

# **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。

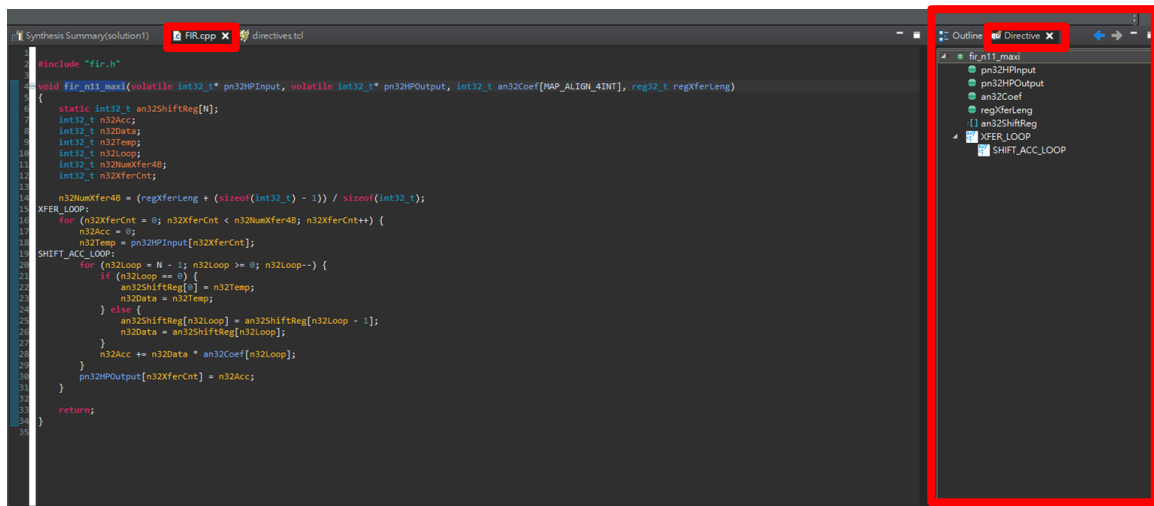
(Review：Parts 使用 **xc7z020c1g400-1** → 複製檔案到相對位置 → add new source\*2, testbench\*1, top function → directives → C simulation → C synthesis → Export RTL)

### 1.2.2. Vitis Directive controls

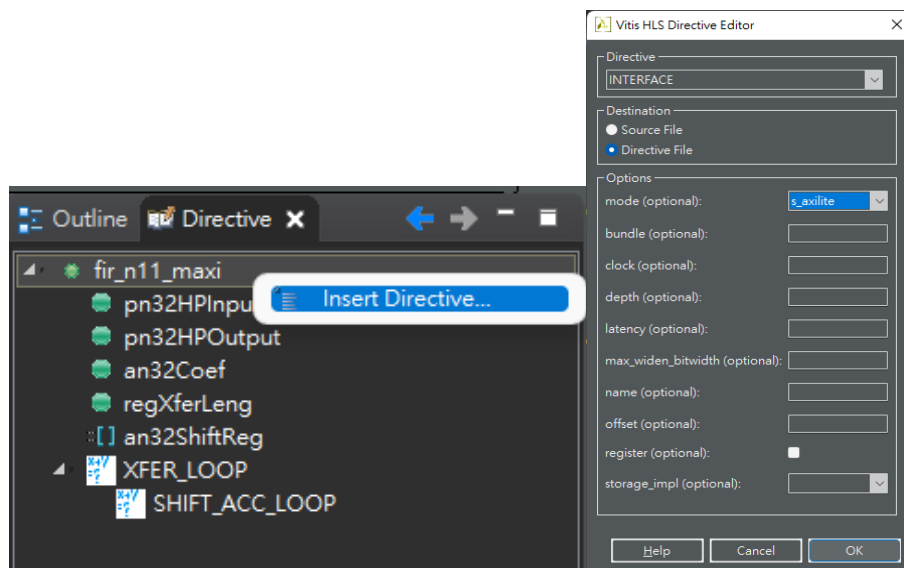
可以從 FIR.cpp 及 solution1 裡看出這次是使用 directives.tcl 檔來設定 directives。

參照 solution1 檔案夾裡的 directives.tcl 檔案路徑設定。

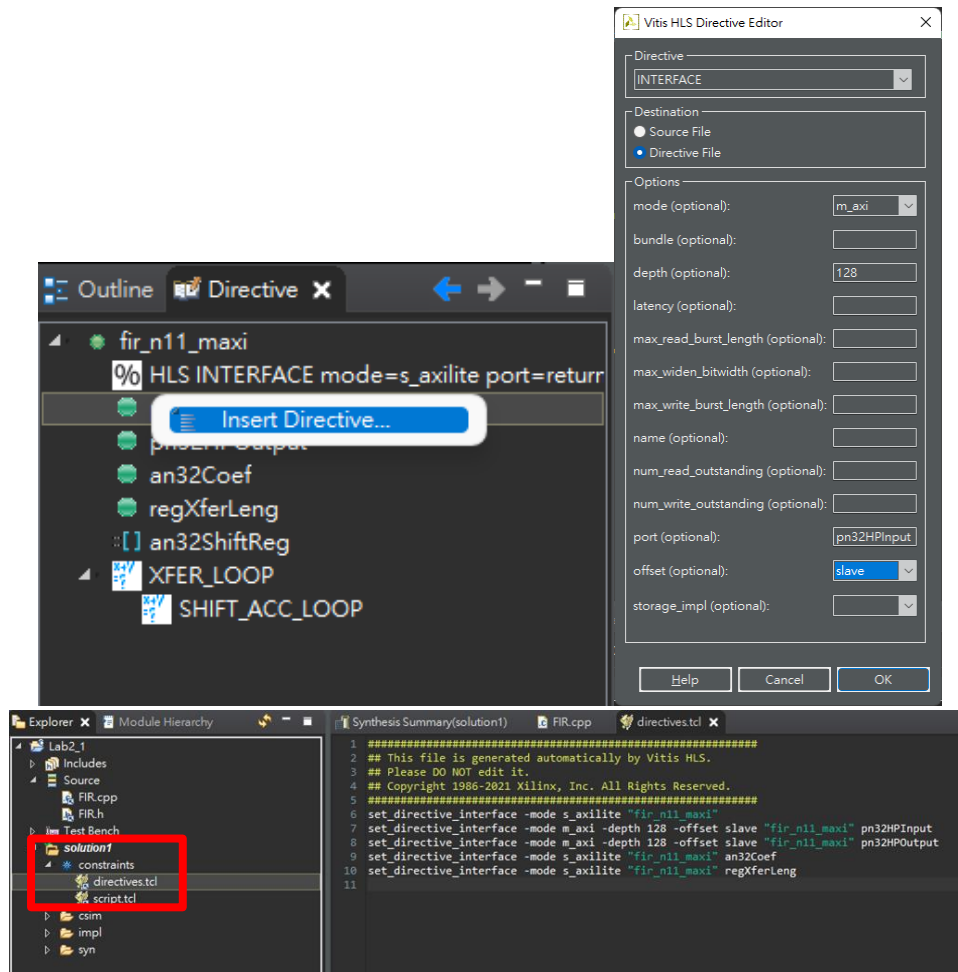
打開 FIR.cpp 後右側點擊 Directive。



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) (depth 請設定超過 600)



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。

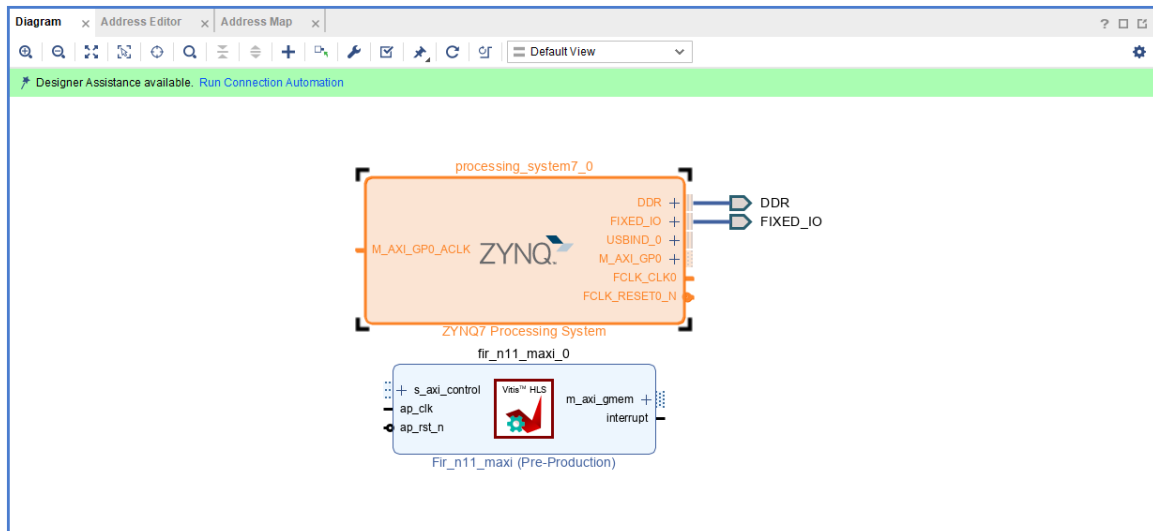
(settings → ip → repository → add ip (vitis project folder) → ok)

### 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。



Re-customize IP

### ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator

- Zynq Block Design
- PS-PL Configuration**
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

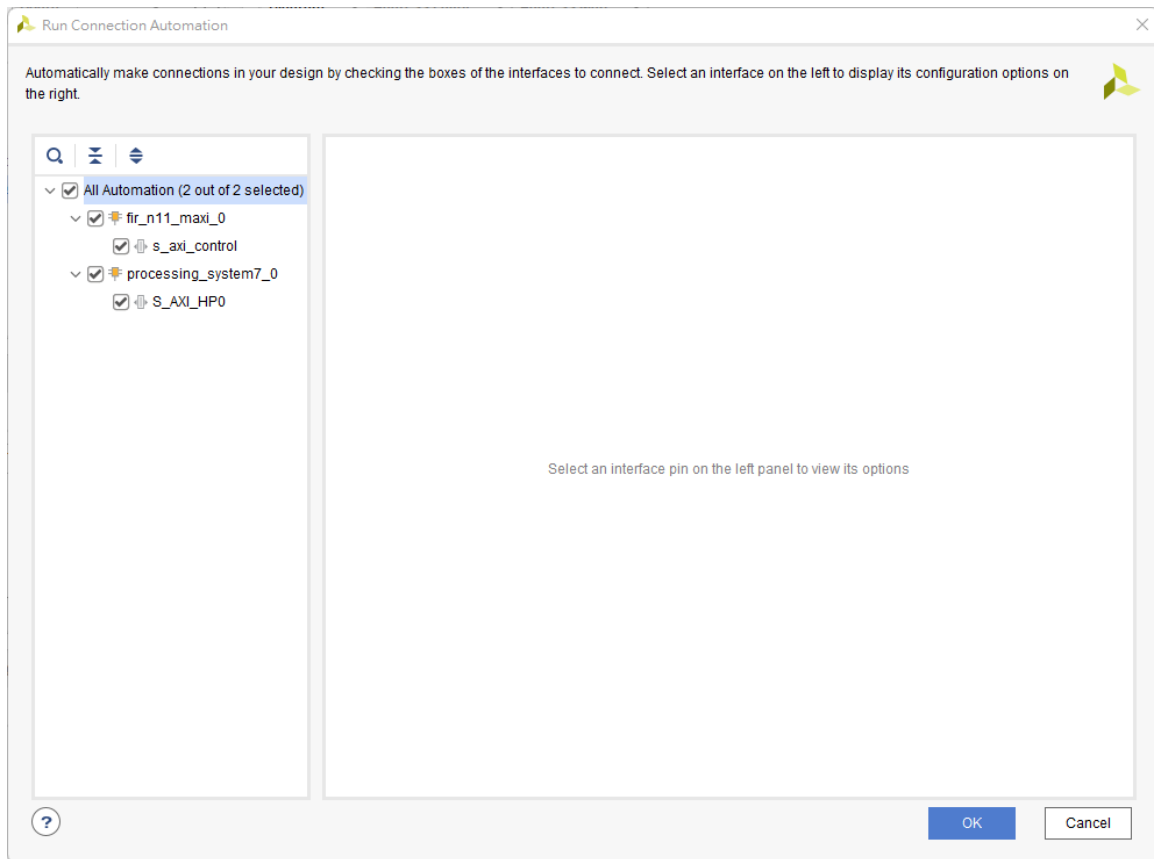
#### PS-PL Configuration

Search: Q-

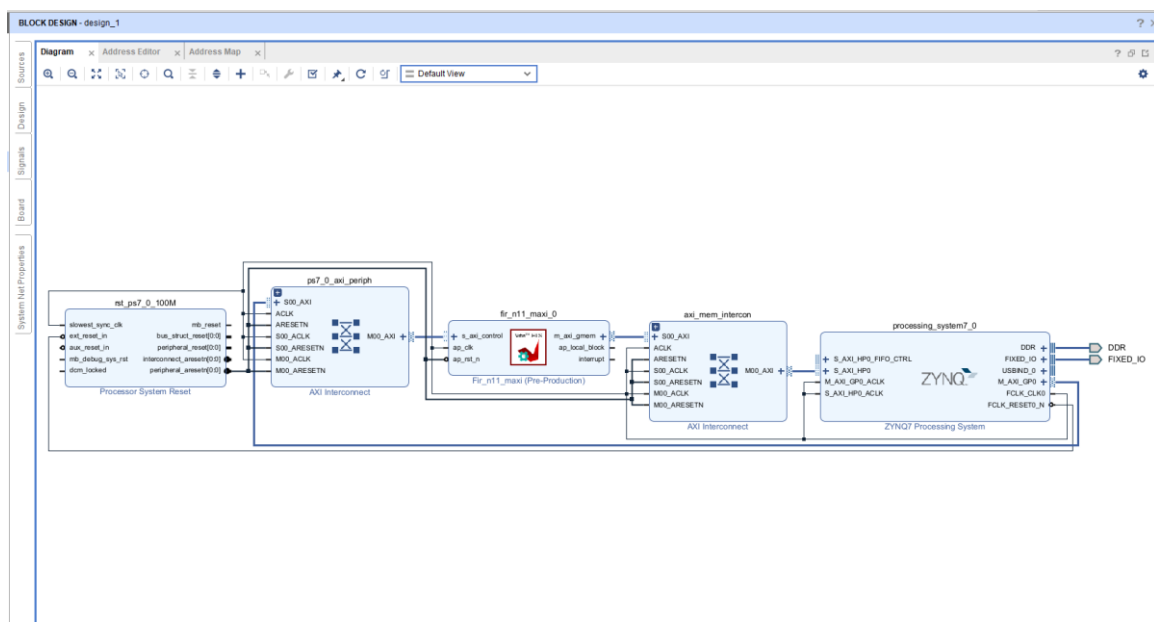
Name	Select	Description
General		
> AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
> GP Slave AXI Interface		
> HP Slave AXI Interface		
> S AXI HP0 interface	<input checked="" type="checkbox"/>	Enables AXI high performance slave interface 0
> S AXI HP1 interface	<input type="checkbox"/>	Enables AXI high performance slave interface 1
> S AXI HP2 interface	<input type="checkbox"/>	Enables AXI high performance slave interface 2
> S AXI HP3 interface	<input type="checkbox"/>	Enables AXI high performance slave interface 3
> ACP Slave AXI Interface		
> DMA Controller		
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger signals to PS and vice-versa

OK Cancel

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。

Diagram x Address Editor x Address Map x						
<input checked="" type="checkbox"/> Assigned (2) <input checked="" type="checkbox"/> Unassigned (0) <input checked="" type="checkbox"/> Excluded (0) <input type="button" value="Hide All"/>						
Name	Interface	Slave Segment	Master Base Address	Range	Master High Address	
Network 0						
f1r_n11_maxi_0						
f1r_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 ])						
f1r_n11_maxi_0/s_axi_control	s_axi_control	Reg	0x4000_0000	64K	0x4000_FFFF	

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

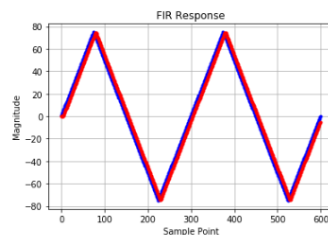
同 Lab.#1，請自行參照 Workbook 1 說明步驟操作。

(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。



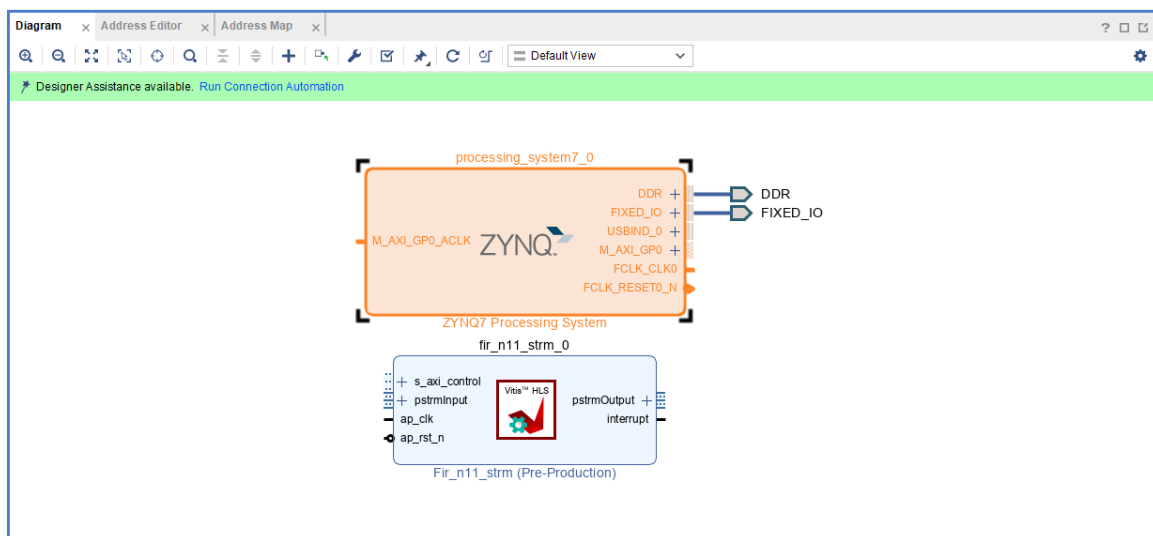
## 2.2.2. Import IP

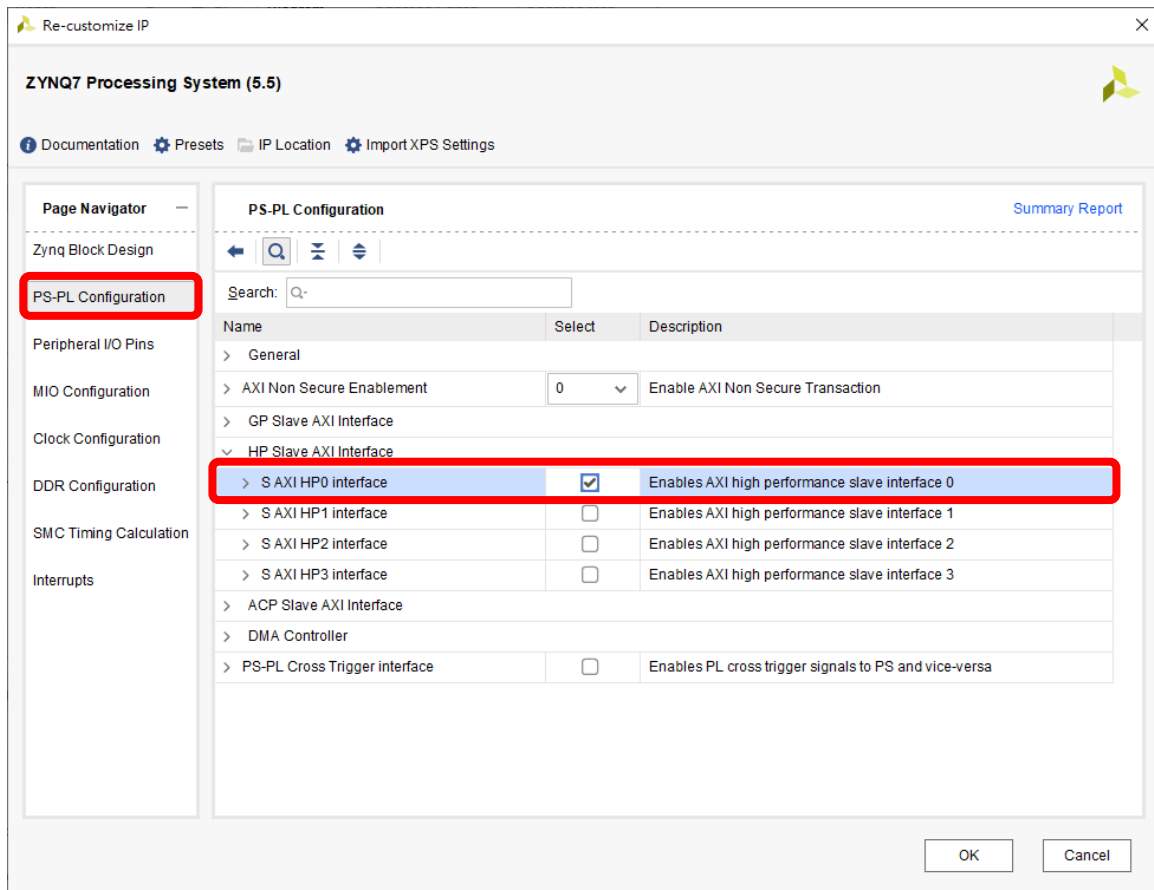
同 Lab.#1，請自行參照 Workbook 1 說明步驟操作。

## 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。

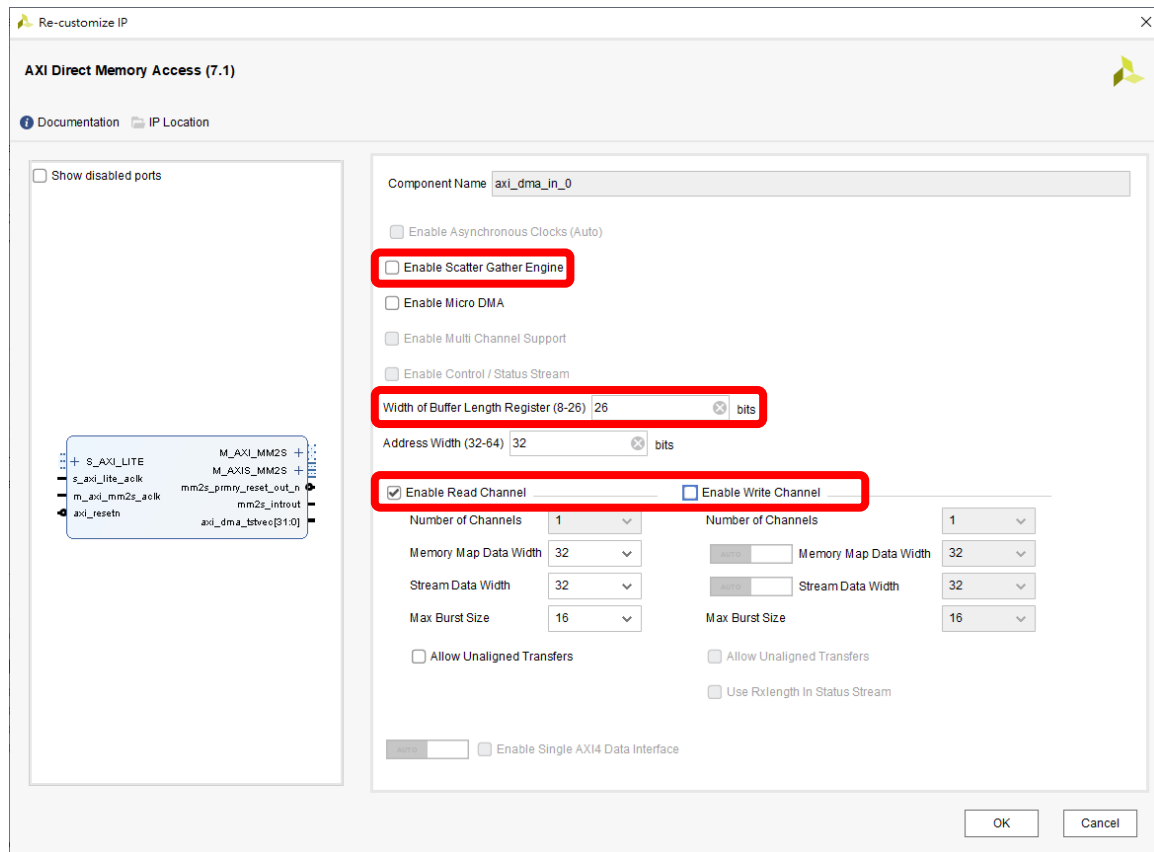
AXI-Lite 與 AXI-Stream 在 processing system block 使用的 port 並不相同，所以在 Run Block Automation 後用滑鼠左鍵雙擊 processing system block，必須開啟 HP port 設定。



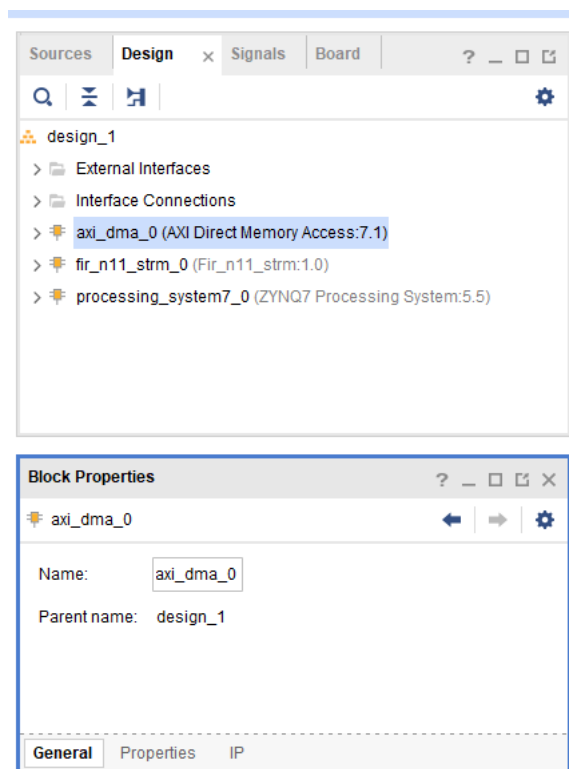


另外，AXI-Master to stream 是用 Xilinx DMA IP 來實現，針對 DMA IP 需要調整及設定。在 Block Design 時，除匯入 HLS fir\_n11\_strm IP 並將 IP component 加入至 Diagram 中之外，還需要加入 Xilinx DMA IP component，加入後以滑鼠雙擊 DMA block。下圖圈選處：

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。



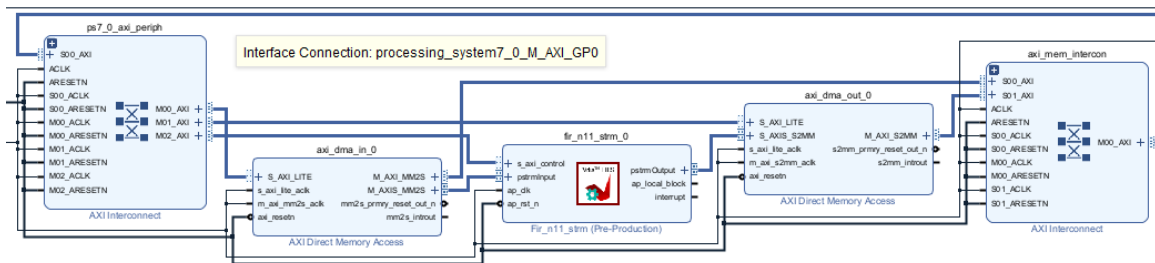
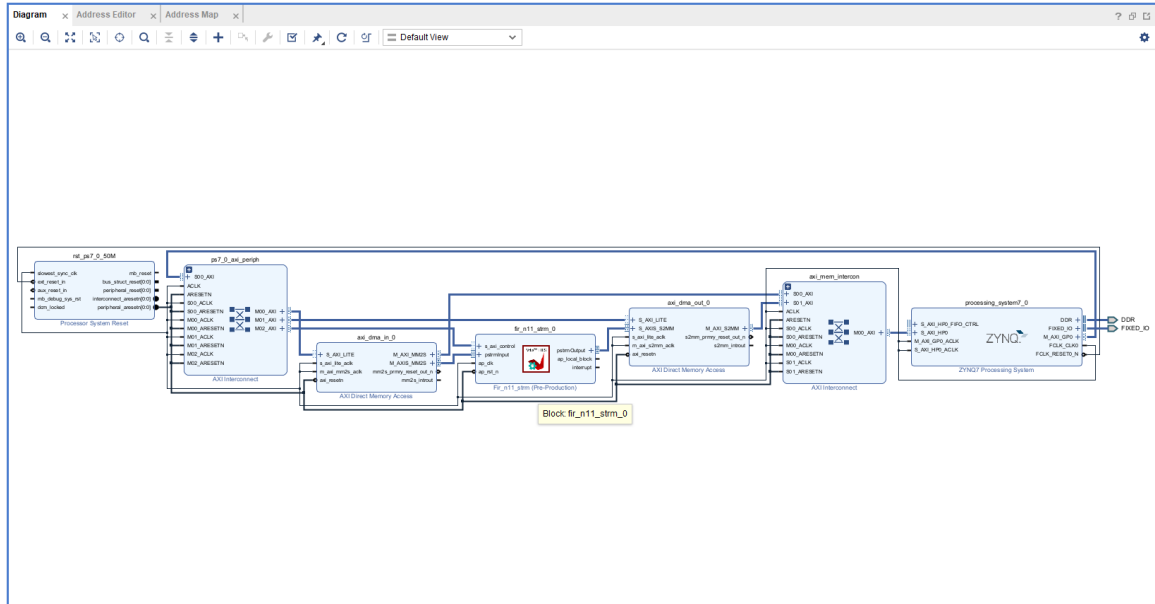
接著按 run connection automation 到沒有出現為止。

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 [ 1G ])					
/axi_dma_in_0/S_AXI_LITE	S_AXI_LITE	Reg	0x41E0_0000	64K	0x41E0_FFFF
/axi_dma_out_0/S_AXI_LITE	S_AXI_LITE	Reg	0x41E1_0000	64K	0x41E1_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

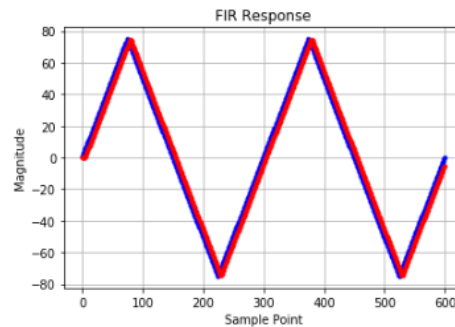
同 Lab.#1，請自行參照 Workbook 1 說明步驟操作。

## 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
```

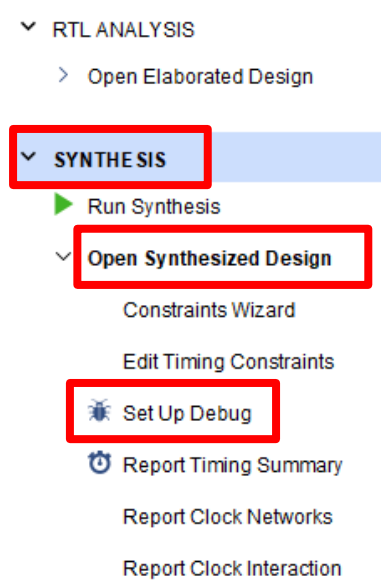


### 3. 附錄：Integrated Logic Analyzer (ILA)

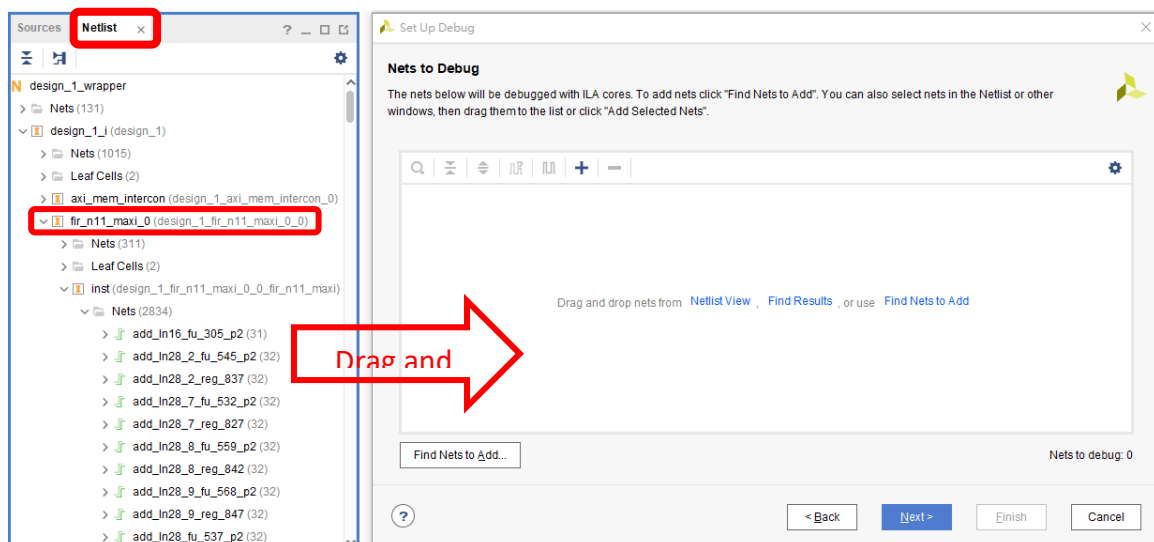
【施作環境為在使用者 PC/laptop/notebook (Windows Base) 。】


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	D
design_1_i/fir_n11_maxi_0/inst/ap_start	design_1_i/processing_system7_0/inst/FCLK_CLK0	FDRE	D
design_1_i/fir_n11_maxi_0/inst/ap_done	design_1_i/processing_system7_0/inst/FCLK_CLK0	LUT2	D

Nets to debug: 64


Set Up Debug

### ILA Core Options

Choose features for the ILA debug cores.

Sample of data depth:

65536

▼

Input pipe stages:

1024

2048

4096

8192

16384

32768

65536

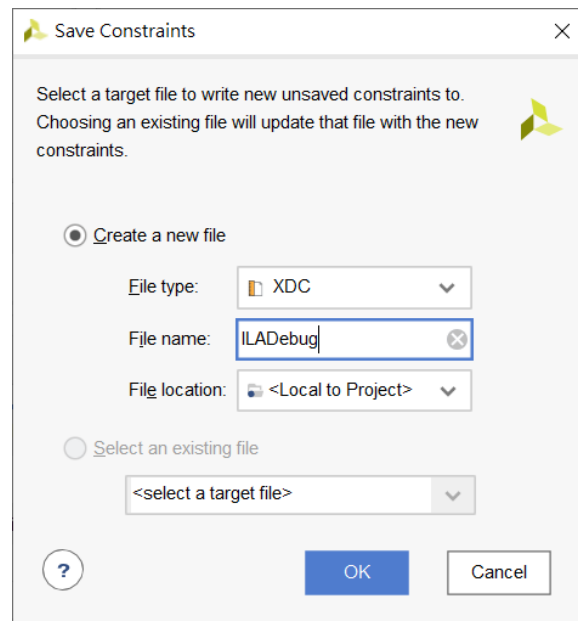
131072

Trigger and Storage Size:

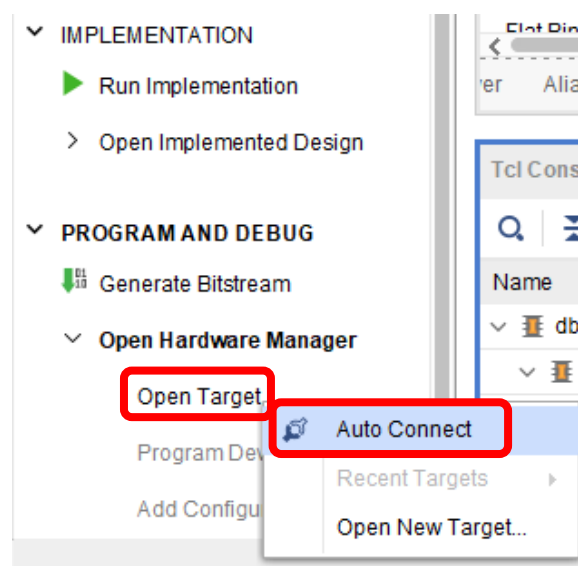
☐ Capture control

☐ Advanced trigger

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** 按鈕。彈出 **Save Constraints** 的對話盒，可選擇鍵入檔名另存新檔。

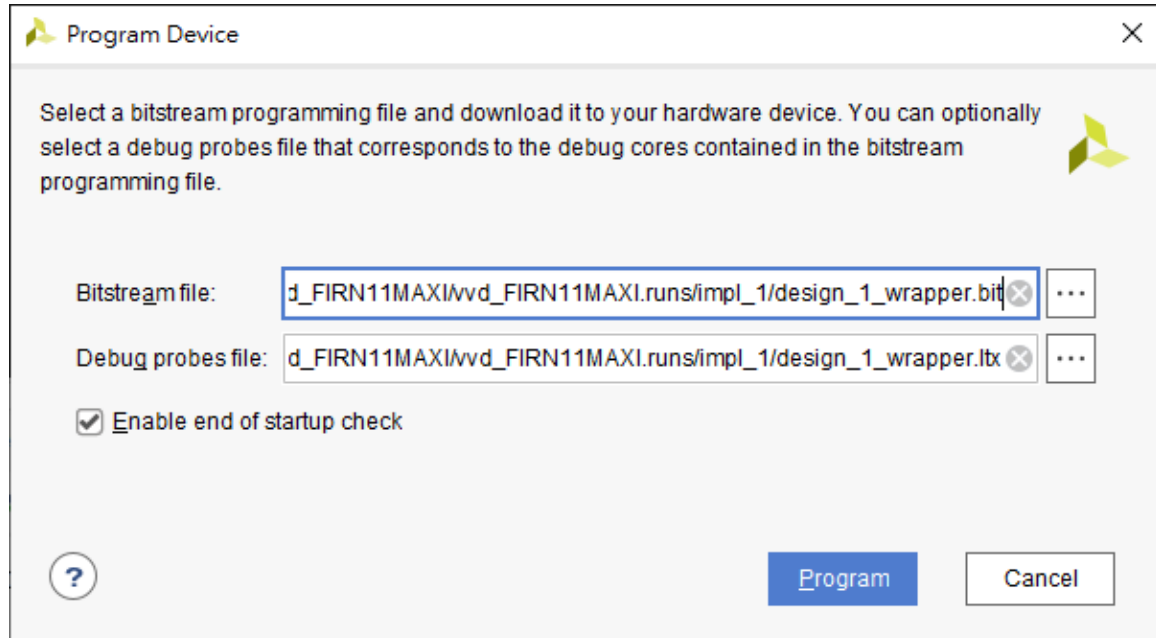
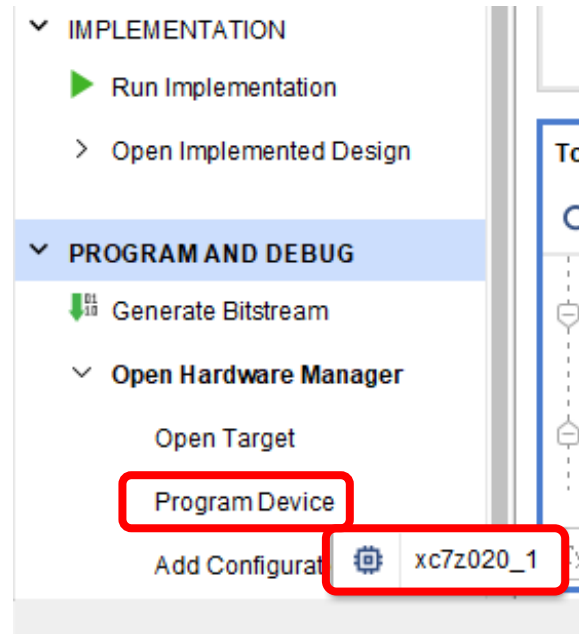


4. 開始產生 Bitstream。專案管理點擊 **Generate Bitstream** 選項或由工具列按下 **Generate Bitstream** 按鈕。
5. 將 .bit/.hwh 藉由 MobaXterm/Samba 傳送到 PYNQ-Z2 上存放的目錄下。
6. 在 Vivado 專案 IDE 畫面，在專案管理點擊 **Open Target** 選項，在浮動視窗選擇 **Auto Connect**。此時 Vivado IDE 會完成與 PYNQ-Z2 的 USB JTAG 的連接動作。

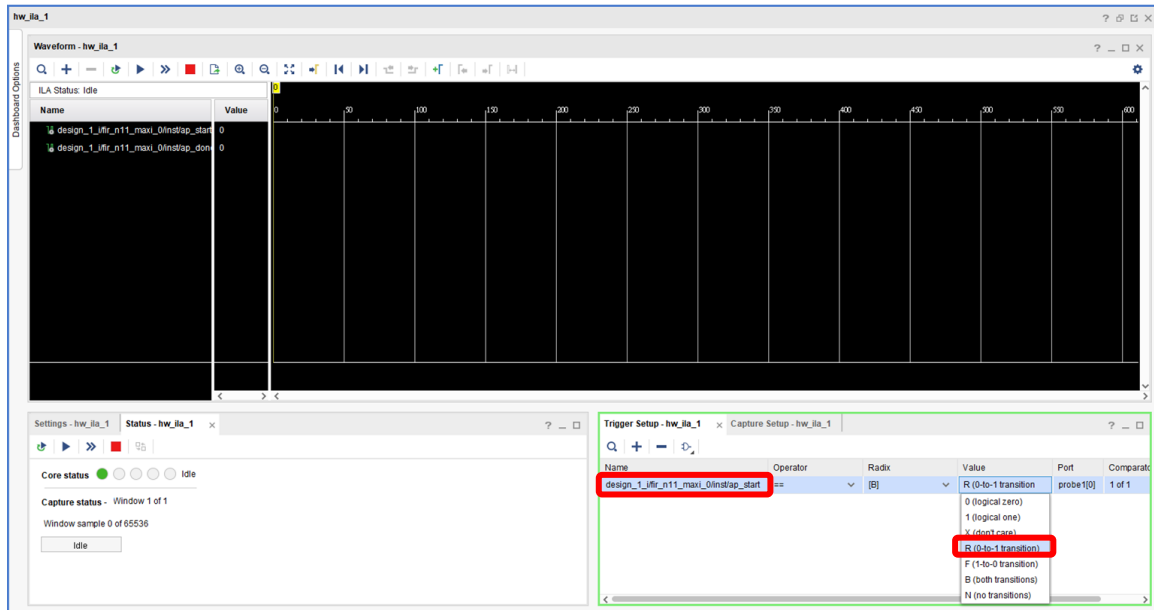




7. 在 Vivado 專案 IDE 畫面，在專案管理點擊 Program Device 選項，在浮動視窗選擇 PYNQ-Z2 的 model name device。然後會跳出 Program Device 的對話盒，按壓 Program 鍵將 Bitstream 燒錄到 PYNQ-Z2 的 FPGA。



- 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] + "\"")

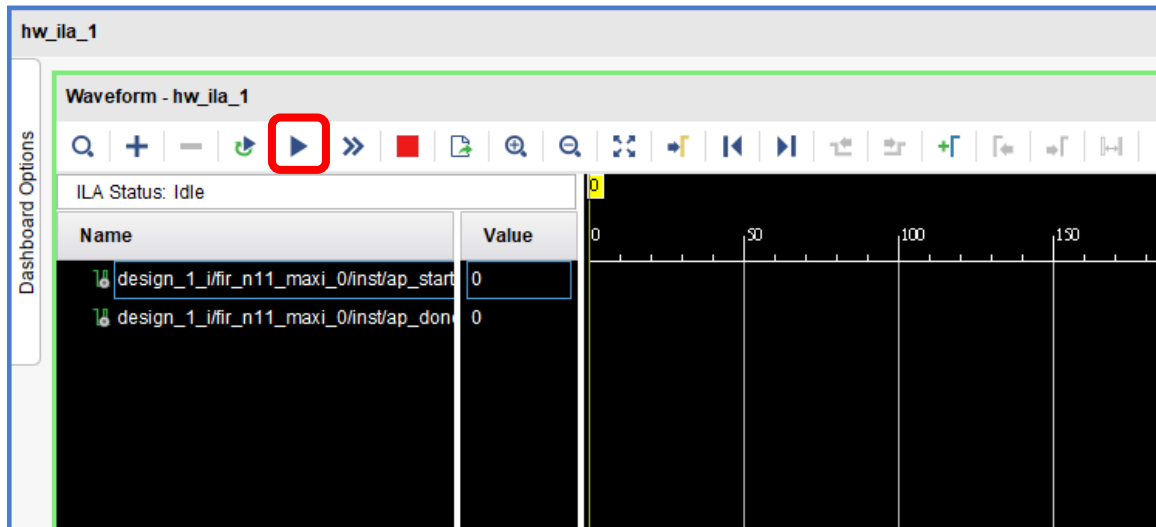
    ol = Overlay("/home/xilinx/HLS_Summer_Course/Lab2/FIRN11MAXI/FIRN11MAXI.bit")
```

```
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。

