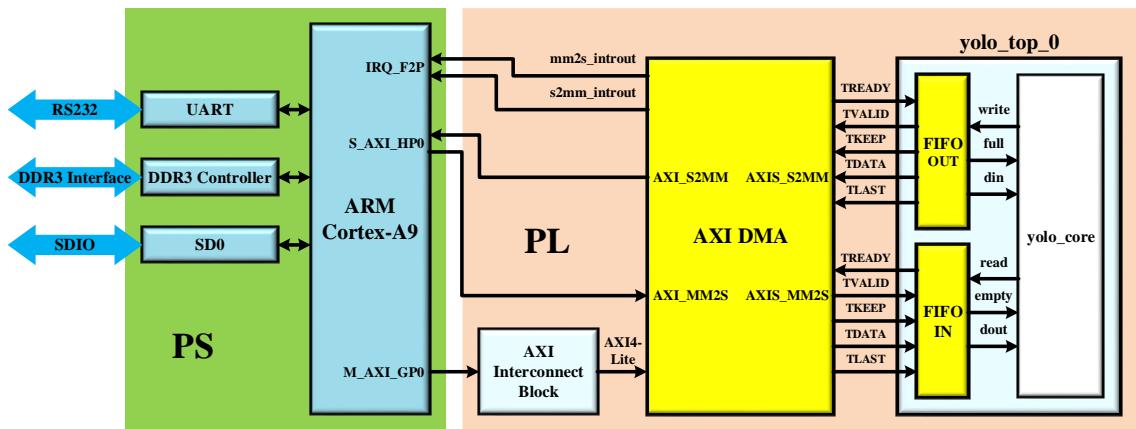


Lab 7 SD Card R/W & DMA Transfer

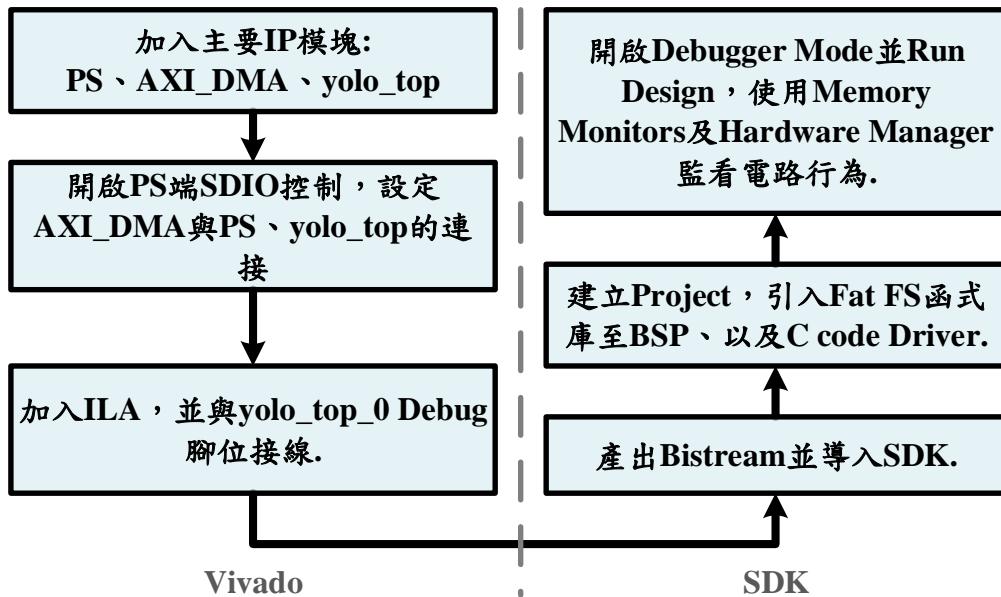
Design Introduction

本次 Lab 介紹 SOC Design 中常使用的 SD Card R/W，以及在 DDR 與自訂 IP 間透過 DMA 搬運資料的行為。實驗流程中，首先將一串數值寫入 SD Card 中，在從 SD Card 讀取資料存至 DDR，接著藉由 AXI_DMA 將 DDR 中的資料搬運至自訂 IP 進行簡單四則運算，再將自訂 IP 算完的結果經由 DMA 搬運至 DDR 中，最後將存入 DDR 之結果與 Pattern 進行比對。

Design Overview



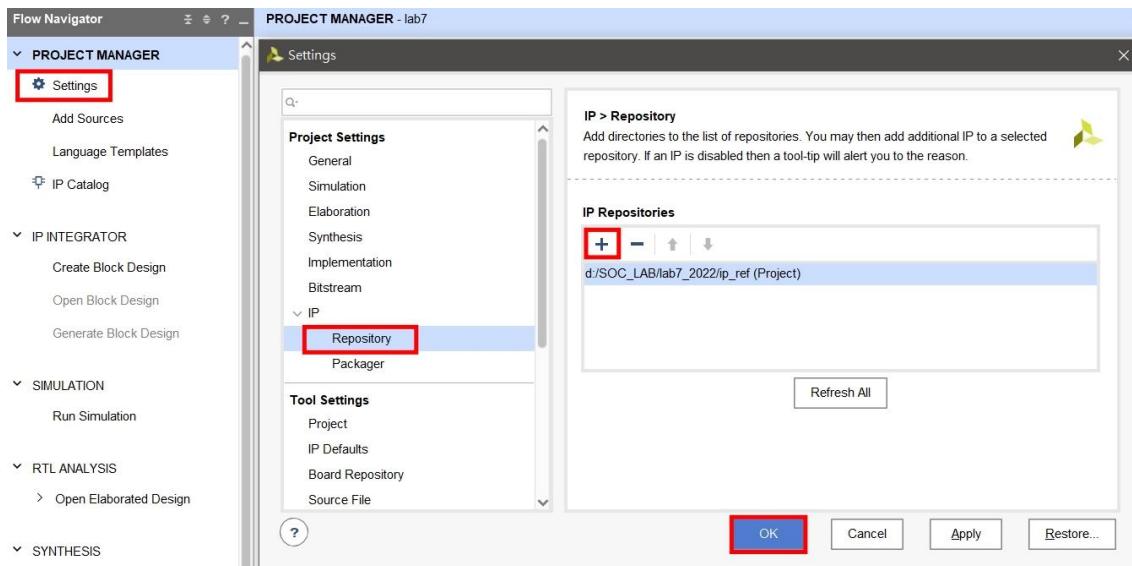
General Flow for this Lab



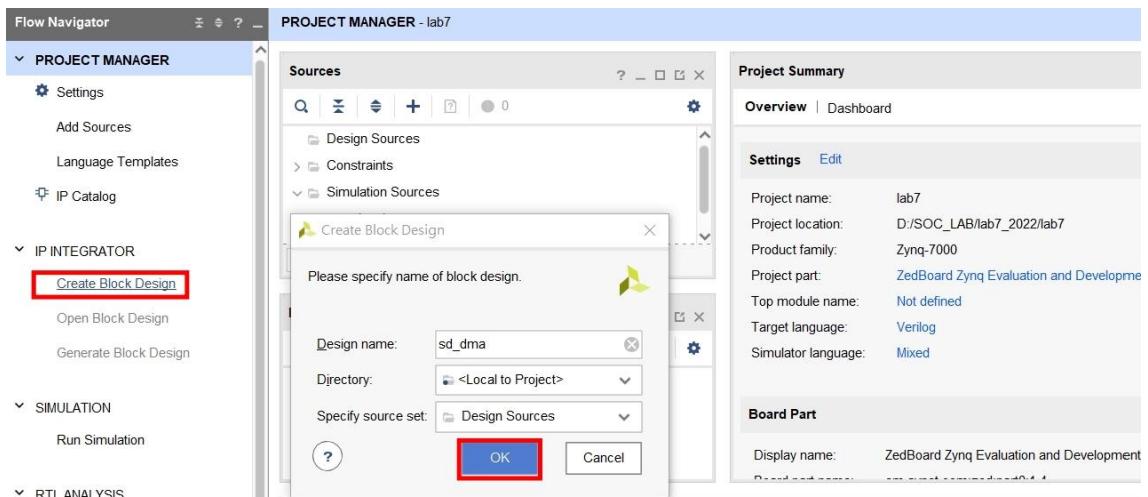
1. 建立專案

1.1 開啟 Vivado，選擇 Create Project→RTL Project→不用 Add Source、Constraint→選擇板子(Zedboard)→finish.

1.2 開啟 Setting→Repository，加入包好的 IP 專案 ip_ref 資料夾.

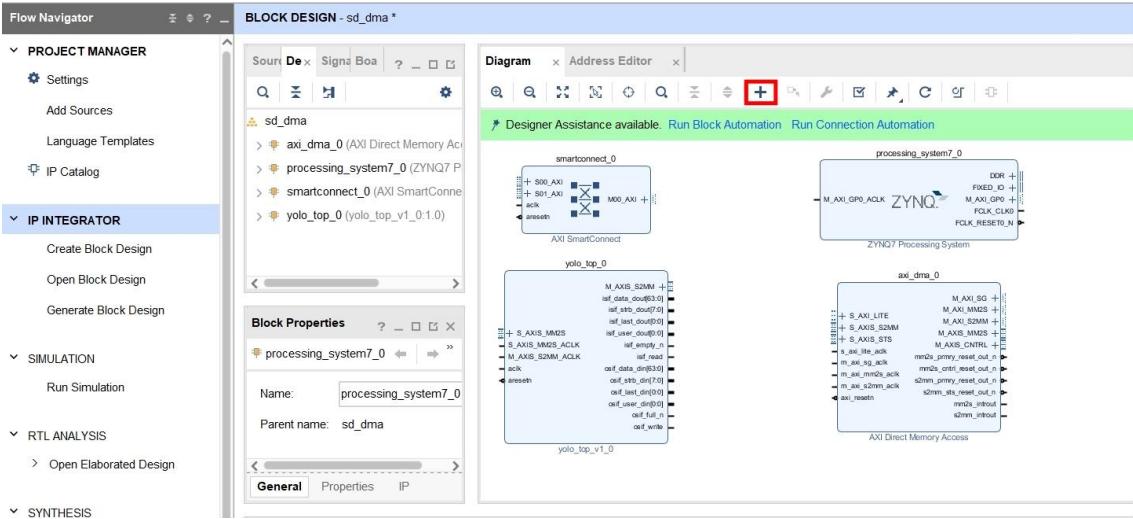


1.3 創建 Block Design，點選 Flow Navigator 的 Create Block Design.

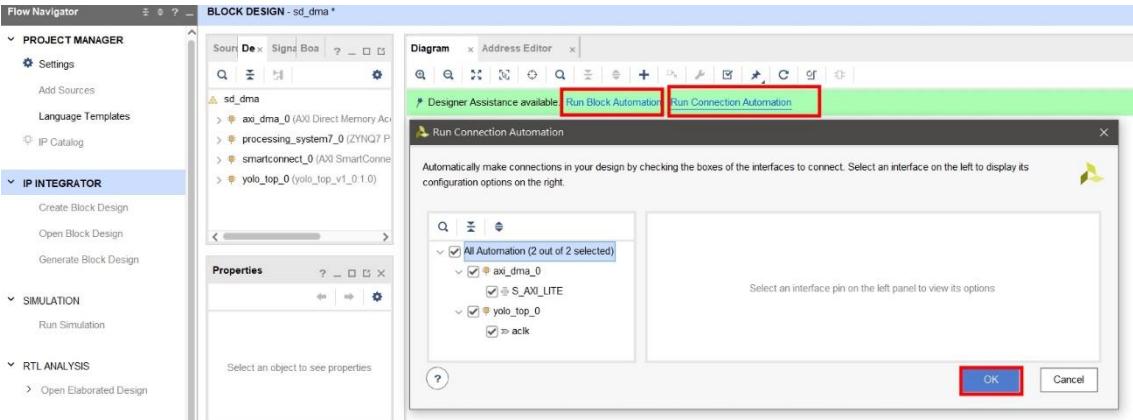


1.4 點選 Add IP，並加入以下 IP:

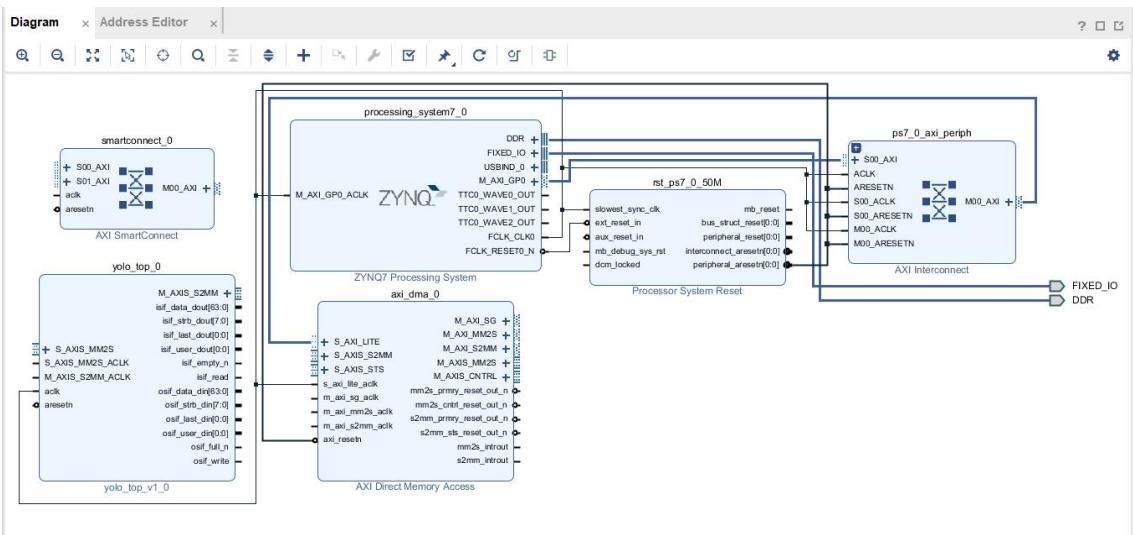
1. Zynq7 Processing System
2. AXI Direct Memory Access
3. AXI SmartConnect
4. yolo_top_v1_0



1.5 點選 Run Block Automation 讓 CAD Tool 進行 IP 初始設定，再來點選 Run Connect Automation(全勾選)將基礎的線接上。



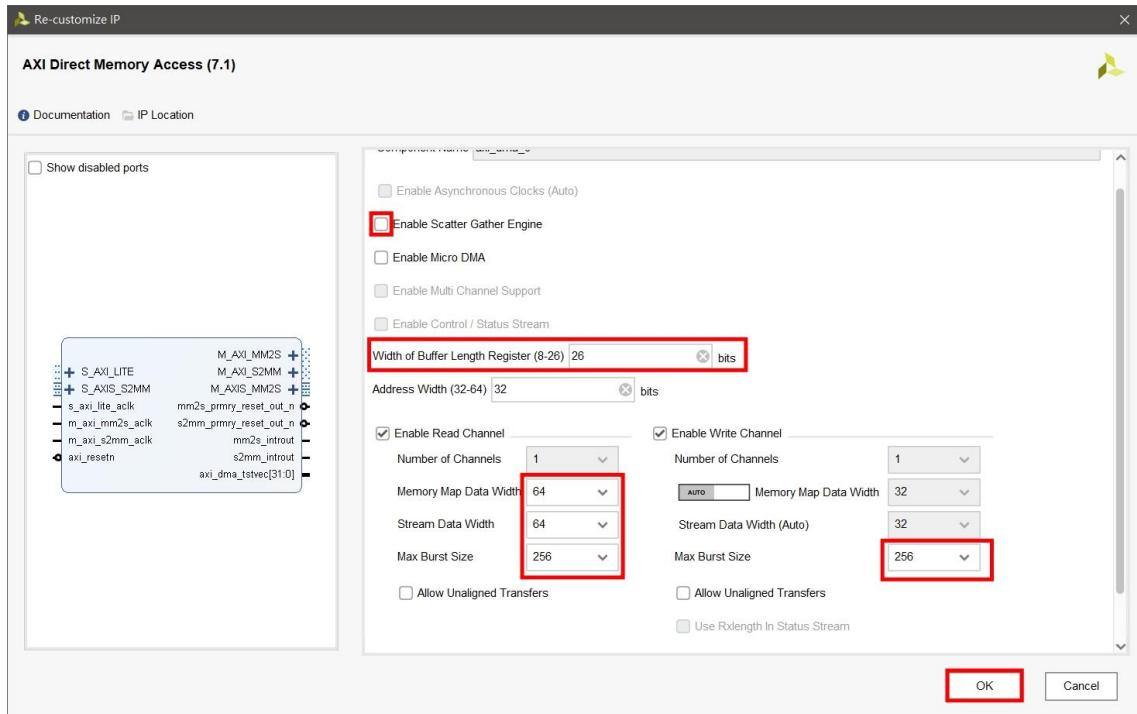
設定完成後應如下圖：



1.6 AXI Direct Memory Access 設定

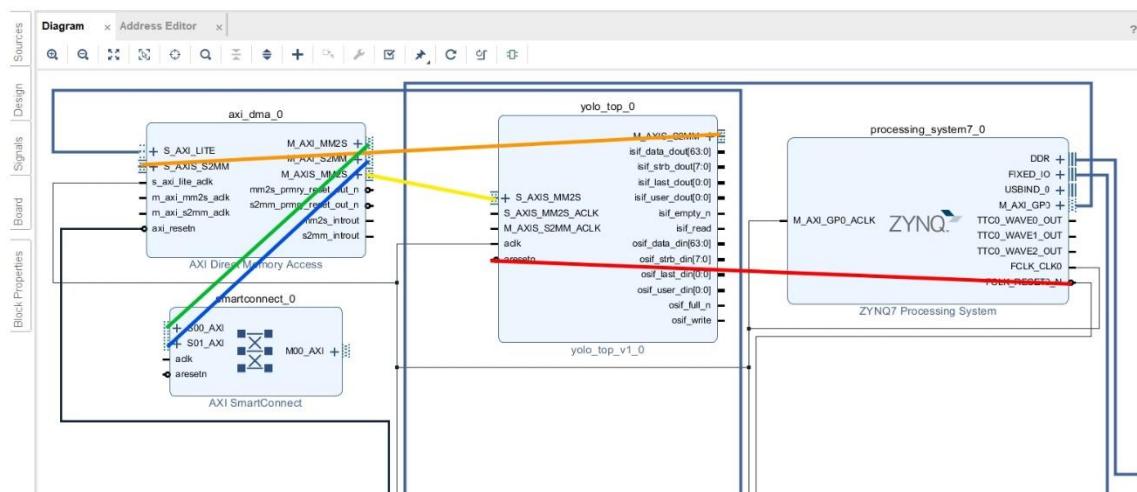
1. 點開 AXI DMA IP

2. 此 Lab 所使用的 DMA 範例為單 Channel 讀寫分開運作，所以不須使用多通道的傳輸，將 **Enable Scatter Gather Engine** 取消選擇。
3. 指定 MM2S 與 S2MM 在資料傳輸上最大的有效字節，**Width of Buffer Length Register** 設定成 26bits (即最大傳輸量為 2^{26} Byte)。
4. 設定 MM2S 與 S2MM 傳輸時 Read/Write 的 Memory MAP Data Width。Memory MAP Data Width 為 64bits，Stream Data Width 為 64bits。
5. 設定單次傳輸的資料筆數，**Max Burst Size** 設定為 256。



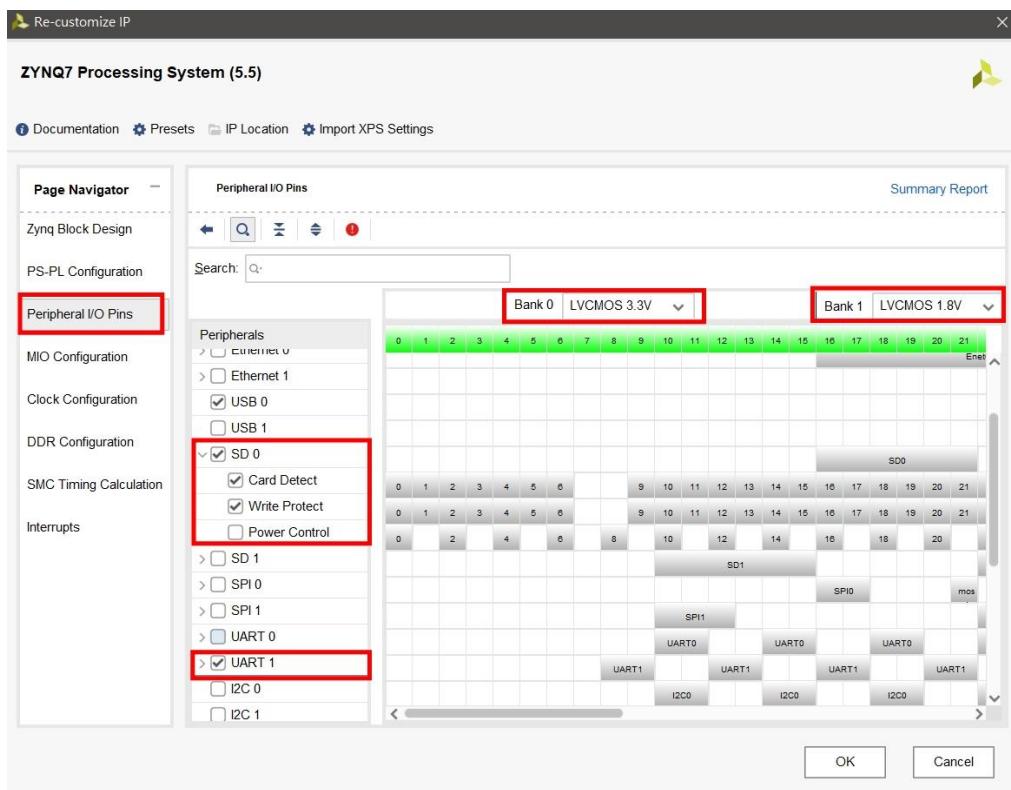
1.7 DMA 與 PL(yolo_top_0)連接

於 Block Diagram 中，將 yolo_top_0 與 AXI DMA 的 S2MM 與 MM2S PIN 腳接上，以及將 yolo_top_0 接上 PS 的 Reset 訊號。



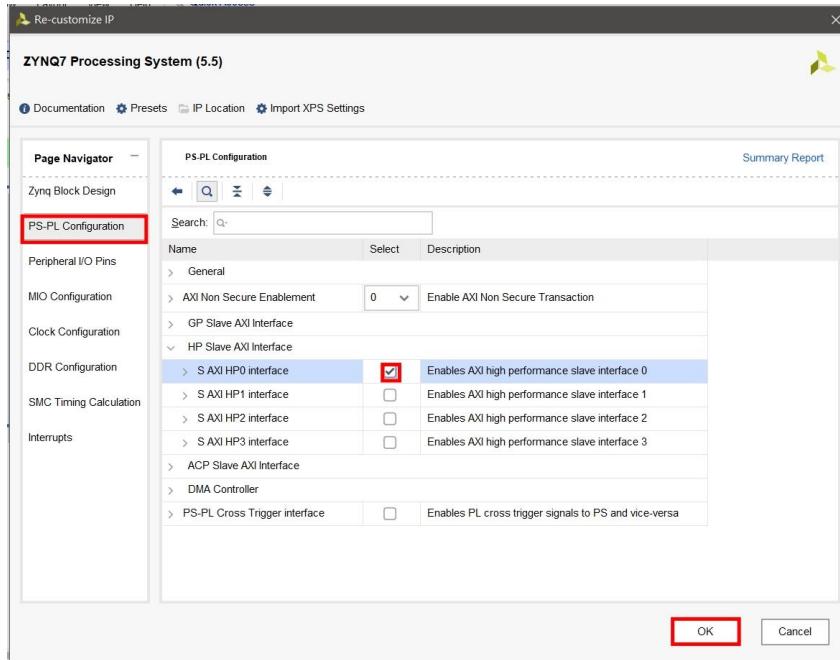
1.8 PS 端 Peripheral I/O 設定，點開 Processing System 後，進行以下步驟：

1. 確認 UART 1 Enable.
2. SD 0 內選擇 Card Detect、Write Protect (此 SD0 為開啟 SD 卡的功能)
3. 於右方綠色欄位上方，確認 Bank0 LVCMS 為 3.3V、Bank1 LVCMS 為 1.8V.

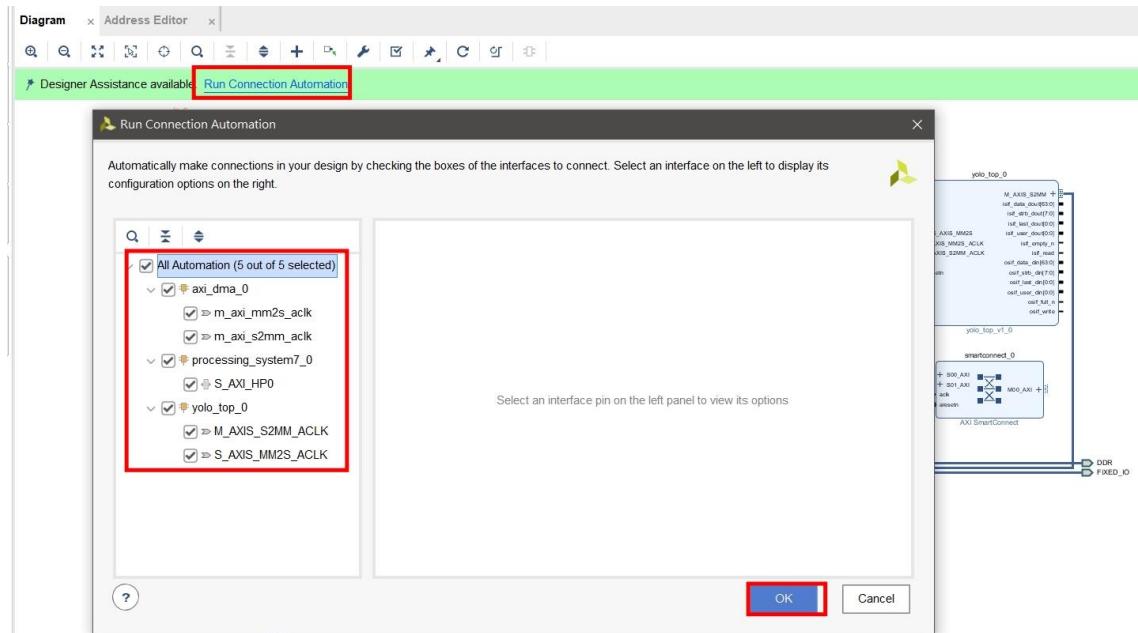


1.9 PS 端開啟與 DMA 進行資料交換的 AXI HP Interface.

至 PS-PL Configuration 選單，將 HP Slave AXI Interface 中 HP0 勾選，完成 PS 與 DMA 的基礎設定.

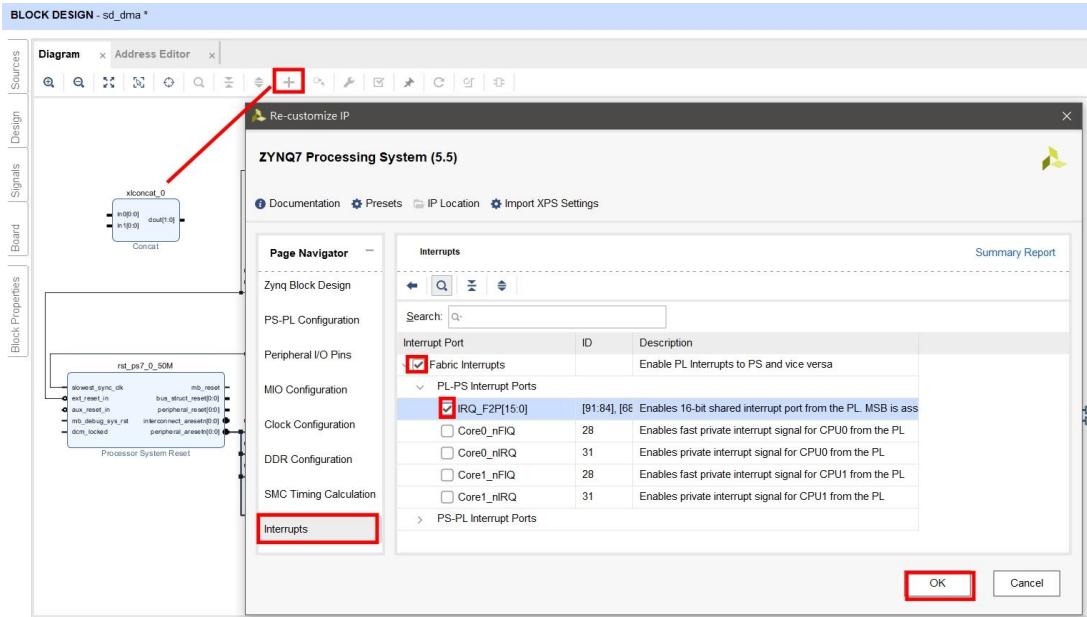


點選 Run Connection Automation.

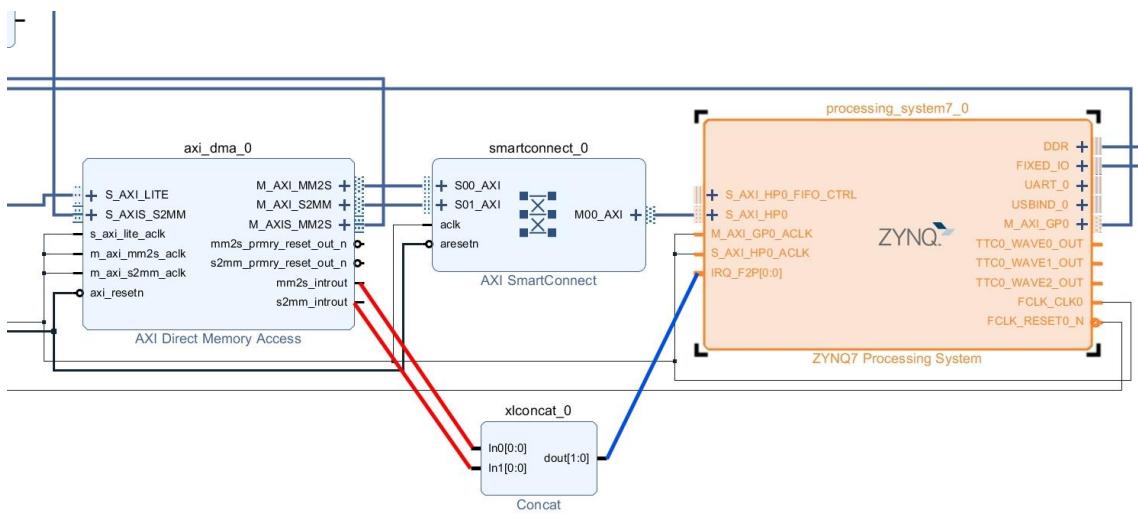


1.10 點選 Add IP 並加入 Concat，接著開啟 PS 的 IP 設定，點開 Interrupts 的 IRQ_F2P Port.

*Concat-IP 作用在當 DMA 搬完資料時，MM2S 與 S2MM 皆會產生 Interrupt 訊號(結束訊號)，而 Concat-IP 則負責將兩訊號 Concat 並接往 PS 端。

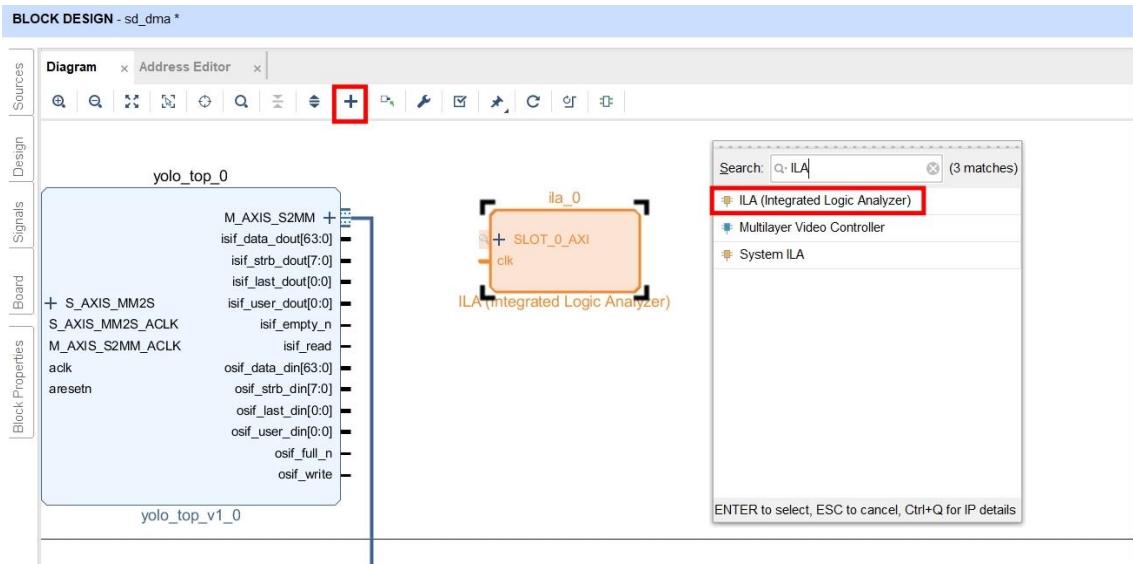


接著進行 Concat 與 PS、DMA 的接線，如下圖。



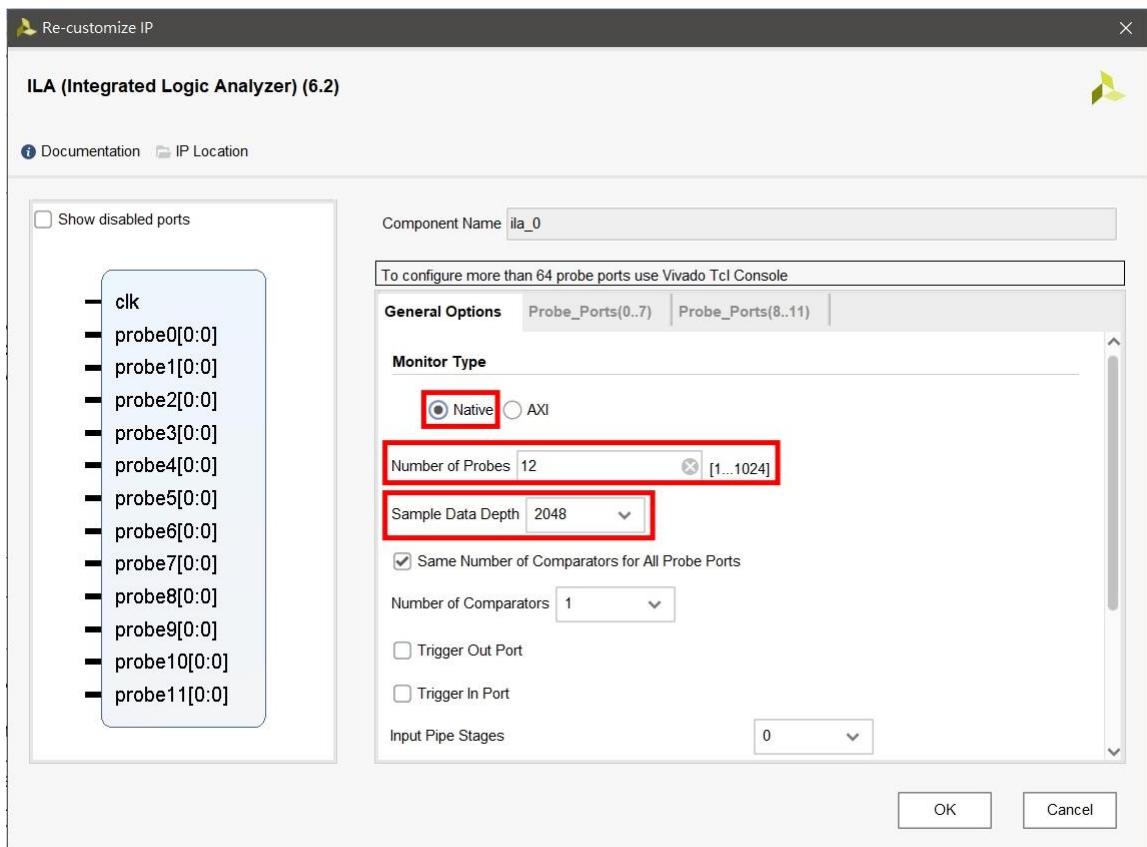
1.11 再來加入 Debug IP, 點選 Add IP 並加入 ILA (Integrated Logic Analyzer).

*ILA 作用為在 FPGA Programming 時，能使用 Hardware Manager 來叫出波形來 Debug IP 問題.

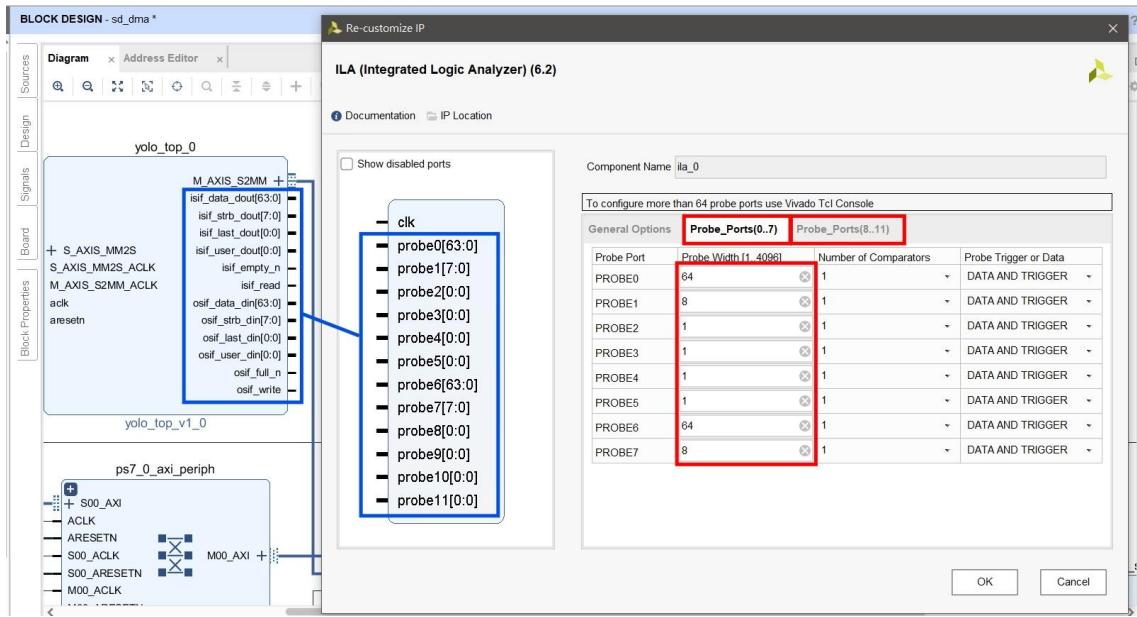


加入 ILA-IP 後，點兩下開啟 IP 設定：

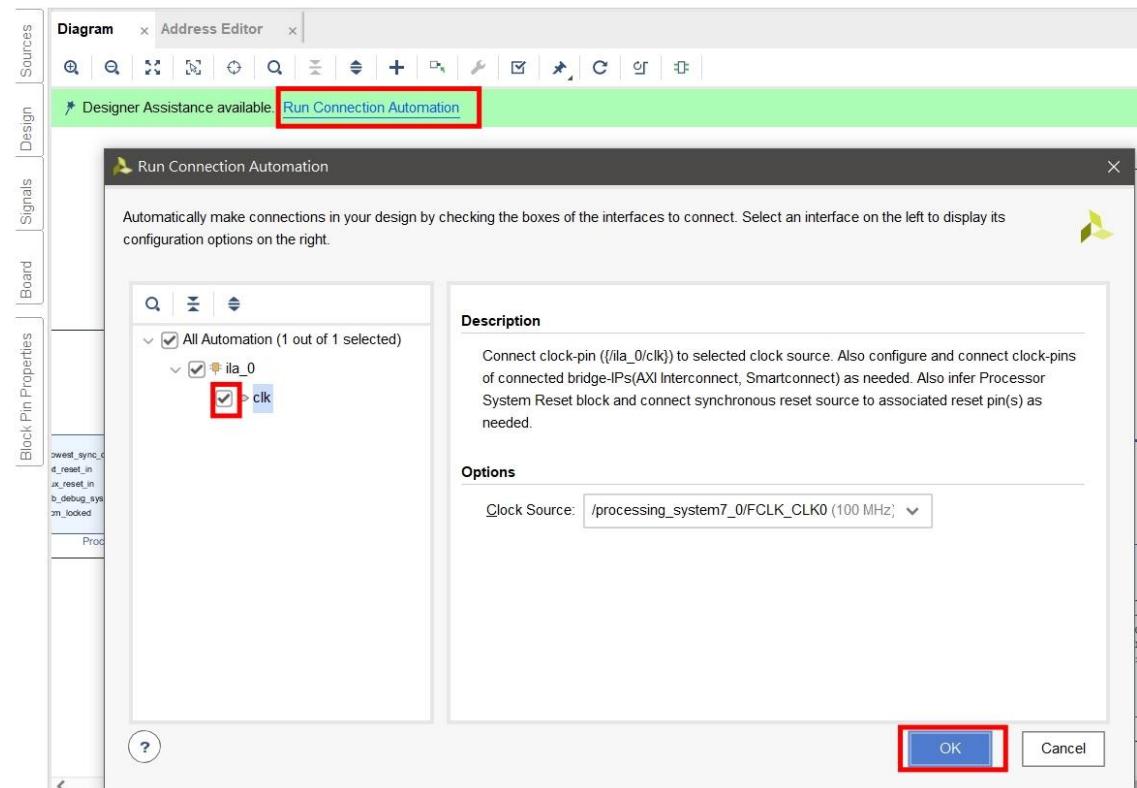
1. 改為 Native Mode.
2. Number of Probes 欄位輸入 12，以符合 yolo_top_0 IP 所拉出之 Debug 角位.
3. Sample Data Depth 設定成 2048. 此設定為 ILA-Debug 監看之 Window Size (Cycle 數).



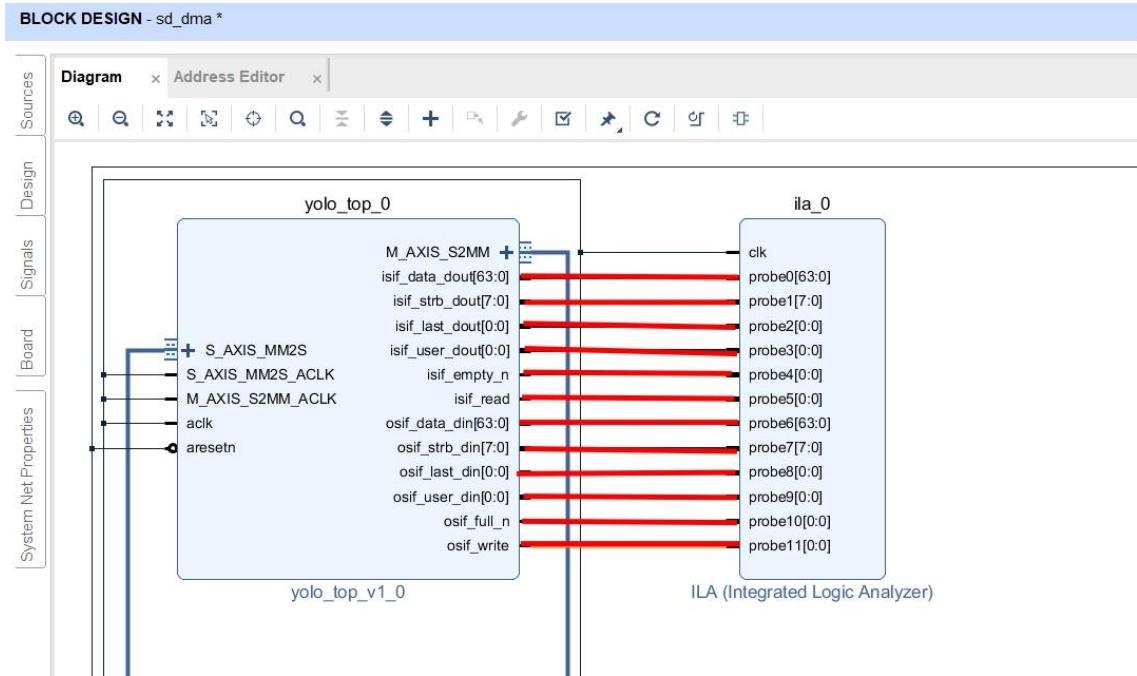
接著對設定 Debug Probes 的 Bit 數，這邊設定時需對照 yolo_top_0 拉出角位之 Bit 數，Probe_Ports(0~7)、(8~11)皆要設定，共 12 個 Port.



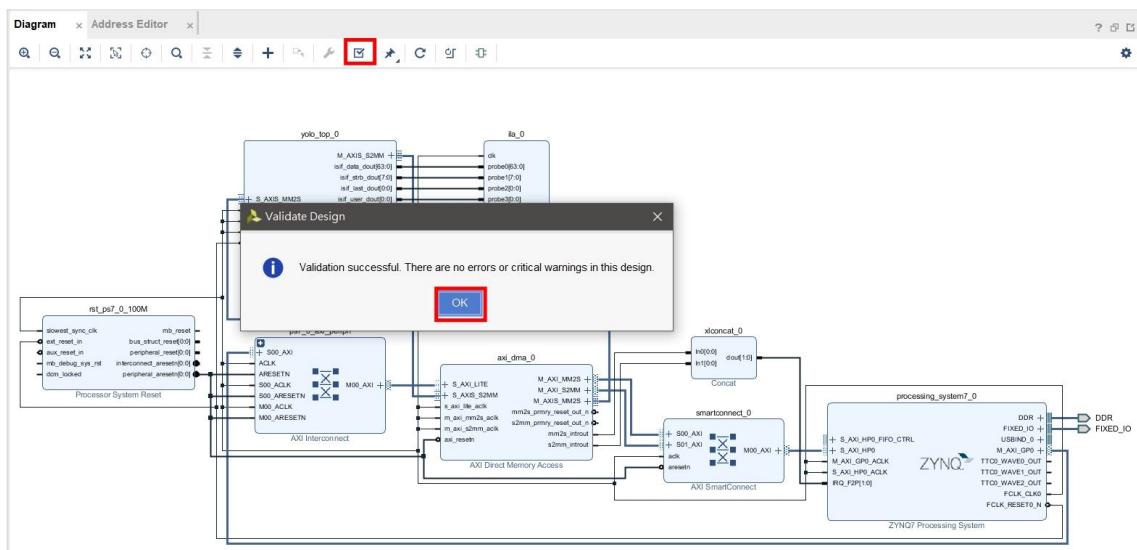
設定完後，執行 Run Connection Automation，將 ILA-IP 的 Clk 與 PS 接上。



接著對 **yolo_top_0** 與 ILA-IP 的腳位互相接上，這裡需對照相同的 Bits 數進行接線。

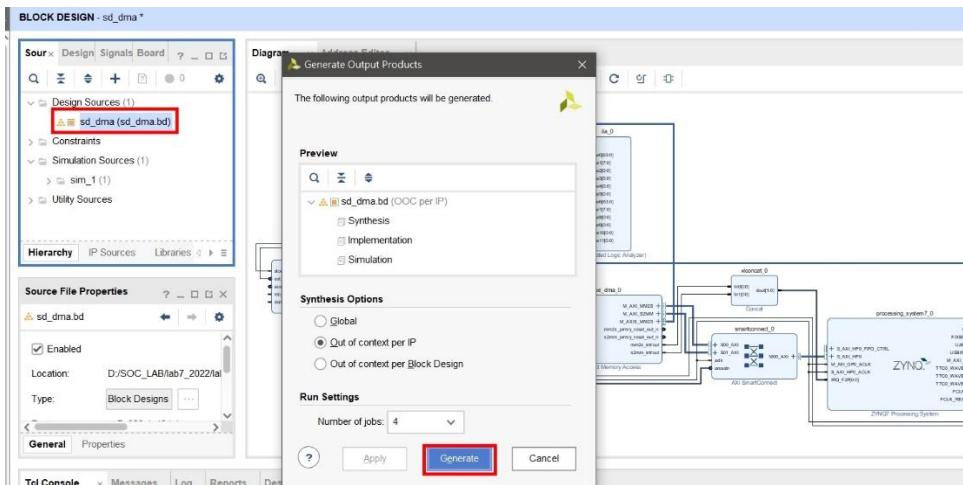


1.12 檢查 Design 目前有無問題，按下 Validate Design.

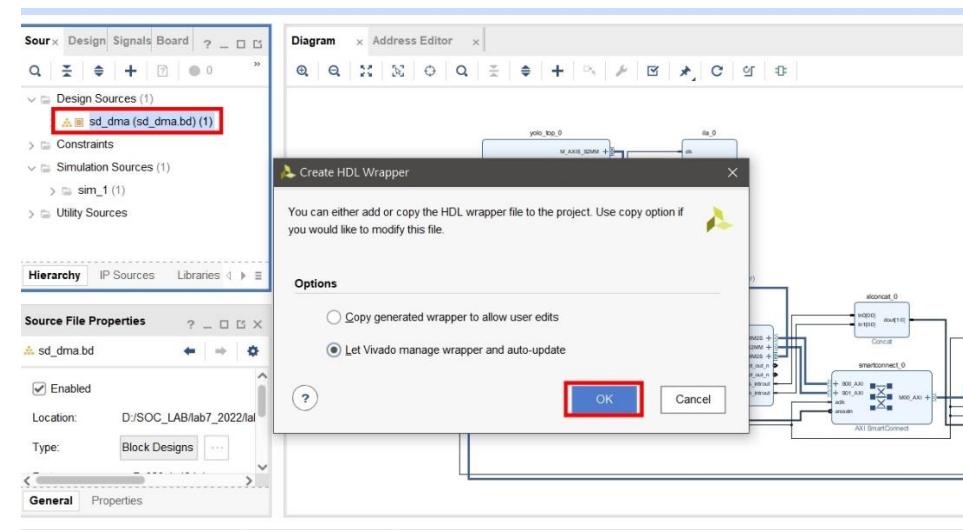


1.13 進行 Generate Output Products

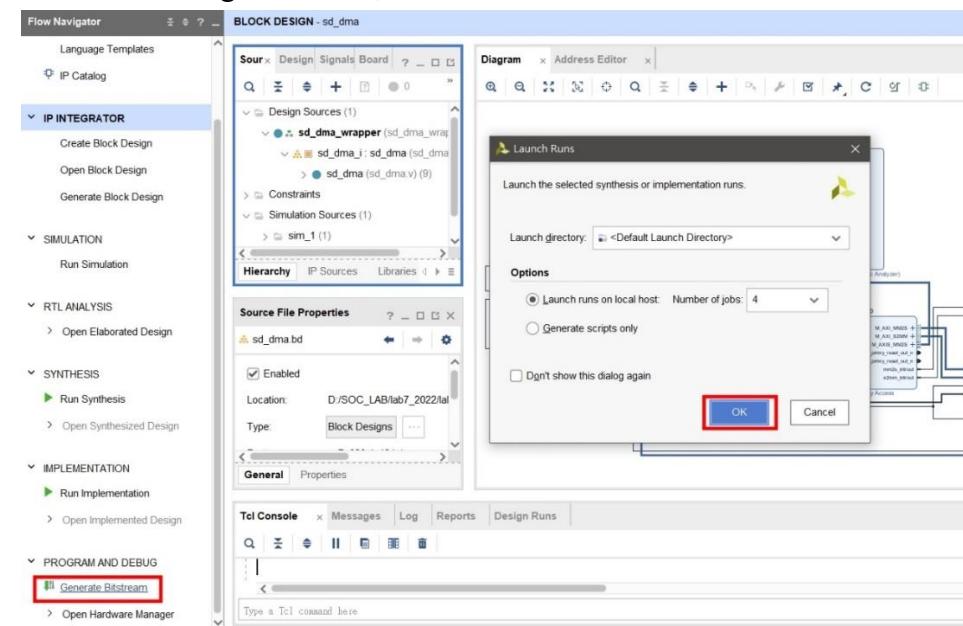
1. 對 Source window 的 Block Design 右鍵點選 Generate Output Products.



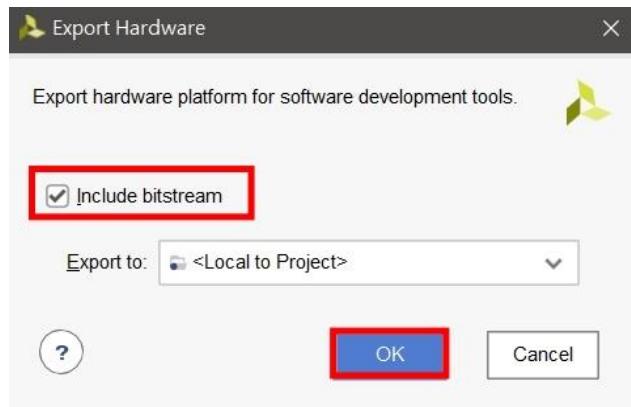
2. 對 Source window 的 Block Design 右鍵點選 Create HDL Wrapper.



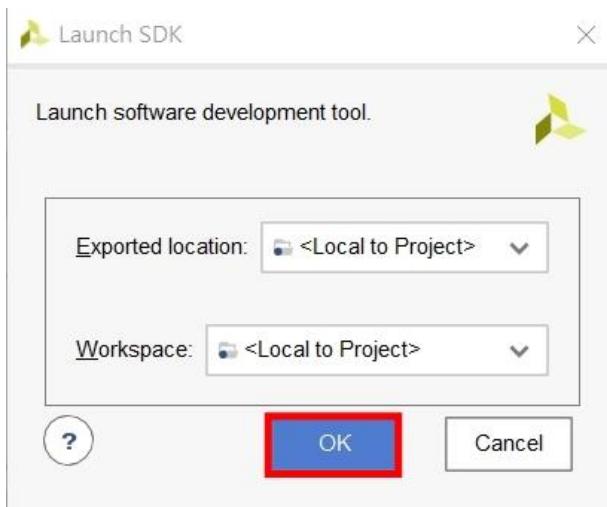
3. 於 Flow Navigator 處選擇 Generate Bistream.



**1.14 接著匯出 Bistream，File→Export.. →Export Hardware，跳出窗格後選
Include bistream.**

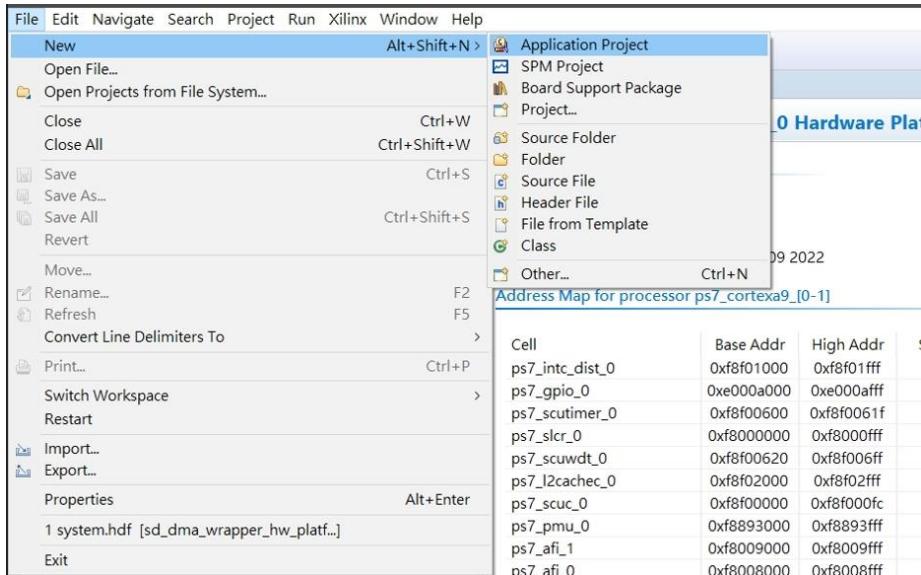


1.15 開啟 SDK，File→Launch SDK.

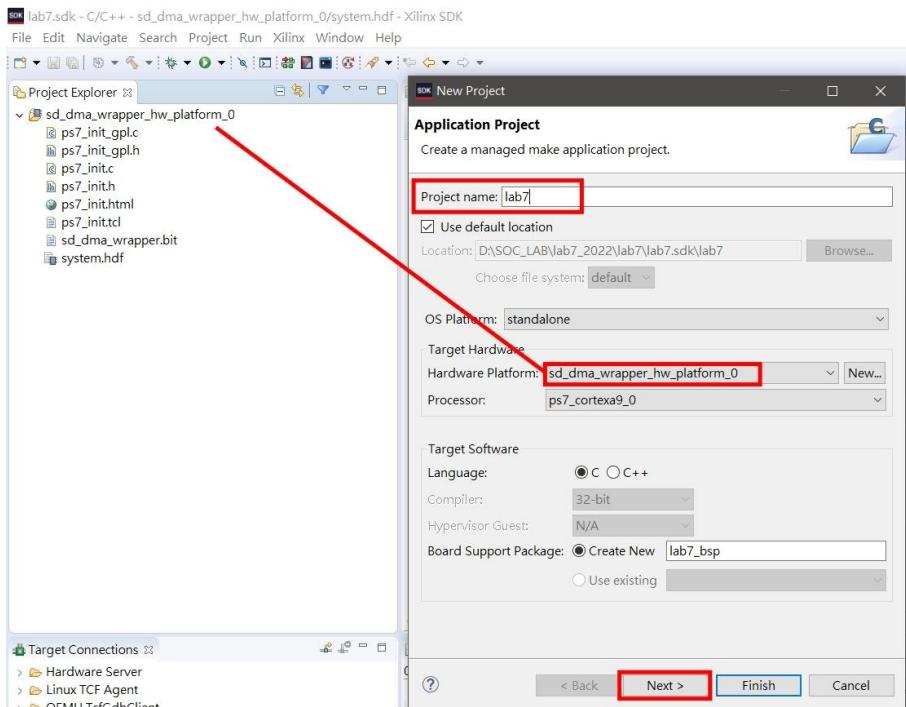


2. SDK: SD Card R/W & DMA Transfer

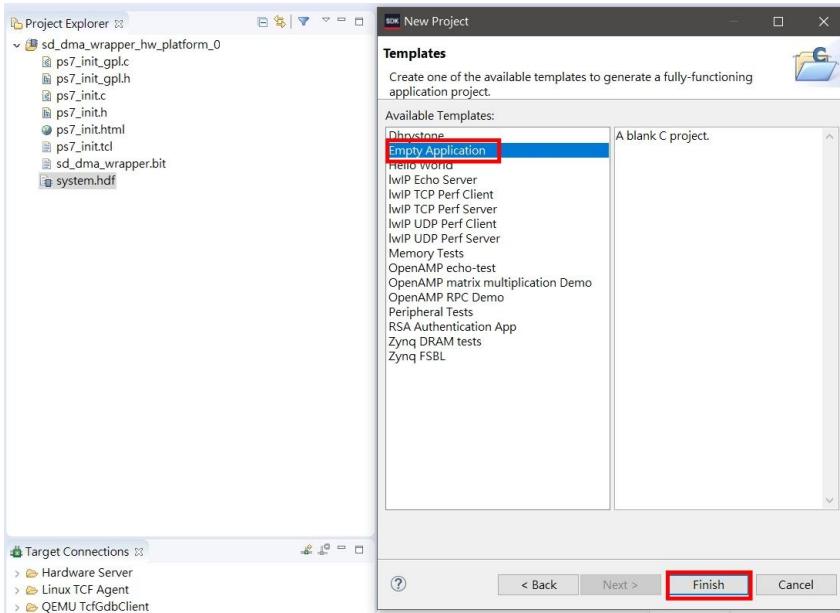
2.1 開啟 SDK 後，建立新 Project.



設定 Project Name、選擇當前 Hardware Platform.

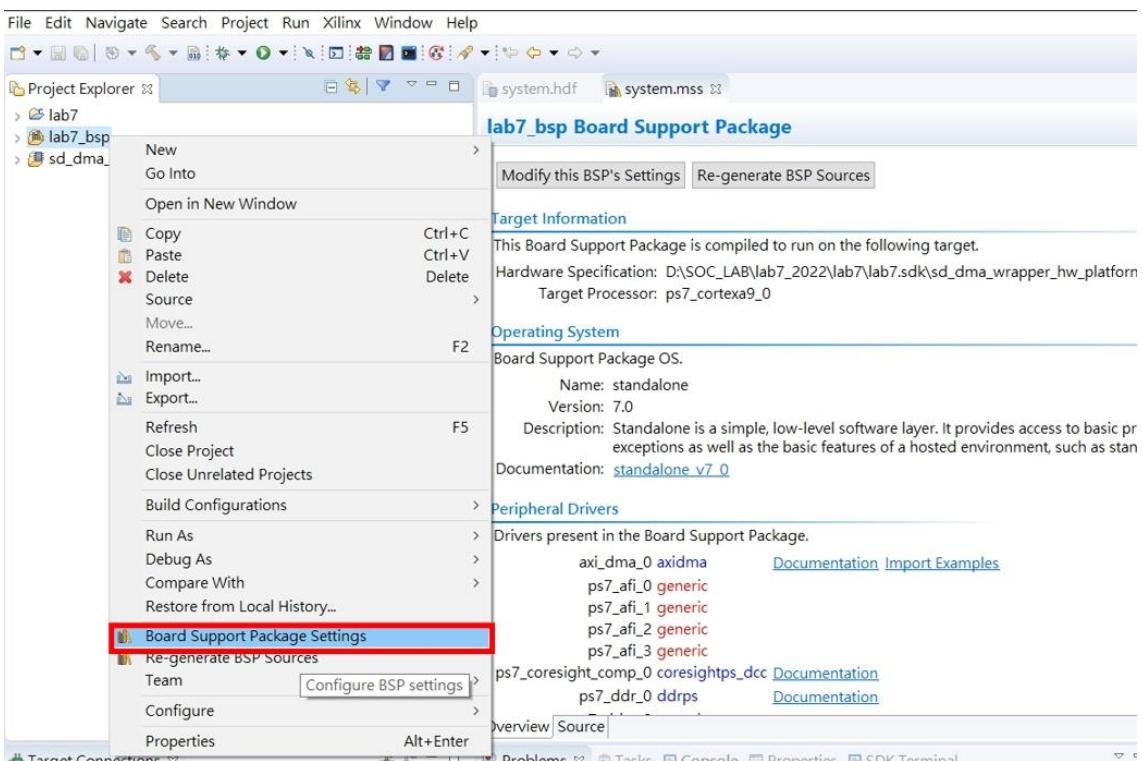


建立 Empty Application

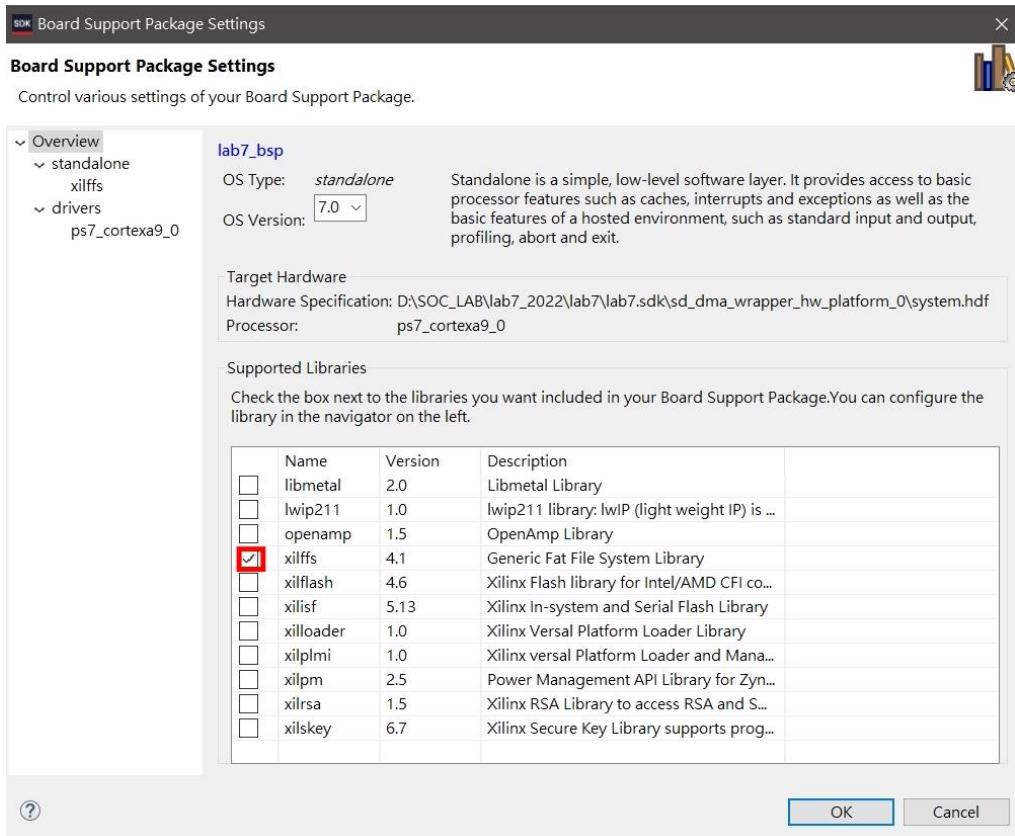


2.2 對 BSP 進行設定

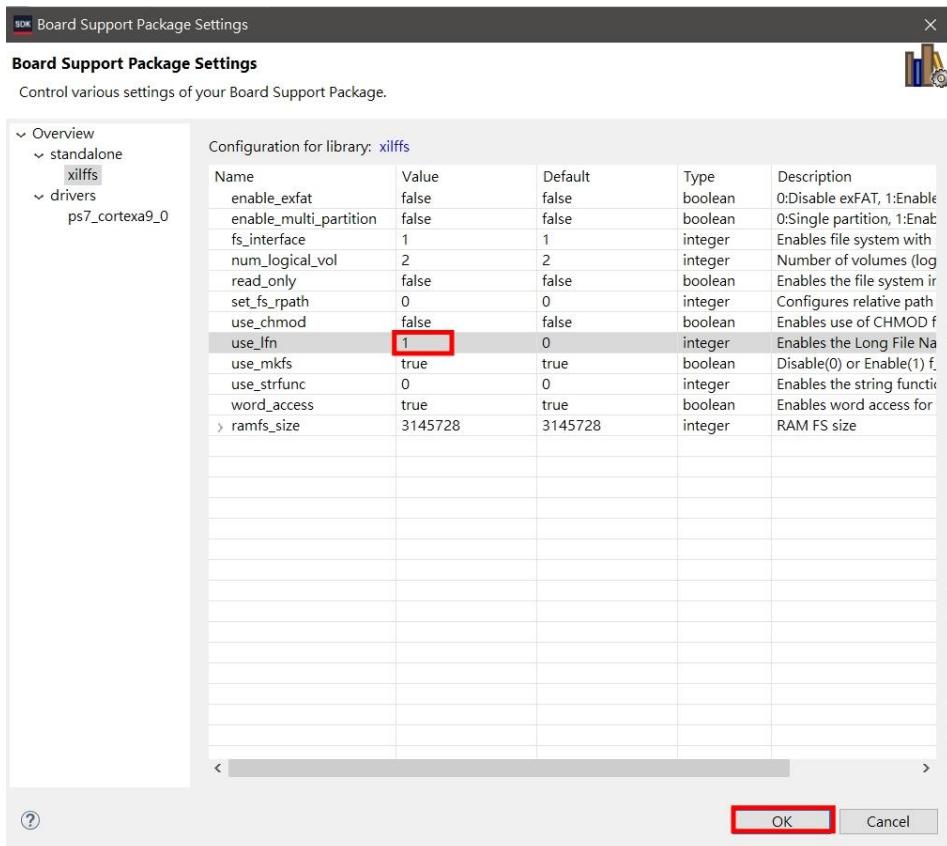
對 Project_Explorer 的 lab7_bsp 右鍵點選 Board Support Package Settings



1. 開啟 xilfss 以支援接下來的 SD Card function.

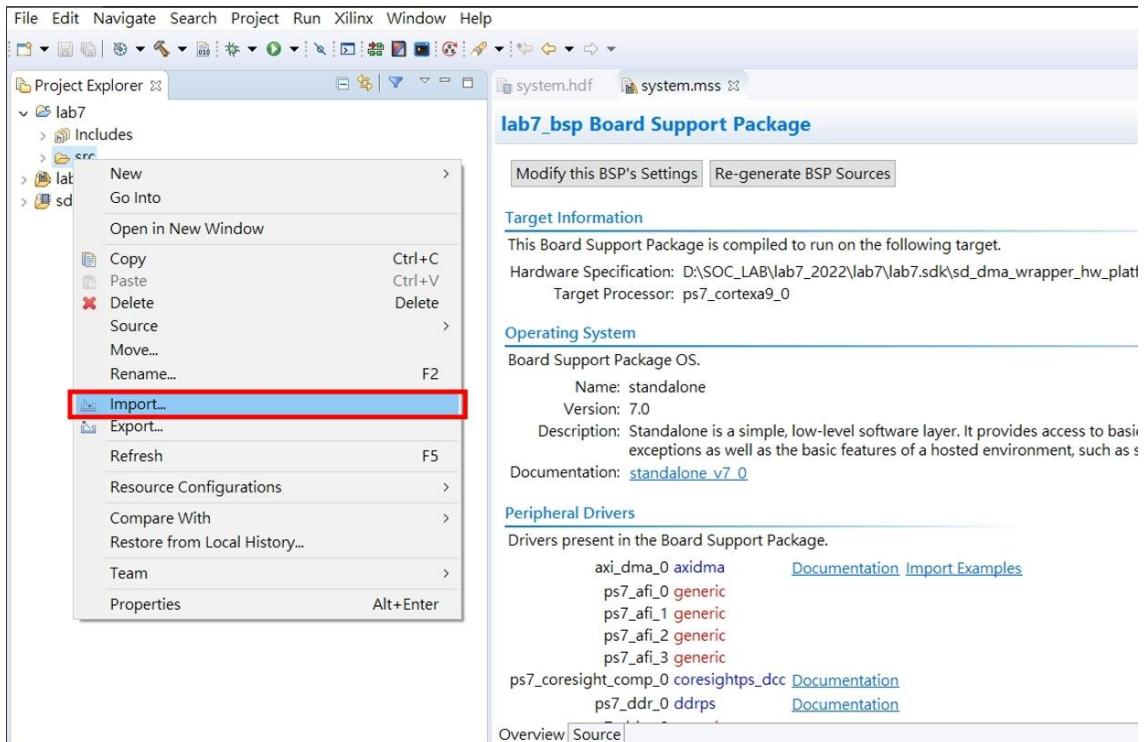


2. 到 standalone 下的 xilffs 頁面，將 use_lfn 的 Value 欄位改成 1，開啟長文字文件讀取功能。

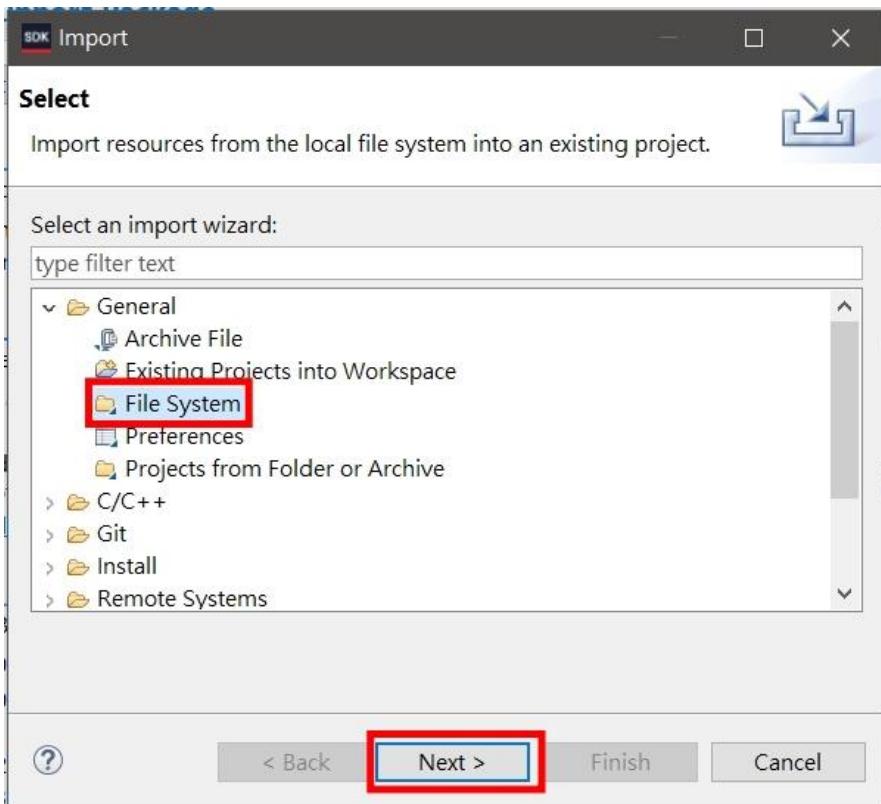


2.3 Load 進本 Lab 所需的 C code

1. 在 Project Explorer 下，對新增的 lab7 Project 下的 src 資料夾右鍵點選 Import.

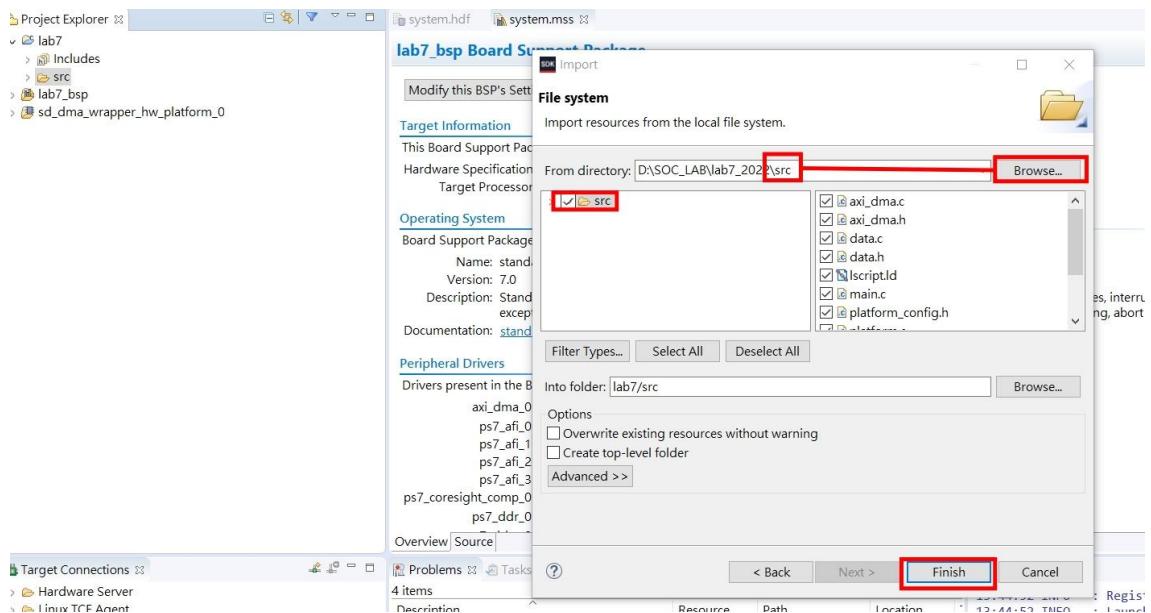


2. 選擇 File System



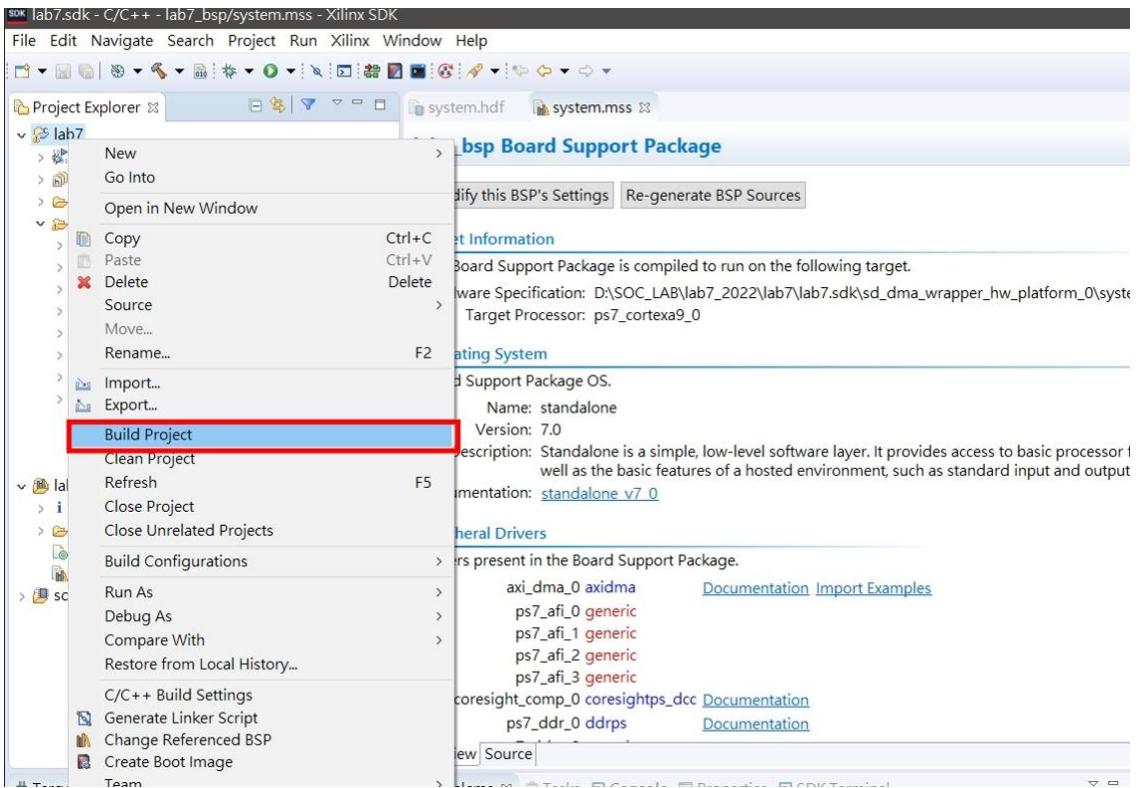
3. 將 lab7 資料夾中的 src 檔案整包 load 進來並選擇 Yes to overwrite 覆寫原有檔案.

*src 資料夾為本 Lab 所用之 C code, 包含 SD Card R/W function、DMA Transfer function.



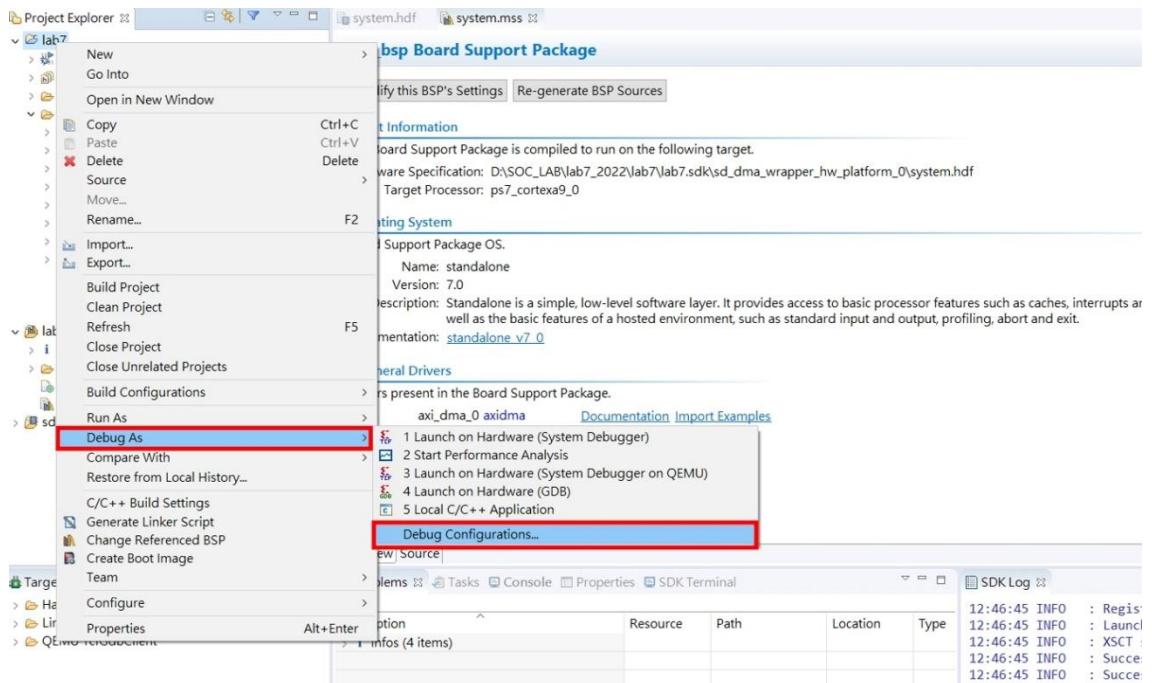
4. 對 Project Explorer 下的 lab7 右鍵點選 Build Project 重新 Compile 新增的檔案.

*此步驟若有 Error 表示 BSP 設定沒有新增齊全.



2.4 進行 Debug Mode

1. 對 Project Explorer 下的 lab7 右鍵點選 Debug As→Debug Configuration.

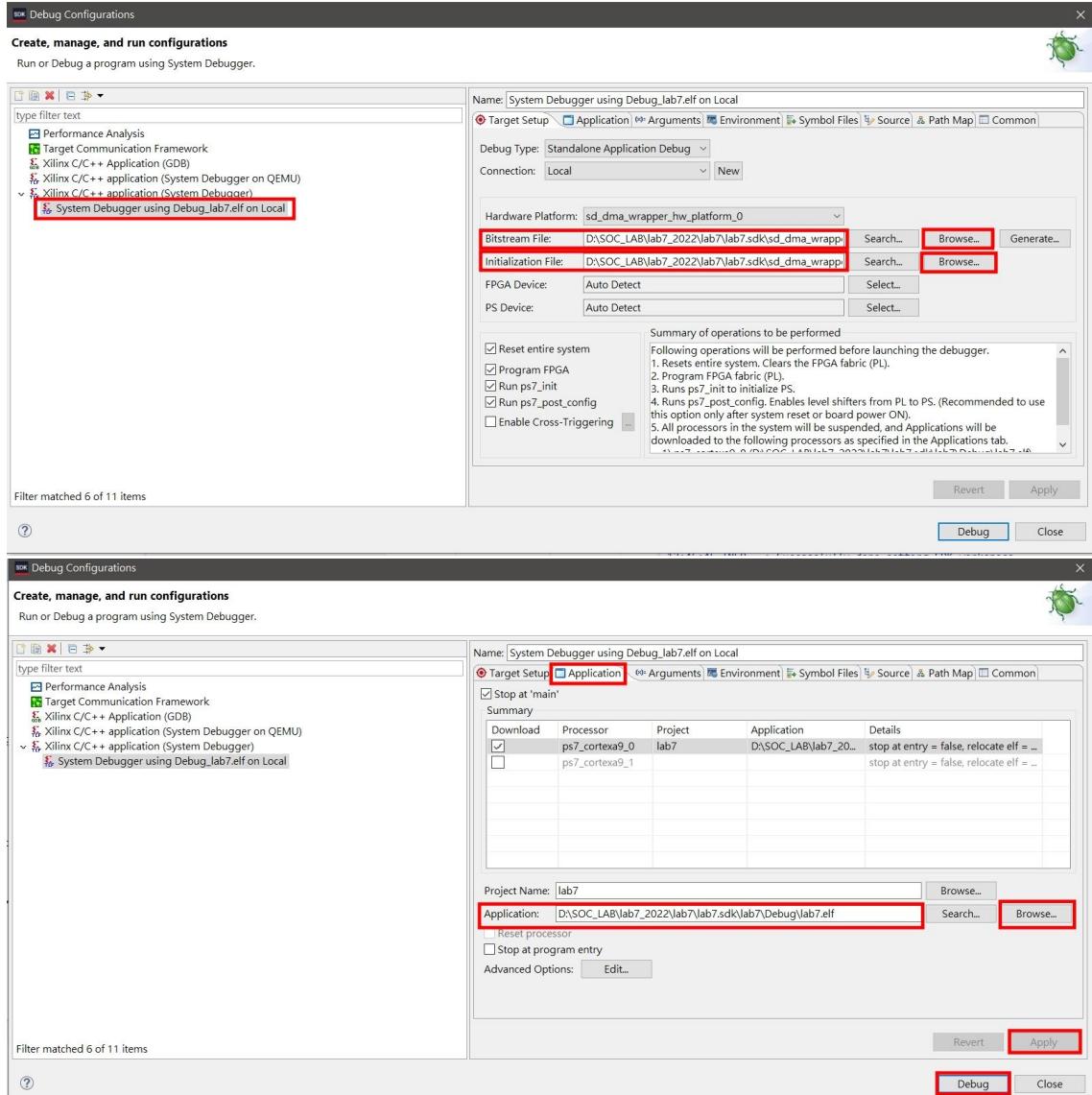


2. 於 Debug Configurations 窗格點選 System Debugger using Debug_lab7.elf on Local，接著以下步驟設定檔案絕對位置.

- Target Setup 欄位的 Bistream File，點選 Browse..找到此專案

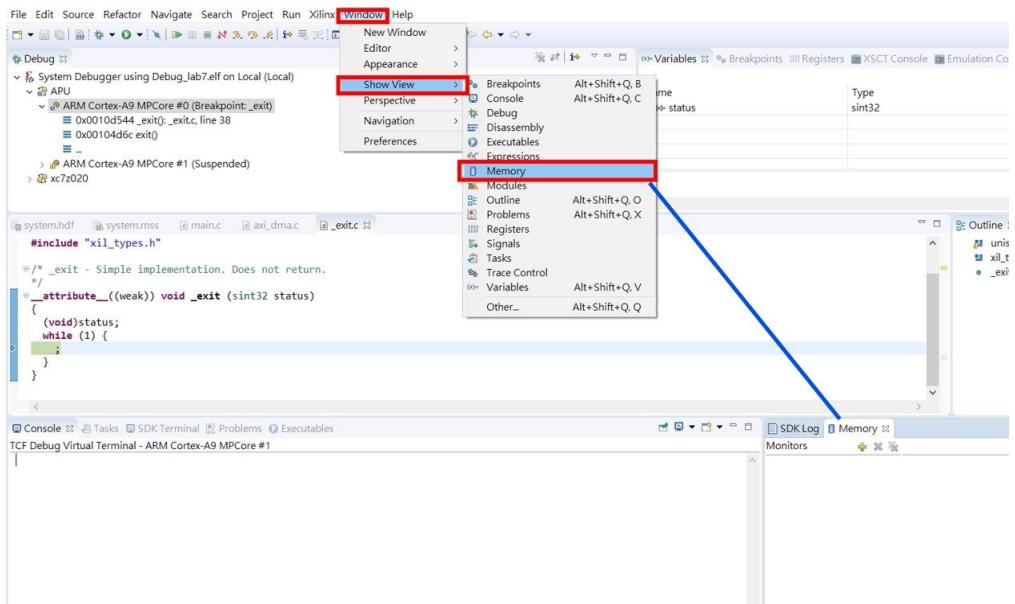
lab7.sdk→design_wrapper_hw_platform_0 下的.bit file.

- Target Setup 欄位的 Initialization File，點選 Browse..並與 Bistream File 同個位置下的.tcl file.
- Application 欄位的 Application，選擇 Debug 資料夾下的.elf file.



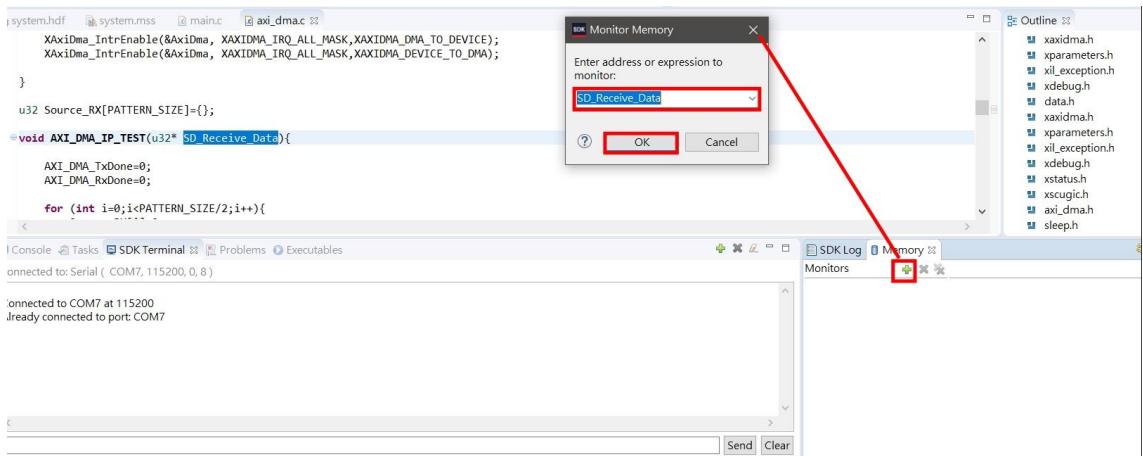
2.5 Debug 頁面基礎設定

1. Memory 監看窗格，Windows→Show View→Memory，於螢幕右下方看到 Memory Monitors，可新增多記憶體來監看。

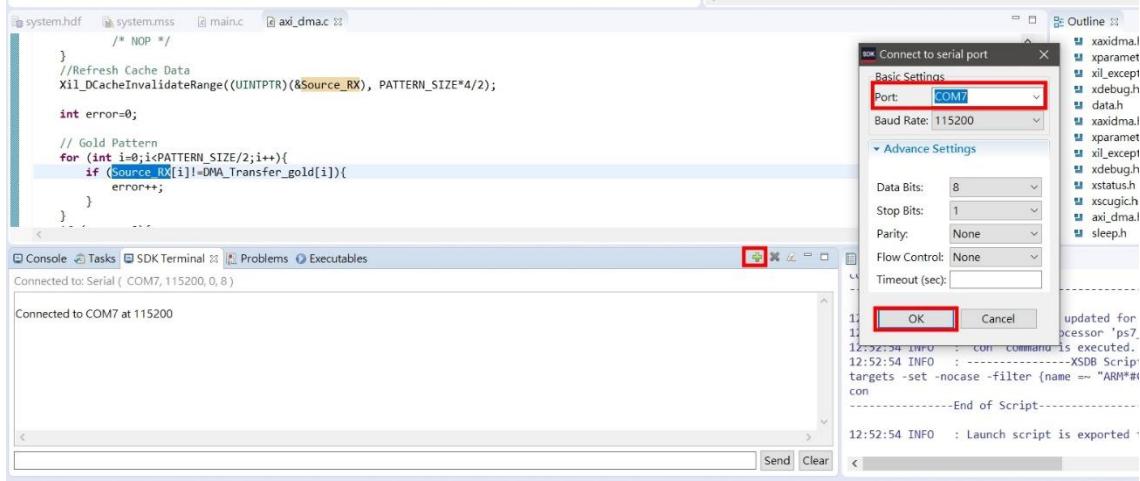


於 Memory 窗格加入要監看的記憶體空間，本 Lab 主要監看 3 個 Memory:

- 1.1、準備寫入 SD 卡的資料: SD_Transfer_Data
- 1.2、從 SD 卡讀取出的資料: SD_Receive_Data
- 1.3、DMA Transfer 後的資料: Source_RX

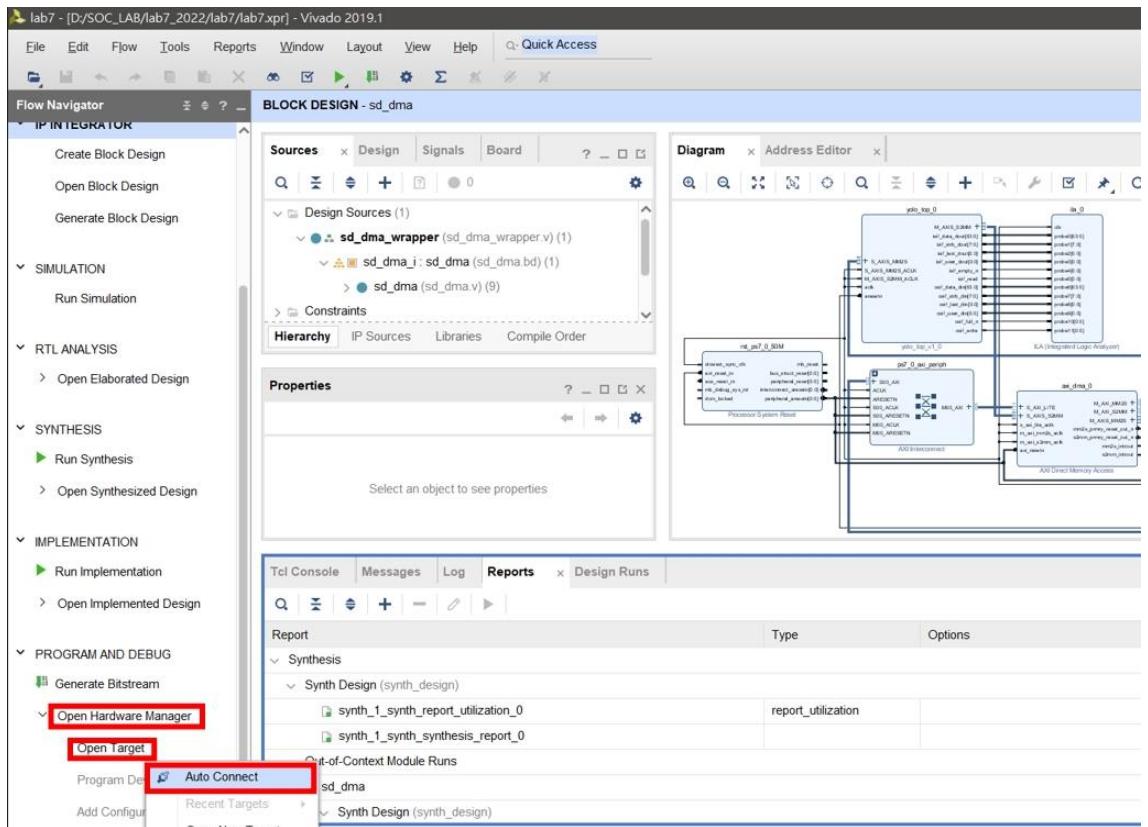


2. 於 SDK Terminal 設定與 Uart1 連接.

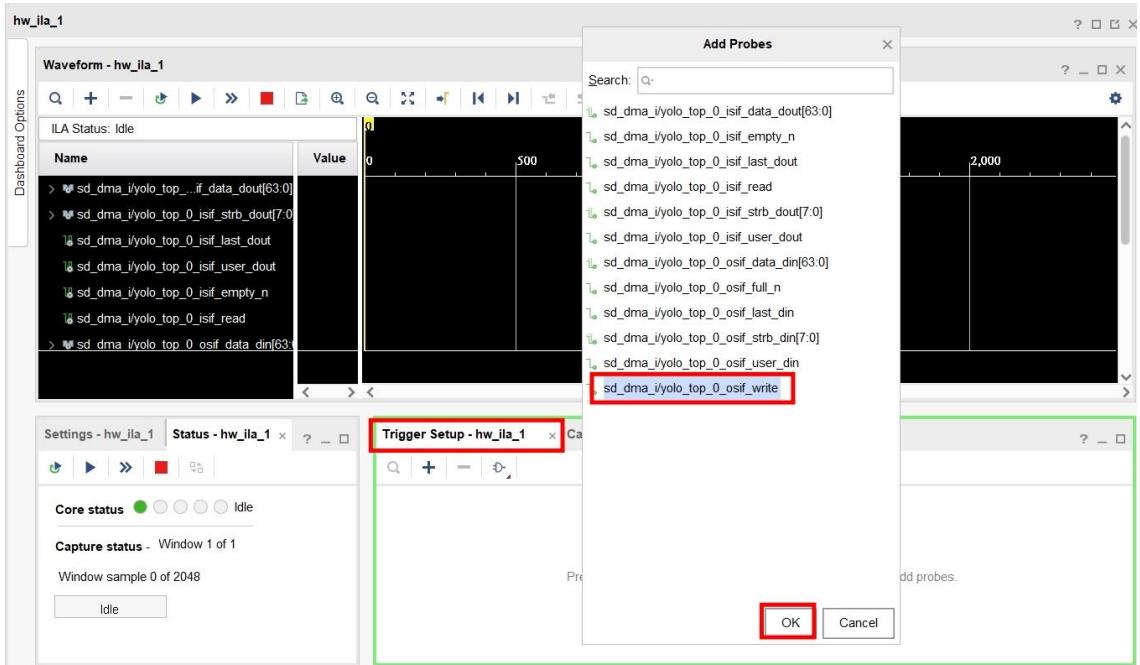


2.6 Vivado 端 Hardware Manager 設定

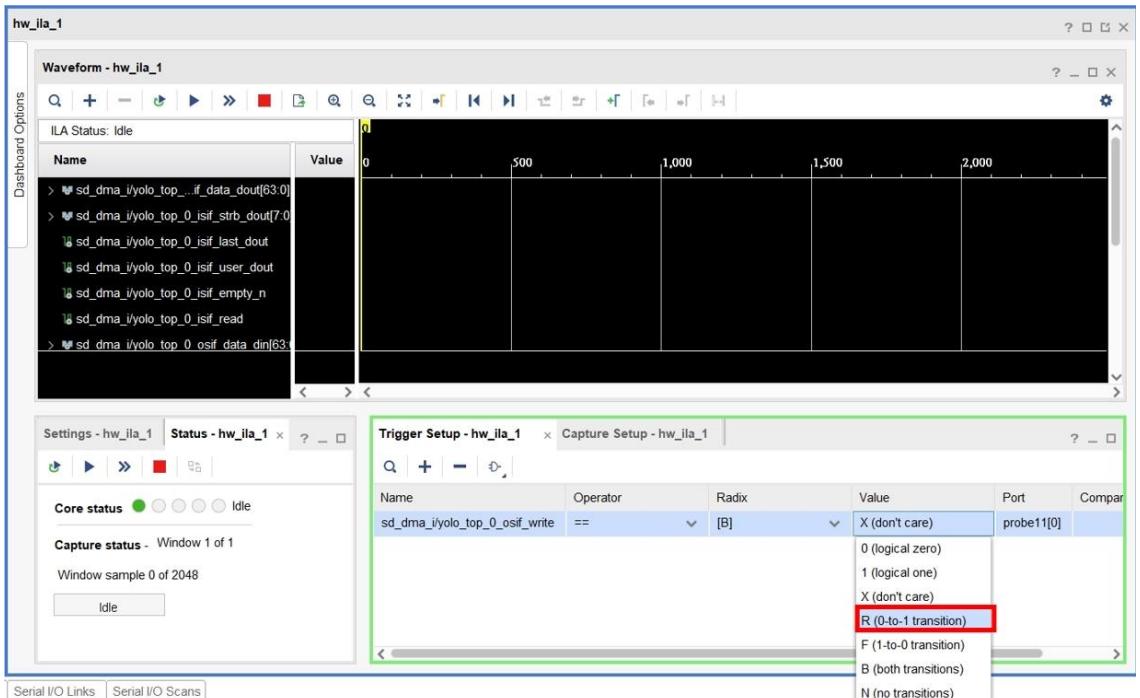
1. Flow Navigator 下的 Open Hardware Manager→Open Target→Auto Connect



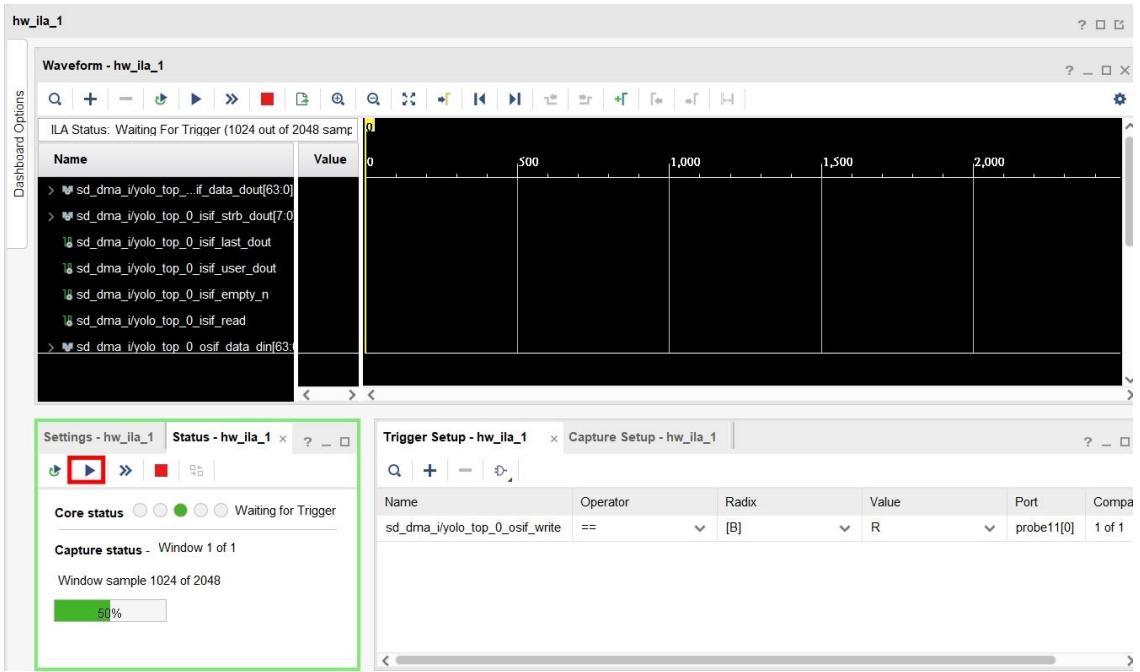
2. 在 Waveform 窗格中的 Trigger Setup 設定中斷採樣點，點選”+”並於 Add Probes 中選擇 osif_write 訊號



並於 Value 處選擇 R，表示當此訊號於 0 變 1 時 Trigger 開始採樣。

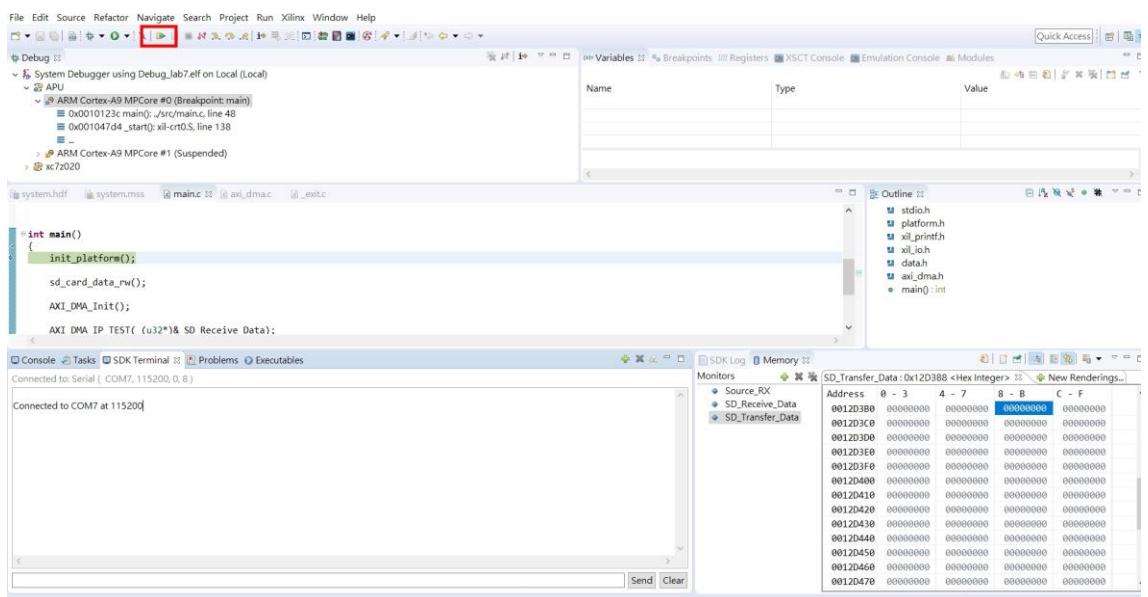


最後在 Status 窗格按下撥放鍵，以等待 SDK 那裏開始運行。



2.7 SDK Run Compile

1. 回到 SDK，並開始 Compile.



- Compile 結果於 SDK Terminal 可見 UART 回傳之執行情況，分為 SD 卡讀取與 DMA Transfer 行為，皆正常運行則會在 Terminal 回傳 Success!.

```

Connected to: Serial ( COM7, 115200, 0, 8 )
Connected to COM7 at 115200
--- SD Card reading ---
SD Card W/R Success!!
--- axi dma initial ---
DMA Transfer Success!!!
-- Data Check --
Addr 0, Rx: 2 , Gold: 2
Addr 1, Rx: 4 , Gold: 4
Addr 2, Rx: 10 , Gold: 10
Addr 3, Rx: 12 , Gold: 12
Addr 4, Rx: 18 , Gold: 18
Addr 5, Rx: 20 , Gold: 20
Addr 6, Rx: 26 , Gold: 26
Addr 7, Rx: 28 , Gold: 28
Addr 8, Rx: 34 , Gold: 34
Addr 9, Rx: 36 , Gold: 36
-- Done --

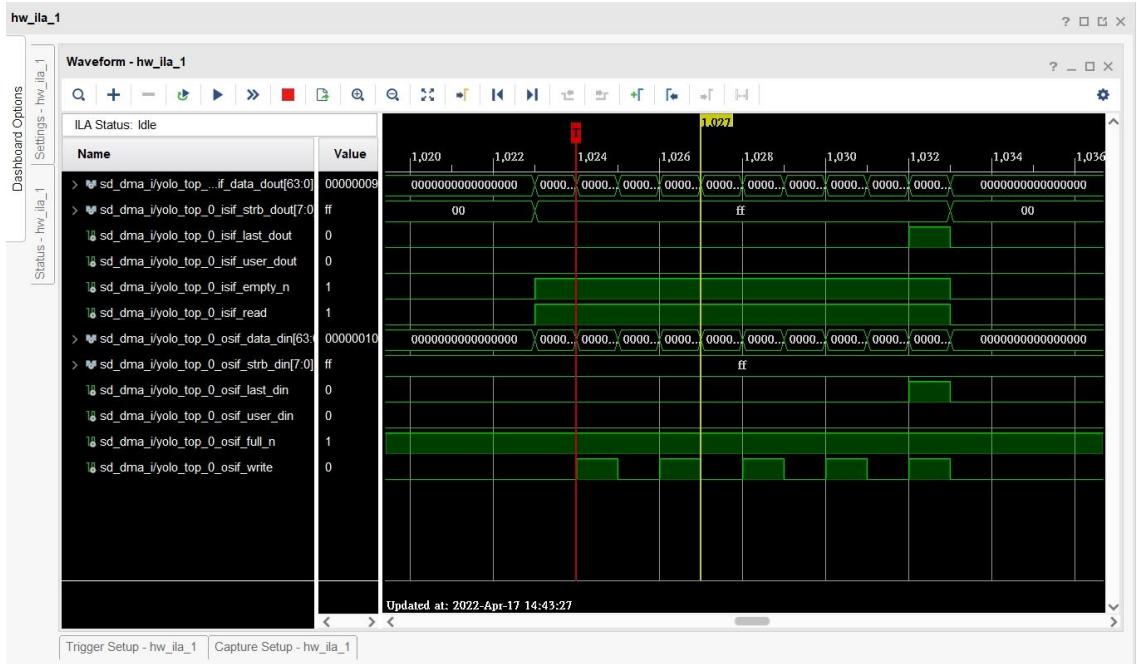
```

- Memory Monitors 處可看到紅色的值為經過 Compile 後刷新的值，可以比對資料於 SD_Transfer_Data 與 SD_Receive_Data_Source 三個記憶體空間內的值的運行狀況。

Source_RX: 0x12C798 <Hex Integer>

Address	0 - 3	4 - 7	8 - B	C - F
0012C760	00000000	00000000	00000000	00000000
0012C770	00000000	00000000	00000000	00000001
0012C780	00000001	00000000	00000020	0012397C
0012C790	11111111	00000000	00000002	00000004
0012C7A0	0000000A	0000000C	00000012	00000014
0012C7B0	0000001A	0000001C	00000022	00000024
0012C7C0	00000000	00000000	00000000	00000000
0012C7D0	00000000	00000000	00000000	00000000
0012C7E0	00000000	00000000	00000000	00000000
0012C7F0	00000000	00000000	00000000	00000000
0012C800	00000000	00000000	00000000	00000000

- 最後可於 Vivado 端的 Hardware Manager 看到執行的 Waveform，如行為不如預期可藉由此方法監看訊號的行為。



3. 附錄

SD Card [FatFs API] http://elm-chan.org/fsw/ff/00index_e.html