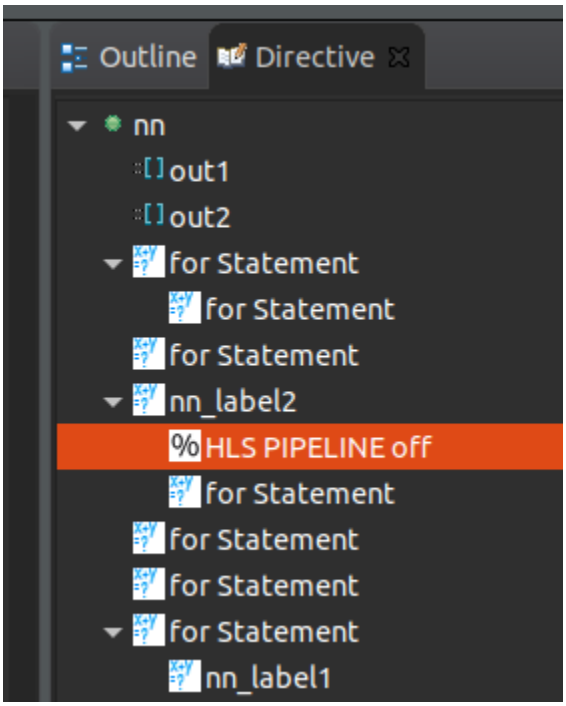

Figure 5 PIPELINE off

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 6 With Pragma outcome

PYNQ 驗證

將硬體合成 IP 放到 pynq 上做準確度測試。

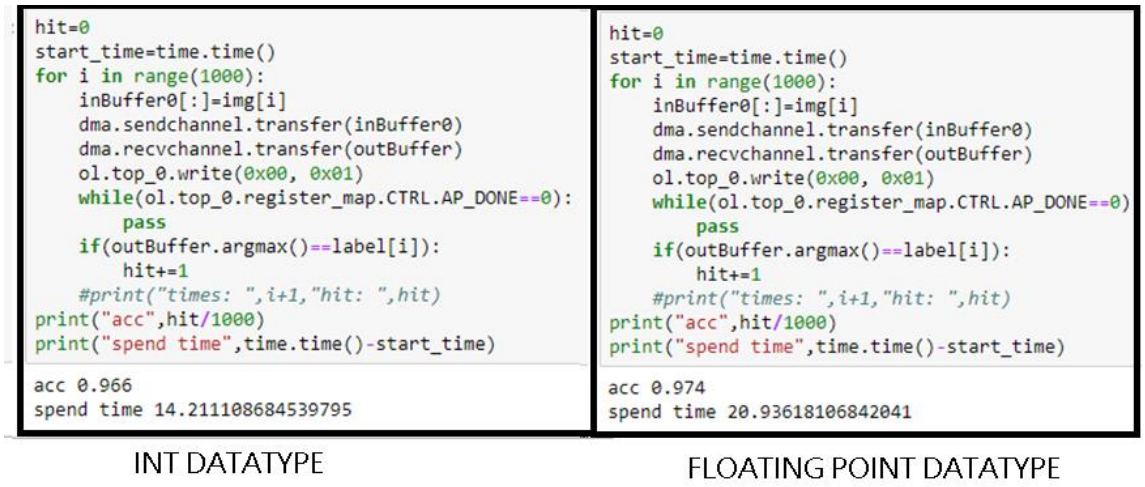

Figure 7 Comparison fig

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