

# Lab2 – Submission Guide

311651055\_林柏宇

A. 這次主要是嘗試 Stream 和 MAXI (Memory-mapped AXI Interface, 記憶體映射 AXI 介面) 兩種不同的通訊和資料傳輸方式。次要再用除了上一次 inline (用 #pragma) 的作法, 這次用 directives.tcl 來控制。

## 1. Stream :

(1)特點：一種連續的資料流傳輸方式，資料從一個模組傳遞到另一個模組，不需要明確的記憶體位址或讀寫操作。用於處理即時數據流，如音訊、視訊和感測器數據。

(2)通訊方式：將資料流從一個模組傳遞到另一個模組來實現，資料透過 FIFO (First-In-First-Out) 緩衝區傳輸，而不是透過記憶體位址。資料可以是連續的、幀同步的，或是任意格式的資料。

(3)應用：Stream 通訊適用於需要低延遲、高吞吐量的即時資料傳輸和處理應用，如數位訊號處理、影像處理、視訊編解碼等。

## 2. MAXI (Memory-mapped AXI Interface) :

(1)特點：將資料儲存在 FPGA 的記憶體中，並透過記憶體位址存取。適用於處理大量的離散數據，需要隨機存取記憶體的應用。

(2)通訊方式：MAXI 通訊透過 AXI 匯流排協定進行，讓 CPU 或其他硬體模組透過記憶體位址讀取或寫入資料。它通常與記憶體控制器（如 BRAM 或 DDR 記憶體控制器）一起使用，以便在 FPGA 中進行記憶體存取。

(3)應用：MAXI 通訊適用於需要大量資料儲存和隨機存取的應用，例如資料快取、演算法處理中的中間資料儲存等。

B. 以下是我實作上遇到需要注意的細節：

1. 加入 directive 有 inline (用#pragma)與 directives.tcl 來控制。Lab 1 使用第一種方式，而 Lab2 則是第二種。

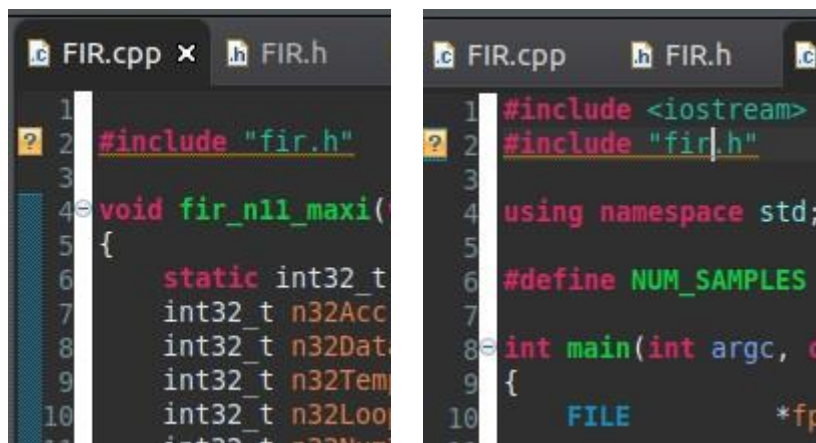
Tcl (Tool Command Language, .tcl) 是一種腳本語言，它允許你編寫腳本來控制建置、編譯和連結等操作。你可以編寫一個包含 Tcl 腳本的.tcl 文件，然後在建置系統中使用它來自訂建置過程。.tcl 檔案中包含了一個自訂建置規則，使用了 gcc 指令來建立一個可執行檔。

```
1 #####
2 ## This file is generated automatically by Vitis HLS.
3 ## Please DO NOT edit it.
4 ## Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
5 #####
6 set_directive_interface -mode s_axilite "fir_n11_maxi"
7 set_directive_interface -mode m_axi -depth 600 -offset slave "fir_n11_maxi" pn32HPInput
8 set_directive_interface -mode m_axi -depth 600 -offset slave "fir_n11_maxi" pn32HPOutput
9 set_directive_interface -mode s_axilite "fir_n11_maxi" an32Coef
10 set_directive_interface -mode s_axilite "fir_n11_maxi" regXferLeng
```

2. 在 Testbench 中，這裡用絕對路徑較好，確保不會找不到檔案：

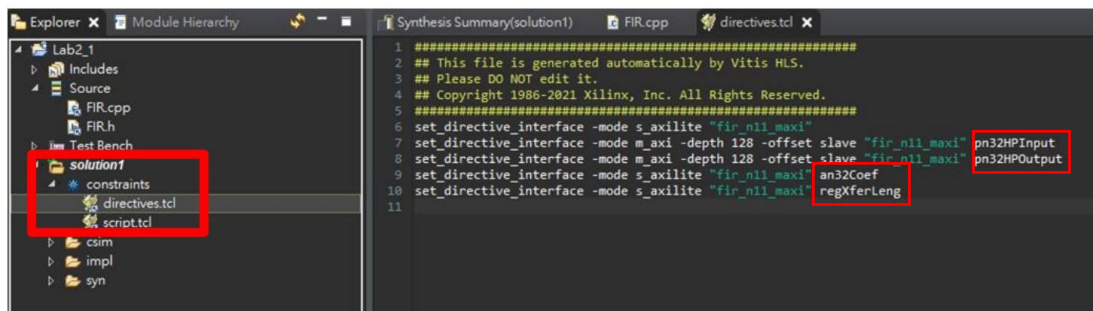
```
cout << ">> Comparing against output data..." << endl;
// /home/course-lab 2/hls_ip/solution1/csim/build/out.dat
// /home/course-lab 2/hls_FIRN11MAXI/out_gold.dat
if (system("diff out.dat /home/ubuntu/course-lab 2/hls_FIRN11MAXI/out_gold.dat")) {
    cout << ">> Test failed!" << endl;
    return 1;
}
```

3. 標頭檔大小要注意(與.h 檔案符合)：要改 fir->FIR

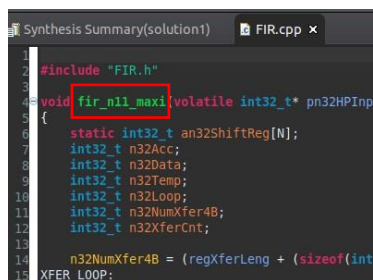
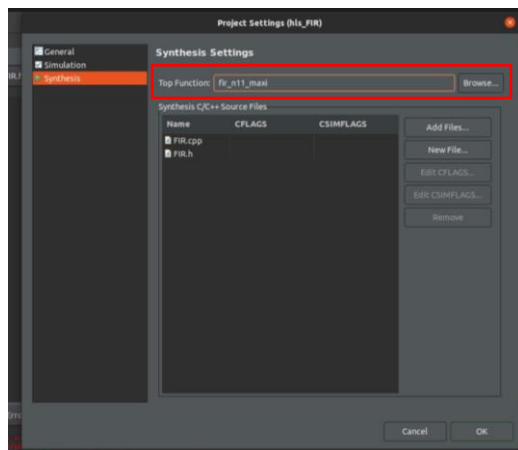


4. directive 的設置要先做(用圖形化，照著下面)，才能跑 simulation：depth

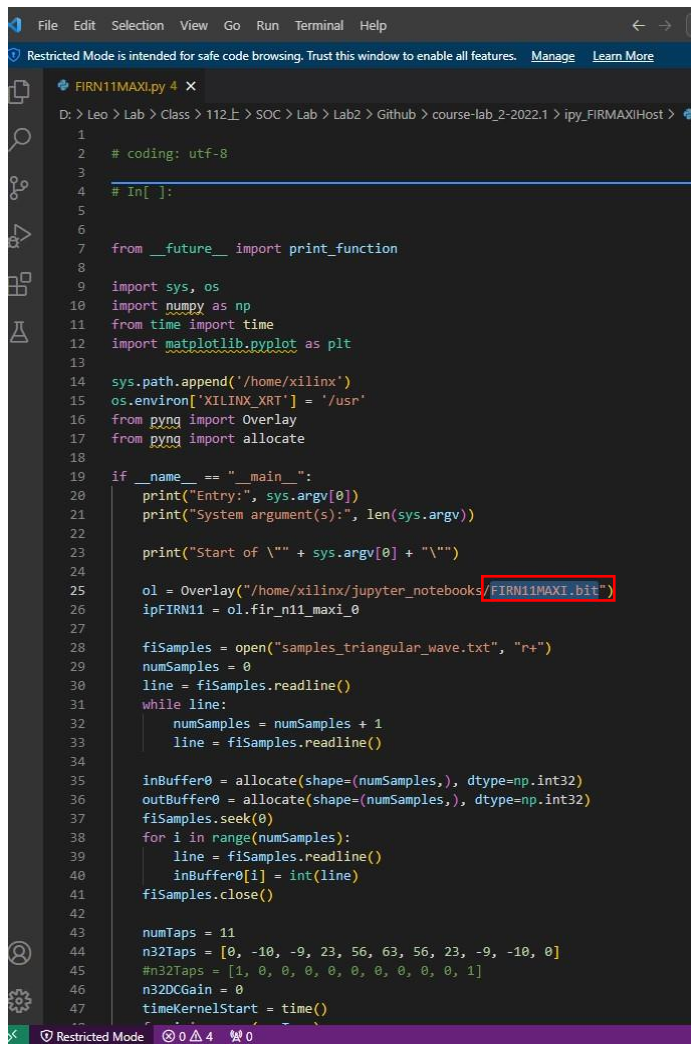
要設 600 也就是說 Testbench 的 kernel 要夠用。



5. TopFunction 中包含一些與輸入和輸出資料流的接口，這裡軟體的設置要改和 .cpp 中的函式名稱一樣。



6. 這個.py 檔用於在 Xilinx PYNQ 開發板上執行一個 FIR 濾波器的加速應用程式。跑完的.hwh 與 .bit 檔名要注意和 .py 函式中的檔名一樣(副檔名不用改)

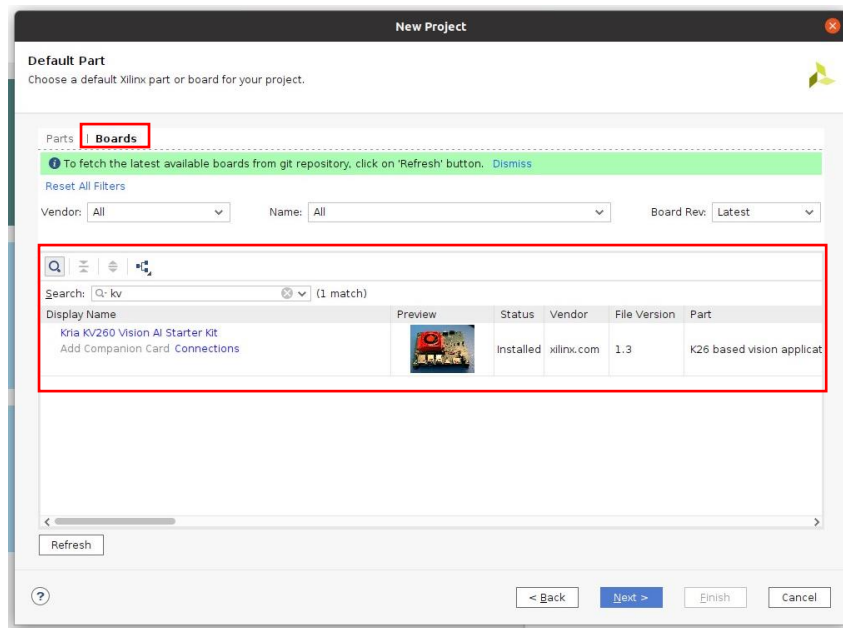


```
1
2 # coding: utf-8
3
4 # In[ ]:
5
6
7 from __future__ import print_function
8
9 import sys, os
10 import numpy as np
11 from time import time
12 import matplotlib.pyplot as plt
13
14 sys.path.append('/home/xilinx')
15 os.environ['XILINX_XRT'] = '/usr'
16 from pyng import Overlay
17 from pyng import allocate
18
19 if __name__ == "__main__":
20     print("Entry:", sys.argv[0])
21     print("System argument(s):", len(sys.argv))
22
23     print("Start of \"" + sys.argv[0] + "\"")
24
25     ol = Overlay("/home/xilinx/jupyter_notebooks/FIRN11MAXI.bit")
26     ipFIRN11 = ol.fir_n11_maxi_0
27
28     fiSamples = open("samples_triangular_wave.txt", "r+")
29     numSamples = 0
30     line = fiSamples.readline()
31     while line:
32         numSamples = numSamples + 1
33         line = fiSamples.readline()
34
35     inBuffer0 = allocate(shape=(numSamples,), dtype=np.int32)
36     outBuffer0 = allocate(shape=(numSamples,), dtype=np.int32)
37     fiSamples.seek(0)
38     for i in range(numSamples):
39         line = fiSamples.readline()
40         inBuffer0[i] = int(line)
41     fiSamples.close()
42
43     numTaps = 11
44     n32Taps = [0, -10, -9, 23, 56, 63, 56, 23, -9, -10, 0]
45     #n32Taps = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
46     n32DCGain = 0
47     timeKernelStart = time()
```

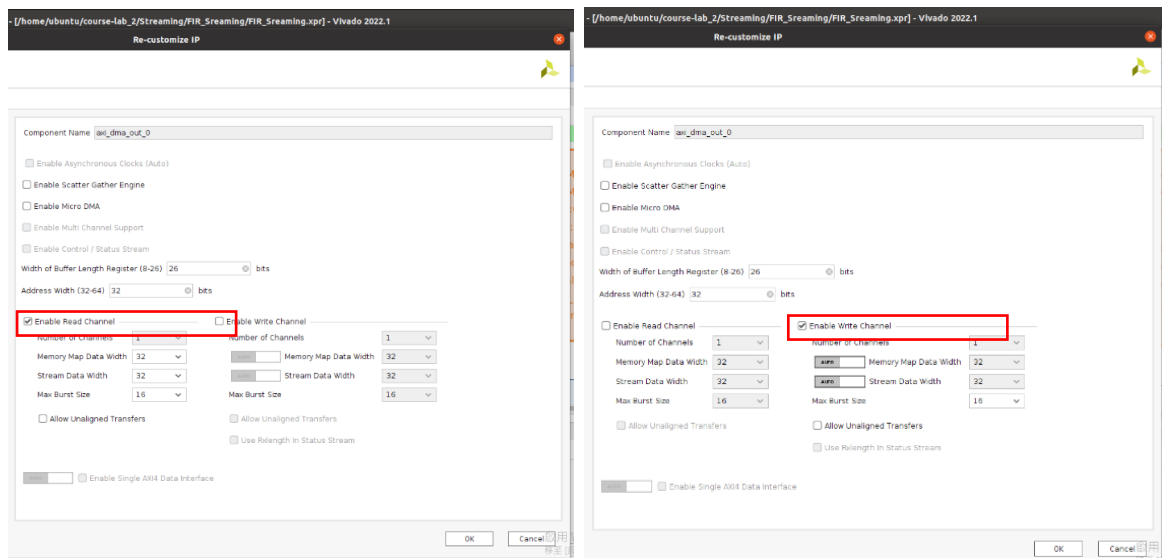
.hwh (Hardware Handoff)：透過 HLS(Vivado)產生的。 .hwh 是在硬體設計和軟體設計之間建立了連接，讓軟體開發人員了解 FPGA 硬體的基本結構，以便在軟體中與 FPGA 進行通訊。

.bit (Configuration Bitstream)：透過將 FPGA 設計綜合、映射、佈局和路由後產生的。包括邏輯元件的佈局、連接、IP 核的配置等。一旦載入到 FPGA 上，.bit 檔案會配置 FPGA 以實現特定的硬體功能。

7. 如果用的是 KV260 板子，要多選這個。



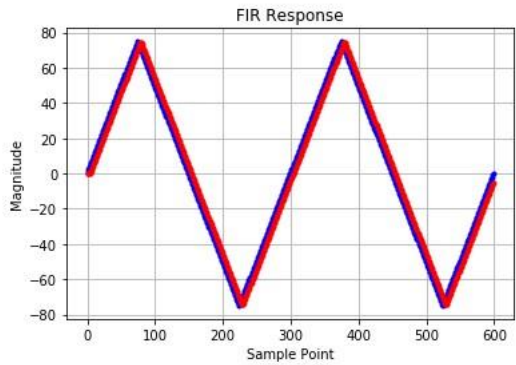
8. AXI DMA 的 Input、Output 分別勾選左邊和右邊



C. 結果呈現：

以 MAXI 來做的結果：

```
Entry: /usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py
System argument(s): 3
Start of "/usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py"
Kernel execution time: 0.0005288124084472656 s
```



=====  
Exit process

Synthesis Summary(solution1) x FIR.cpp Co-simulation Report(solution1)

Synthesis Summary Report of 'fir\_n11\_maxi'

General Information

Date: Wed Oct 4 09:10:54 2023  
Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)  
Project: hls\_fir\_maxi

Solution: solution1 (Vivado IP Flow Target)  
Product family: zynq  
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.300 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops

Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
-	-	-	-	-	-	-	-	-	-	no	0	33	4456	3053
-	-	-	-	-	-	-	-	-	-	no	0	33	2794	1084

HW Interfaces

Synthesis Summary(solution1) x FIR.cpp Co-simulation Report(solution1) x

Cosimulation Report for 'fir\_n11\_maxi'

General Information

Date: Wed 04 Oct 2023 11:37:02 AM EDT  
Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)  
Project: hls\_fir\_maxi  
Status: Pass

Solution: solution1 (Vivado IP Flow Target)  
Product family: zynq  
Target device: xc7z020-clg400-1

Cosim Options

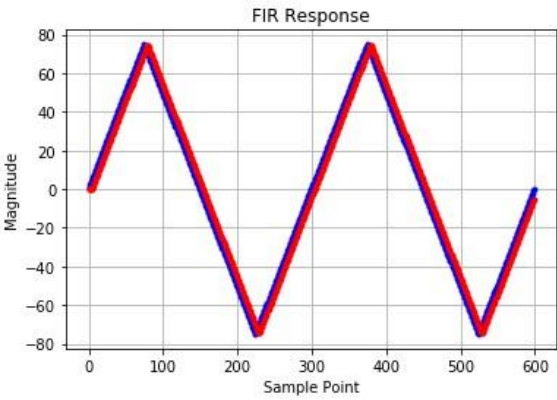
Tool: Vivado XSIM  
RTL: Verilog  
Dump Trace: all

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
- fir_n11_maxi				728	728	728
- fir_n11_maxi Pipeline_XFER_LOOP				715	715	715

以 Stream :

```
Entry: /usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py
System argument(s): 3
Start of "/usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py"
Kernel execution time: 0.0016736984252929688 s
```



```
=====
Exit process
```

Synthesis Summary(solution1) x FIR.cpp Co-simulation Report(solution1)

Synthesis Summary Report of 'fir\_n11\_strm'

General Information

Date: Wed Oct 4 02:20:15 2023

Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)

Project: hls\_fir\_streaming

Solution: solution1 (Vivado IP Flow Target)

Product family: zynq

Target device: xc7z020-clg400-1

Timing Estimate

Target Estimated Uncertainty

10.00 ns 6.923 ns 2.70 ns

Performance & Resource Estimates

Modules & Loops

Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URA
▼ fir_n11_strm			-	11019	1.100E5	-	11020	-	no	0	33	3024	1409	
► fir_n11_strm Pipeline_XFER_LOOP	II Violation		-	11016	1.100E5	-	11016	-	no	0	33	2834	1153	

Cosimulation Report for 'fir\_n11\_strm'

General Information

Date: Wed 04 Oct 2023 11:33:18 AM EDT

Version: 2022.1 (Build 3526262 on Mon Apr 18 15:47:01 MDT 2022)

Project: hls\_fir\_streaming

Status: Pass

Solution: solution1 (Vivado IP Flow Target)

Product family: zynq

Target device: xc7z020-clg400-1

Cosim Options

Tool: Vivado XSIM

Dump Trace: all

RTL: Verilog

Performance Estimates

Modules & Loops

Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
▼ fir_n11_strm			6606	6606	6606
► fir_n11_strm Pipeline_XFER_LOOP			6603	6603	6603

D. 其他關於 code 的註解，我會再放到 Github。