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

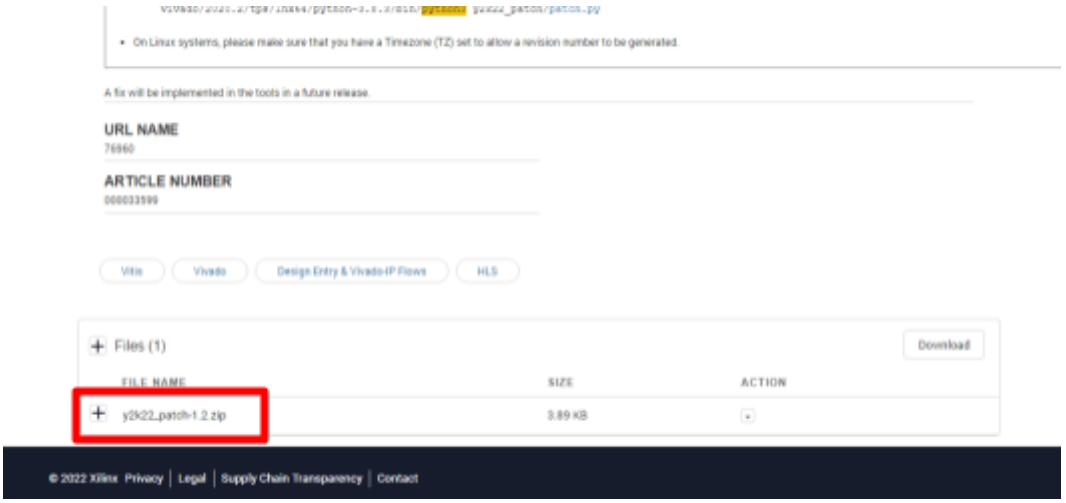
1. 前置作業

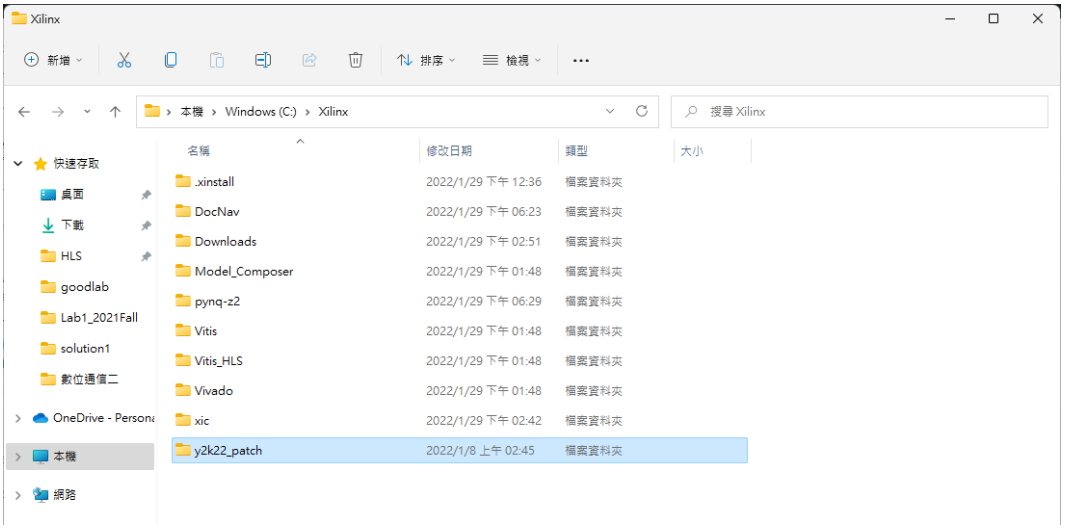
1.1. Time overflow patch update

由於vitis2021.2版本有time overflow問題，若直接使用會無法generate ip (vitis hls及vivado都無法輸出)，需下載patch才可使用。

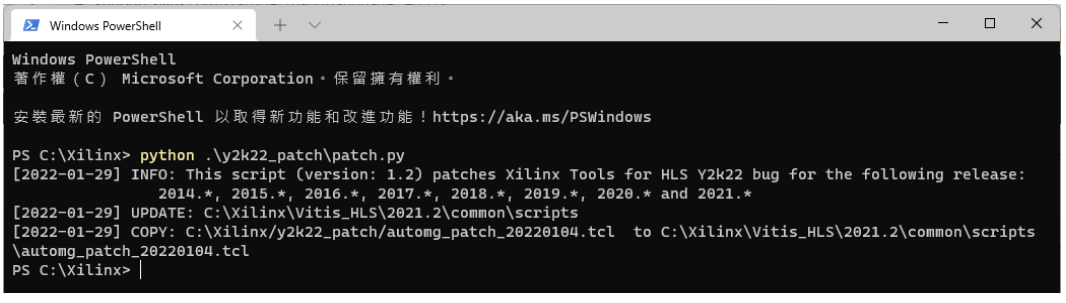
先到以下網站下載 **y2k22_patch-1.2.zip**，解壓縮並把資料丟到 C:/Xilinx

https://support.xilinx.com/s/article/76960?language=en_US





在此位置打開terminal，輸入 **python .\y2k22_patch\patch.py**。這樣即完成更新。

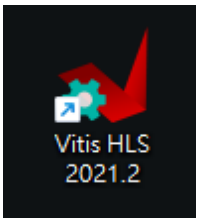


2. Implement Flow

2.1. Vitis HLS/IP Design

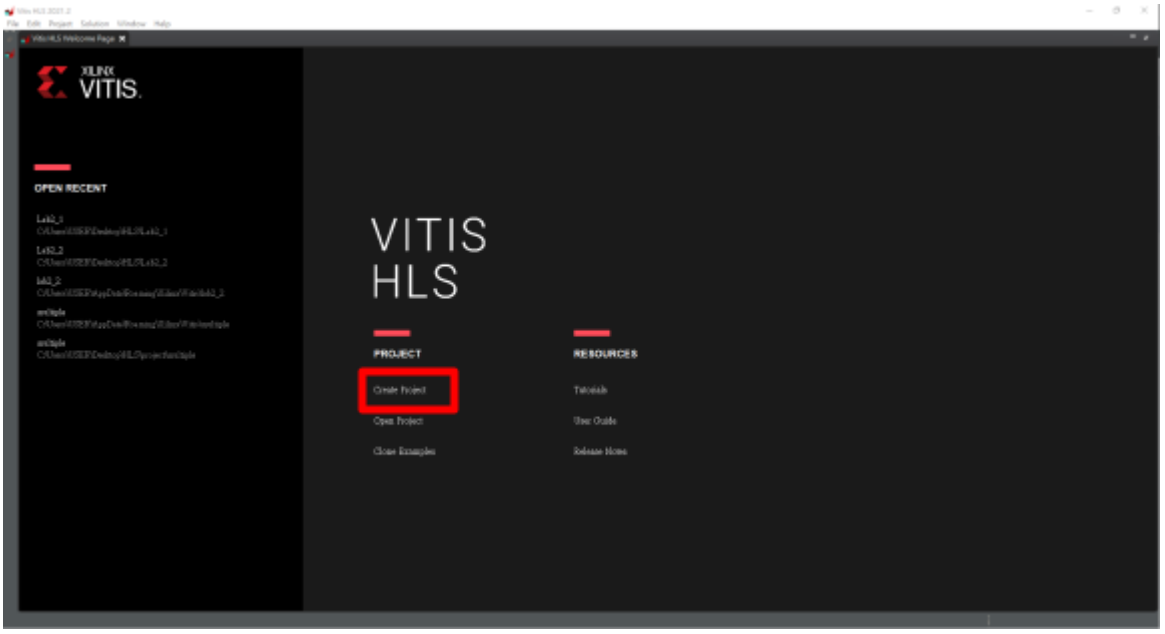
【施作環境為Windows】

啟動Vitis HLS開發套件。



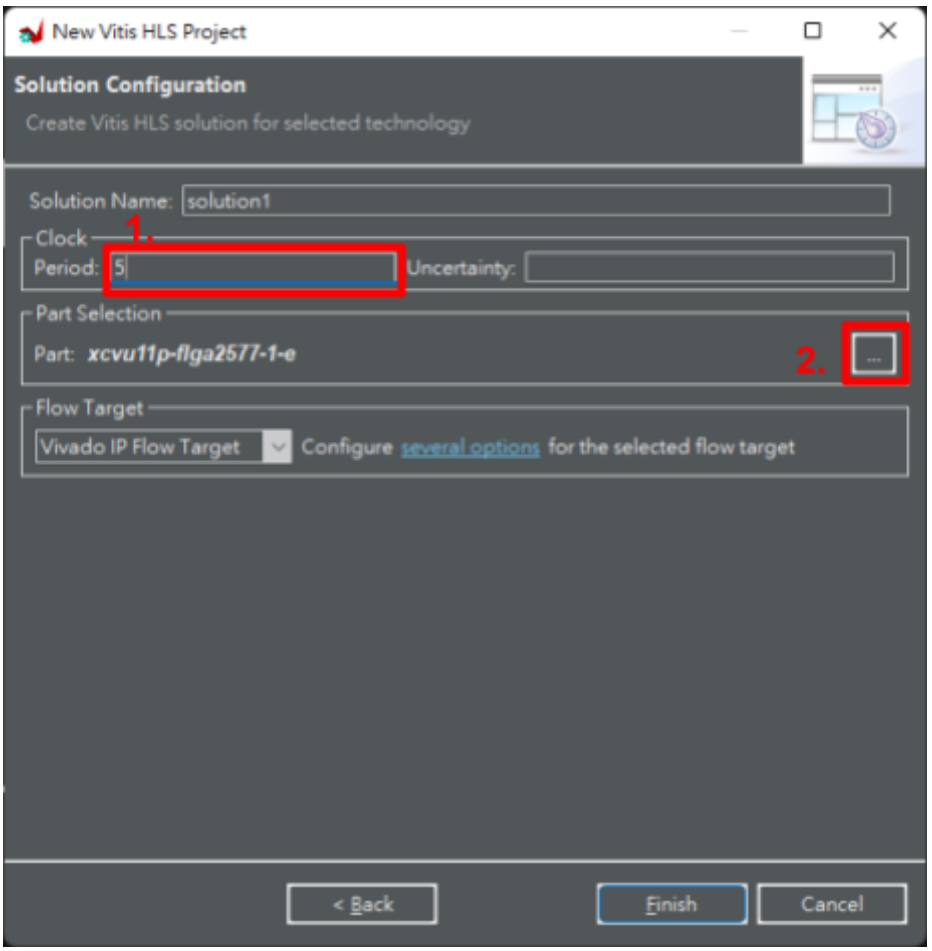
2.1.1. Create Source Project

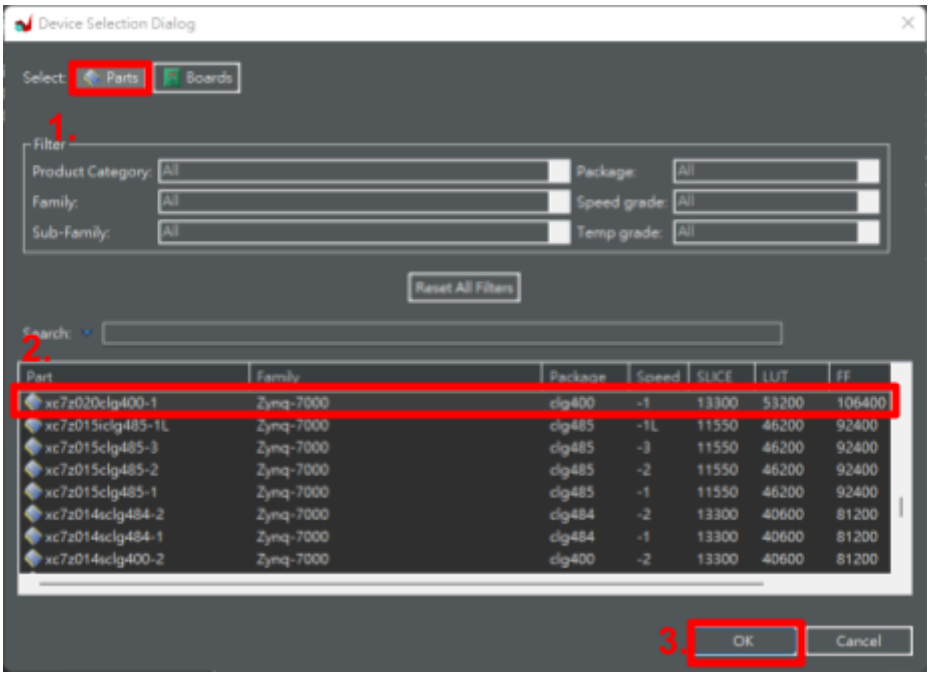
開啟新的專案，設定好專案存放路徑：



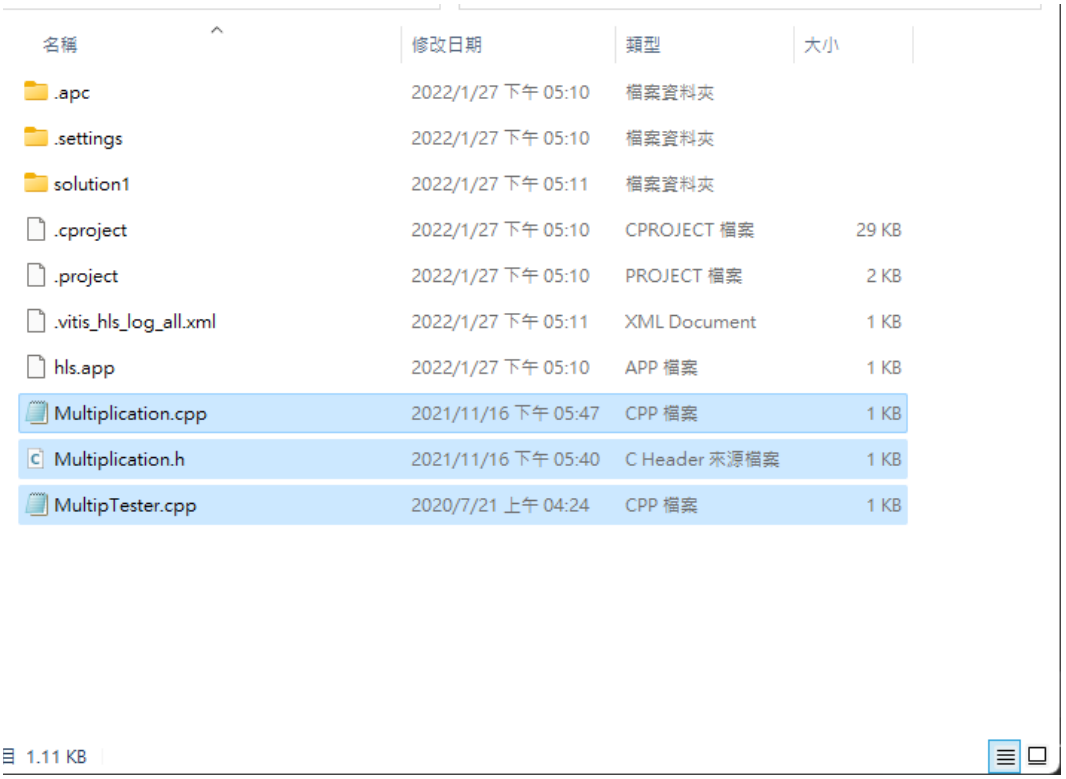
1.若PYNQ-Z2的FPGA可運行於時脈200MHz以上，可以選擇週期為5 ns。

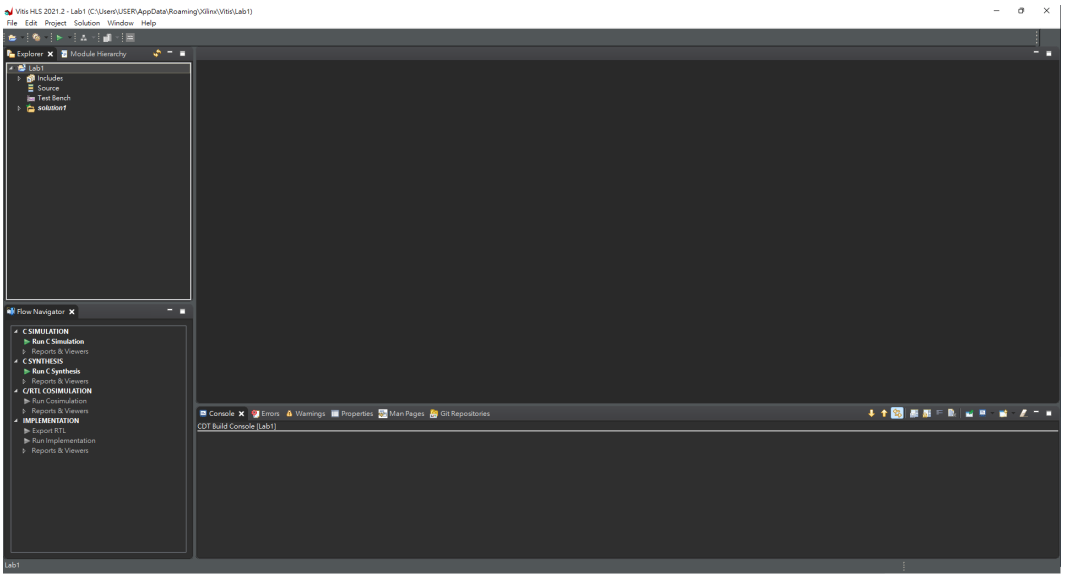
2.因vitis 2021.2版本讀不到pynq z2 board file，所以選用與pynq z2相同Part 的 **xc7z020clg400-1**





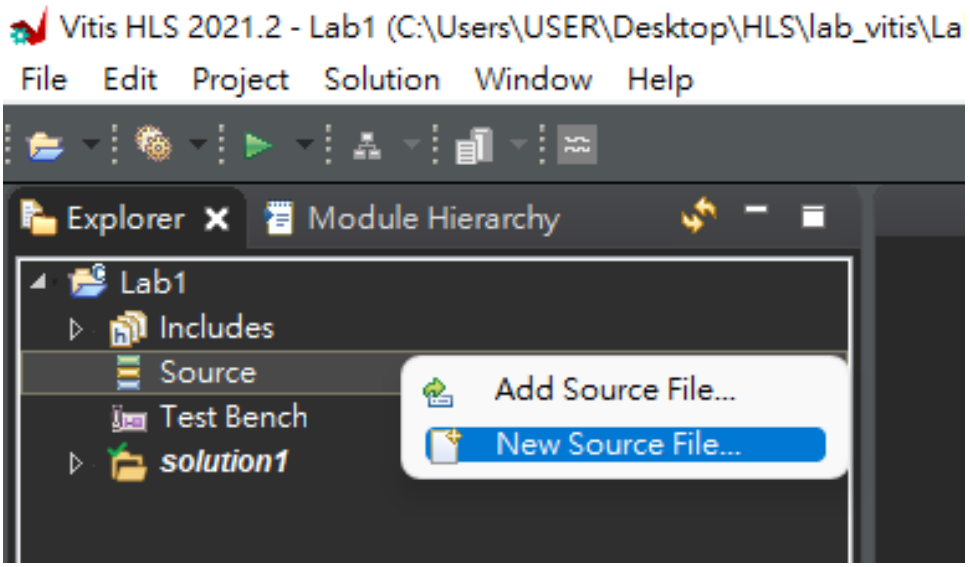
接著將hls_Multiplication中三個檔案放到vitis hls專案對應的位置



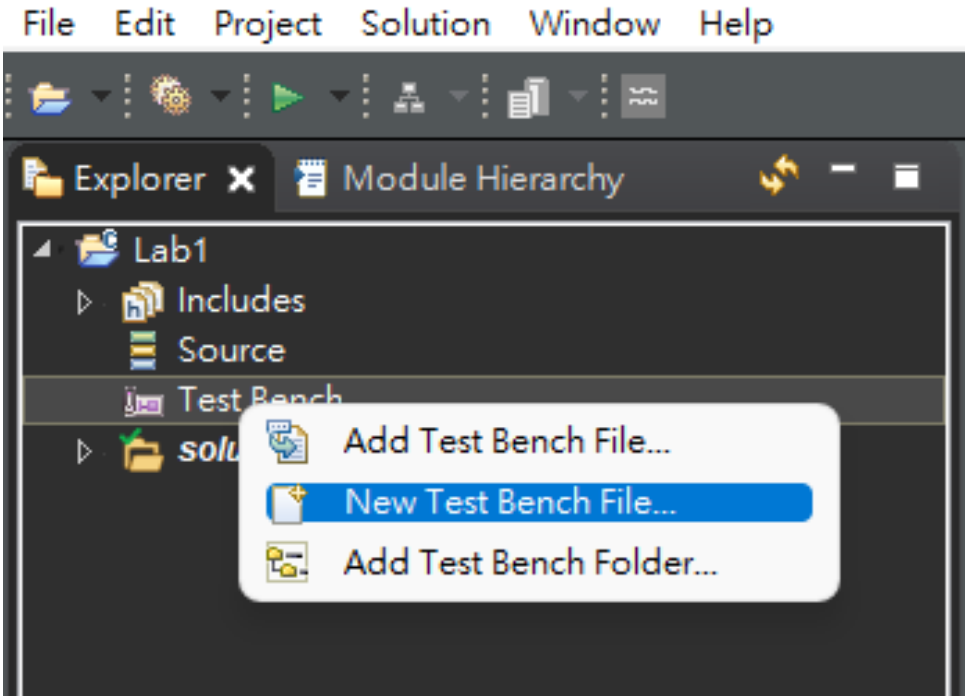


進入Vitis HLS專案IDE畫面後，加入Source/Test Bench的原始碼檔。

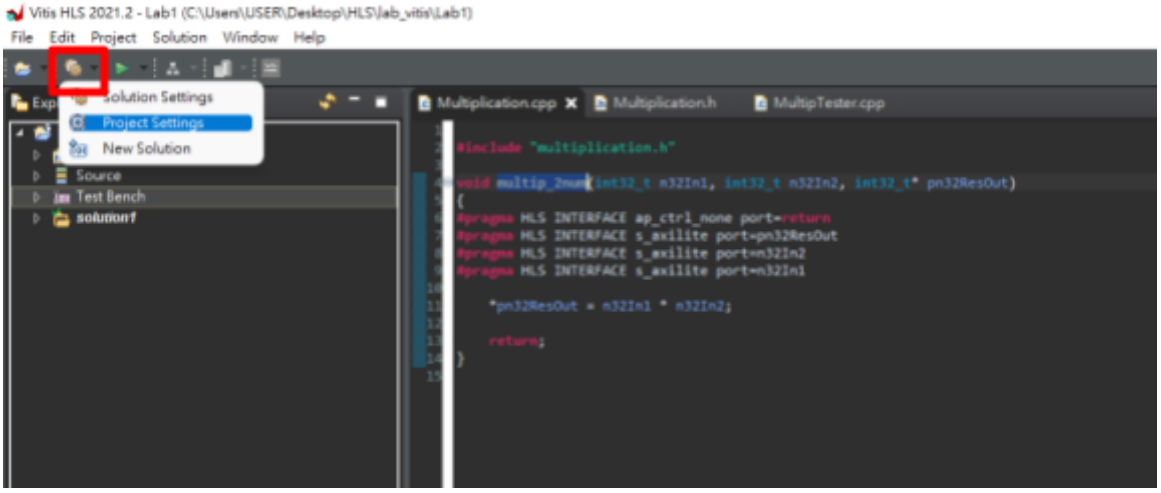
在專案Source按滑鼠右鍵加入新檔，在Source選擇Multiplication.cpp及Multiplication.h檔。(#Include部分注意大小寫問題。)

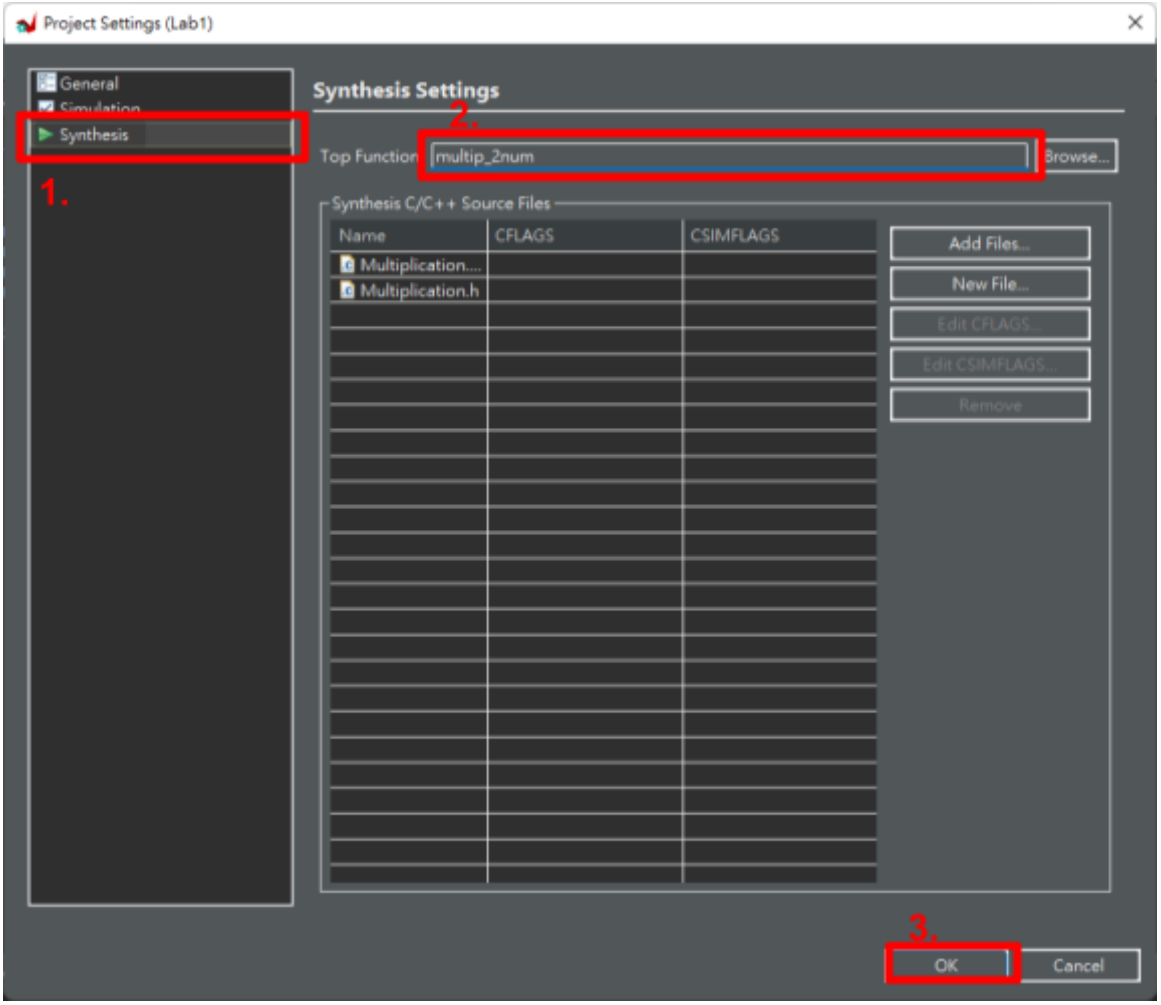


在專案Test Bench按滑鼠右鍵加入新檔，在Test Bench選擇MultipTester.cpp檔。



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)

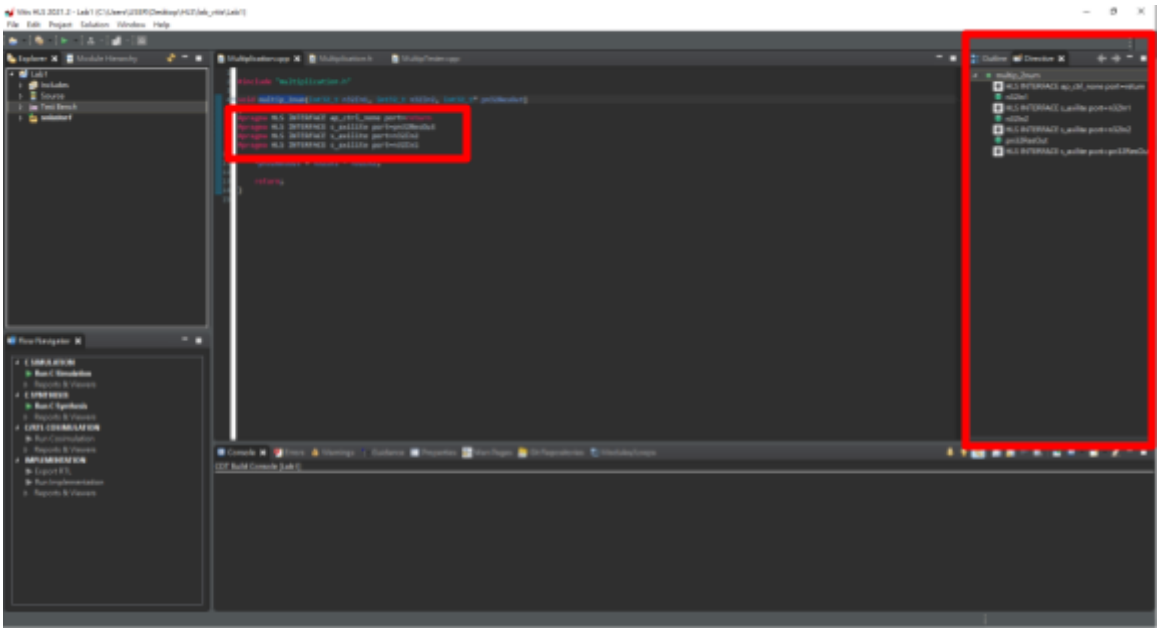




2.1.2. Directives Control

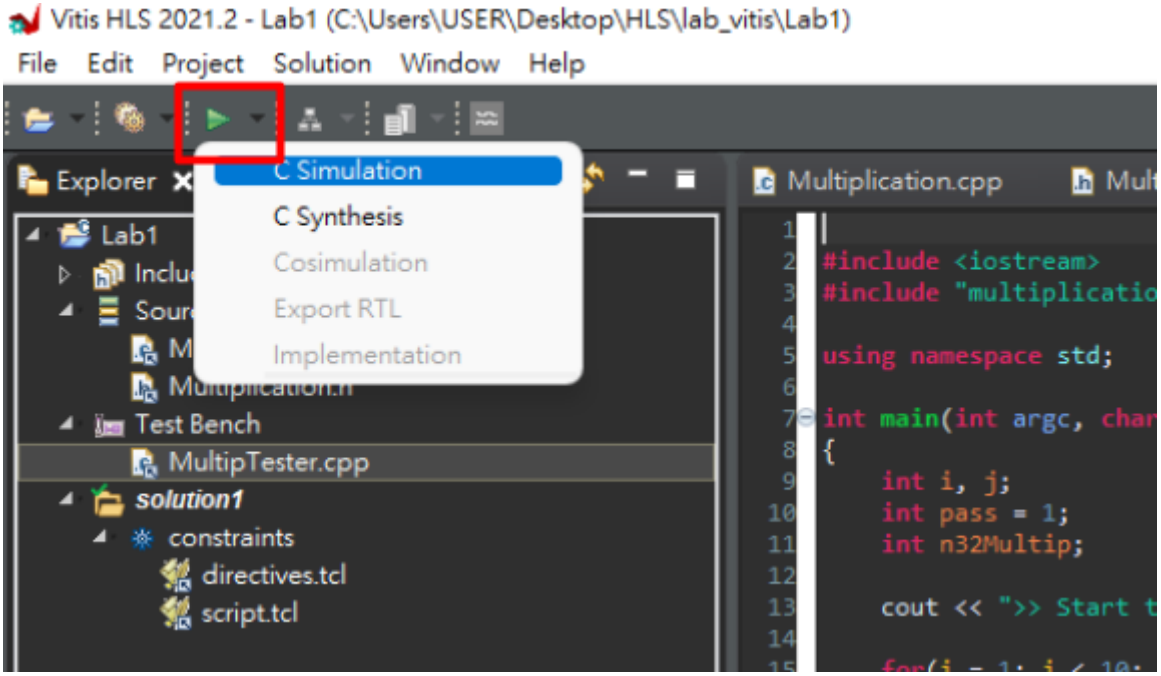
在開始Synthesis之前必須先加入directive描述。

加入directive有兩種方式，一種是inline方式(使用#pragma)，另一種是以directives.tcl來控制。Lab 1是使用第一種方式，所以不需另外增加directives。



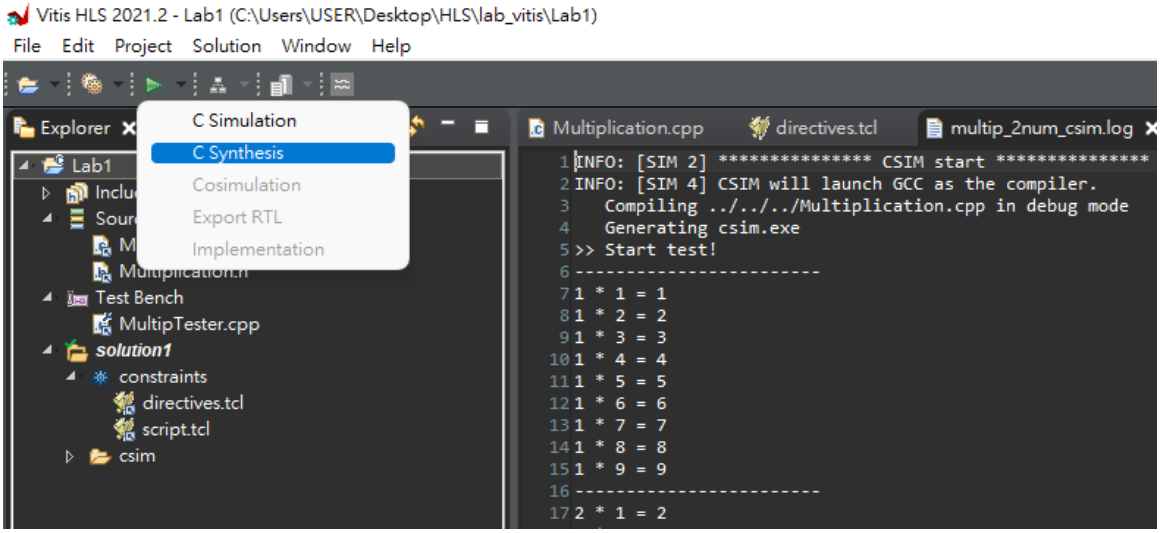
2.1.3. C Simulation

在Lab. #1裡提供Test Bench的C Simulation原始碼檔，已經有匯入專案。可執行C Simulation來驗證IP的執行結果。結果會顯示在下方console欄。

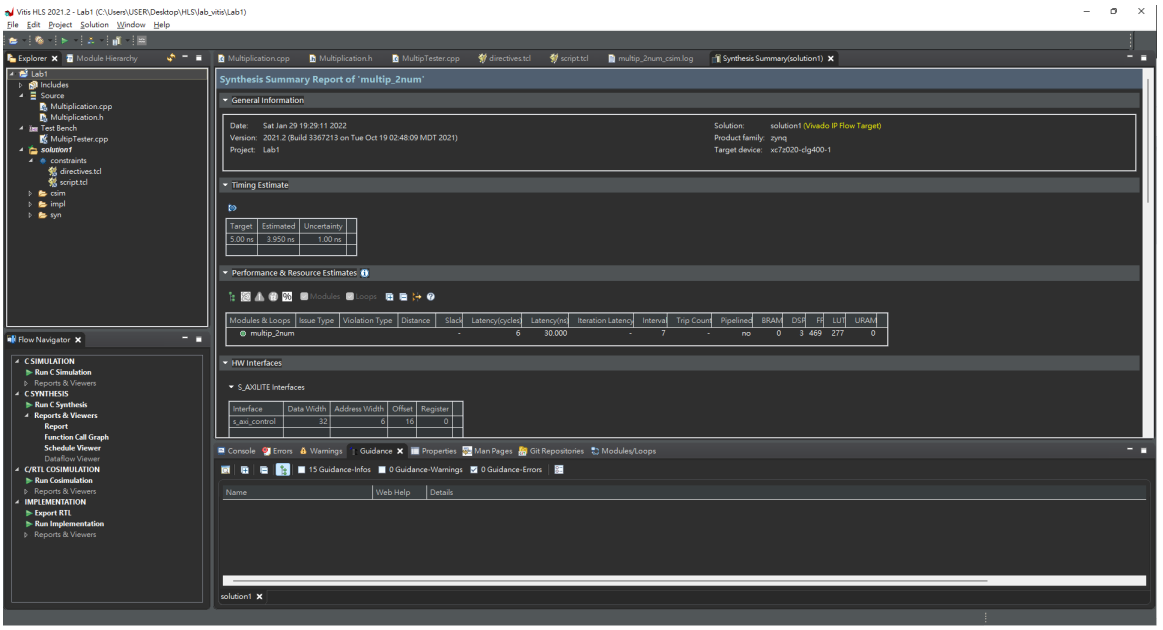


2.1.4. Synthesis

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。

若有slack / time violation，可調整period。

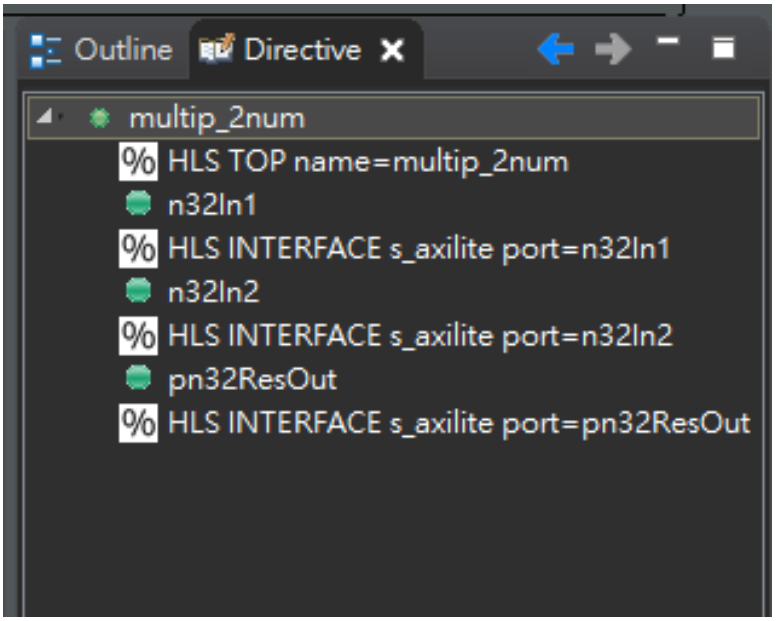
2.1.5. Cosimulation

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。

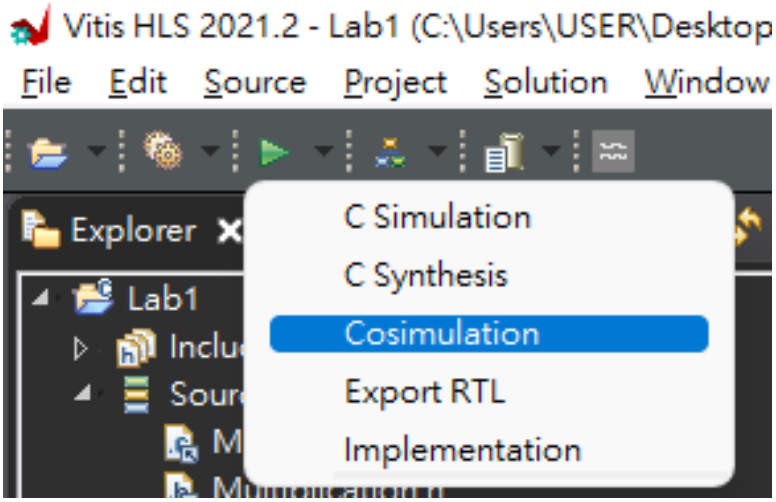
參考網址：

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

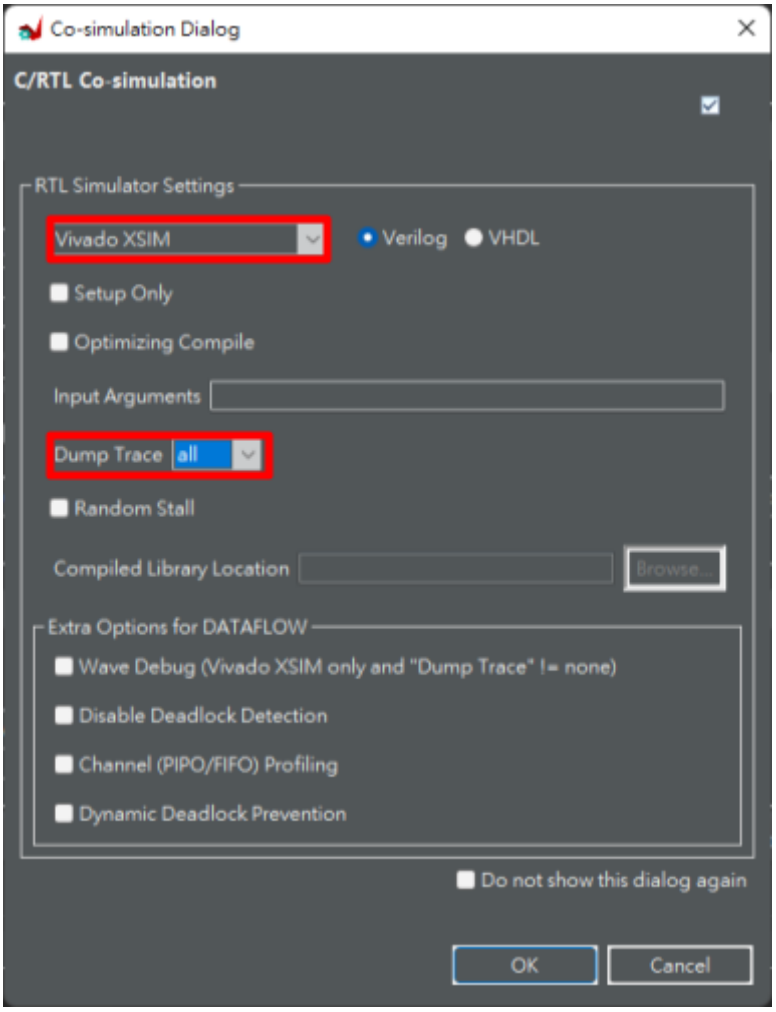
https://forums.xilinx.com/t5/High-Level-Synthesis-HLS/Using-ap-memory-with-ap-control-none/td-p/681187

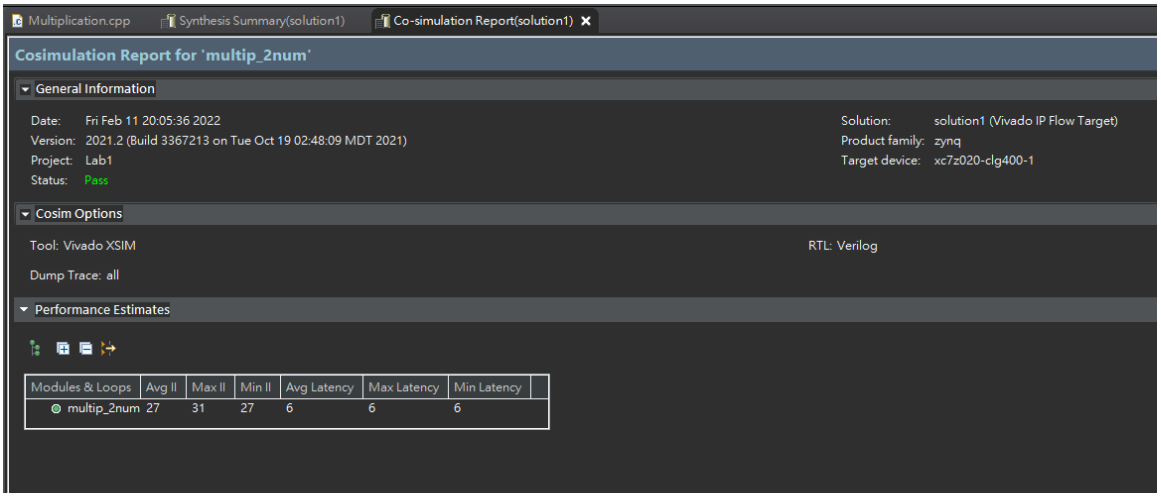


按下Cosimulation鍵來驗證設計。

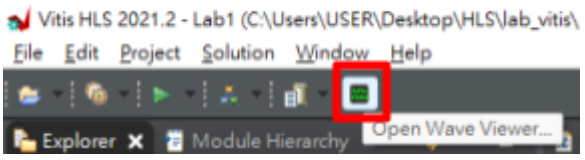


按下Cosimulation鍵後會彈出對話視窗，選擇Vivado XSIM，將dump trace選擇all



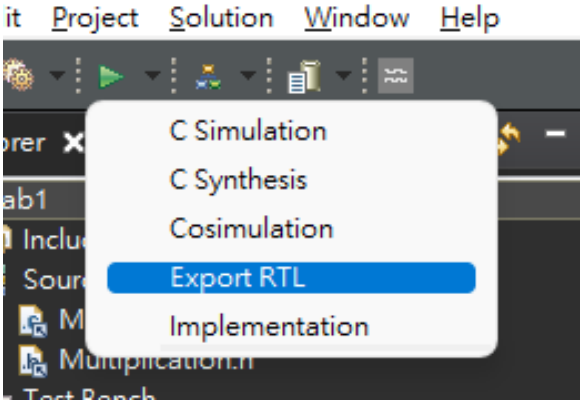


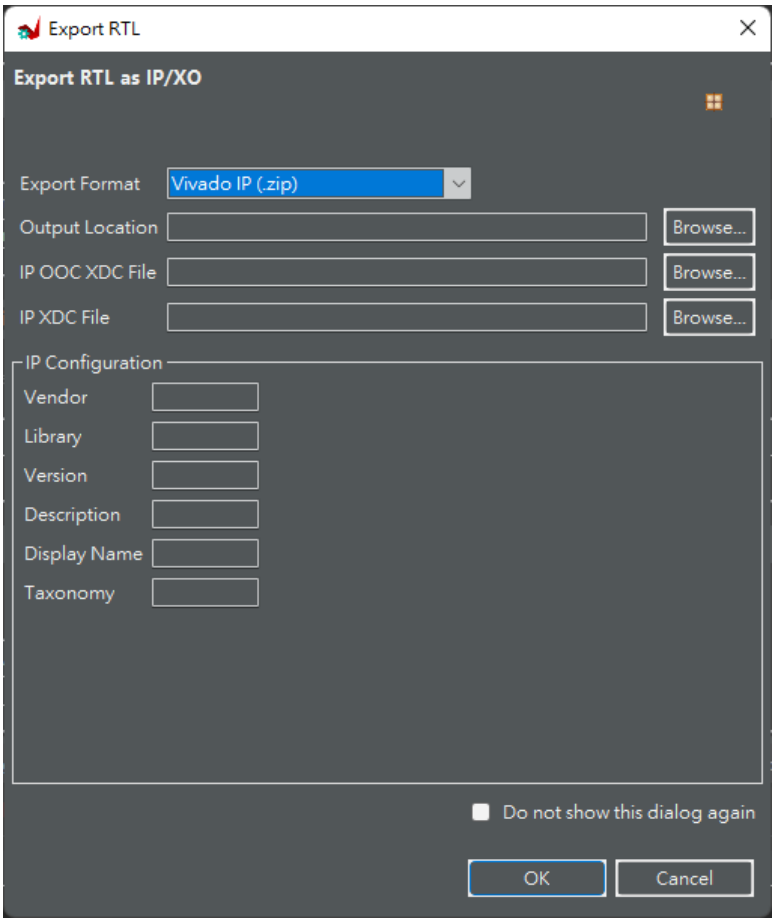
在輸出視窗檢視。要檢視波形可執行Open Wave Veiwer。



2.1.6. Export IP

完成IP的設計後，先從Vitis HLS匯出IP，之後在Vivado Design Suite的Vivado還需要匯入由Vitis HLS匯出的IP。現階段操作Vitis HLS匯出RTL功能，在匯出RTL前先還原原本的directive，之後重新Synthesis才是做出的設計：

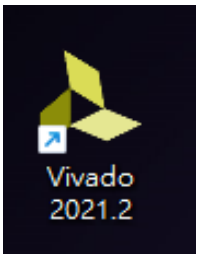




2.2. Vivado/Implement Flow

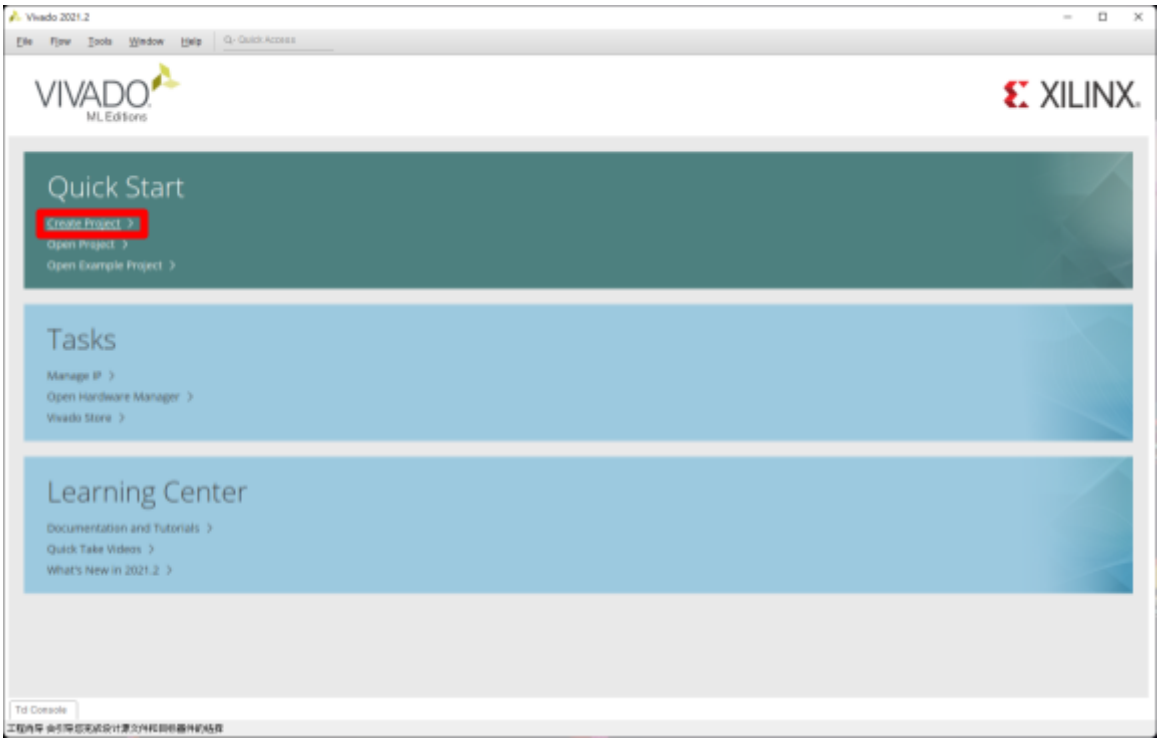
【施作環境為在使用者PC/laptop/notebook (Windows Base)。】

啟動Vivado Design Suite的Vivado開發套件。



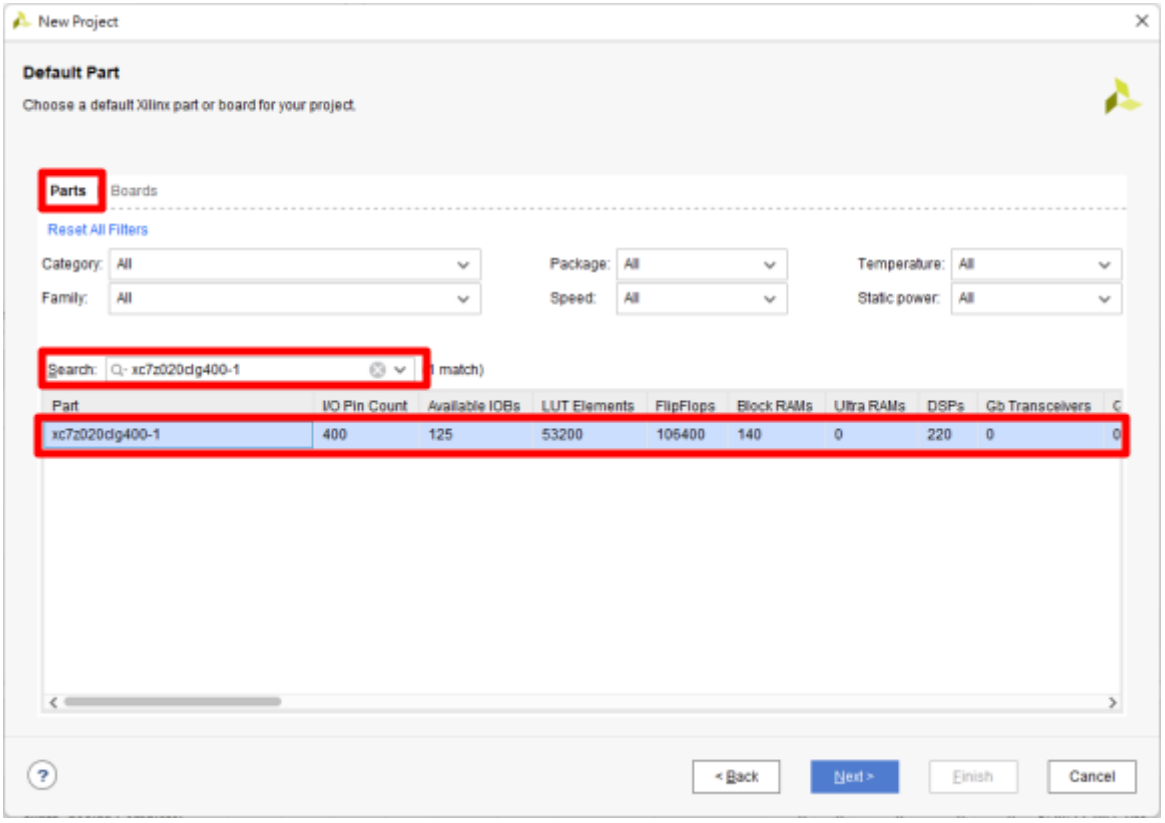
2.2.1. Create Design Project

開啟新的專案，設定好專案存放路徑：



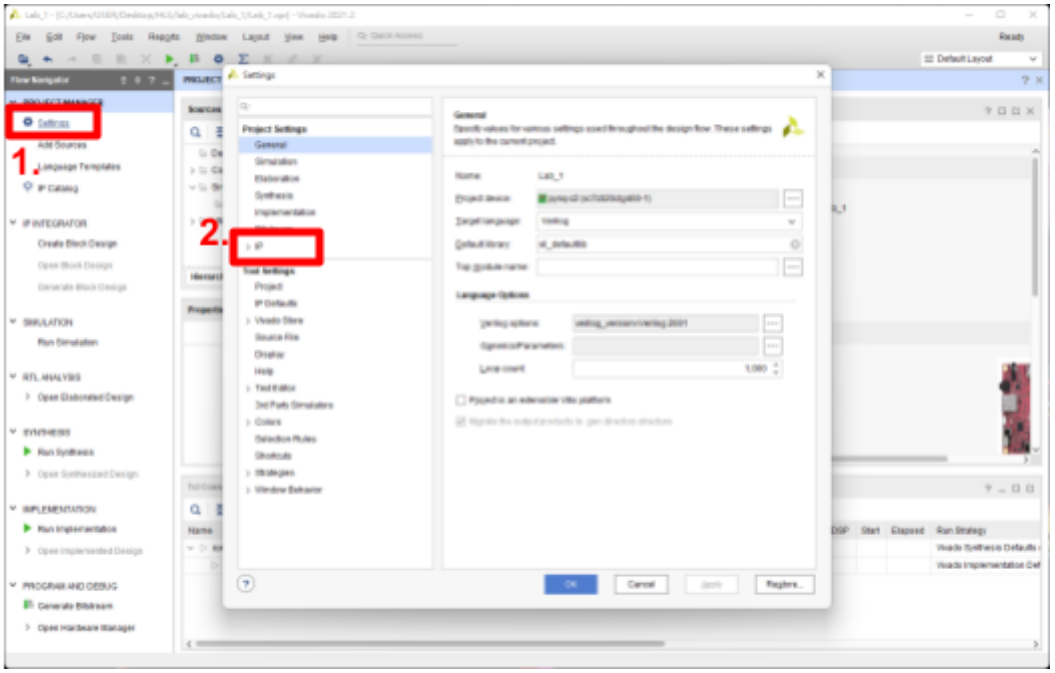
選擇好專案儲存路徑。RTL Project、sources 跟 constraints留空之後再加。

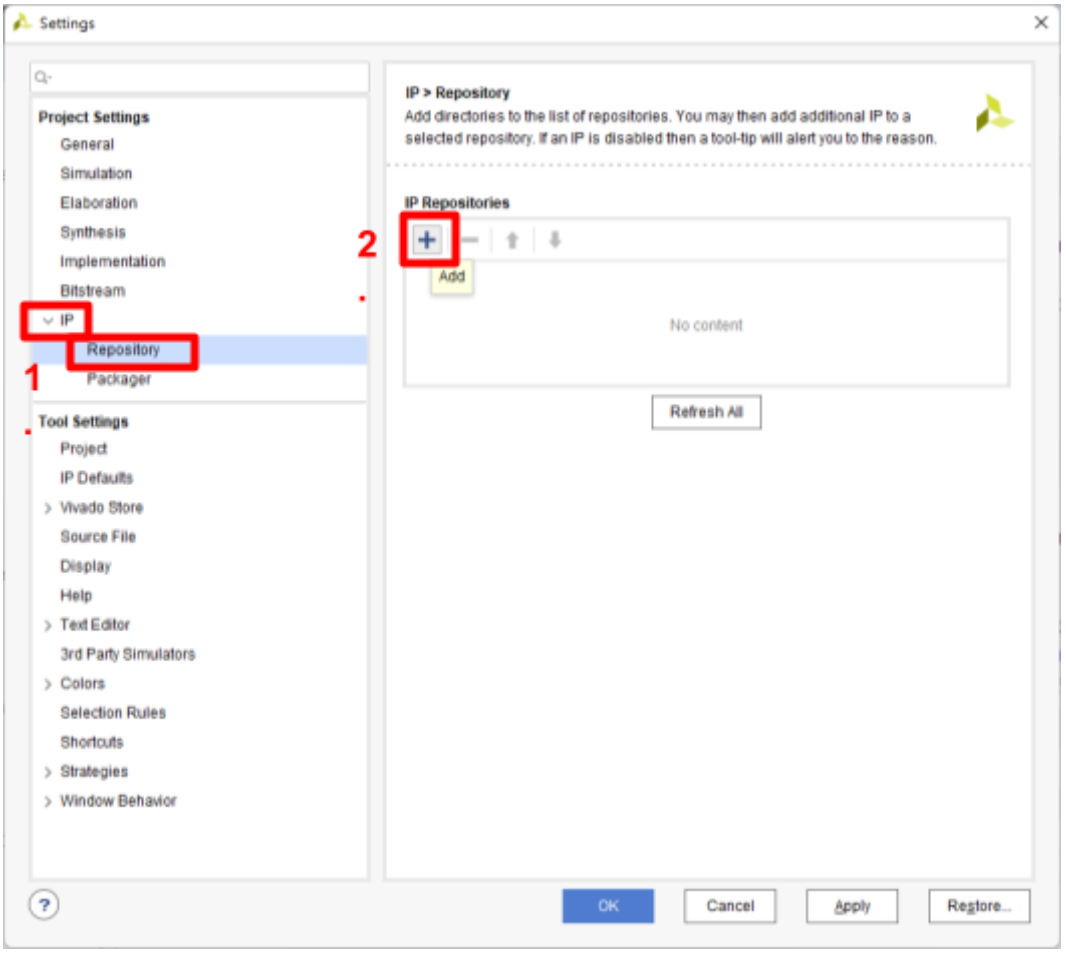
Parts選擇**xc7z020clg400-1**。



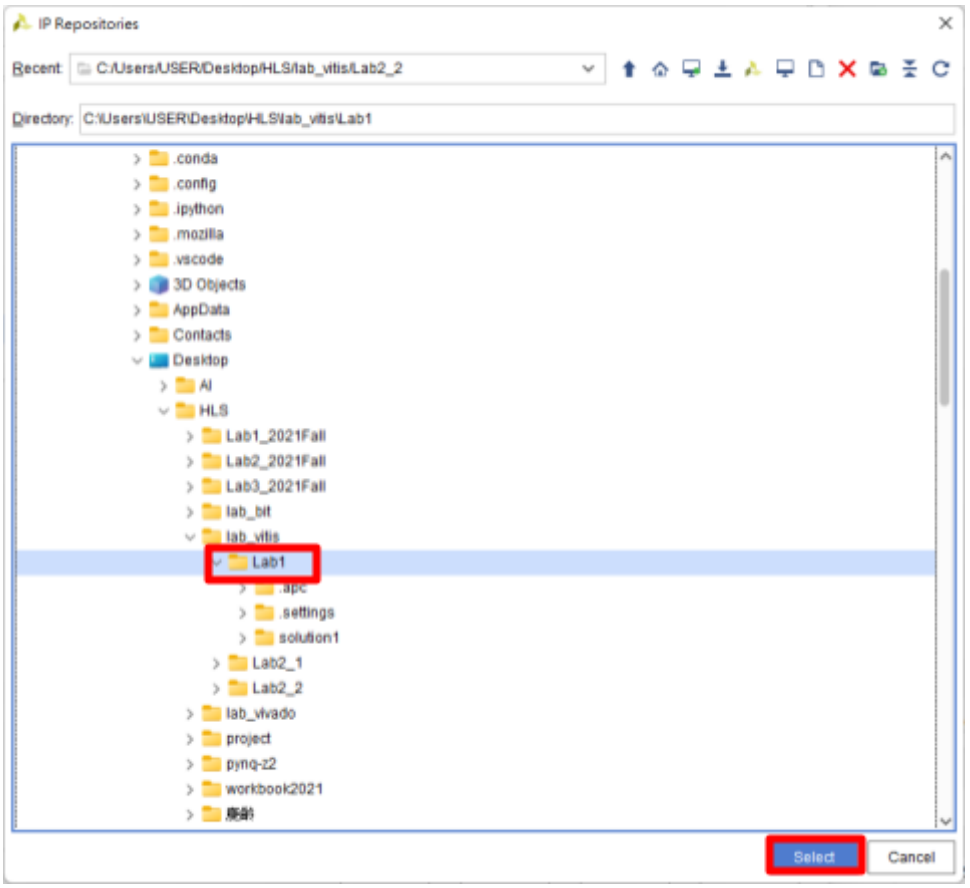
2.2.2. Import IP

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。



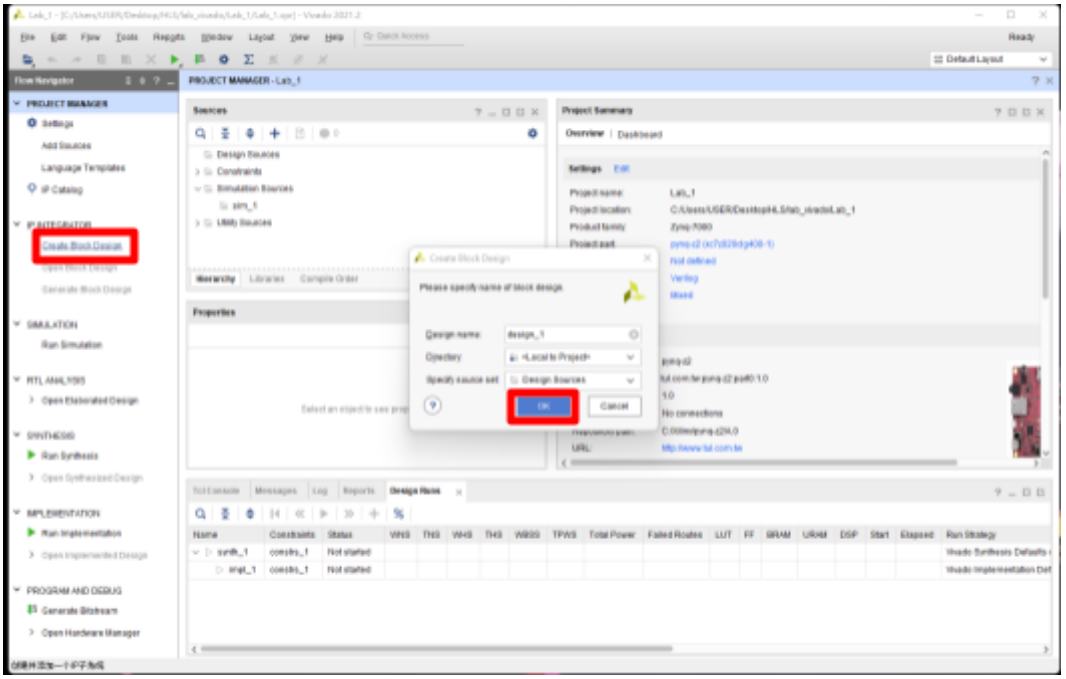


將IP Repositories指定到Vitis HLS專案目錄，下一步會匯入由Vitis HLS專案開發的IP

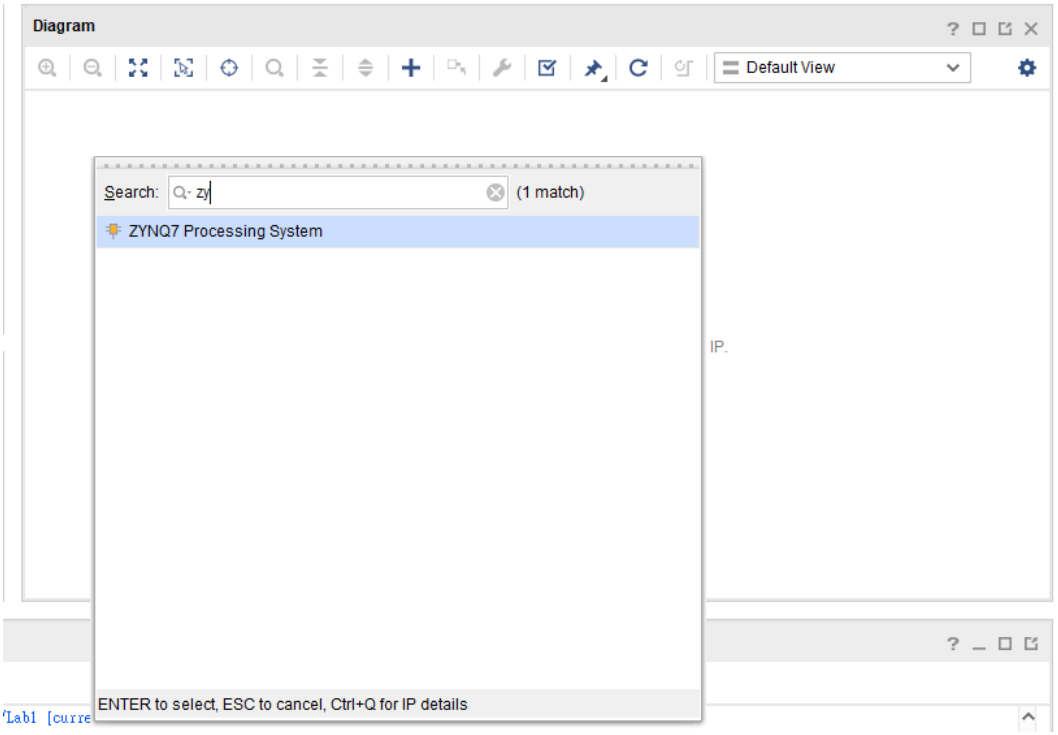


2.2.3. Block Design

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200MHz。



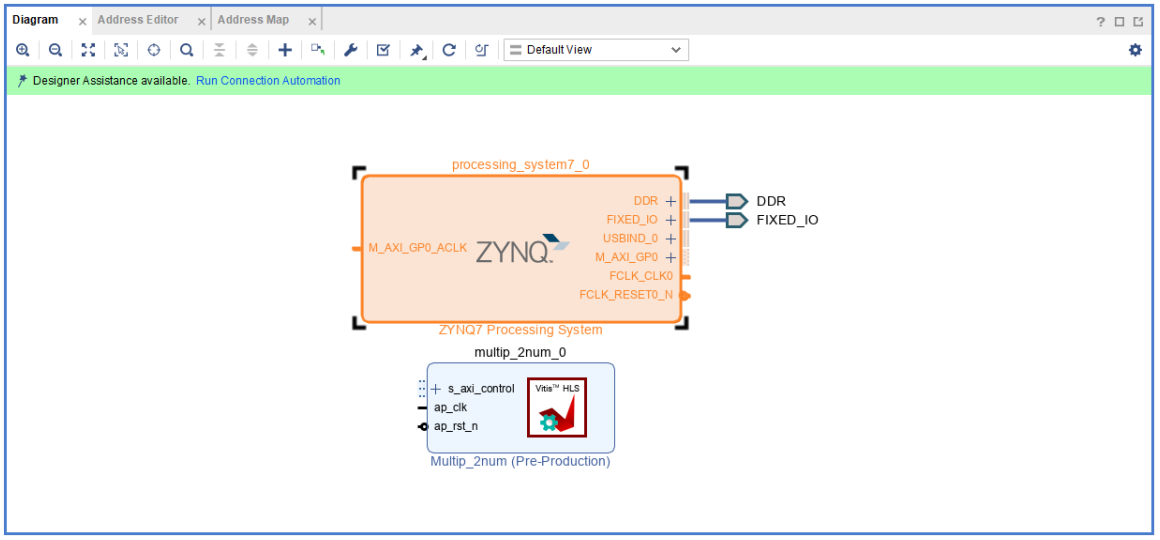
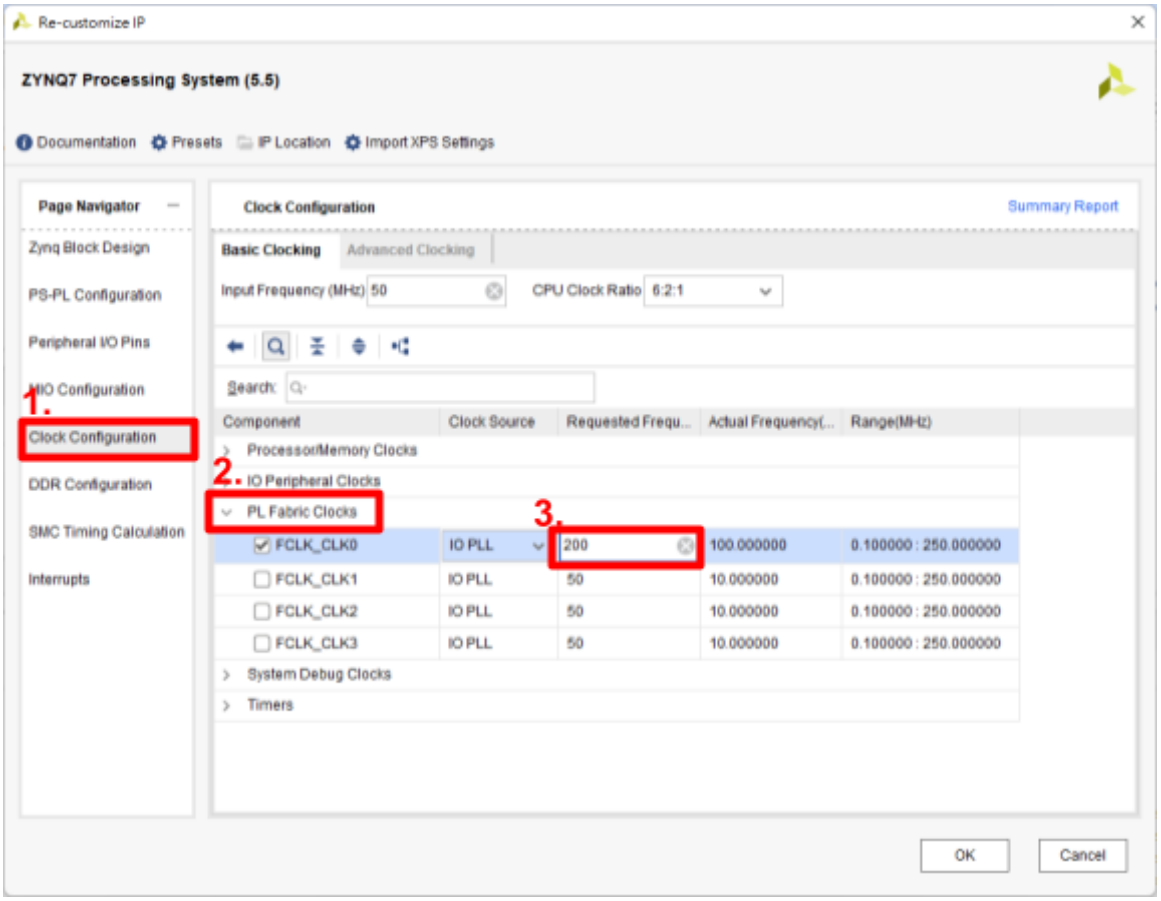
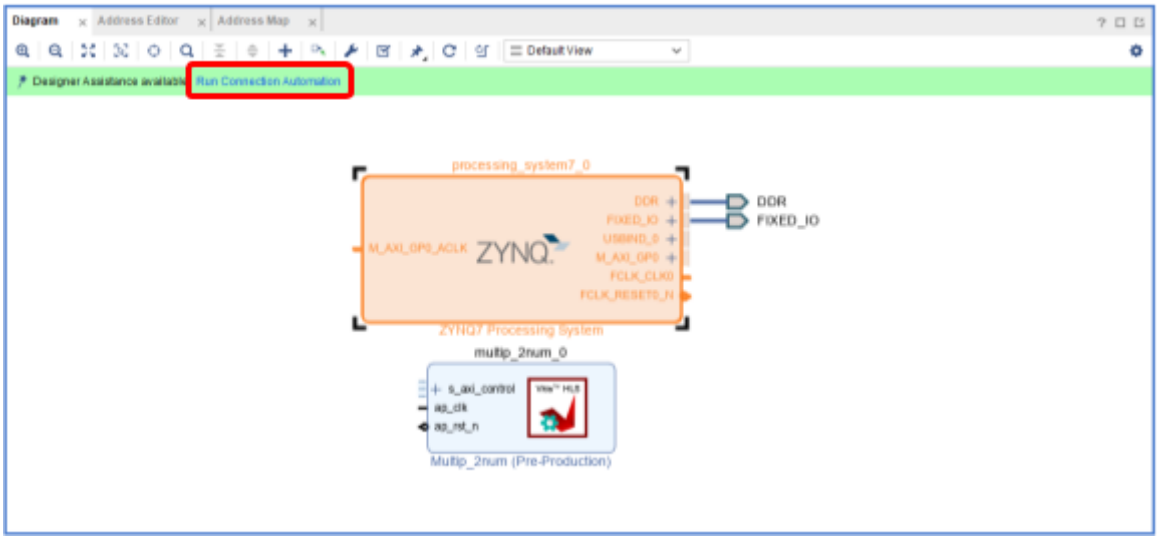


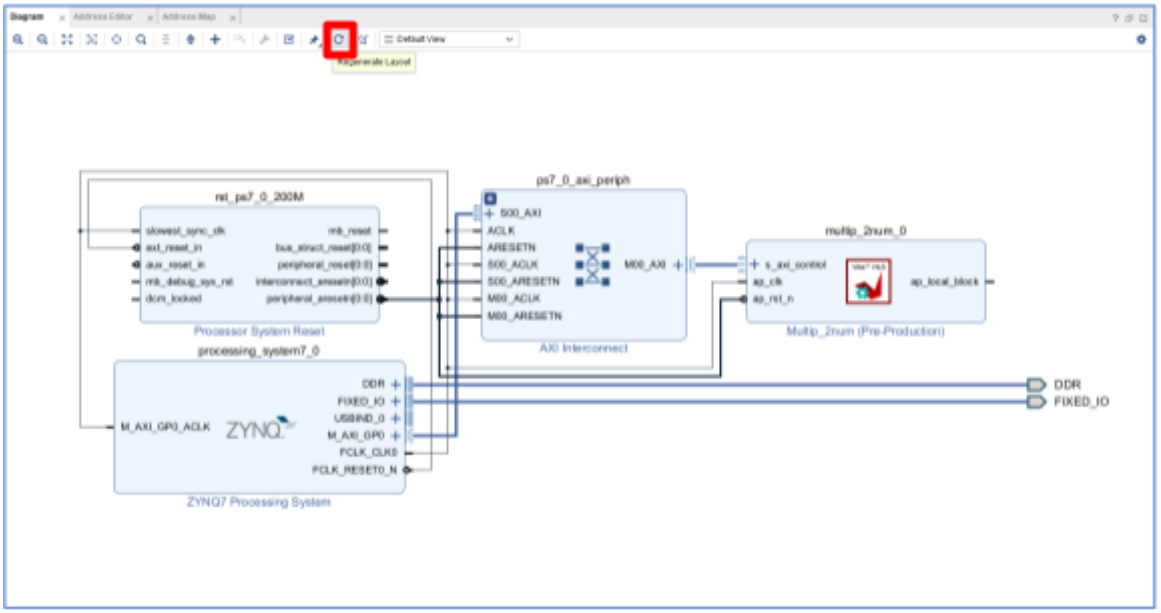
Diagram
Address Editor
Address Map
Default View
Designer Assistance available. Run Connection Automation
processing_system7_0
M_AXI_GP0_ACLK
ZYNQ
DDR
FIXED_IO
USBIND_0
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
DDR
FIXED_IO
ZYNQ7 Processing System
multip_2num_0
s_axi_control
ap_clk
ap_rst_n
Vitis™ HLS
Multip_2num (Pre-Production)

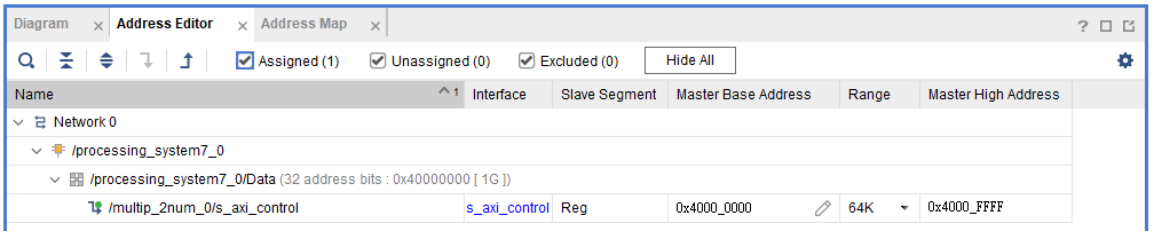

Re-customize IP
ZYNQ7 Processing System (5.5)
Documentation
Presets
IP Location
Import XPS Settings
Page Navigator
Zynq Block Design
PS-PL Configuration
Peripheral I/O Pins
MIO Configuration
Clock Configuration
DDR Configuration
SMC Timing Calculation
Interrupts
1.
Clock Configuration
Summary Report
Basic Clocking
Advanced Clocking
Input Frequency (MHz) 50
CPU Clock Ratio 6:2:1
Search:
Component
Clock Source
Requested Frequ...
Actual Frequency(...
Range(MHz)
Processor/Memory Clocks
2.
IO Peripheral Clocks
PL Fabric Clocks
3.
FCLK_CLK0
IO PLL
200
100.000000
0.100000 : 250.000000
FCLK_CLK1
IO PLL
50
10.000000
0.100000 : 250.000000
FCLK_CLK2
IO PLL
50
10.000000
0.100000 : 250.000000
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。

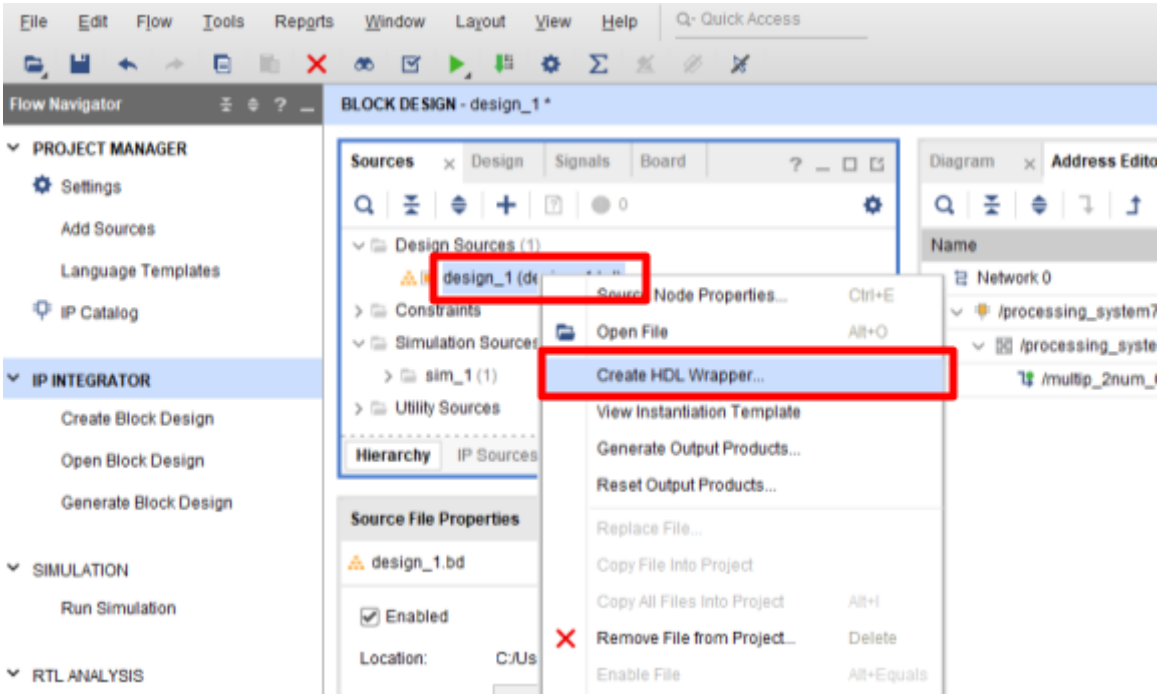


完成後整個完整的diagram圖，可點選上方regenerate layout，檢查接線是否有誤，接著可切換到Address Editor tab檢視memory map：



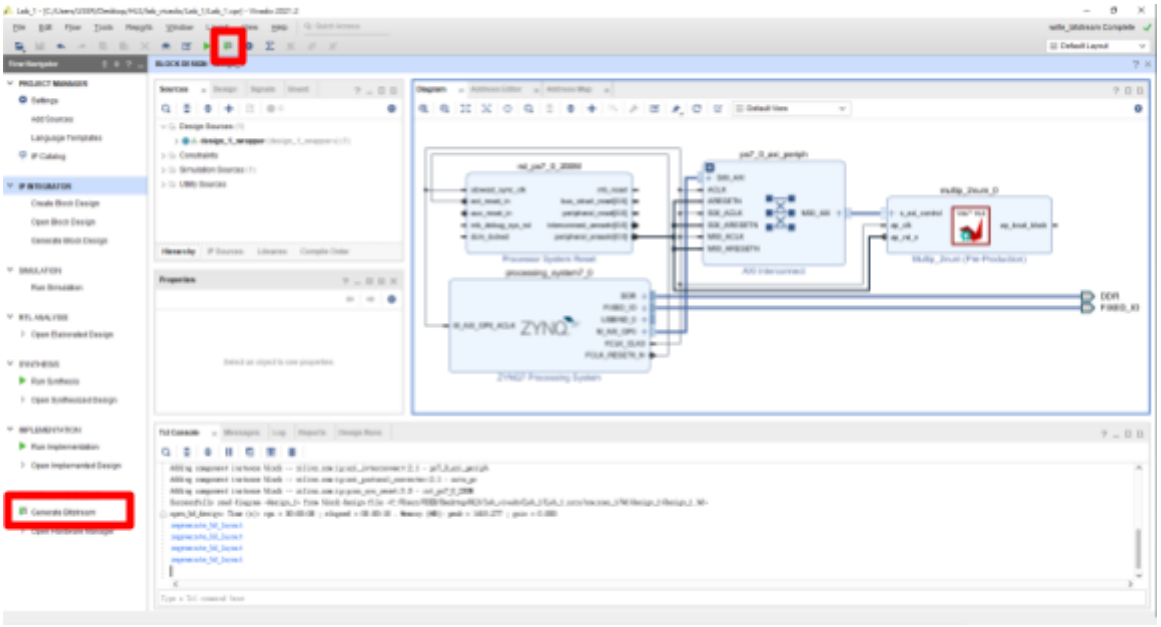


接下來進行HDL Wrapper動作：在Design子視窗Source tab頁面Design Sources的design_1.bd點擊滑鼠右鍵進行HDL Wrapper：



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

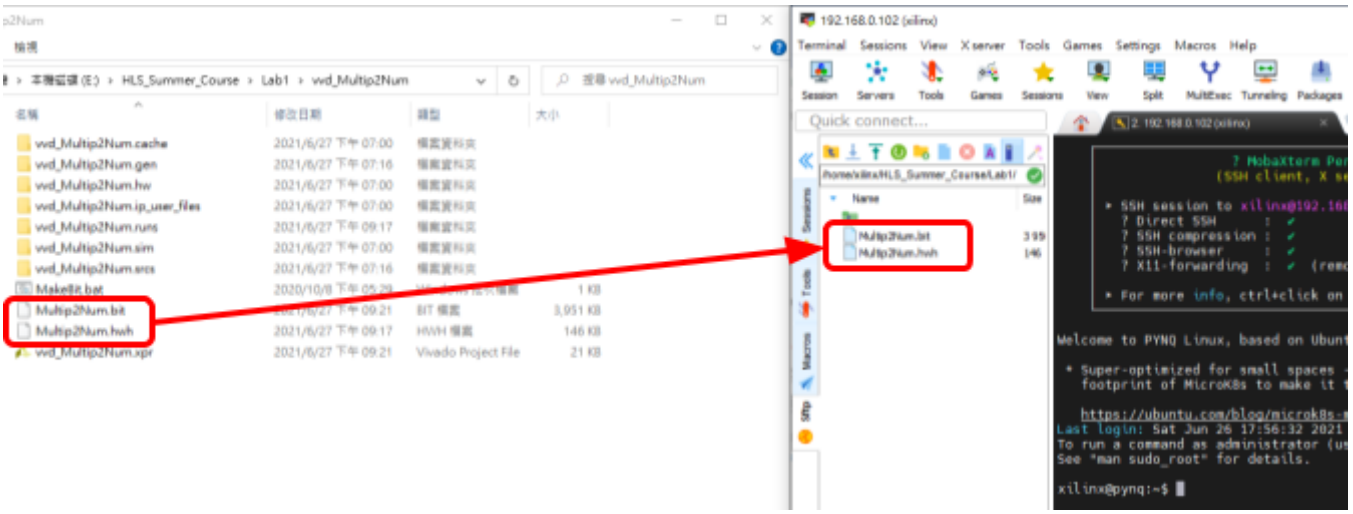
產生FPGA所需要的bit-stream file，由於要產生bit-stream file需要經過Synthesis/Placement/Routing的步驟，但可以省去單步執行的流程以一鍵執行由Vivado IDE自動完成所有流程。在Vivado專案IDE畫面，在專案管理點擊Generate Bitstream選項或由工具列按下Generate Bitstream按鍵。



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

2.2.5. 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

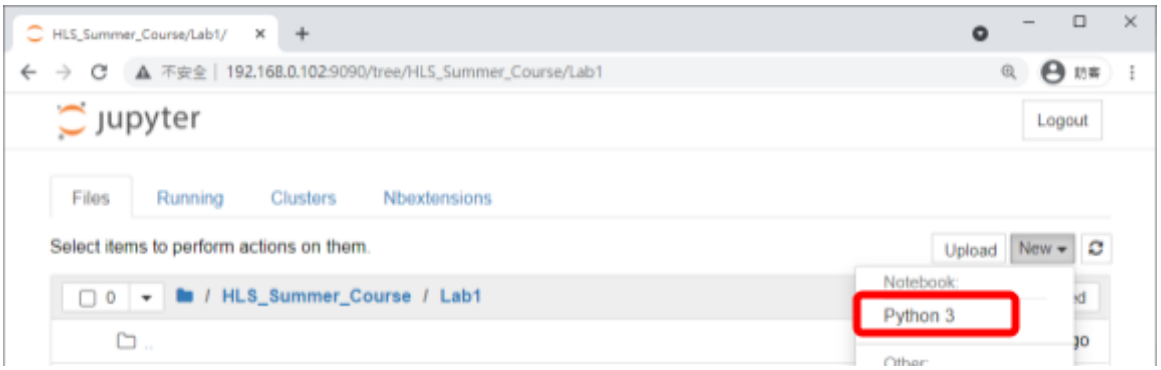
hwh檔位置 \Lab_1\Lab_1.gen\sources_1\bd\design_1\hw_handoff\design_1.hwh

更改檔名，並記得存放在pynq版上的位置。

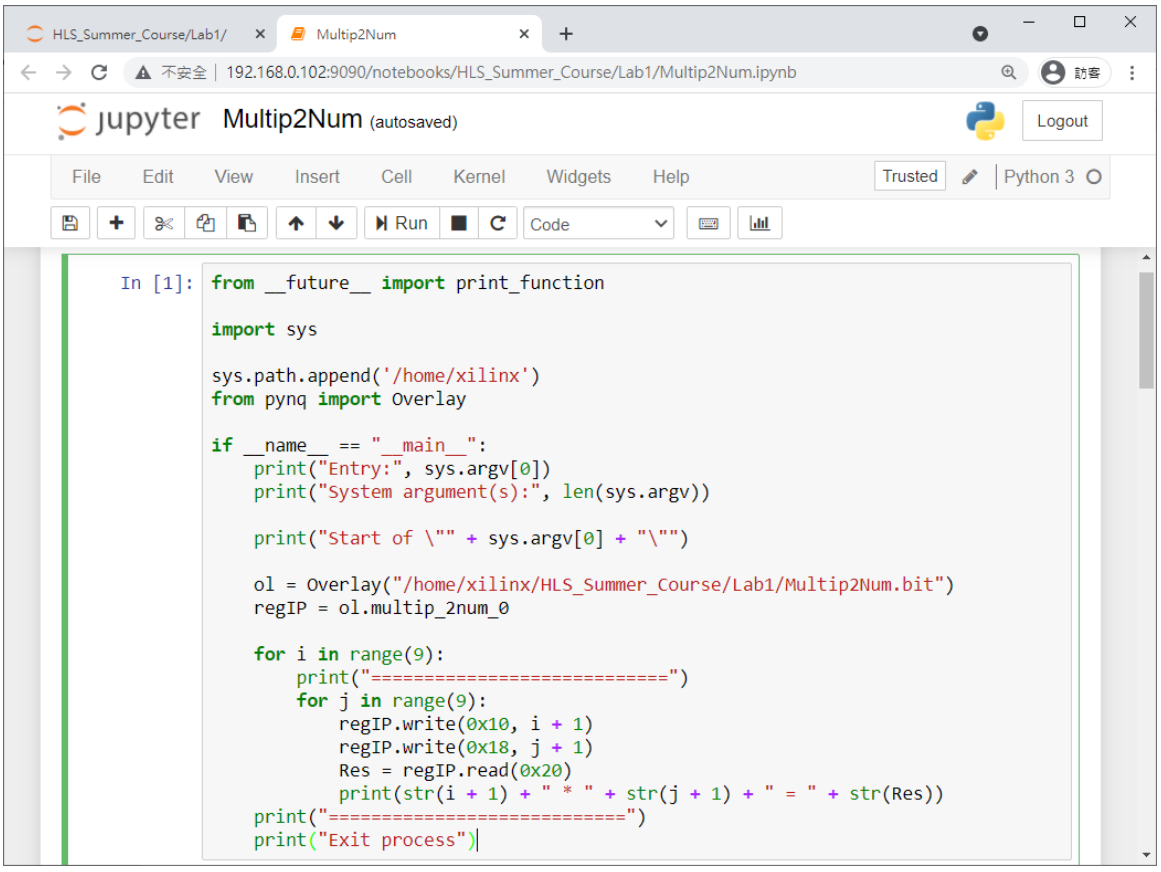
2.3. PYNQ/Host Program

2.3.1. 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)



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

2.3.2. 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