**國立清華大學**

**系統晶片設計**

**SOC Design**

一張含有 文字, 筆跡, 字型, 圓形 的圖片

自動產生的描述

**Lab 2**

**系所級:電子所二年級**

**學號:111063548**

**姓名:蕭方凱**

**指導老師:賴瑾教授**

目錄

[I. FIR with Interface AXI-Master 3](#_Toc147167310)

[1. HLS/IP Design 3](#_Toc147167311)

[(1) FIR.h 3](#_Toc147167312)

[(2) FIR.cpp 3](#_Toc147167313)

[(3) FIRTester.cpp 4](#_Toc147167314)

[(4) Directives.tcl 4](#_Toc147167315)

[2. Vivado Implementation 5](#_Toc147167316)

[3. Python Code Validation via Jupyter Notebook 6](#_Toc147167317)

[II. FIR with Interface Streaming 7](#_Toc147167318)

[1. HLS/IP Design 7](#_Toc147167319)

[(1) FIR.h 7](#_Toc147167320)

[(2) FIR.cpp 7](#_Toc147167321)

[(3) FIRTester.cpp 8](#_Toc147167322)

[2. Vivado Implementation 9](#_Toc147167323)

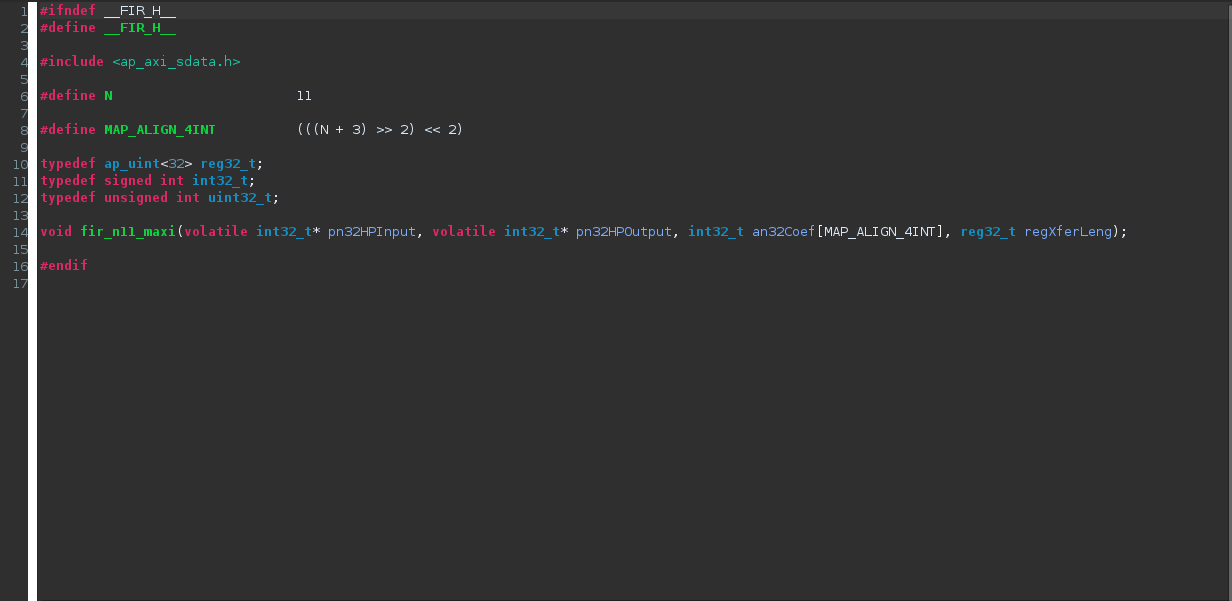
[3. Python Code Validation via Jupyter Notebook 11](#_Toc147167324)

[III. Discussion 12](#_Toc147167325)

# FIR with Interface AXI-Master

HLS/IP Design

### FIR.h



ifndef&define上一個lab已經使用過，故不再贅述。這個標頭檔首先定義了常數N為11，接著〝typedef ap\_uint<32> reg32\_t;〞代表定義一個無符號數reg32\_t，其位元長度為32，這種定義方法是HLS提供給用戶進行任意長度的數據運算，範圍可從1至1024。至於MAP\_ALIGN\_4INT的部分，是控制an32coef為大於11同時又是4的倍數，方便FIR運算。

### FIR.cpp

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述

這是一個實現FIR濾波器運算的程式碼，首先定義多個參數，型別為int32\_t，用於存儲變數或累積變數(for迴圈)。XFER\_LOOP為進行數據傳輸的迴圈，並在每次執行for迴圈時將n32Acc初始化為零。SHIFT\_ACC\_LOOP迴圈則是負責移位和累積的運算，將移位暫存器 an32ShiftReg 中的值移位，並將新的值存儲在 n32Data 中，用於與濾波器的係數相乘(n32Acc += n32Data \* an32Coef[n32Loop])，最終將n32Acc存儲在輸出pn32HPOutput[n32XferCnt]。

### 一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片 自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片 自動產生的描述FIRTester.cpp

這段程式碼是為了生成一鋸齒波信號，並與outgold.dat進行比對，若相符則顯示Test Passed，否則顯示Test failed。

### Directives.tcl

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

本次lab與上次使用#pragma方式不同，是直接在FIR.cpp的DIRECTIVE部分insert，insert完成後可於solution/constraints/directives.tcl部分檢查。

Vivado Implementation

在Vitis\_HLS Export RTL後，接著在Vivado開啟IP。步驟大致與LAB1相同，較不同的地方是針對ZYNQ7 Processing System，需調整HP Port設定，如下圖:

一張含有 文字, 螢幕擷取畫面, 軟體, 數字 的圖片

自動產生的描述

調整完成後即可將ZYNQ7與自行設計的IP連線，進行auto connection，完成後如下圖:

一張含有 文字, 螢幕擷取畫面, 軟體, 行 的圖片

自動產生的描述

接著與LAB1相同，Create HDL Wrapper → Generate bitstream。

Python Code Validation via Jupyter Notebook

Upload .bit, .hwh, .ipynb, samples\_triangular\_wave.txt檔案至Jupyter Notebook:

一張含有 文字, 螢幕擷取畫面, 軟體, 數字 的圖片

自動產生的描述

執行.ipynb檔案，觀察下方輸出之圖形:

一張含有 文字, 行, 圖表, 繪圖 的圖片

自動產生的描述

# FIR with Interface Streaming

## HLS/IP Design

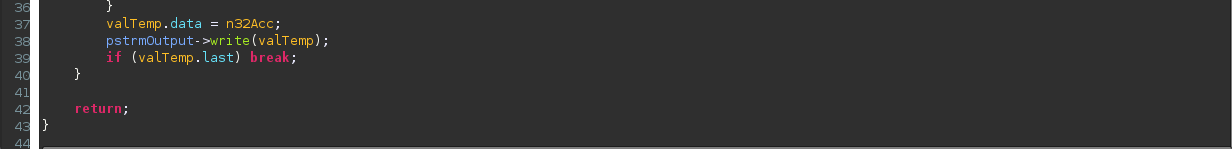
1. FIR.h

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

與FIRMAXI大致相同，唯一不同處在於〝typedef hls::stream<value\_t> stream\_t;〞，hls::stream是用來對流數據結構進行建模，流數據(stream data)是指數據的採樣是按順序進行的，所以不需要地址層面的寫入或讀出等操作。

1. FIR.cpp



與AXI大致相同，〝value\_t valTemp = pstrmInput->read();〞value\_t是存儲數據流內讀取的數據，並從valTemp 中提取整數數據，將其存儲在 n32Temp 變數中。在for迴圈內做一些位移判斷或是乘積運算(FIR運算)，最後將得到的輸出值n32Acc存回valTemp，寫入輸出數據流 pstrmOutput 中。

1. FIRTester.cpp

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體, 陳列 的圖片

自動產生的描述

與FIRN11MAXI的FIRTester大致相同，除了一些部份改成用streaming data方式傳輸，主要還是產生一鋸齒波功能。

此部分主要是學習AXI與Stream的差別，其實現的功能都是FIR運算，還有學會如何不靠指令完成directives的insertion。針對HLS C++語言還是有非常多特別的用法，需要花時間慢慢理解每一行的意義及其目的。

Vivado Implementation

與之前相同，先至settings將vitis\_hls export的IP匯入。

匯入後，因AXI-Lite 與 AXI-Stream 在 processing system block 使用的 port 並不相同，所以須調整HP port設定，如下圖:

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦圖示 的圖片

自動產生的描述

另外，AXI-Master to stream 是用 Xilinx DMA IP 來實現，除了自行設計的IP及ZYNQ7，還需加入2個AXI Direct Memory Access，下圖為其中一個IP(axi\_dma\_out\_0):

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦圖示 的圖片

自動產生的描述

加入完成後即可進行連線(auto connection)，自動連線完成還需手動連線，

1. FIR\_N11\_STRM 的 pstrminput 接到 axi\_dma\_in\_0 的 M\_AXIS\_MM2S

2. FIR\_N11\_STRM 的 pstrmoutput 接到 axi\_dma\_out\_0 的 S\_AXIS\_S2MM

如下圖:

一張含有 文字, 螢幕擷取畫面, 平行, 圖表 的圖片

自動產生的描述

整體block diagram如下圖:

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

Python Code Validation via Jupyter Notebook

Upload .bit, .hwh, .ipynb, samples\_triangular\_wave.txt檔案至Jupyter Notebook:

一張含有 文字, 螢幕擷取畫面, 數字, 軟體 的圖片

自動產生的描述

執行.ipynb檔案，觀察下方輸出之圖形:

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

# Discussion

這次lab主要是學習兩種數據傳輸、儲存的方式，分別為AXI Master Interface與Stream Interface，接下來針對這兩者整理出其差別之處:

**數據交互方式:**

AXI Master Interface通常可以指定特定位址進行讀取及寫入，且可以在暫存器上進行讀寫，而Stream Interface的數據傳輸因為是連續的，沒有位址的概念，可應用在信號處理或是圖像處理等。

**接口配置:**

AXI Master Interface需要配置地址、數據深度(depth)、傳輸協議等參數，而Stream Interface只需配置數據寬度，因為數據是以流(stream)的方式傳輸，故不涉及地址層面的配置參數。

**數據傳輸方式:**

AXI Master Interface支持隨機傳輸，可以對任意點對點之間進行傳輸讀寫，而Stream Interface則以數據流的方式進行傳輸，沒有地址或暫存器的概念，客製程度不如AXI Master Interface，但其運算處理的速度會更為快速。