Vitis Training Lab

Agenda

- **Lab 1: Get data from PL BRAM through AXI Bus**
  - Vivado Block Design
  - Vitis code result of execution
- **Lab 2: Use High-Level API Driver to drive AXI GPIO (with ILA)**
  - Vivado Block Design
  - Vitis code result of execution
  - ILA result

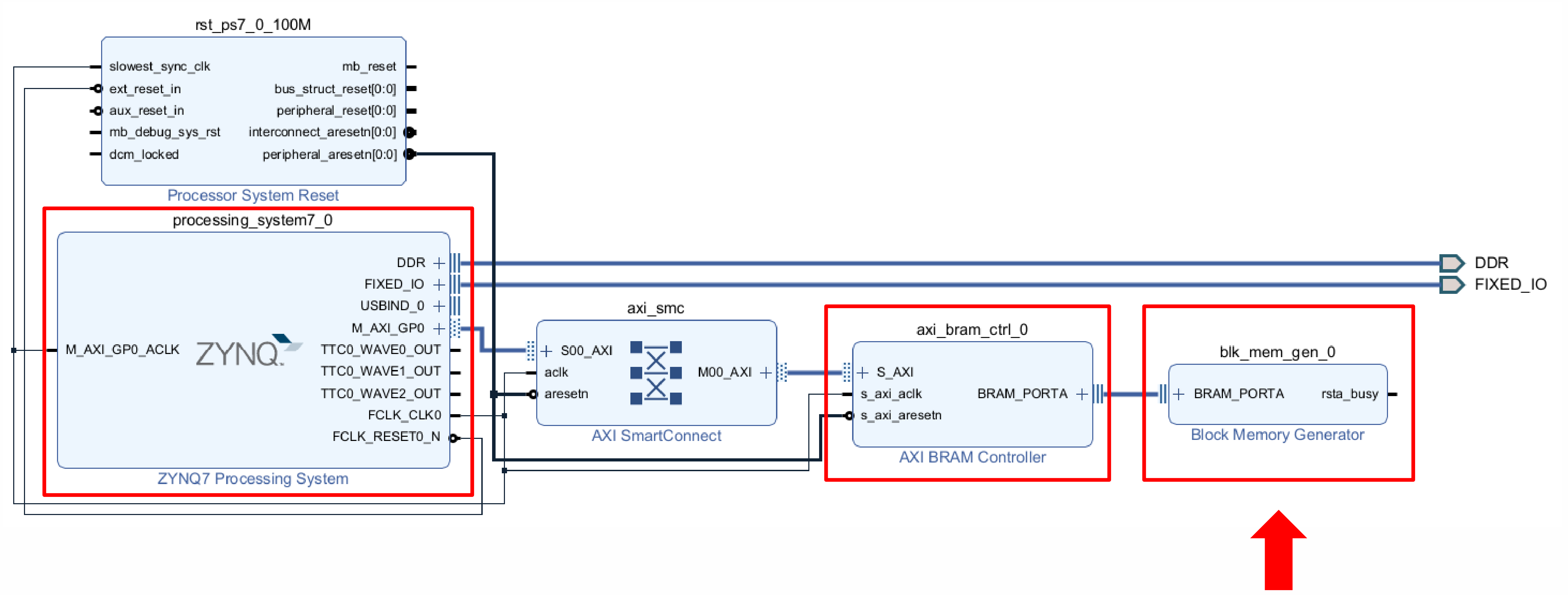
Lab 1: Get data from PL BRAM through AXI Bus

Vivado Part

Vivado Block Design

- Block Design

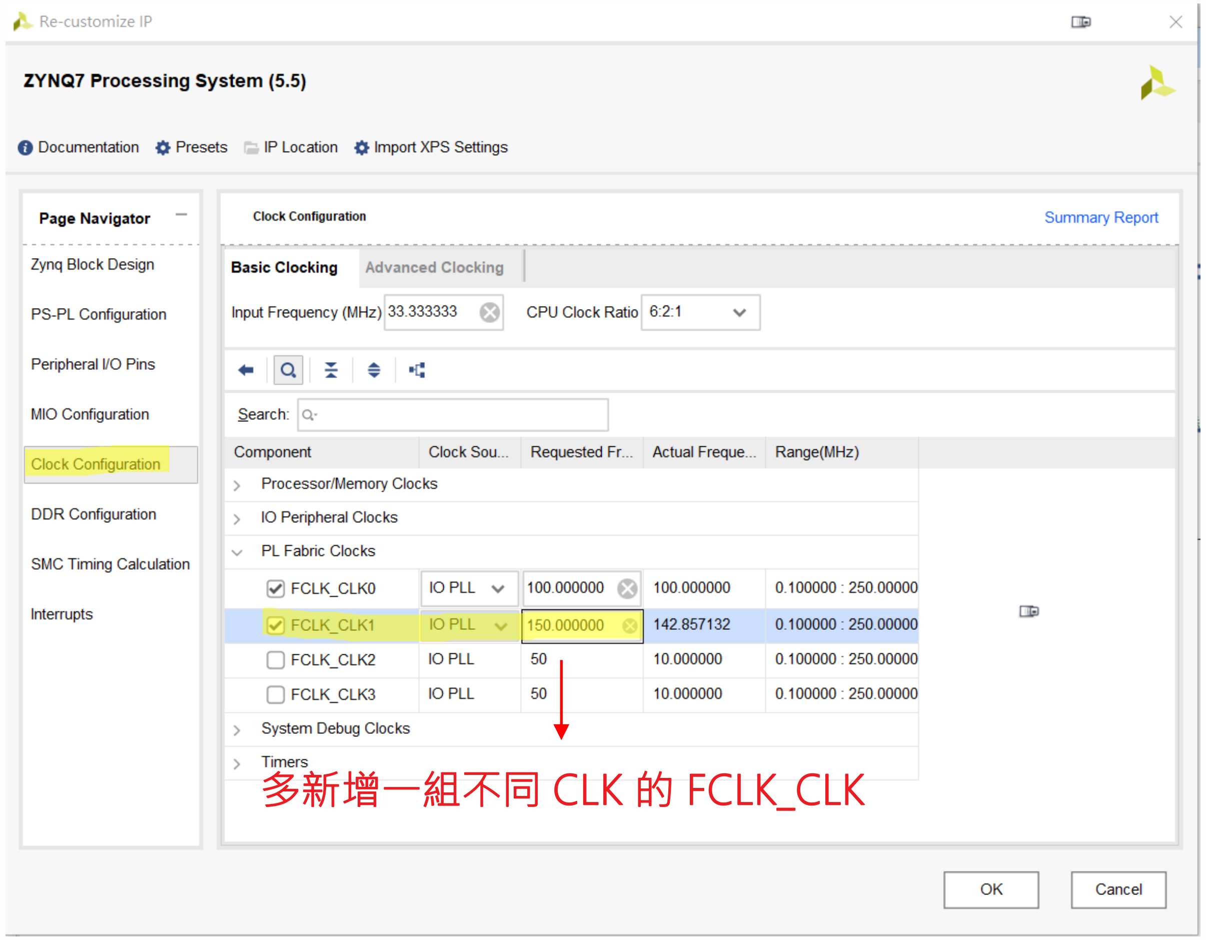
Designer Assistance available. Run Connection Automation



XILINX.

Vivado Block Design

- Question: 若 PS 端的工作頻率與 AXI BRAM Controller 不同怎辦？
- Answer: 打開 PS 端的自定義，如下



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

Clock Configuration — Summary Report

Basic Clocking Advanced Clocking

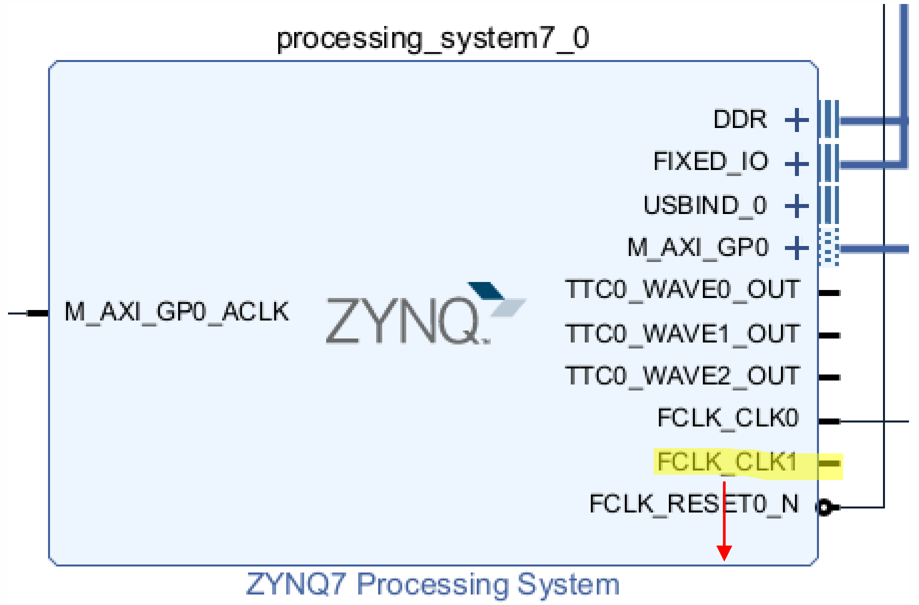
Input Frequency (MHz) 33.333333   CPU Clock Ratio 6:2:1

Search:

Component	Clock Sou...	Requested Fr...	Actual Freque...	Range(MHz)
› Processor/Memory Clocks				
› IO Peripheral Clocks				
⌄ PL Fabric Clocks				
✓ FCLK_CLK0	IO PLL	100.000000	100.000000	0.100000 : 250.00000
✓ FCLK_CLK1	IO PLL	150.000000	142.857132	0.100000 : 250.00000
☐ FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.00000
☐ FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.00000
› System Debug Clocks				
› Timers				

OK
Cancel

多新增一組不同 CLK 的 FCLK_CLK

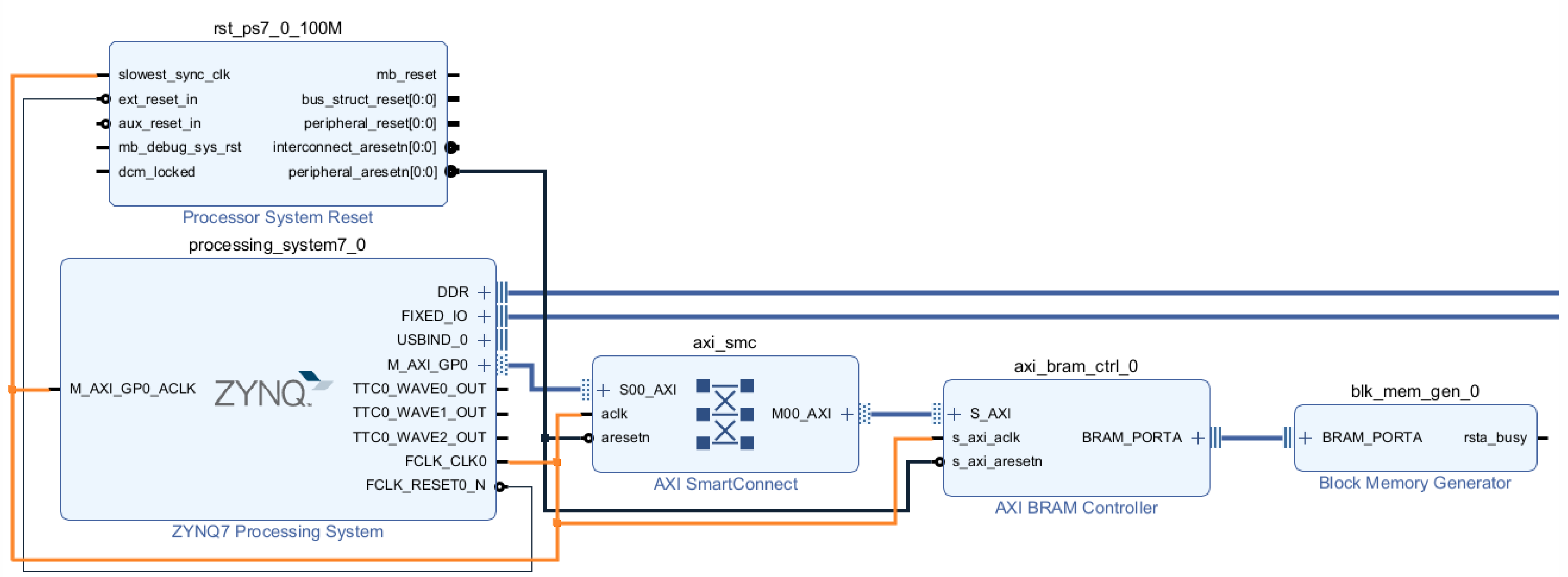


新增後的 CLK 會出現在 PS 的 pin 腳上，可用於連接至不同 CLK 的元件

XILINX

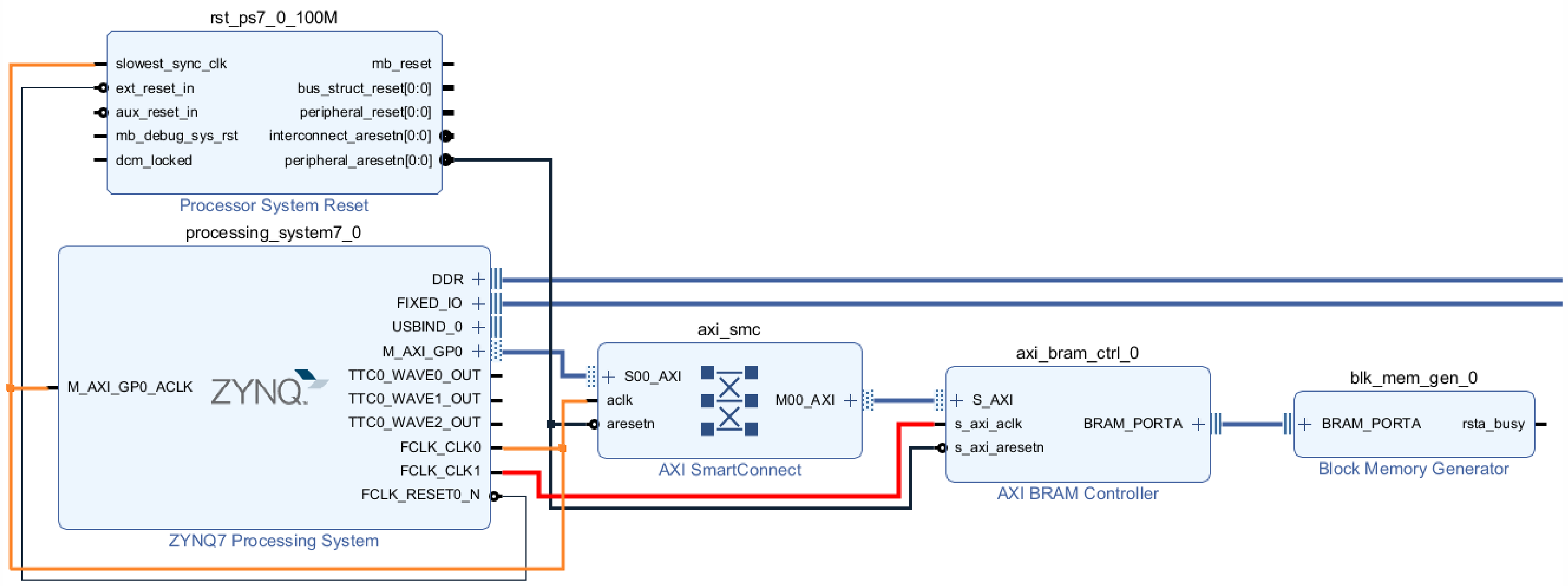
Vivado Block Design

- Question: 若 PS 端的工作頻率與 AXI BRAM Controller 不同怎辦？
- Answer: 比較圖 — 同頻率



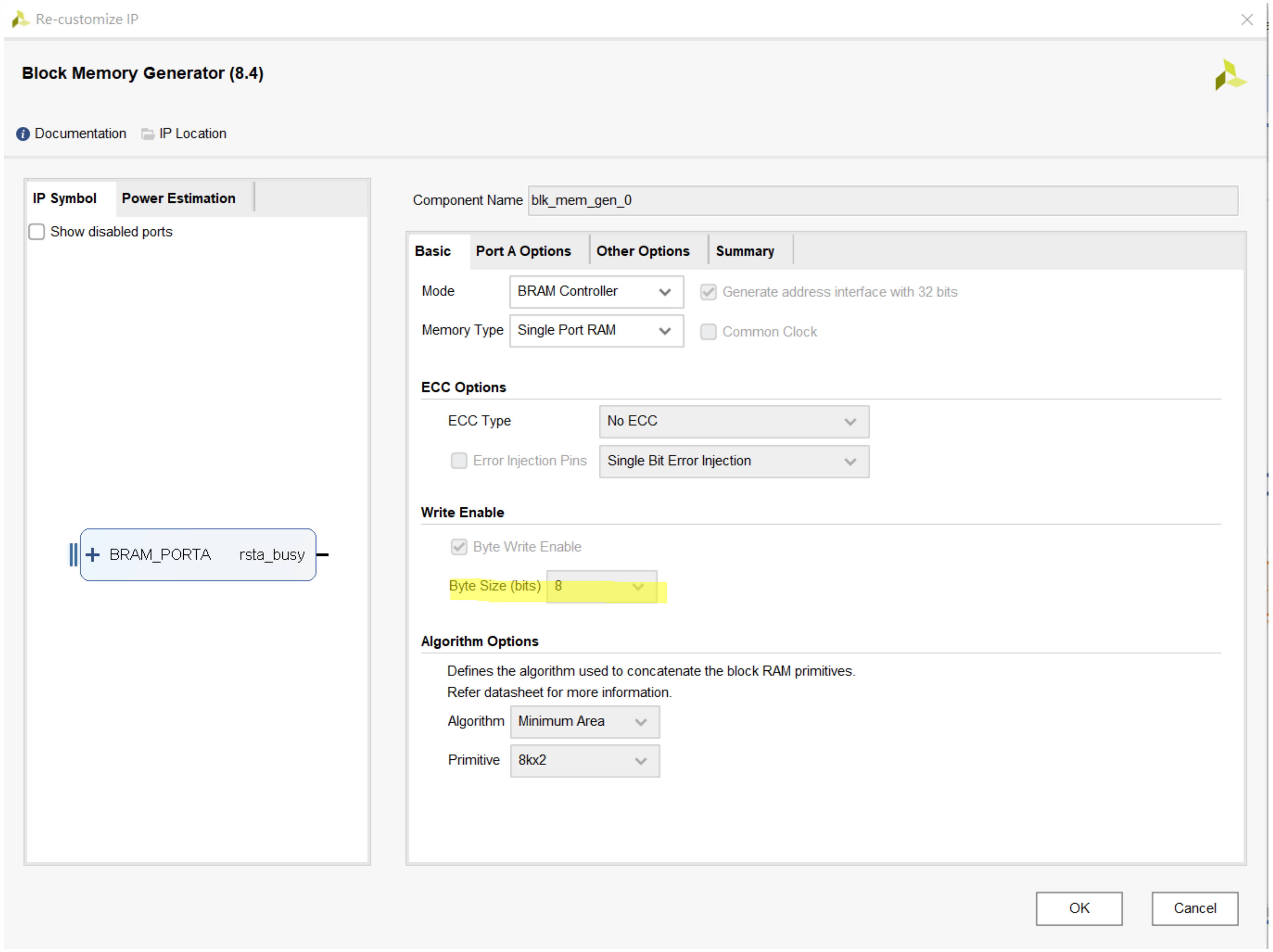
Vivado Block Design

- Question: 若 PS 端的工作頻率與 AXI BRAM Controller 不同怎辦？
- Answer: 比較圖 — 不同頻率



Vivado Block Design

- Block Design



Re-customize IP

Block Memory Generator (8.4)

Documentation   IP Location

Show disabled ports

Component Name blk_mem_gen_0

Basic | Port A Options | Other Options | Summary

Mode: BRAM Controller
Generate address interface with 32 bits [✓]
Memory Type: Single Port RAM
Common Clock [ ]

ECC Options

ECC Type: No ECC
Error Injection Pins [ ]: Single Bit Error Injection

Write Enable

Byte Write Enable [✓]
Byte Size (bits): 8

Algorithm Options

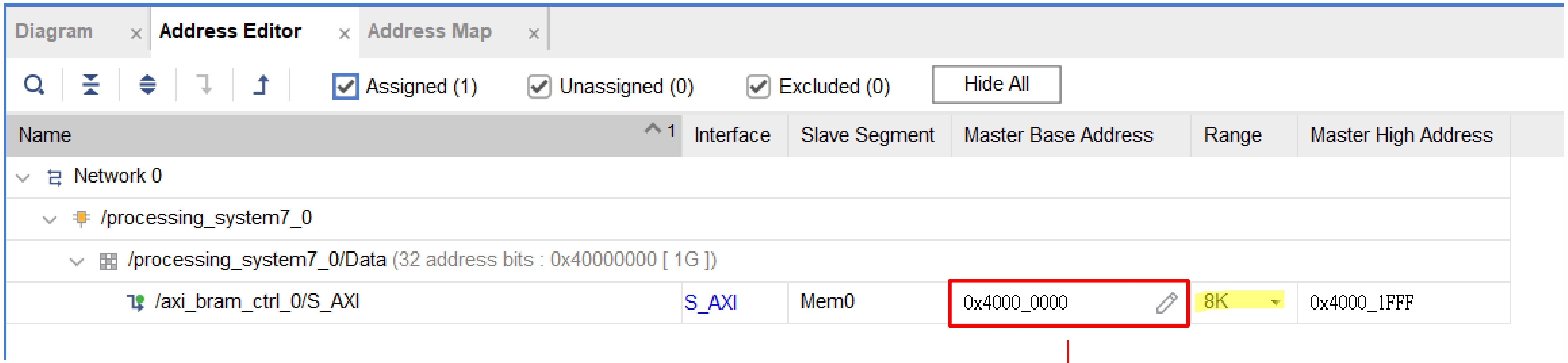
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area
Primitive: 8kx2

OK   Cancel

Vivado Block Design

- Block Design

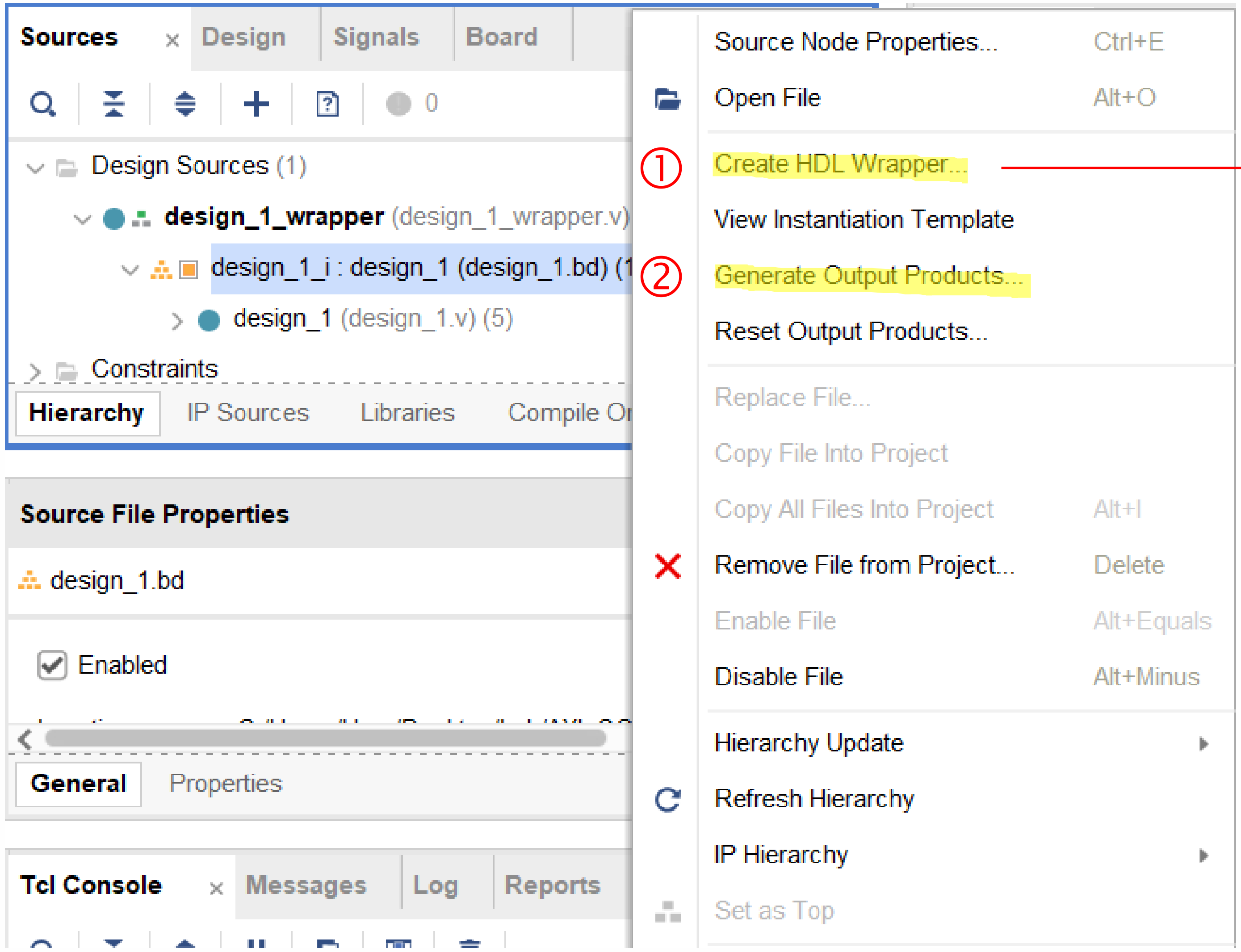


Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x4000_0000	8K	0x4000_1FFF

初始地址若由 **Vivado** 自動連線的話則會被自動 **assign**，可提供使用者進行修改，但必須依照該開發板的 **address map** 進行範圍內的修改

Vivado Block Design

- Export XSA file



Sources × | Design | Signals | Board

0

Design Sources (1)

design_1_wrapper (design_1_wrapper.v)

design_1_i : design_1 (design_1.bd) (1

design_1 (design_1.v) (5)

Constraints

Hierarchy | IP Sources | Libraries | Compile Or

Source File Properties

design_1.bd

☑ Enabled

General | Properties

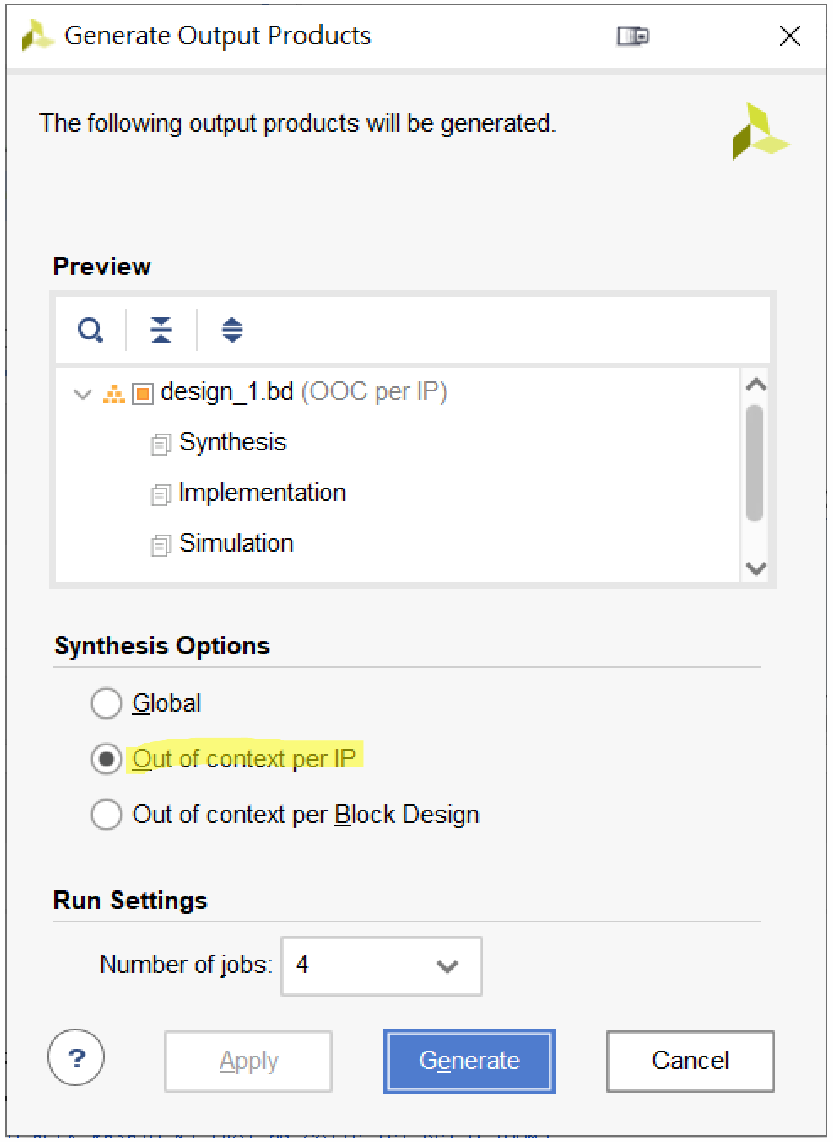
Tcl Console × | Messages | Log | Reports

Source Node Properties...	Ctrl+E
Open File	Alt+O
① Create HDL Wrapper...	
View Instantiation Template	
② Generate Output Products...	
Reset Output Products...	
Replace File...	
Copy File Into Project	
Copy All Files Into Project	Alt+I
Remove File from Project...	Delete
Enable File	Alt+Equals
Disable File	Alt+Minus
Hierarchy Update	
Refresh Hierarchy	
IP Hierarchy	
Set as Top	

$\longrightarrow$ Generate Verilog files

Vivado Block Design

- Export XSA file



Generate Output Products

The following output products will be generated.

Preview

design_1.bd (OOC per IP)
- Synthesis
- Implementation
- Simulation

Synthesis Options

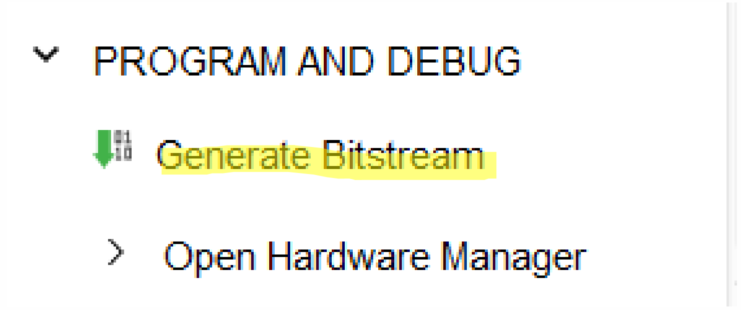
( ) Global
(●) Out of context per IP
( ) Out of context per Block Design

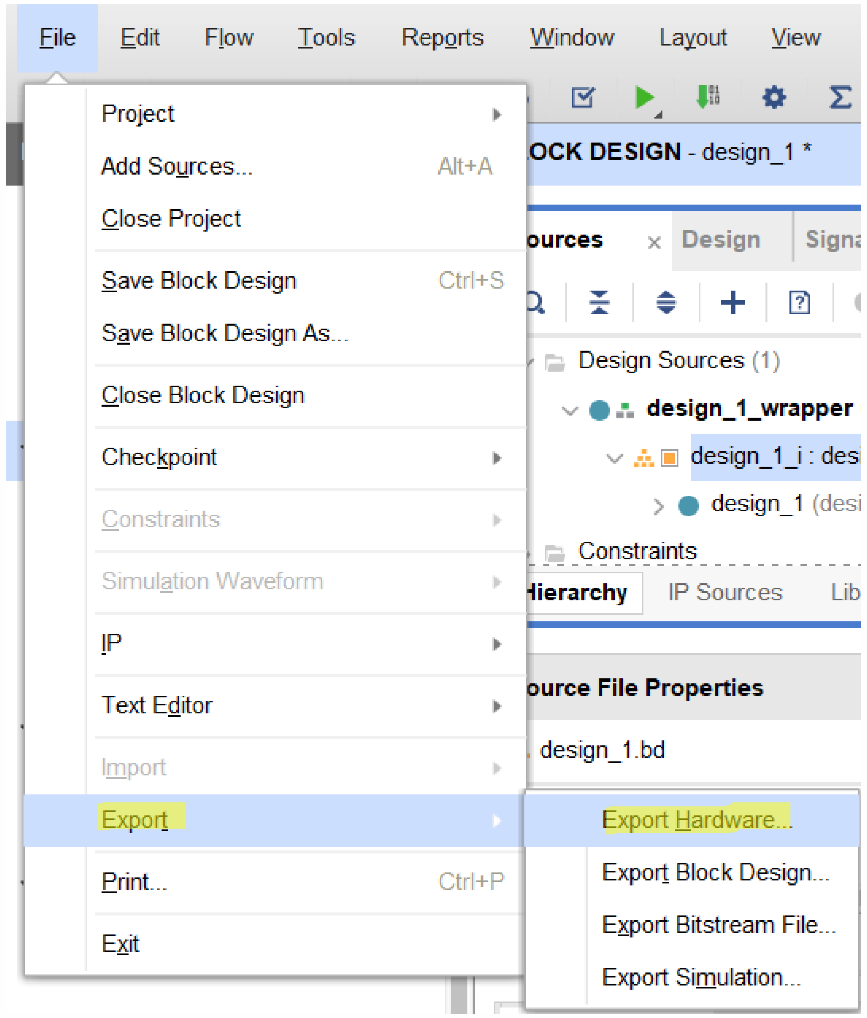
Run Settings

Number of jobs: 4

Vivado Block Design

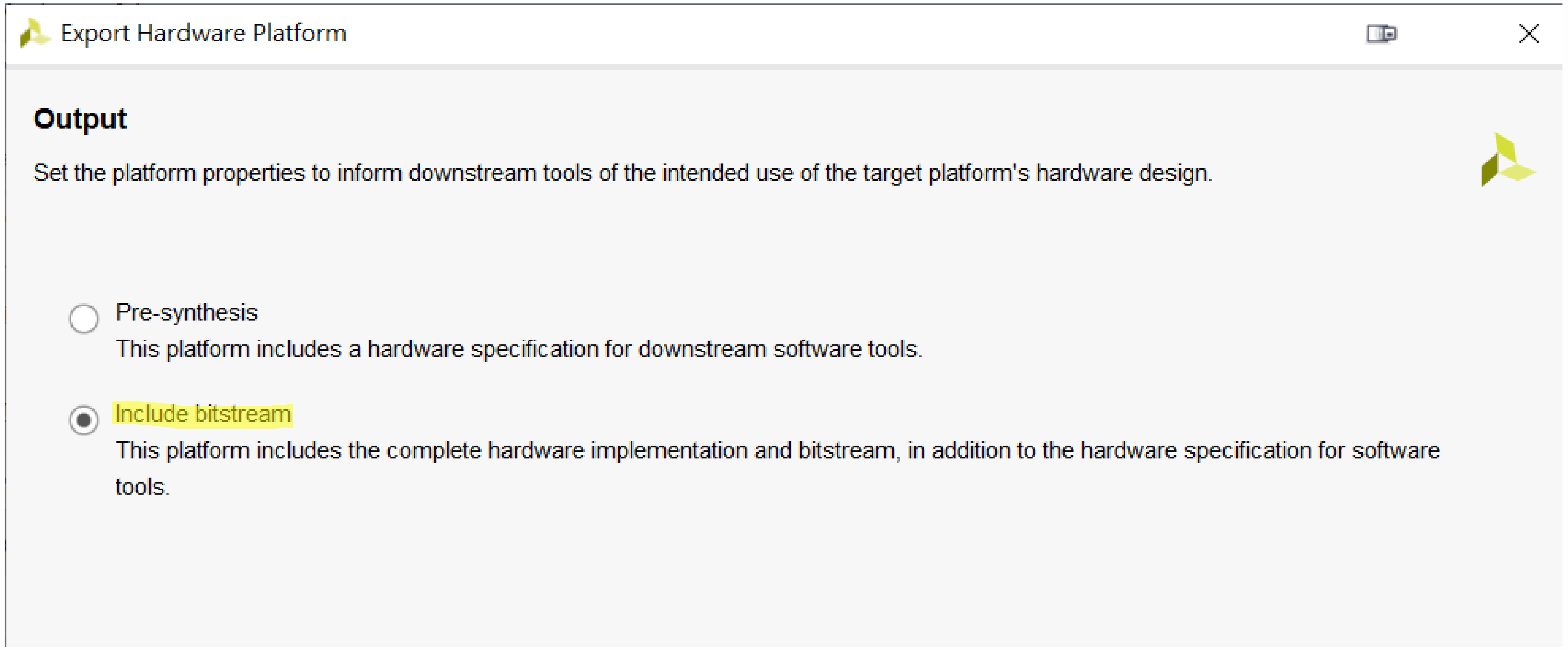
- Export XSA file





Vivado Block Design

- Export XSA file



Export Hardware Platform

Output

Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

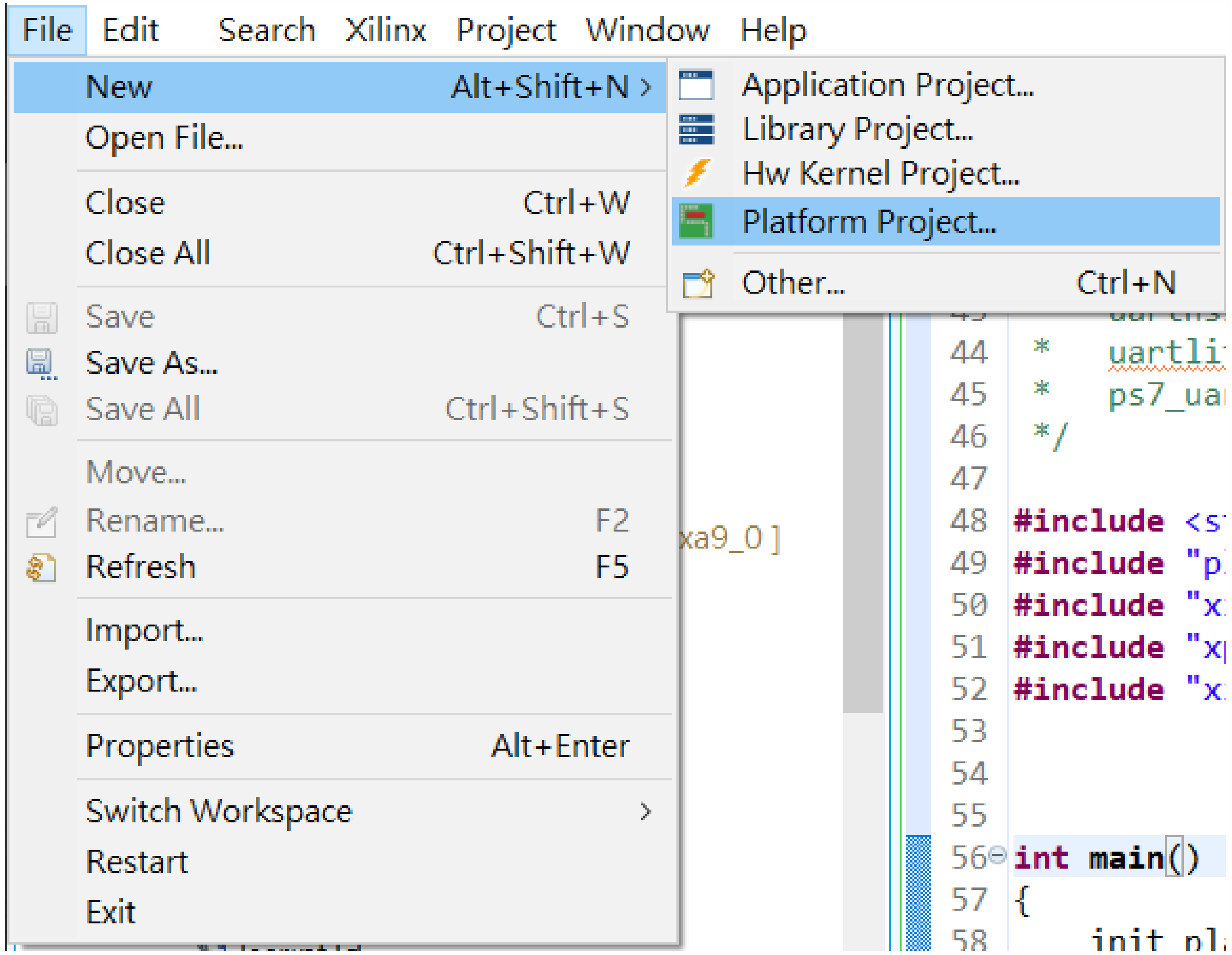
○ Pre-synthesis
This platform includes a hardware specification for downstream software tools.

● Include bitstream
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

Vitis Part

Vitis code result of execution

- Create Platform



File Edit Search Xilinx Project Window Help

New	Alt+Shift+N >
Open File...	
Close	Ctrl+W
Close All	Ctrl+Shift+W
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
Move...	
Rename...	F2
Refresh	F5
Import...	
Export...	
Properties	Alt+Enter
Switch Workspace	>
Restart	
Exit	

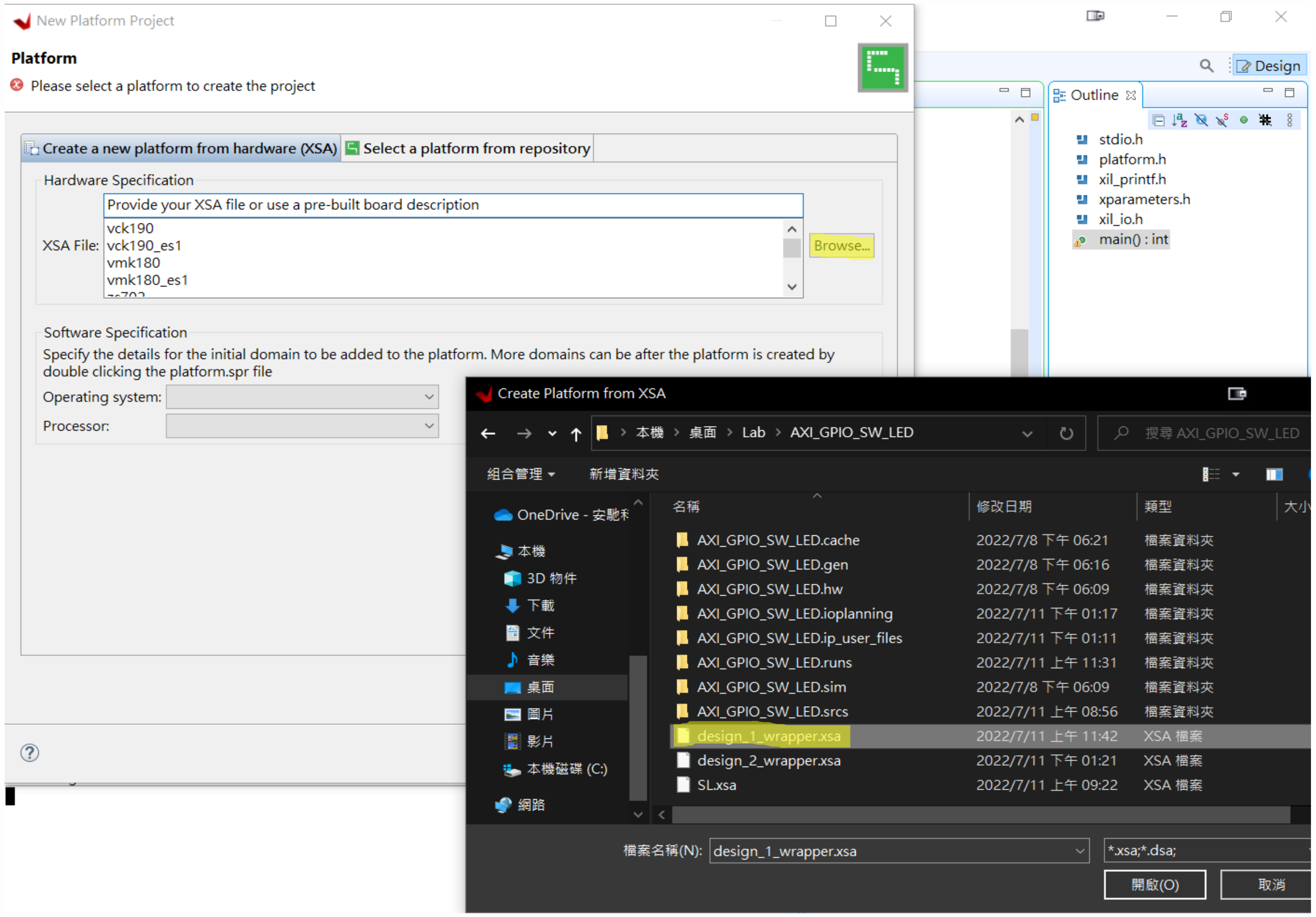
Application Project...	
Library Project...	
Hw Kernel Project...	
Platform Project...	
Other...	Ctrl+N

```
 *    uartli
 *    ps7_ua
 */

#include <s
#include "p
#include "x
#include "x
#include "x



int main()
{
    init_pl
```

XILINX

Vitis code result of execution

- Create Platform



New Platform Project

Platform

Please select a platform to create the project

Create a new platform from hardware (XSA) | Select a platform from repository

Hardware Specification

XSA File: Provide your XSA file or use a pre-built board description

vck190
vck190_es1
vmk180
vmk180_es1

Software Specification

Specify the details for the initial domain to be added to the platform. More domains can be after the platform is created by double clicking the platform.spr file

Operating system:

Processor:

Outline: stdio.h, platform.h, xil_printf.h, xparameters.h, xil_io.h, main() : int

Create Platform from XSA

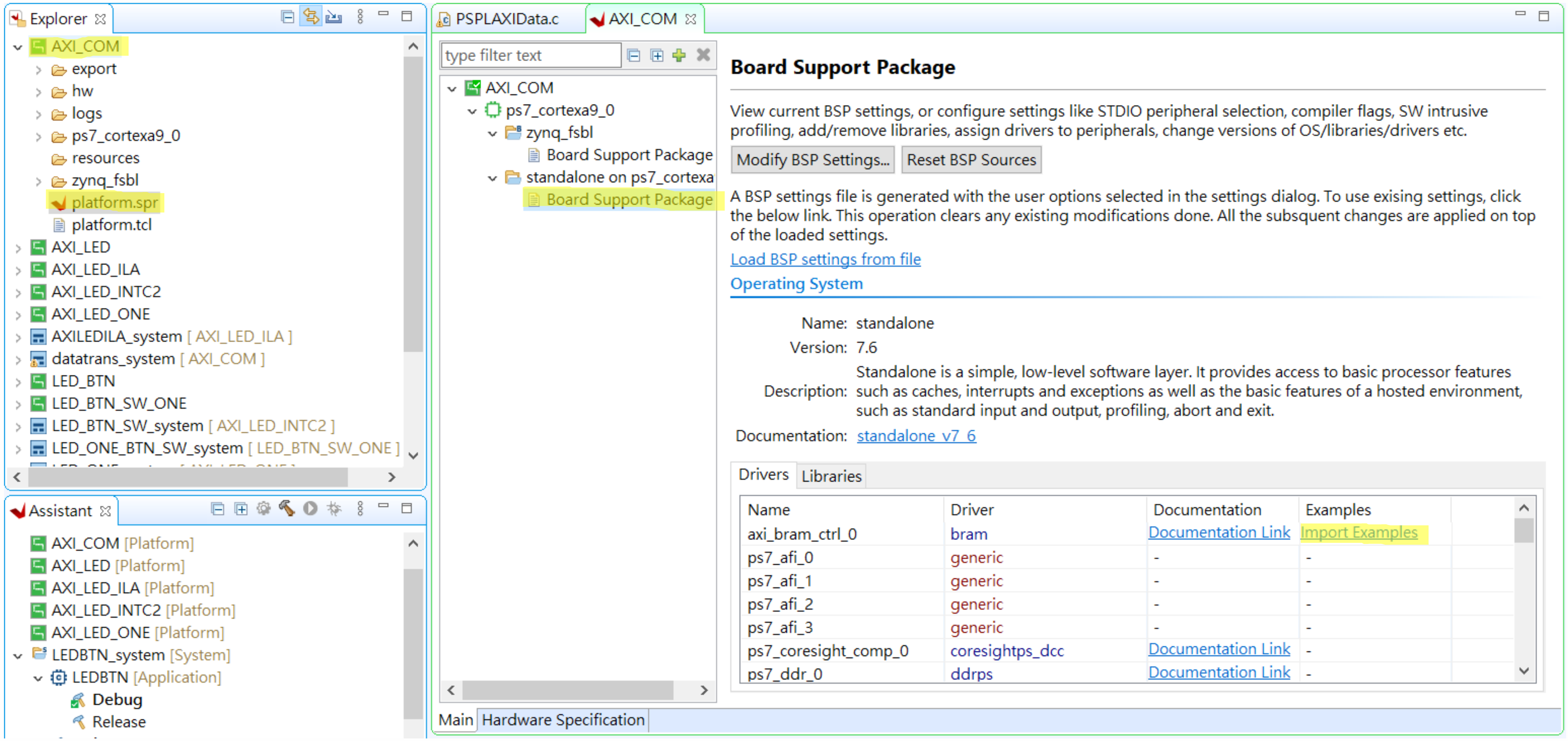
本機 > 桌面 > Lab > AXI_GPIO_SW_LED

名稱	修改日期	類型
AXI_GPIO_SW_LED.cache	2022/7/8 下午 06:21	檔案資料夾
AXI_GPIO_SW_LED.gen	2022/7/8 下午 06:16	檔案資料夾
AXI_GPIO_SW_LED.hw	2022/7/8 下午 06:09	檔案資料夾
AXI_GPIO_SW_LED.ioplanning	2022/7/11 下午 01:17	檔案資料夾
AXI_GPIO_SW_LED.ip_user_files	2022/7/11 下午 01:11	檔案資料夾
AXI_GPIO_SW_LED.runs	2022/7/11 上午 11:31	檔案資料夾
AXI_GPIO_SW_LED.sim	2022/7/8 下午 06:09	檔案資料夾
AXI_GPIO_SW_LED.srcs	2022/7/11 上午 08:56	檔案資料夾
design_1_wrapper.xsa	2022/7/11 上午 11:42	XSA 檔案
design_2_wrapper.xsa	2022/7/11 下午 01:21	XSA 檔案
SL.xsa	2022/7/11 上午 09:22	XSA 檔案

檔案名稱(N): design_1_wrapper.xsa   *.xsa;*.dsa;

Vitis code result of execution

- Create Application – Import example from Board Support Package



Explorer

- AXI_COM
  - export
  - hw
  - logs
  - ps7_cortexa9_0
  - resources
  - zynq_fsbl
  - platform.spr
  - platform.tcl
- AXI_LED
- AXI_LED_ILA
- AXI_LED_INTC2
- AXI_LED_ONE
- AXILEDILA_system [AXI_LED_ILA]
- datatrans_system [AXI_COM]
- LED_BTN
- LED_BTN_SW_ONE
- LED_BTN_SW_system [AXI_LED_INTC2]
- LED_ONE_BTN_SW_system [LED_BTN_SW_ONE]

Assistant

- AXI_COM [Platform]
- AXI_LED [Platform]
- AXI_LED_ILA [Platform]
- AXI_LED_INTC2 [Platform]
- AXI_LED_ONE [Platform]
- LEDBTN_system [System]
  - LEDBTN [Application]
    - Debug
    - Release

PSPLAXIData.c | AXI_COM

type filter text

- AXI_COM
  - ps7_cortexa9_0
    - zynq_fsbl
      - Board Support Package
    - standalone on ps7_cortexa
      - Board Support Package

Board Support Package

View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

Modify BSP Settings... | Reset BSP Sources

A BSP settings file is generated with the user options selected in the settings dialog. To use exising settings, click the below link. This operation clears any existing modifications done. All the subsquent changes are applied on top of the loaded settings.

Load BSP settings from file

Operating System

Name:	standalone
Version:	7.6
Description:	Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.
Documentation:	standalone_v7_6

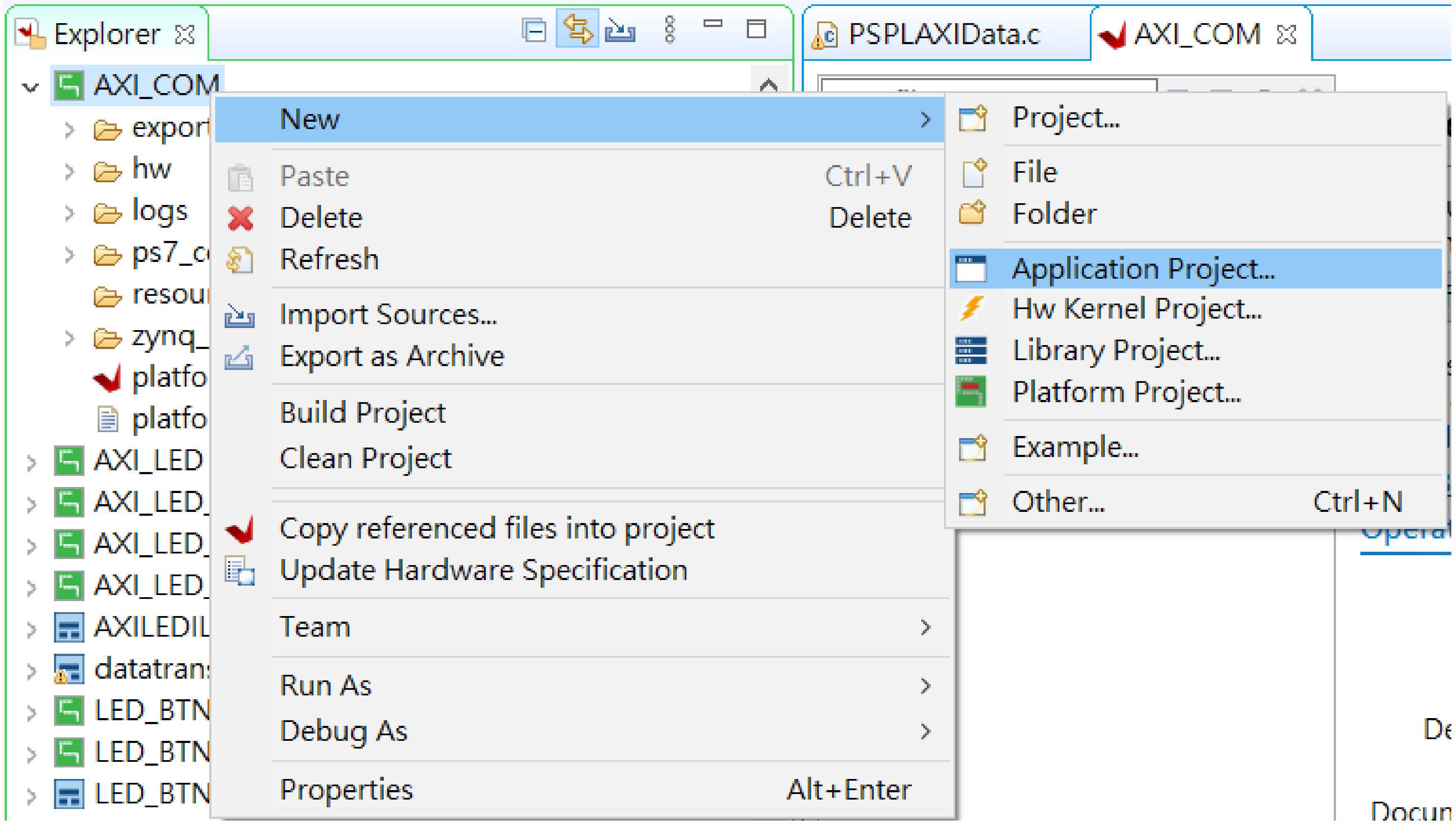
Drivers | Libraries

Name	Driver	Documentation	Examples
axi_bram_ctrl_0	bram	Documentation Link	Import Examples
ps7_afi_0	generic	-	-
ps7_afi_1	generic	-	-
ps7_afi_2	generic	-	-
ps7_afi_3	generic	-	-
ps7_coresight_comp_0	coresightps_dcc	Documentation Link	-
ps7_ddr_0	ddrps	Documentation Link	-

Main | Hardware Specification

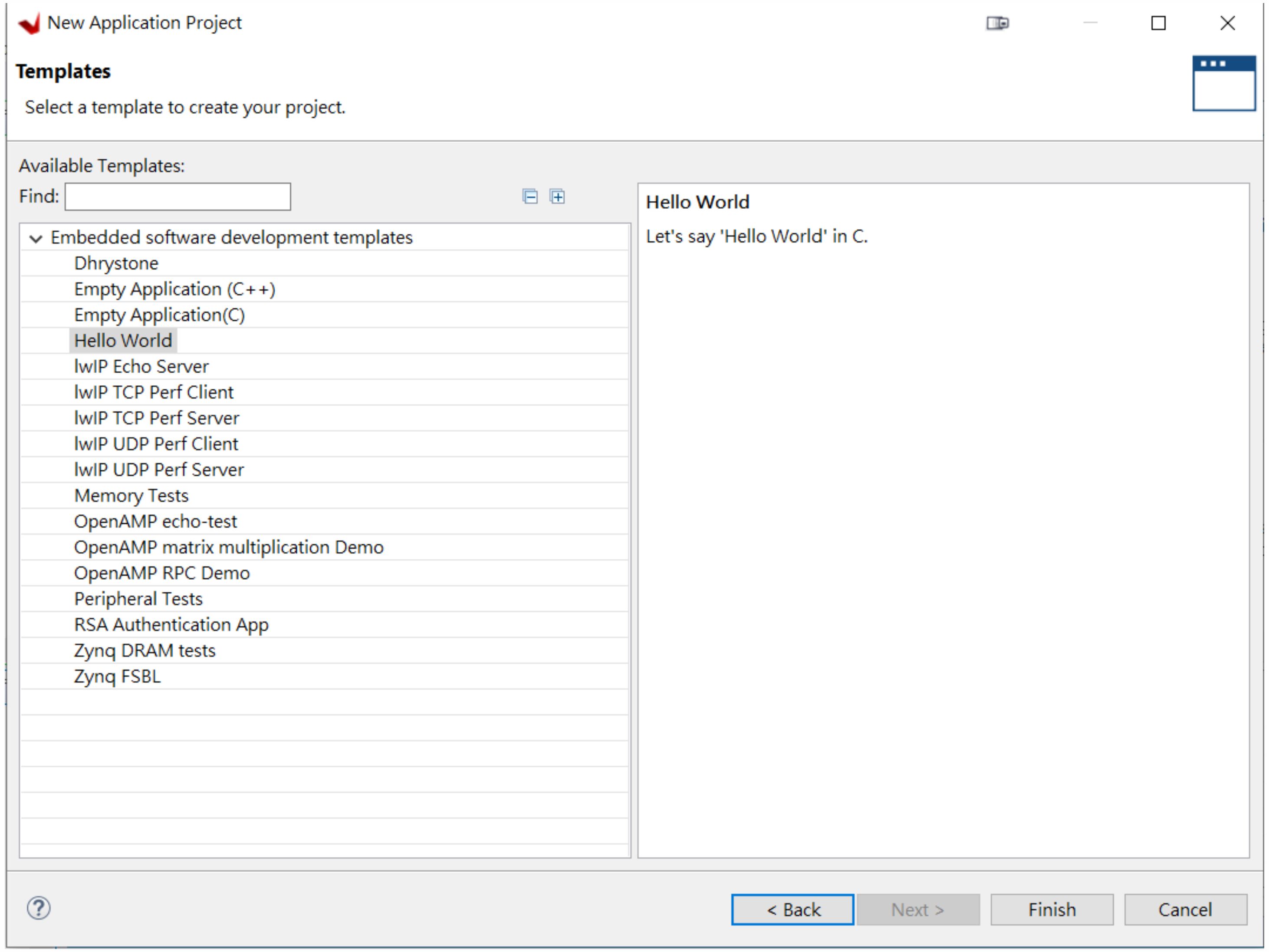
Vitis code result of execution

- Create Application – Create a new Application Project



Vitis code result of execution

- Create Application – Import example from Application project



New Application Project

Templates

Select a template to create your project.

Available Templates:

Find:

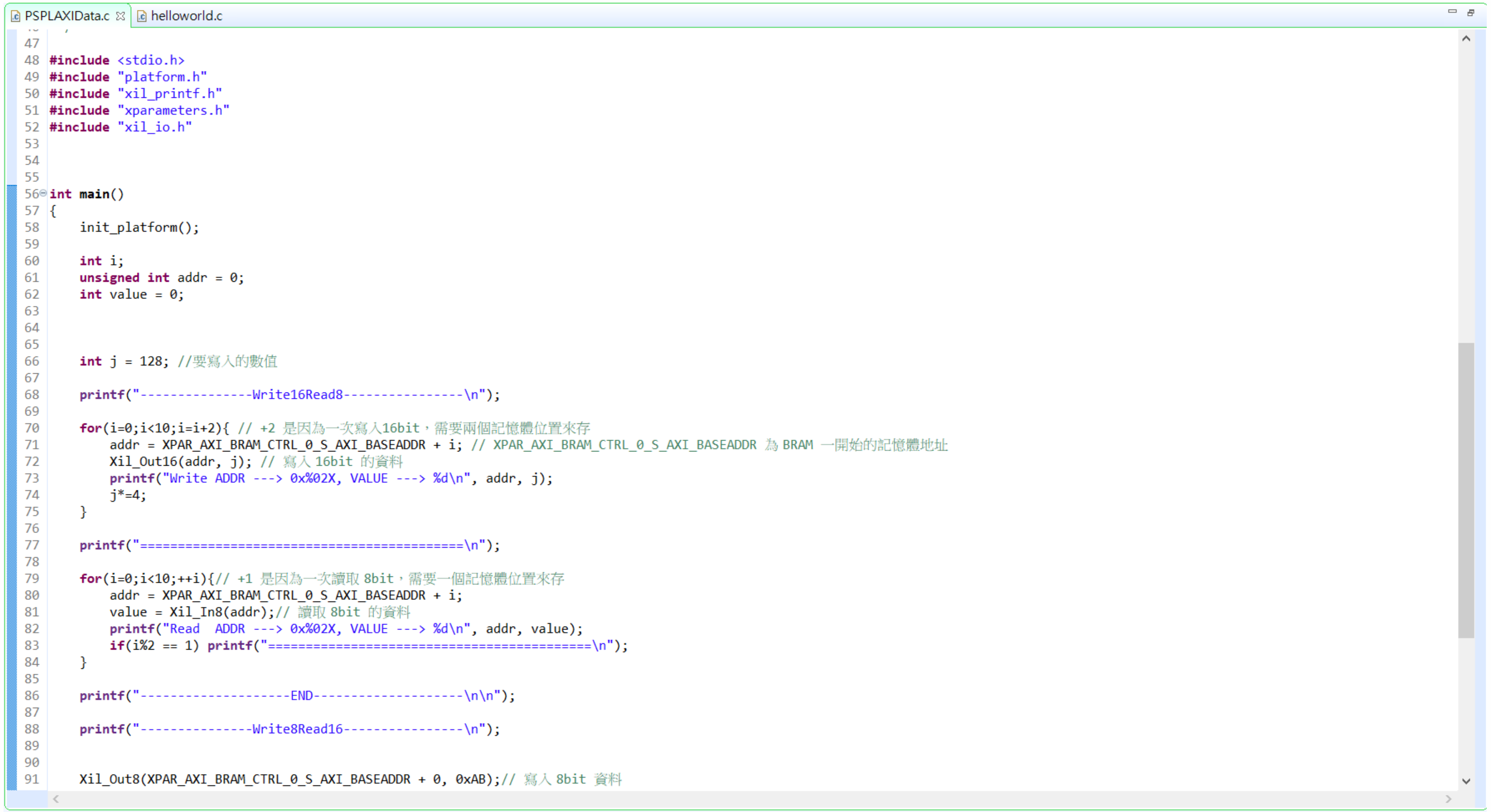
- Embedded software development templates
  - Dhrystone
  - Empty Application (C++)
  - Empty Application(C)
  - Hello World
  - lwIP Echo Server
  - lwIP TCP Perf Client
  - lwIP TCP Perf Server
  - lwIP UDP Perf Client
  - lwIP UDP Perf Server
  - Memory Tests
  - OpenAMP echo-test
  - OpenAMP matrix multiplication Demo
  - OpenAMP RPC Demo
  - Peripheral Tests
  - RSA Authentication App
  - Zynq DRAM tests
  - Zynq FSBL

Hello World

Let's say 'Hello World' in C.

Vitis code result of execution

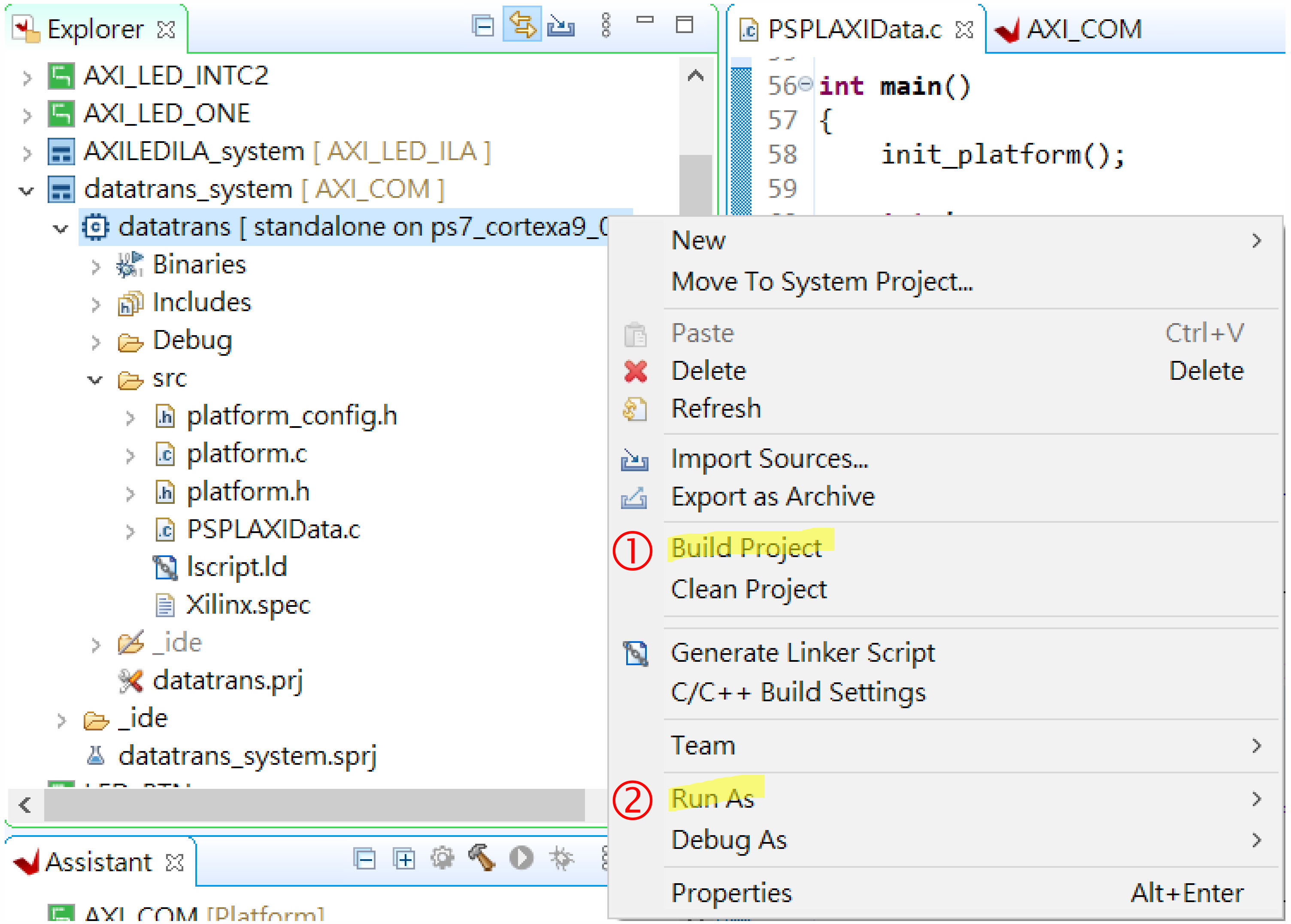
- Open Vitis to review the code


PSPLAXIData.c helloworld.c

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xil_io.h"



int main()
{
    init_platform();

    int i;
    unsigned int addr = 0;
    int value = 0;



    int j = 128; //要寫入的數值

    printf("---------------Write16Read8----------------\n");

    for(i=0;i<10;i=i+2){ // +2 是因為一次寫入16bit，需要兩個記憶體位置來存
        addr = XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i; // XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR 為 BRAM 一開始的記憶體地址
        Xil_Out16(addr, j); // 寫入 16bit 的資料
        printf("Write ADDR ---> 0x%02X, VALUE ---> %d\n", addr, j);
        j*=4;
    }

    printf("===========================================\n");

    for(i=0;i<10;++i){// +1 是因為一次讀取 8bit，需要一個記憶體位置來存
        addr = XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i;
        value = Xil_In8(addr);// 讀取 8bit 的資料
        printf("Read  ADDR ---> 0x%02X, VALUE ---> %d\n", addr, value);
        if(i%2 == 1) printf("===========================================\n");
    }

    printf("--------------------END--------------------\n\n");

    printf("---------------Write8Read16----------------\n");


    Xil_Out8(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + 0, 0xAB);// 寫入 8bit 資料
```

Vitis code result of execution

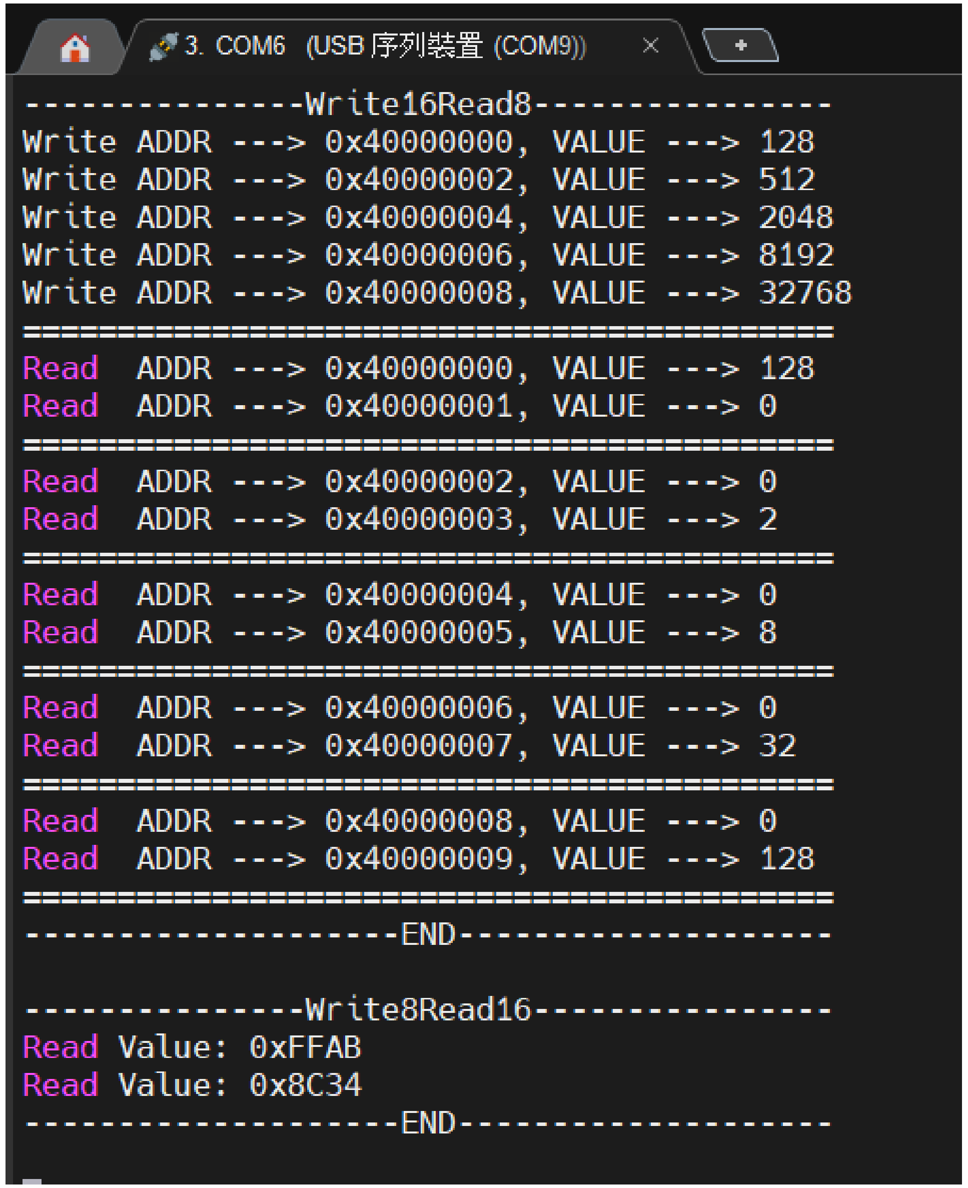
- Build the Project and Run



```
int main()
{
    init_platform();
```

Vitis code result of execution

- Result on MobaXterm



```
---------------Write16Read8---------------
Write ADDR ---> 0x40000000, VALUE ---> 128
Write ADDR ---> 0x40000002, VALUE ---> 512
Write ADDR ---> 0x40000004, VALUE ---> 2048
Write ADDR ---> 0x40000006, VALUE ---> 8192
Write ADDR ---> 0x40000008, VALUE ---> 32768
===========================================
Read  ADDR ---> 0x40000000, VALUE ---> 128
Read  ADDR ---> 0x40000001, VALUE ---> 0
===========================================
Read  ADDR ---> 0x40000002, VALUE ---> 0
Read  ADDR ---> 0x40000003, VALUE ---> 2
===========================================
Read  ADDR ---> 0x40000004, VALUE ---> 0
Read  ADDR ---> 0x40000005, VALUE ---> 8
===========================================
Read  ADDR ---> 0x40000006, VALUE ---> 0
Read  ADDR ---> 0x40000007, VALUE ---> 32
===========================================
Read  ADDR ---> 0x40000008, VALUE ---> 0
Read  ADDR ---> 0x40000009, VALUE ---> 128
===========================================
--------------------END--------------------

---------------Write8Read16---------------
Read Value: 0xFFAB
Read Value: 0x8C34
--------------------END--------------------
```

Vitis code result of execution

- volatile

某變數被抓取時必須都是最新的狀態，並從該變數的地址取值，而不是直接從Cache抓取資料時，必須命為volatile

e.g., 兩CPU對某一變數都會進行修改，但CPU取該變數數值時必須為最新的值，而不是抓上次存取至Cache的值，因此每次都必須從該變數的地址取值

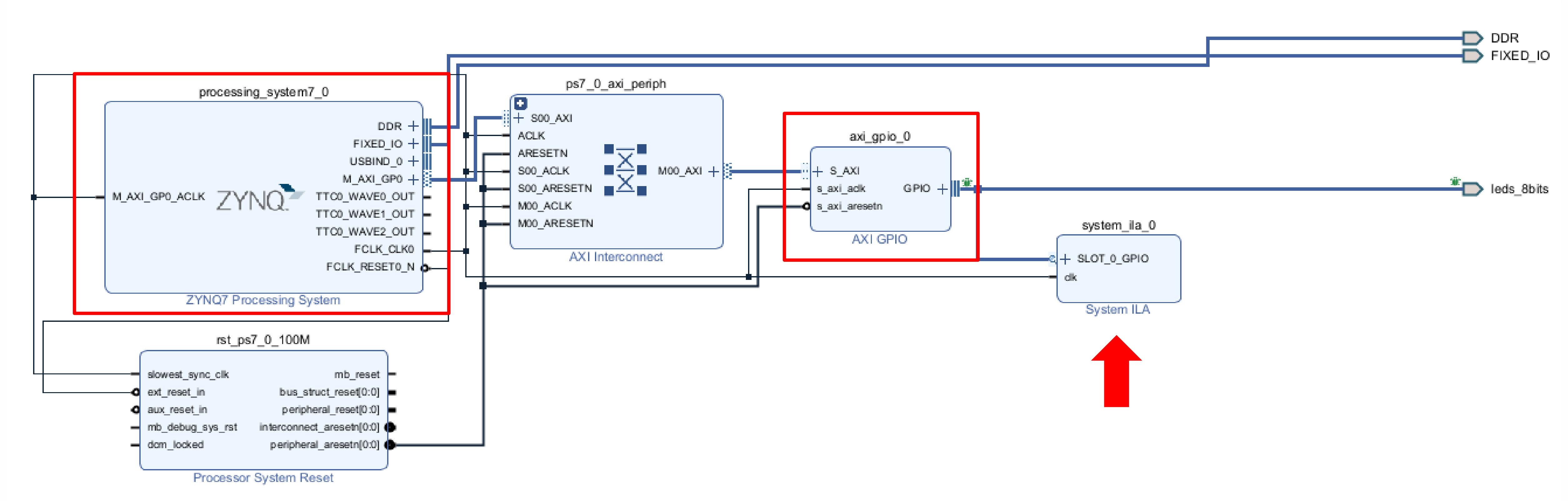
Lab 2: Use High-Level API Driver to drive AXI GPIO (with ILA)

XILINX.

Vivado Part

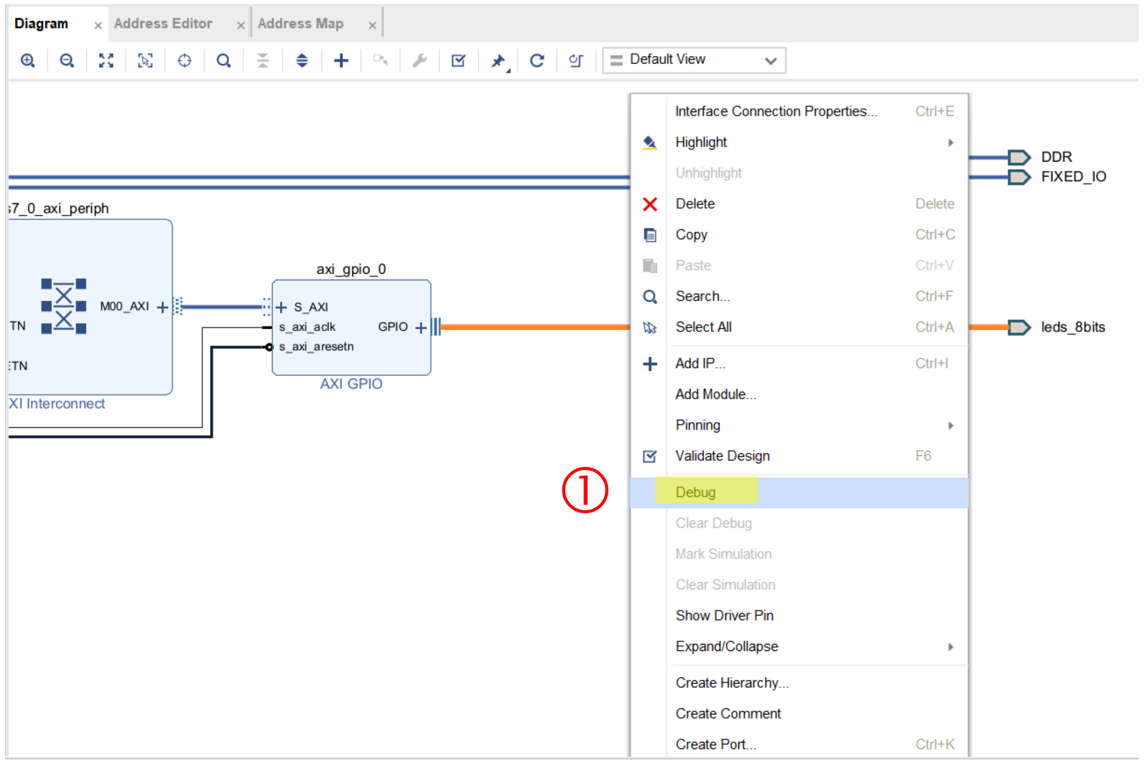
Vivado Block Design

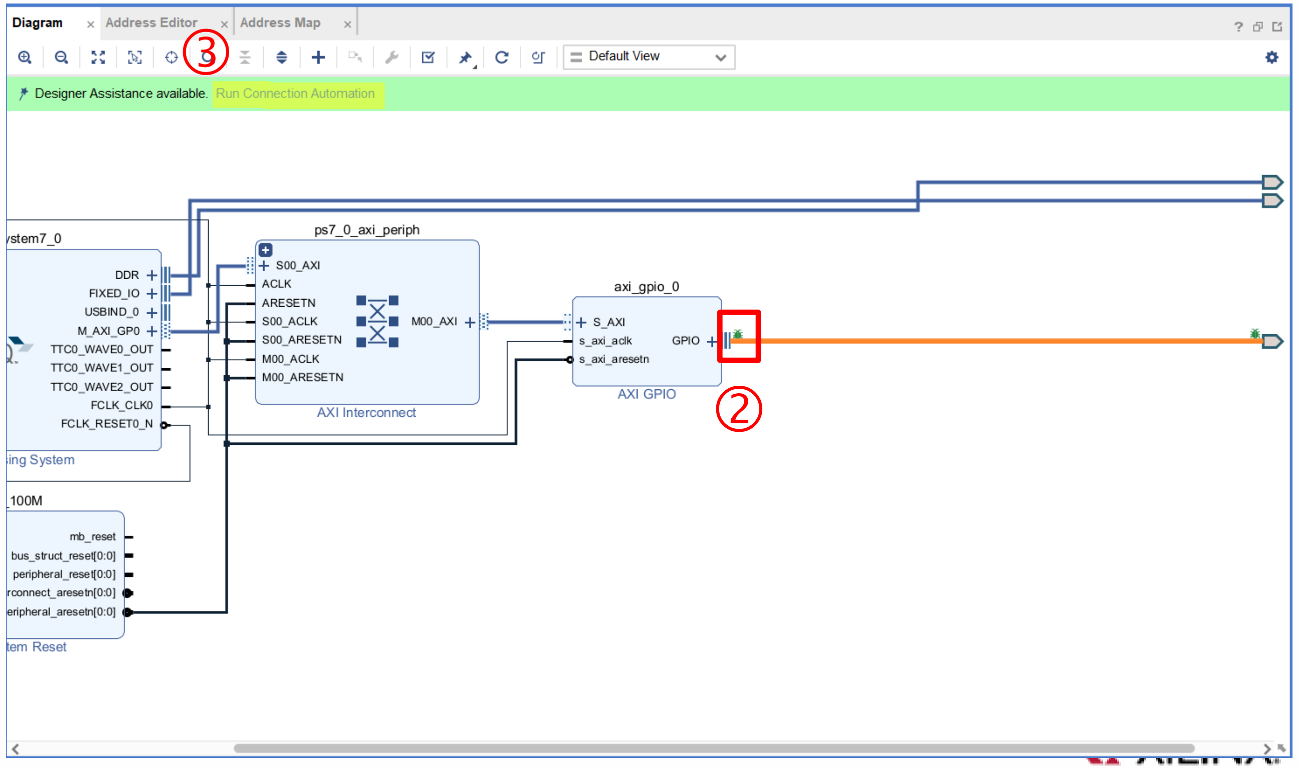
- Block Design



Vivado Block Design

- Block Design

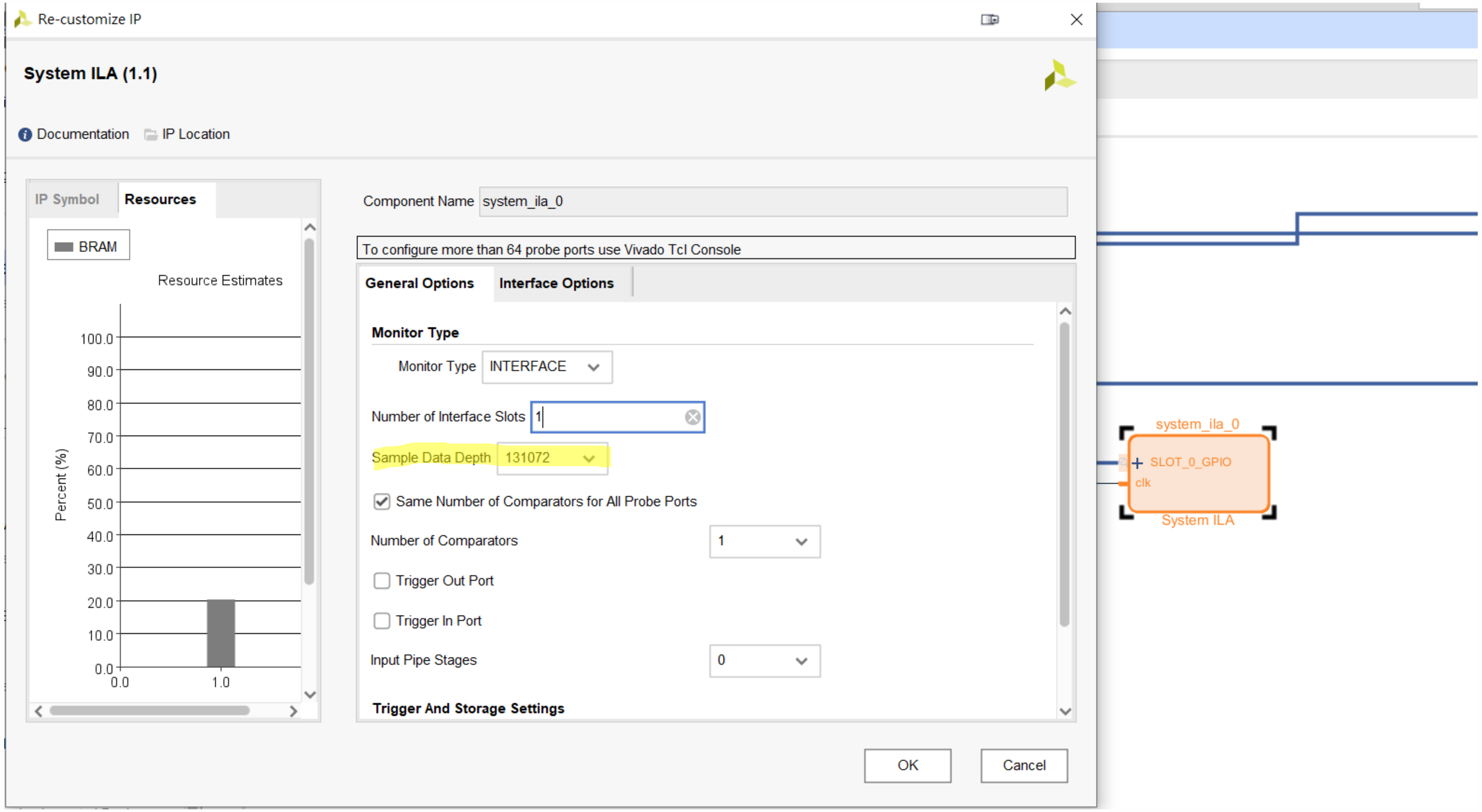




Designer Assistance available. Run Connection Automation

Vivado Block Design

- Re-customize System ILA


Re-customize IP

System ILA (1.1)

Documentation IP Location

Component Name system_ila_0

To configure more than 64 probe ports use Vivado Tcl Console

General Options | Interface Options

Monitor Type

Monitor Type: INTERFACE

Number of Interface Slots: 1

Sample Data Depth: 131072

[✓] Same Number of Comparators for All Probe Ports

Number of Comparators: 1

[ ] Trigger Out Port

[ ] Trigger In Port

Input Pipe Stages: 0

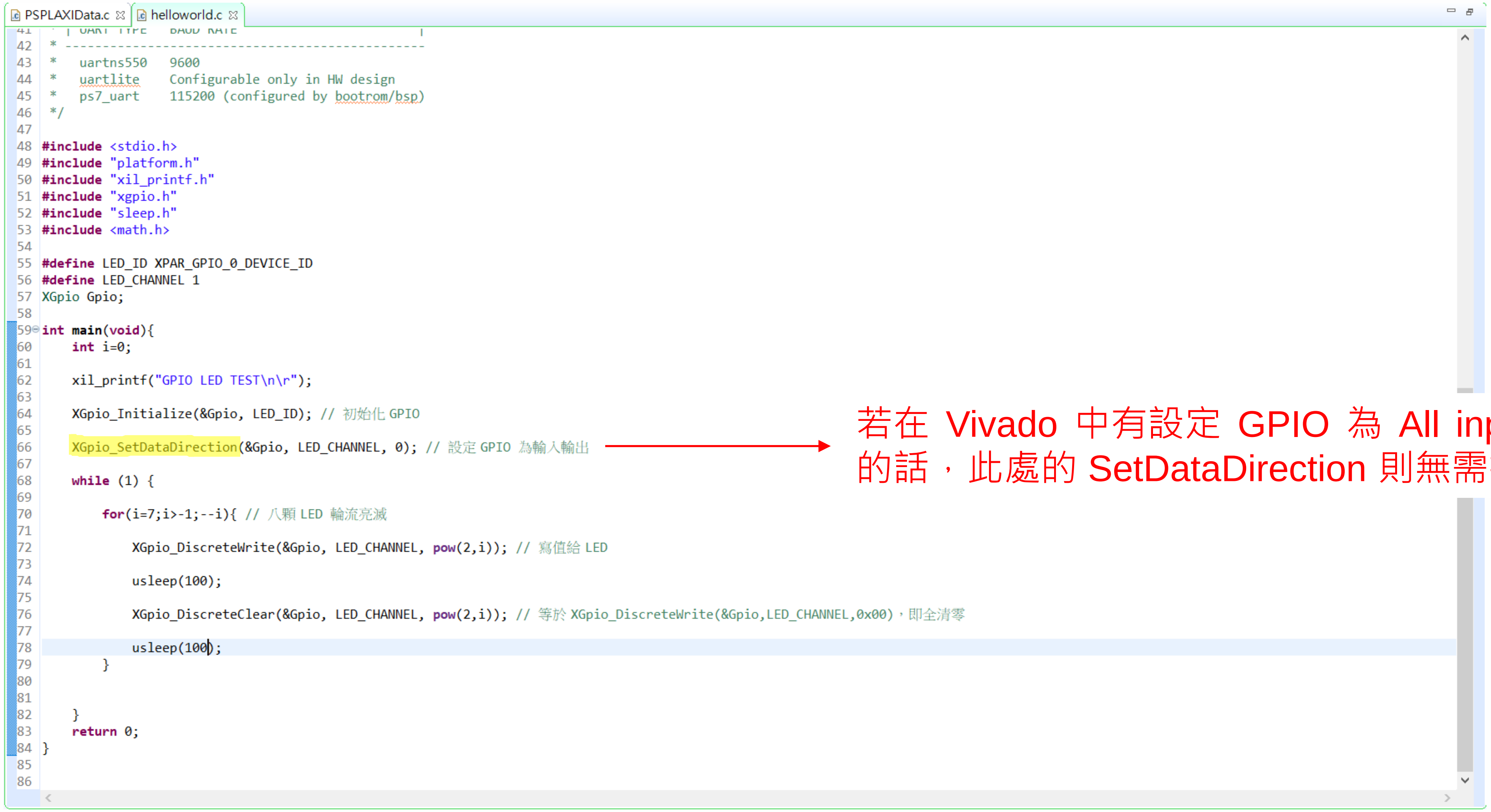
Trigger And Storage Settings

OK Cancel

Vitis Part

Vitis code result of execution

- Open Vitis to review the code



PSPLAXIData.c helloworld.c

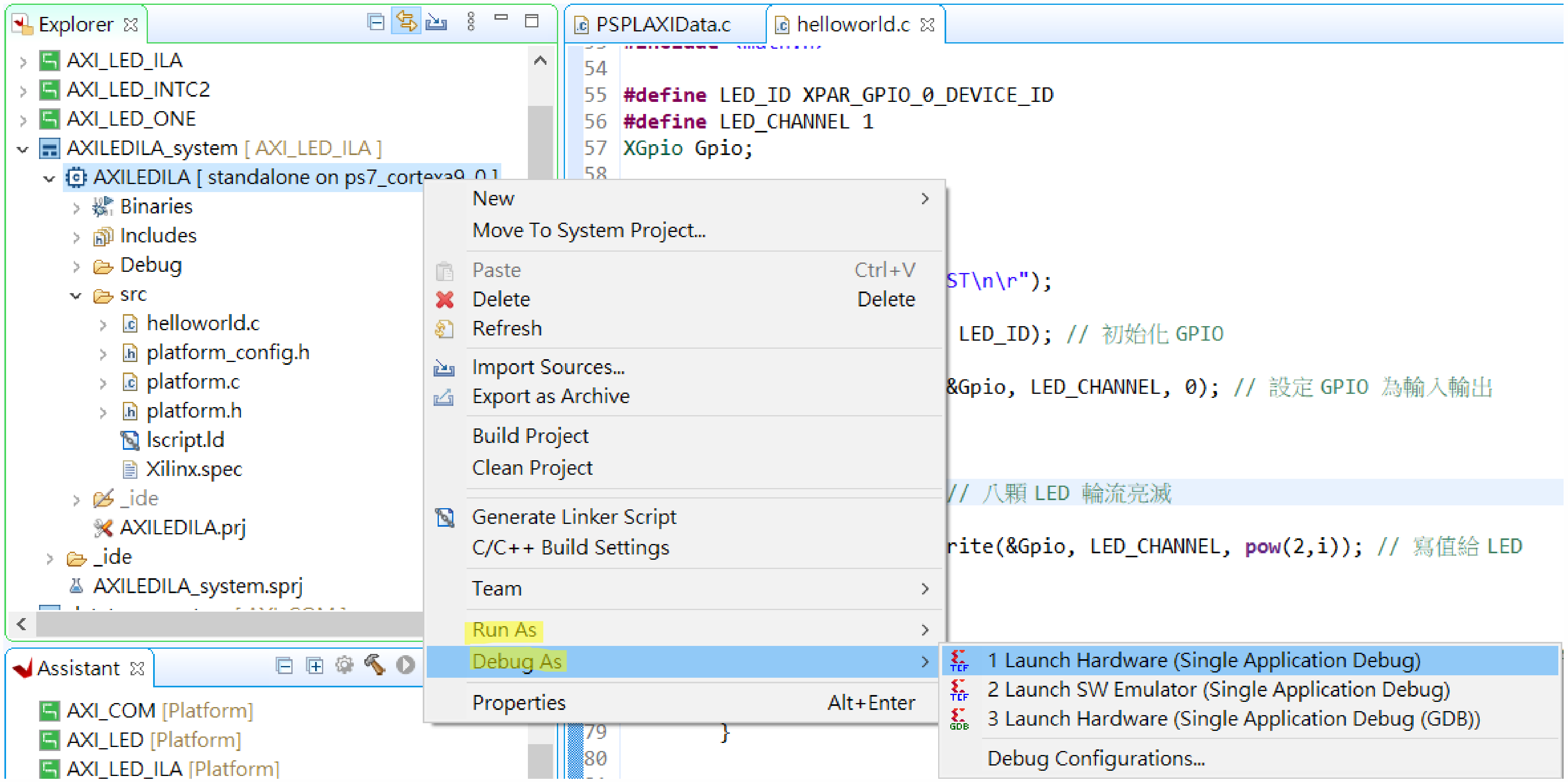
```c
 * | UART TYPE   BAUD RATE                                |
 * ------------------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "sleep.h"
#include <math.h>

#define LED_ID XPAR_GPIO_0_DEVICE_ID
#define LED_CHANNEL 1
XGpio Gpio;

int main(void){
    int i=0;

    xil_printf("GPIO LED TEST\n\r");

    XGpio_Initialize(&Gpio, LED_ID); // 初始化 GPIO

    XGpio_SetDataDirection(&Gpio, LED_CHANNEL, 0); // 設定 GPIO 為輸入輸出

    while (1) {

        for(i=7;i>-1;--i){ // 八顆 LED 輪流亮滅

            XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, pow(2,i)); // 寫值給 LED

            usleep(100);

            XGpio_DiscreteClear(&Gpio, LED_CHANNEL, pow(2,i)); // 等於 XGpio_DiscreteWrite(&Gpio,LED_CHANNEL,0x00)，即全清零

            usleep(100);
        }


    }
    return 0;
}


```

若在 Vivado 中有設定 GPIO 為 All input/All output 的話，此處的 SetDataDirection 則無需撰寫與加入

XILINX.

Vitis code result of execution

- Build the Project and Run

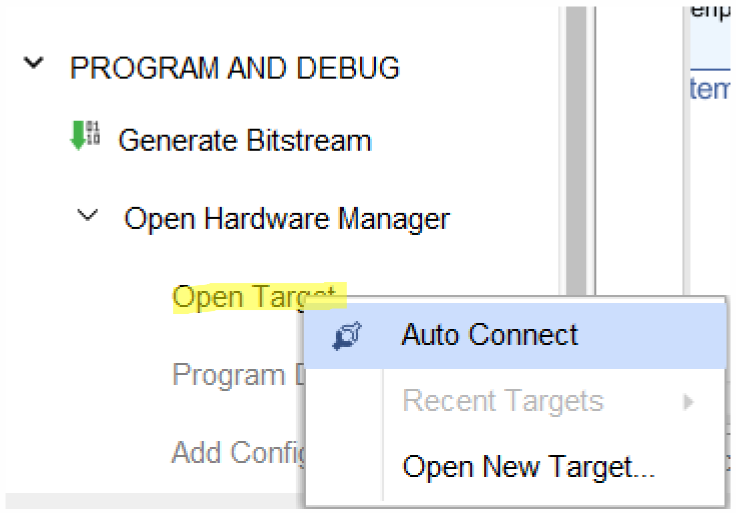


```
#define LED_ID XPAR_GPIO_0_DEVICE_ID
#define LED_CHANNEL 1
XGpio Gpio;
```

```
ST\n\r");
 LED_ID); // 初始化 GPIO
&Gpio, LED_CHANNEL, 0); // 設定 GPIO 為輸入輸出
// 八顆 LED 輪流亮滅
rite(&Gpio, LED_CHANNEL, pow(2,i)); // 寫值給 LED
}
```

Vivado Part

ILA result



PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

Open Target

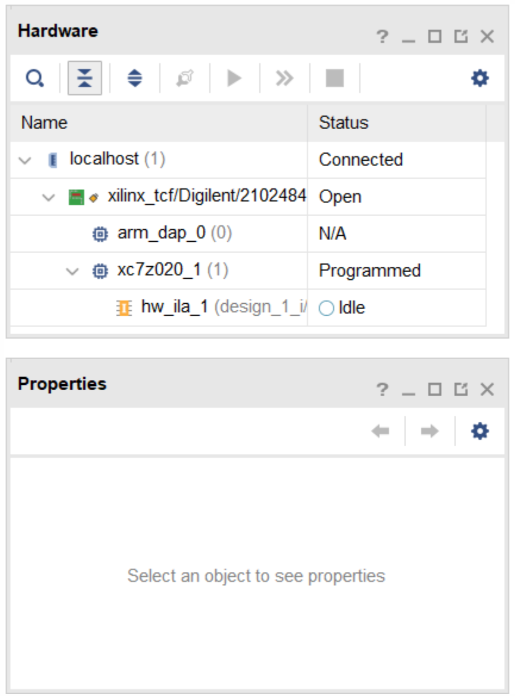
Program D

Add Confi

Auto Connect

Recent Targets

Open New Target...

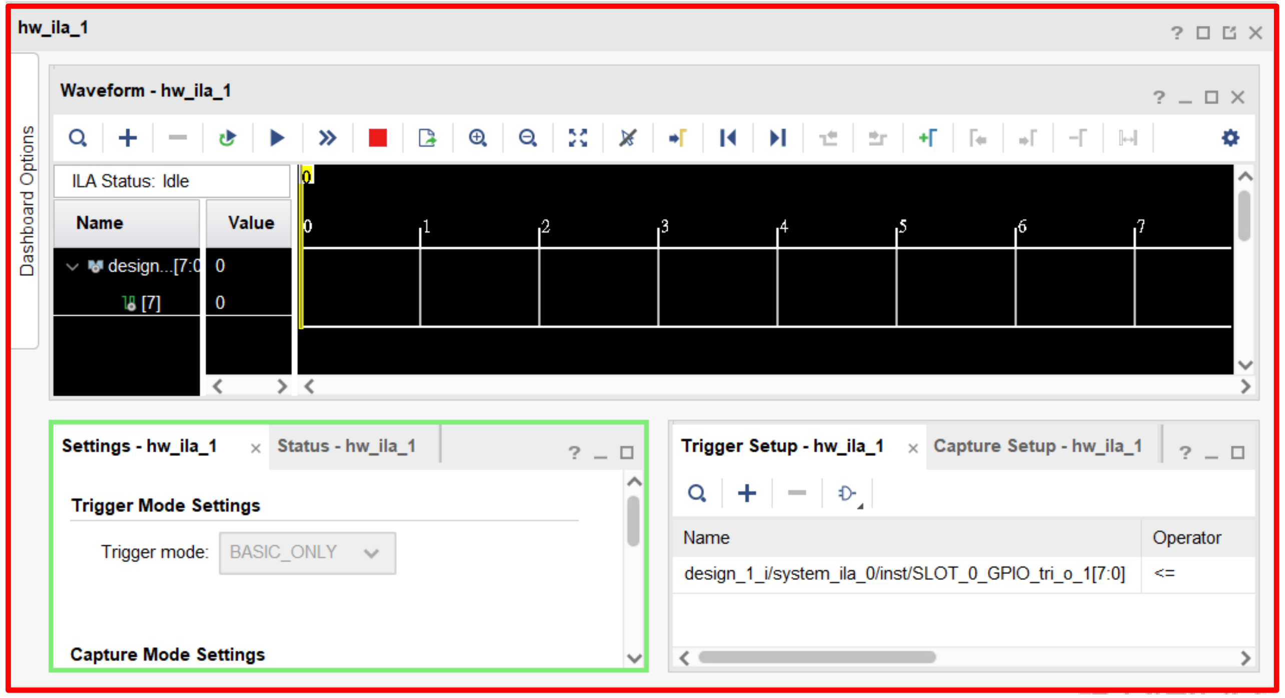


Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102484	Open
arm_dap_0 (0)	N/A
xc7z020_1 (1)	Programmed
hw_ila_1 (design_1_i/	Idle

Properties

Select an object to see properties



hw_ila_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
design...[7:0]	0
[7]	0

Settings - hw_ila_1

Status - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

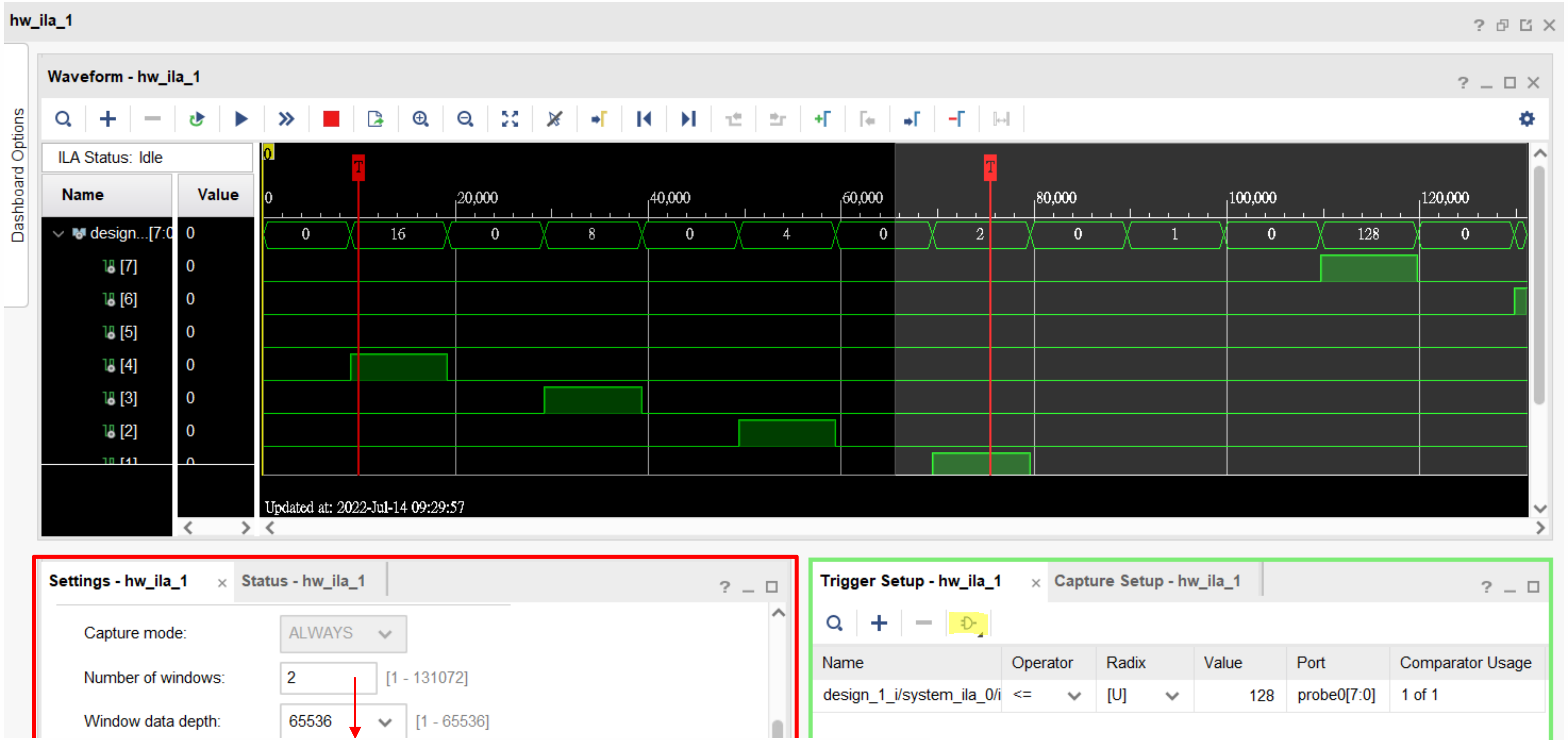
Capture Mode Settings

Trigger Setup - hw_ila_1

Capture Setup - hw_ila_1

Name	Operator
design_1_i/system_ila_0/inst/SLOT_0_GPIO_tri_o_1[7:0]	<=

ILA result



ILA Status: Idle

Name	Value
design...[7:0]	0
[7]	0
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0
[1]	0

Updated at: 2022-Jul-14 09:29:57

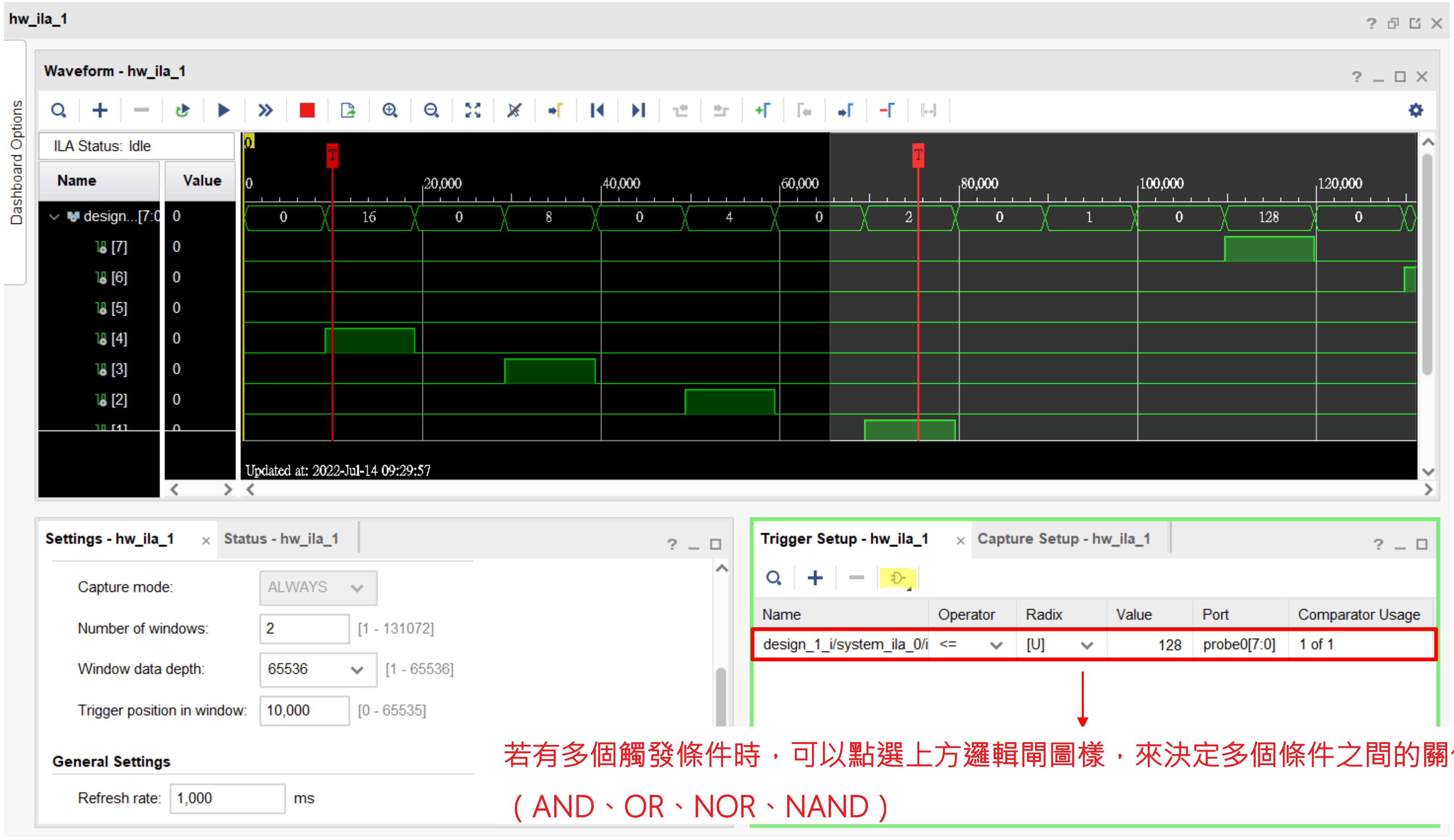
Capture mode:	ALWAYS	
Number of windows:	2	[1 - 131072]
Window data depth:	65536	[1 - 65536]

Name	Operator	Radix	Value	Port	Comparator Usage
design_1_i/system_ila_0/i	<=	[U]	128	probe0[7:0]	1 of 1

Numbers of windows 的用意在於，會將 Window data depth 進行劃分，以此圖為例，則為 65536 為一 window，共兩個 windows，一個 window 都會進行一次 trigger 的採樣條件，設定幾個 Number 就會進行幾次，來達到節省多次人工除錯的時間

XILINX.

ILA result



Capture mode: ALWAYS

Number of windows: 2 [1 - 131072]

Window data depth: 65536 [1 - 65536]

Trigger position in window: 10,000 [0 - 65535]

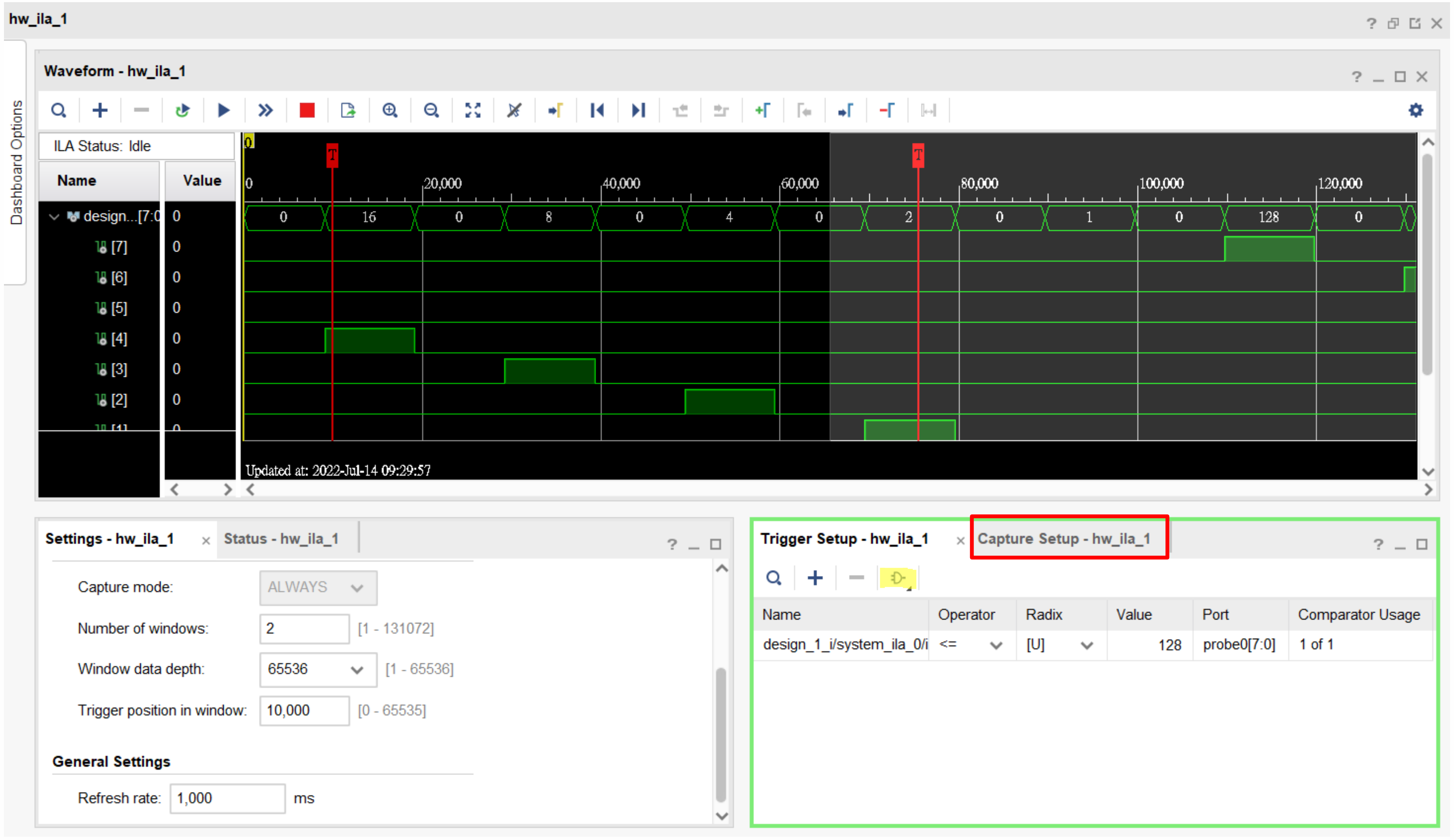
General Settings

Refresh rate: 1,000 ms

Name	Operator	Radix	Value	Port	Comparator Usage
design_1_i/system_ila_0/i	<=	[U]	128	probe0[7:0]	1 of 1

若有多個觸發條件時，可以點選上方邏輯閘圖樣，來決定多個條件之間的關係要怎麼觸發（AND、OR、NOR、NAND）

ILA result



Capture mode: ALWAYS

Number of windows: 2 [1 - 131072]

Window data depth: 65536 [1 - 65536]

Trigger position in window: 10,000 [0 - 65535]

General Settings

Refresh rate: 1,000 ms

Name	Operator	Radix	Value	Port	Comparator Usage
design_1_i/system_ila_0/i	<=	[U]	128	probe0[7:0]	1 of 1

ILA result

- Capture Setup

可以當作一種觸發條件，與 trigger 進行連動而不會使用到針腳以節省資源

Thank you very much for your attention!

XILINX.

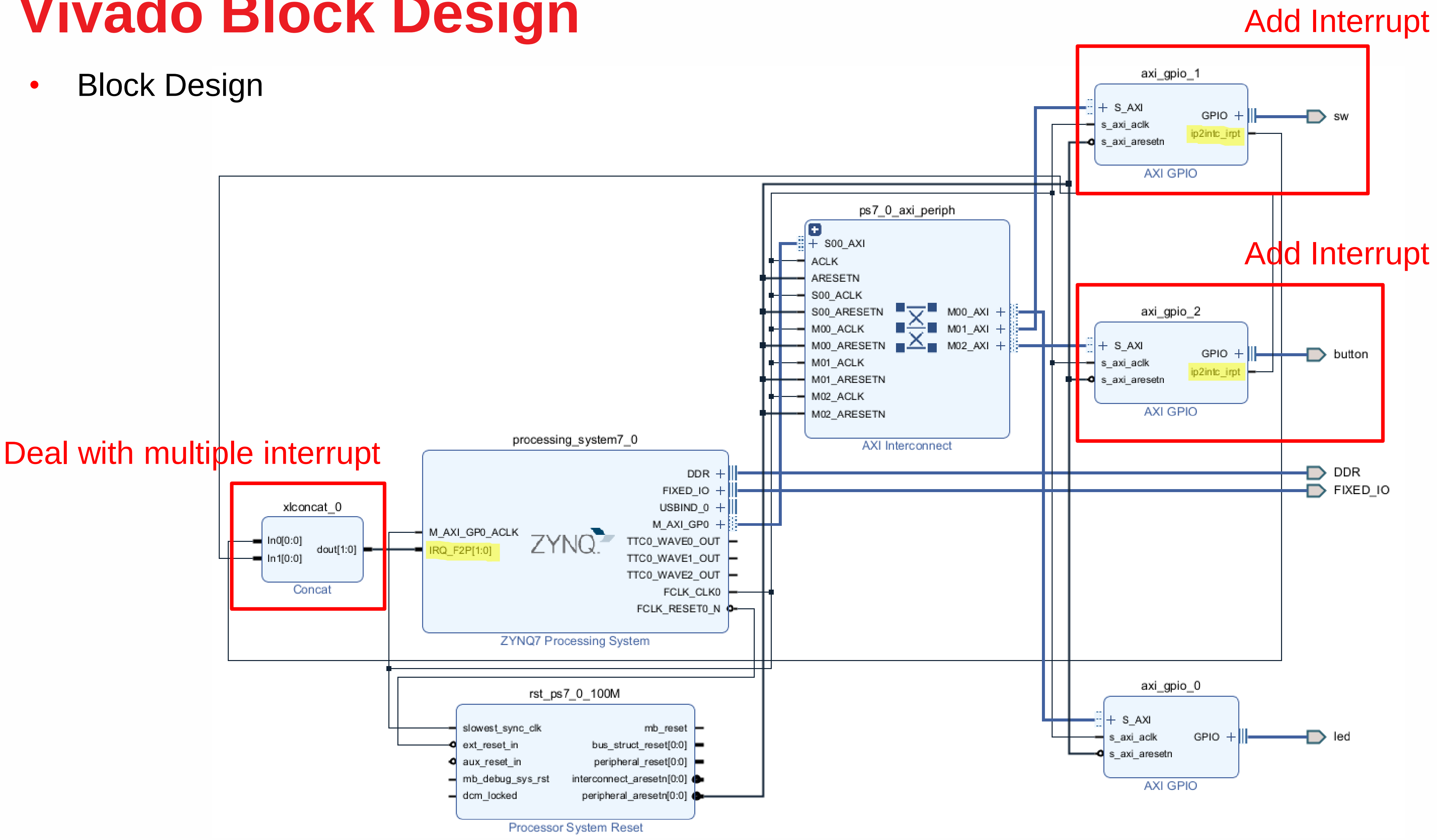
Appendix: Use Interrupt to Control AXI GPIO (Through PS GIC)

XILINX.

Vivado Part

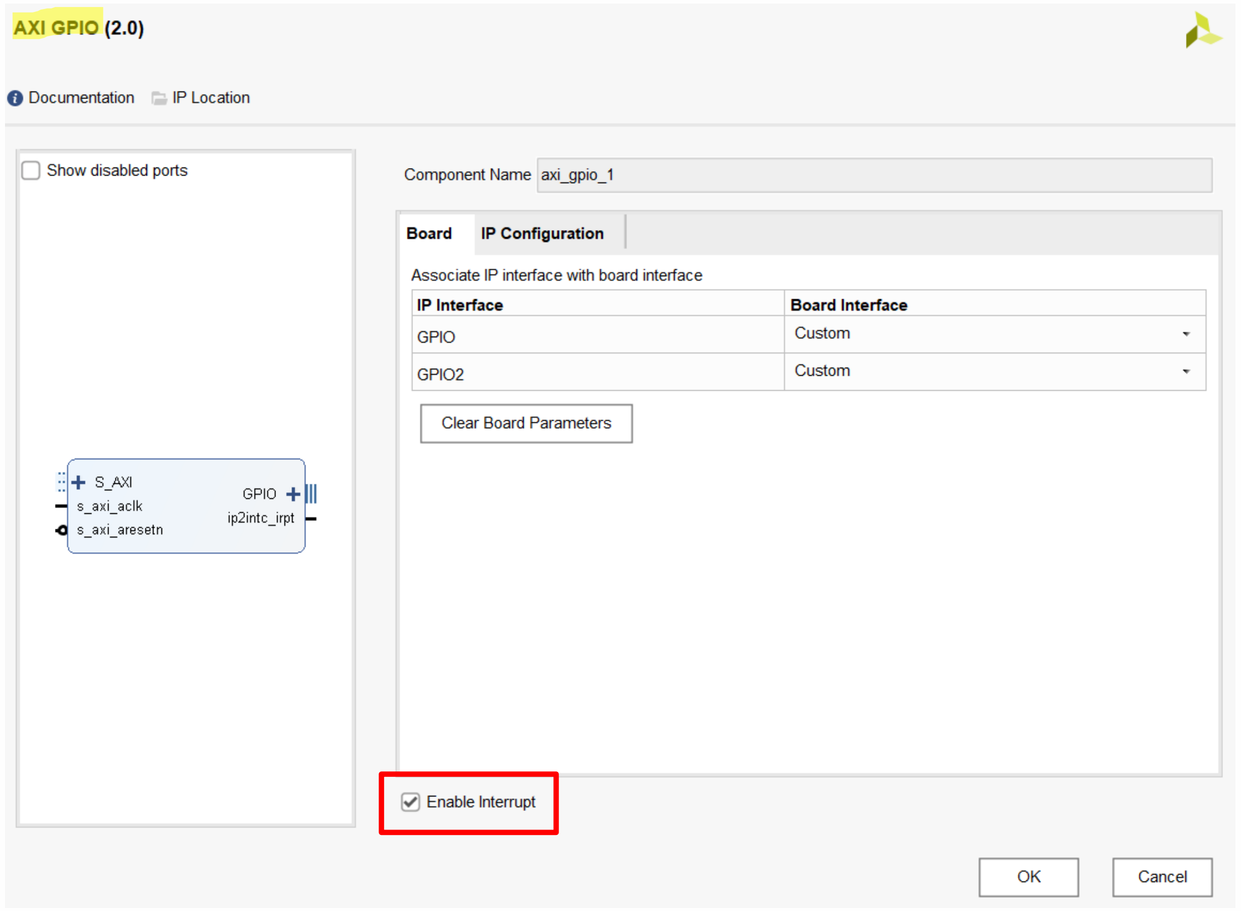
Vivado Block Design

- Block Design



Vivado Block Design

- Open the Interrupt option



AXI GPIO (2.0)

Documentation IP Location

Show disabled ports

Component Name axi_gpio_1

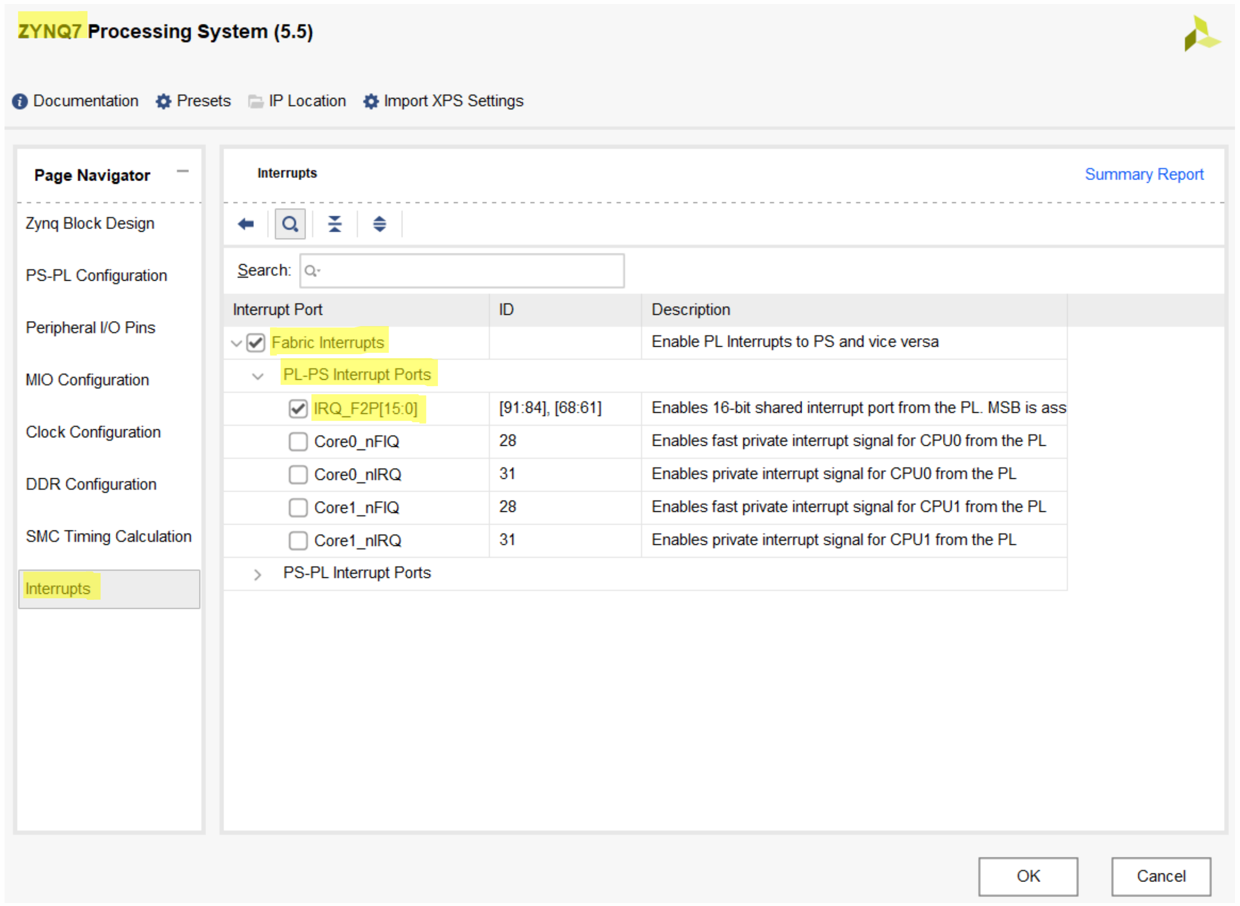
Board | IP Configuration

Associate IP interface with board interface

IP Interface	Board Interface
GPIO	Custom
GPIO2	Custom

Clear Board Parameters

☑ Enable Interrupt



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

Interrupts — Summary Report

Search:

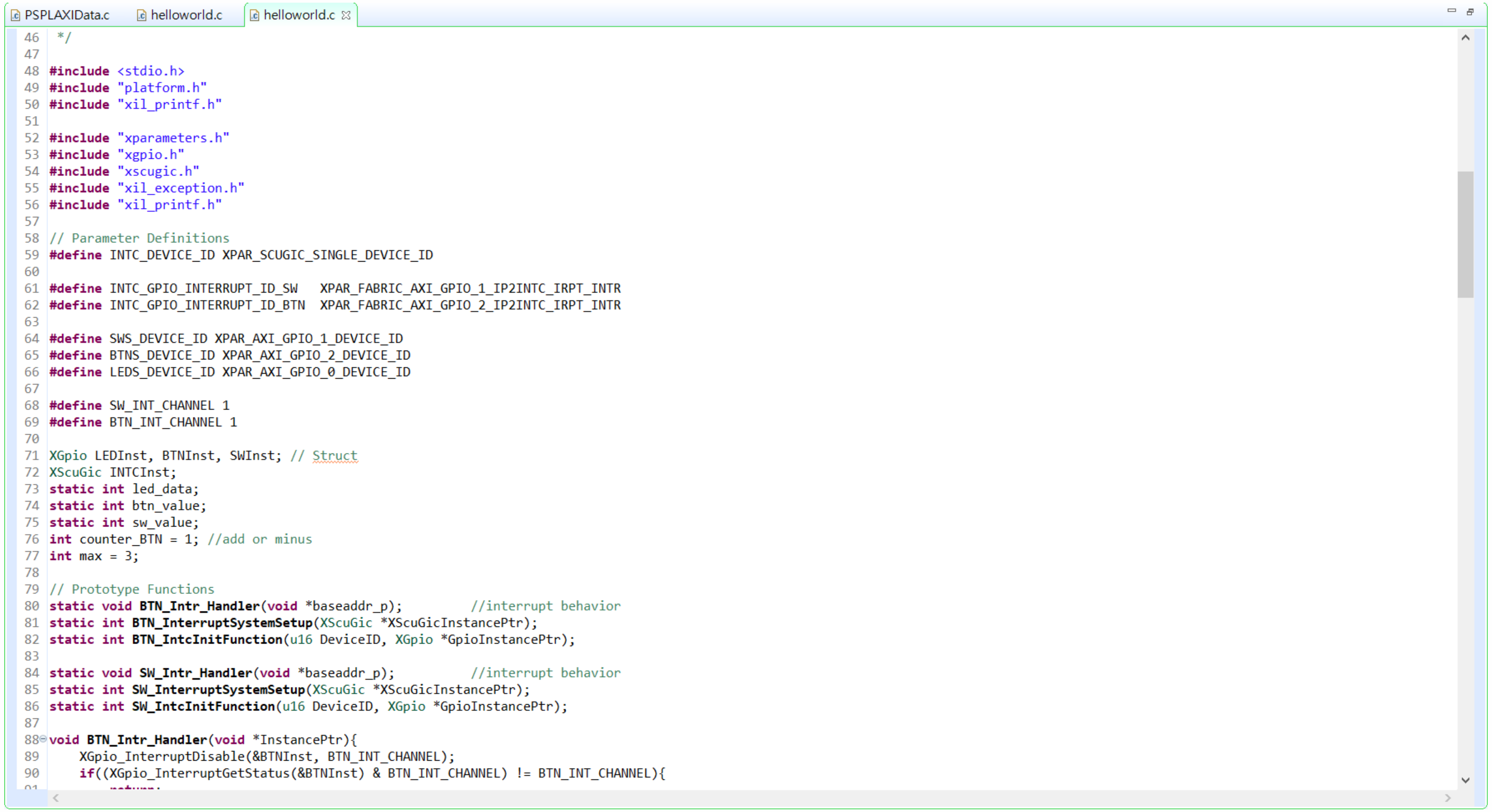
Interrupt Port	ID	Description
☑ Fabric Interrupts		Enable PL Interrupts to PS and vice versa
PL-PS Interrupt Ports		
☑ IRQ_F2P[15:0]	[91:84], [68:61]	Enables 16-bit shared interrupt port from the PL. MSB is ass
☐ Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL
☐ Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL
☐ Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL
☐ Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL
PS-PL Interrupt Ports		

◆ If your AXI GPIO is customized, you need to do I/O planning manually.

Vitis Part

Vitis code result of execution

- Open Vitis to review the code



```c
 */

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

#include "xparameters.h"
#include "xgpio.h"
#include "xscugic.h"
#include "xil_exception.h"
#include "xil_printf.h"

// Parameter Definitions
#define INTC_DEVICE_ID XPAR_SCUGIC_SINGLE_DEVICE_ID

#define INTC_GPIO_INTERRUPT_ID_SW   XPAR_FABRIC_AXI_GPIO_1_IP2INTC_IRPT_INTR
#define INTC_GPIO_INTERRUPT_ID_BTN  XPAR_FABRIC_AXI_GPIO_2_IP2INTC_IRPT_INTR

#define SWS_DEVICE_ID XPAR_AXI_GPIO_1_DEVICE_ID
#define BTNS_DEVICE_ID XPAR_AXI_GPIO_2_DEVICE_ID
#define LEDS_DEVICE_ID XPAR_AXI_GPIO_0_DEVICE_ID

#define SW_INT_CHANNEL 1
#define BTN_INT_CHANNEL 1

XGpio LEDInst, BTNInst, SWInst; // Struct
XScuGic INTCInst;
static int led_data;
static int btn_value;
static int sw_value;
int counter_BTN = 1; //add or minus
int max = 3;

// Prototype Functions
static void BTN_Intr_Handler(void *baseaddr_p);          //interrupt behavior
static int BTN_InterruptSystemSetup(XScuGic *XScuGicInstancePtr);
static int BTN_IntcInitFunction(u16 DeviceID, XGpio *GpioInstancePtr);

static void SW_Intr_Handler(void *baseaddr_p);           //interrupt behavior
static int SW_InterruptSystemSetup(XScuGic *XScuGicInstancePtr);
static int SW_IntcInitFunction(u16 DeviceID, XGpio *GpioInstancePtr);

void BTN_Intr_Handler(void *InstancePtr){
    XGpio_InterruptDisable(&BTNInst, BTN_INT_CHANNEL);
    if((XGpio_InterruptGetStatus(&BTNInst) & BTN_INT_CHANNEL) != BTN_INT_CHANNEL){
```

Vitis code result of execution

- Result