

# Lab8

## Microblaze

2022.05.25

指導教授：范志鵬

主講人：王星耀



# Outline

- ❖ 實驗介紹
- ❖ 實驗流程
- ❖ 實驗步驟

# Outline

- ❖ 實驗介紹
- ❖ 實驗流程
- ❖ 實驗步驟

# 實驗介紹

- ❖ 本實驗介紹如何添加一個Microblaze的soft core，並添加1個GPIO的介面IP，以連接板卡上的LEDs；同時添加uartlite的IP core，以連接板卡上的串列埠，以終端機發送的ASCII碼轉換成2進位制，在LED上顯示。
- ❖ 本實驗不需要寫任何的Verilog！

# MicroBlaze 簡介

- ❖ MicroBlaze嵌入式軟核是一個被Xilinx公司優化過的可以嵌入在FPGA中的RISC處理器軟核，具有運行速度快、佔用資源少、可配置強等優點，和其他外設IP核一起，可以完成可編程系統芯片(SOPC)的設計。
- ❖ FPGA是可編程的硬件邏輯電路，而MicroBlaze是一種處理器電路，使用MicroBlaze就相當於在FPGA內部做了一個CPU在裡面，可以用C語言編寫程序，在這個CPU上跑C語言的軟件程序。FPGA偏向邏輯，做控制比較麻煩，CPU做控制比較方便。

# IP 簡介

- ❖ 矽智財，全稱智慧財產權核(intellectual property core)，通常被用於滿足特定的規格，是可以經過事前設計、定義以及驗證重複使用的功能區塊。
- ❖ 由於其晶片具有特定功能的積體電路設計技術，同時具備模組化的特性，也因其有效性經過驗證並且能夠被重複使用，因此，當相關廠商以矽智財作為基礎設計，就可以從既有的資料庫當中，尋找出其他相對應的矽智財進一步結合，這樣不僅可以組成具備複雜功能的IC外，也能縮短新產品設計所需要的時間。

# IP 簡介

- ❖ 更簡單來說，矽智財就是在IC設計的過程中，必須使用到且能簡化設計流程的智慧財產權，扮演的角色極為重要。
- ❖ 矽智財公司主要營收來源大約可分為三個部分，分別為前期的授權費用、當產品正式銷售後支付一定比例的權利金以及其他費用，其產業進入門檻相對較高，通常矽智財公司都能有較高的毛利率。在矽智財授權與外包需求與日俱增的帶動之下，台灣相關公司包含智原、創意等。
- ❖ 再更簡單來說，一個好的IP可以賣錢。

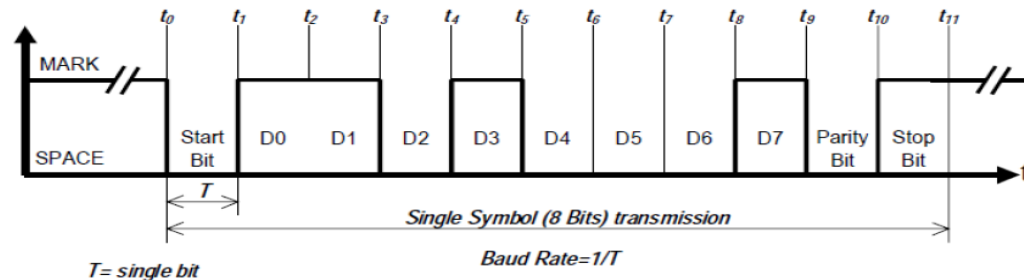
# UART 簡介

- ❖ UART(Universal Asynchronous Receiver/Transmitter)是通用非同步收發器的英文縮寫，它包括了RS232、RS449、RS423、RS422和RS485等介面標準規範和匯流排標準規範。現在PC的COM 接口均為RS232。
  - ◆ 若配有多個非同步串列通信口，則分別稱為COM1、COM2，COMx。
- ❖ UART的工作就是從CPU一次接收8bits的資料(parallel)，然後將這些資料 1次 1bit 的送往周邊設備(serially)。
- ❖ 同時，UART還可以接收周邊設備傳送來的資料，當組成8bits時，再將資料送CPU。



# 鮑率 baud rate

- ❖ 每秒傳送的位元數，單位為bps(bits per second)，以此次實驗為例，為115200 bps。
- ❖ 開始傳送資料時，須先發出一個低電位訊號(0)，當作Start bit，從低位元開始，傳送8bits資料後，需要傳送1bit High訊號(1)作為Stop bit，每bit時間都需要維持1個鮑率時間長度。



# UART 簡介-RX/TX

## ❖ RX :

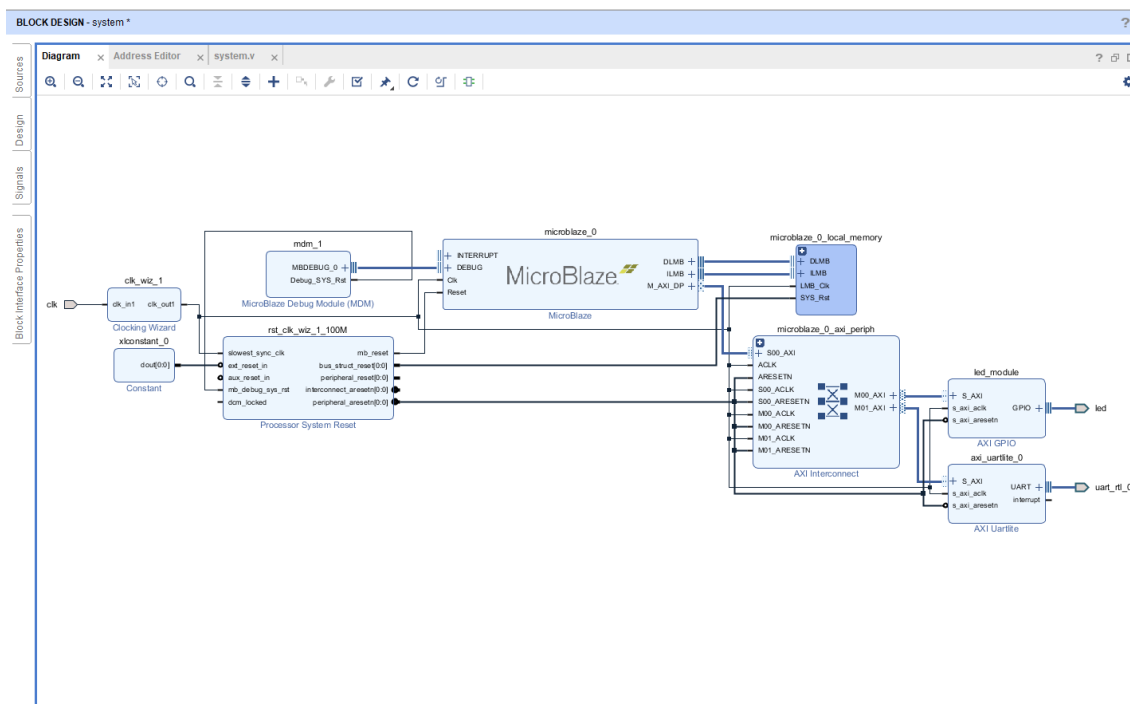
- ◆ 以鍵盤輸入一文字，並將其轉為ASCII code，再透過UART傳送至FPGA板，再經由七段顯示器顯示其16進制表示法。

## ❖ TX :

- ◆ 以開關作為輸入8bits訊號，並透過按壓按鍵，將訊號透過UART傳輸至電腦，再以「Terminal」程式，收取com port訊號。

# Block design 簡介

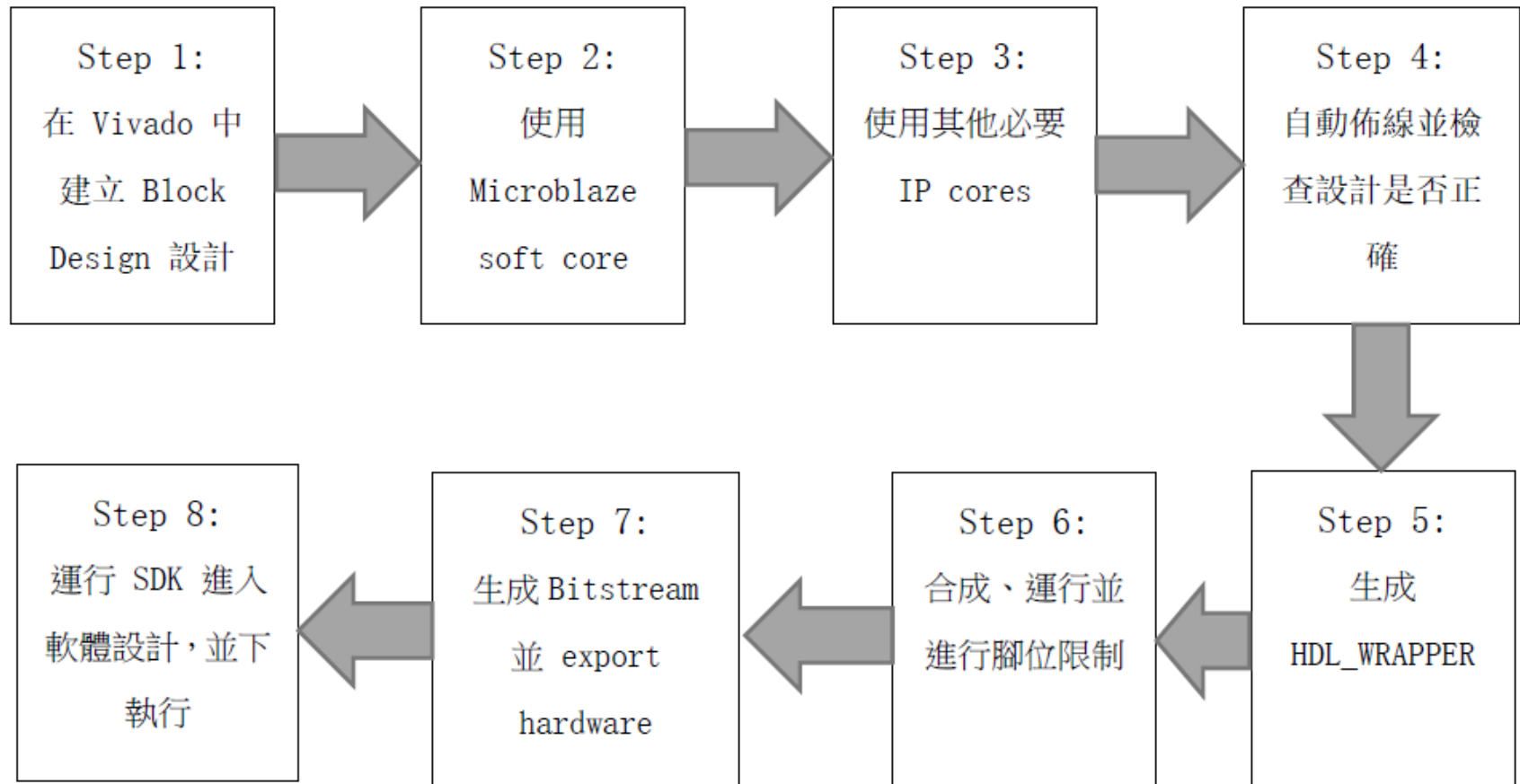
- ❖ Block design的創建類似於視圖的創建，即在視圖中中可以拖動我們的BLOCK。



# Outline

- ❖ 實驗介紹
- ❖ 實驗流程
- ❖ 實驗步驟

# 實驗流程

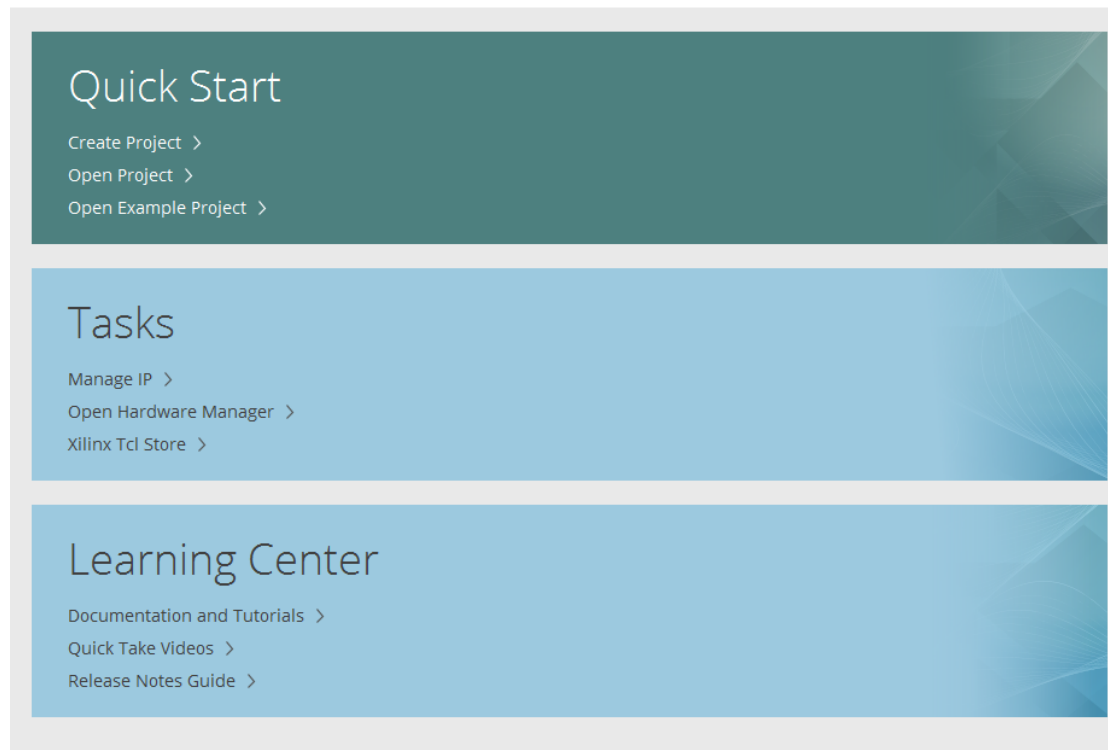


# Outline

- ❖ 實驗介紹
- ❖ 實驗流程
- ❖ 實驗步驟

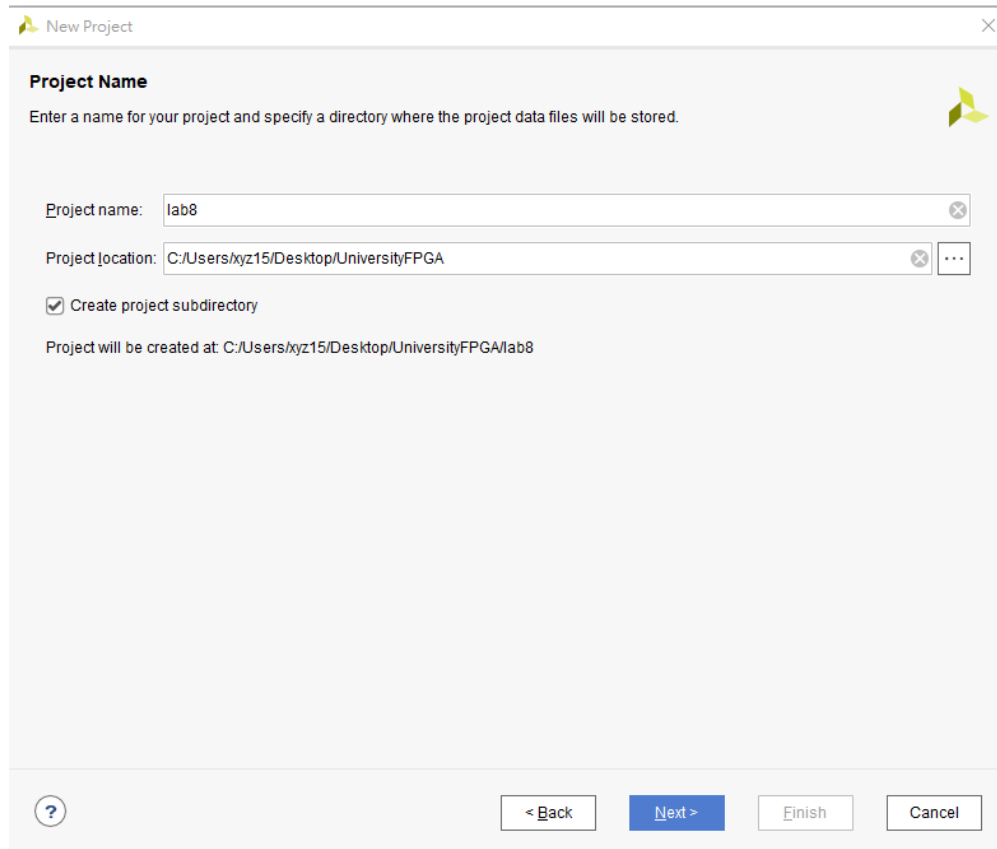
# 實驗步驟

- ❖ 打開Vivado工具，並建立一個全新的專案，點擊create new project。



# 實驗步驟

## ❖ 設定專案名稱。



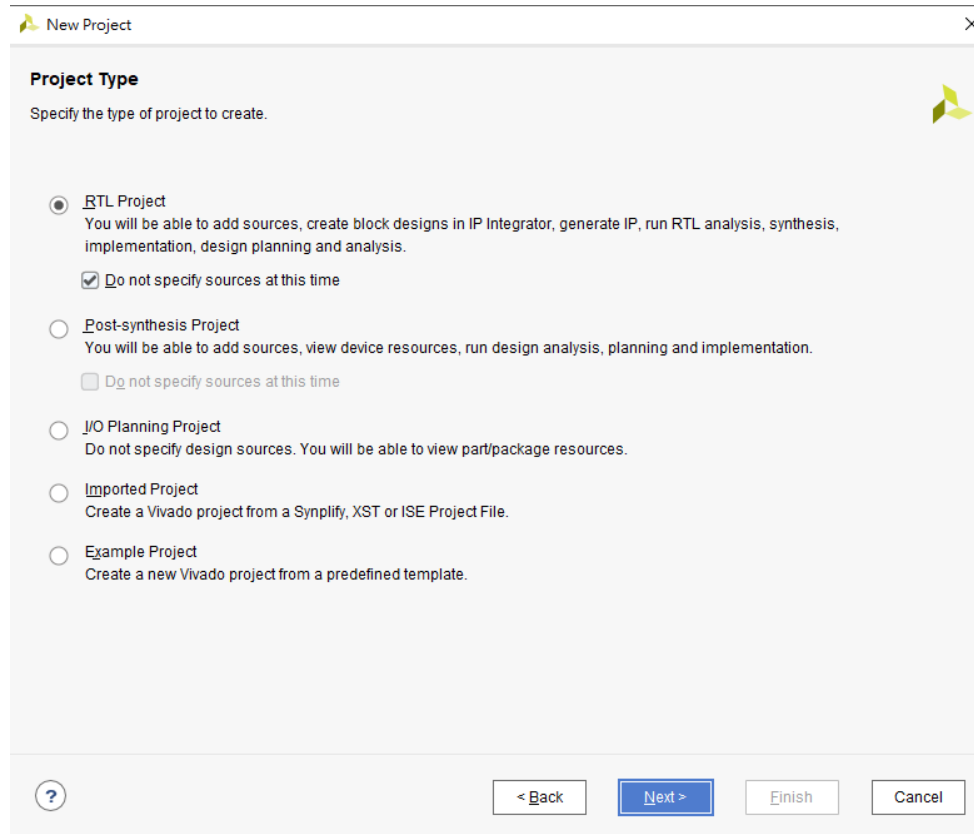
The image shows a 'New Project' dialog box with the following fields and options:

- Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.
- Project name:** lab8
- Project location:** C:/Users/xyz15/Desktop/UniversityFPGA
- ☒ Create project subdirectory
- Project will be created at: C:/Users/xyz15/Desktop/UniversityFPGA/lab8
- Navigation buttons: < Back, Next >, Finish, Cancel



# 實驗步驟

## ❖ RTL Project (記得打勾)。



New Project

**Project Type**  
Specify the type of project to create.

☒ **RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
☒ Do not specify sources at this time

☐ **Post-synthesis Project**  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time

☐ **I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.

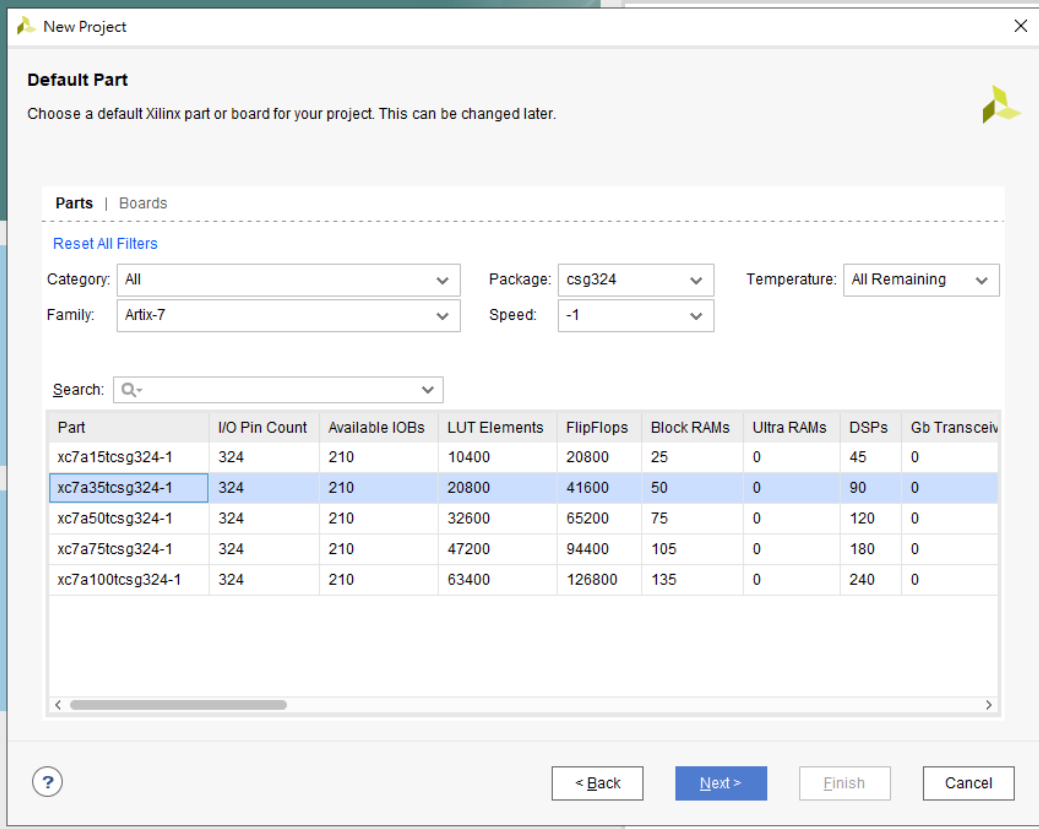
☐ **Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**  
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

# 實驗步驟

- ❖ 板子選xc7a35tcsg324-1。



**New Project**

**Default Part**  
Choose a default Xilinx part or board for your project. This can be changed later.

Parts | Boards

[Reset All Filters](#)

Category: All Package: csg324 Temperature: All Remaining

Family: Artix-7 Speed: -1

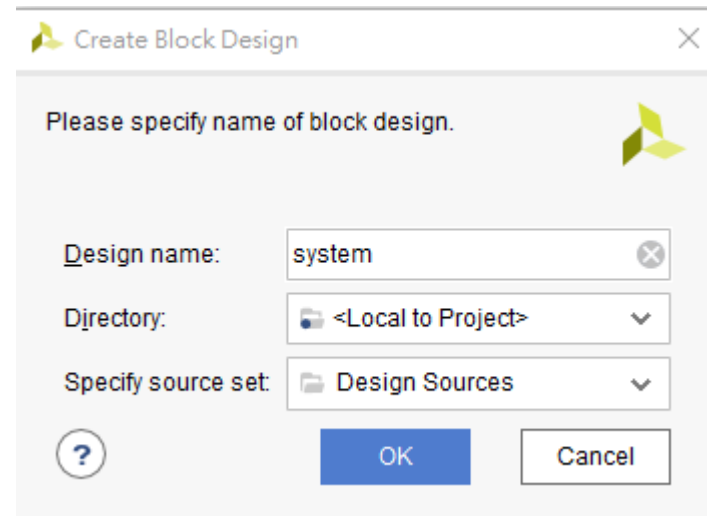
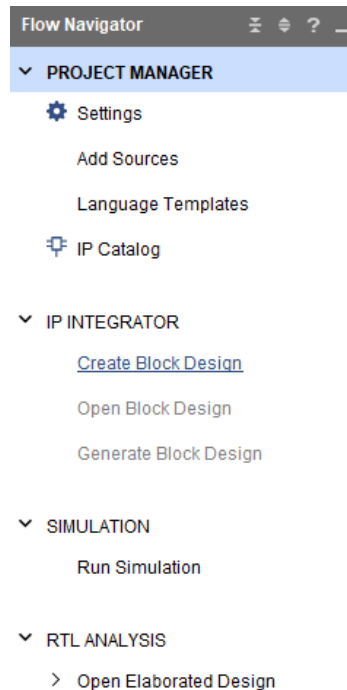
Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceiv
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

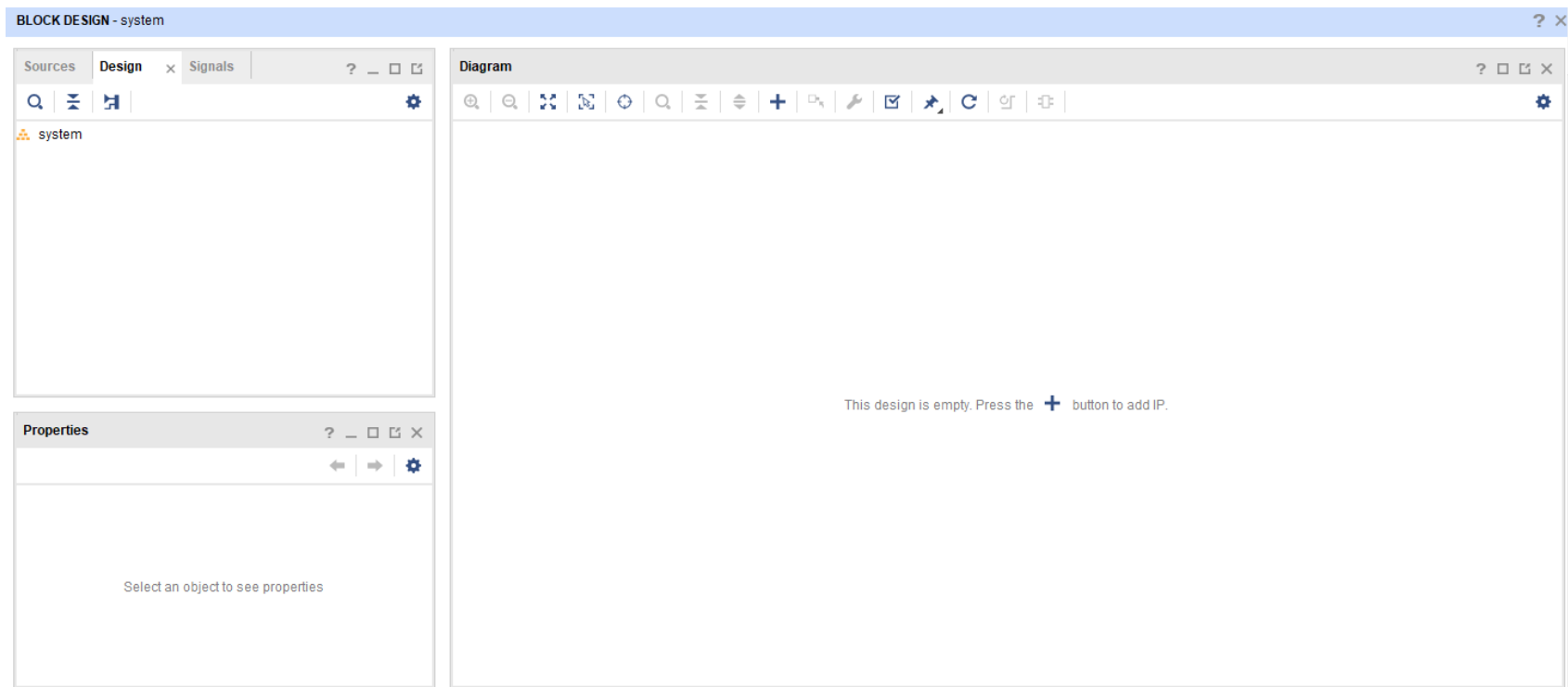
# 實驗步驟

- ❖ 點擊Create Block Design，建立一個全新的模組化設計。
- ❖ 輸入一個模組化設計的設計名稱，並點擊OK。



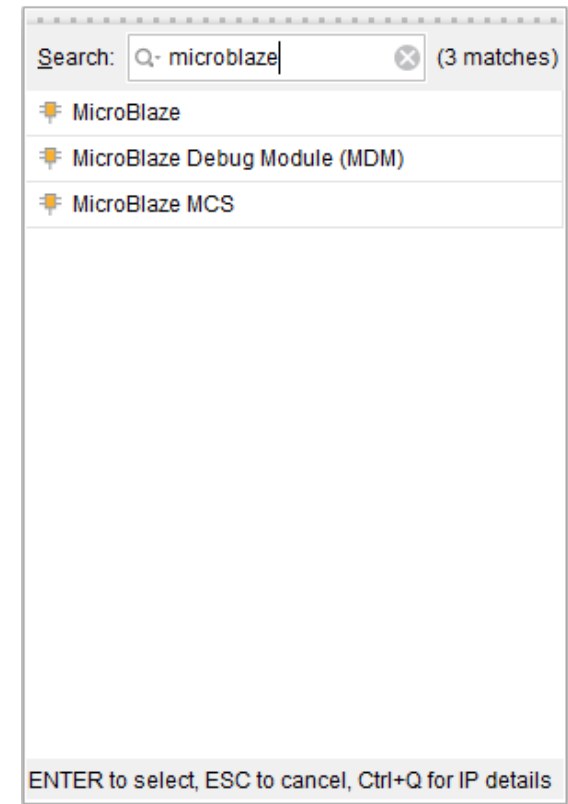
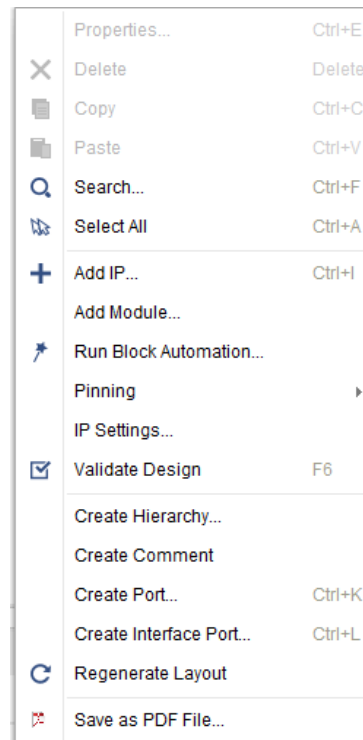
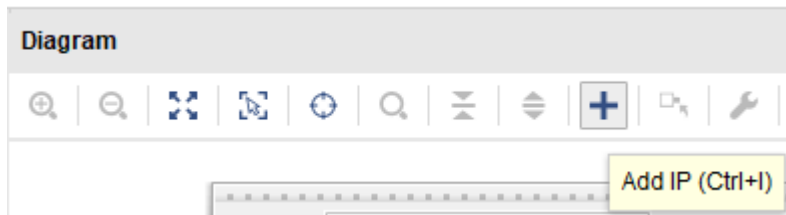
# 實驗步驟

- ❖ 此時一個空白模組化設計頁面被建立出來，並在設計原始檔案中有顯示。



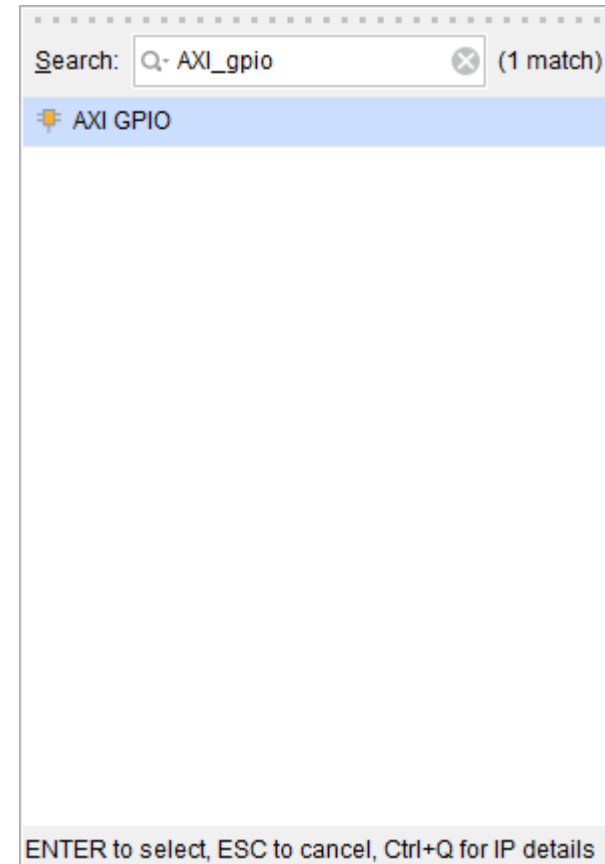
# 實驗步驟

- ❖ 點擊Add IP，或在模組化設計頁面空白處點擊右鍵，選擇Add IP。
- ❖ 搜索MicroBlaze並按兩下添加。



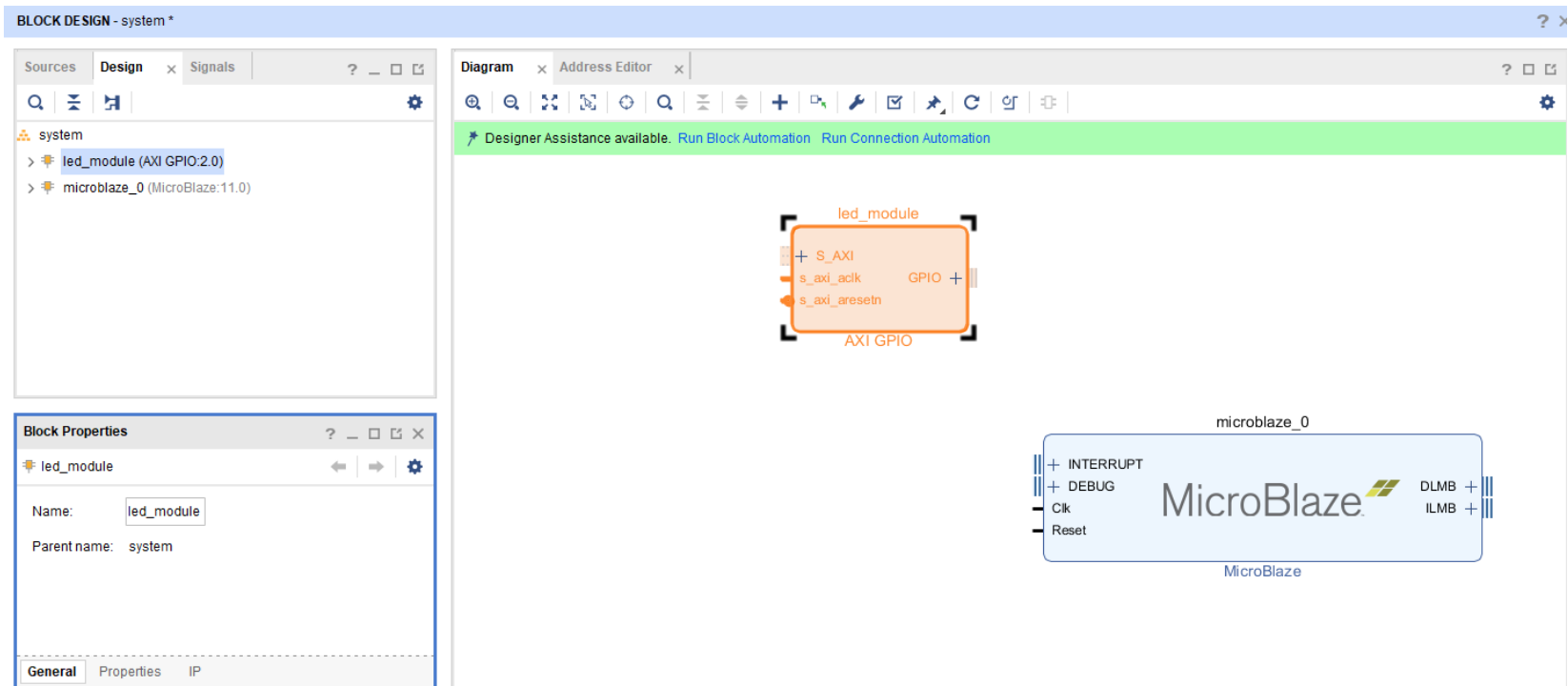
# 實驗步驟

- ❖ 同上方法添加AXI\_gpio IP。



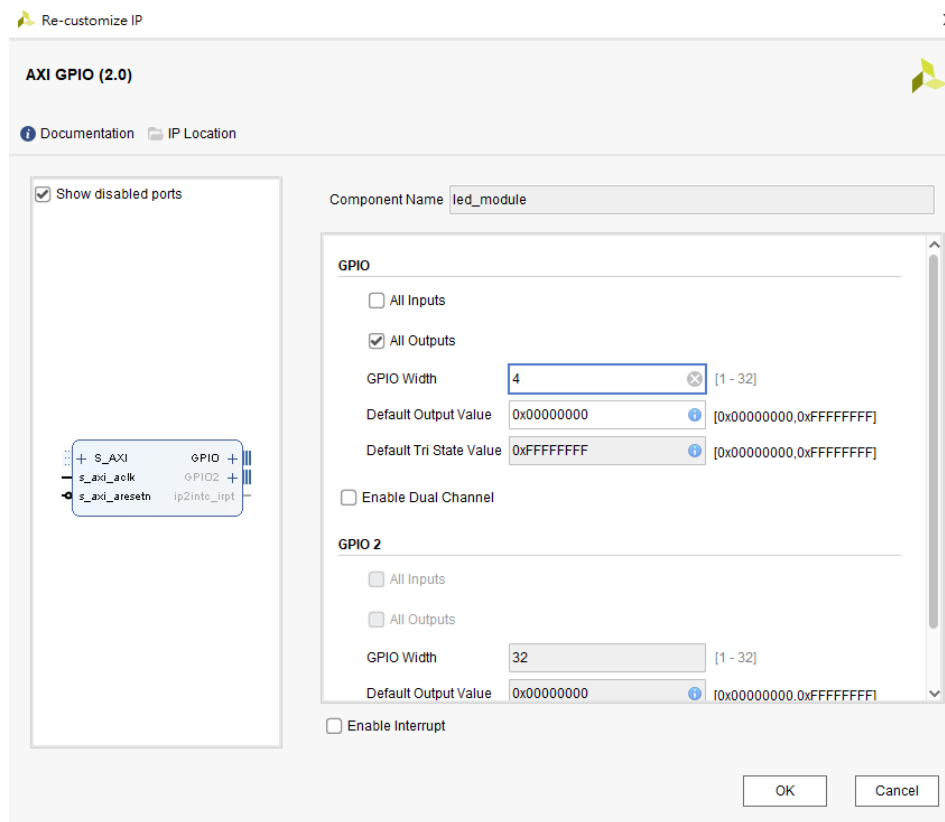
# 實驗步驟

- ❖ 點擊axi\_gpio\_0，在Block Properties視窗可以更改IP的名稱，將名稱改為led\_module。



# 實驗步驟

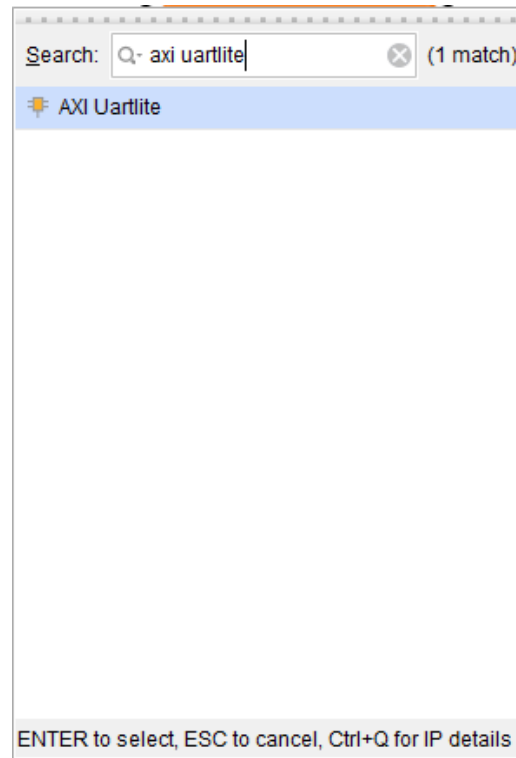
- ❖ 按兩下 AXI\_GPIO 模組，因為作為 LED 輸出，所以勾選 all outputs。





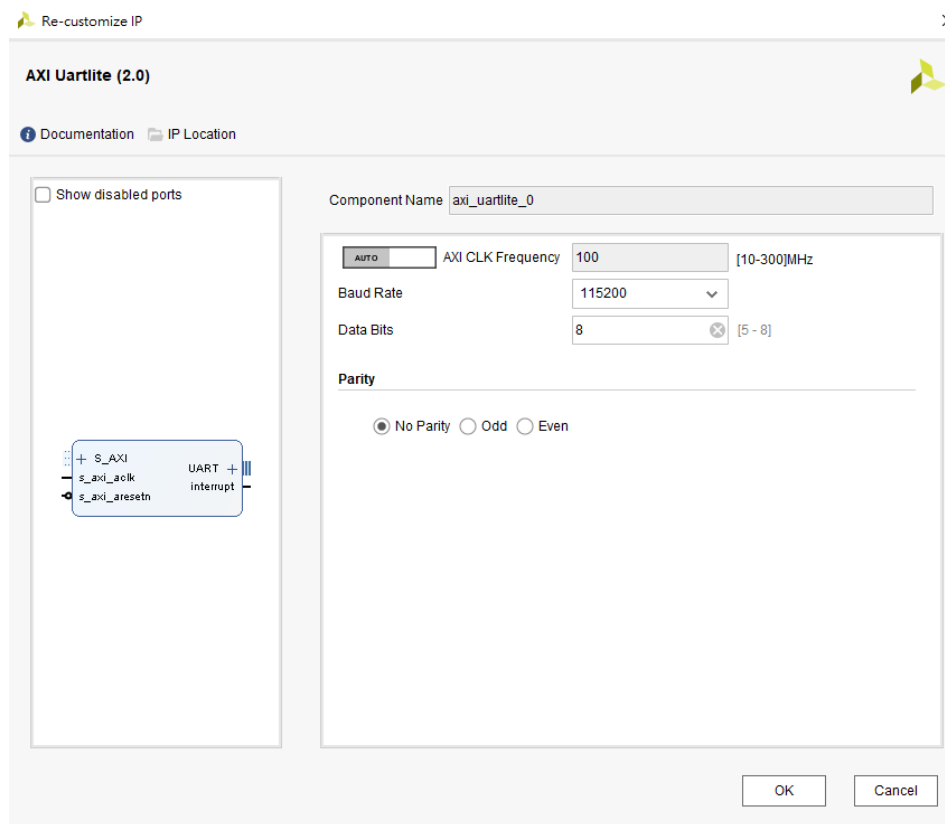
# 實驗步驟

- ❖ 選擇Add IP，添加AXI Uartlite IP core。



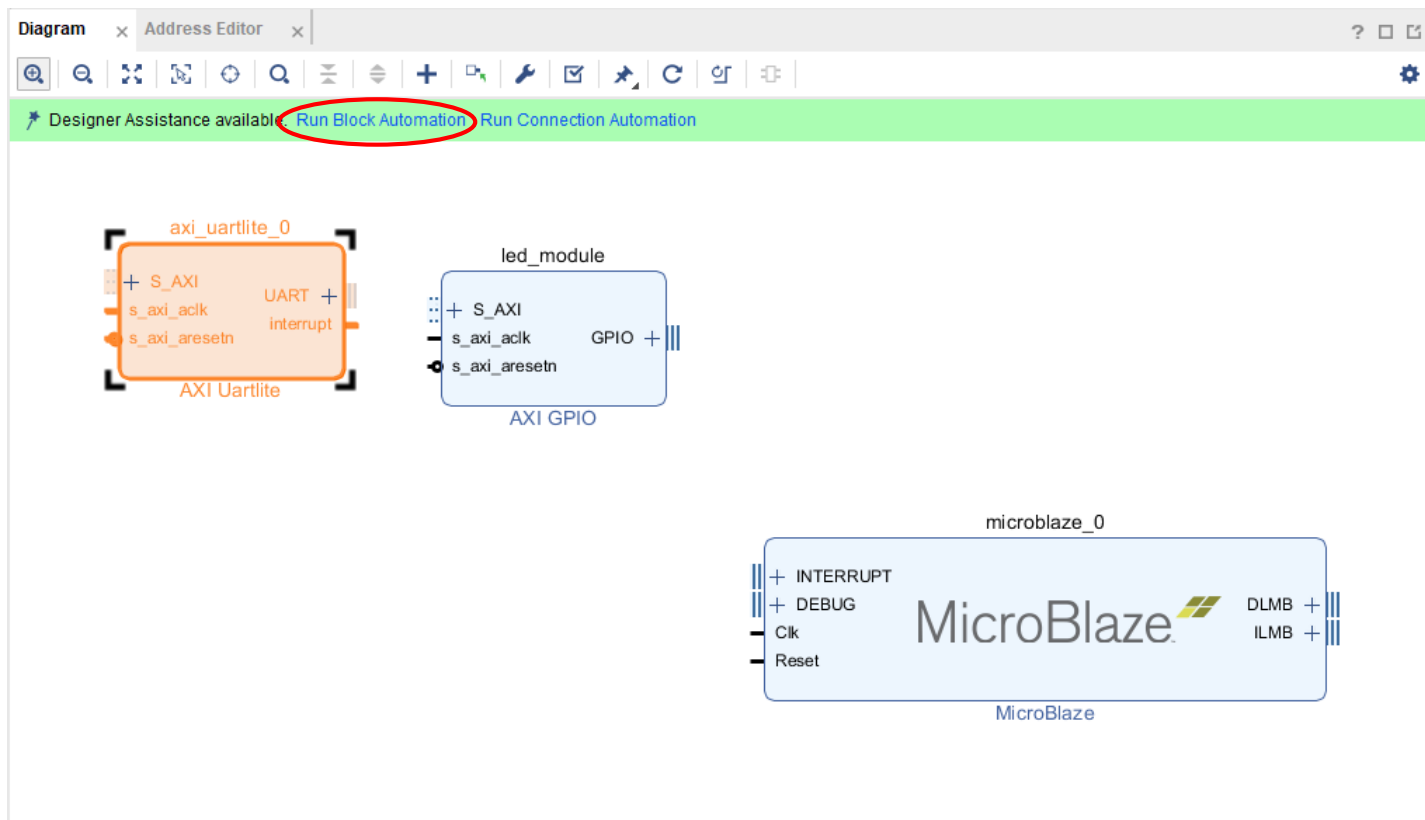
# 實驗步驟

- ❖ 按兩下axi\_uartlite\_0，修改串列埠傳輸速率為115200。



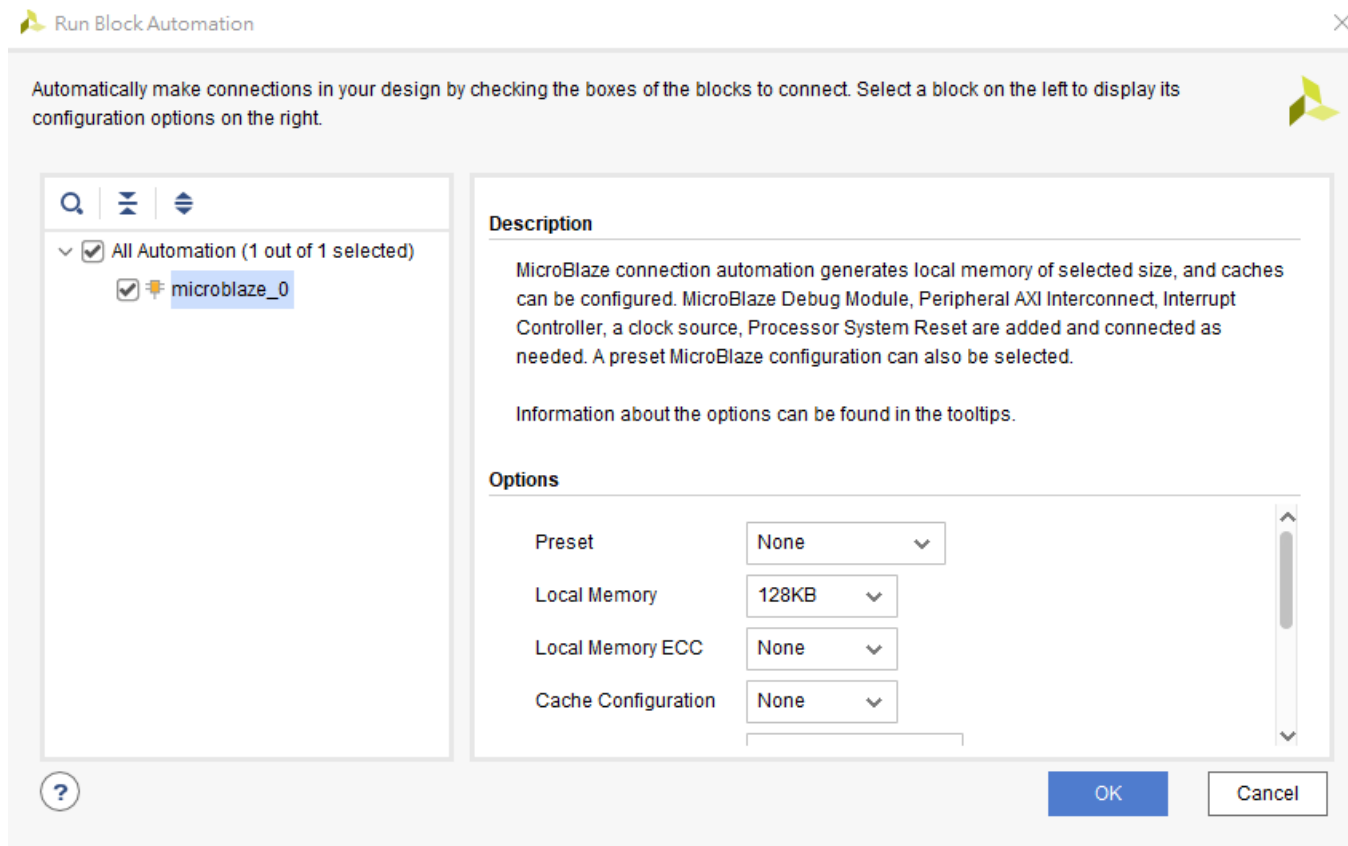
# 實驗步驟

- ❖ 點擊Run Block Automation，軟體自動生成基於本系統需求的IP，例如時脈模組、重置模組、匯流排介面等，點擊OK。



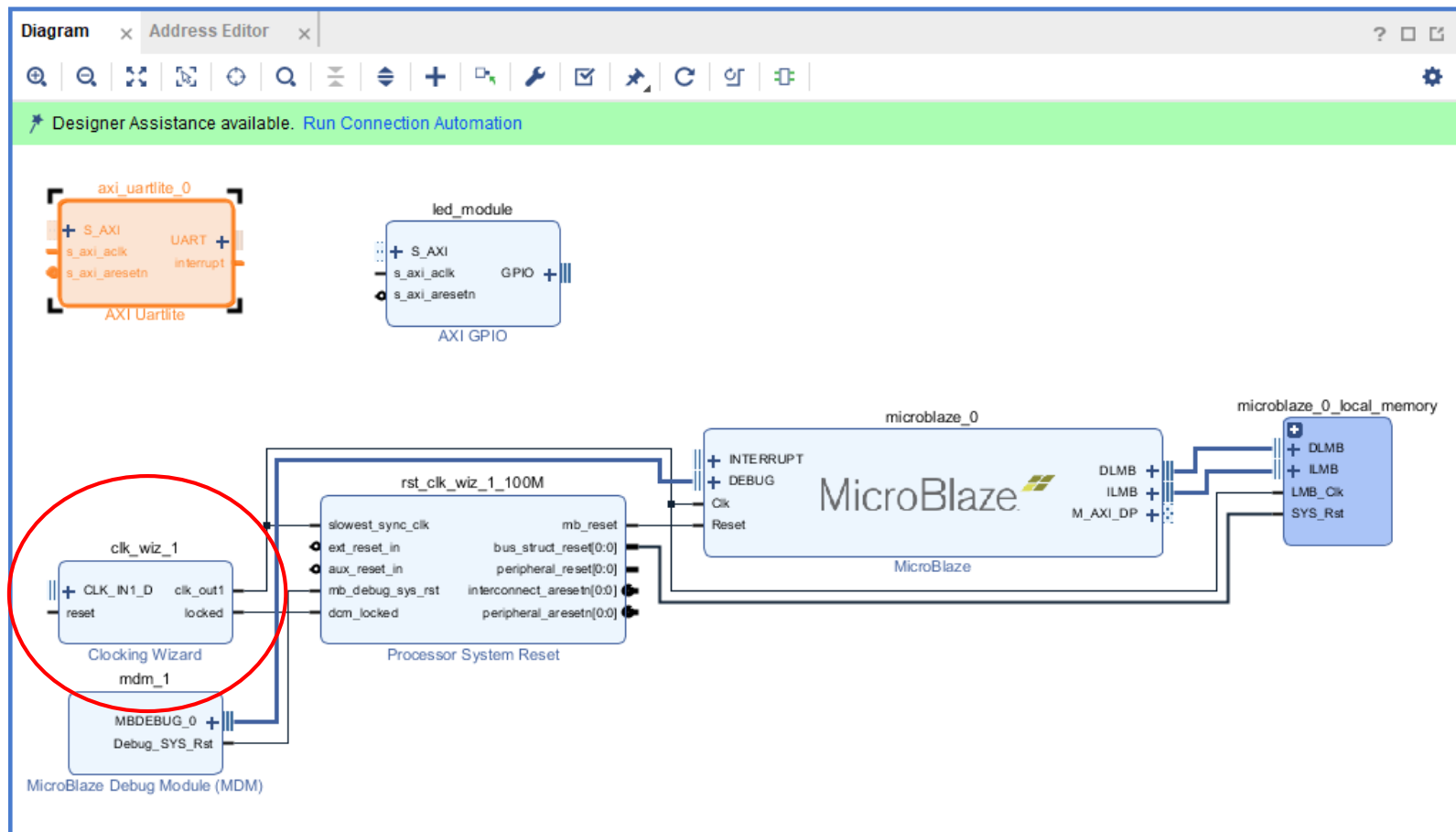
# 實驗步驟

- ❖ Local Memory改128KB設定，點擊OK。



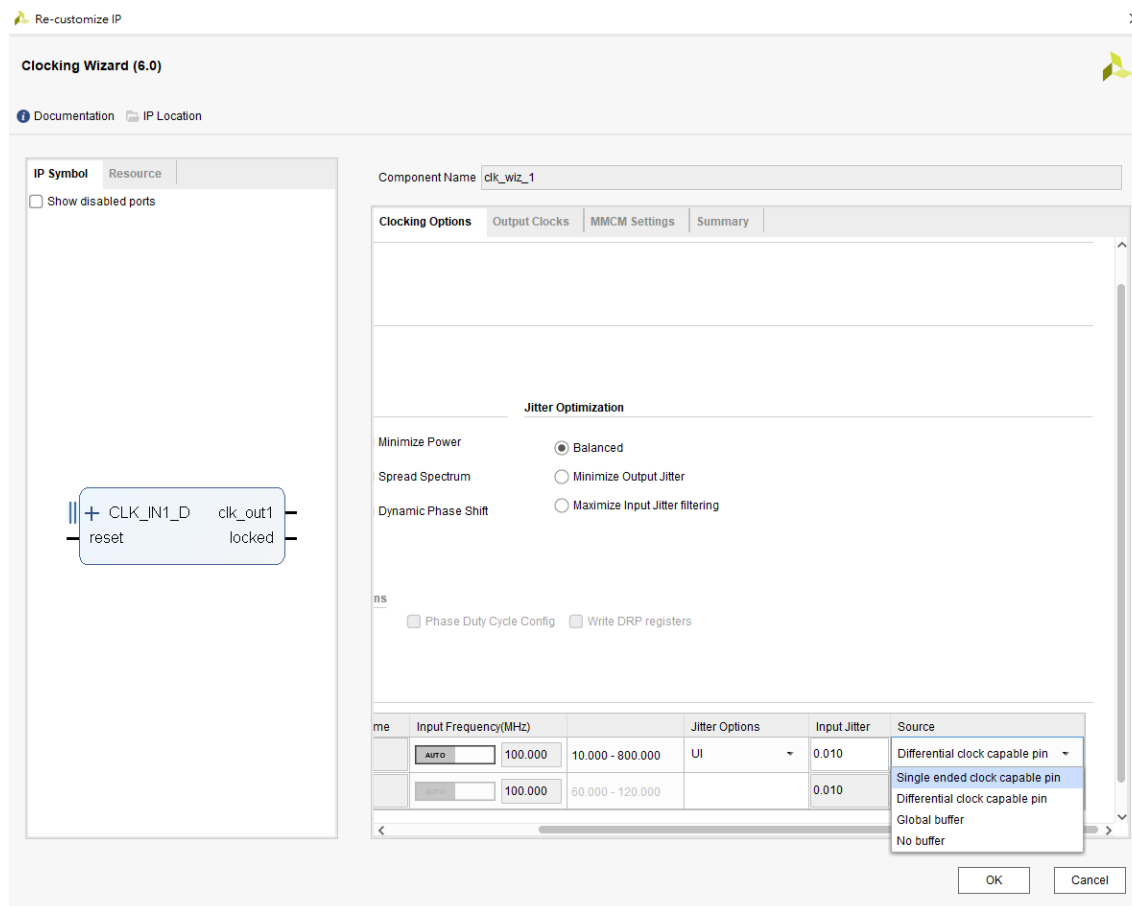
# 實驗步驟

- ❖ 選擇時脈管理模組clk\_wiz\_1，並按兩下。



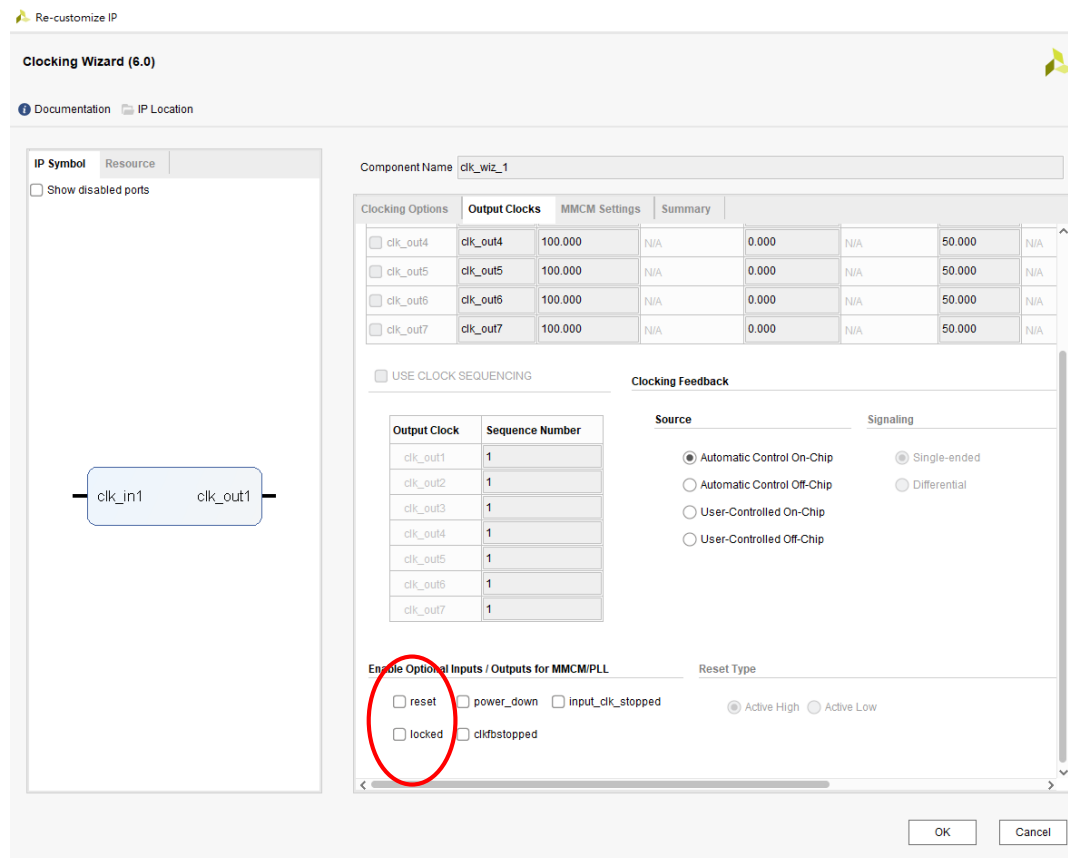
# 實驗步驟

- ❖ 把下方輸入時脈原始檔案改為single ended clock capable。



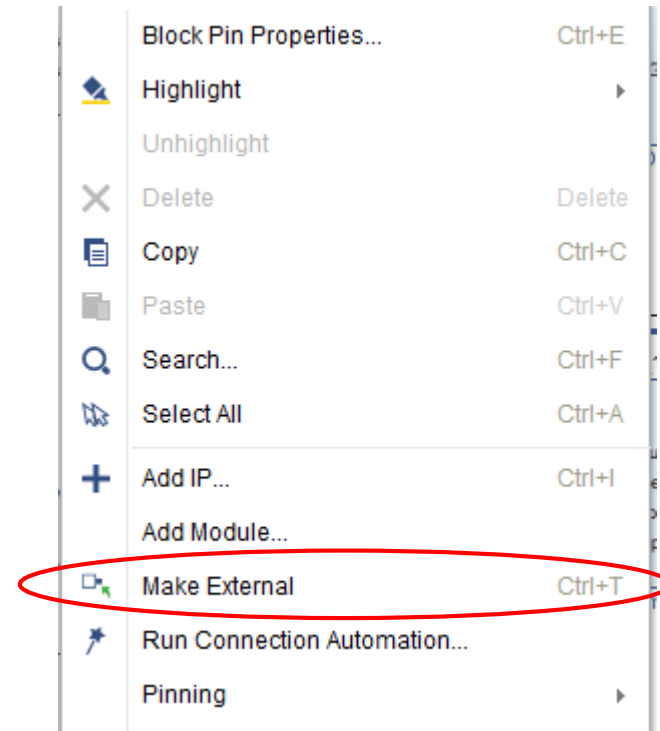
# 實驗步驟

- ❖ 輸出時脈維持100，並把視窗拉到最下面，把reset、locked的欄位去掉，點擊OK。



# 實驗步驟

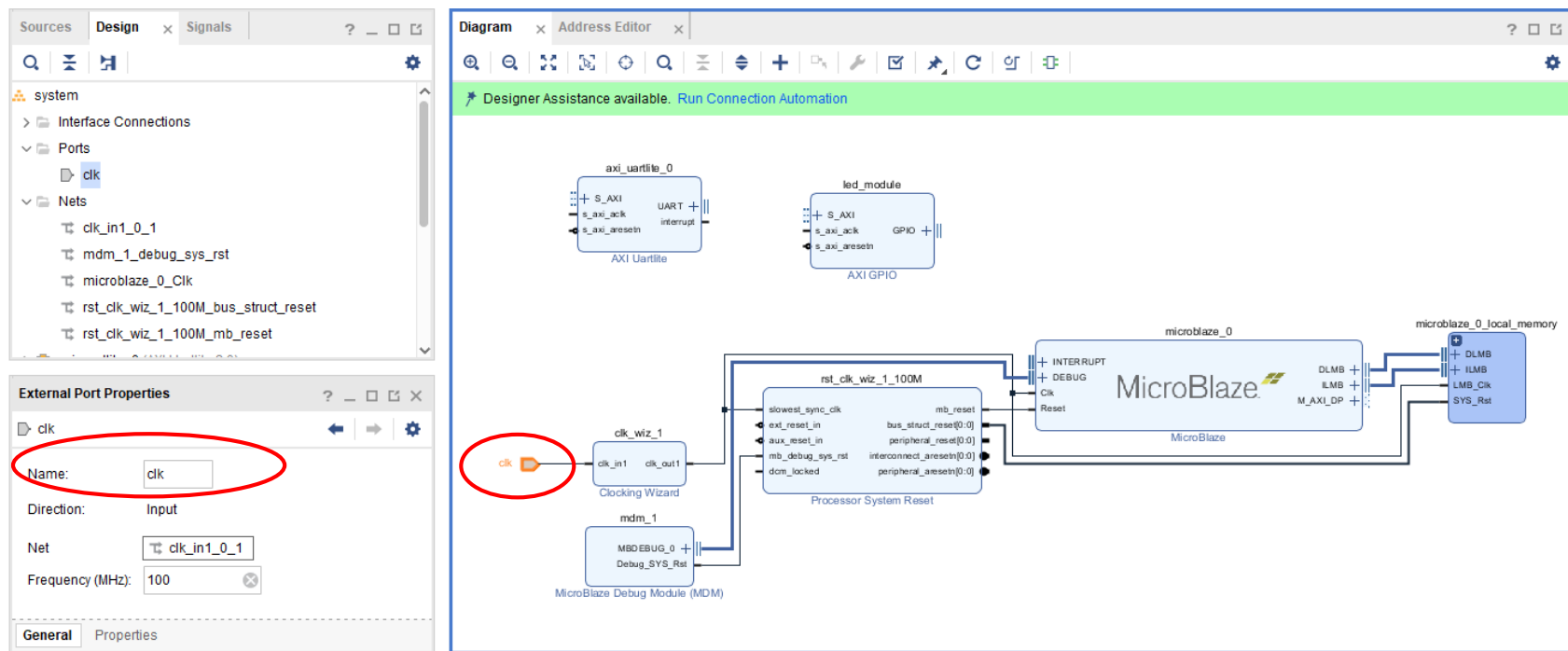
- ❖ 選擇clk\_in1腳位讓其變成黃色，對腳位點擊右鍵，選擇make external。





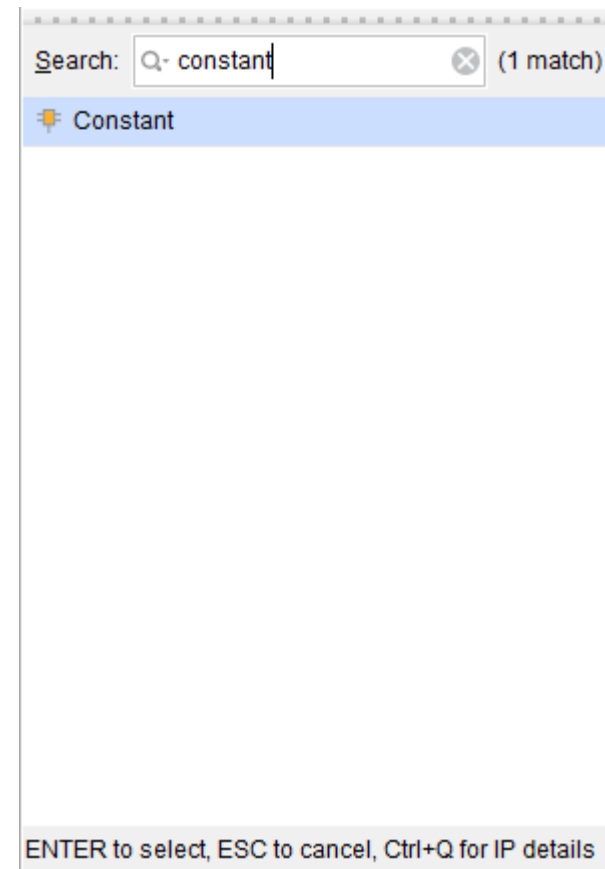
# 實驗步驟

❖ 點選腳位並改成clk。



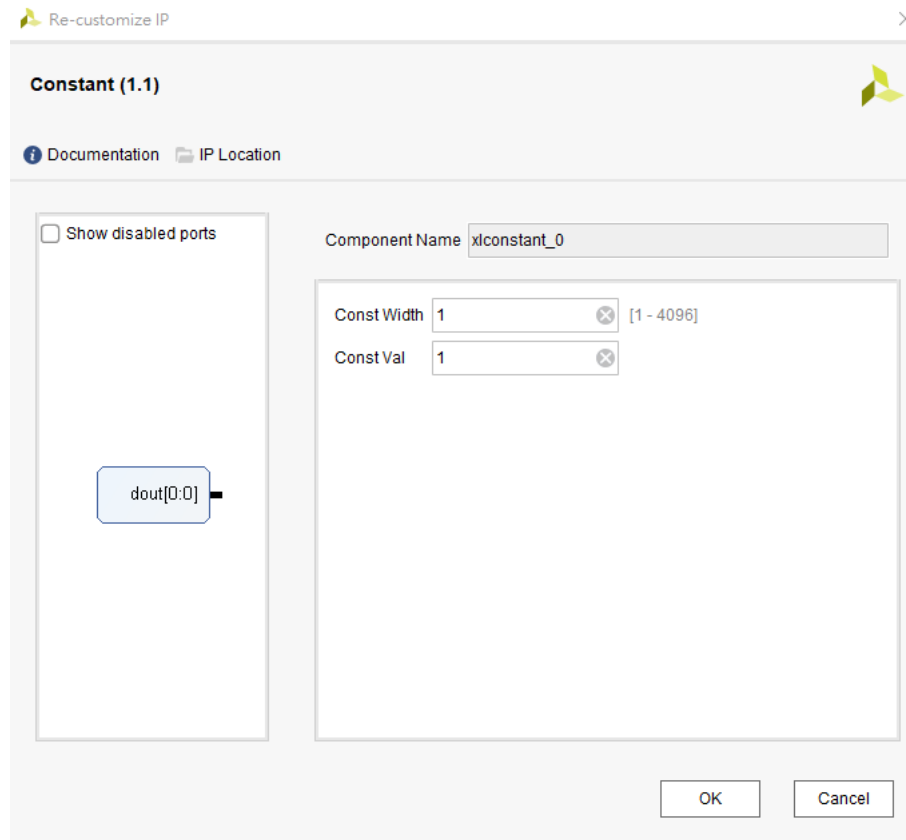
# 實驗步驟

- ❖ 由於Microblze統一用低電位重置。在此不準備重置，因此給一個常數值1，讓其一直保持。
- ❖ 加入一個常數IP。



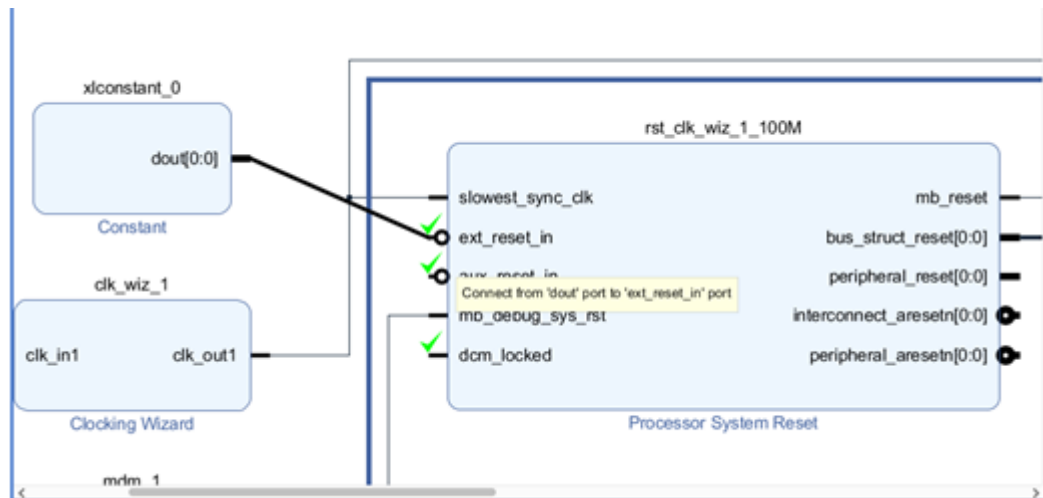
# 實驗步驟

- ❖ 按兩下常數IP，把輸出設置為1，常量值為1，保持不重置模式，MB復位為低電位。



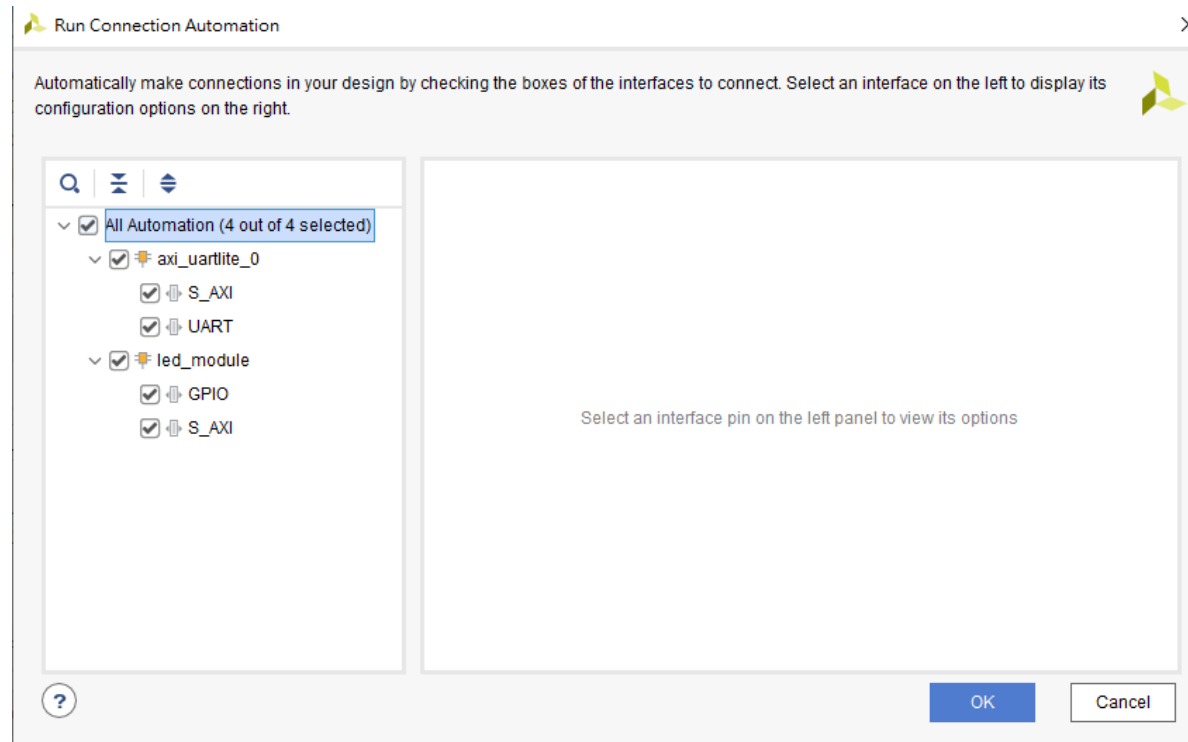
# 實驗步驟

- ❖ 把輸出與ext\_reset\_in進行相連，點擊dout的埠，按住左鍵，出現鉛筆標誌，進行連線，連到ext\_reset\_in介面。



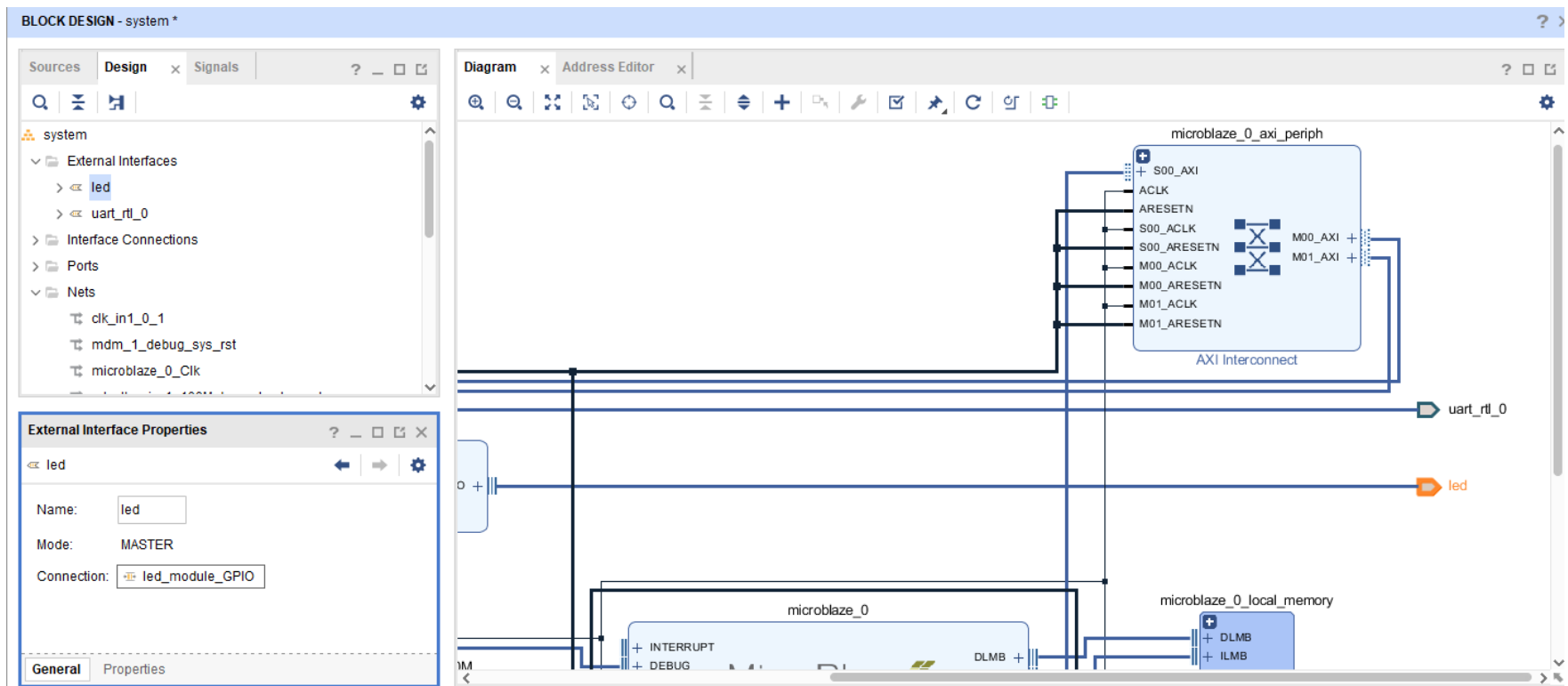
# 實驗步驟

- ❖ 點擊上方run connection automation，並勾選all automation，點擊OK。軟體自動把gpio led\_module和uartlite與MicroBlaze core進行相連。



# 實驗步驟

- ❖ 選擇新生成的gpio\_rtl\_0埠，把名稱改為led。



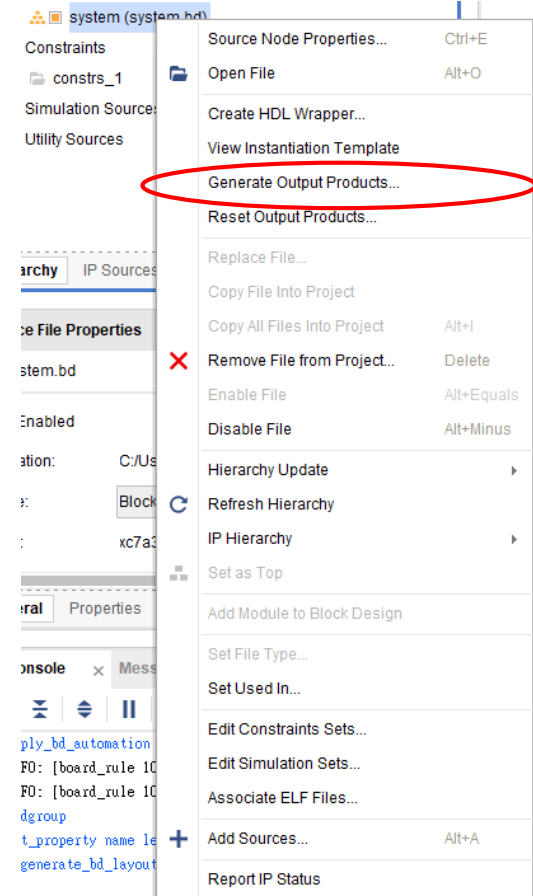
# 實驗步驟

- ❖ 點擊Regenerate Layout按鍵，讓軟體自動對模組圖進行排序，讓圖更好看。



# 實驗步驟

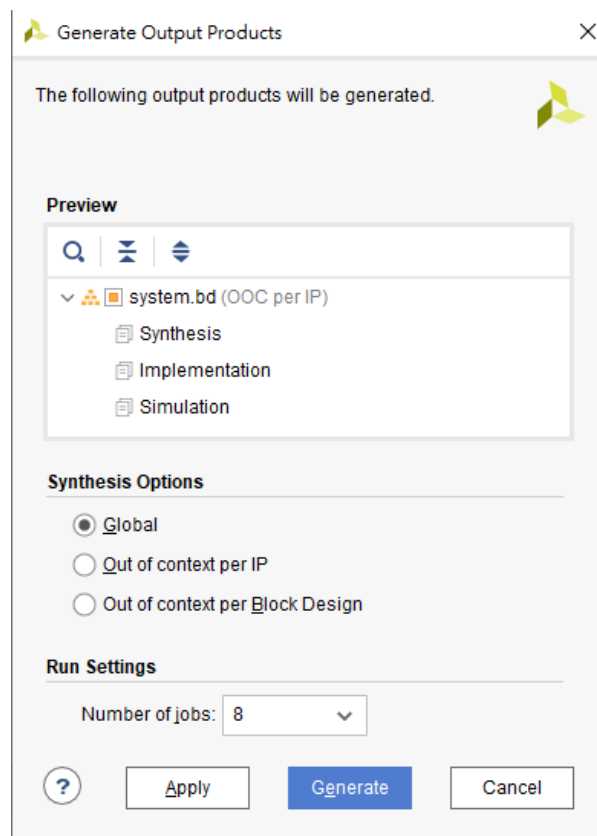
- ❖ 選擇Source視窗的system(你的模組化設計名稱(投影片第12頁))，點擊右鍵，選擇Generate Output Products，生成相關IP的網絡、設計、模擬檔。





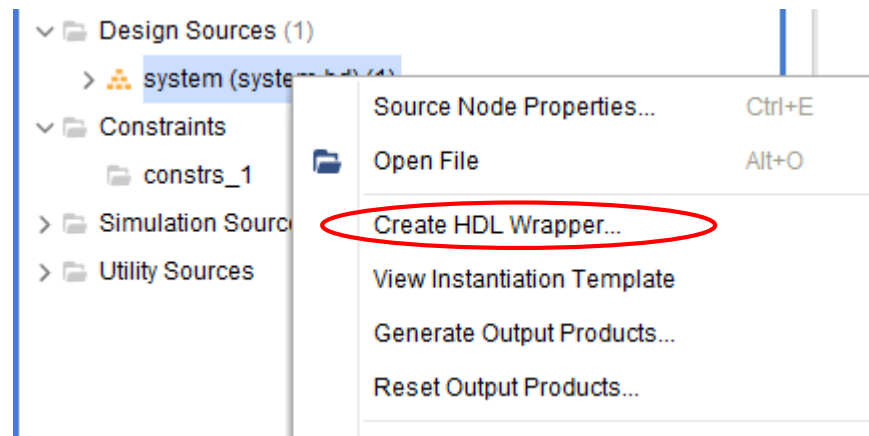
# 實驗步驟

- ❖ Synthesis Options選Global，按下Generate。



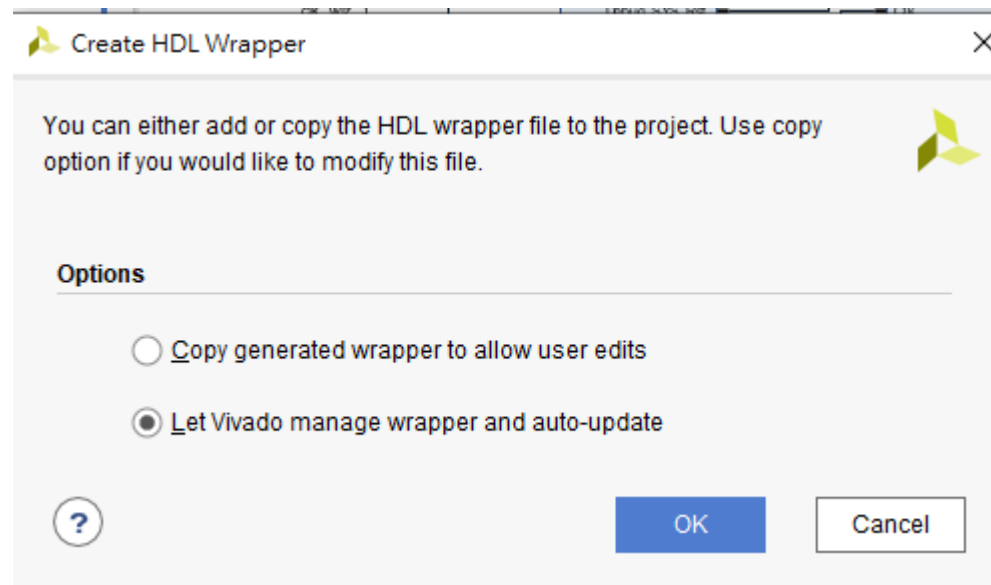
# 實驗步驟

- ❖ 再次點擊Source視窗的system，點擊右鍵，選擇Create HDL Wrapper，自動生成HDL頂層代碼。



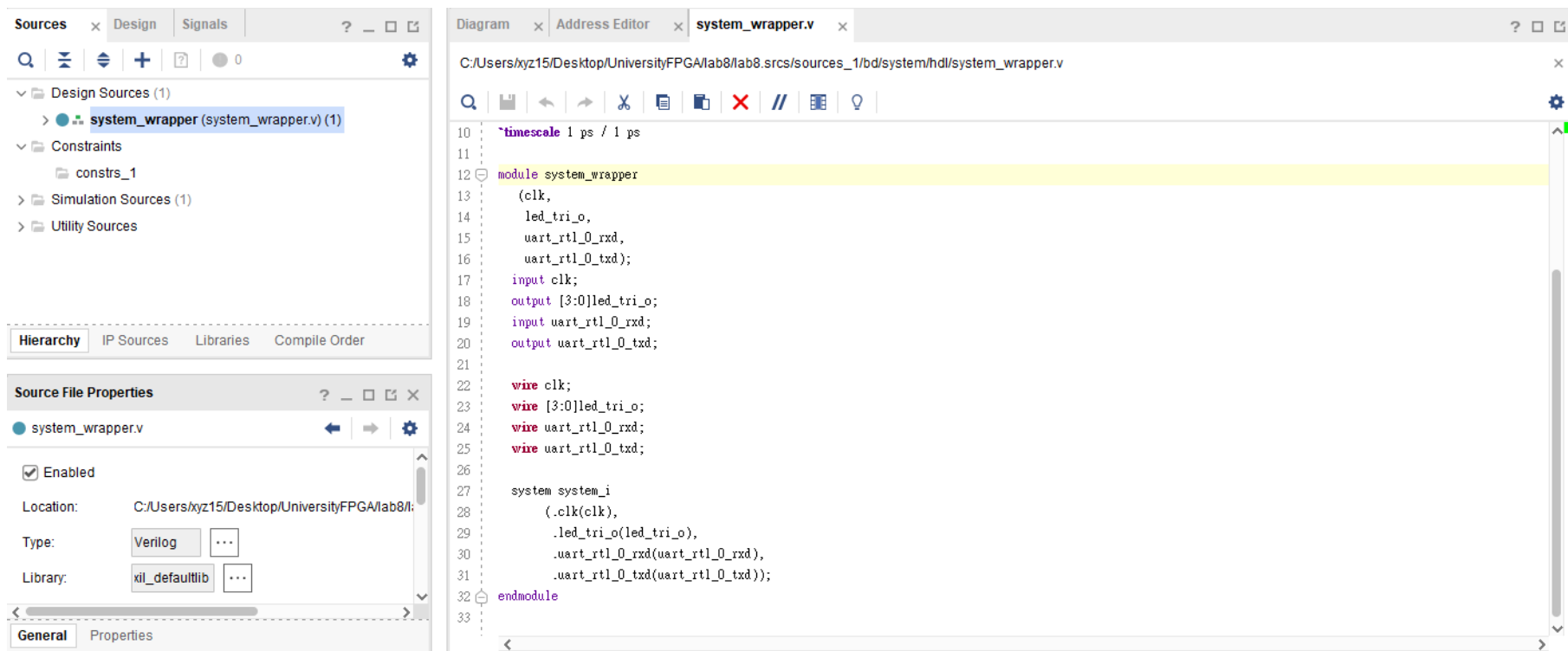
# 實驗步驟

- ❖ 選擇Let Vivado manage wrapper and auto-update，點擊OK。



# 實驗步驟

- ❖ 軟體自動產生頂層HDL代碼。

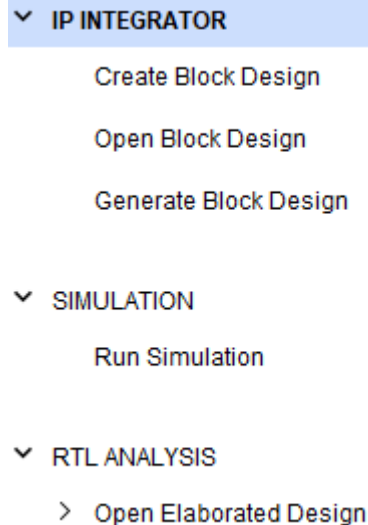


The screenshot displays the Xilinx Vivado IDE interface. On the left, the 'Sources' pane shows the project hierarchy with 'system\_wrapper (system\_wrapper.v) (1)' selected. Below it, the 'Source File Properties' pane for 'system\_wrapper.v' is visible, showing it is 'Enabled' and located at 'C:/Users/xyz15/Desktop/UniversityFPGA/lab8/srcs/sources\_1/bd/system/hdl/system\_wrapper.v'. The 'General' tab is active. The main editor window shows the Verilog code for 'system\_wrapper.v'. The code includes a timescale, module definition, port declarations, wire declarations, and an instantiation of 'system\_i'.

```
10 *timescale 1 ps / 1 ps
11
12 module system_wrapper
13     (clk,
14      led_tri_o,
15      uart_rtl_0_rxd,
16      uart_rtl_0_txd);
17     input clk;
18     output [3:0]led_tri_o;
19     input uart_rtl_0_rxd;
20     output uart_rtl_0_txd;
21
22     wire clk;
23     wire [3:0]led_tri_o;
24     wire uart_rtl_0_rxd;
25     wire uart_rtl_0_txd;
26
27     system system_i
28     (
29         .clk(clk),
30         .led_tri_o(led_tri_o),
31         .uart_rtl_0_rxd(uart_rtl_0_rxd),
32         .uart_rtl_0_txd(uart_rtl_0_txd));
33 endmodule
```

# 實驗步驟

- ❖ 點擊左邊Open Elaborated Design，檢驗設計有無問題。



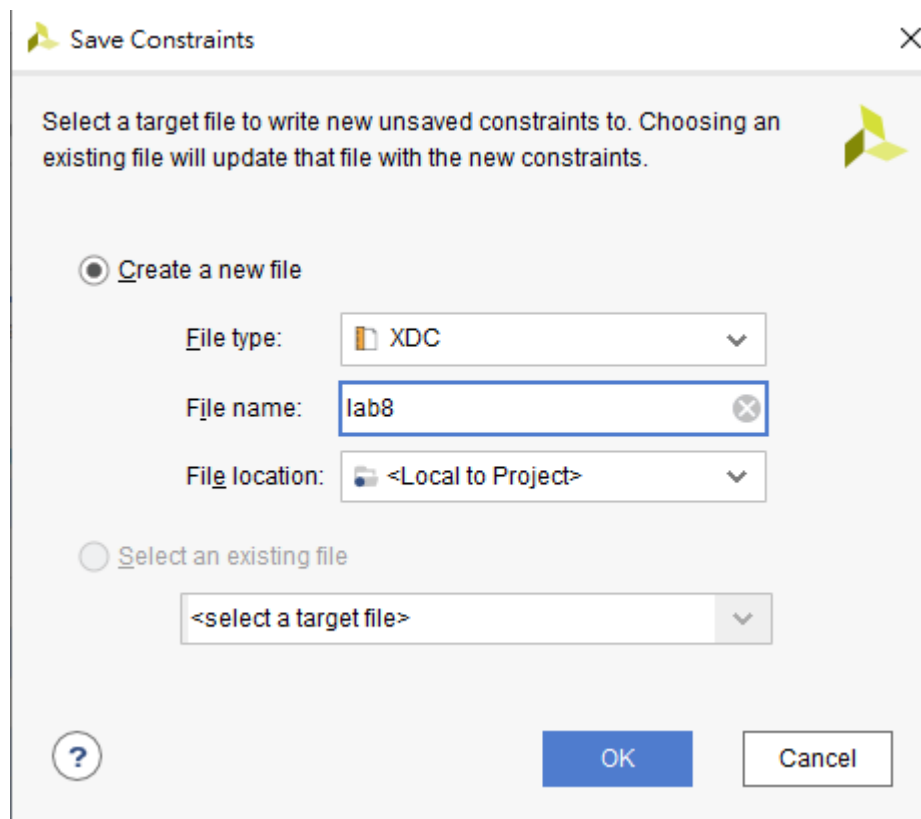
# 實驗步驟

- ❖ 把I/O Ports視窗裡面所有電壓都設成LVCMOS333。
- ❖ Pin腳跟圖一樣。

Tcl Console	Messages	Log	Reports	Design Runs	Package Pins	I/O Ports			
<div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div>									
Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco		
All ports (7)									
CLK.CLK_12642 (1)	IN			✓	14	LVCMOS33*	3.300		
Scalar ports (1)									
clk	IN		P17	✓	14	LVCMOS33*	3.300		
GPIO_12284 (4)	OUT			✓	35	LVCMOS33*	3.300		
led_tri_o (4)									
led_tri_o[3]	OUT		H4	✓	35	LVCMOS33*	3.300		
led_tri_o[2]	OUT		J3	✓	35	LVCMOS33*	3.300		
led_tri_o[1]	OUT		J2	✓	35	LVCMOS33*	3.300		
led_tri_o[0]	OUT		K2	✓	35	LVCMOS33*	3.300		
Scalar ports (0)									
uart_rtl_0_12642 (2)									
uart_rtl_0_12642 (2)	(Multiple)			✓	34	LVCMOS33*	3.300		
Scalar ports (2)									
uart_rtl_0_rxd	IN		N5	✓	34	LVCMOS33*	3.300		
uart_rtl_0_txd	OUT		T4	✓	34	LVCMOS33*	3.300		
Scalar ports (0)									

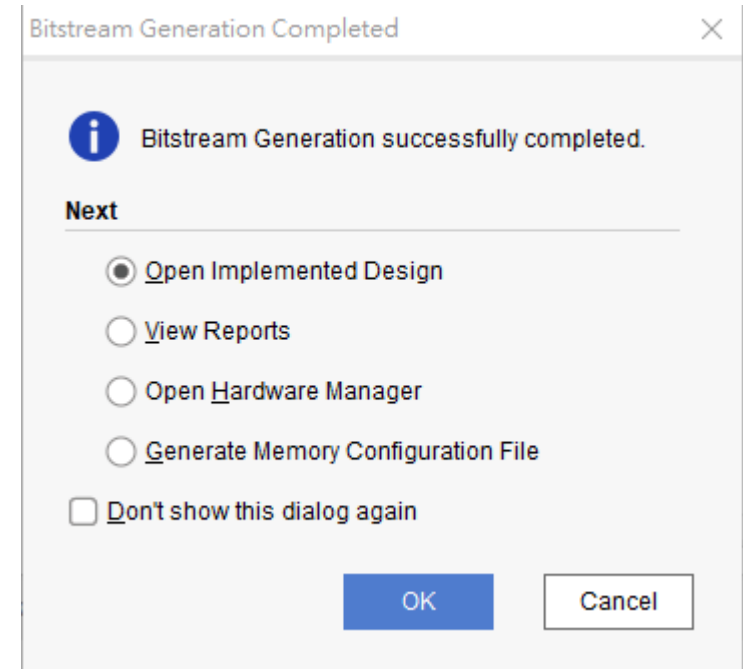
# 實驗步驟

- ❖ 按下保存，並為constraints檔命名。



# 實驗步驟

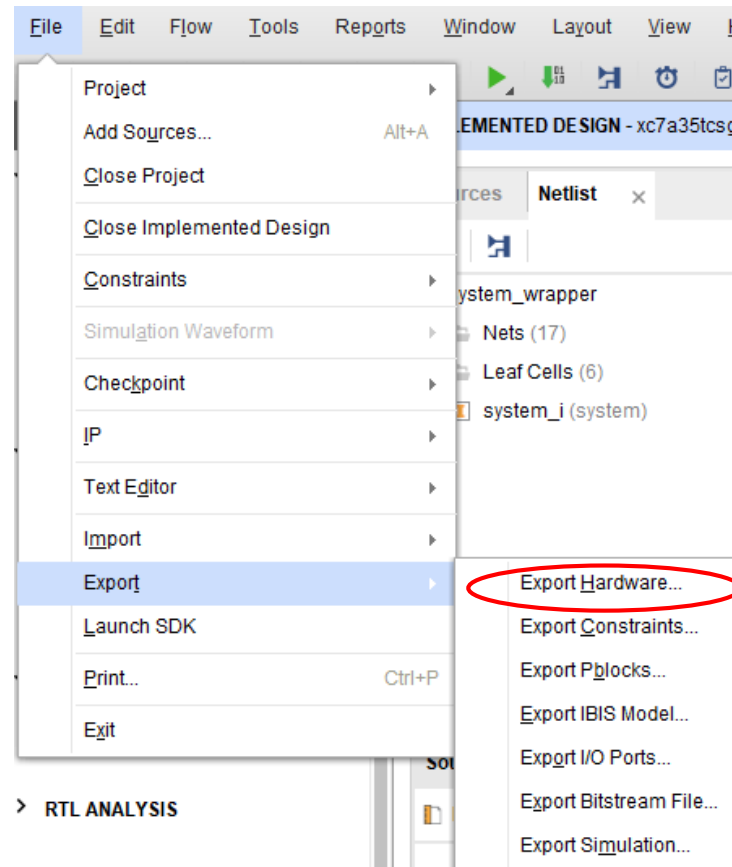
- ❖ 點擊Run Synthesis，讓軟體先對做好的專案進行合成。
- ❖ 再點擊Run Implementation運行佈局和佈線。
- ❖ 再運行Generate Bitstream。
- ❖ 最後點 open implemented Design。
- ❖ (P.S)系統會問你要不要關掉設計。
- ❖ 按NO。





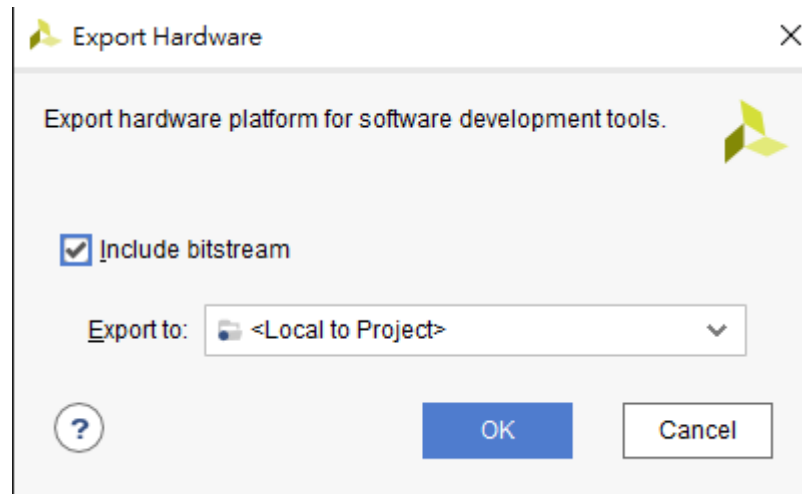
# 實驗步驟

- ❖ 點擊左上角File，選擇Export，選擇Export Hardware。



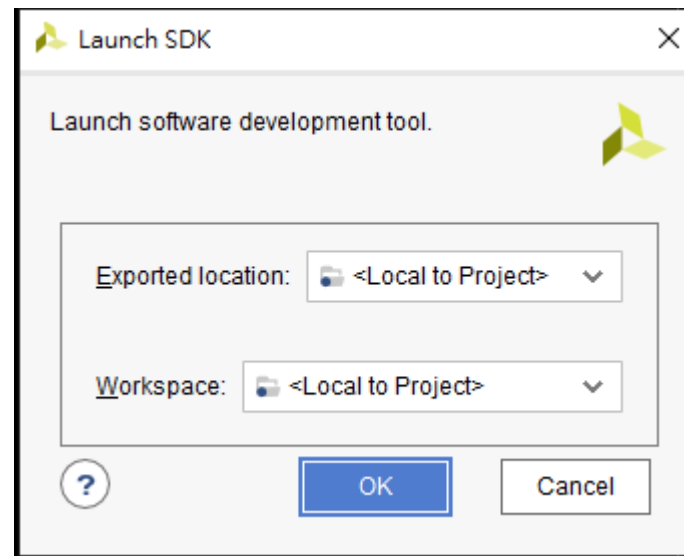
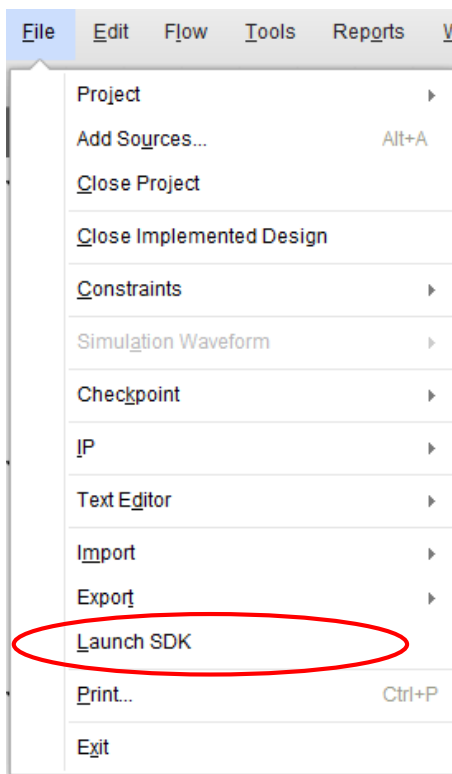
# 實驗步驟

❖ 記得打勾。



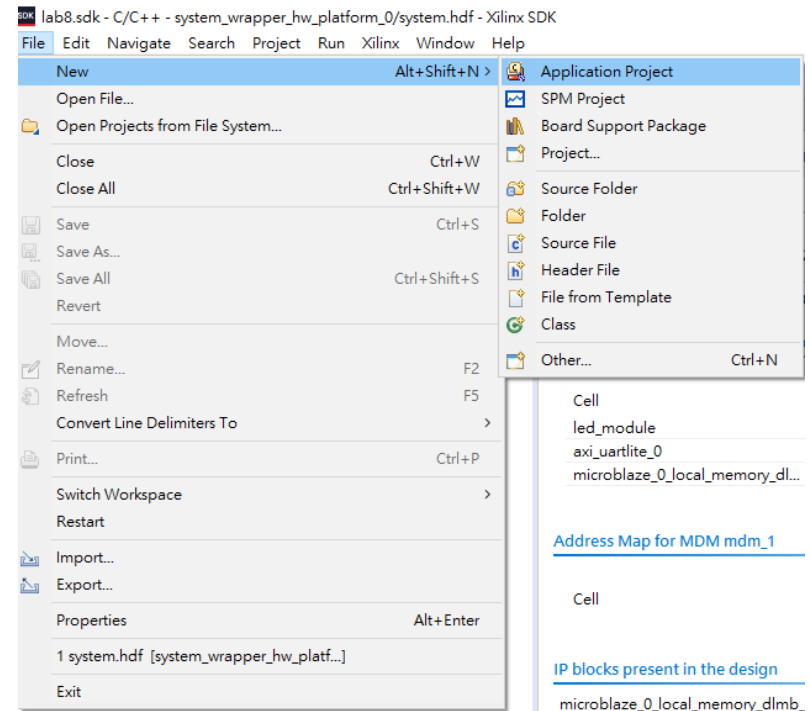
# 實驗步驟

- ❖ 點擊左上角File，選擇Launch SDK，並點擊OK。



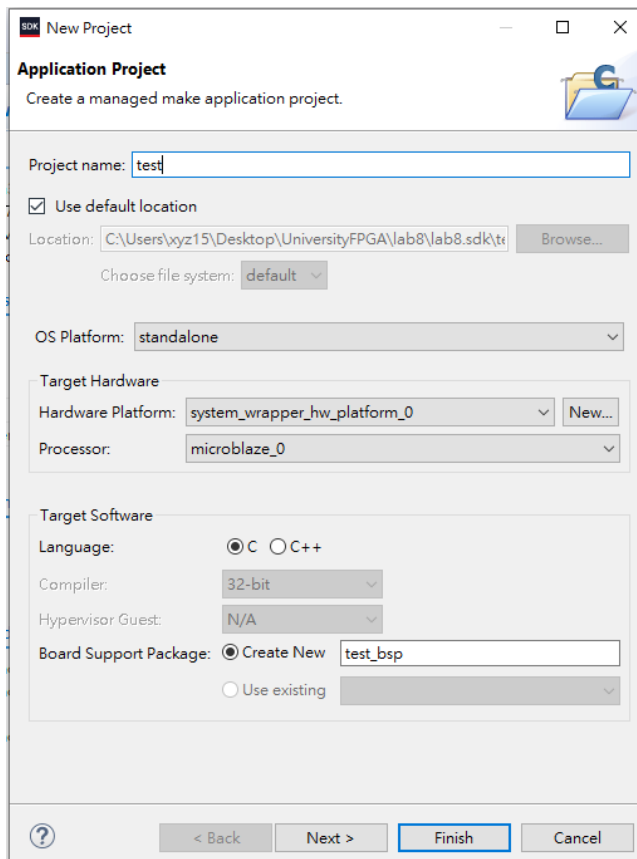
# 實驗步驟

- ❖ 記得等SDK跑完再動作。
- ❖ 接下來將針對建立好的Microblaze專案進行軟體程式設計。
- ❖ 點擊左上角File -> New -> Application Project。



# 實驗步驟

- ❖ Project name裡面輸入專案名字，點擊下一步。



**SDK New Project**

**Application Project**  
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

**Target Hardware**

Hardware Platform:

Processor:

**Target Software**

Language: ☒ C ☐ C++

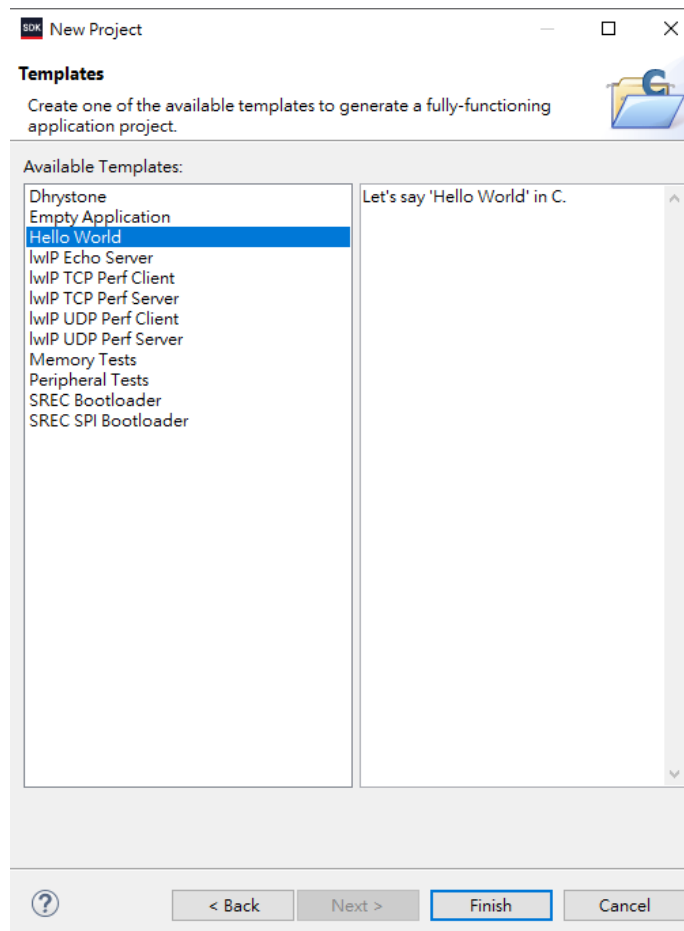
Compiler:

Hypervisor Guest:

Board Support Package: ☒ Create New  ☐ Use existing

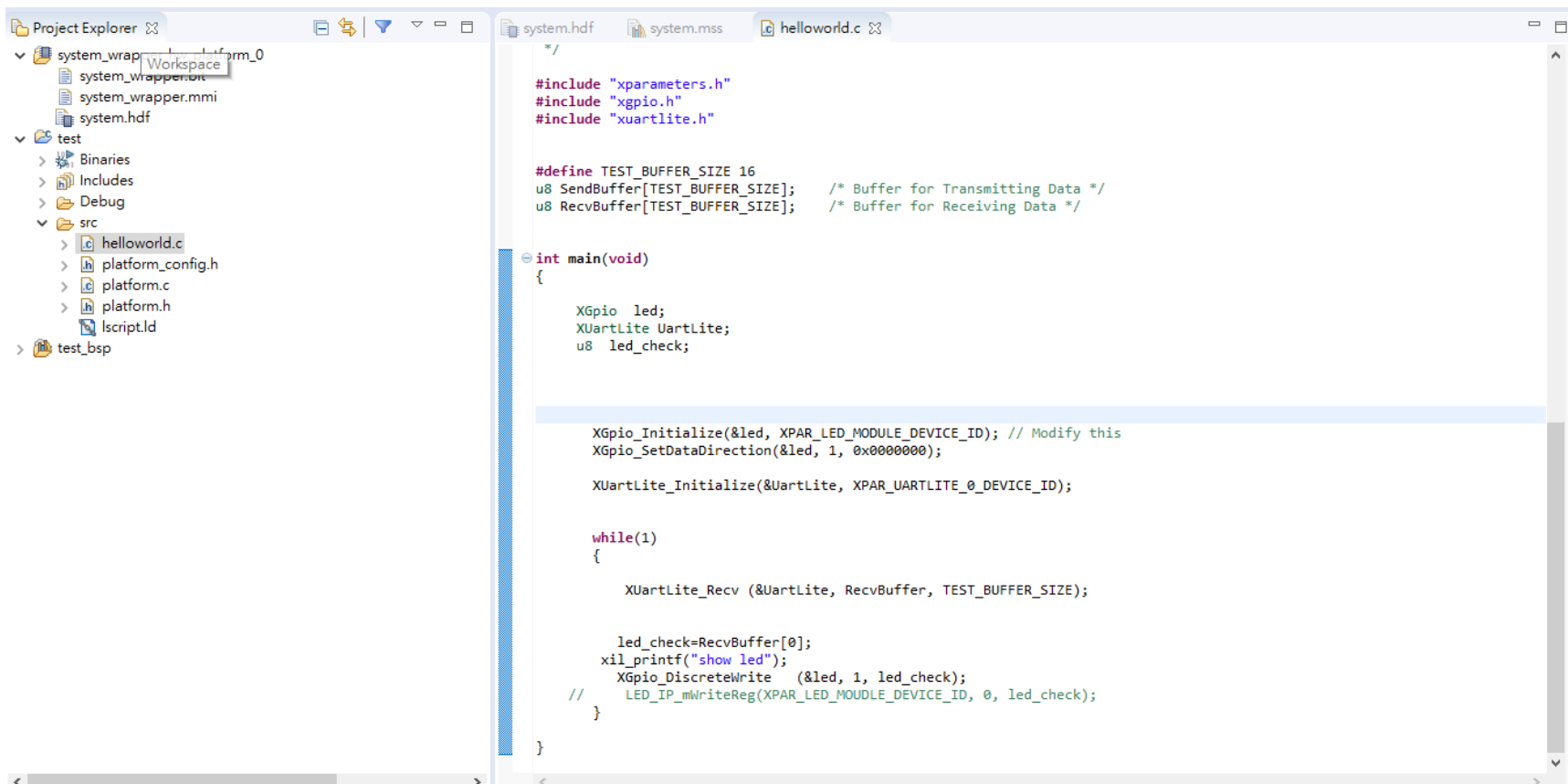
# 實驗步驟

- ❖ 本次實驗選擇Hello world進行修改。



# 實驗步驟

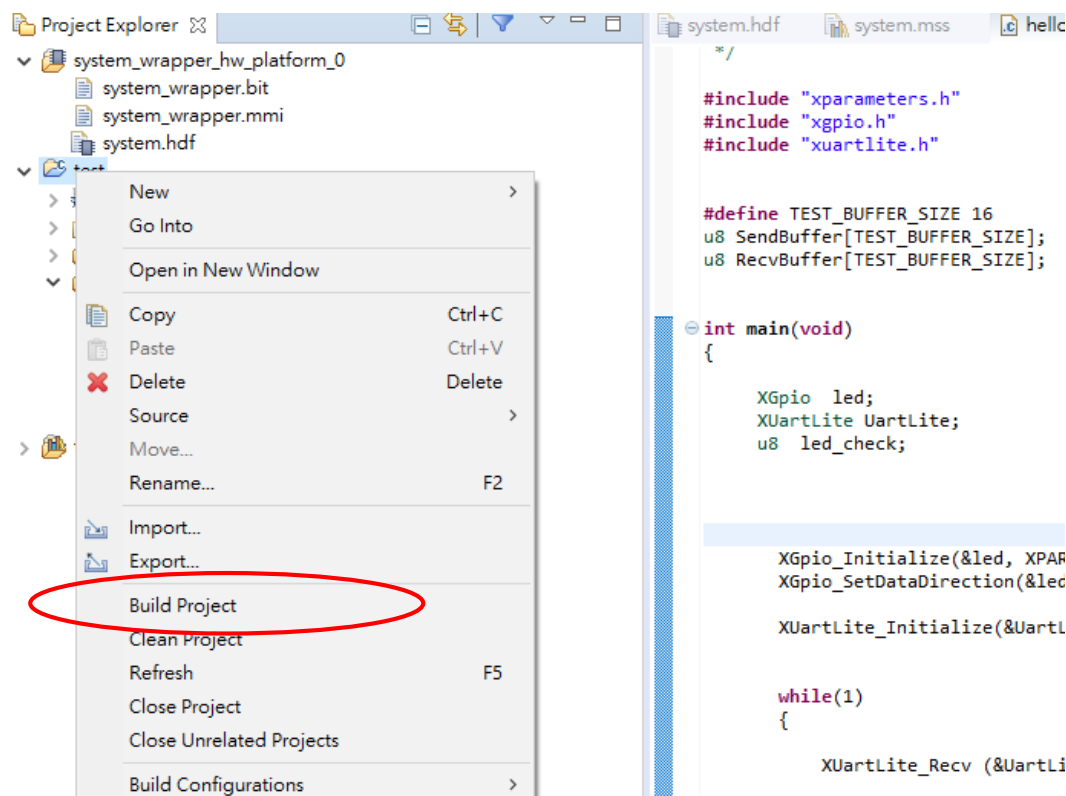
❖ 點擊helloworld.c，並修改。



```
/*  
#include "xparameters.h"  
#include "xgpio.h"  
#include "xuartlite.h"  
  
#define TEST_BUFFER_SIZE 16  
u8 SendBuffer[TEST_BUFFER_SIZE]; /* Buffer for Transmitting Data */  
u8 RecvBuffer[TEST_BUFFER_SIZE]; /* Buffer for Receiving Data */  
  
int main(void)  
{  
    XGpio led;  
    XUartLite UartLite;  
    u8 led_check;  
  
    XGpio_Initialize(&led, XPAR_LED_MODULE_DEVICE_ID); // Modify this  
    XGpio_SetDataDirection(&led, 1, 0x00000000);  
  
    XUartLite_Initialize(&UartLite, XPAR_UARTLITE_0_DEVICE_ID);  
  
    while(1)  
    {  
        XUartLite_Recv (&UartLite, RecvBuffer, TEST_BUFFER_SIZE);  
  
        led_check=RecvBuffer[0];  
        xil_printf("show led");  
        XGpio_DiscreteWrite (&led, 1, led_check);  
        // LED_IP_mWriteReg(XPAR_LED_MODULE_DEVICE_ID, 0, led_check);  
    }  
}
```

# 實驗步驟

- ❖ 點擊保存(ctrl+s)，軟體自動會對代碼進行build。或者對原始檔案點擊右鍵，再點擊build project。





# 實驗步驟

## ❖ 程式碼

```
#include "xparameters.h"
#include "xgpio.h"
#include "xuartlite.h"
#include "sleep.h"

#define TEST_BUFFER_SIZE 16
u8 SendBuffer[TEST_BUFFER_SIZE];
u8 RecvBuffer[TEST_BUFFER_SIZE];

int main()
{
    XGpio led;
    XUartLite UartLite;
    u8 led_check;

    XGpio_Initialize(&led, XPAR_LED_MODULE_DEVICE_ID); // Modify this
    XGpio_SetDataDirection(&led, 1, 0x00000000);

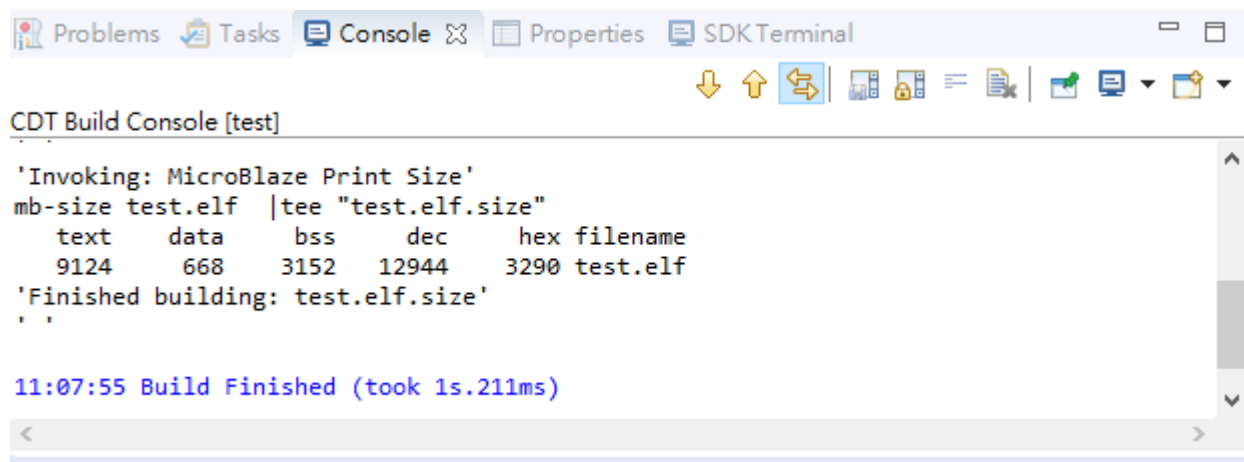
    XUartLite_Initialize(&UartLite, XPAR_UARTLITE_0_DEVICE_ID);

    while(1) {
        XUartLite_Recv(&UartLite, RecvBuffer, TEST_BUFFER_SIZE);

        led_check = RecvBuffer[0];
        xil_printf("show led");
        XGpio_DiscreteWrite(&led, 1, led_check);
        sleep(1);
        //LED_IP_mWriteReg(XPAR_LED_MODULE_DEVICE_ID, 0, led_check);
    }
}
```

# 實驗步驟

- ❖ 當在Console中看到Build Finished，即為成功。



The screenshot shows the CDT Build Console interface. The title bar includes 'Problems', 'Tasks', 'Console', 'Properties', and 'SDK Terminal'. The 'Console' tab is active, displaying the following output:

```
CDT Build Console [test]

'Invoking: MicroBlaze Print Size'
mb-size test.elf |tee "test.elf.size"
  text  data  bss   dec   hex filename
  9124   668   3152  12944  3290 test.elf
'Finished building: test.elf.size'
.,

11:07:55 Build Finished (took 1s.211ms)
```

The console output indicates that the build process for 'test.elf' has completed successfully, showing memory usage statistics and the final build time of 1s.211ms.

# 實驗步驟

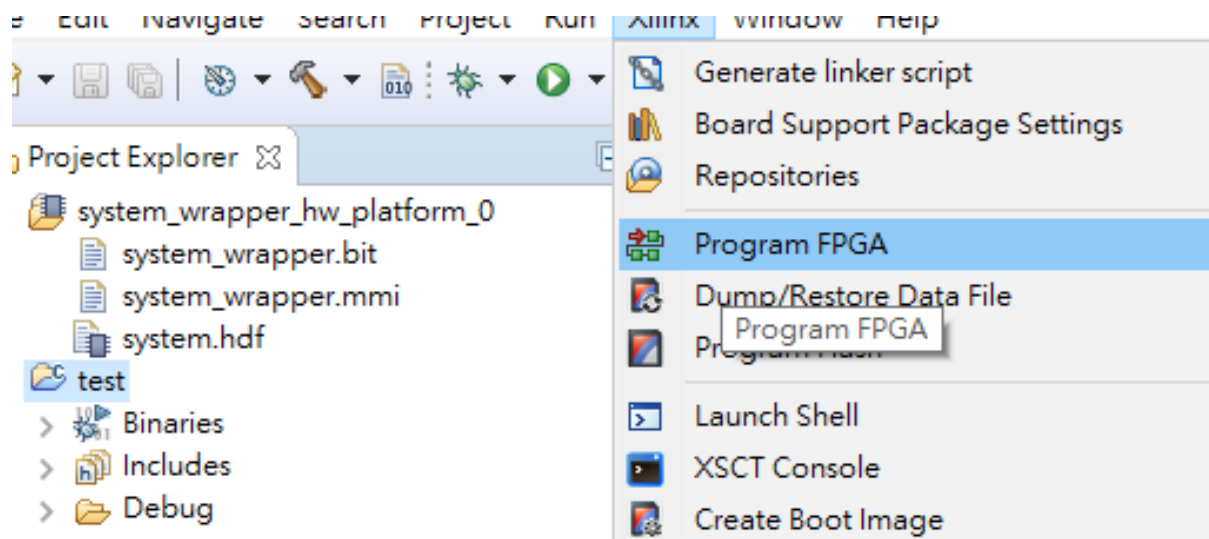
- ❖ 把FPGA板接上電腦並開啟。
- ❖ 確認FPGA版在哪個com腳。



USB Serial Port (COM7)

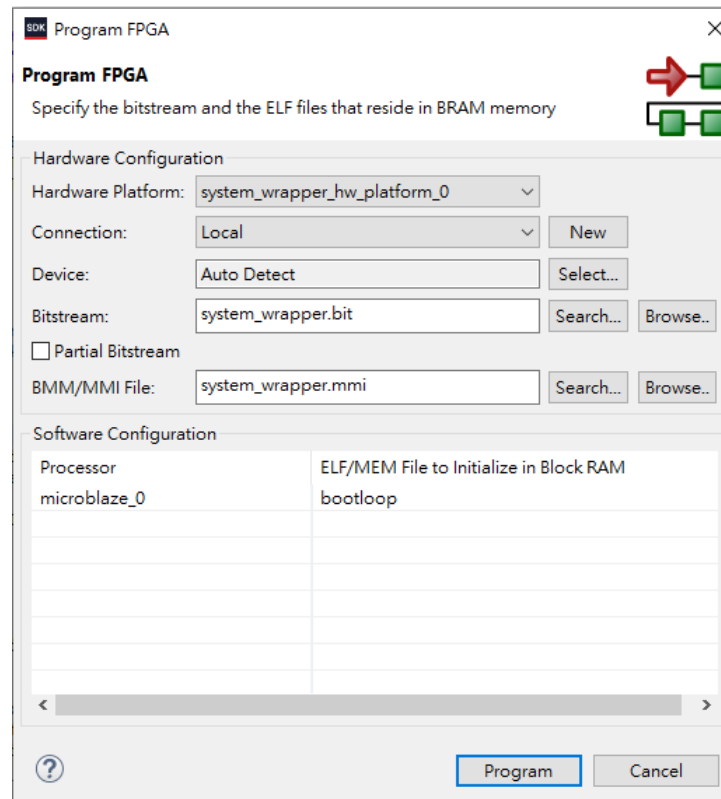
# 實驗步驟

- ❖ 選擇Xilinx Tools -> Program FPGA。



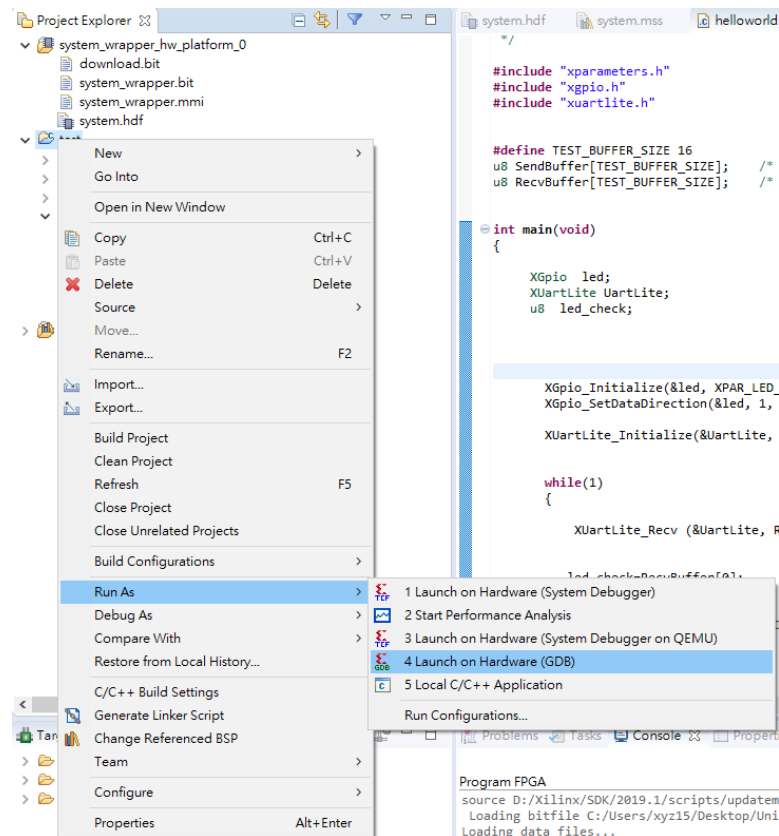
# 實驗步驟

- ❖ 軟體自動找到本專案的.bit和BMM檔，找不到則點擊Search找到專案對應位置。點擊Program。



# 實驗步驟

- ❖ 選擇檔案按右鍵，選擇run as -> launch on Hardware (GDB)。
- ❖ 軟體會自動把 elf 檔配置到Microblaze CPU。



# 實驗步驟

❖ 打開瀏覽器並搜尋串列埠調試助手。



## 串口調試助手

lingguang • 開發人員工具 > 公用程式

- 1.接收從串口進來的數據並在窗口顯示。
- 2.接收到的數據顯示方式可以選擇為“字符串”或“HEX”。
- 3.中文顯示無亂碼。可以在設置中更改字符串編碼類型。支持多種字符串編碼

[更多](#)



一般

普通級

免費+

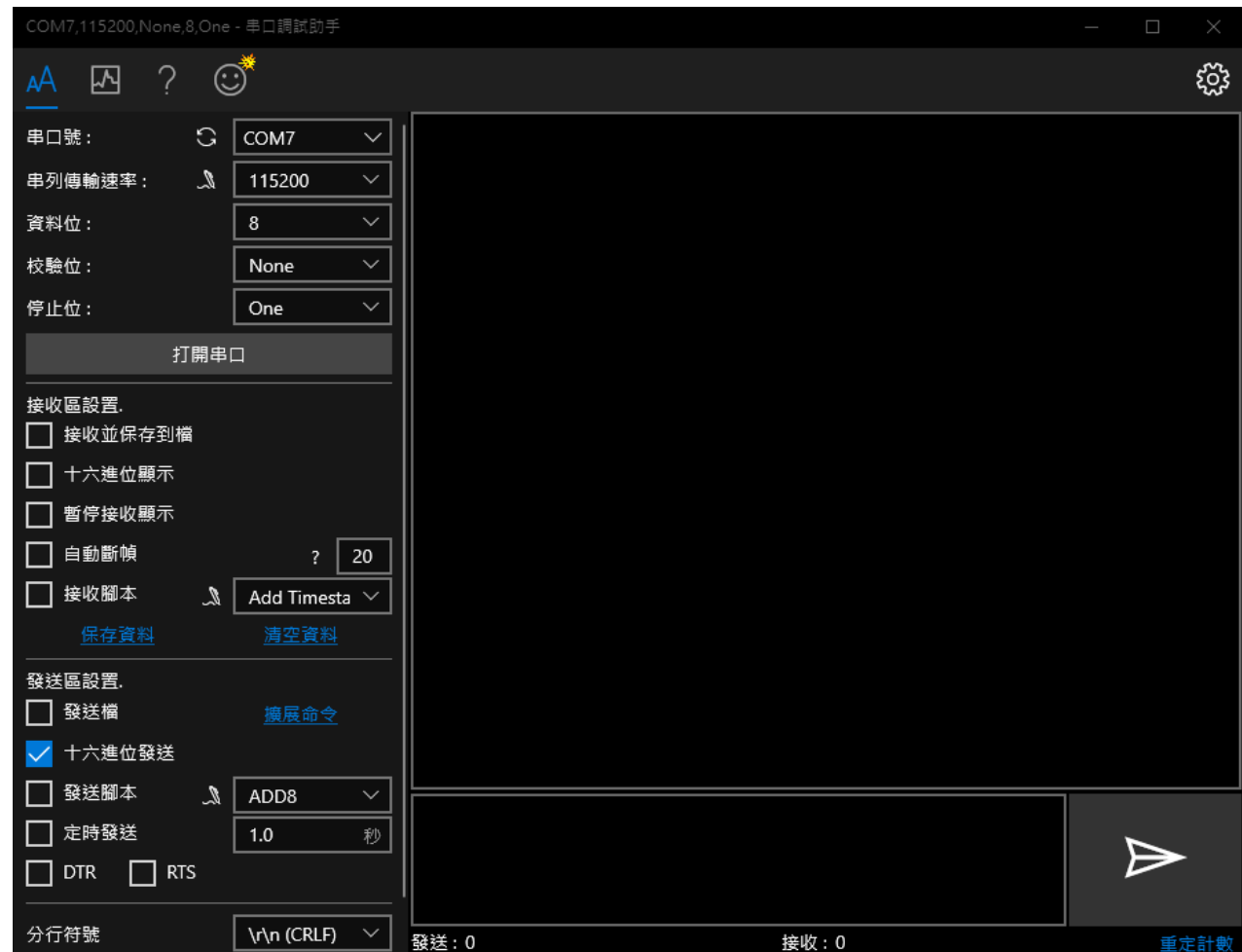
取得

+ 提供在應用程式內購買

△ [查看系統需求](#)

# 實驗步驟

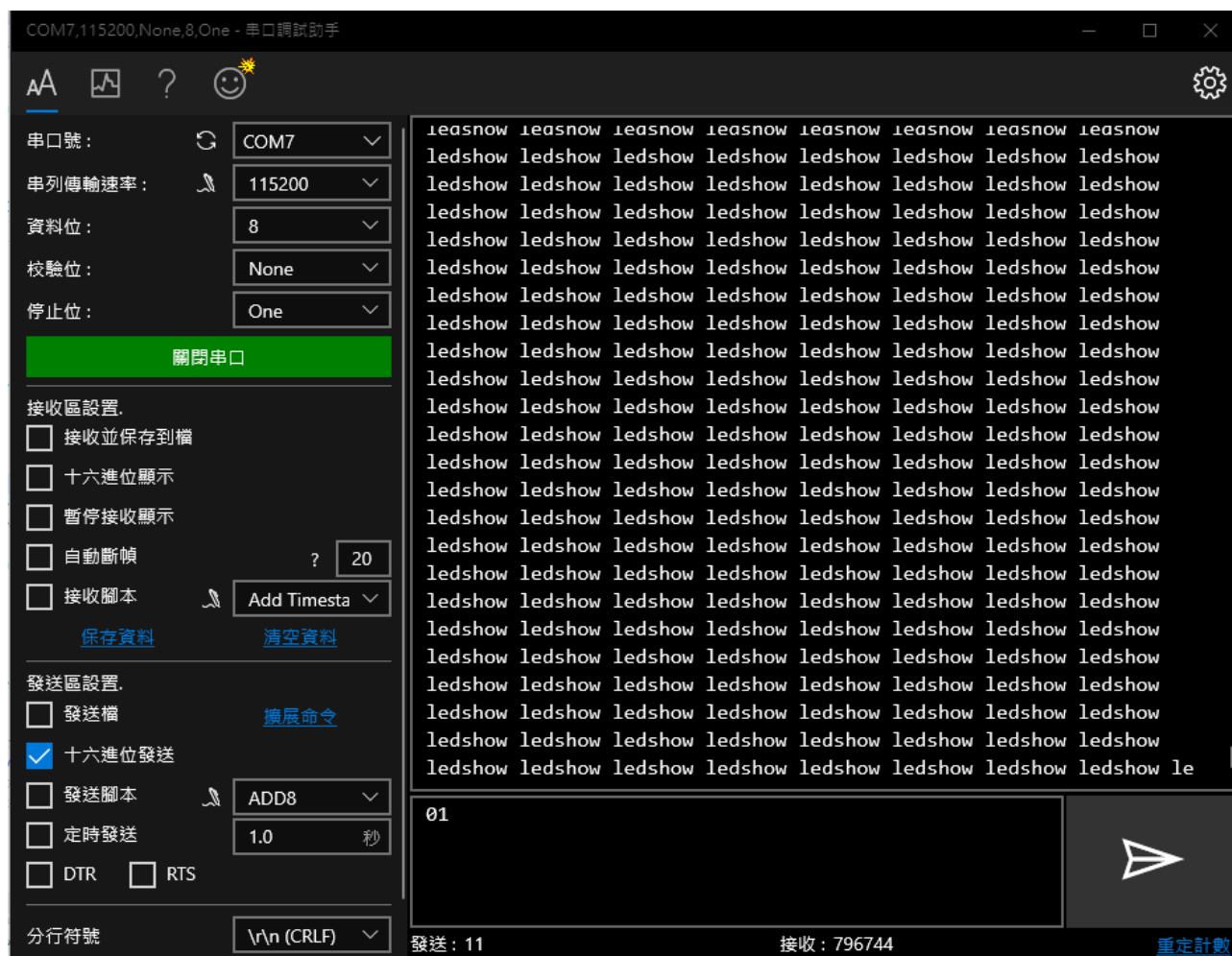
❖ 設定如下。





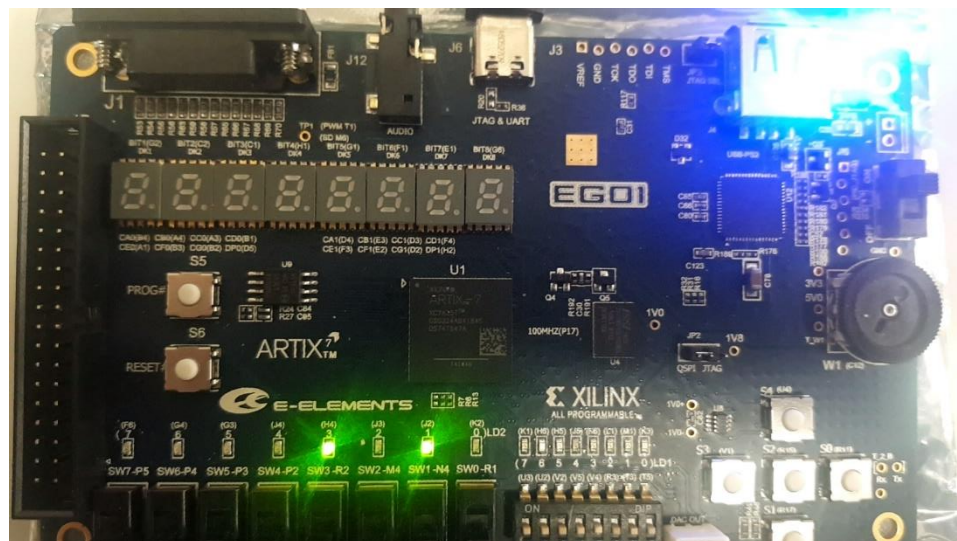
# 實驗步驟

## ❖ 打開串口。



# 實驗步驟

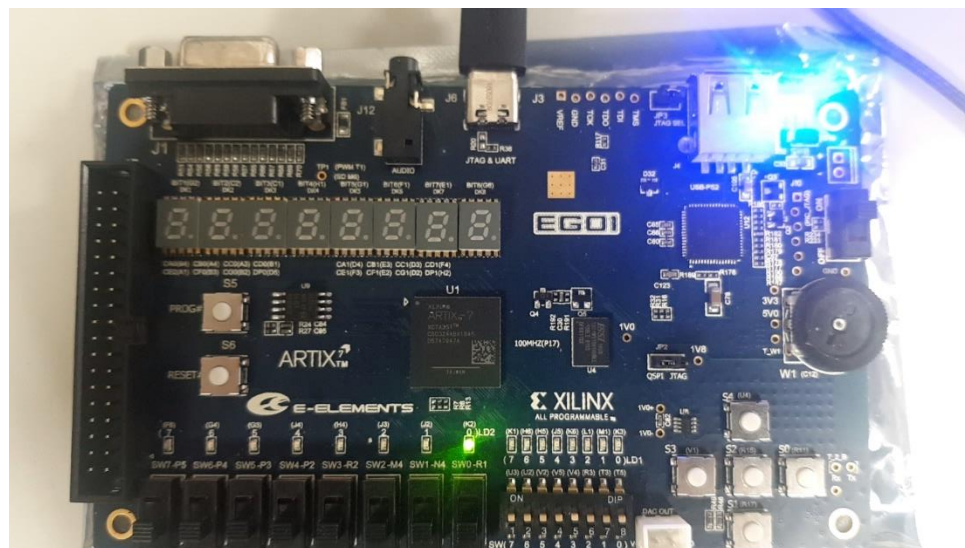
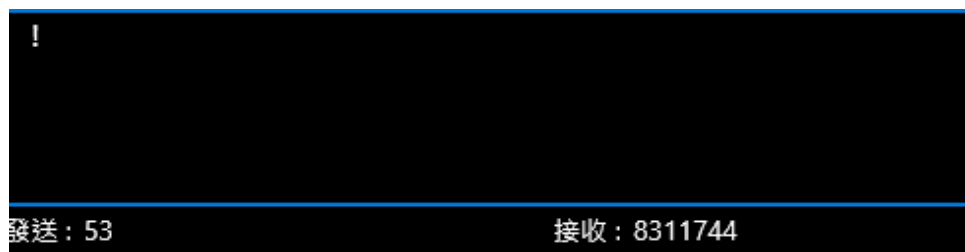
## ❖ 輸入16進制：



二進位		十進位	十六進位
0000	0000	0	00
0000	0001	1	01
0000	0010	2	02
0000	0011	3	03
0000	0100	4	04
0000	0101	5	05
0000	0110	6	06
0000	0111	7	07
0000	1000	8	08
0000	1001	9	09
0000	1010	10	0A

# 實驗步驟

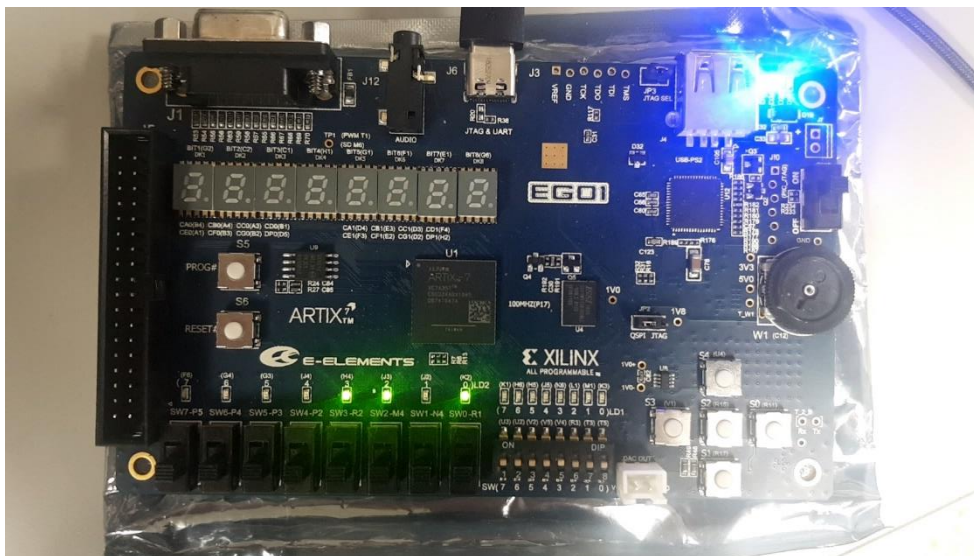
## ❖ 數入標點符號：



二進位	十進位	十六進位	圖形
0010 0000	32	20	(space)
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	\$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)

# 實驗步驟

❖ 輸入大寫英文字母：

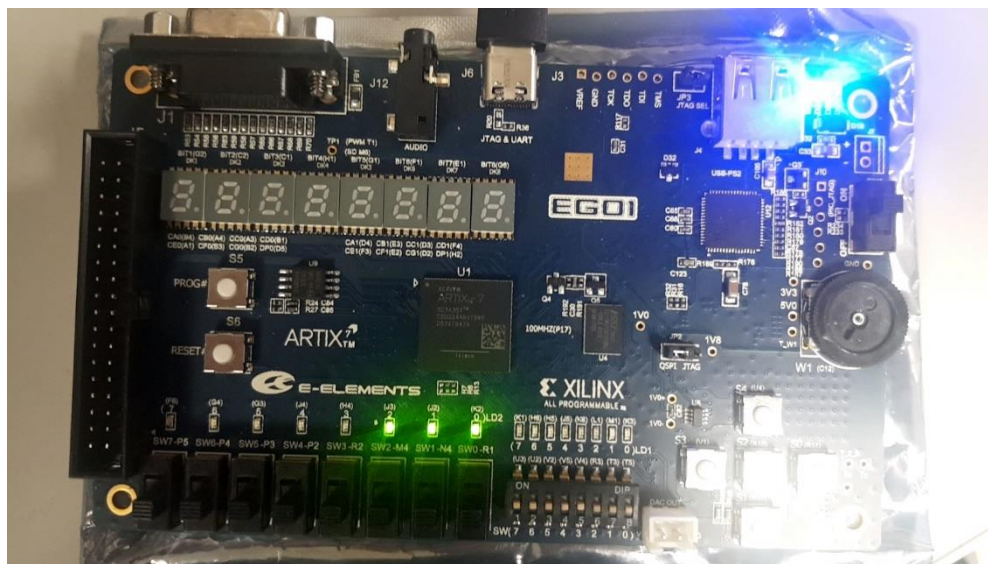


二進位	十進位	十六進位	圖形
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O



# 實驗步驟

## ❖ 輸入小寫英文字母：



二進位	十進位	十六進位	圖形
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o



# Thank you for your attention!

