

Jini Intelligent Computing Workbook of Lab. #1

Preamble

在 Lab. #1 的檔案中主要有下列專案目錄：

- hls_Multiplication
Vitis HLS 乘法器原始碼檔案
- vvd_Multip2Num
範例乘法器 Vivado Design Suite 參考檔案
 - design_1.tcl
範例乘法器 Block Design 完成 Generate Bitstream 後匯出之 TCL Script 檔
 - MakeBit.bat
範例乘法器完成 Generate Bitstream 後，將.bit/.hwh 拷貝至專案根目錄之批次檔
- ipy_Multip2Num
範例乘法器系統程式 Python 原始碼檔及 Jupyter Notebook 原始碼編輯檔

1. Implement Flow

1.1. Vitis HLS/IP Design

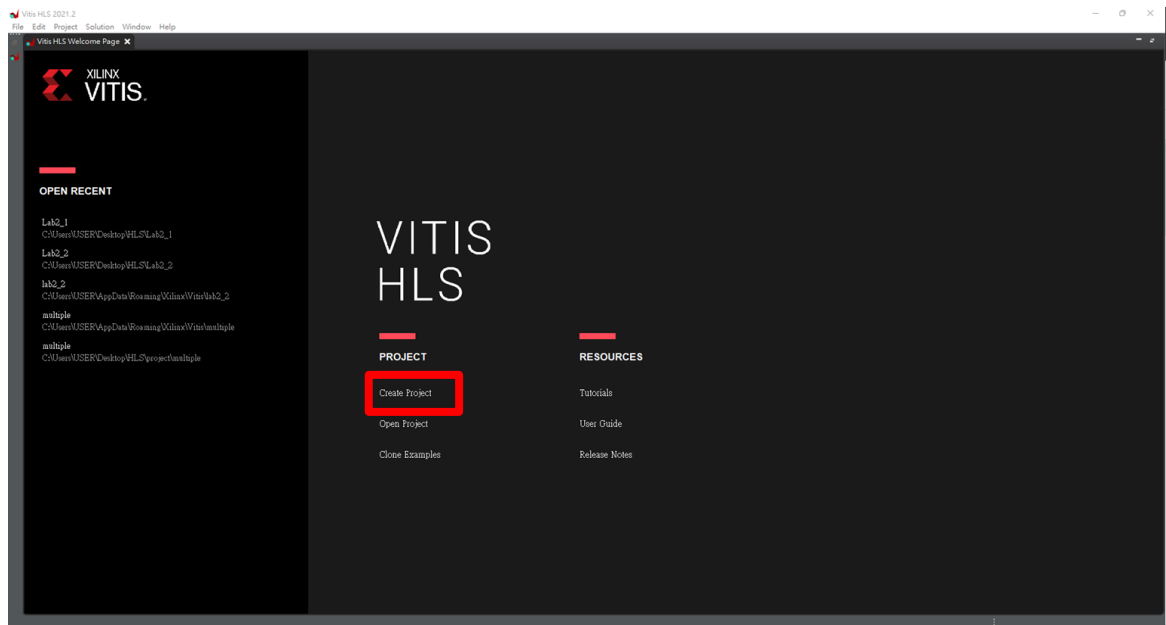
【施作環境為 Windows】

啟動 Vitis HLS 開發套件。



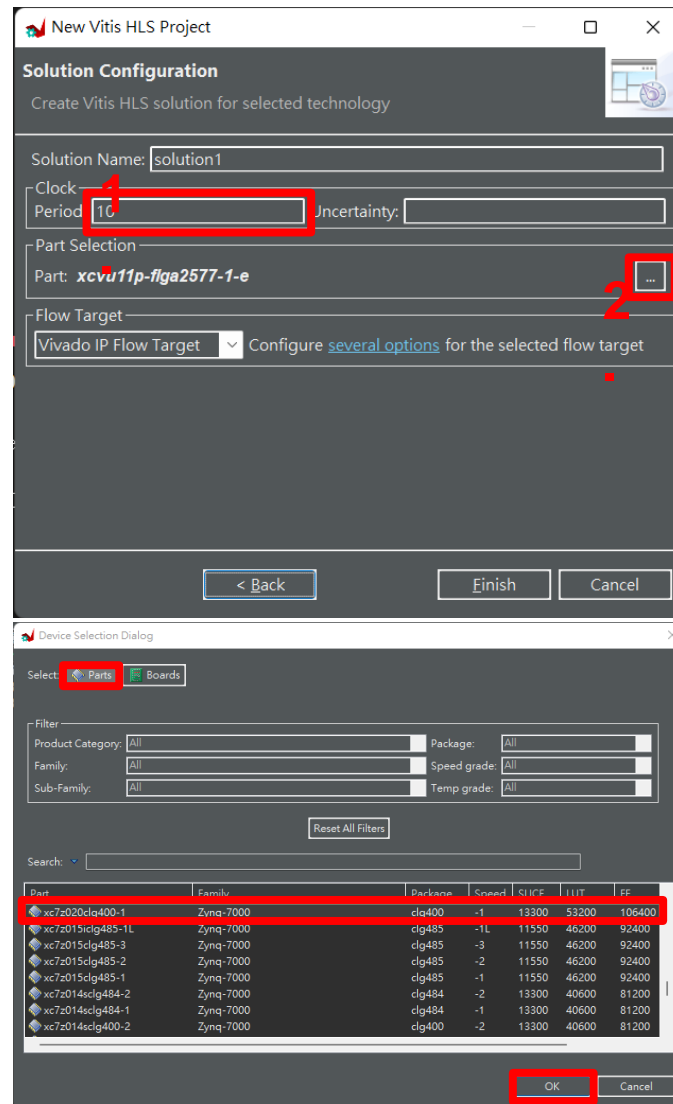
1.1.1. Create Source Project

開啟新的專案，設定好專案存放路徑：



1.若 PYNQ-Z2 的 FPGA 可運行於時脈 200MHz 以上，可以選擇週期為 10 ns。

2.因 vitis 2021.2 版本讀不到 pynq z2 board file，所以選用與 pynq z2 相同 Part 的 **xc7z020clg400-1**

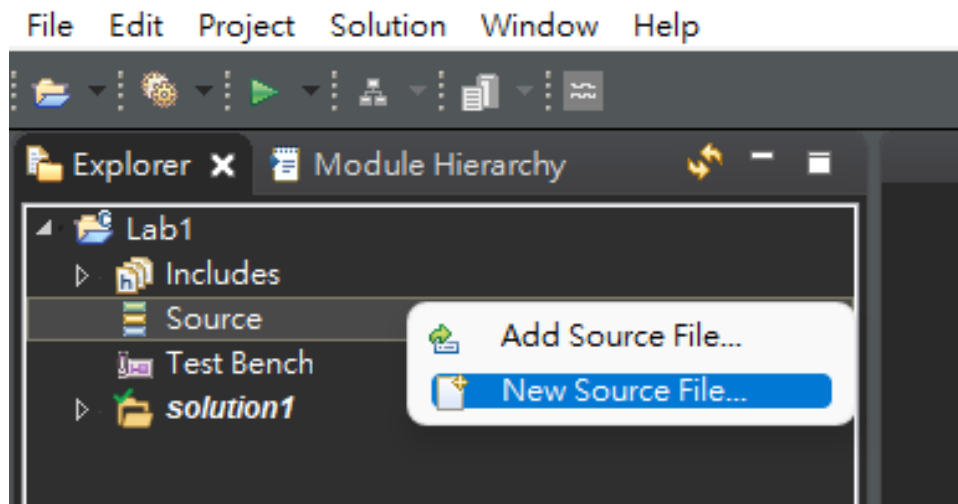


接著將 hls_Multiplication 中三個檔案放到 vitis hls 專案對應的位置

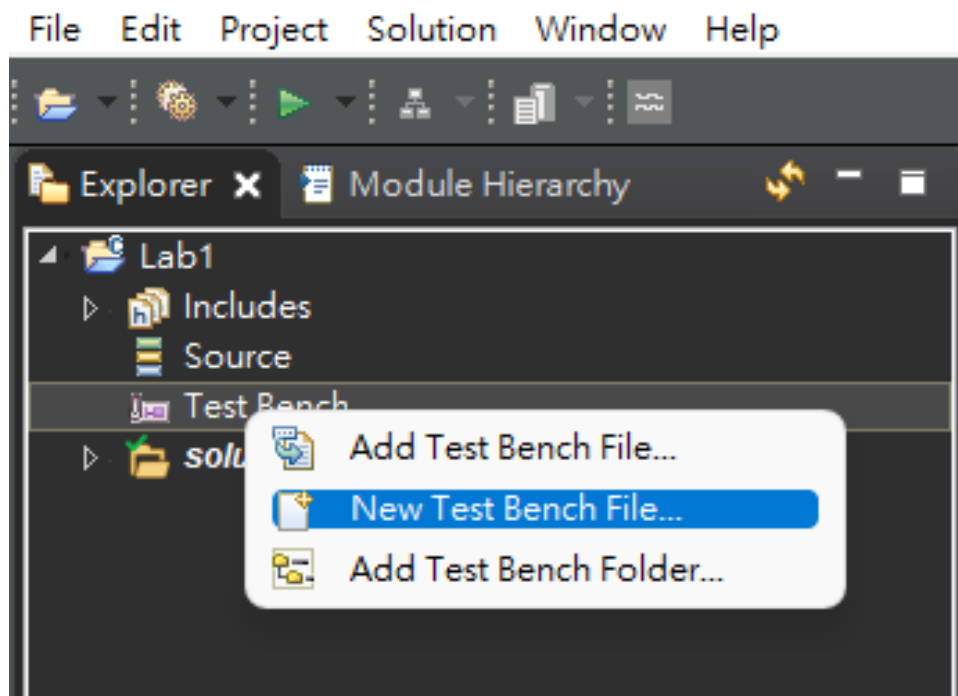
名稱	修改日期	類型	大小
.apc	2022/1/27 下午 05:10	檔案資料夾	
.settings	2022/1/27 下午 05:10	檔案資料夾	
solution1	2022/1/27 下午 05:11	檔案資料夾	
.cproject	2022/1/27 下午 05:10	CPROJECT 檔案	29 KB
.project	2022/1/27 下午 05:10	PROJECT 檔案	2 KB
.vitis_hls_log_all.xml	2022/1/27 下午 05:11	XML Document	1 KB
hls.app	2022/1/27 下午 05:10	APP 檔案	1 KB
Multiplication.cpp	2021/11/16 下午 05:47	CPP 檔案	1 KB
Multiplication.h	2021/11/16 下午 05:40	C Header 來源檔案	1 KB
MultipTester.cpp	2020/7/21 上午 04:24	CPP 檔案	1 KB

進入 Vitis HLS 專案 IDE 畫面後，加入 Source/Test Bench 的原始碼檔。

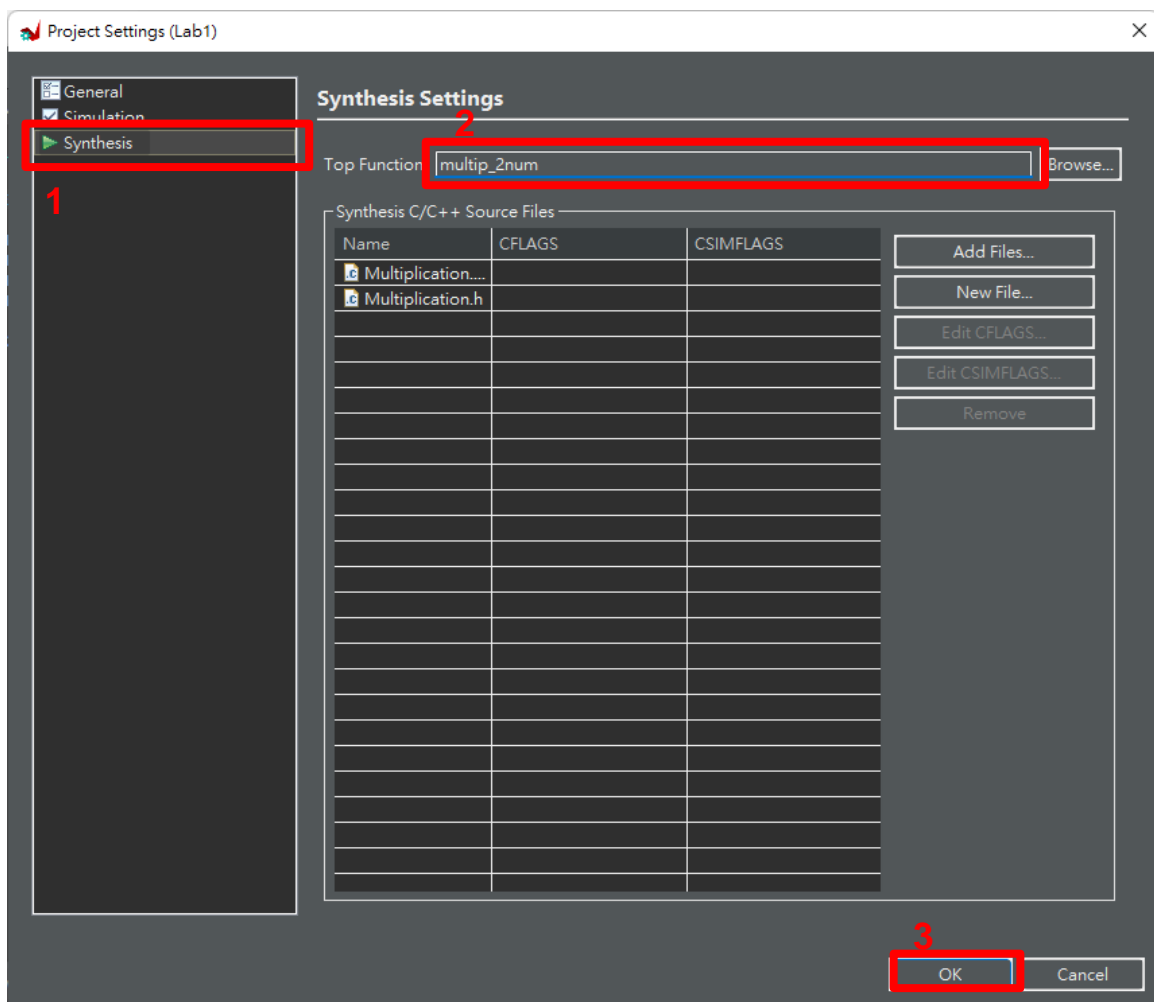
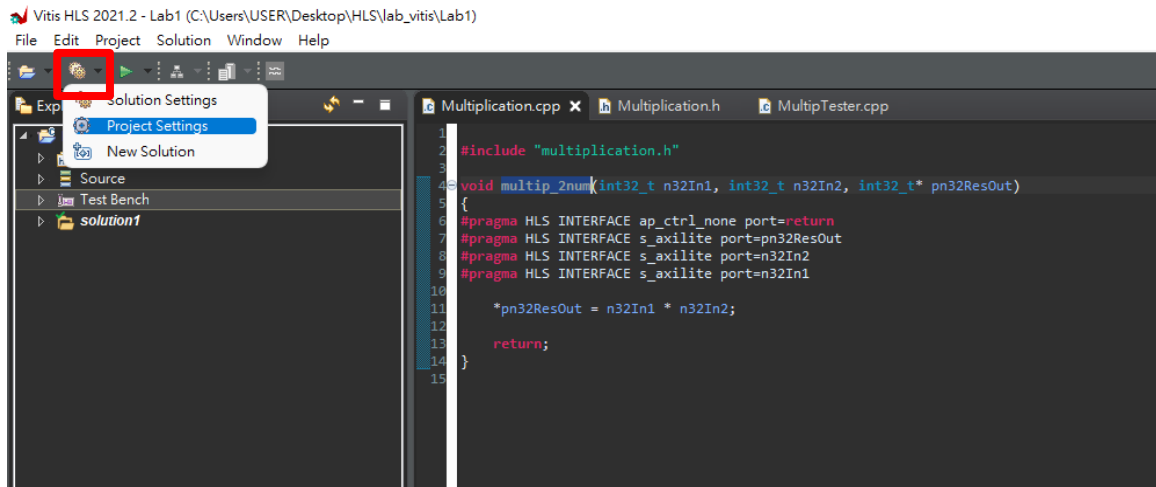
在專案 Source 按滑鼠右鍵加入新檔，在 Source 選擇 Multiplication.cpp 及 Multiplication.h 檔。（#Include 部分注意大小寫問題。）



在專案 Test Bench 按滑鼠右鍵加入新檔，在 Test Bench 選擇 MultipTester.cpp 檔。



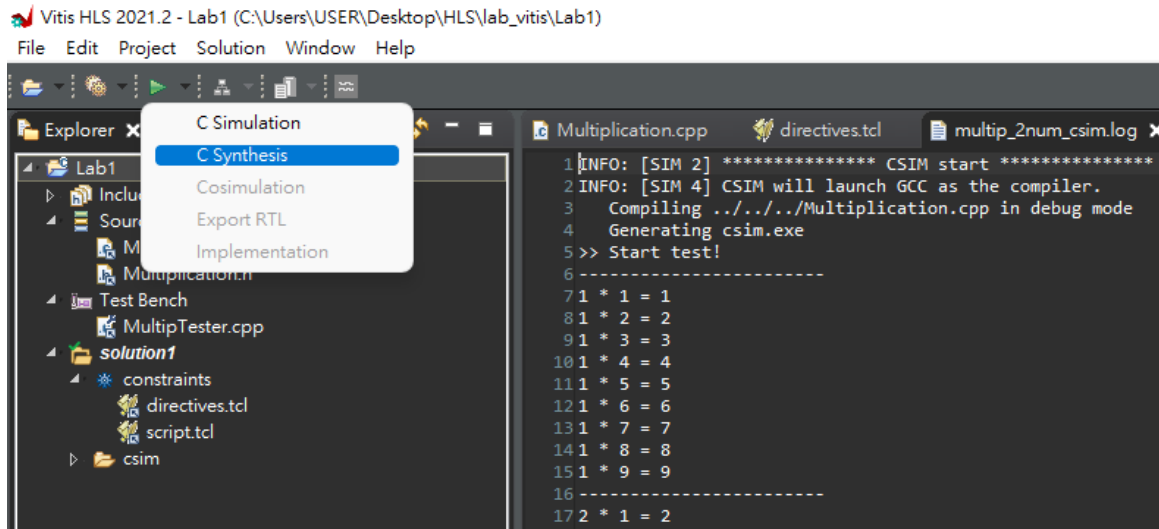
專案必須指明 IP 匯出的 Top Function，在工具列設定圖加入 Top Function 名稱。(Top Function 名稱即為 Multiplication.cpp Function 名稱)



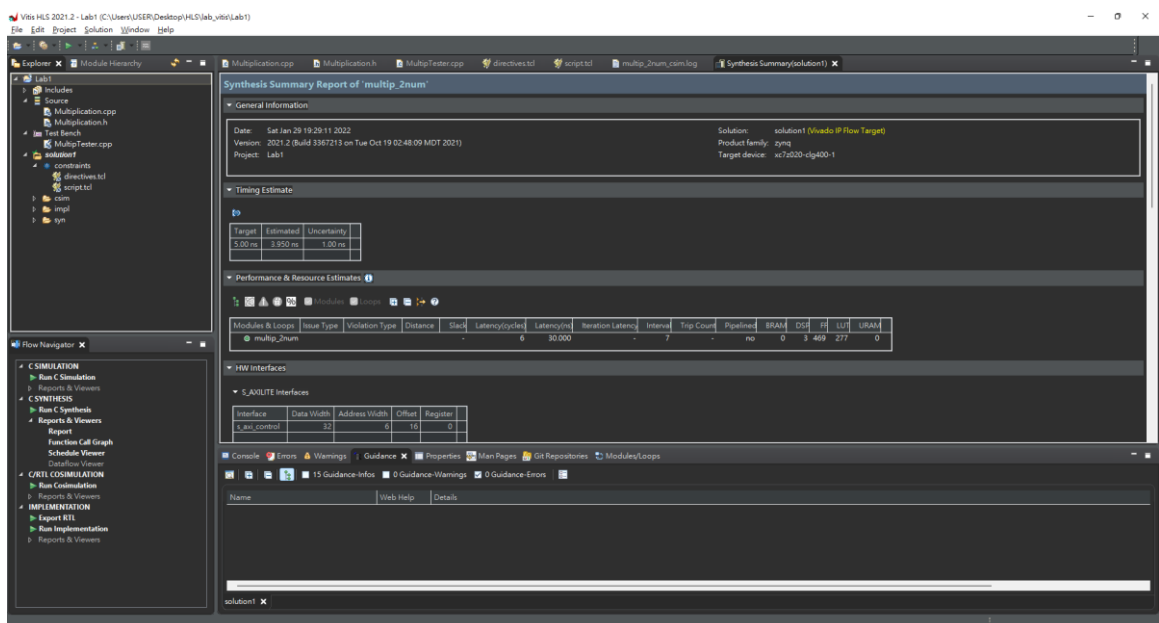


1.1.4. Synthesis

在 Vitis HLS 的功能列執行 C Synthesis：



彈出視窗按 ok 進行 C synthesis



完成 C Synthesis 會在主視窗回報 synthesis report。

(補充：若有 slack / time violation，可調整 period。)

1.1.5. Cosimulation

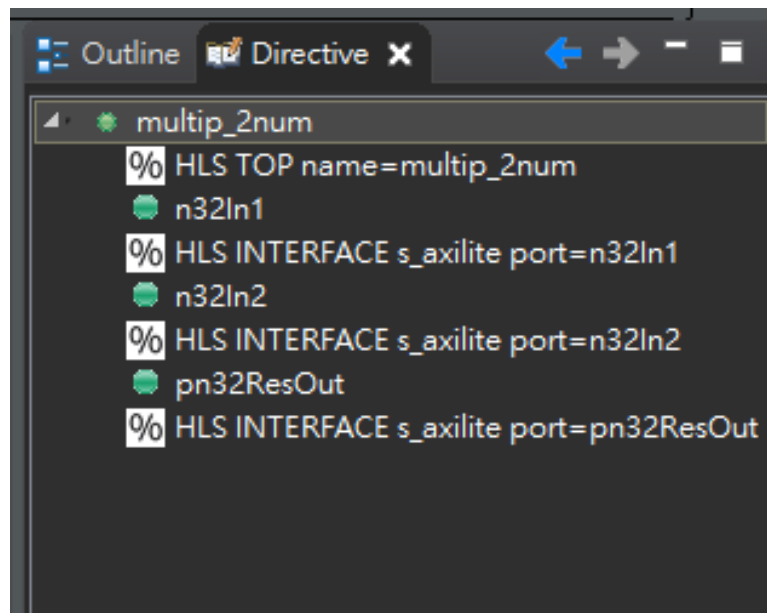
當進行 Cosimulation 時 directive 的組態需改成如下圖 (將 ap_ctrl_none 那行拿掉)，不然會出現錯誤無法完成 Cosimulation。在執行 Cosimulation 必須重新跑一次

C Synthesis。

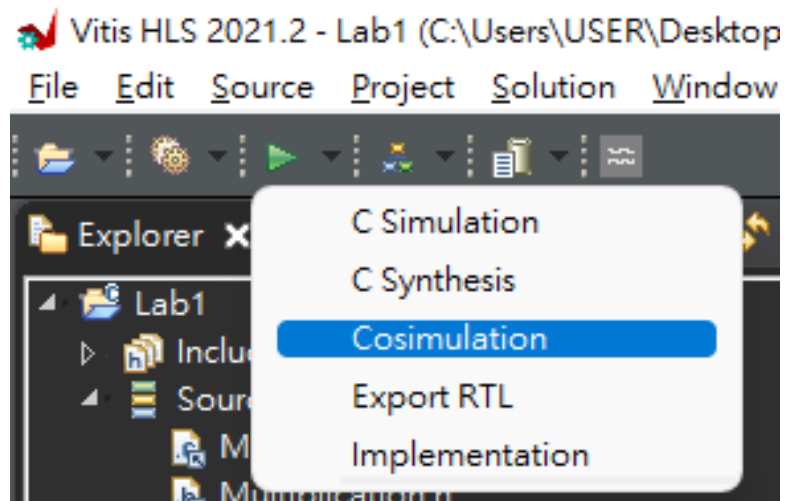
參考網址：

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_1/ug871-vivado-high-level-synthesis-tutorial.pdf

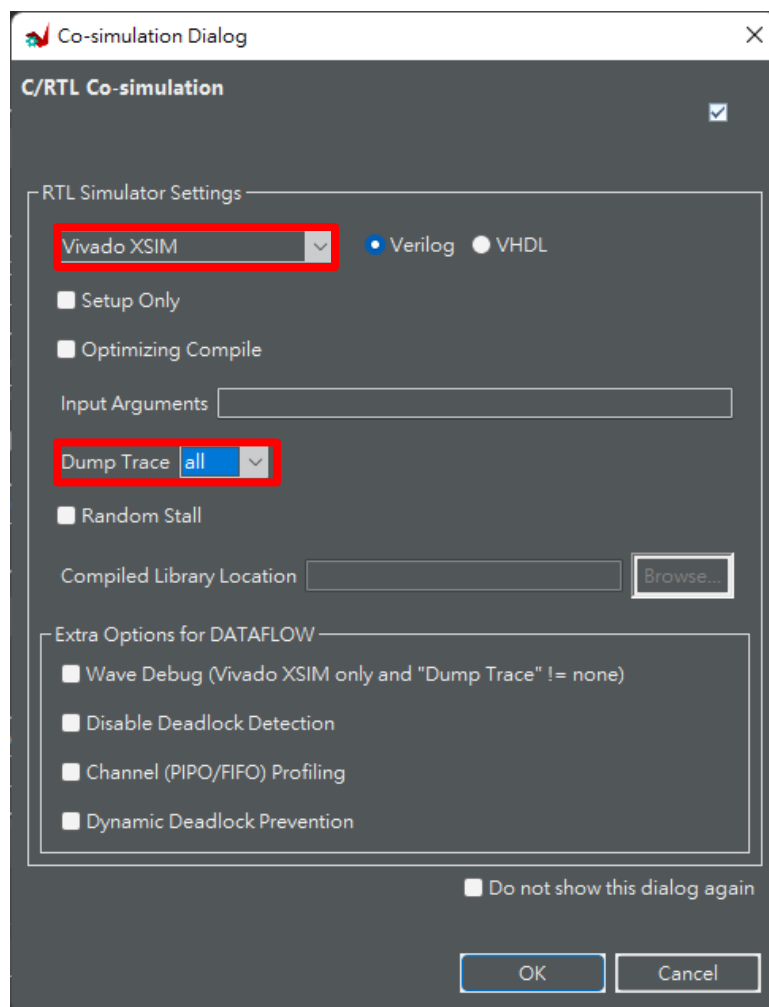
<https://forums.xilinx.com/t5/High-Level-Synthesis-HLS/Using-ap-memory-with-ap-control-none/td-p/681187>

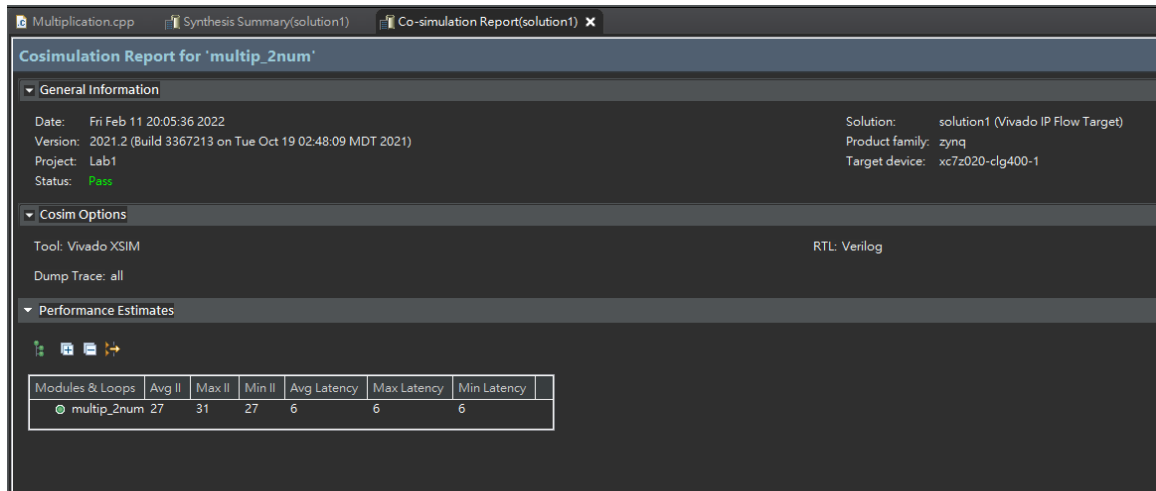


按下 Cosimulation 來驗證設計。

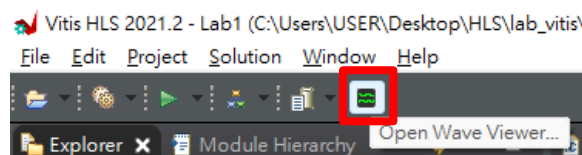


按下 Cosimulation 鍵後會彈出對話視窗，選擇 Vivado XSIM，將 dump trace 選擇 all





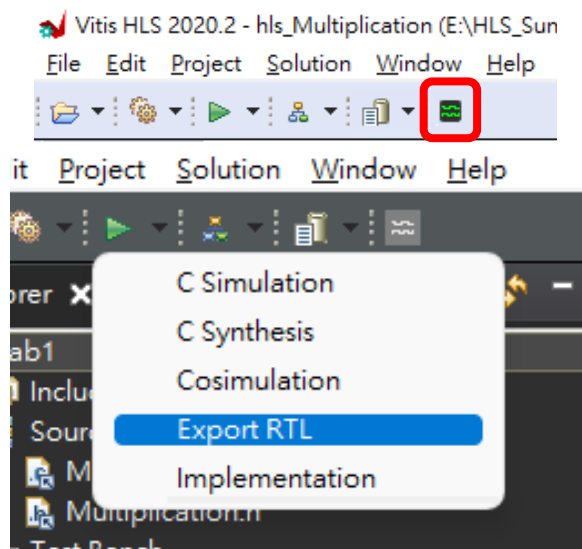
在輸出視窗檢視。要檢視波形可執行 Open Wave Veiwer。

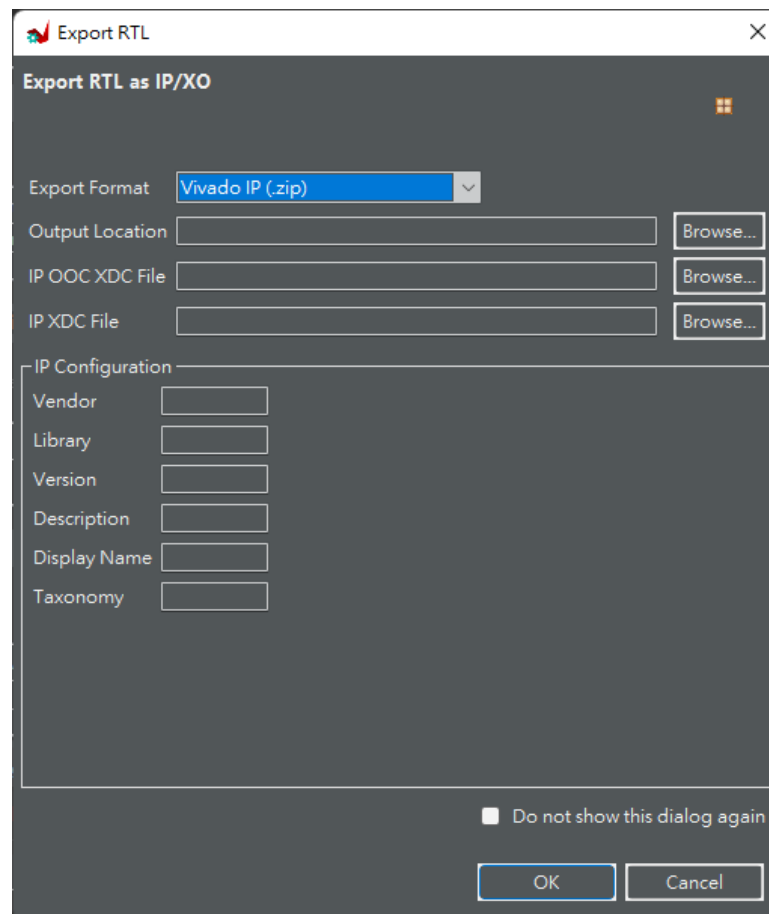


1.1.6. Export IP

完成 IP 設計後，從 Vitis HLS 匯出 IP，之後 Vivado 會需要匯入由 Vitis HLS 匯出的 IP。

注意：若前面有跑 cosimulation 的話記得先還原成原本的 directive，重新 Synthesis 再匯出 RTL。

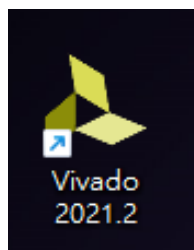




1.2. Vivado/Implement Flow

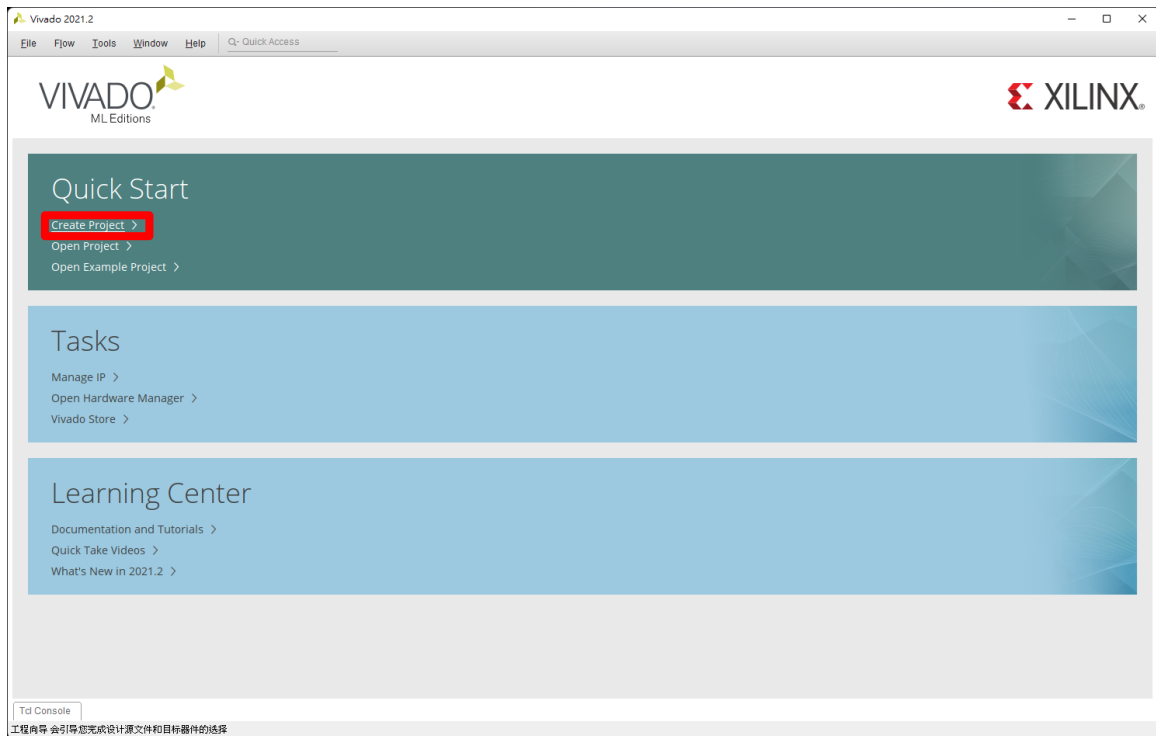
【施作環境為在使用者 PC/laptop/notebook (Windows Base) 。】

啟動 Vivado Design Suite 的 Vivado 開發套件。



1.2.1. Create Design Project

開啟新的專案，設定好專案存放路徑：



選擇好專案儲存路徑。RTL Project、sources 跟 constraints 留空之後再加。

Parts 選擇 **xc7z020clg400-1**。

New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts Boards

[Reset All Filters](#)

Category: All Package: All Temperature: All

Family: All Speed: All Static power: All

Search: xc7z020clg400-1 (1 match)

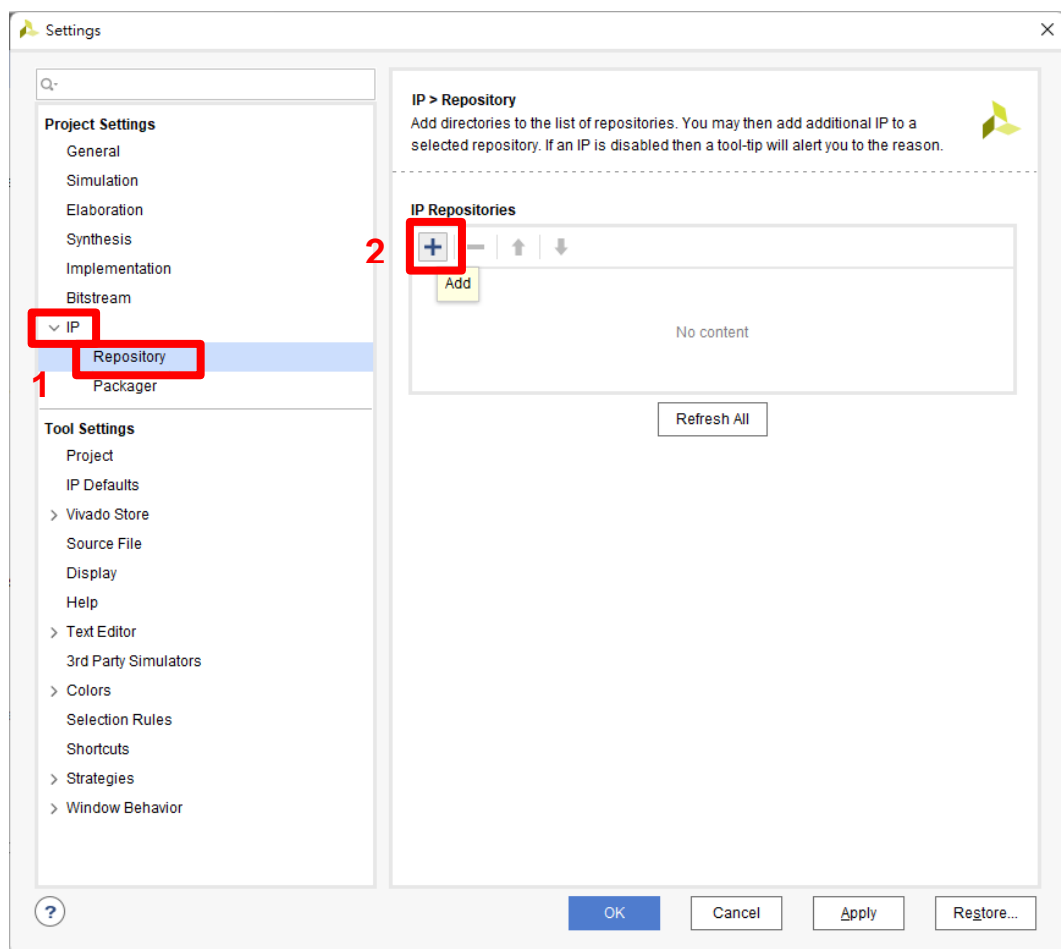
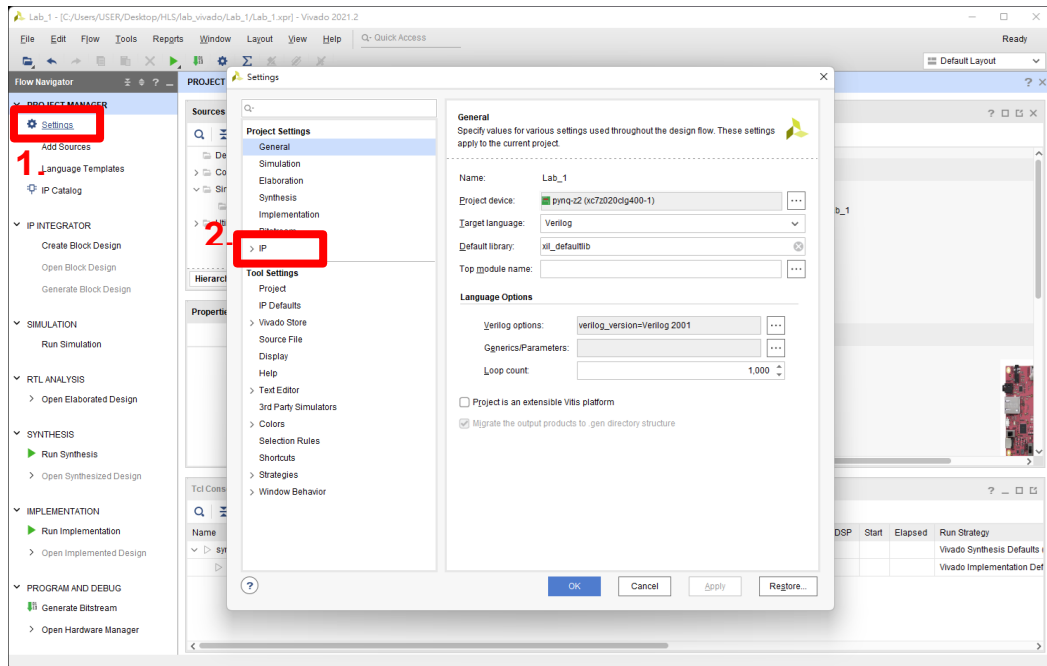
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	C
xc7z020clg400-1	400	125	53200	106400	140	0	220	0	0

< Back Next > Finish Cancel

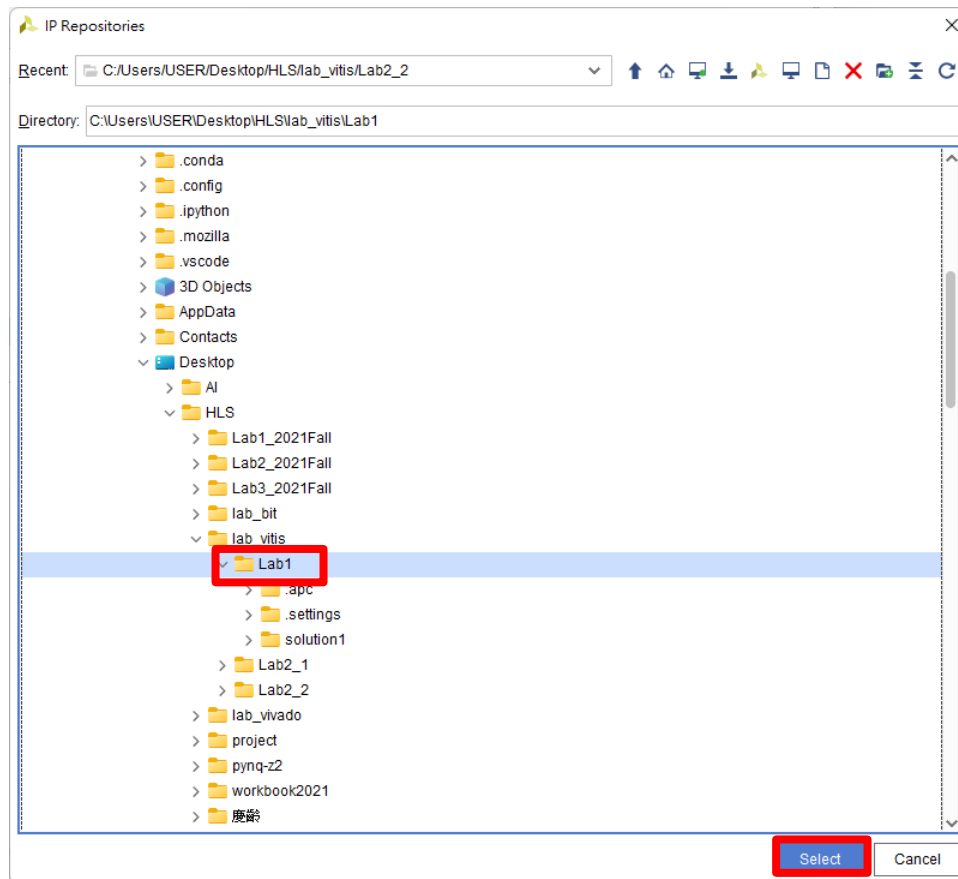
1.2.2. Import IP

進入 Vivado 專案 IDE 畫面後，第一步驟是匯入由 Vitis HLS 所產生的 IP，在專案

管理點擊 Settings 選項。

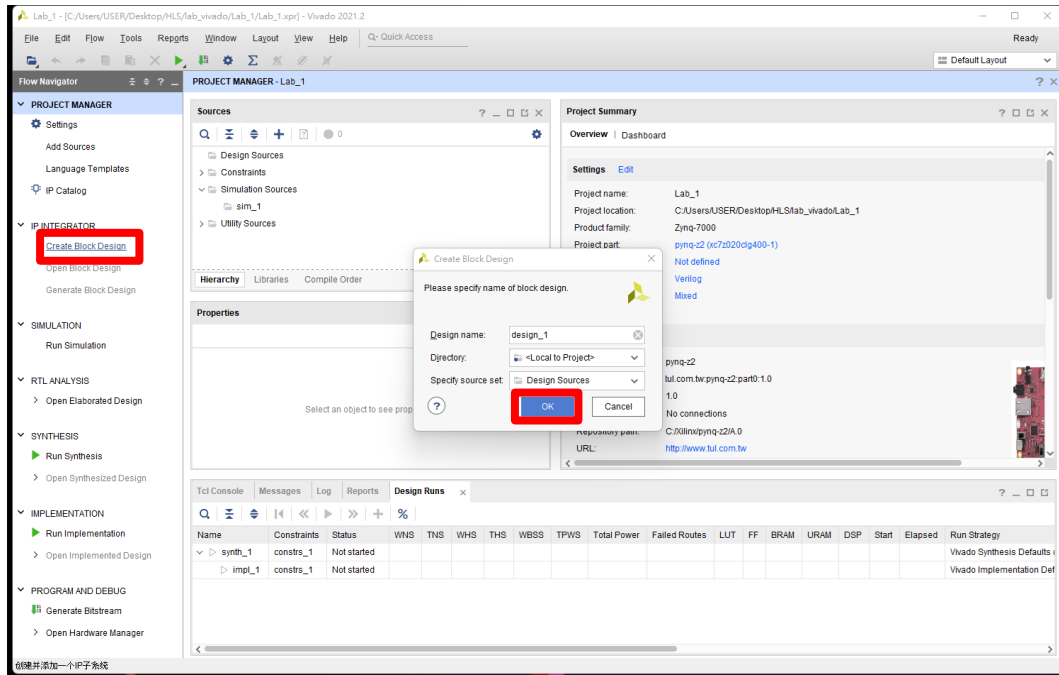


將 IP Repositories 指定到 Vitis HLS 專案目錄，下一步會匯入由 Vitis HLS 專案開發的 IP。

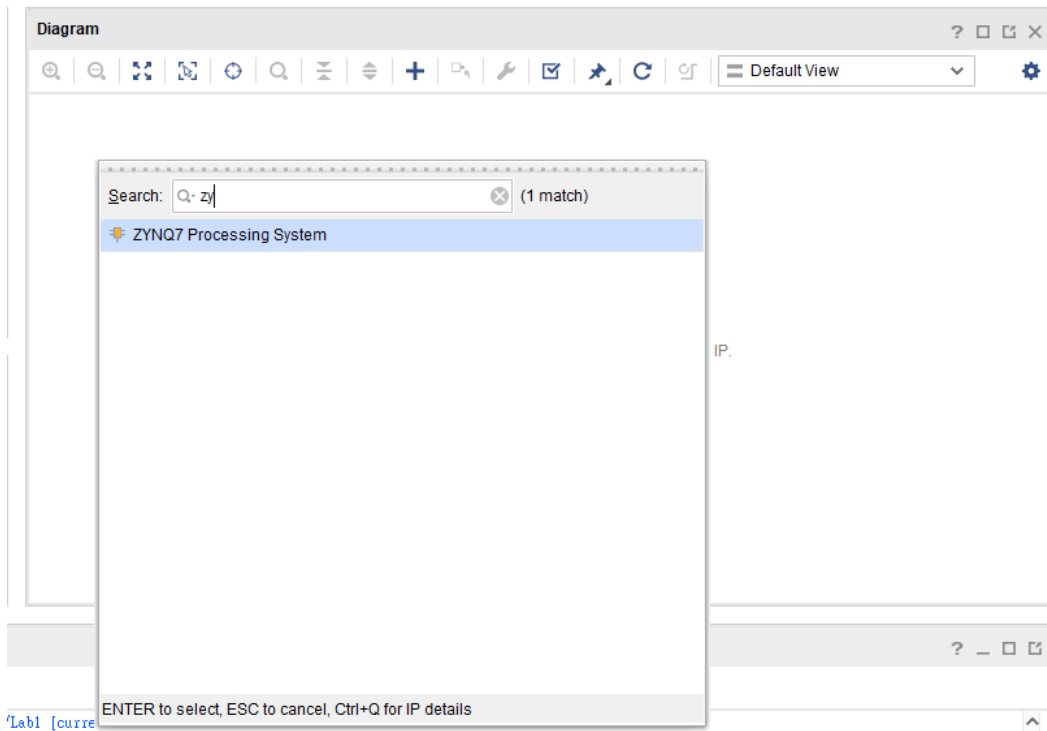


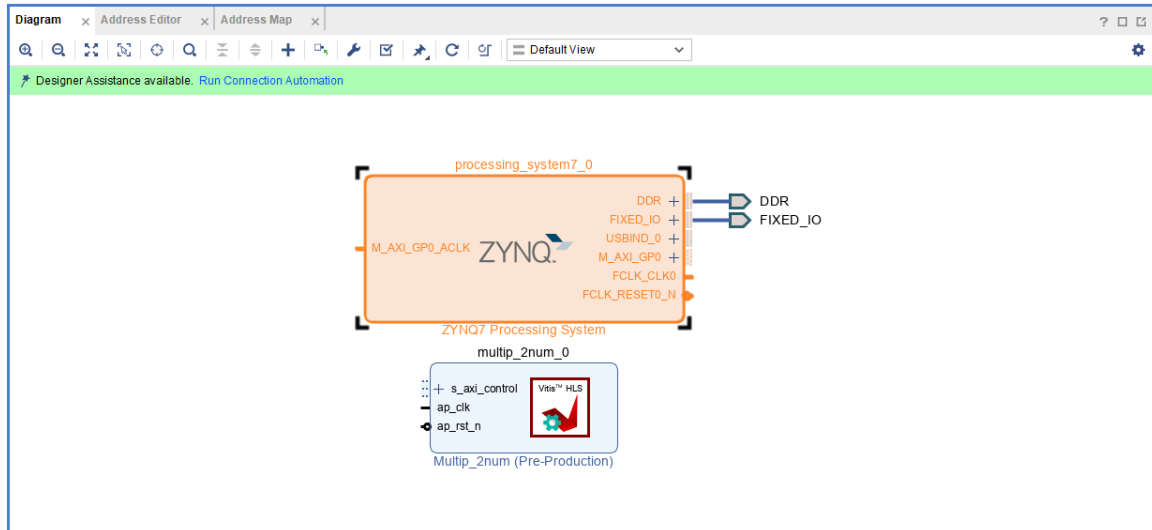
1.2.3. Block Design

回到 Vivado 專案 IDE 畫面，在專案管理點擊 Create Block Design 選項。



在 Diagram tab 視窗加入 components(ZYNQ7 Processing System 及 multip_2num_0)，並在 Run Block Automation 後用滑鼠左鍵雙擊 processing system block，將 PLL Fabric clock 設定為 **100MHz**。





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
- 1. I/O Configuration
- 1. Clock Configuration**
- DDR Configuration
- SMC Timing Calculation
- Interrupts

Clock Configuration

Summary Report

Basic Clocking Advanced Clocking

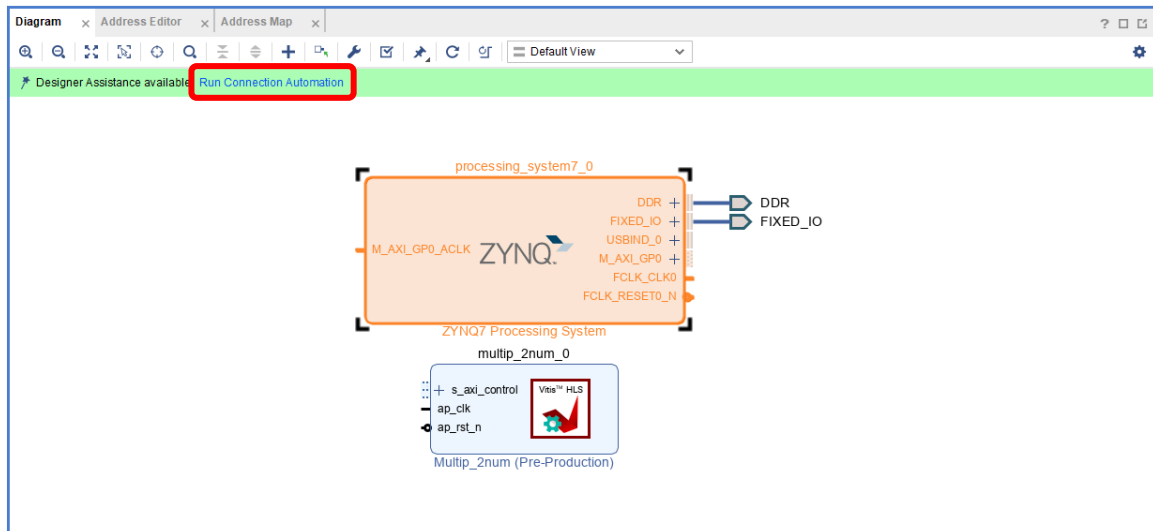
Input Frequency (MHz) 50 CPU Clock Ratio 6:2:1

Search: Q

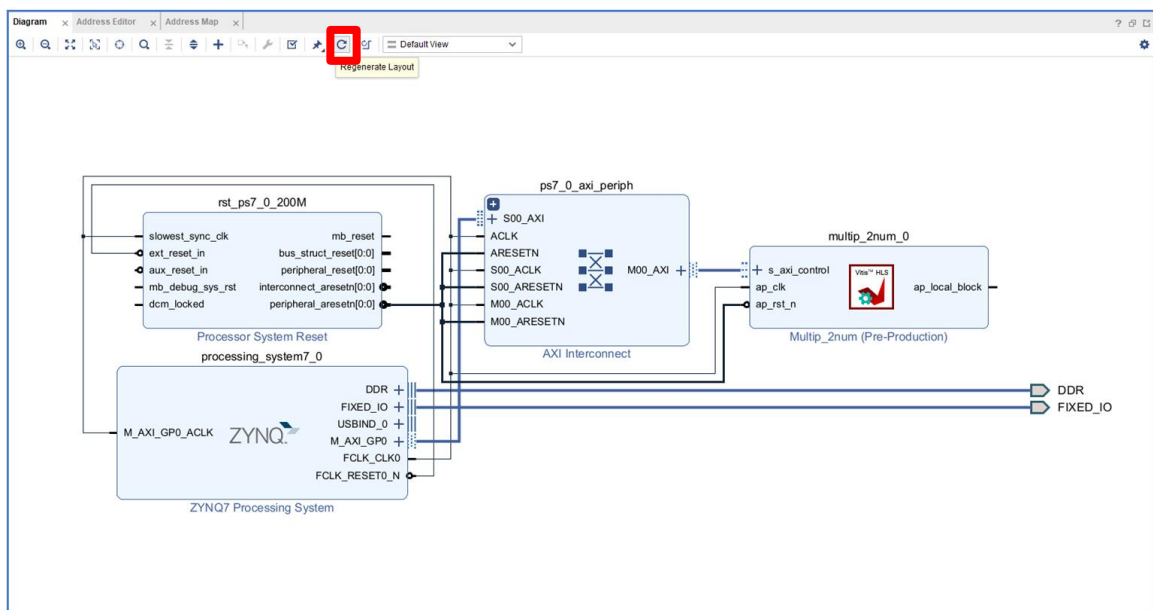
Component	Clock Source	Requested Frequ...	Actual Frequency(...)	Range(MHz)
> Processor/Memory Clocks				
2. IO Peripheral Clocks				
2. PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	3. 200	100.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK1	IO PLL	50	10.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000
> System Debug Clocks				
> Timers				

OK Cancel

因為 block design 只有 1 個 IP 的設計，連線完成可由系統自動完成，直接執行 Run Connection Automation 即可。

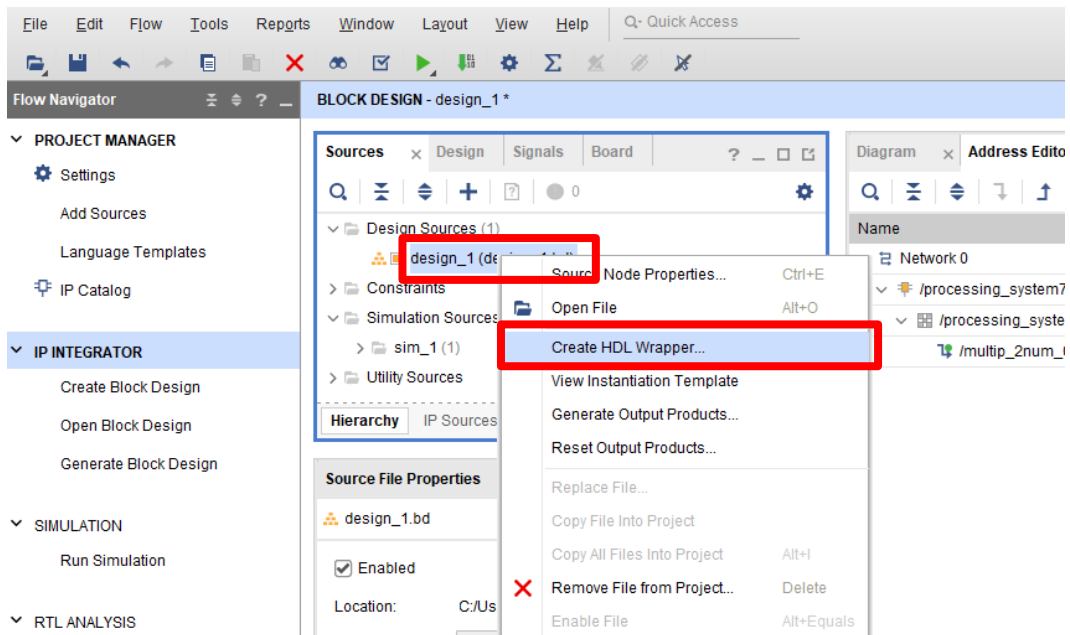


完成後整個完整的 diagram 圖，可點選上方 regenerate layout，檢查接線是否有誤，接著可切換到 Address Editor tab 檢視 memory map：



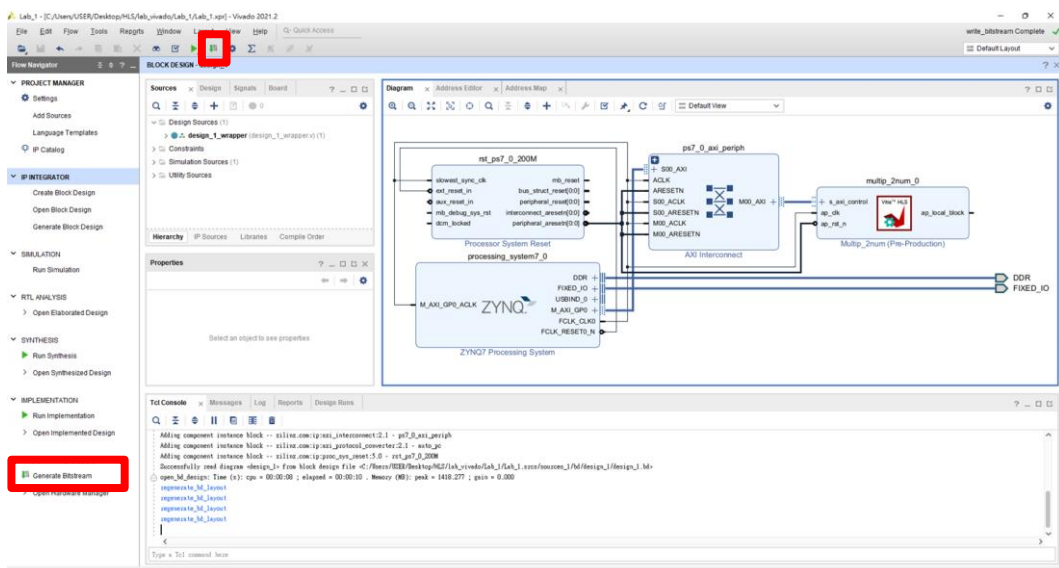
Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/multip_2num_0/s_axi_control	s_axi_control	Reg	0x4000_0000	64K	0x4000_FFFF

接下來進行 HDL Wrapper 動作：在 Design 子視窗 Source tab 頁面 Design Sources 的 design_1.bd 點擊滑鼠右鍵進行 HDL Wrapper：



1.2.4. Synthesis/Placement/Routing/Generate Bit-stream

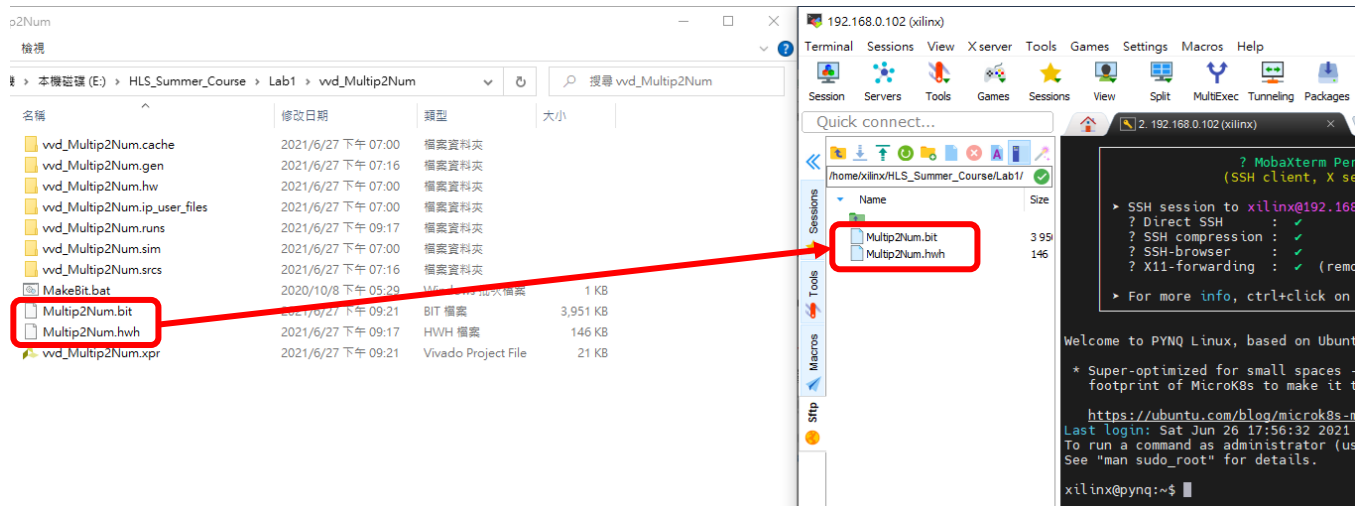
產生 FPGA 所需要的 bit-stream file，由於要產生 bit-stream file 需要經過 Synthesis/Placement/Routing 的步驟，但可以省去單步執行的流程以一鍵執行由 Vivado IDE 自動完成所有流程。在 Vivado 專案 IDE 畫面，在專案管理點擊 Generate Bitstream 選項或由工具列按下 Generate Bitstream 按鍵。



Launch Runs 的 Numbers of jobs 可以選多一點，會快一些。

1.2.5. Bit-stream Transfer from Development Kit to Device

本實作已建立一個批次檔 MakeBit.bat 將 FPGA 運行時所需要 .bit/.hwh 拷貝到專案根目錄，此時只要將 .bit/.hwh 藉由 MobaXterm 或 Samba 傳送到 PYNQ-Z2 即可。提醒！開發者建立的專案名稱可能與 MakeBit.bat 內的專案名稱不相符，請自行修改 .bat 內專案路徑名稱。



bit 檔位置 \Lab_1\Lab_1.runs\impl_1\design_1_wrapper.bit

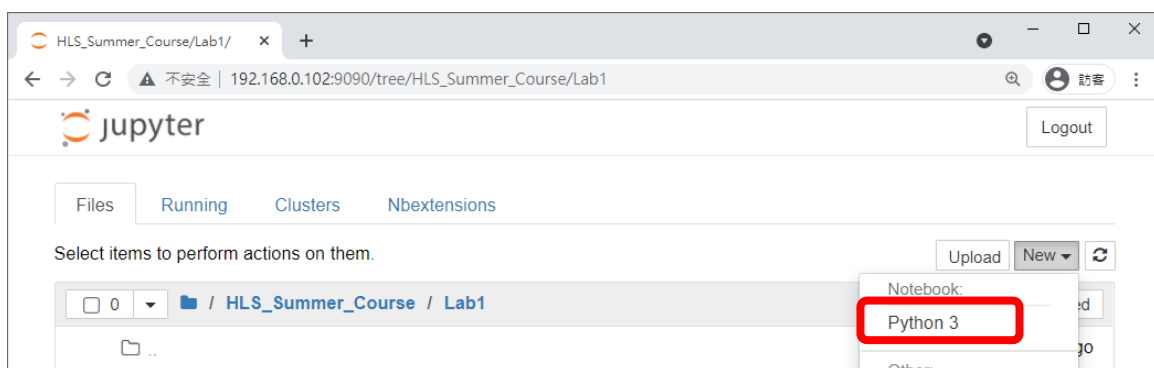
hwh 檔位置 \Lab_1\Lab_1.gen\sources_1\bd\design_1\hw_handoff\design_1.hwh

更改檔名，並記得存放在 pynq 版上的位置。

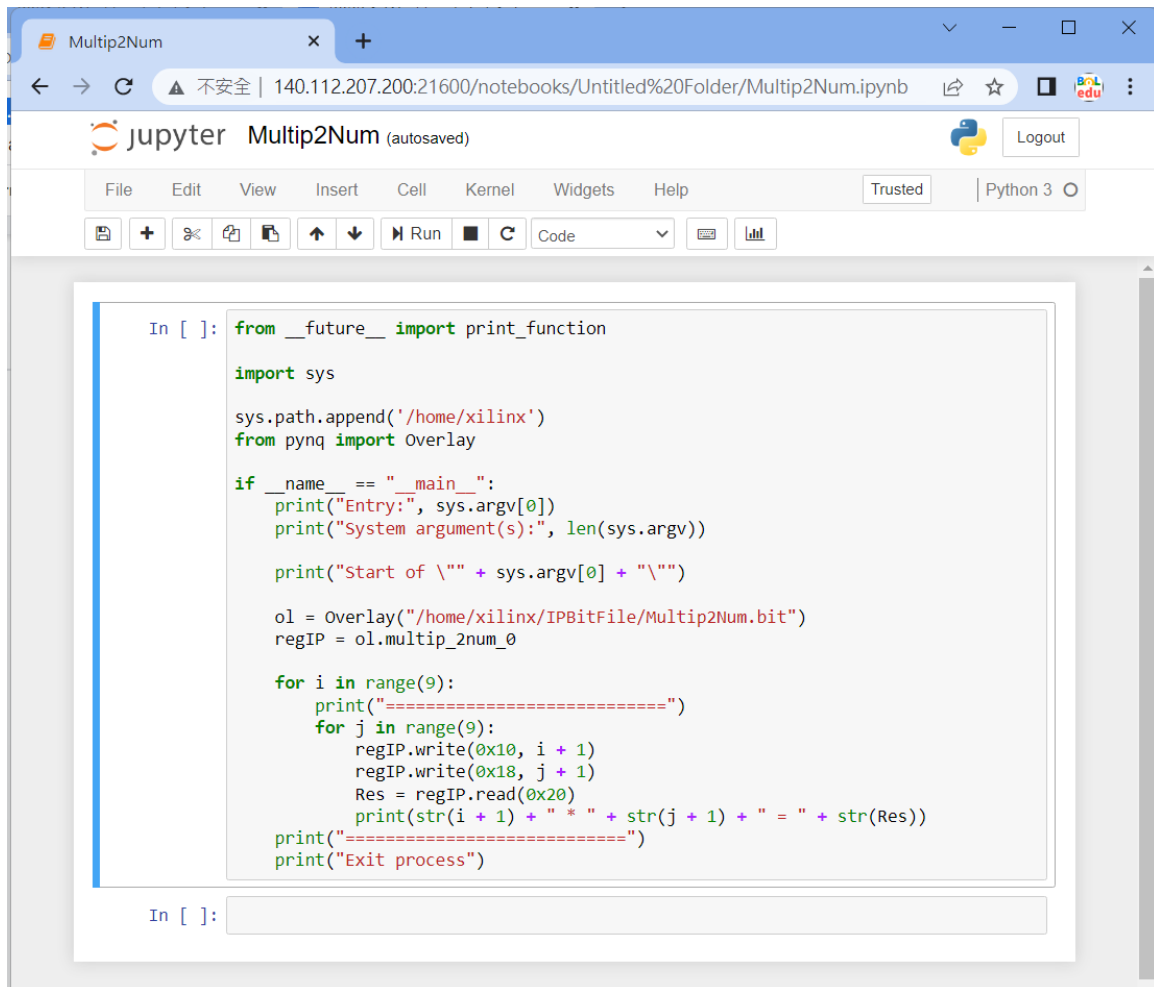
1.3. PYNQ/Host Program

1.3.1. Jupyter Notebook Browser Remote Editor

1. 啟動 Jupyter Notebook，並開啟一個新的 Python 3 檔案。



2. 將 Lab. #1 提供的 host program .py 內容拷貝到瀏覽器編輯視窗，並運行後檢視結果。(記得檢查 python code 裡用到的檔案位置是否正確)



A screenshot of a Jupyter Notebook interface in a web browser. The browser address bar shows the URL: 140.112.207.200:21600/notebooks/Untitled%20Folder/Multip2Num.ipynb. The notebook title is "Multip2Num (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and output. The code cell contains the following Python code:

```
In [ ]: from __future__ import print_function

import sys

sys.path.append('/home/xilinx')
from pynq import Overlay

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/IPBitFile/Multip2Num.bit")
    regIP = ol.multip_2num_0

    for i in range(9):
        print("=====")
        for j in range(9):
            regIP.write(0x10, i + 1)
            regIP.write(0x18, j + 1)
            Res = regIP.read(0x20)
            print(str(i + 1) + " * " + str(j + 1) + " = " + str(Res))
        print("=====")
    print("Exit process")
```

Below the code cell is an input prompt "In []:" followed by an empty text box.

Note :

Kernel 的 register address offset 可在以下檔案中找到：

hls_Multiplication\solution1\impl\misc\drivers\multip_2num_v1_0\src\xmultip_2num_hw.h

1.3.2. Understanding PYNQ

PYNQ 是建立在 Xilinx platform 上的 API 模組，可運行 Python 程式碼的環境，PYNQ 提供 Python 語言模組可進行對 FPGA 組態建立及流程控制。參考連結：

<http://www.pynq.io/board.html>

PYNQ open source: <https://github.com/xilinx/pynq>