



Jini Intelligent Computing

Workbook of Lab. #2

For KV260



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



與PYNQ-Z2不同的步驟會以紅底HIGHLIGHT

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。

Parts 使用 **xck26-sfvc784-2LV-c** → add source*2, add testbench*1, add top function → directives → C simulation → C synthesis → Export RTL

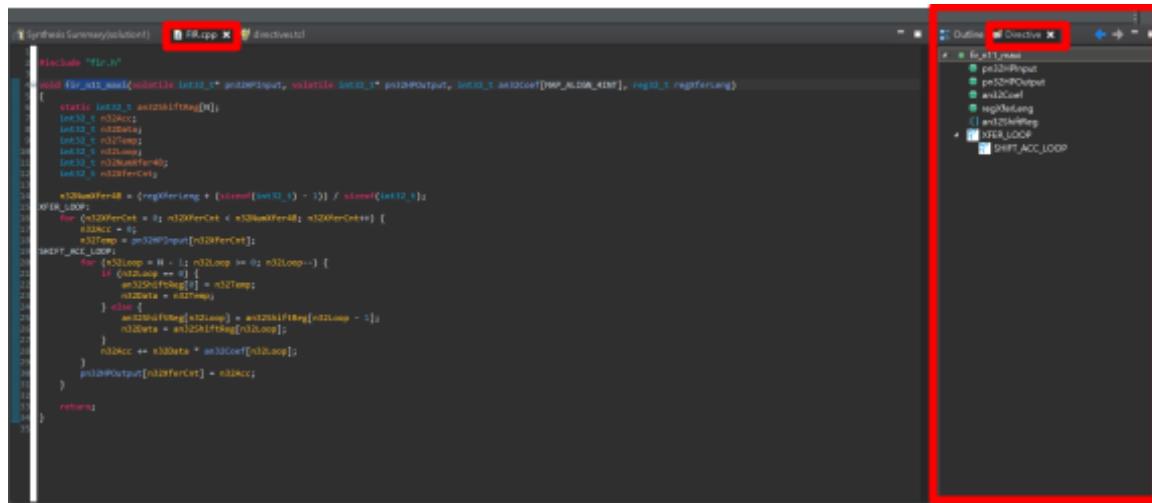
(雖然VITIS HLS有KV260的Board, 但會有error, 無法跑C sim 及 C syn)

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。

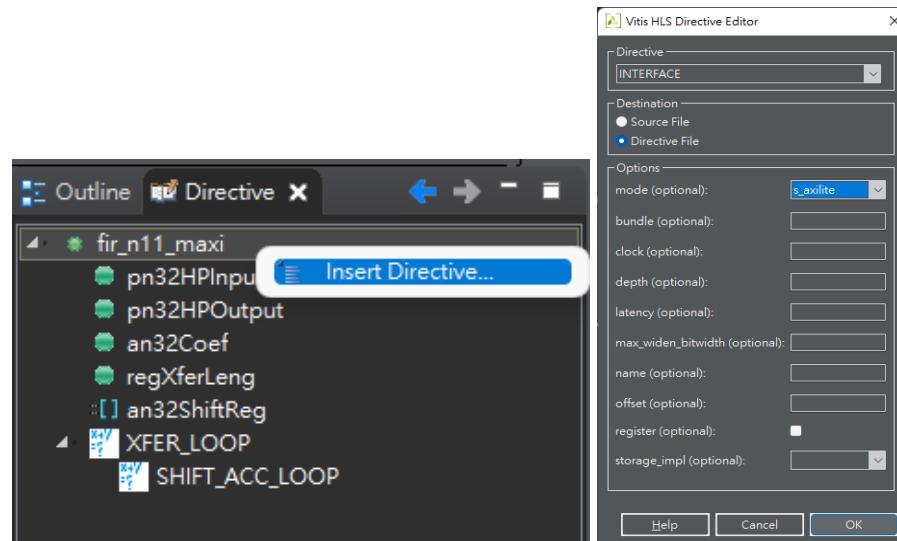


```

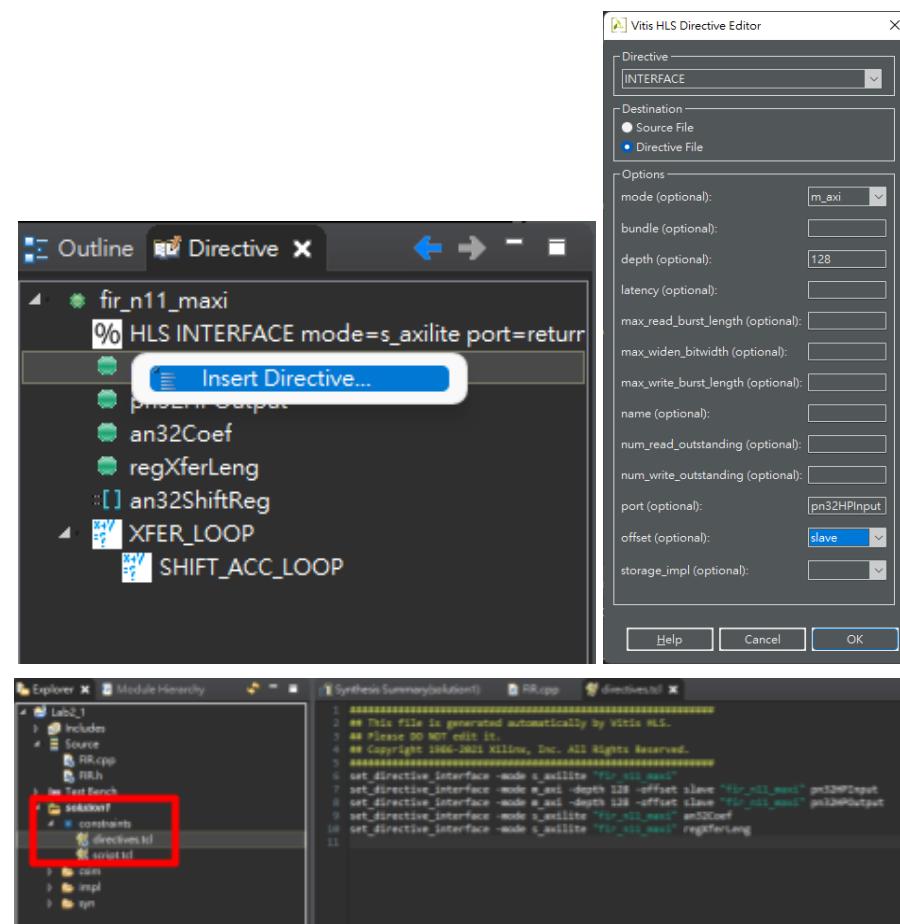
1 // SynthesisSummary(solution)
2 // FIR.cpp
3 // directives.hd
4
5 #include "fir.h"
6
7 void fir_n11_maxi(volatile int32_t* pn32HPIput, volatile int32_t* pn32HPOutput, volatile int32_t* an32Coef[1600_128_41MP], reg32_t* regXferLeng)
8 {
9     static int32_t an32ShiftReg[1];
10    int32_t n32Acc;
11    int32_t n32Data;
12    int32_t n32Loop;
13    int32_t n32AccTemp;
14    int32_t n32AccTemp0;
15    int32_t n32AccTemp1;
16
17    n32Acc = regXferLeng + (an32Coef[1600_128_41MP] - 1) / an32Coef[1600_128_41MP];
18    if(n32Acc >= 0):
19        for (n32LoopCnt = 0; n32LoopCnt < n32Acc; n32LoopCnt++) {
20            n32Acc -= 1;
21            n32AccTemp = pn32HPOutput[n32LoopCnt];
22            XFER_LOOP:
23                for (n32Loop = N - 1; n32Loop >= 0; n32Loop--) {
24                    if (n32Loop == 0) {
25                        an32ShiftReg[0] = n32AccTemp;
26                        n32Data = an32ShiftReg[0];
27                    }
28                    an32ShiftReg[n32Loop] = an32ShiftReg[n32Loop - 1];
29                    n32Data = an32ShiftReg[n32Loop];
30                }
31            n32Acc += n32Data * an32Coef[n32Loop];
32        }
33        pn32HPOutput[n32LoopCnt] = n32Acc;
34    }
35
36    return;
37 }

```

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。

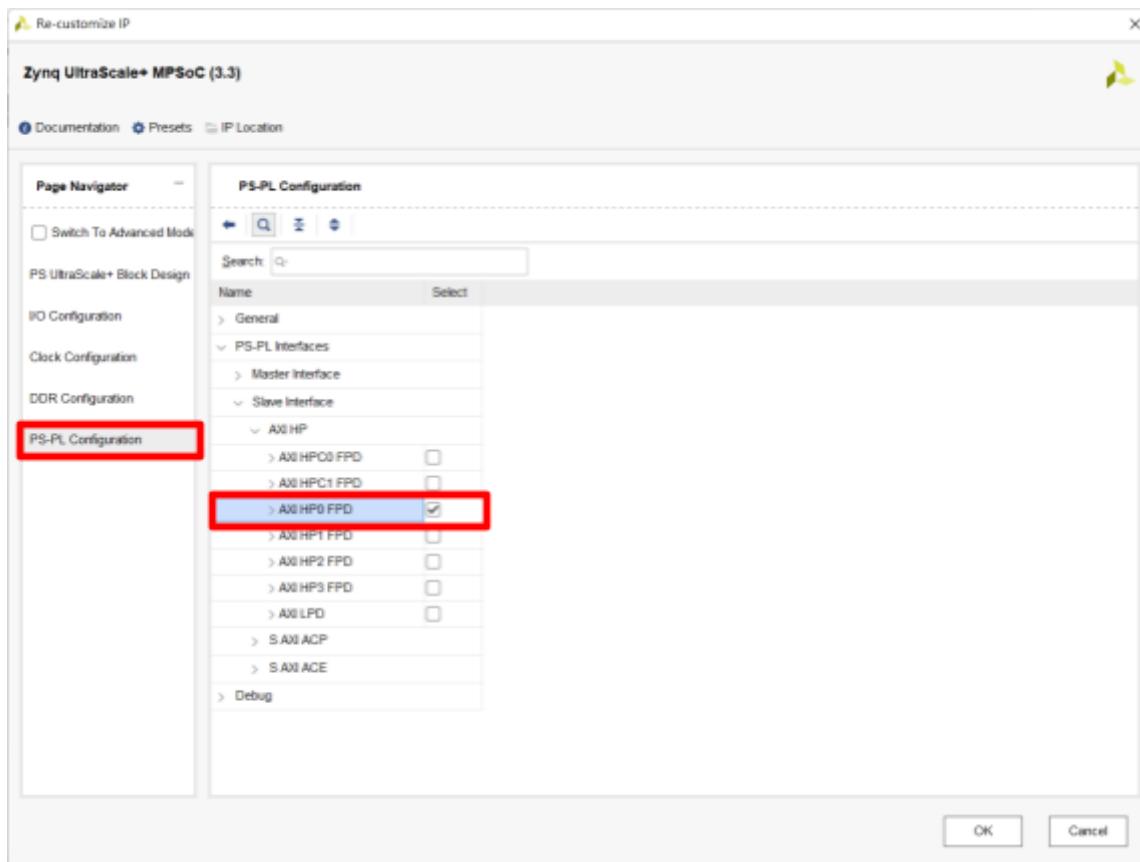
(settings → ip → repository → add ip (vitis project folder) → ok)

1.2.4. Block Design

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

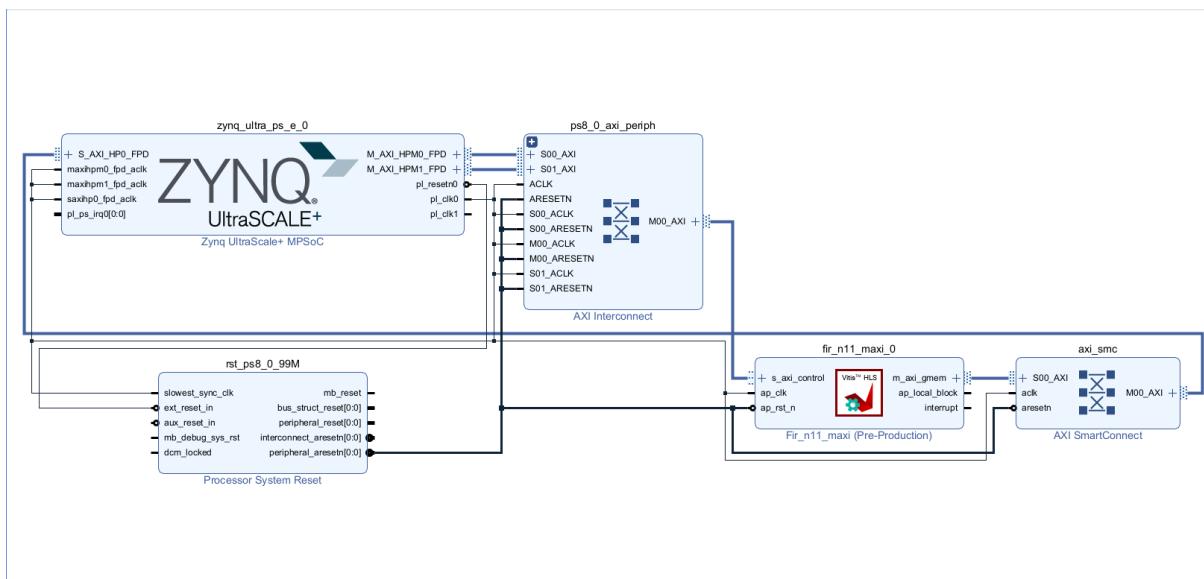
以下僅針對AXI-Master在processing system block的設定補充。(Run block automation會重置ZYNQ的configuration, 一定要先run block automation然後再完成ZYNQ的configure)

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

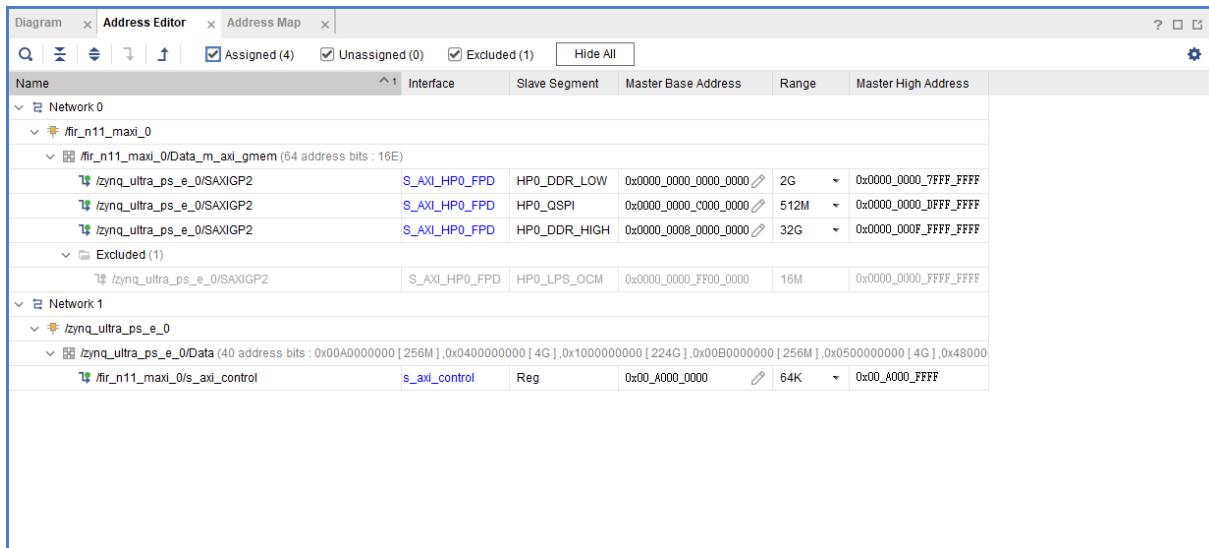


接著按Run Connection Automation。

最終完成的Block Diagram如下圖：



可開啟Address editor檢查Address是否相同。



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。

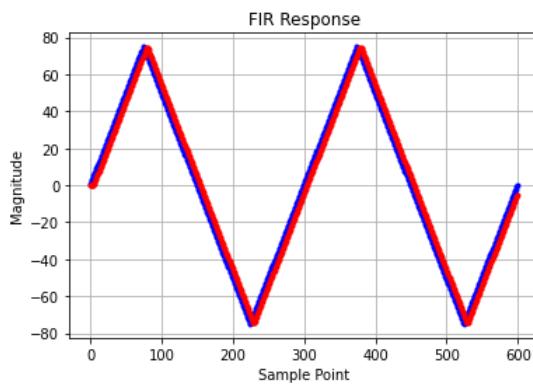
(Create HDL Wrapper → Generate bitstream)

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, 記得更改code裡使用到的檔案路徑。

```
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: 0.00026535987854003906 s
```



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

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 : **xck26-sfvc784-2LV-c** → Add Source*2, Add testbench*1, Add 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。

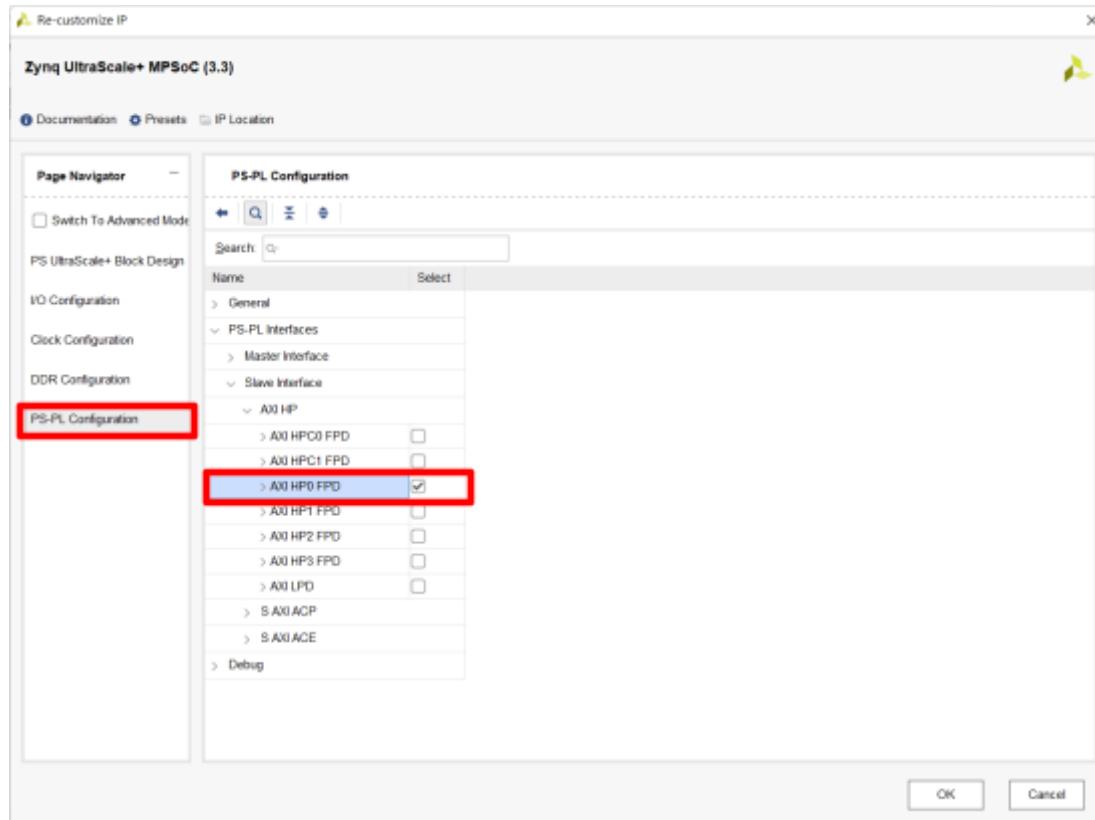
2.2.2. Import IP

同Lab.#1, 請自行參照Workbook 1說明步驟操作。

2.2.3. Block Design

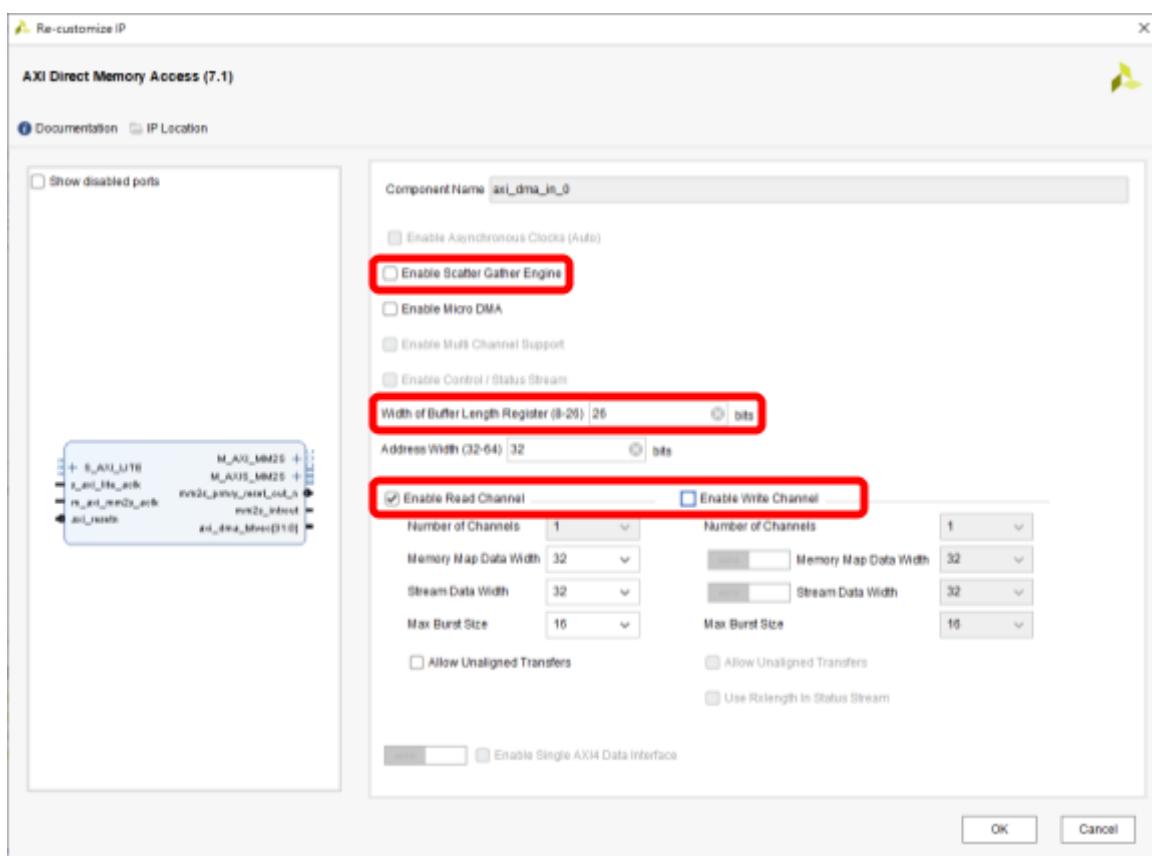
同Lab.#1, 請自行參照Workbook 1說明步驟操作。以下僅針對AXI-Stream在processing system block的設定補充。

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

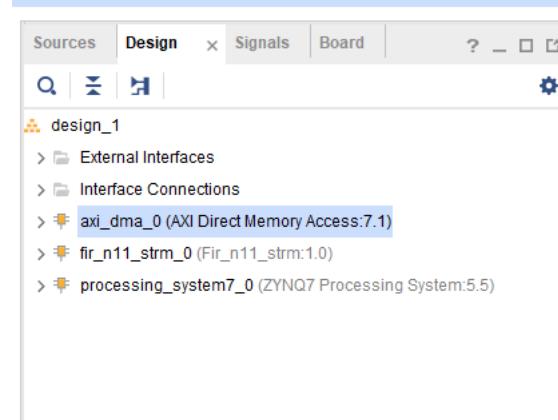


另外，AXI-Master to stream是用Xilinx DMA IP來實現，針對DMA IP需要調整及設定。在Block Design時，除匯入HLS fir_n11_strm IP並將IP component加入至Diagram中之外，還需要加入Xilinx DMA IP component，加入後以滑鼠雙擊DMA block。下圖圈選處：

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。



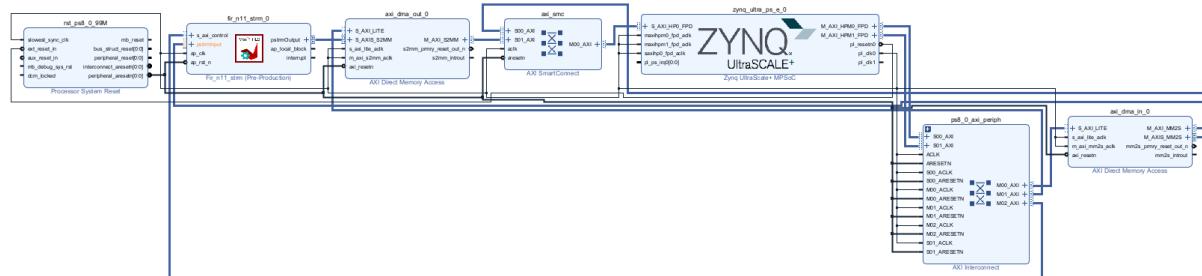
接著按run connection automation到沒有出現為止。

注意這裡要手動連兩條wire:

FIR_N11_STM的pstrminput接到axi_dma_in_0的M_AXIS_MM2S

FIR_N11_STM的pstrmoutput接到axi_dma_out_0的S_AXIS_S2MM

最終完成的Block Diagram如下圖:



Address Editor						
<input type="checkbox"/> Assigned (7)		<input checked="" type="checkbox"/> Unassigned (0)	<input checked="" type="checkbox"/> Excluded (4)	Hide All		
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)						
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_DDR_LOW	0x0000_0000	<input type="button"/>	2G	0x7FFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_QSPI	0xC000_0000	<input type="button"/>	512M	0xDFFF_FFFF
Excluded (2)						
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_LPS_OCM	0xFF00_0000	<input type="button"/>	16M	0xFFFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_DDR_HIGH				
/axi_dma_out_0						
/axi_dma_out_0/Data_S2MM (32 address bits : 4G)						
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_DDR_LOW	0x0000_0000	<input type="button"/>	2G	0x7FFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_QSPI	0xC000_0000	<input type="button"/>	512M	0xDFFF_FFFF
Excluded (2)						
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_LPS_OCM	0xFF00_0000	<input type="button"/>	16M	0xFFFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP2	S_AXI_HP0_FPD	HP0_DDR_HIGH				
Network 1						
/zynq_ultra_ps_e_0						
/zynq_ultra_ps_e_0/Data (40 address bits : 0x00A000000 [256M],0x0400000000 [4G],0x1000000000 [224G],0x00B0000000 [256M],0x0500000000 [4G],0x480						
/axi_dma_in_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A000_0000	<input type="button"/>	64K	0x00_A000_FFFF
/axi_dma_out_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A001_0000	<input type="button"/>	64K	0x00_A001_FFFF
/fir_n11 strm_0/s_axi_control	s_axi_control	Reg	0x00_A002_0000	<input type="button"/>	64K	0x00_A002_FFFF

2.2.4. Synthesis/Placement/Route/Generate

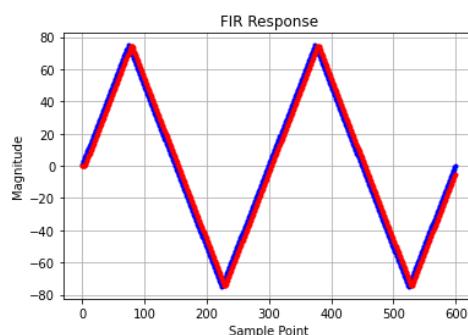
Bit-stream

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。

```
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: 0.0008192062377929688 s
```



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