

# **Jini Intelligent Computing Workbook of Lab. #2**

## Preamble

在Lab. #2範例分為兩個主題，第一類AXI-Master Interface實作範例；第二類Stream Interface實作範例。

第一類實作範例的檔案系統中主要有下列專案目錄：

- hls\_FIRN11MAXI  
Vitis HLS之以AXI-Master Interface為設計FIR原始碼檔案
- vvd\_FIRN11MAXI  
範例乘法器Vivado Design Suite參考檔案
  - design\_1.tcl  
範例FIR之Block Design完成Generate Bitstream後匯出之TCL Script檔
  - MakeBit.bat  
範例FIR完成Generate Bitstream後，將.bit/.hwh拷貝至專案根目錄之批次檔
- ipy\_Multip2Num  
範例FIR系統程式Python原始碼檔及Jupyter Notebook原始碼編輯檔

第二類實作範例的檔案系統中主要有下列專案目錄：

- hls\_FIRN11Stream  
Vitis HLS之以Stream Interface為設計FIR原始碼檔案
- vvd\_FIRN11Stream  
範例乘法器Vivado Design Suite參考檔案
  - design\_1.tcl  
範例FIR之Block Design完成Generate Bitstream後匯出之TCL Script檔
  - MakeBit.bat  
範例FIR完成Generate Bitstream後，將.bit/.hwh拷貝至專案根目錄之批次檔
- ipy\_Multip2Num  
範例FIR系統程式Python原始碼檔及Jupyter Notebook原始碼編輯檔

# 1. FIR with Interface AXI-Master

【施作環境為在使用者PC/laptop/notebook (Windows Base)。】

## 1.1. HLS/IP Design

本Lab.#2實作，HLS開發及驗證同Lab.#1，請自行參照Workbook 1說明步驟操作。  
(如果你在使用Linux，Tester中Linux下不能用 fc，需要用 diff，且需要注意檔案的path 是否正確，以及由於out\_golden.dat檔案來自Windows，所採用的換行符和Linux系統不一樣，需用dos2unix命令轉換。)

## 1.2. Vivado Implementation

### 1.2.1. Create Design Project

同Lab.#1，請自行參照Workbook 1說明步驟操作。

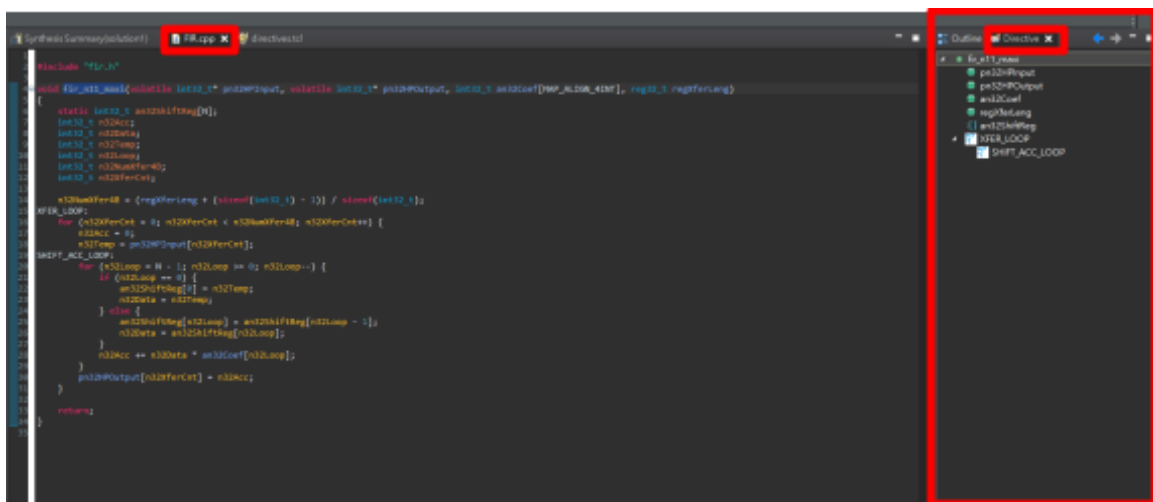
period 設為10ns避免Simulation時出現time violation。

(Review: Parts 使用 **xc7z020clg400-1** → 複製檔案到相對位置 → add new source\*2, testbench\*1, top function → directives → C simulation → C synthesis → Export RTL)

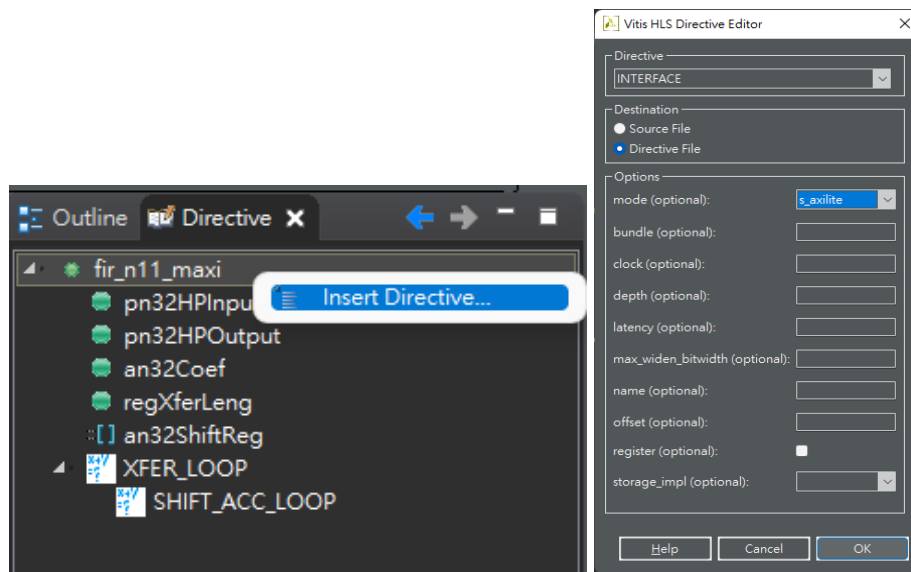
### 1.2.2. Vitis Directive controls

可以從FIR.cpp及solution1裡看出這次是使用directives.tcl檔來設定directives。  
參照solution1檔案夾裡的directives.tcl檔案路徑設定。

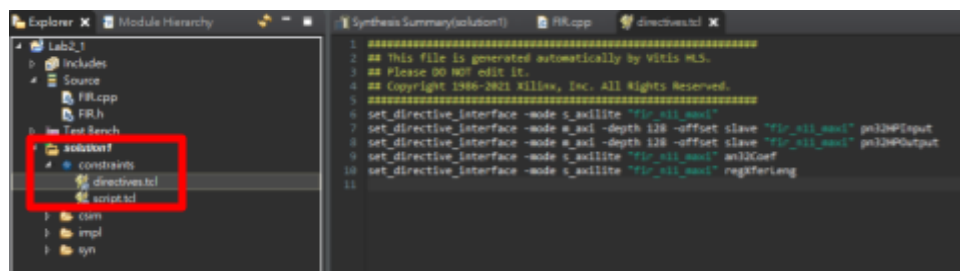
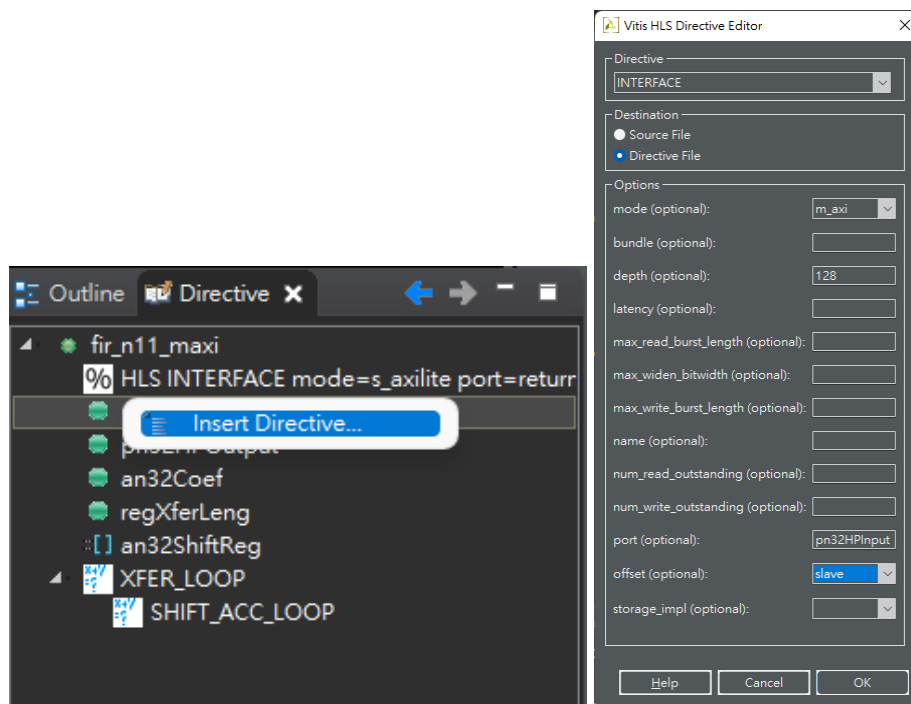
打開FIR.cpp後右側點擊Directive。



首先在directives tab頁面上編輯directives。在top function按滑鼠右鍵插入interface的directive的組態。



相同地，為其他top function引入的引數插入interface的directive的組態。(有的設置要把視窗拉大才看的到)



完成後可以在這裡看directives。

### 1.2.3. Import IP

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

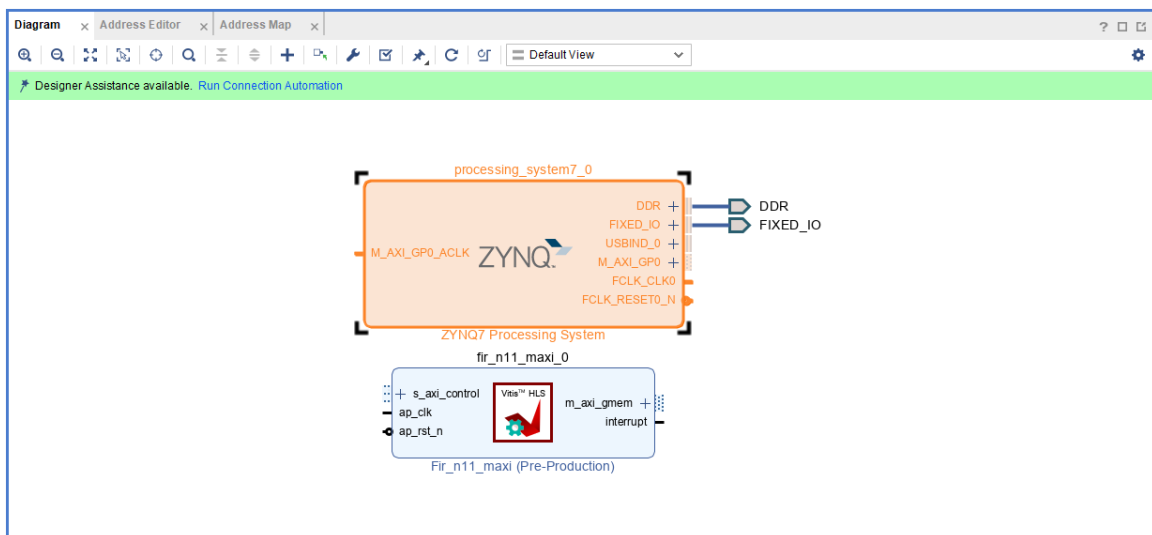
(settings → ip → repository → add ip (vitis project folder) → ok)

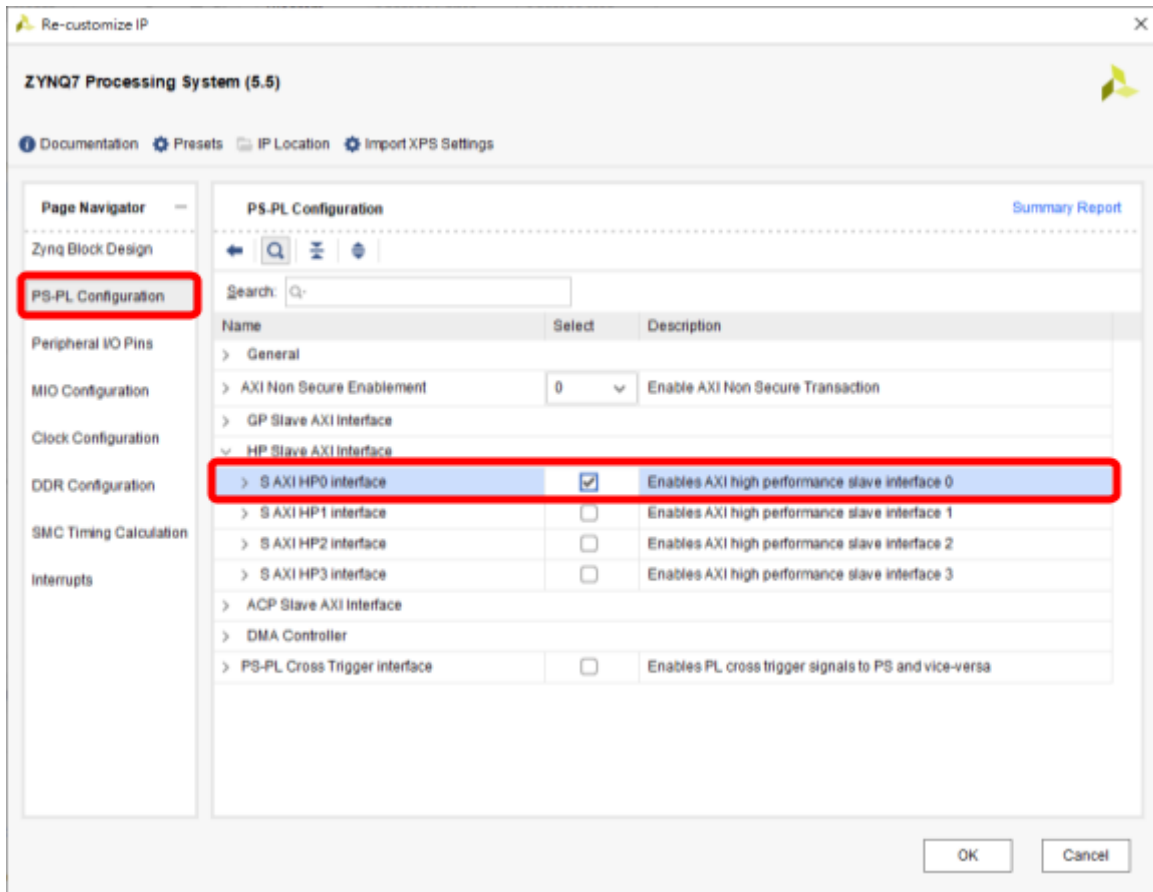
### 1.2.4. Block Design

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

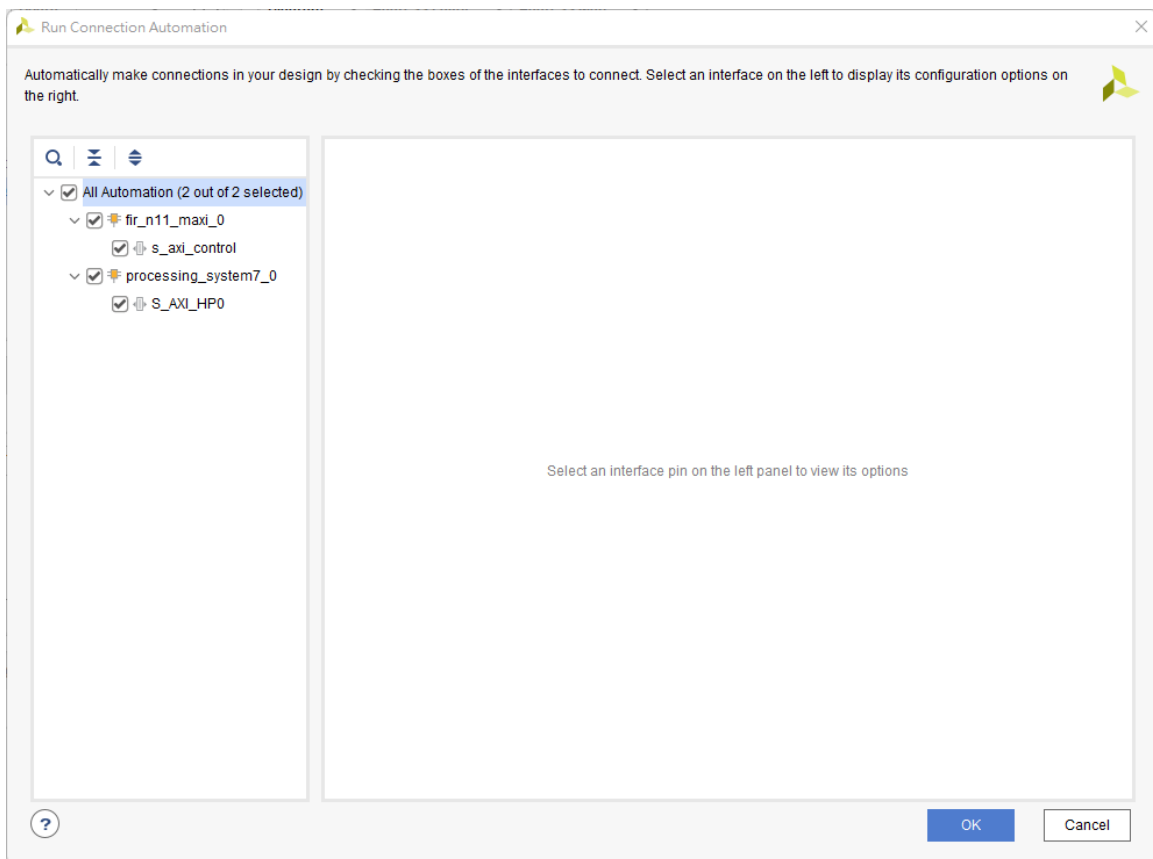
以下僅針對AXI-Master在processing system block的設定補充。(Run block automation會重置ZYNQ的configuration, 一定要先run **block automation**然後再完成ZYNQ的configure)

AXI-Lite與AXI-Master在processing system block使用的port並不相同, 所以在Run Block Automation後用滑鼠左鍵雙擊processing system block, 開啟HP port設定。

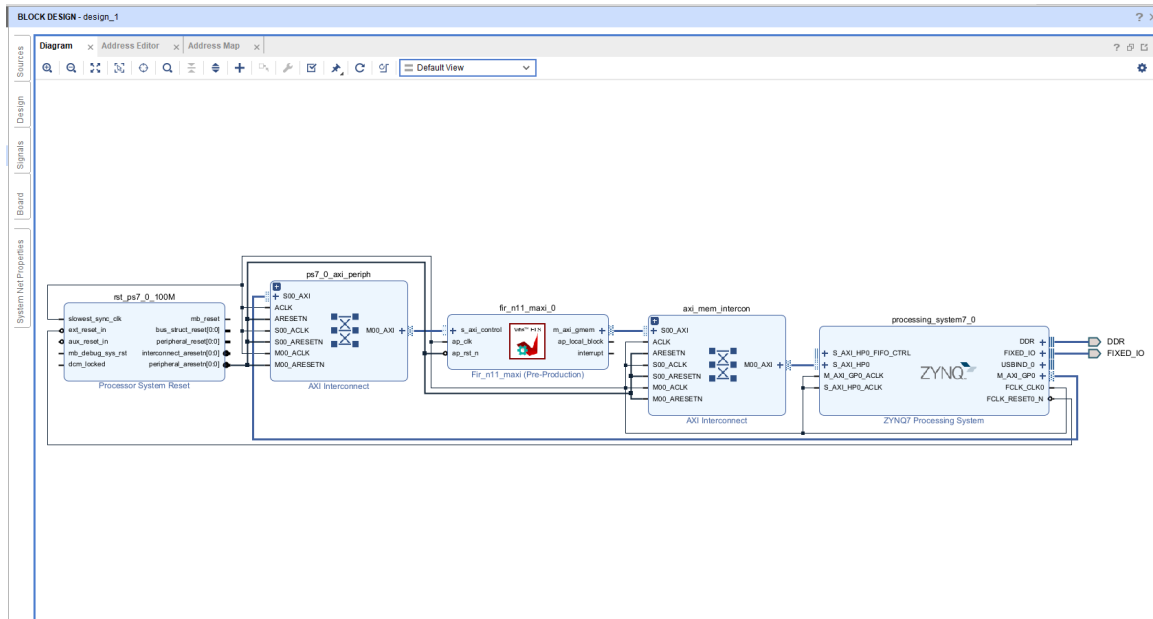




接著按Run Connection Automation。



最終完成的Block Diagram如下圖：



可開啟Address editor檢查Address是否相同。

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/fir_n11_maxi_0					
/fir_n11_maxi_0/Data_m_axi_gmem (64 address bits : 16E)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000_0000_0000	512M	0x0000_0000_1FFF_FFFF
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [ 1G ])					
/fir_n11_maxi_0/s_axi_control	s_axi_control	Reg	0x4000_0000	64K	0x4000_FFFF

## 1.2.5. Synthesis/Placement/Routing/Generate Bit-Stream

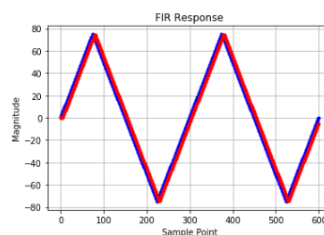
同Lab.#1, 請自行參照Workbook 1說明步驟操作。

(Create HDL Wrapper → Generate bitstream)

## 1.3. Python Code Validation via Jupyter Notebook

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

上傳.bit/.hwh到pynq上, 新建python 3, 記得檢查code裡使用到的檔案路徑。



## 2. FIR with Interface Streaming

【施作環境為在使用者PC/laptop/notebook (Windows Base)。】

### 2.1. HLS/IP Design

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

(Review : Parts : **xc7z020clg400-1** → 複製檔案到相對位置 → Add Source\*2, testbench\*1, Top Function → Set Directives → C Simulation → C Synthesis → Export RTL)

### 2.2. Vivado Implementation

#### 2.2.1. Create Design Project

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

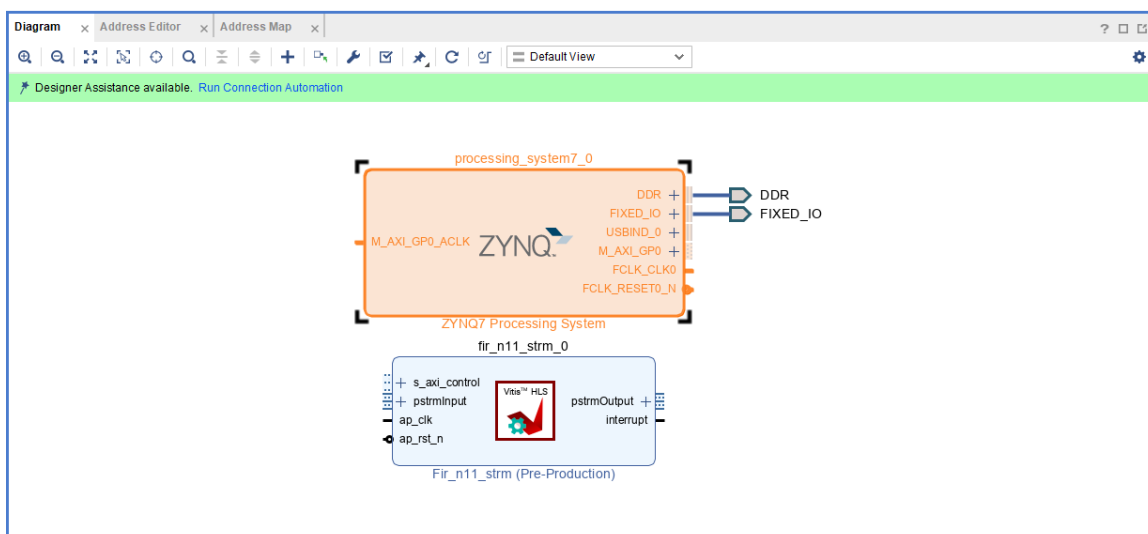
#### 2.2.2. Import IP

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

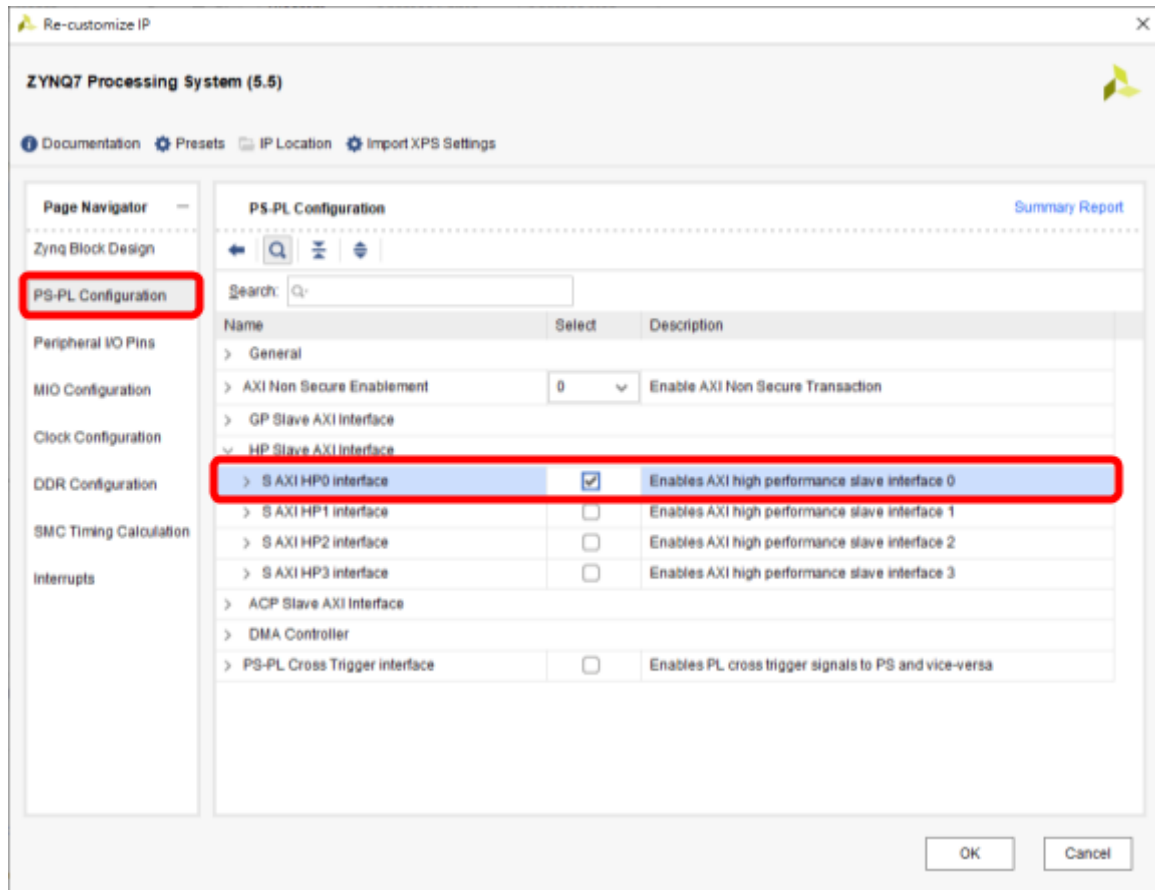
#### 2.2.3. Block Design

同Lab.#1, 請自行參照Workbook 1說明步驟操作。以下僅針對AXI-Stream在processing system block的設定補充。

AXI-Lite與AXI-Stream在processing system block使用的port並不相同, 所以在Run Block Automation後用滑鼠左鍵雙擊processing system block, 必須開啟HP port設定。

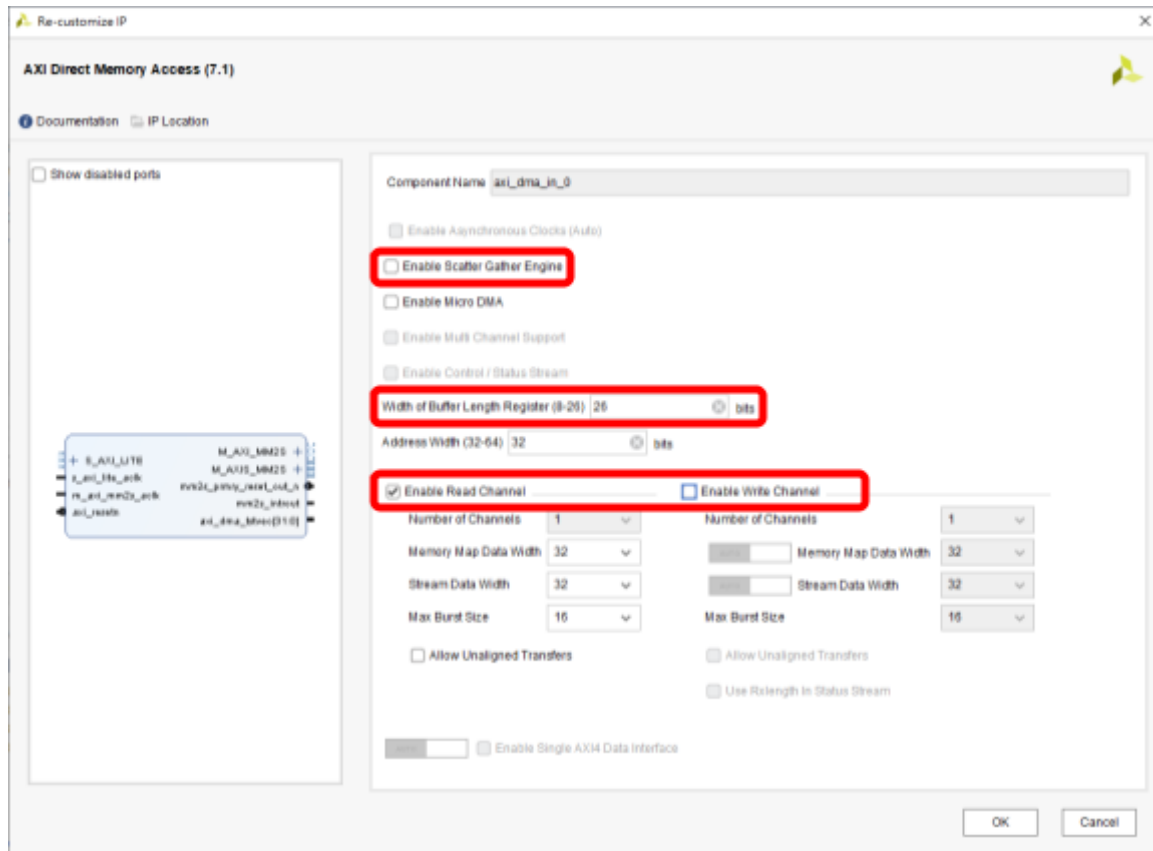




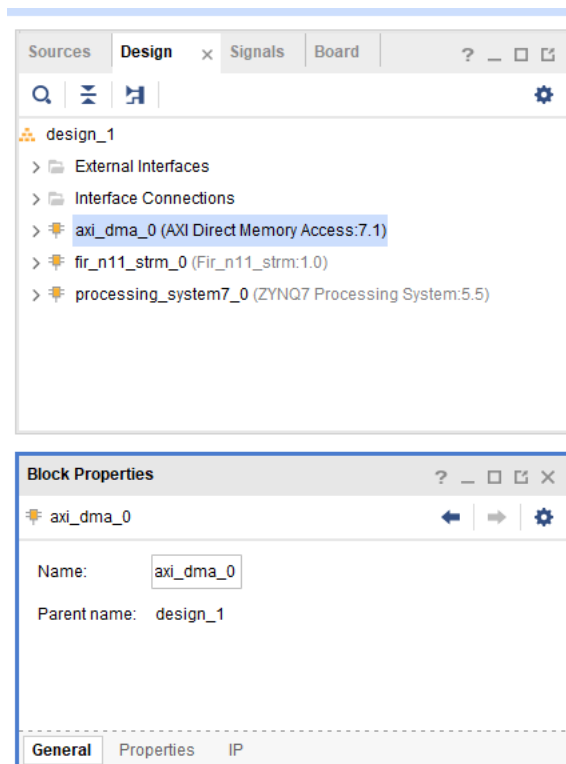


另外，AXI-Master to stream是用Xilinx DMA IP來實現，針對DMA IP需要調整及設定。在Block Design時，除匯入HLS fir\_n11\_strm IP並將IP component加入至Diagram中之外，還需要加入Xilinx DMA IP component，加入後以滑鼠雙擊DMA block。下圖圈選處：

1. 關閉Scatter-Gather Mode
2. 調整Width of Buffer Length Register
3. 選擇單一Read Channel，單一Write Channel，或兩者Read/Write Channel。（由開發者設計需求決定，此lab會需要兩個dma，一個單一Read做為dma in，一個單一Write做為dma out）



dma block名稱可以在左側更改, python code裡有用到dma in跟out的名稱, 記得更改成axi\_dma\_in\_0及axi\_dma\_out\_0。



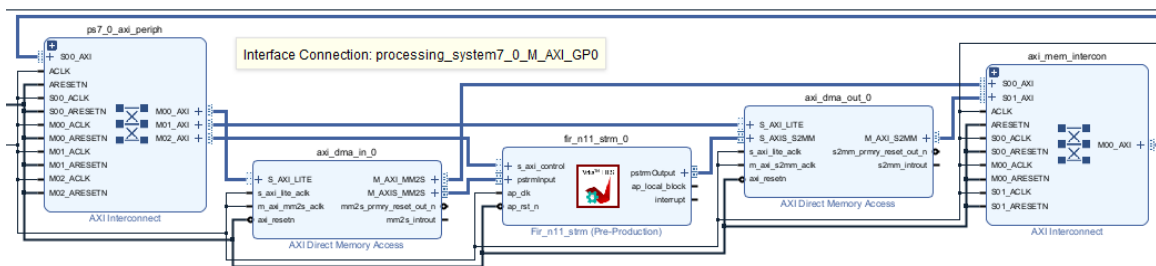
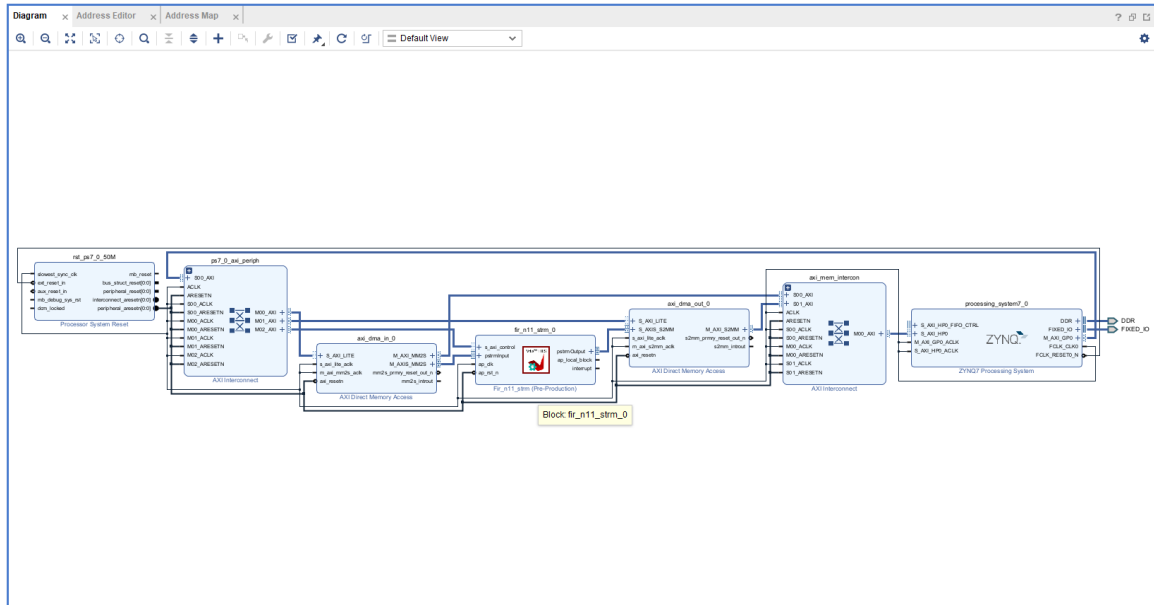
接著按run connection automation到沒有出現為止。

注意這裡要手動連兩條wire：

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

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

最終完成的Block Diagram如下圖：



Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
axi_dma_in_0					
axi_dma_in_0/Data_MM2S (32 address bits : 4G)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
axi_dma_out_0					
axi_dma_out_0/Data_S2MM (32 address bits : 4G)					
/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
processing_system7_0					
processing_system7_0/Data (32 address bits : 0x40000000 [ 1 G ])					
/axi_dma_in_0/S_AXI_LITE	S_AXI_LITE	Reg	0x41B0_0000	64K	0x41E0_FFFF
/axi_dma_out_0/S_AXI_LITE	S_AXI_LITE	Reg	0x41E1_0000	64K	0x41F1_FFFF
/fir_n11_strm_0/s_axi_control	s_axi_control	Reg	0x4000_0000	64K	0x4000_FFFF

## 2.2.4. Synthesis/Placement/Routing/Generate

### Bit-stream

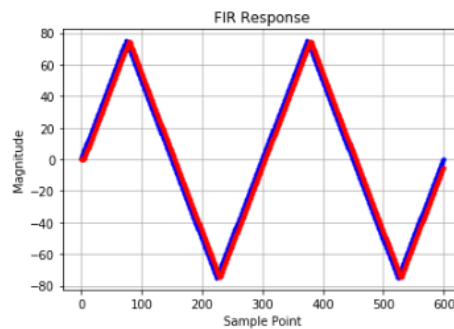
同Lab.#1, 請自行參照Workbook 1說明步驟操作。

## 2.3. Python Code Validation via Jupyter Notebook

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

上傳.bit/.hwh到pynq上, 新建python 3, 記得檢查code裡使用到的檔案路徑。

```
Entry: /usr/lib/python3/dist-packages/ipykernel_launcher.py  
System argument(s): 3  
Start of "/usr/lib/python3/dist-packages/ipykernel_launcher.py"  
Kernel execution time: 0.0010209083557128906 s
```

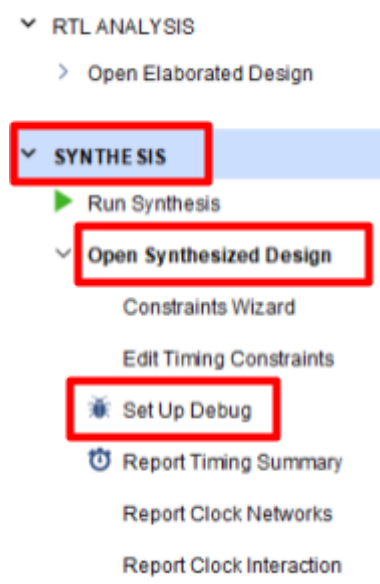


### 3. 附錄: Integrated Logic Analyzer (ILA)

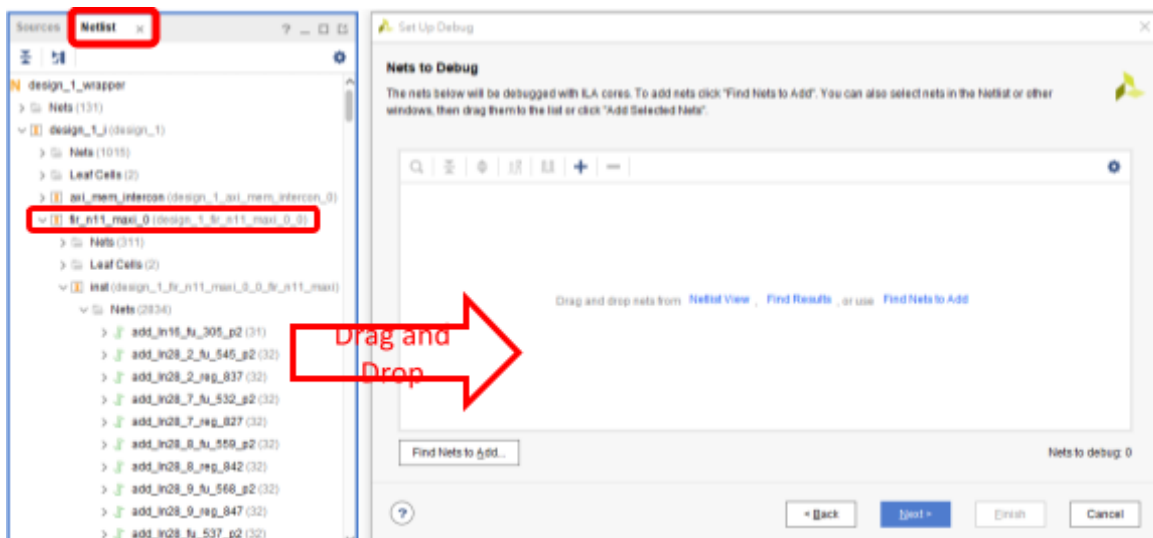
【施作環境為在使用者PC/laptop/notebook (Windows Base)。】


ILA是內嵌的邏輯分析儀工具。現在示範如何在Generate Bitstream File前，將Nets嵌入到bitstream file裡。以專案vvd\_FIRN11MAXI為範例。

1. 首先在Vivado專案IDE畫面，在專案管理點擊Set Up Debug選項。










2. 接下來會彈出一個加Nets的對話盒，以拖拉形式直接將在Netlist Tab中的訊號名稱加進對話盒中，下一步後取樣深度可以調整到大的檔位65536。







Set Up Debug

### Nets to Debug


The nets below will be debugged with ILA cores. To add nets click "Find Nets to Add". You can also select nets in the Netlist or other windows, then drag them to the list or click "Add Selected Nets".

Name	Clock Domain	Driver Cell	P
>  design_1_i/fir_n11_maxi_0/inst/m_axi_gmem_ARADDR (62)	design_1_i/processing_system7_0/inst/FCLK_CLK0	FDRE	C
 design_1_i/fir_n11_maxi_0/inst/ap_start	design_1_i/processing_system7_0/inst/FCLK_CLK0	FDRE	C
 design_1_i/fir_n11_maxi_0/inst/ap_done	design_1_i/processing_system7_0/inst/FCLK_CLK0	LUT2	C

Find Nets to Add...

Nets to debug: 64




< Back

Next >

Finish

Cancel


Set Up Debug


### ILA Core Options

Choose features for the ILA debug cores.

Sample of data depth: 65536

1024  
2048  
4096  
8192  
16384  
32768  
65536  
131072

☐ Capture control  
☐ Advanced trigger



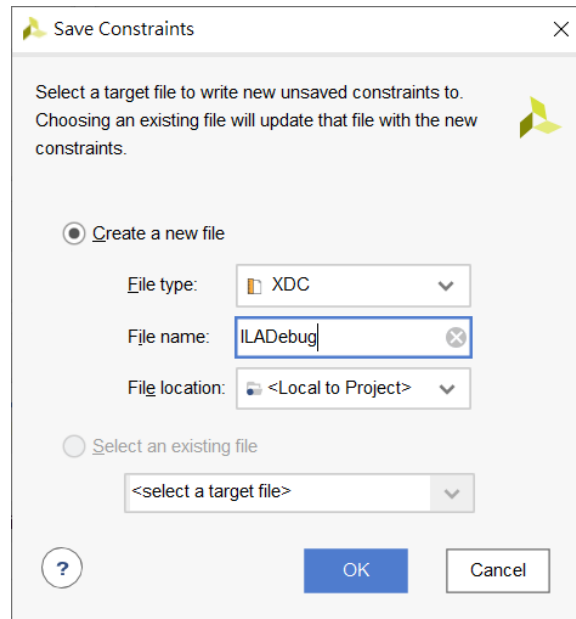
< Back

Next >

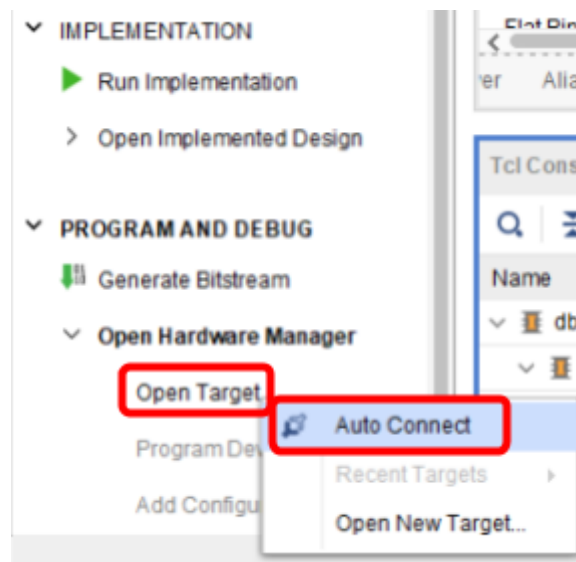
Finish

Cancel

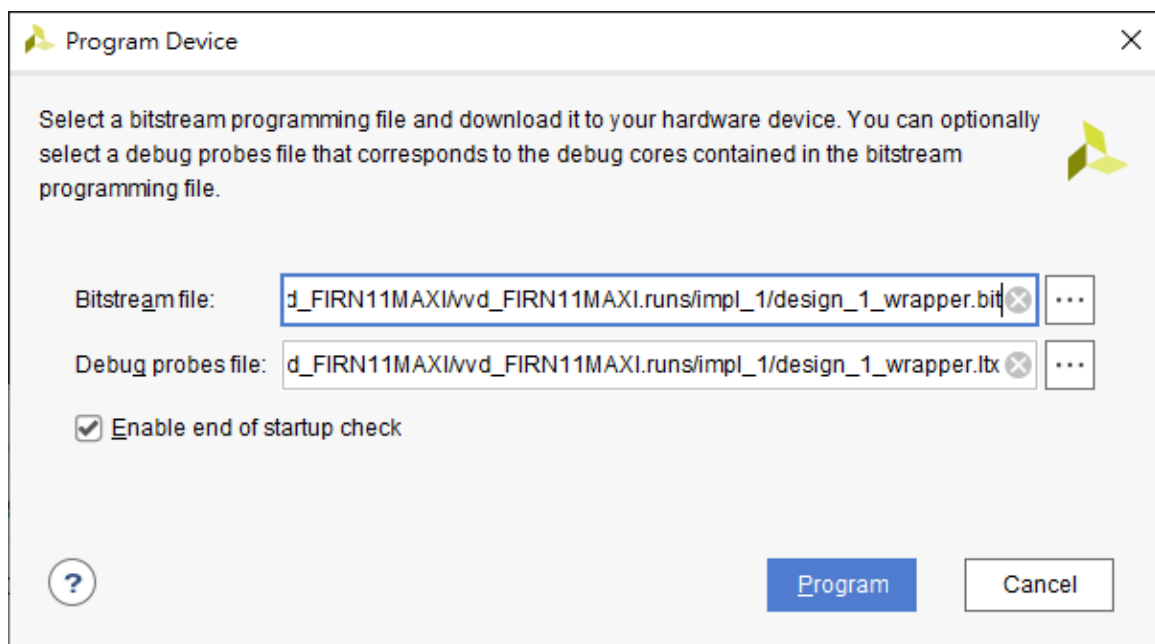
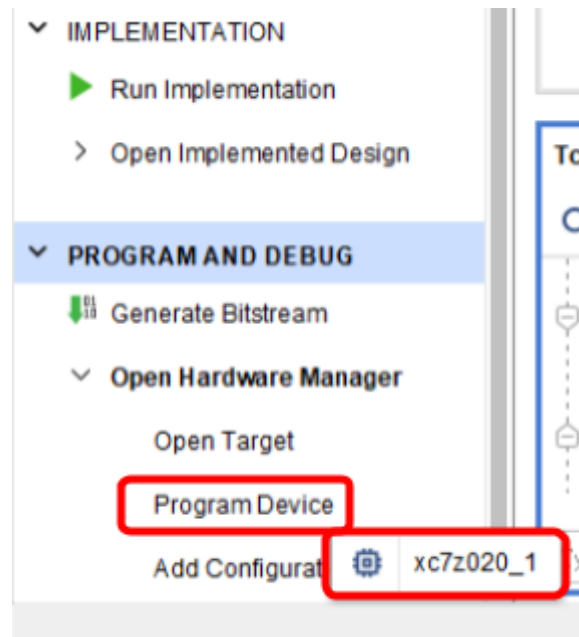
3. 完成後就可以在專案管理點擊Generate Bitstream選項或由工具列按下Generate Bitstream按鍵。彈出Save Constraints的對話盒，可選擇鍵入檔名另存新檔。



4. 開始產生Bitstream。專案管理點擊Generate Bitstream選項或由工具列按下Generate Bitstream按鍵。
5. 將.bit/.hwh藉由MobaXterm/Samba傳送到PYNQ-Z2上存放的目錄下。
6. 在Vivado專案IDE畫面，在專案管理點擊Open Target選項，在浮動視窗選擇Auto Connect。此時Vivado IDE會完成與PYNQ-Z2的USB JTAG的連接動作。

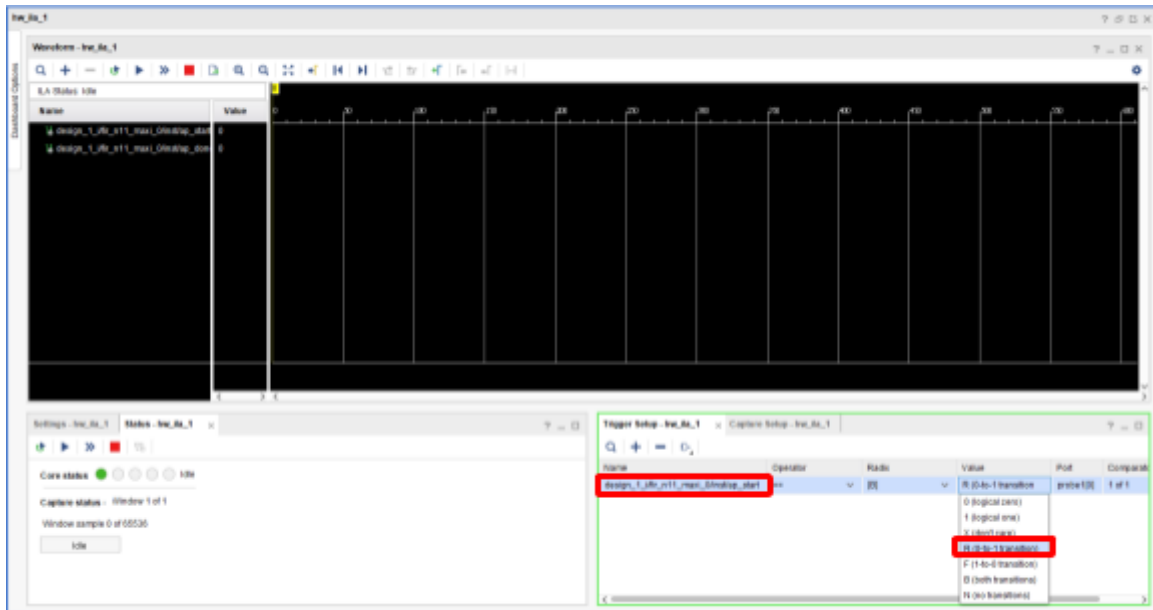


7. 在Vivado專案IDE畫面，在專案管理點擊Program Device選項，在浮動視窗選擇PYNQ-Z2的model name device。然後會跳出Program Device的對話盒，按壓Program鍵將Bitstream燒錄到PYNQ-Z2的FPGA。





- 在Vivado專案IDE畫面下將hw\_ila\_1子視窗放大。設定好ILA觸發訊號，此設定觸發ap\_start由0到1的上升緣訊號。



- 將Jupyter Notebook上的host program於Overlay後設一個中斷點，如下圖。

```
if __name__ == "__main__":
    print("Entry:", sys.argv[0])
    print("System argument(s):", len(sys.argv))

    print("Start of \"" + sys.argv[0] + "\"")

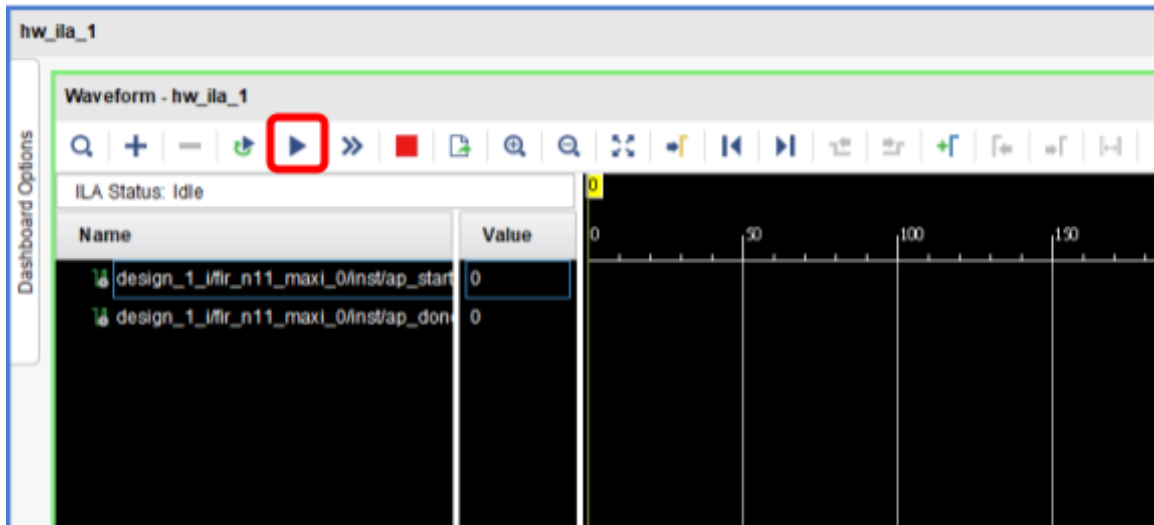
    ol = Overlay("/home/xilinx/HLS_Summer_Course/Lab2/FIRN11MAXI/FIRN11MAXI.bit")
```

```
In [ ]: ipFIRN11 = ol.fir_n11_maxi_0

        fiSamples = open("samples_triangular_wave.txt", "r+")
        numSamples = 0
        line = fiSamples.readline()
        while line:
            numSamples = numSamples + 1
            line = fiSamples.readline()
```

- 先執行完Jupyter Notebook在Overlay之前的host program。

11. 在Vivado專案IDE畫面下點選功能鍵開始觸發的按鍵。



12. 再執行Jupyter Notebook在Overlay之後的host program。運行過程中Vivado ILA會偵測觸發訊號，如果符合觸發條件，所有訊號行為會在IDE畫面下將Waveform子視窗顯示。

