FZ3-Computing-Card FPGA Development Manual

Version V1.0

November 3, 2020

Revision History

Version	Description	Date
V1.0	Initial release	2020/11/03

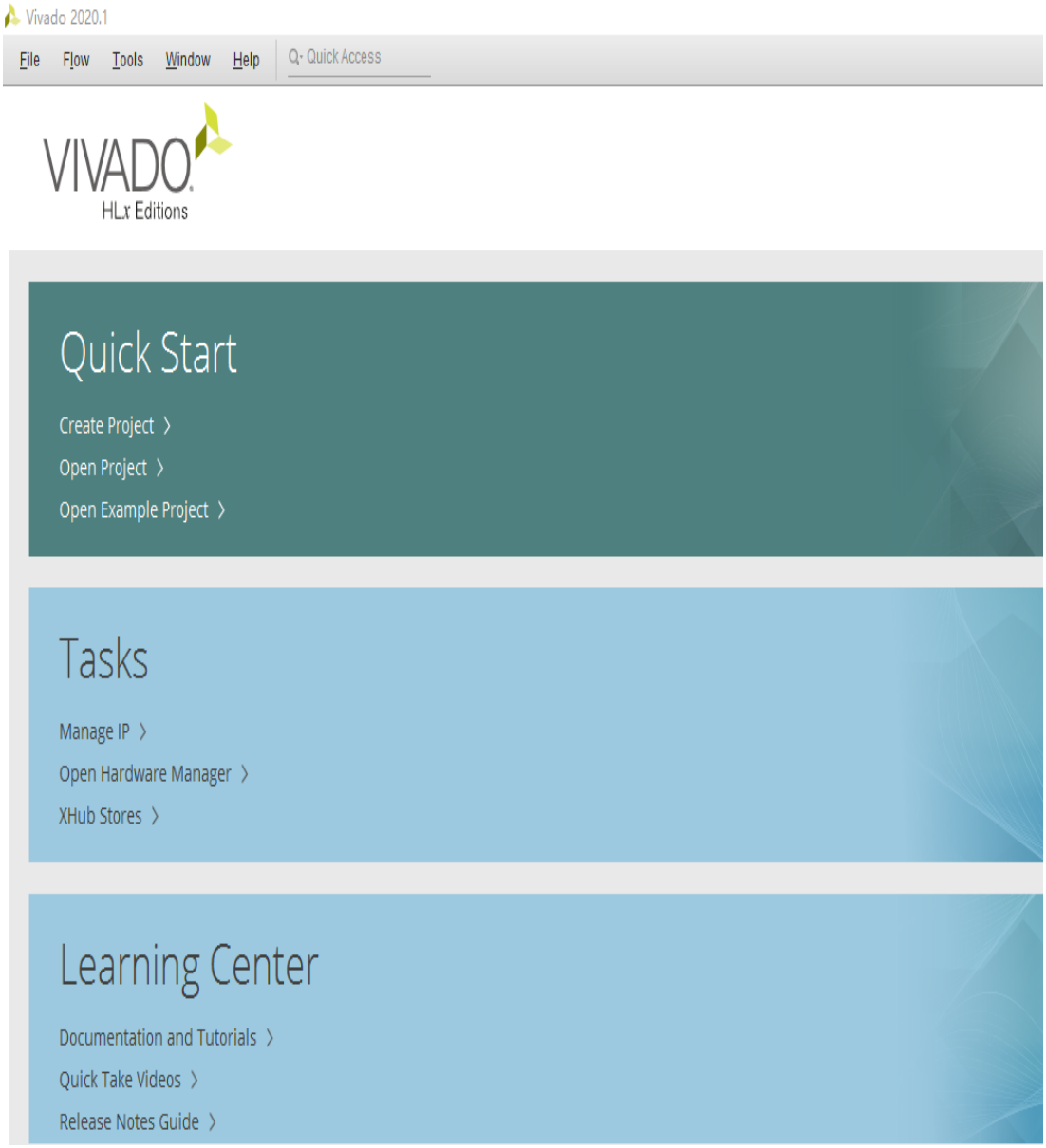
目录

Chapter 1 Hello_World

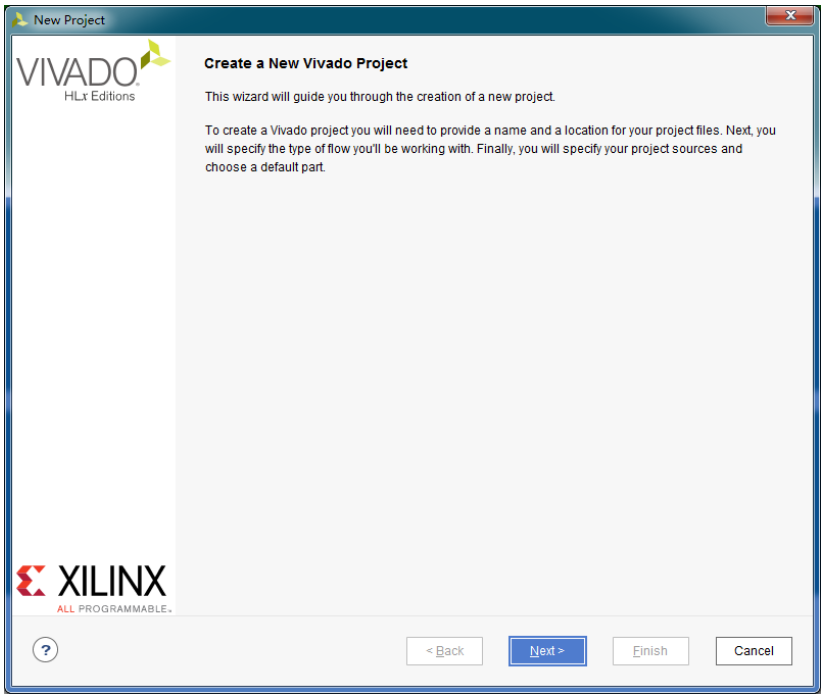
This example realizes printing out "Hello World" from serial port, which does not involve the development of the FPGA terminal, but ZYNQ's ARM and FPGA development are very close, through Hello World you can be familiar with the development environment of the FPGA.

1.1 New Vivado Project

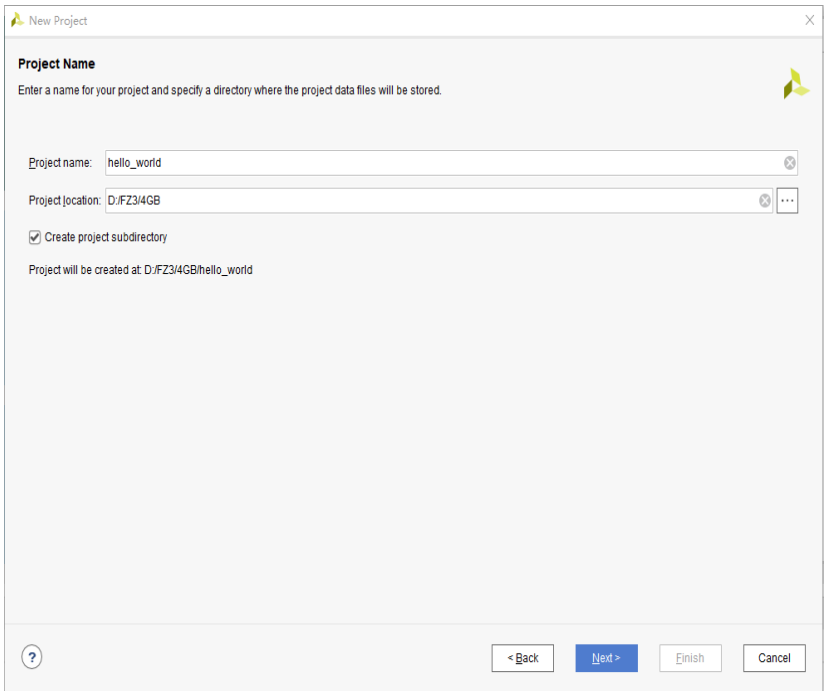
Click on "Create Project" to build a new project



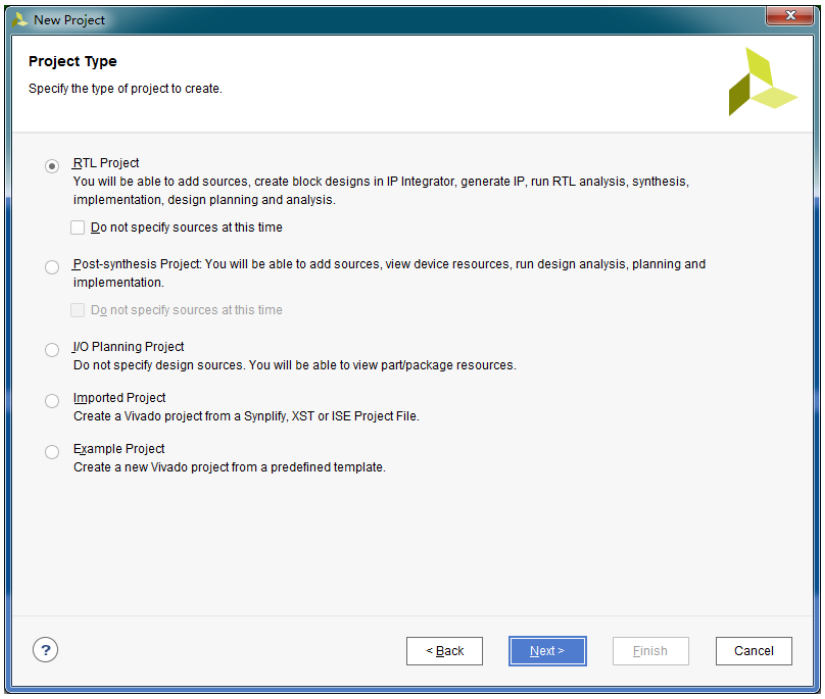
Click Next



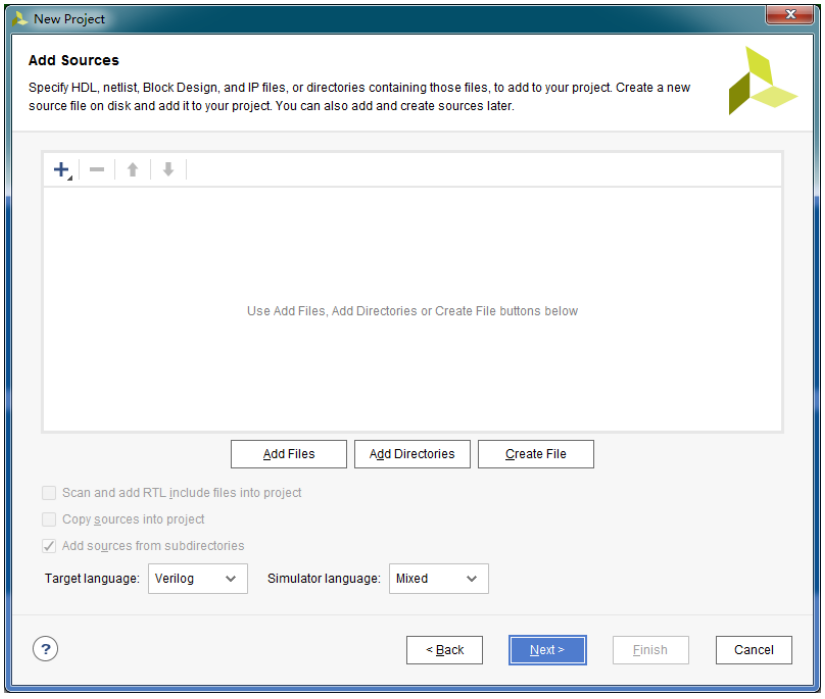
Enter project name and save path



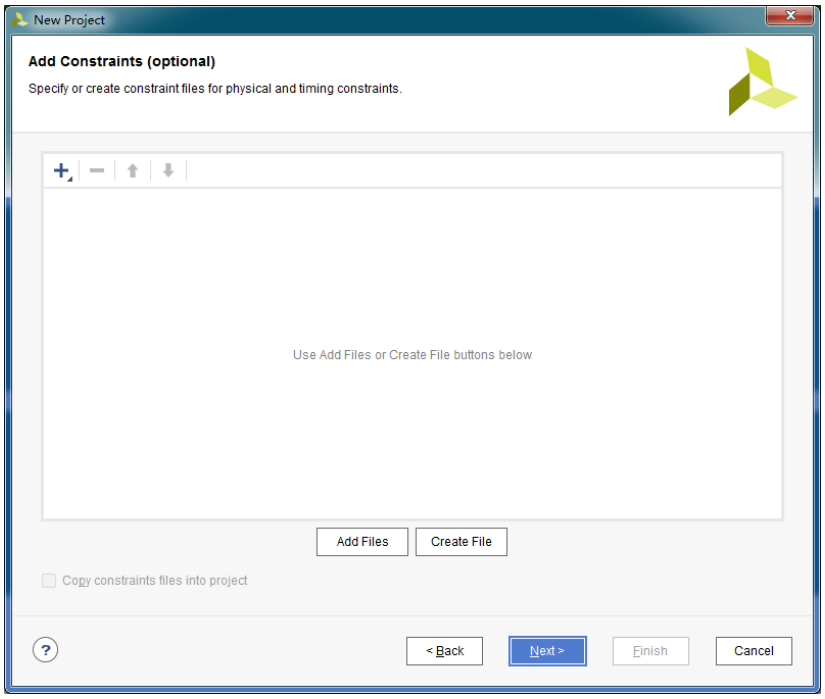
Click Next



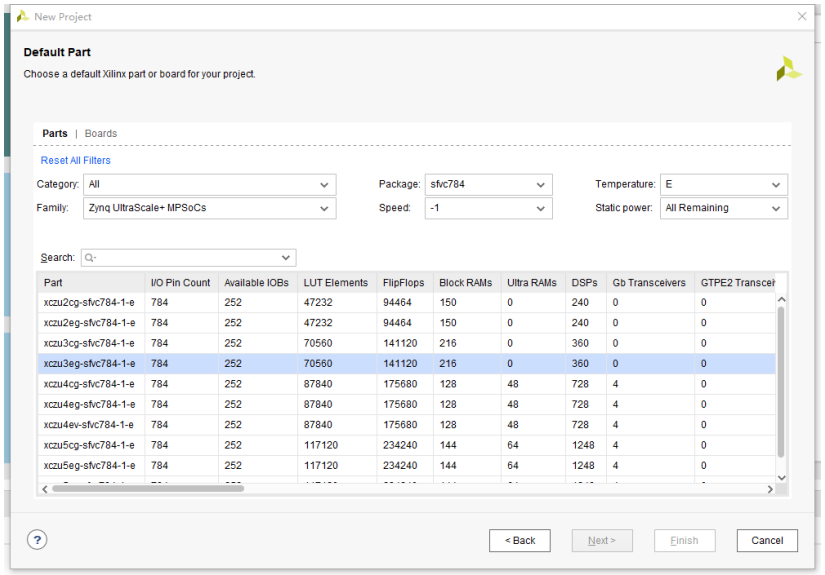
Click Next



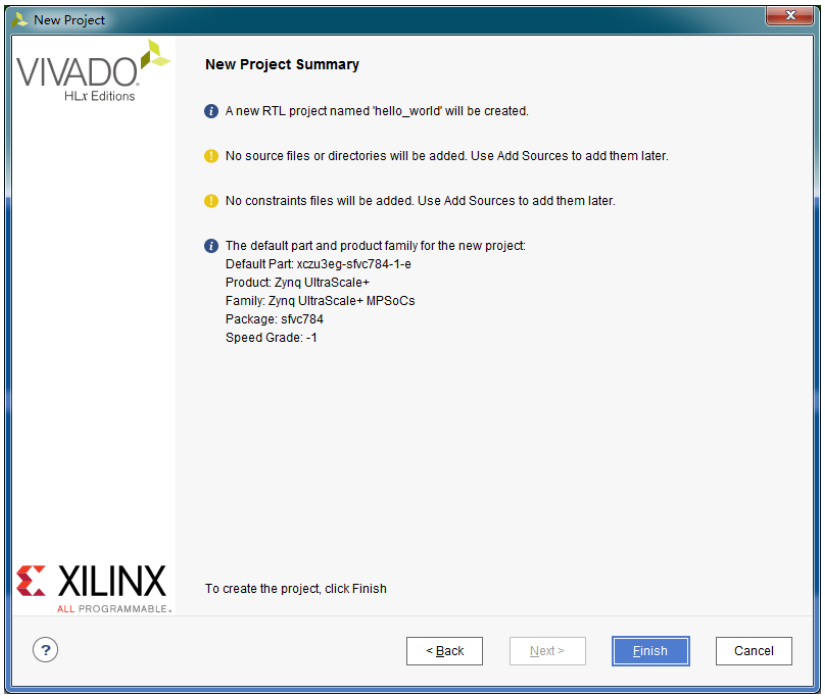
Click Next



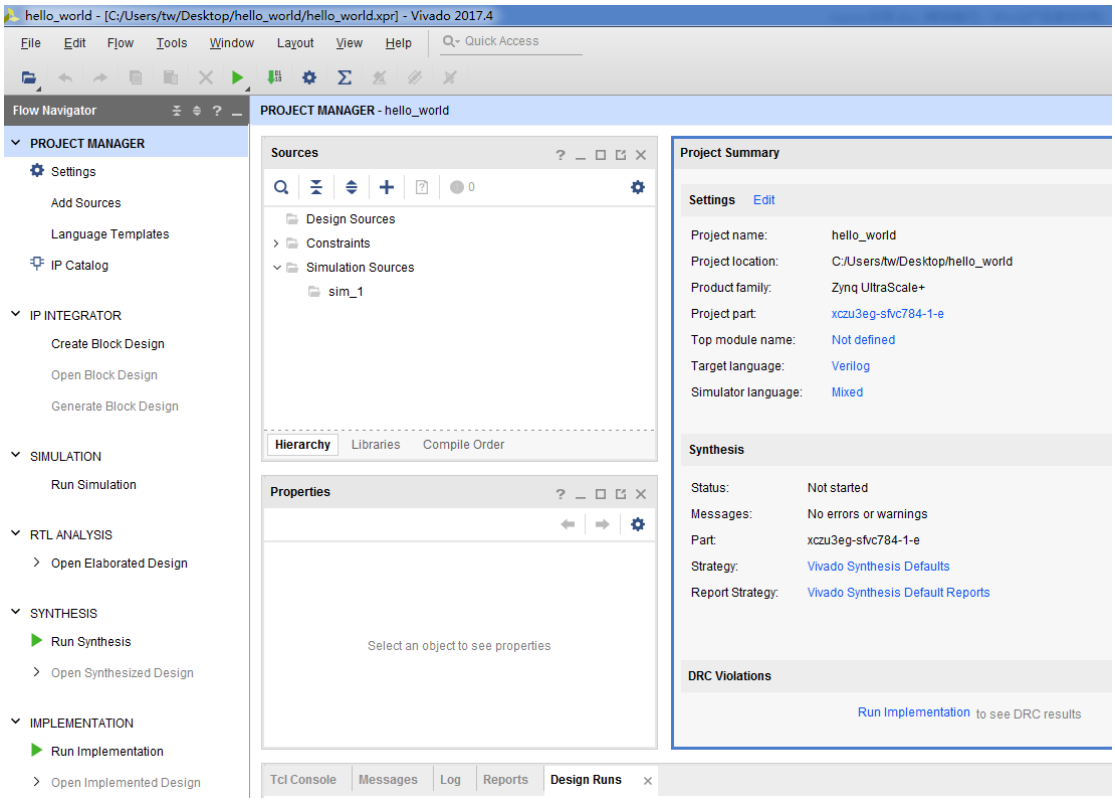
Chip type: xczu3eg-sfvc784-1-e ， Then click Next



Click Finish

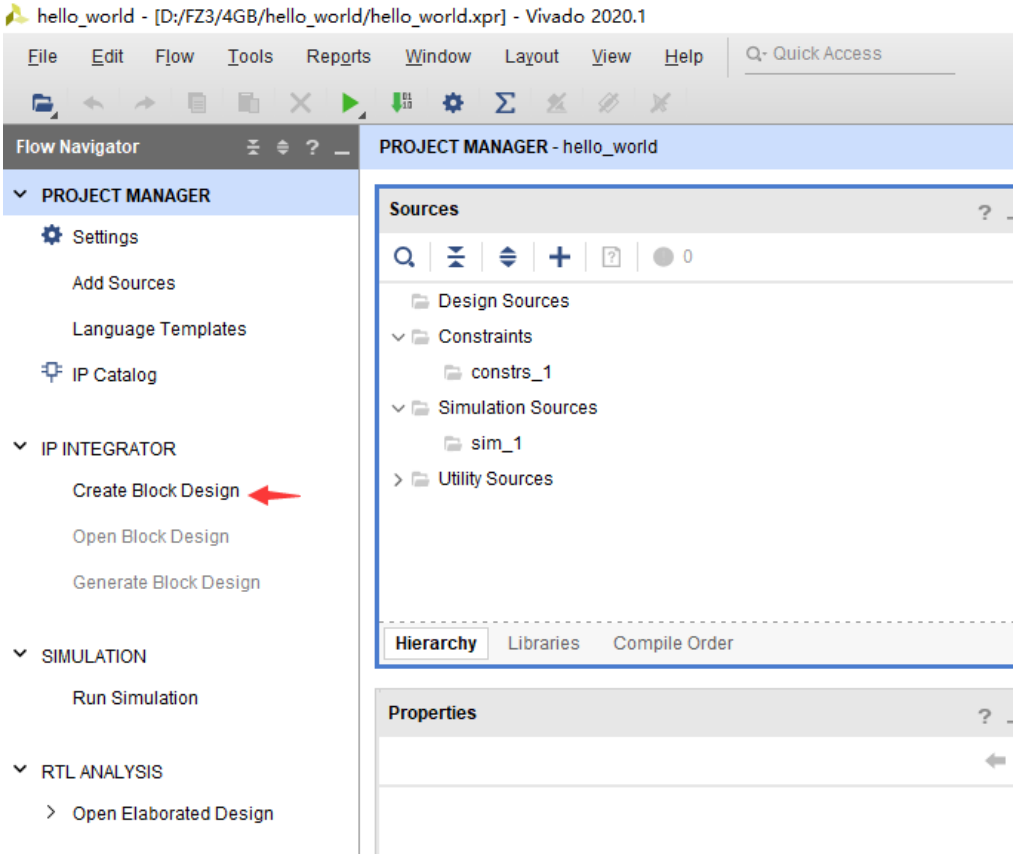


Generated vivado project

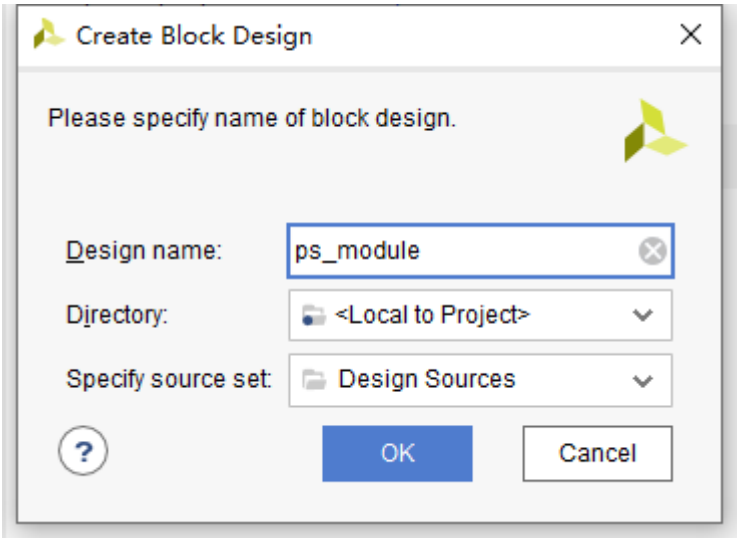


1.2 New Block Design

Click Create Block Design

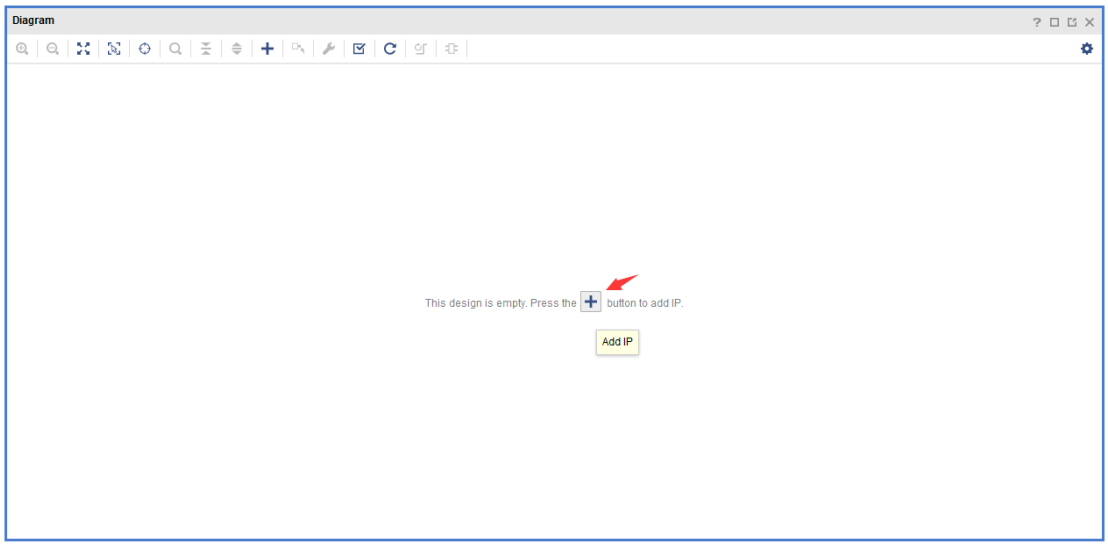


Click OK

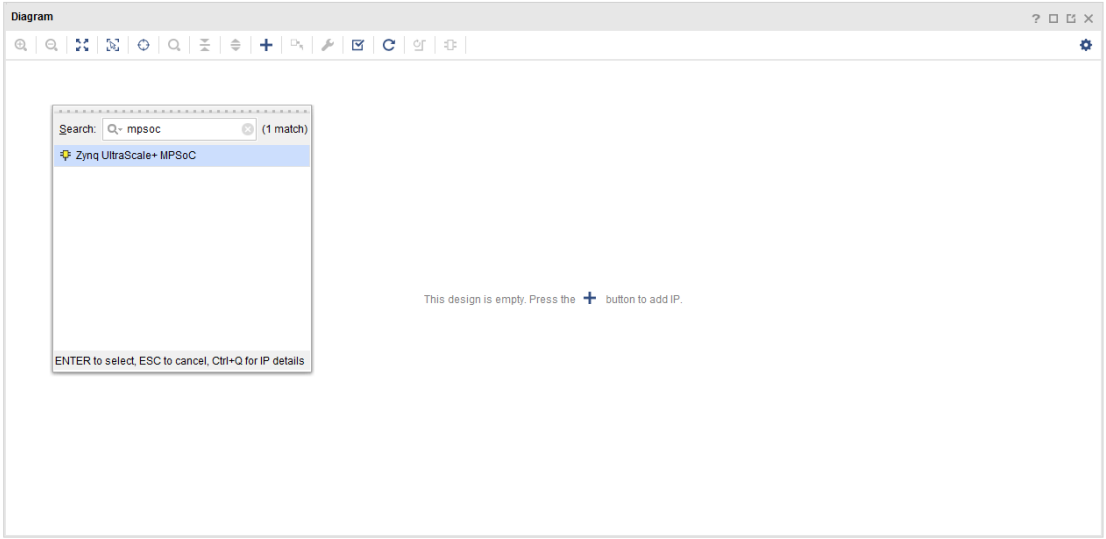


1.3 Add PS IP Core and Configure

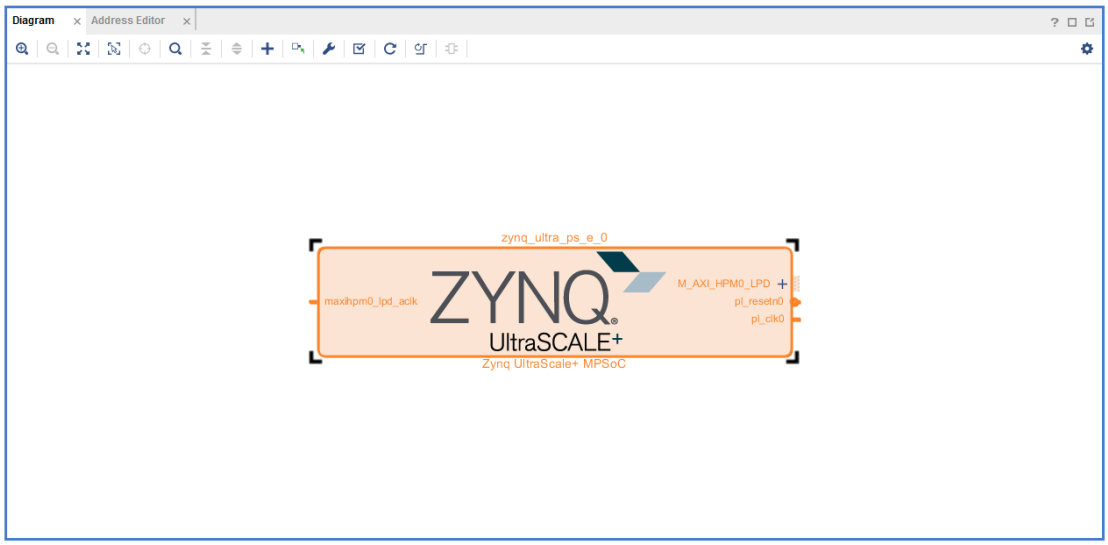
Click Add IP to add IP core



Enter mpsoc, then double-click Zynq UltraScale + MPSoC to add the MPSoC core

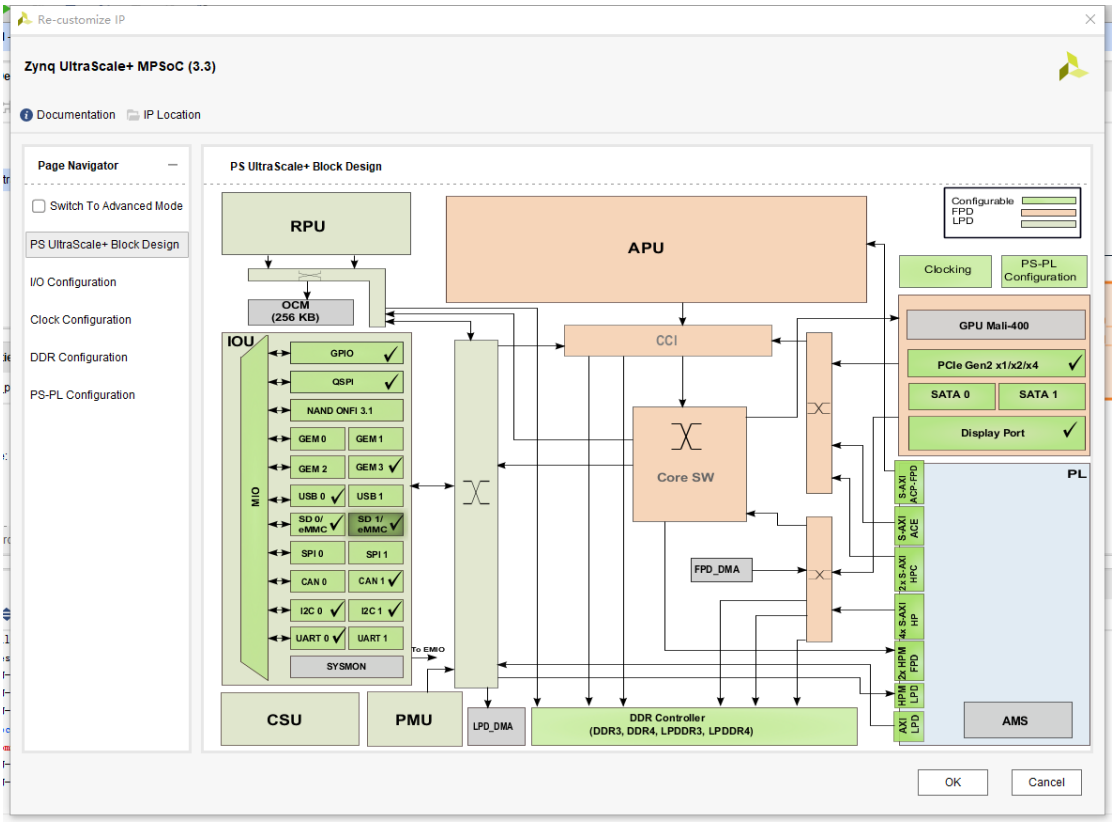


The Zynq MPSoC core is shown in the following figure



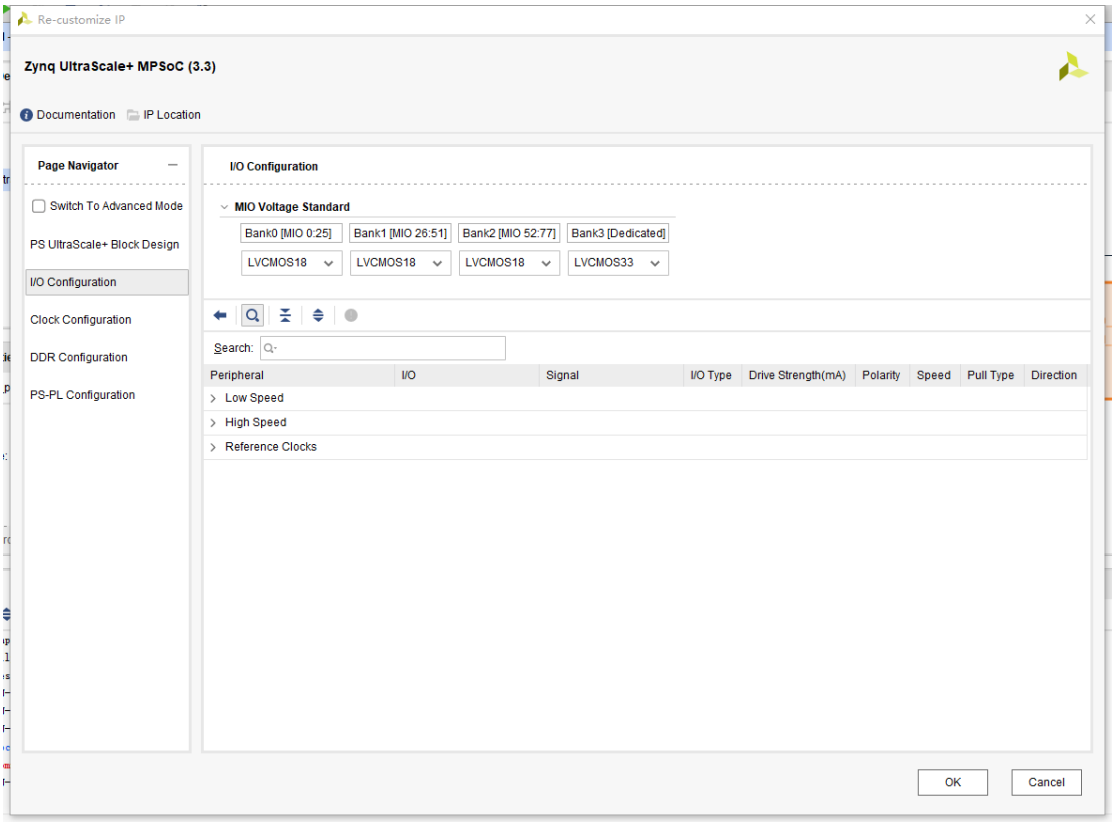
Double-click zynq MPSoC core to import configuration file

The first interface is the architecture diagram of the zynq hard core. You can see its structure clearly. You can refer to the ug1085 document, which has a detailed introduction to zynq. The green parts in the figure are configurable modules. You can click to enter the corresponding editing interface, or you can enter editing interface in the left window. The following describes the configuration of each window, refer to pg201.



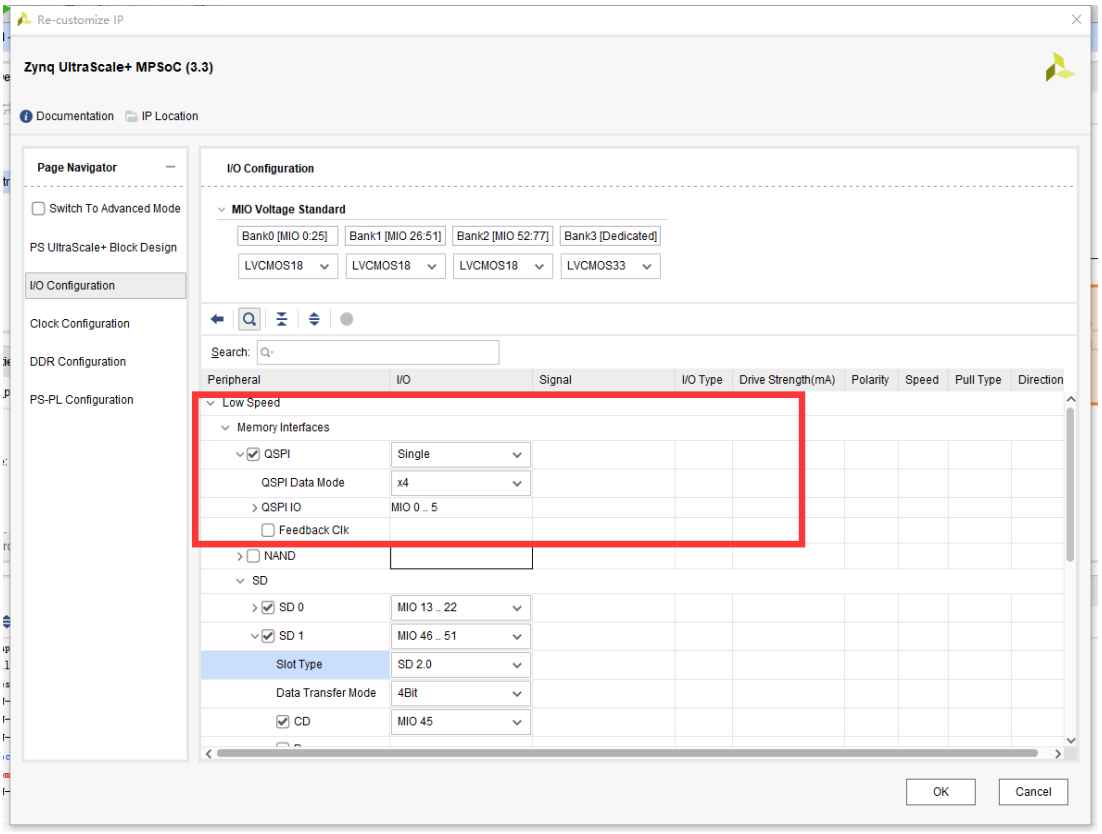
1、Voltage configuration

In the I / O configuration window, configure the voltage of bank0 ~ bank2 as lvcmos18 and bank3 as lvcmos33.

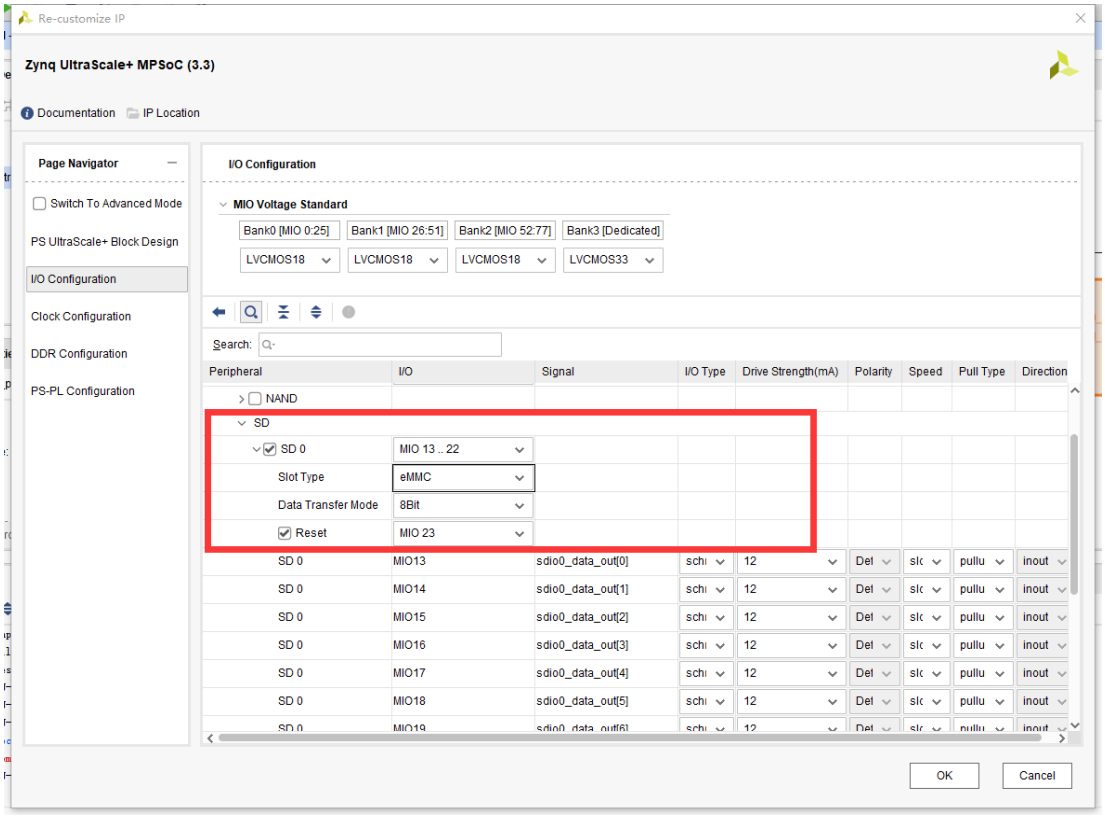


2、Low Speed configuration

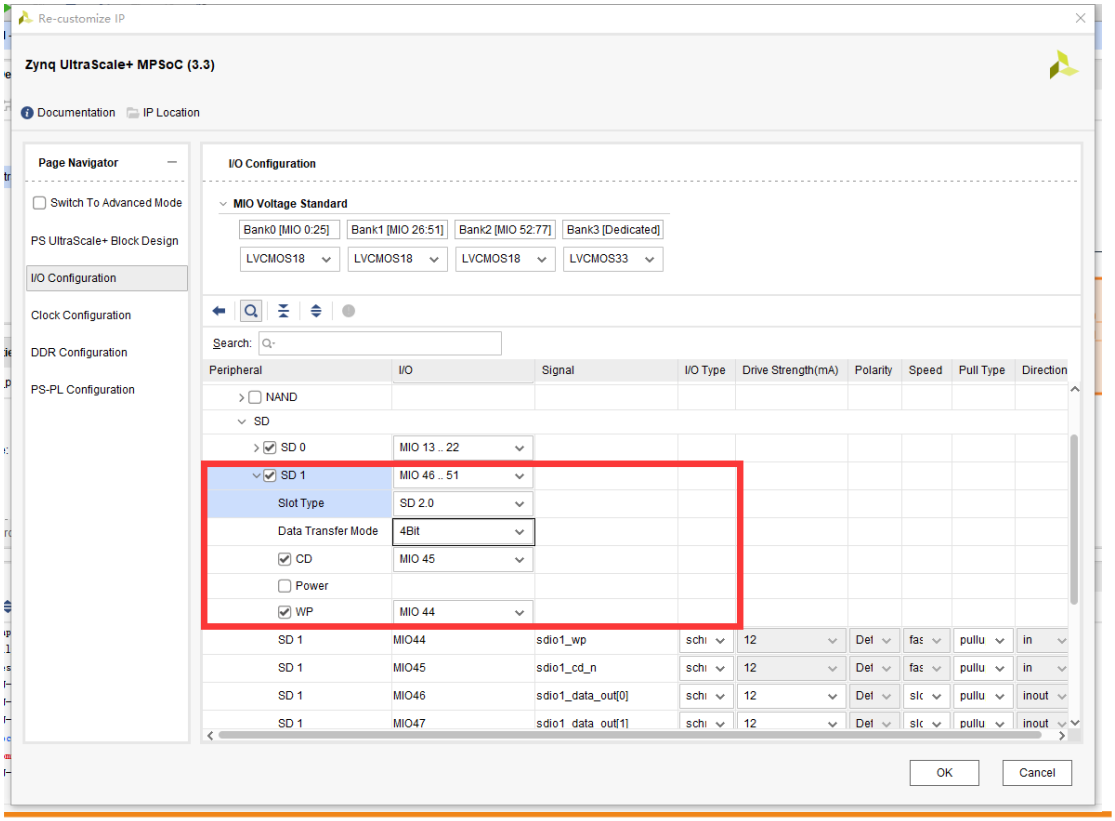
Check QSPI and set it to "single" mode and data mode to "X4"



Select SD 0 and configure EMMC. Select mio13.. 22, slot type EMMC, data transfer mode 8bit, check reset, and select mio23.



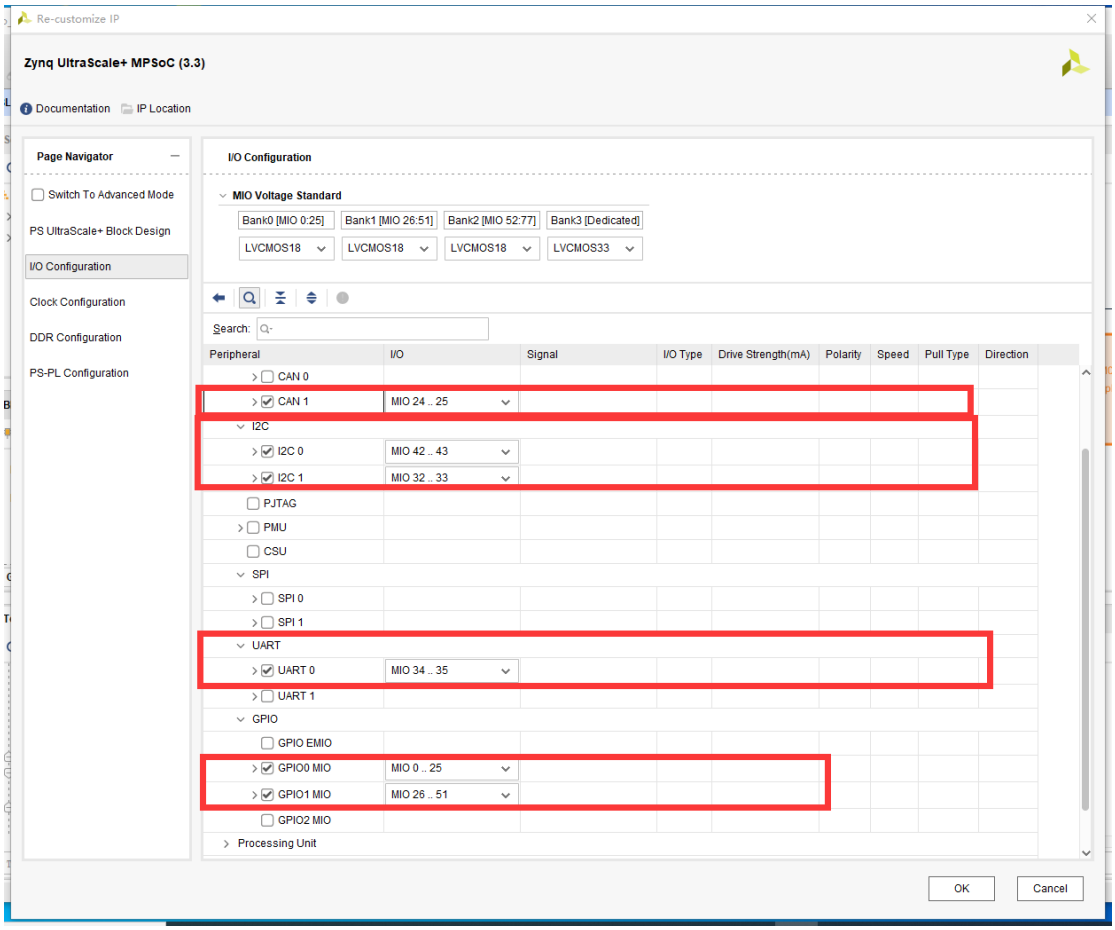
Select SD 1 and configure SD card. Select Mio 46.. 51, slot type SD 2.0, data transfer mode 4bit, CD to detect SD card insertion, mio45, WP, mio44 for SD card write protection .



Tick can 1 and select Mio 24.. 25

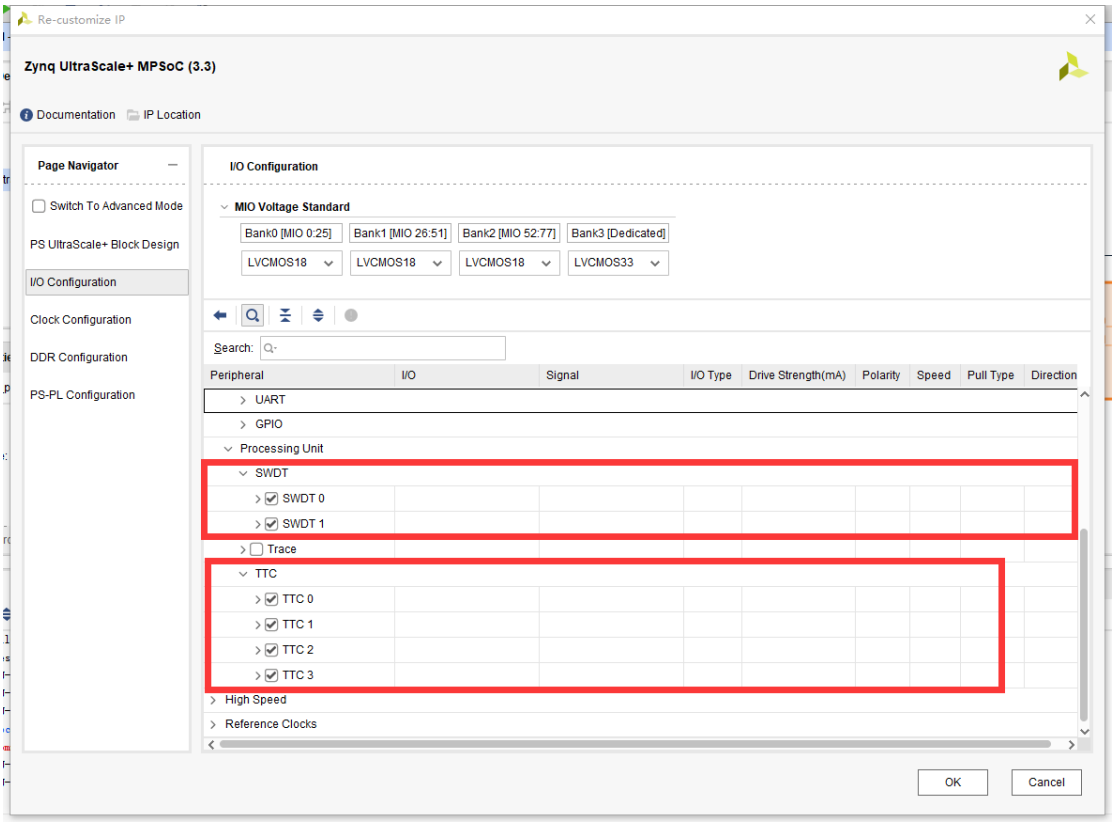
Tick I2C 0, select Mio 42.. 43; tick I2C 1, select mio32.. 33

Tick serial port UART 0, select MIO 34.. 35, tick gpio1 MIO, gpio0 MIO



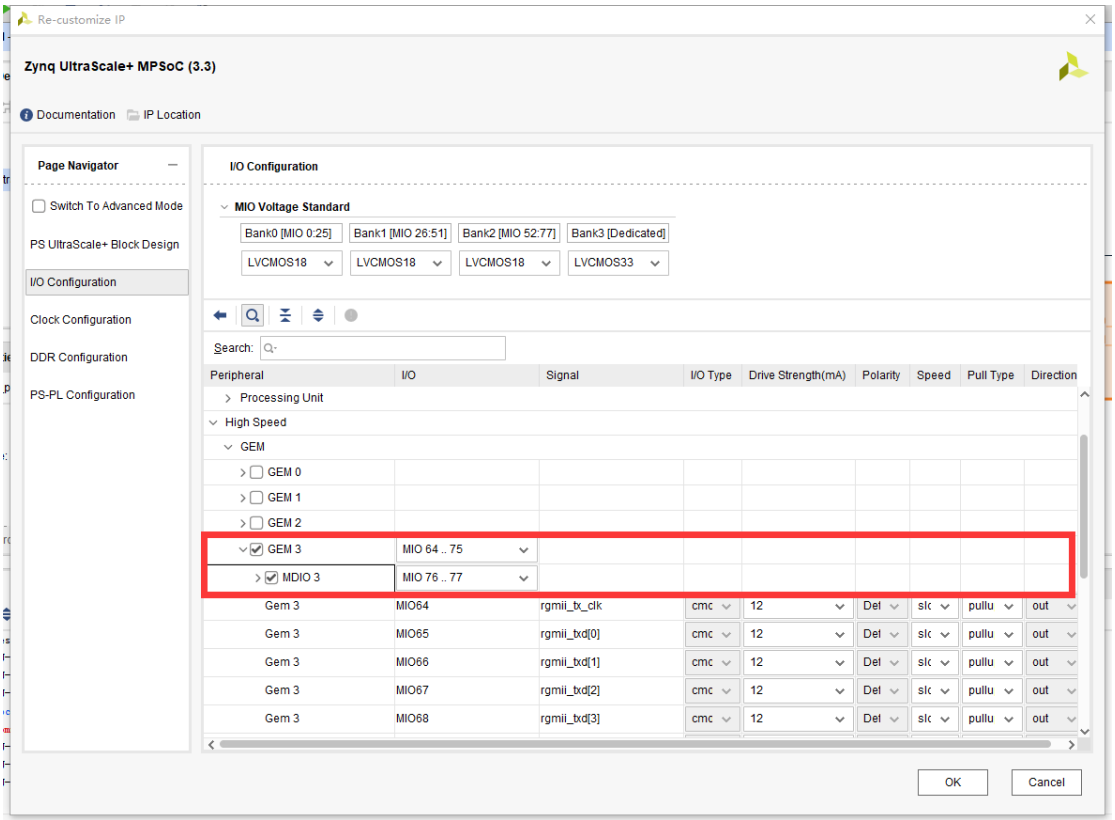
Tick SWDT 0、SWDT 1

Tick TTC 0~TTC 3

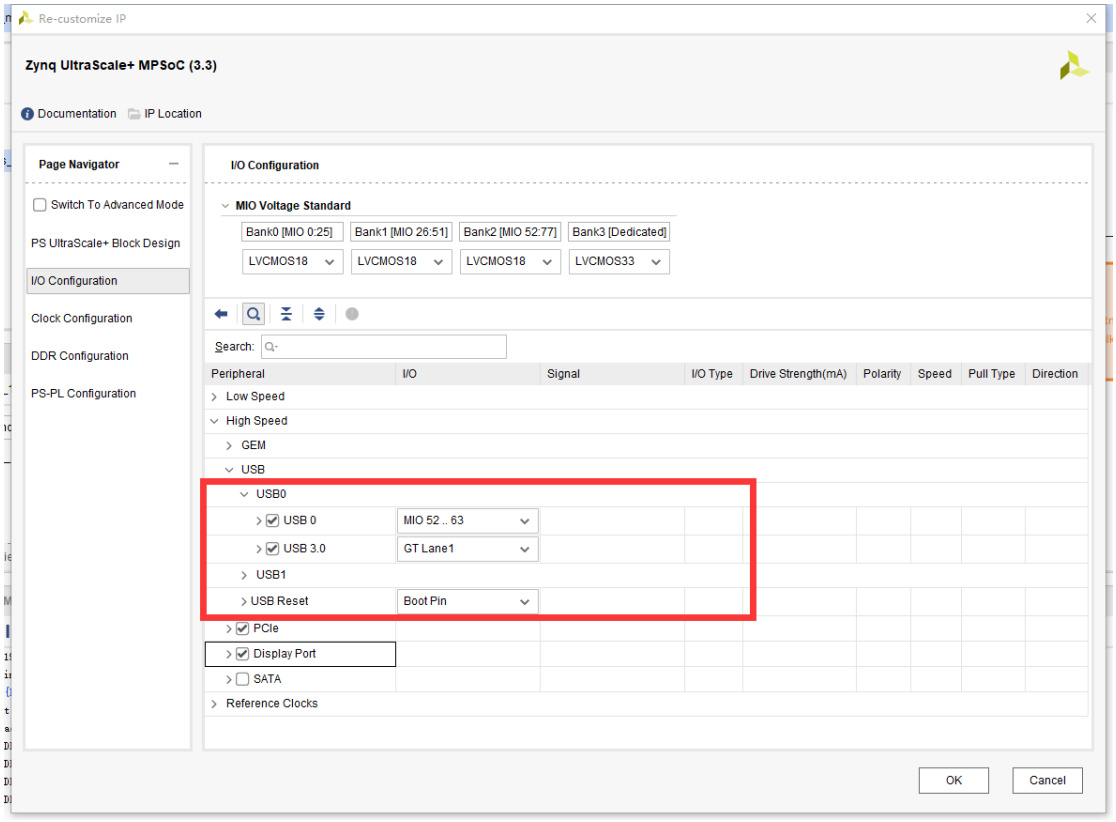


3、High Speed configuration

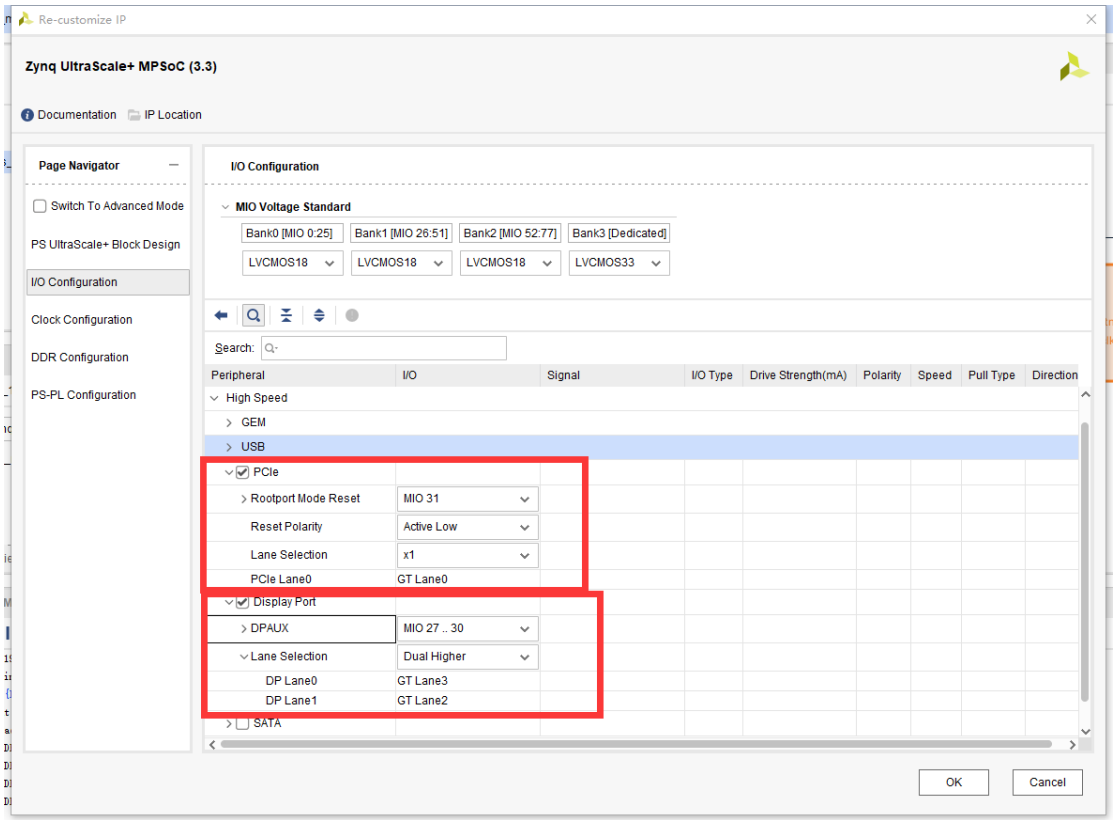
The High Speed part first configures PS Ethernet，tick GEM 3，select MIO 64..75，tick MDIO3，select MIO 76..77



tick USB 0，select MIO 52..63，tick USB 3.0，select GT Lane1



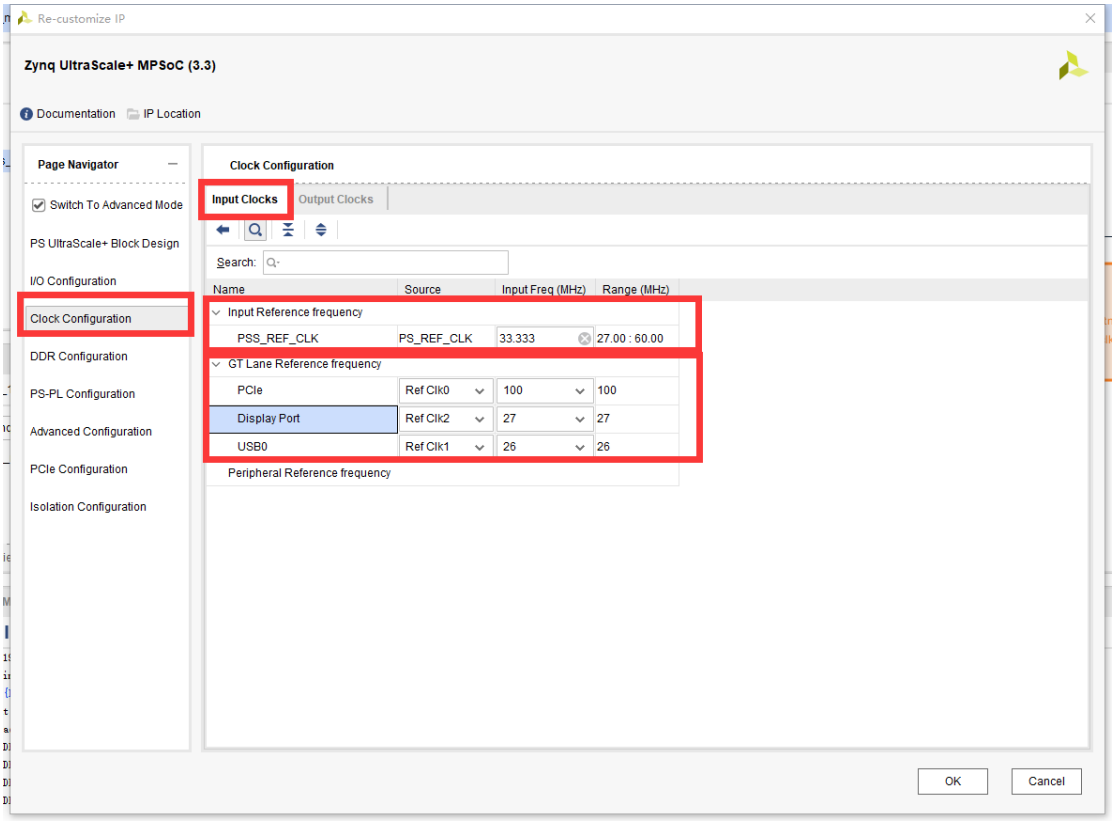
Tick PCIe，Lane Selection select x1，Reset select MIO 31； tick Display Port，DPAUX select MIO 27..30， Lane Selection select Dual Higher



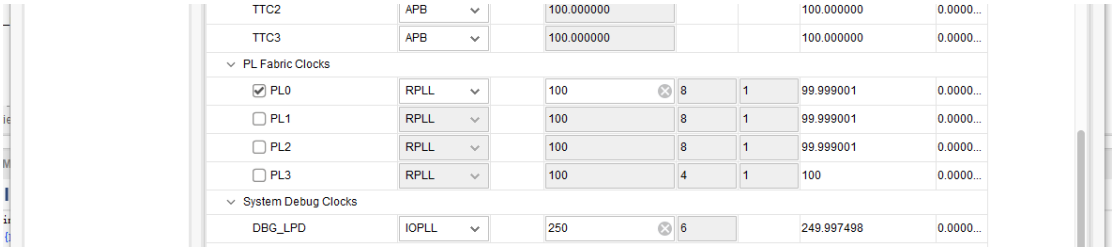
At this point, the IO part of the configuration is complete.

4. Clock configuration

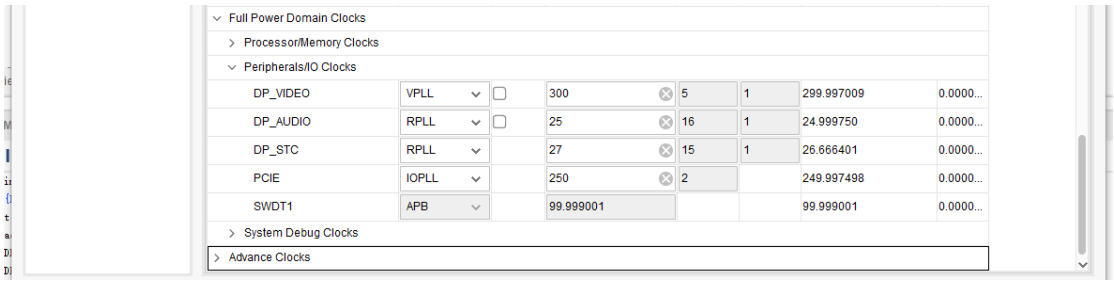
In the Clock Configuration interface， Input Clocks window configures reference clock，Among them, PSS_ REF_ Clock is the reference clock of arm, which is 33.333MHZ by default；PCIe select Ref Clk0，100MHz；Display Port select Ref Clk2，27MHz；USB0 select Ref Clk1， 26MHz。



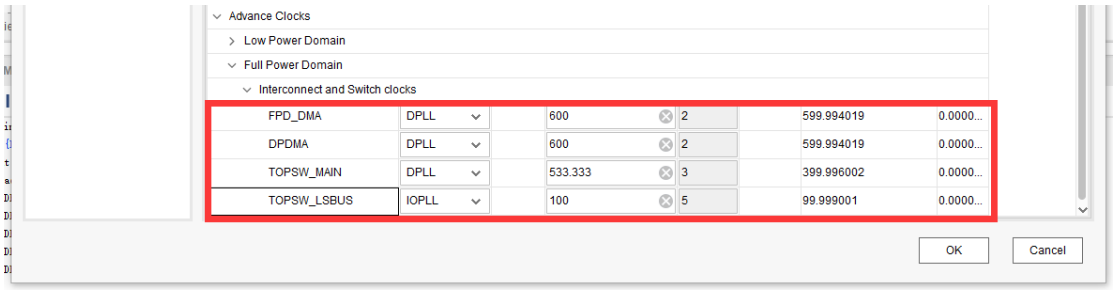
The clock of PL remains the default, which is the clock provided for the PL logic.



For the Full Power part, the others will remain the default, and the DP_VIDEO will be changed to VPLL, DP_ Audio and DP_ STC changed to RPLL.



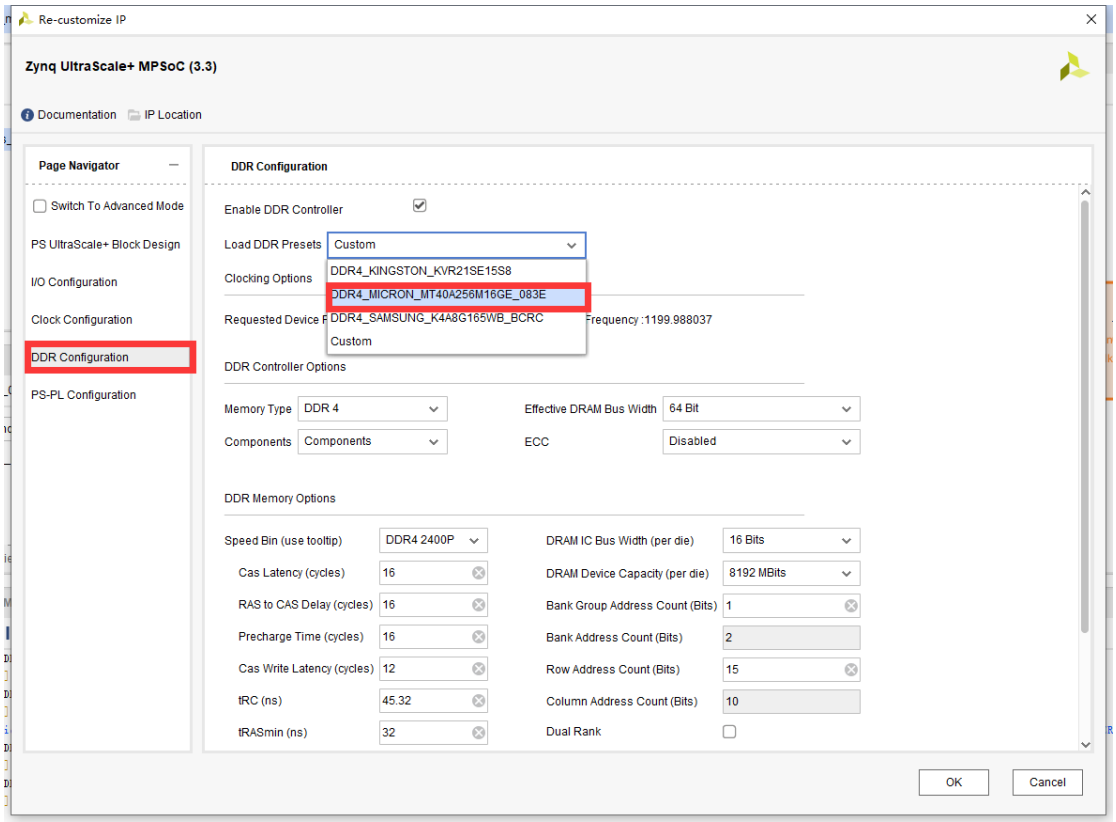
The bottom interconnect is modified as follows



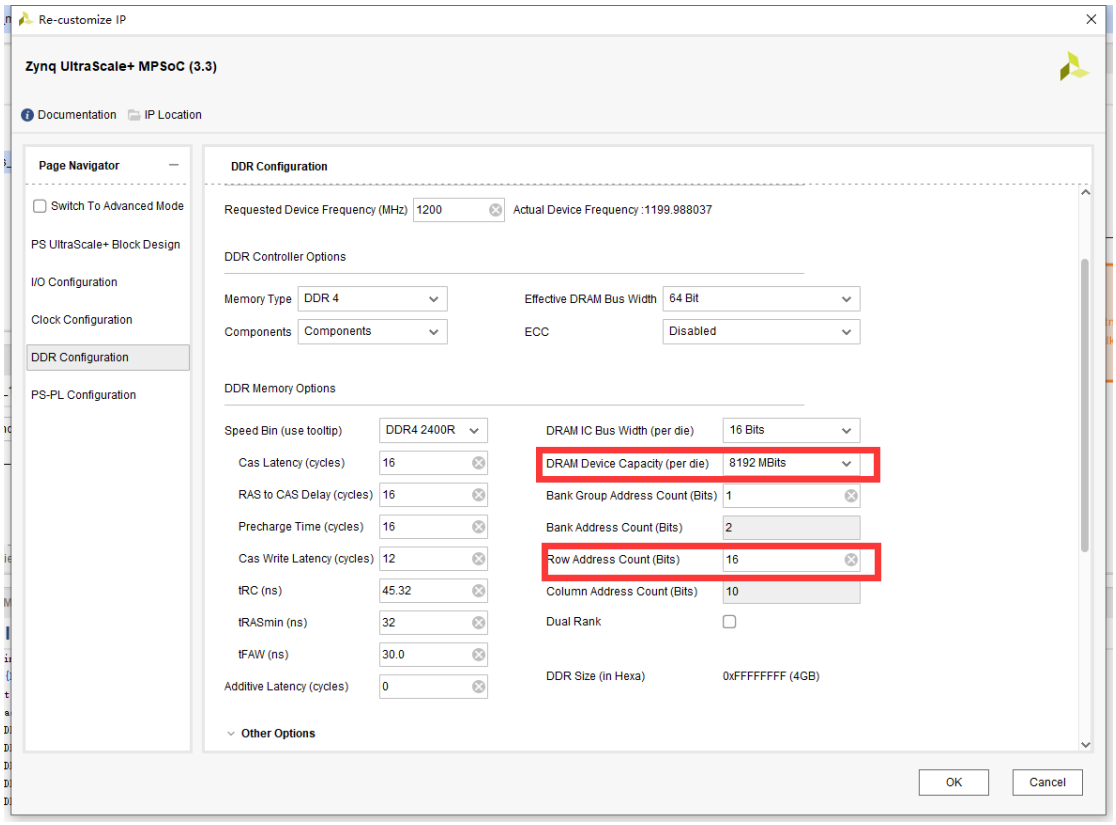
The other parts remain the default, so far, the clock part is configured.

5、DDR Configuration

In the DDR Configuration window， Load DDR Presets select “DDR4_MICRON_MT40A256M16GE_083E”

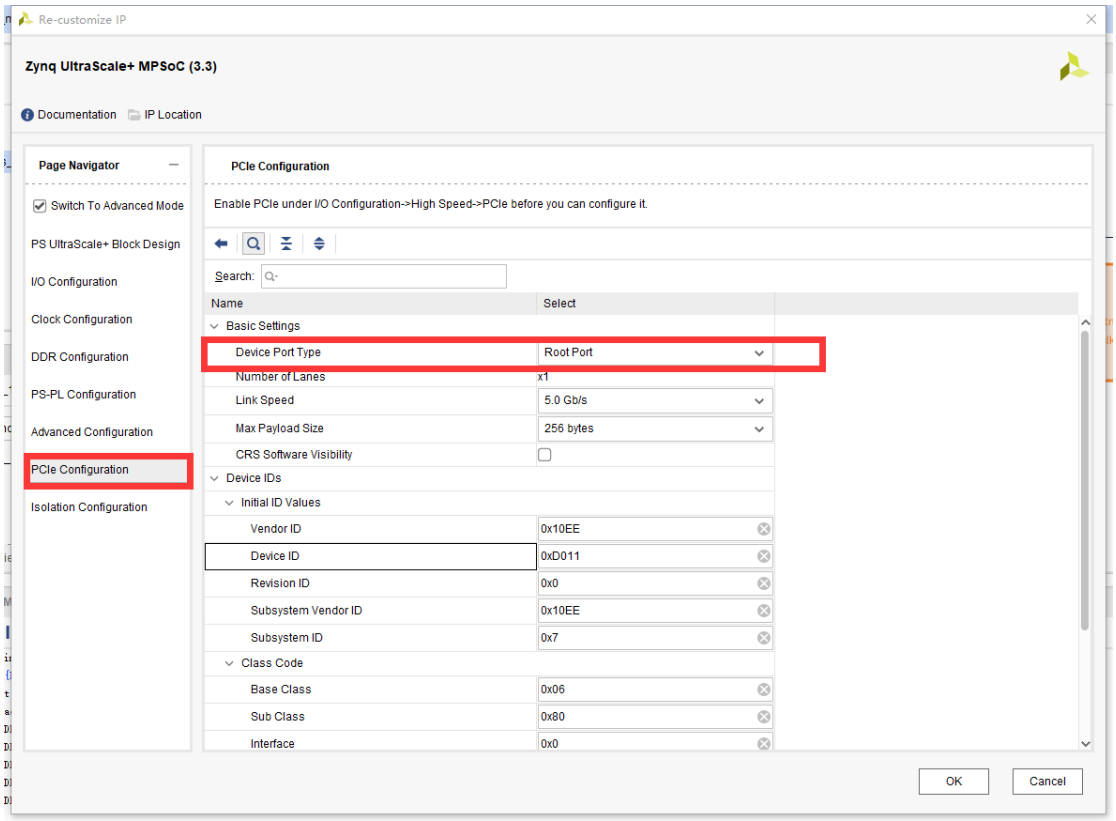


The parameters are modified as follows

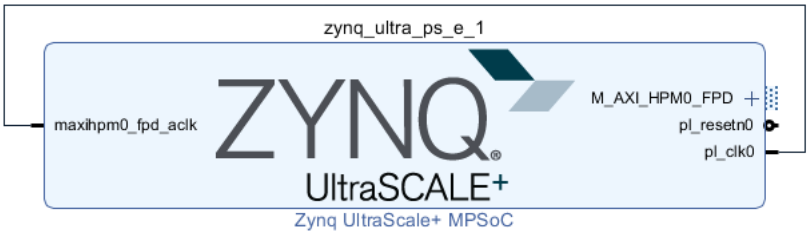


6、PCIe Advanced Configuration

Click on the Switch To Advanced Mode，select PCIe Configuration，DEVICE PORT TYPE select ROOT PORT。

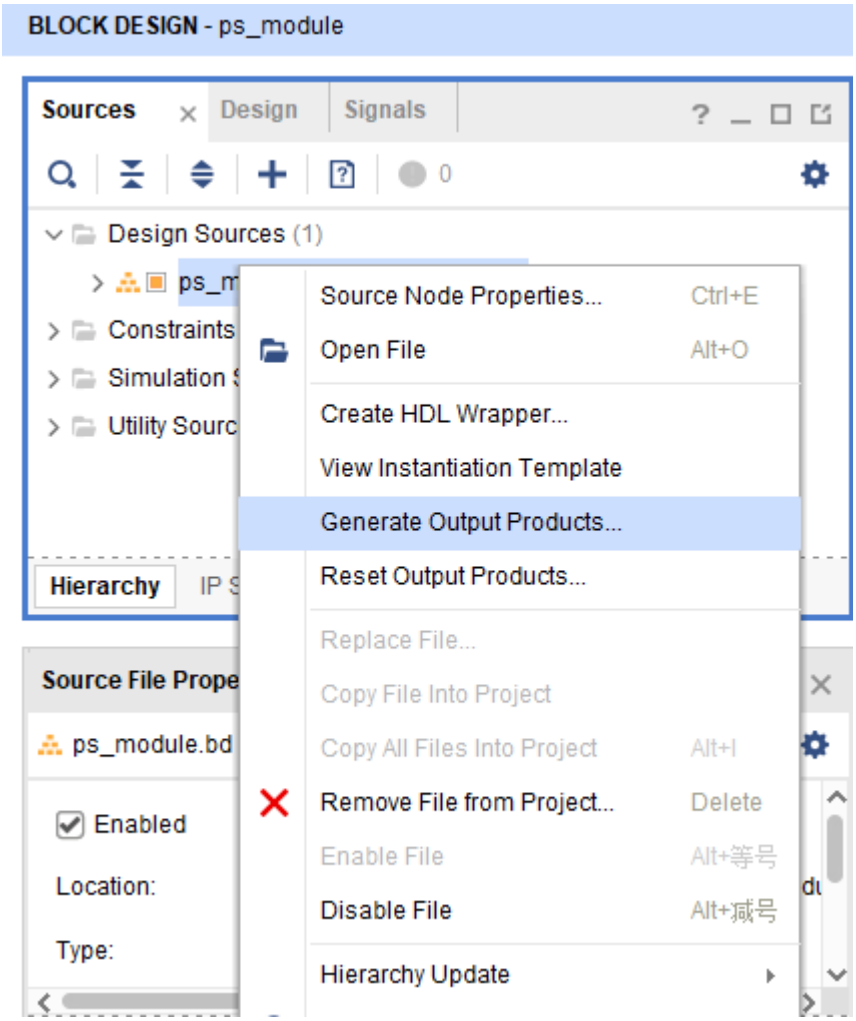


Click OK to complete the configuration and connect the clock as follows:

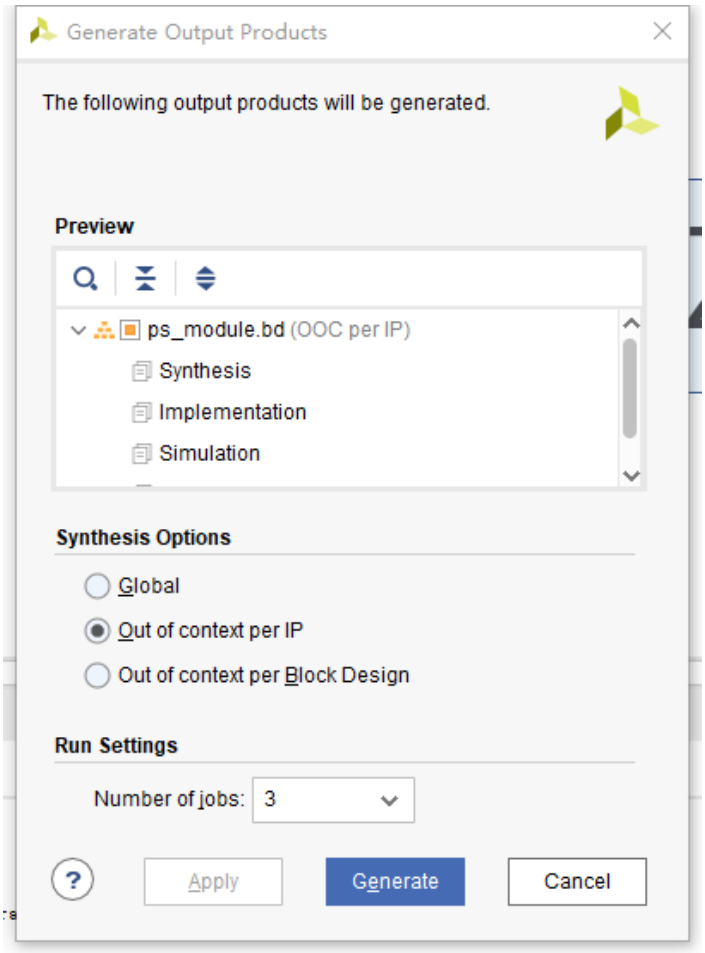


1.4 Generate synthesis files

Right click ps_module->Generate Ouput Products->Generate

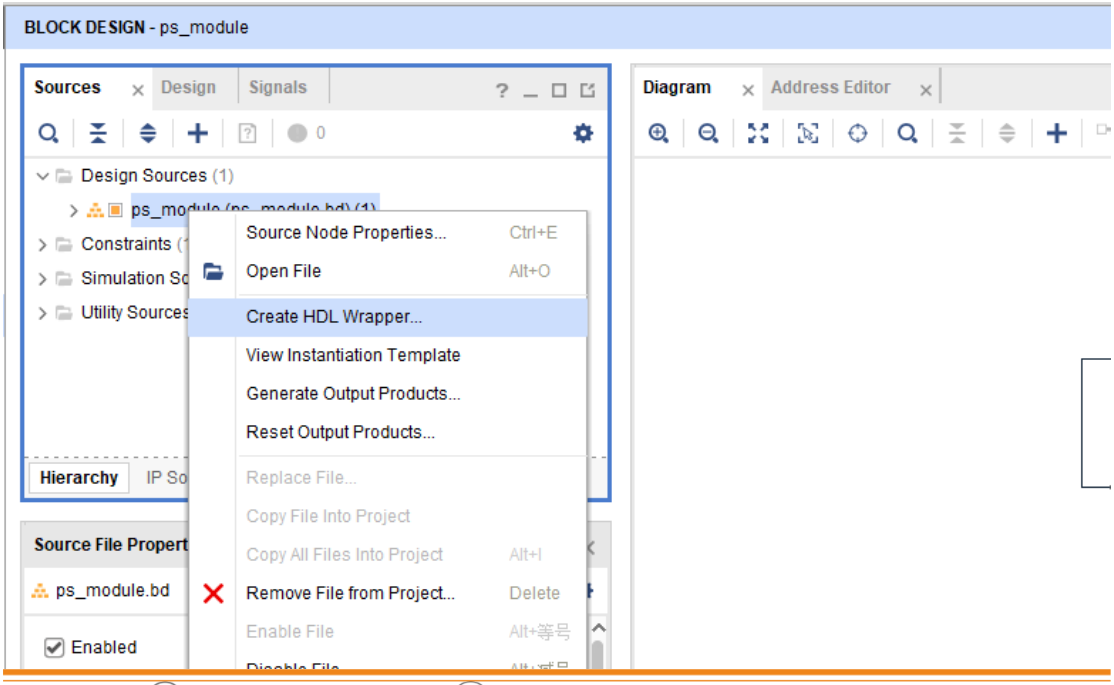


Click Generate

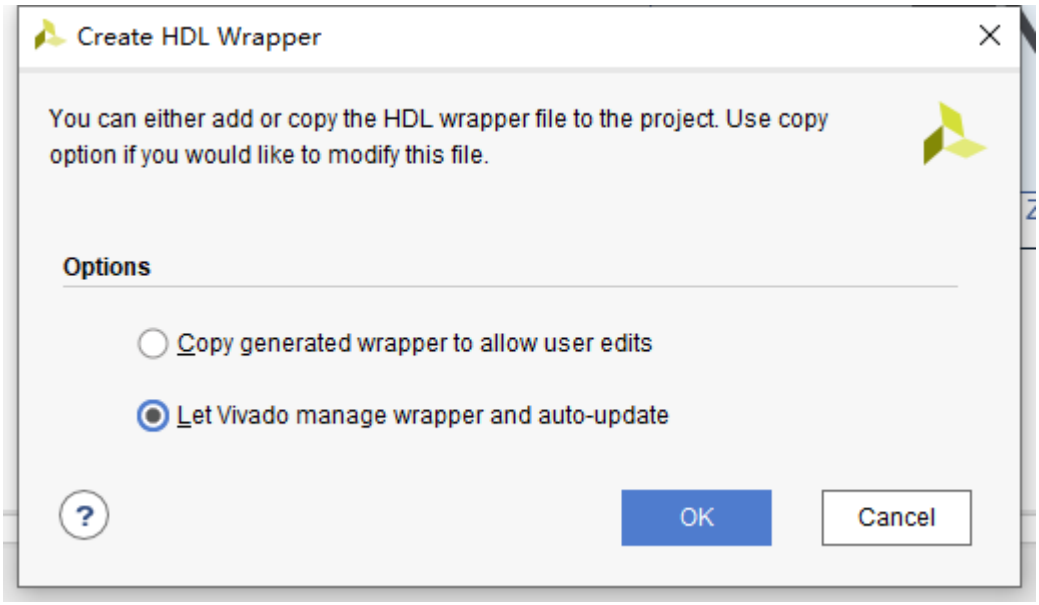


1.5 Generating top-level file of FPGA

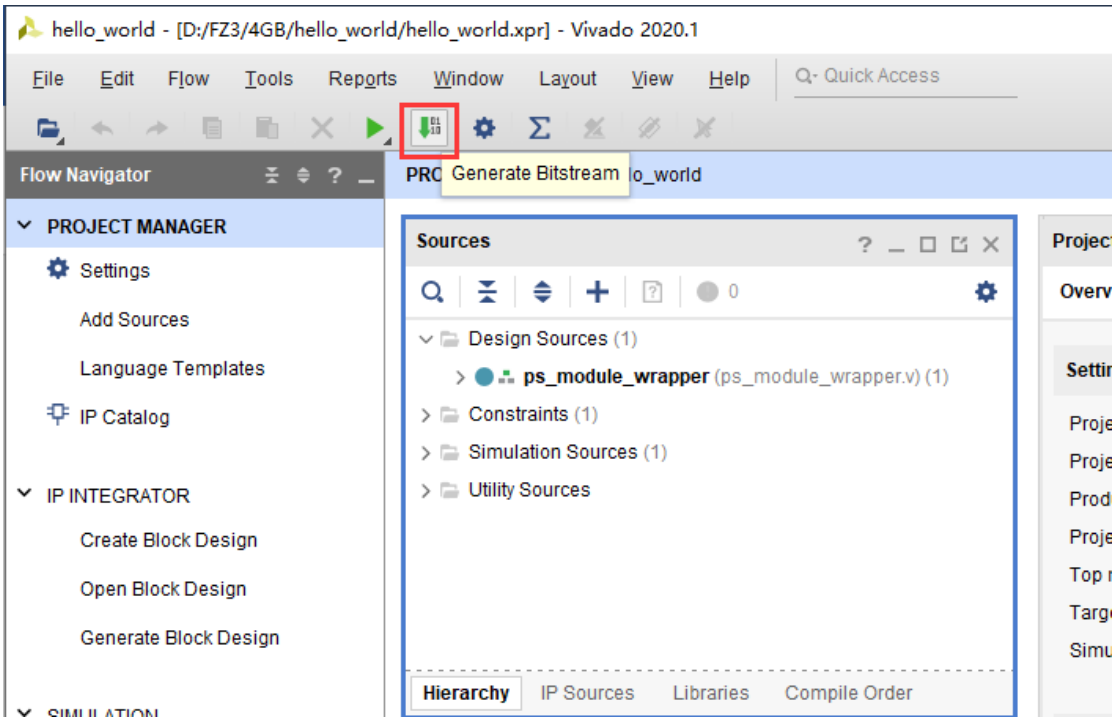
Select ps_module.bd, Right click --CreateHDL Wrapper



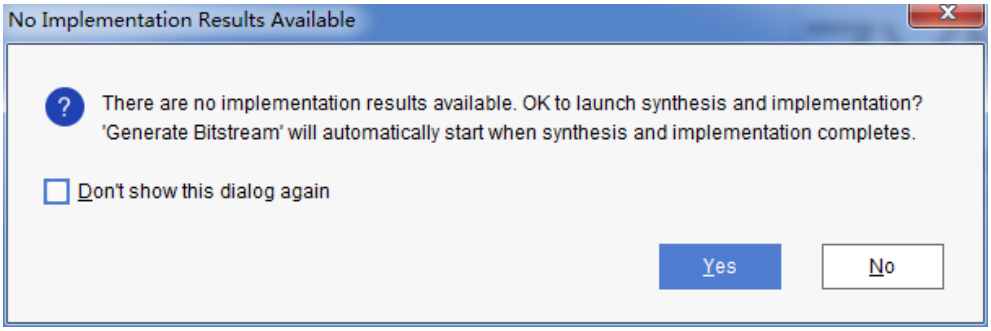
Click OK



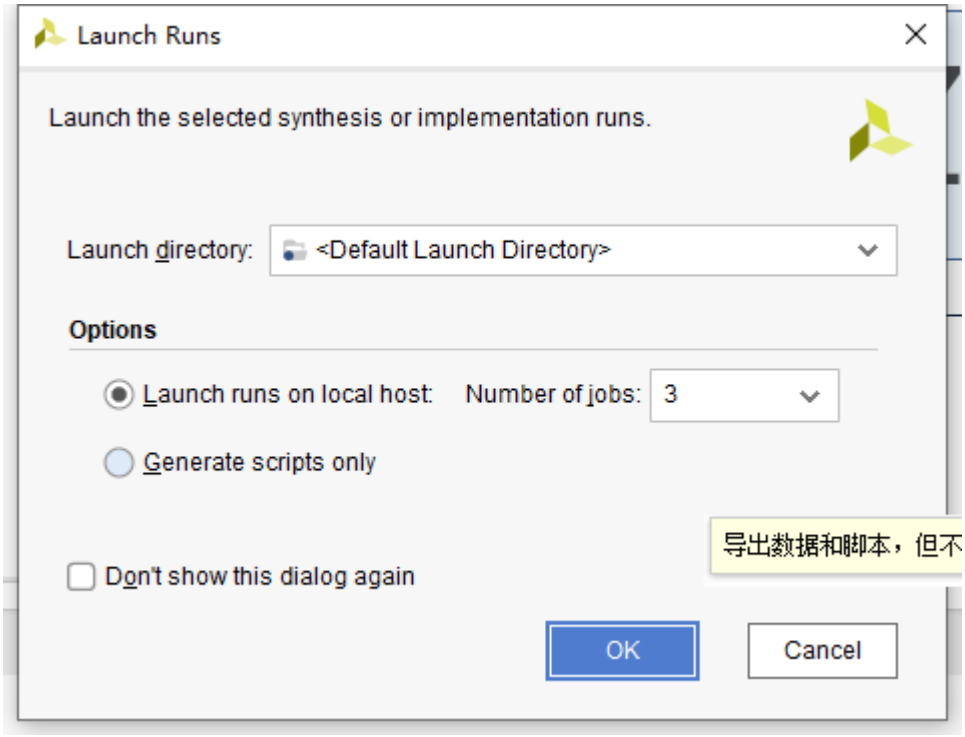
1.6 Generate bit files



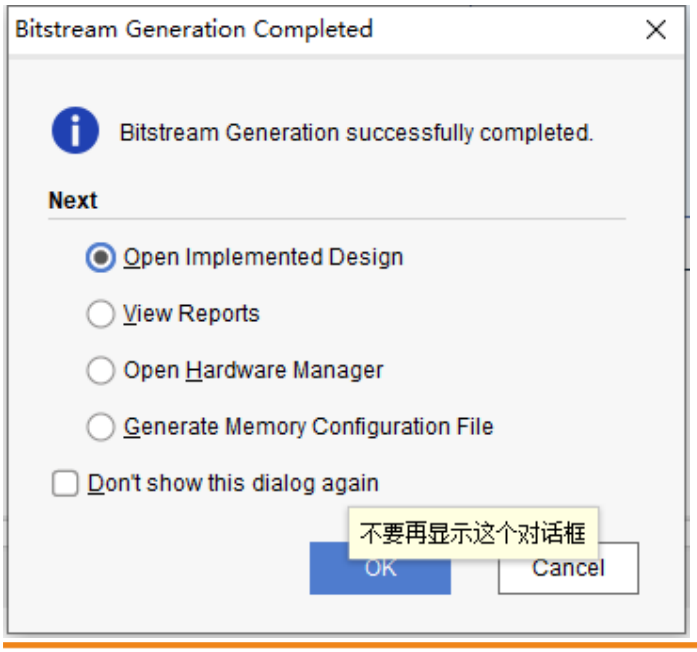
Click Yes



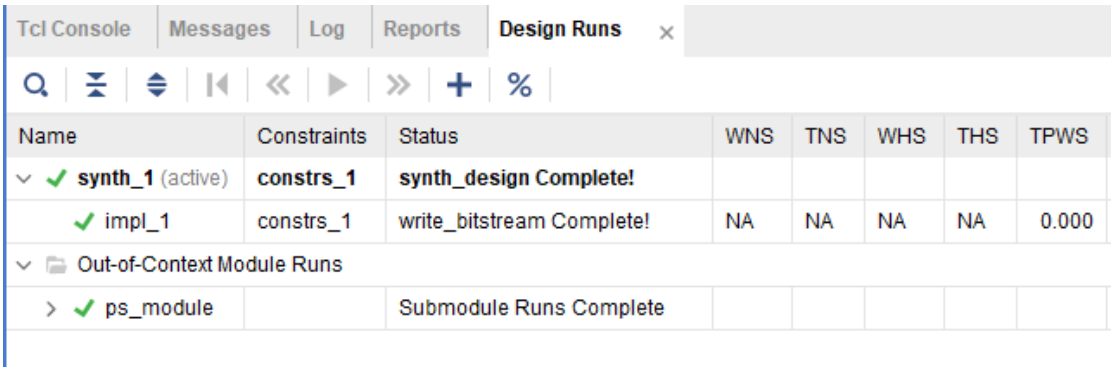
Click OK



Click Cancel

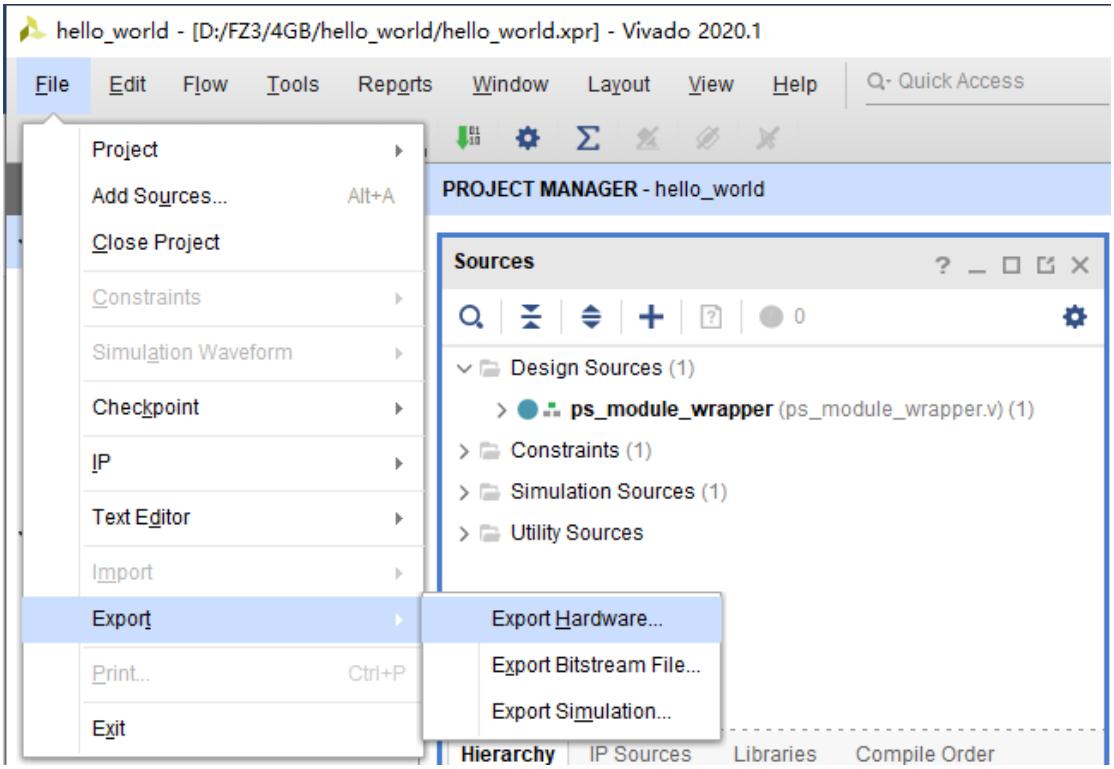


You can see that bitstream compiled successfully

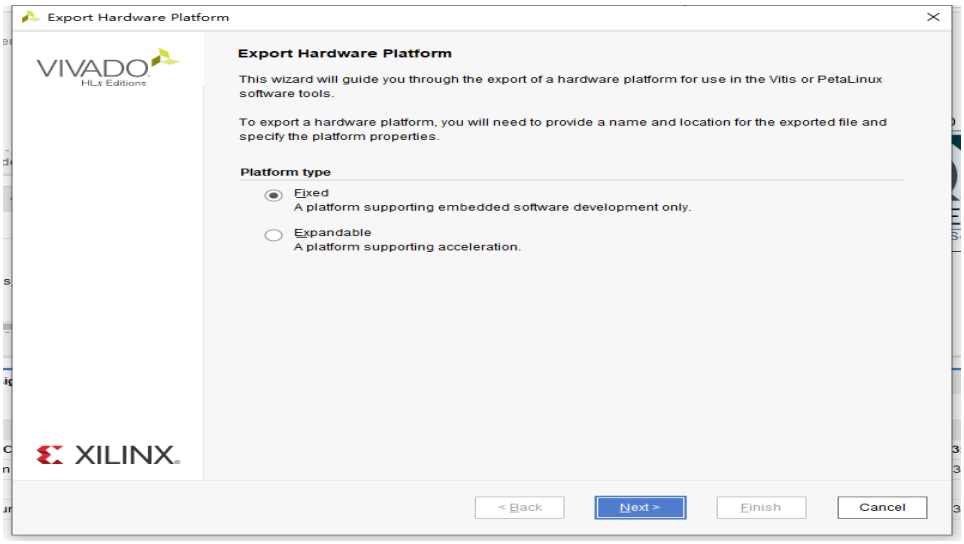


1.7 Export Hardware Profile

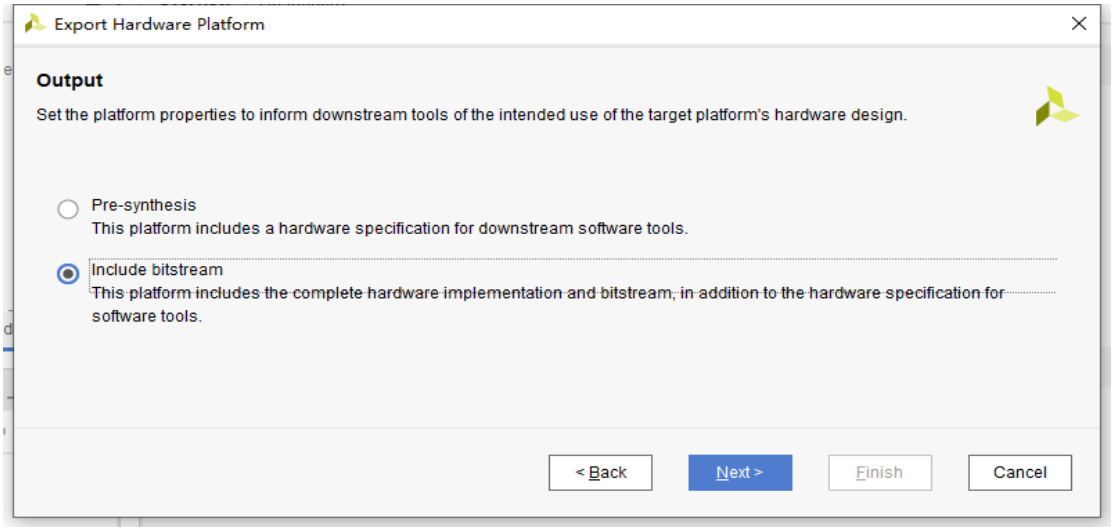
Click File->Export->Export Hardware->OK on the menu bar to export the hardware configuration file



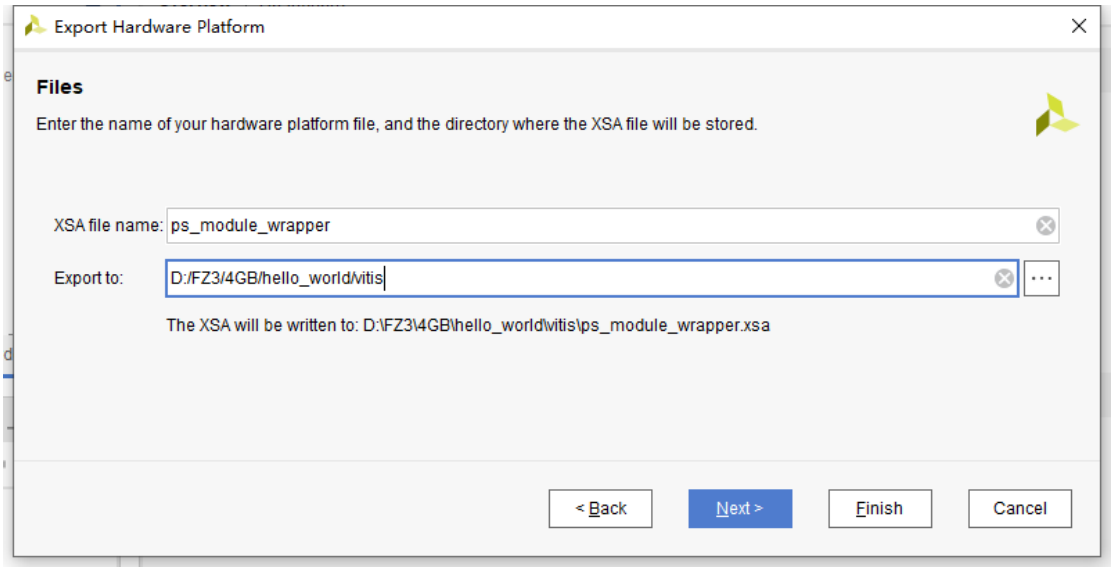
Select fixed，click Next



Select include bitstream，click Next

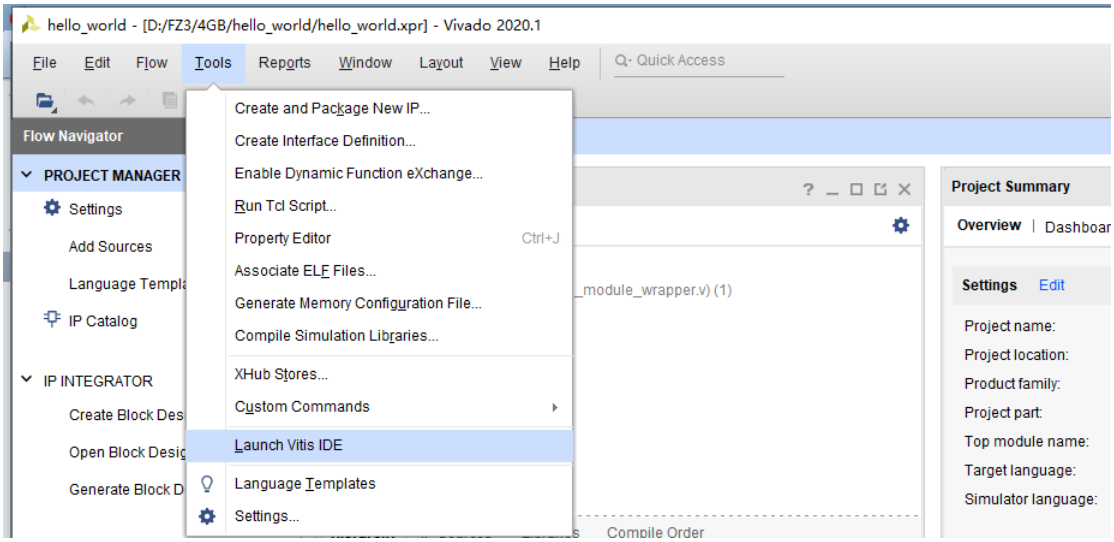


Select the generated xsa file name and file path, where the xsa name remains unchanged, and the path is the vitis subfolder under the original path.

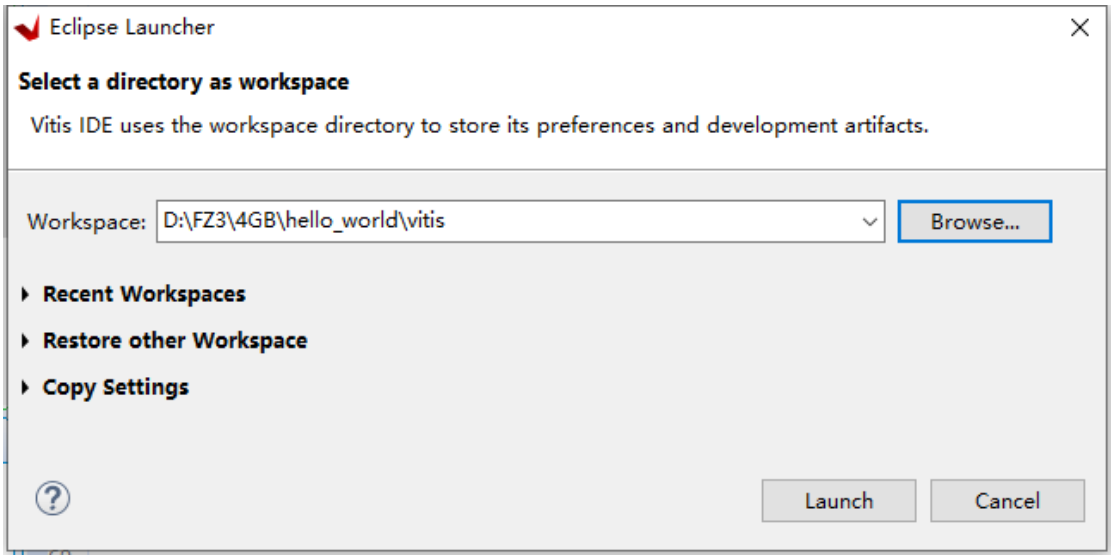


1.8 Start Vitis and create new project

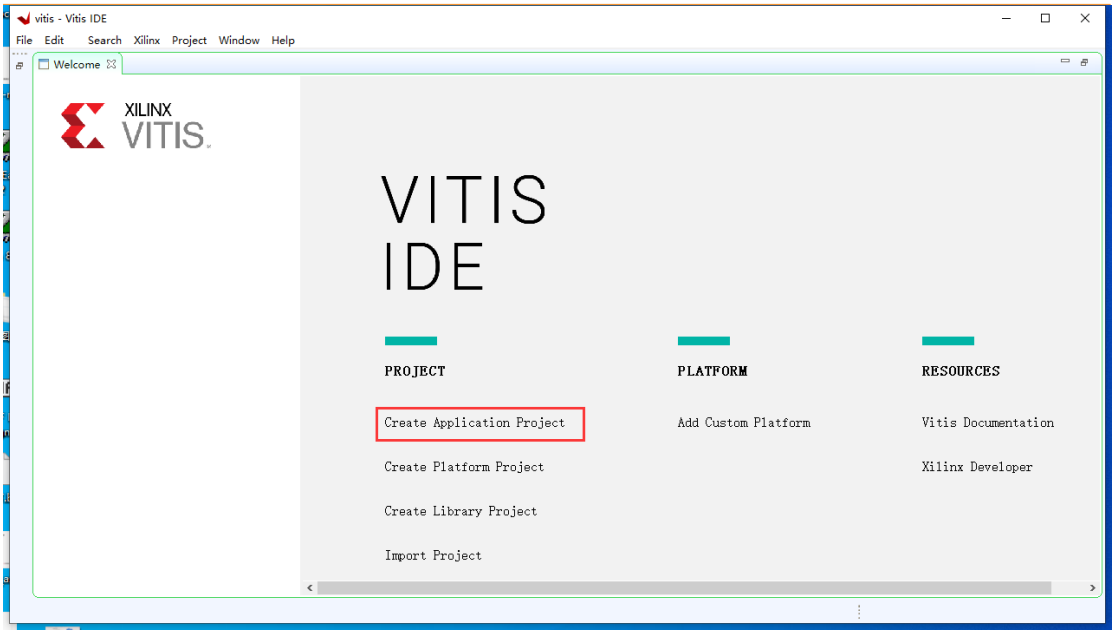
Vitis is an independent software, which can be opened by double clicking the vitis software, or by selecting Tools -- Launch Vitis in vivado software to open Vitis software



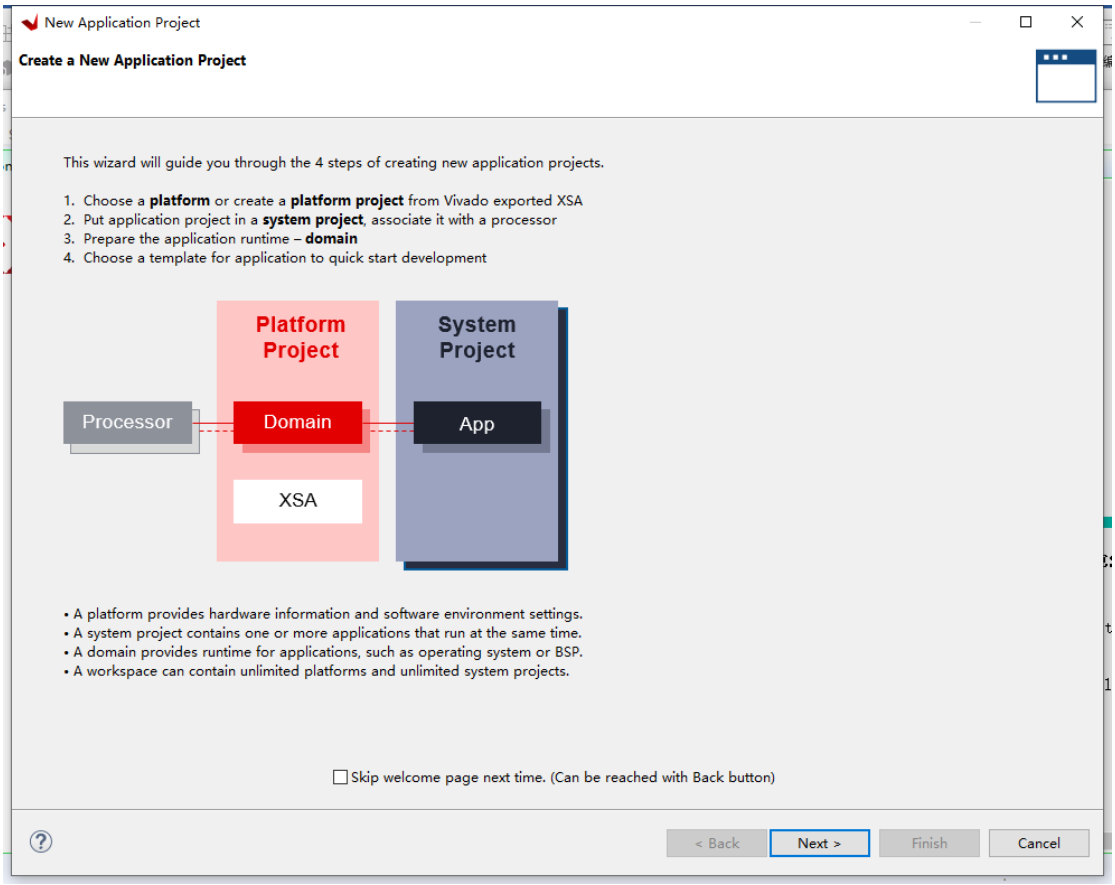
Select the previously created vitis folder and click "Launch"



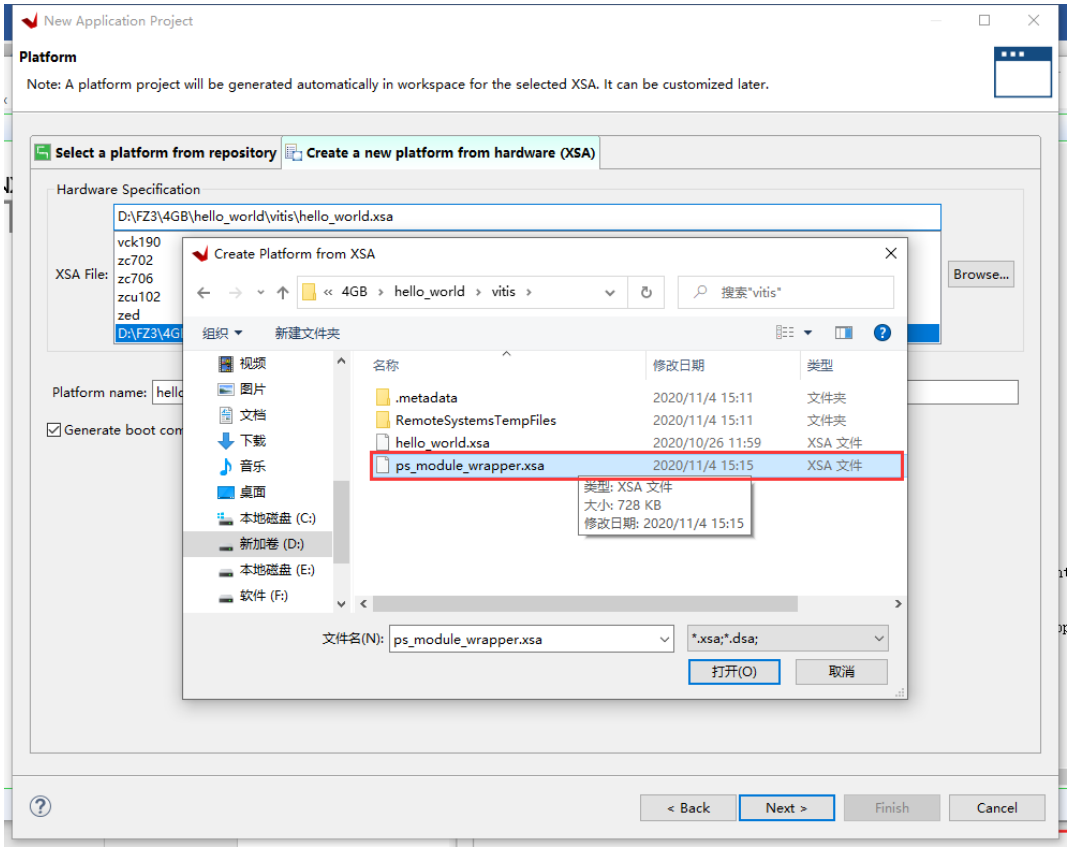
After starting Vitis, the interface is as follows.Click "Create Application project". This option will generate app project and platform from project. The platform project is similar to the hardware platform of previous versions, including relevant files of hardware support and BSP.



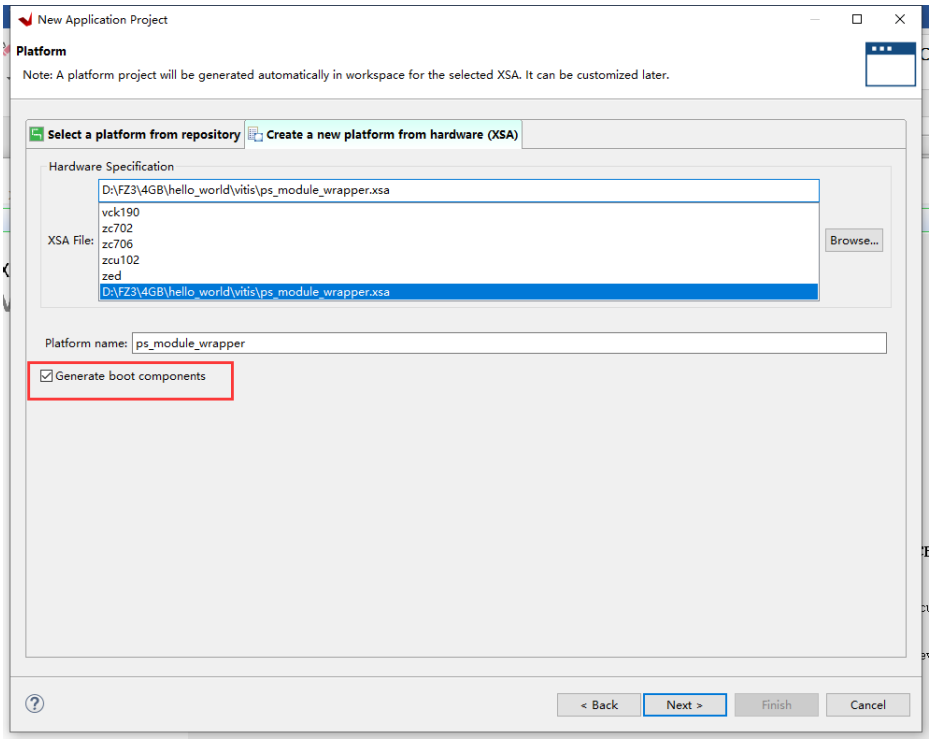
The first page is the introduction page. Skip it directly and click next



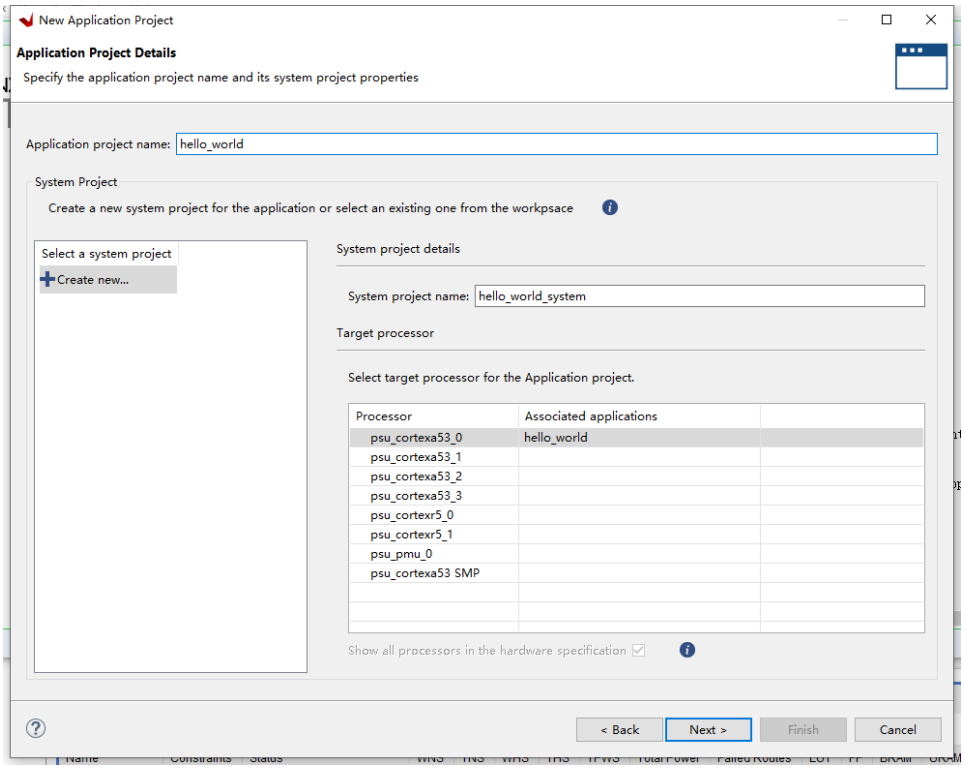
Select “Create a new platform from hardware(XSA)”，open “Browse”， Select the xsa file you just exported.



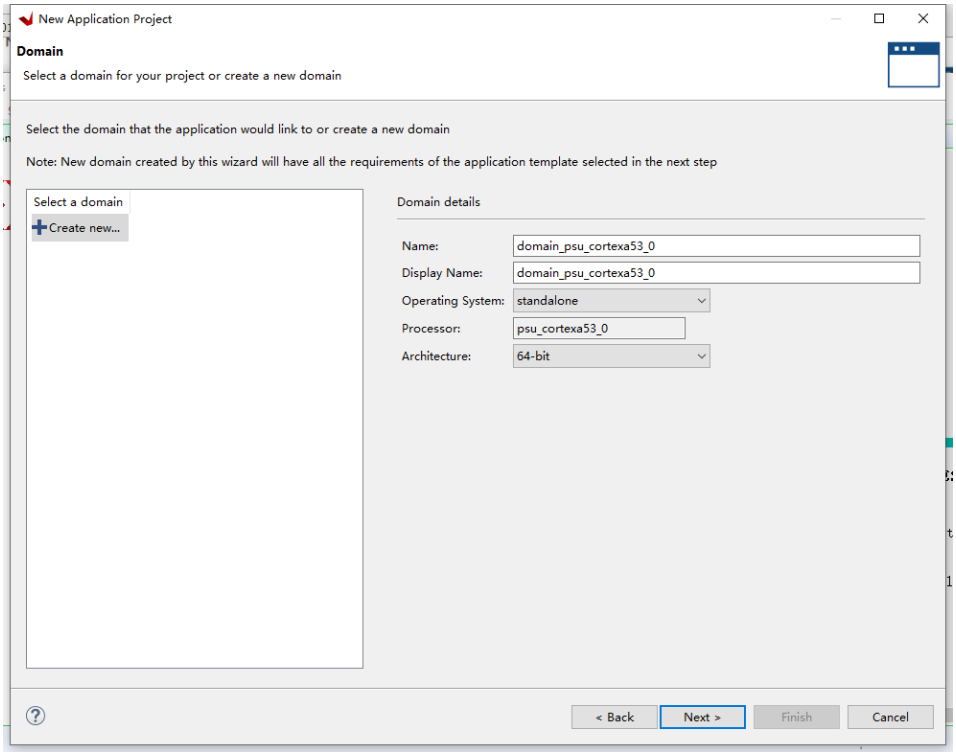
At the bottom is the Generate boot components option. If this option is clicked, the software will automatically generate the fsbl project. We generally choose to check it by default.



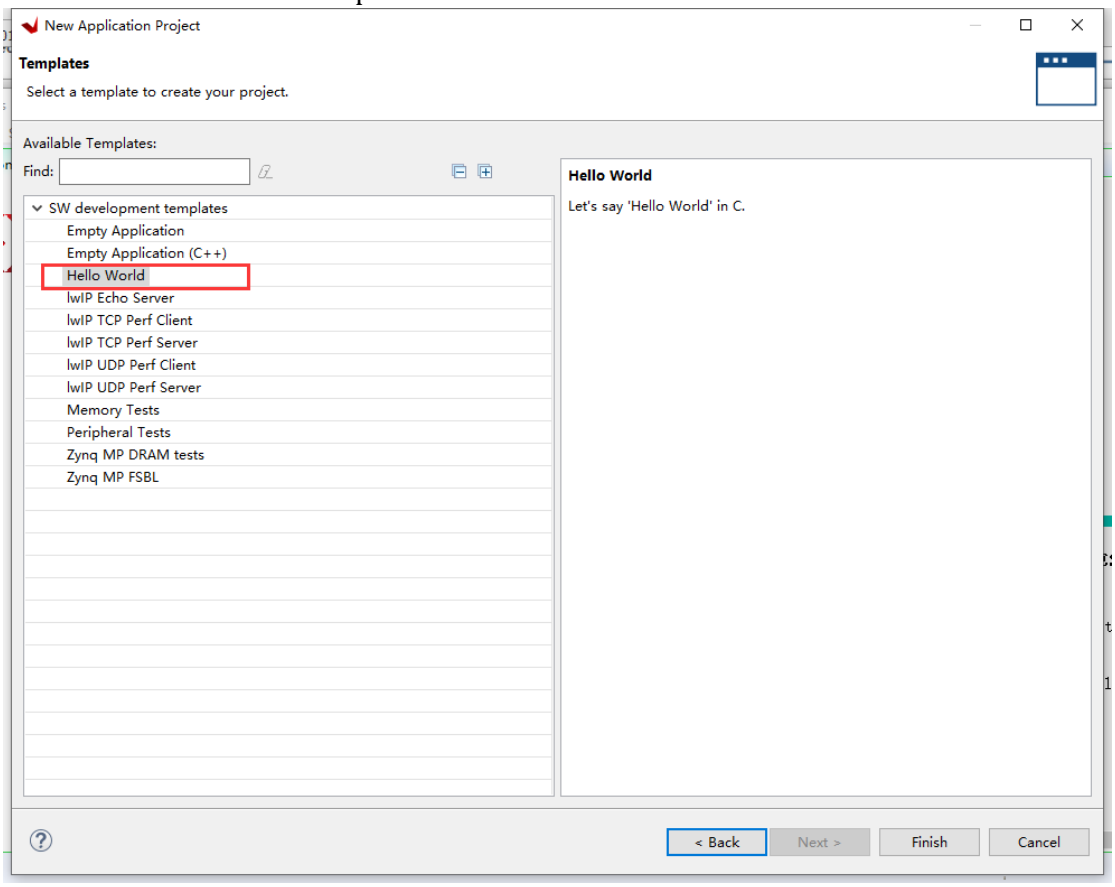
Fill in the app project name hello_ world. Click in the box to select the corresponding processor. We will keep the default here. (note that the app project name cannot be the same as the platform project name, otherwise an error will occur)



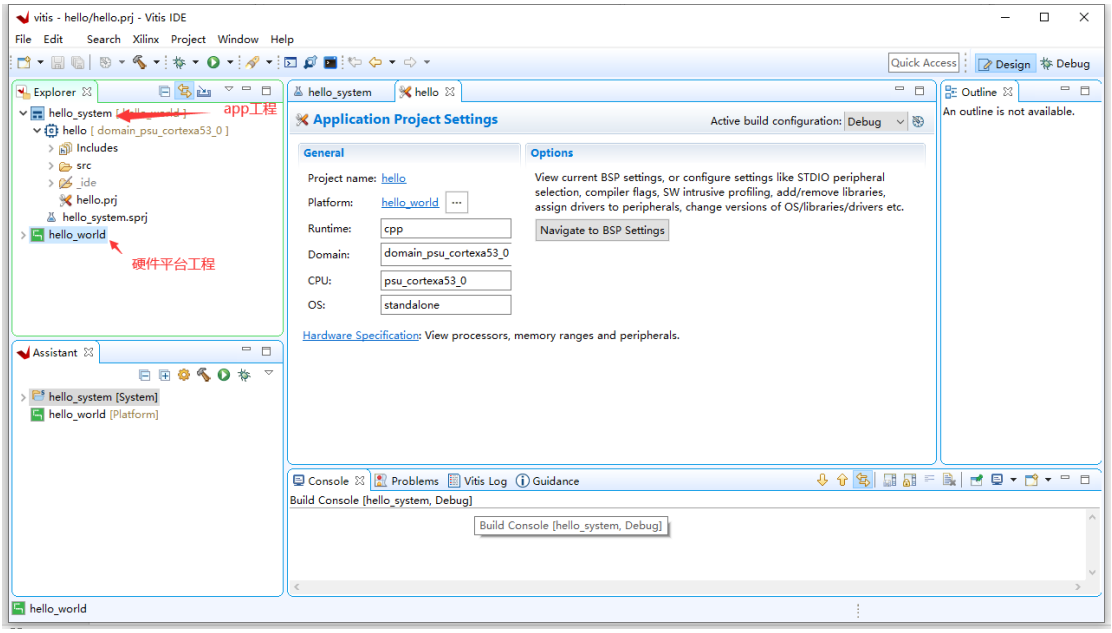
In the following interface, you can modify the domain name, select the operating system, arm architecture, etc. here, the default is maintained, and the operating system is selected as stand-alone, that is, bare machine.



Select the "Hello world" template and click "finish" to finish.

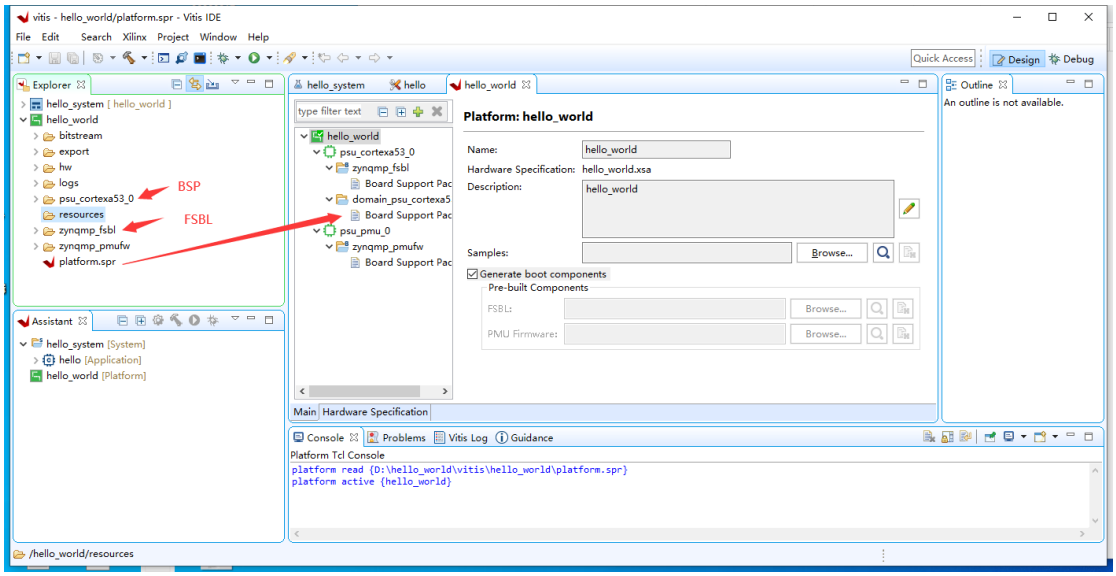


After completion, we can see that two projects have been generated, one is the hardware platform project, that is, the platfrom project mentioned before, and the other is app project.

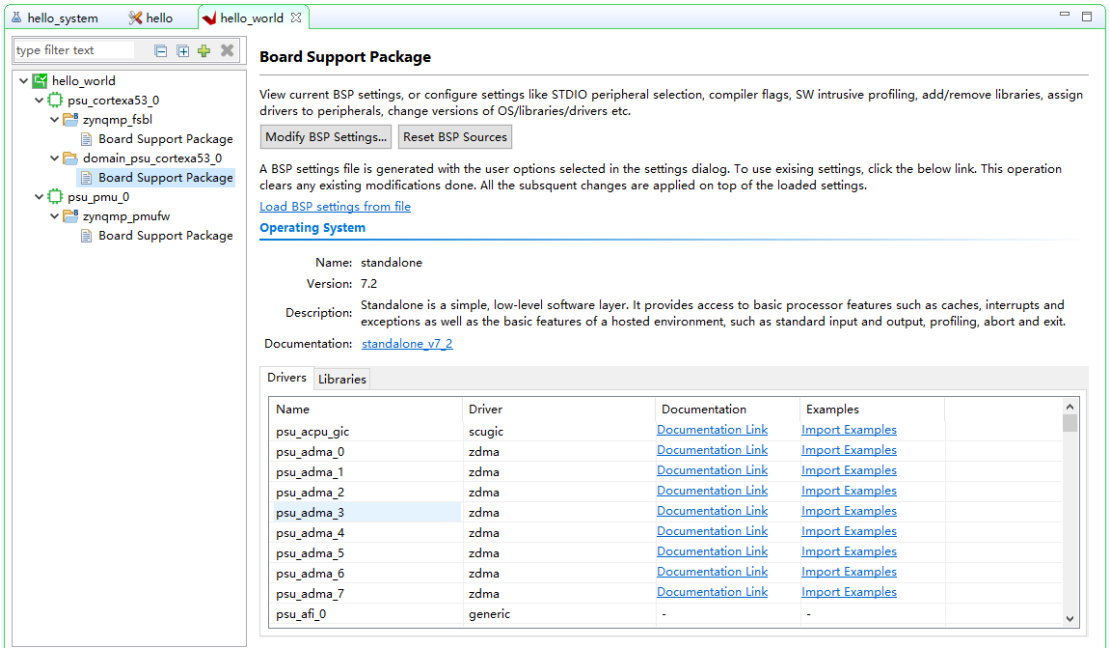


After you expand the platform project, you can see that there are BSP projects and zynq_ Fsbl project (the result of selecting generate boot components), double-click platform.spr You can see the BSP project generated by the platform. BSP means board

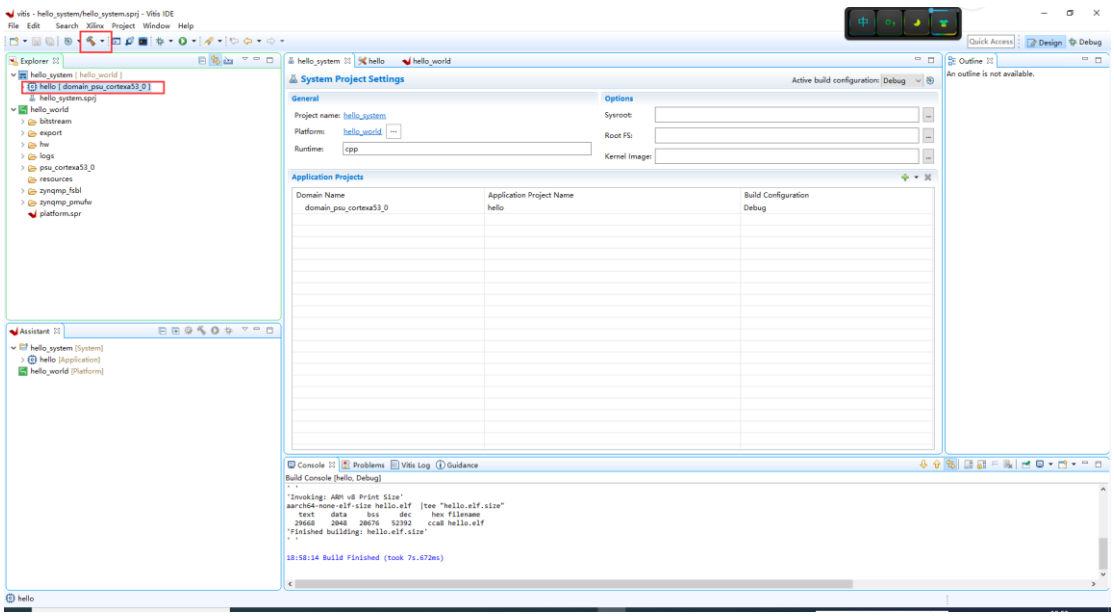
support package, which contains the driver files needed for development, which is used for application development.



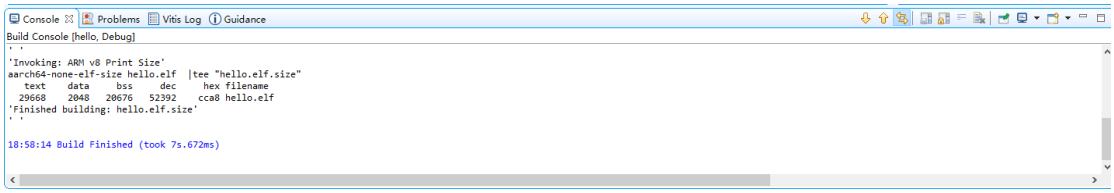
Click BSP to see the peripheral drivers of the project. Documentation Link is the description document of the driver provided by Xilinx, and Import Examples is the example project provided by Xilinx to speed up learning.



Select the App project, right-click Build Project, or click the "hammer" button in the menu bar to compile the project

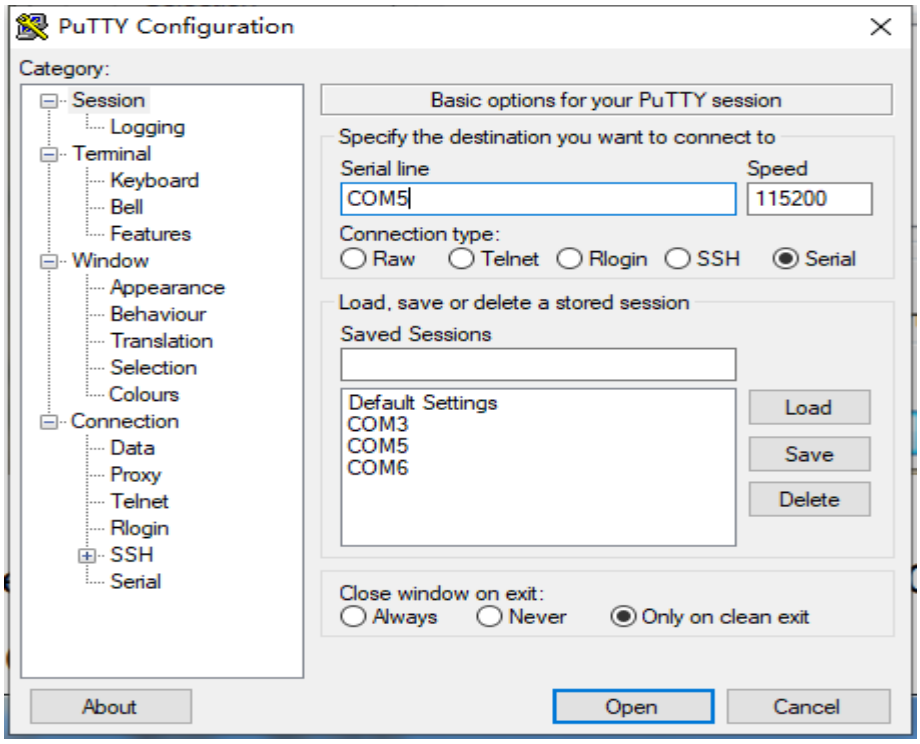


After compiling, an elf file is generated



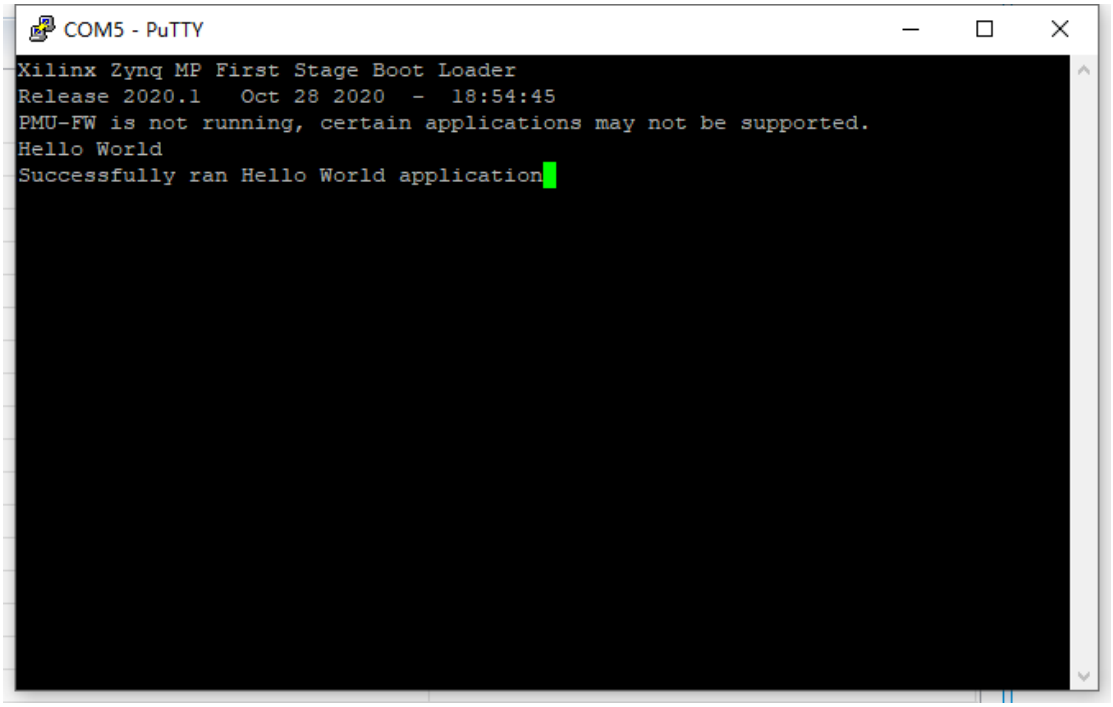
Connect JTAG cable to development board and UART USB cable to PC

Using putty as a debugging tool for serial port



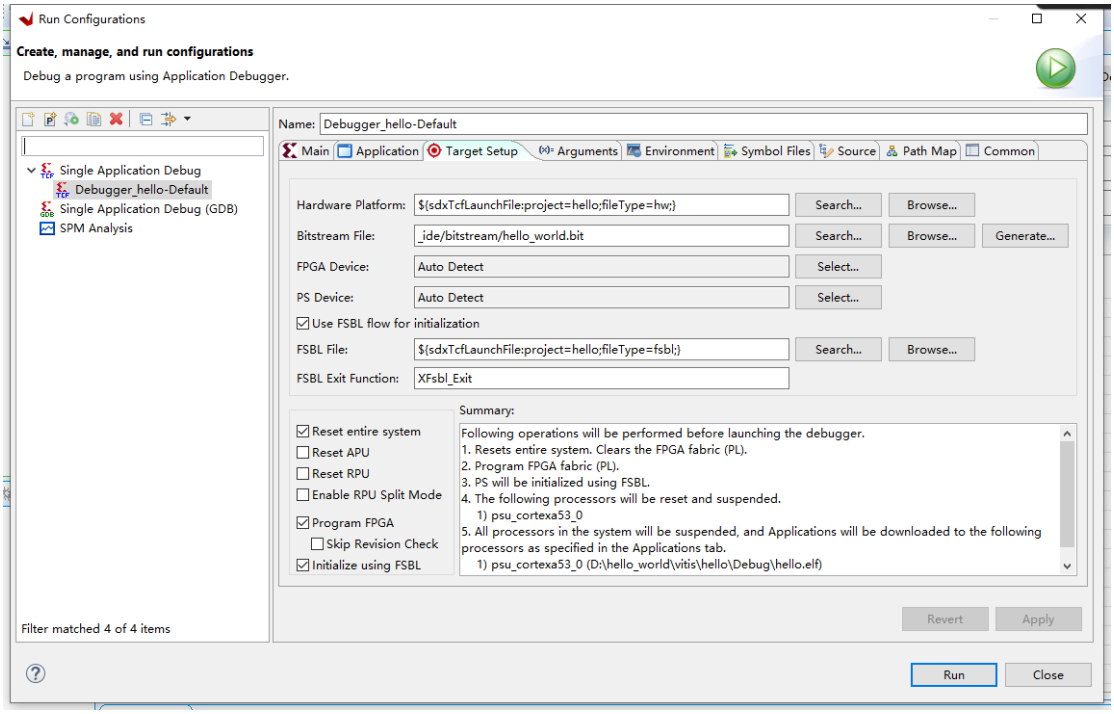
选择“hello”，右键，可以看到很多选项，本实验要用到这里的“Run as”，就是把程序运行起来，“Run as”里又有很对选项，选择第一个“Launch on Hardware(Single Application Debug)”，使用系统调试，直接运行程序。Select "hello" and right-click to see many options.

In this experiment, "Run as" is used to run the program. There are many options in "Run as". Select the first "Launch on Hardware (single application debug)" to run the program directly.



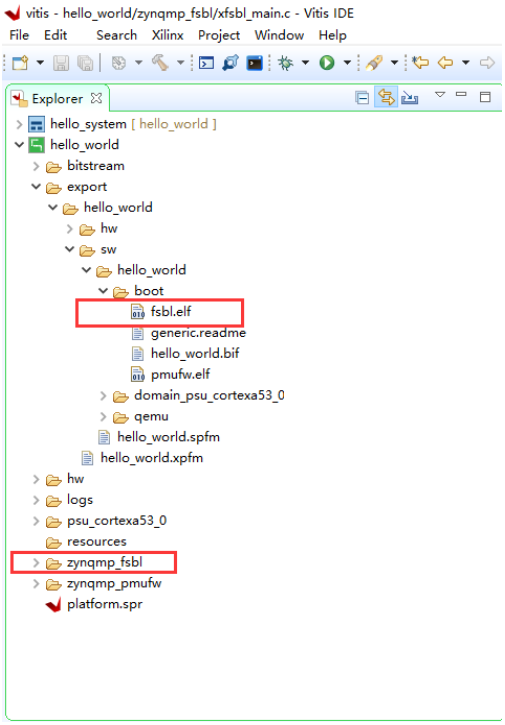
In order to ensure the reliable debugging of the system, it is best to right-click “Run As -> Run Configuration...”

We can take a look at the configuration. The Reset entire system is selected by default, which is different from the previous SDK software. If there is a pl design in the system, you must also select "program FPGA".

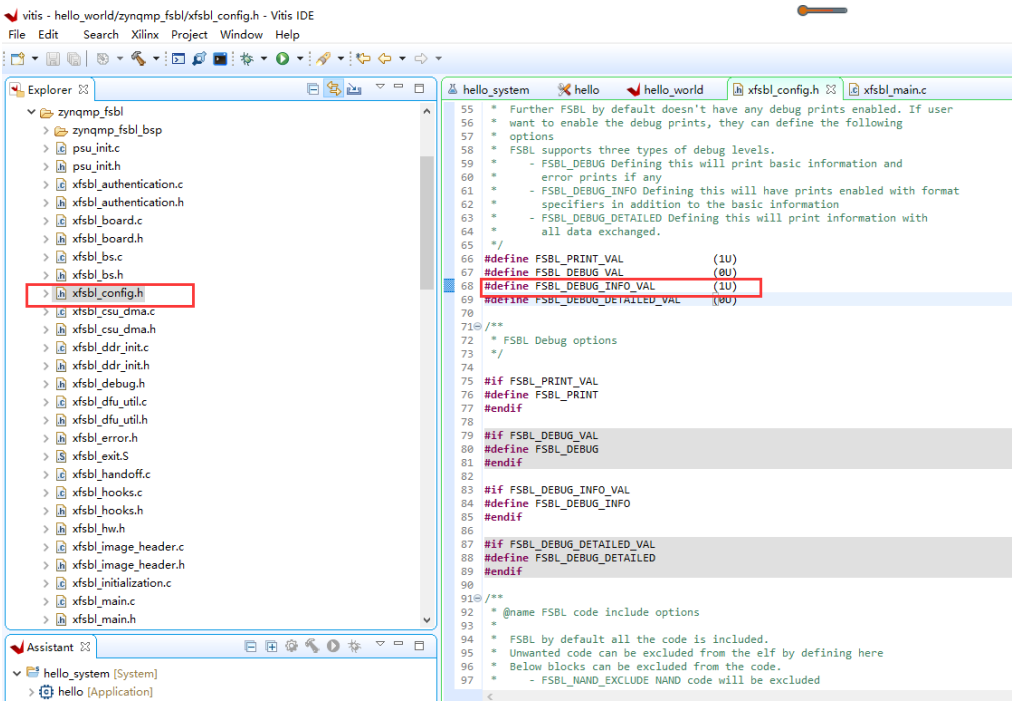


1.9 Generate BOOT.bin file

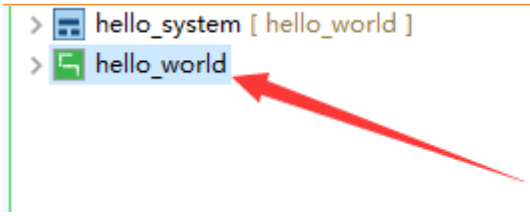
Since the Generate boot components option is selected when creating a new one, the platform has imported the fsbl project and generated the corresponding ELF file



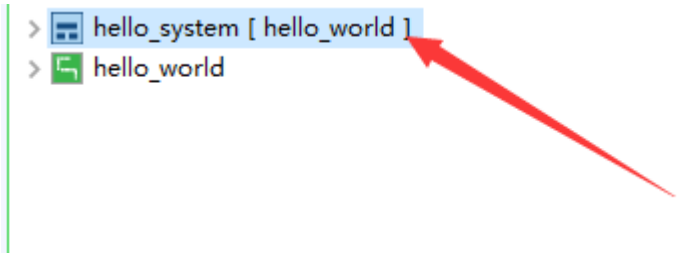
Modify debug macro definition fsbl_ DEBUG_ INFO_ When Val to 1, some status information of fsbl can be output at startup, which is conducive to debugging, but it will lead to longer startup time.



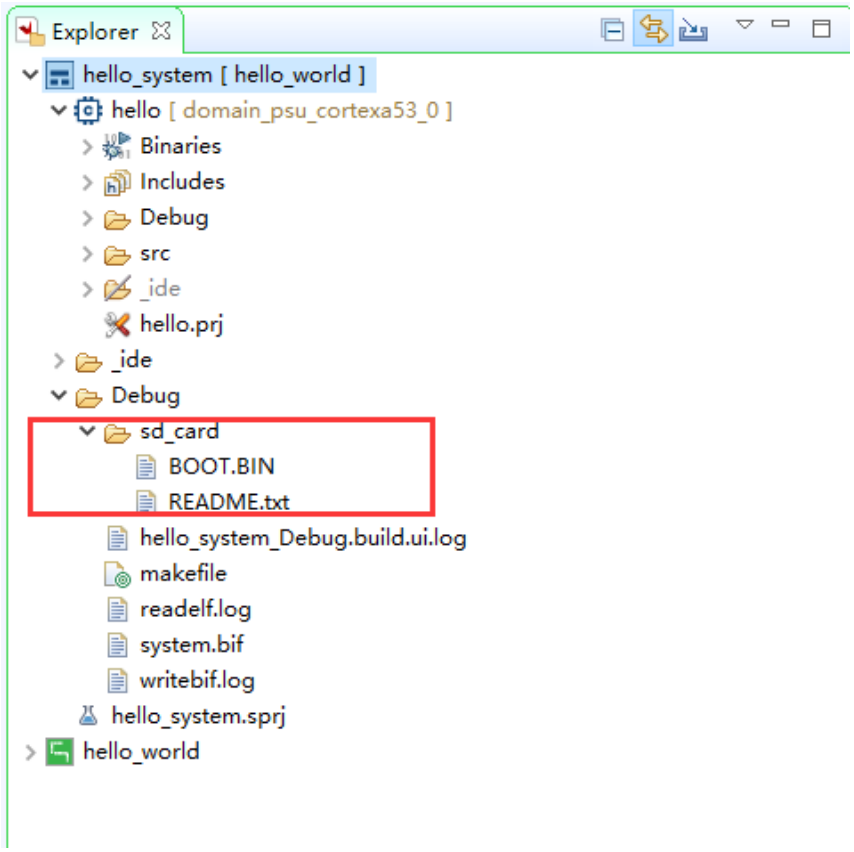
Right click the hardware platform project hello_ World, select Build Project.



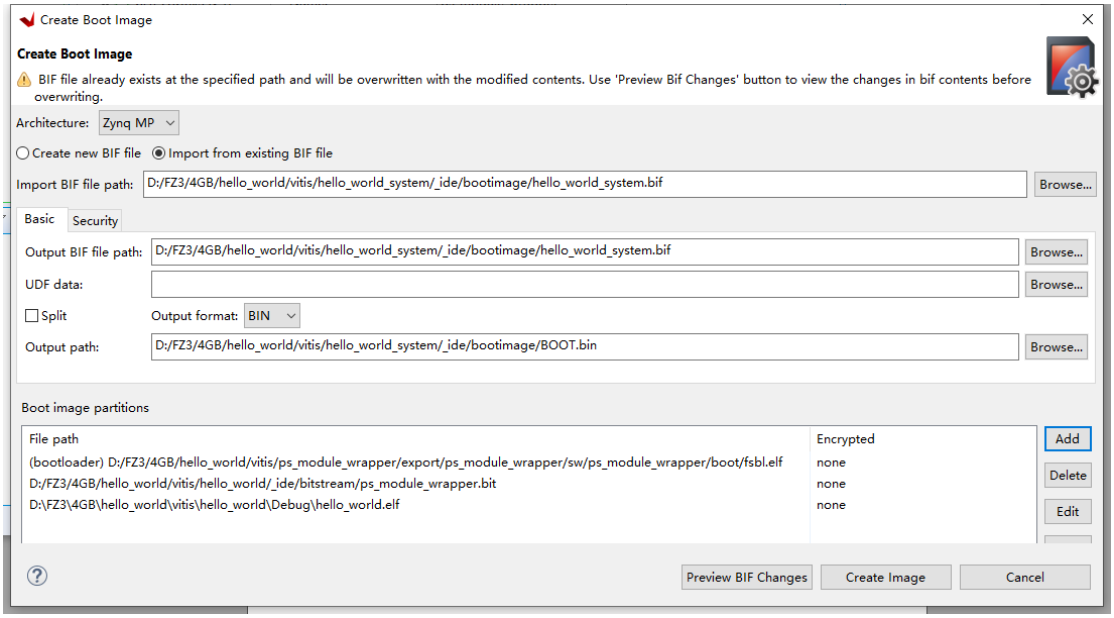
Click system of APP project and right click to select Build Project



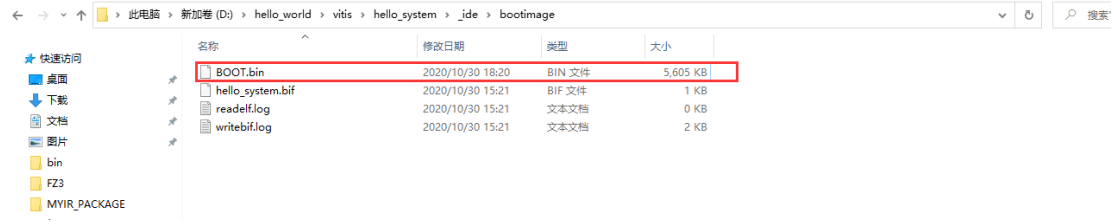
At this time, a debug folder will be created and the corresponding BOOT.BIN



Another method is to right-click the system of APP project and select Create boot image. In the pop-up window, you can see the path of the generated BIF file and BOOT.bin File. The bif file is the configuration file for generating the BOOT file.The BOOT.bin file is the boot file we need. It can be put into SD card to start or burn to QSPI flash.



在 Boot image partitions 列表中有要合成的文件，第一个文件一定是 bootloader 文件，就是上面生成的 fsbl.elf 文件，第二个文件是 FPGA 配置文件 bitstream，在本实验中由于没有 FPGA 的 bitstream，不需要添加，第三个是应用程序，在本实验中为 hello.elf。点击 Create Image 生成 BOOT.bin。There are files to be synthesized in the Boot image partitions list. The first file must be a bootloader file, which is generated above fsbl.elf file, the second file is the FPGA configuration file bitstream. The third is the application program. In this experiment, the hello.elf . Click create image to generate BOOT.bin .



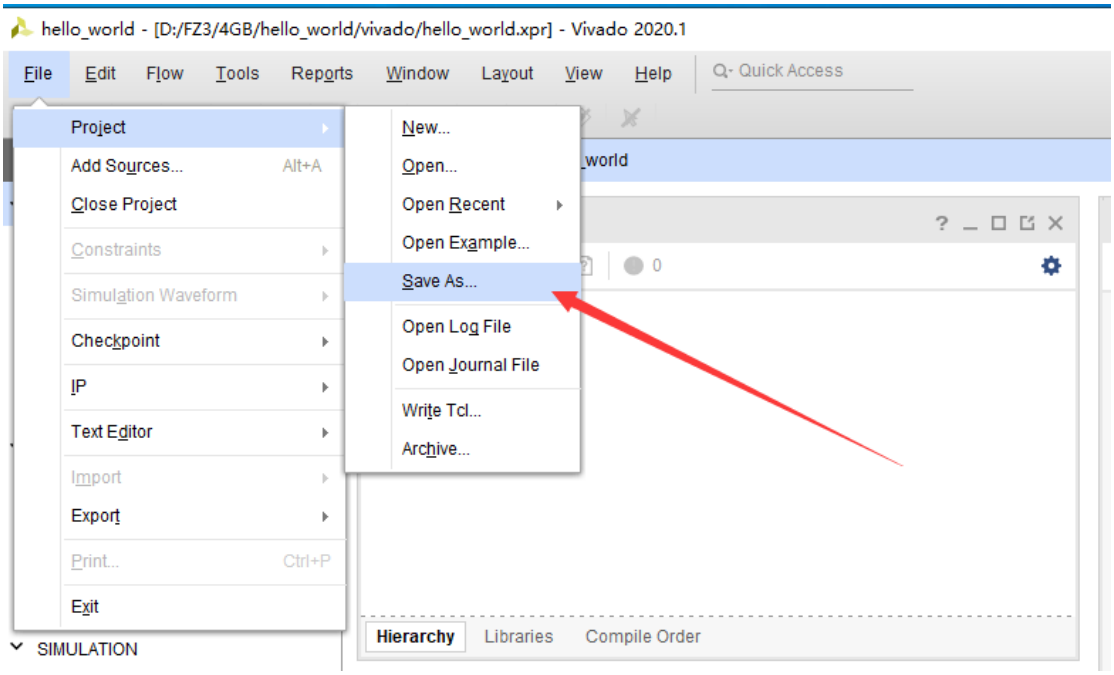
Set the development board to SD card startup mode, and then copy the BOOT.bin to SD card and run it on the development board.

Chapter 2 gpio_emio

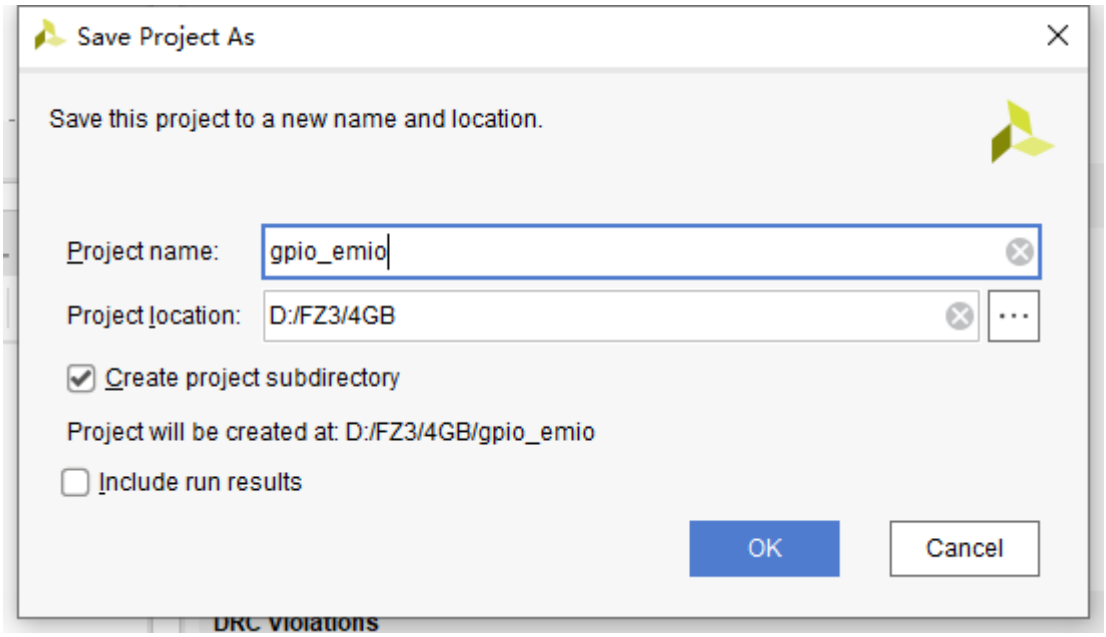
This chapter describes how to export GPIO through Emio and control D11 flicker in Vitis

1.1 Create project and configure IP core of PS

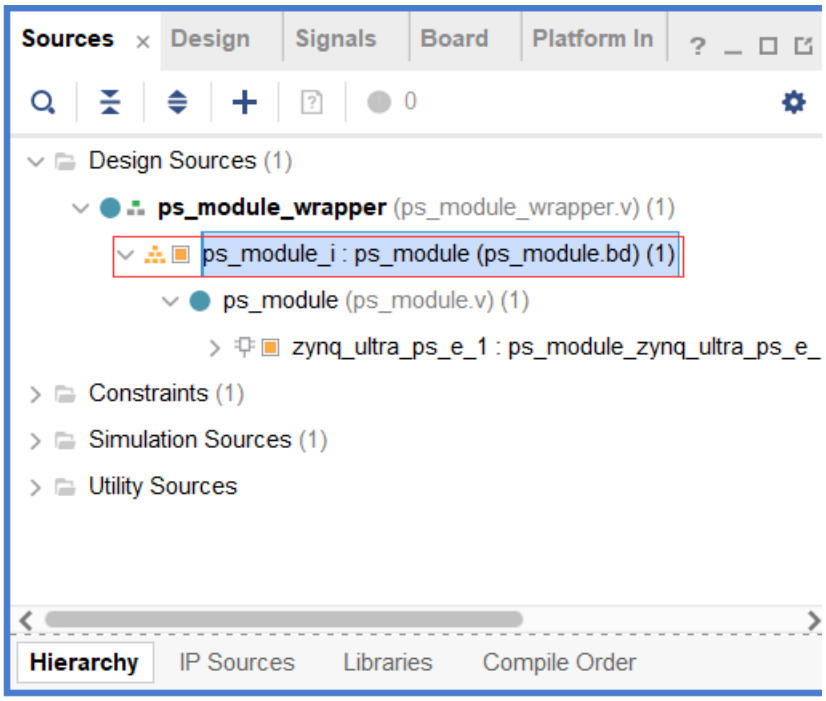
With Based on the hello_world project, save it as ps_emio project, open zynq configuration and check GPIO EMIO. Since there is one LED on the PL side, select the bit width of EMIO as 1 bit in MIO configuration. After configuration, click OK.



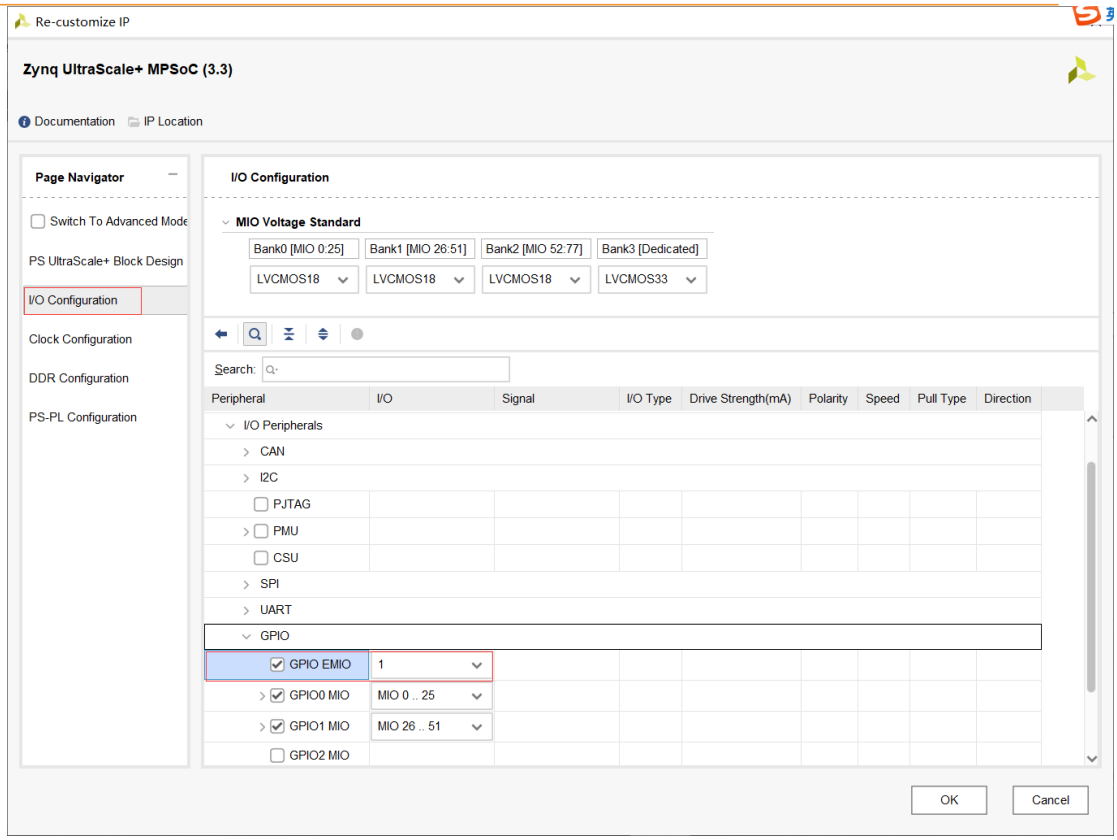
If " Create project subdirectory " is checked, subdirectories will be created under the directory, and checking "Include run results" will contain the compiled results



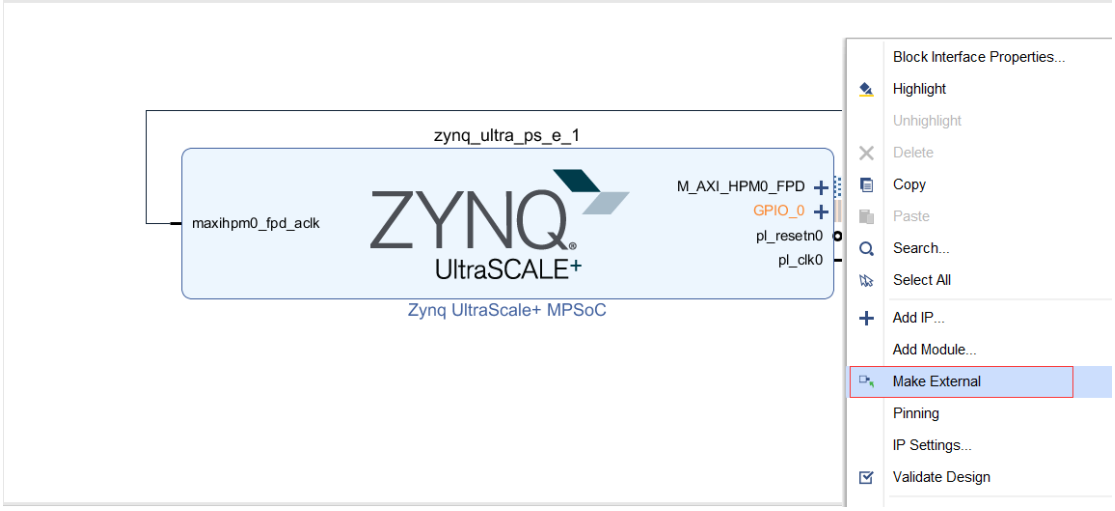
Double click ps_module.bd and open block design



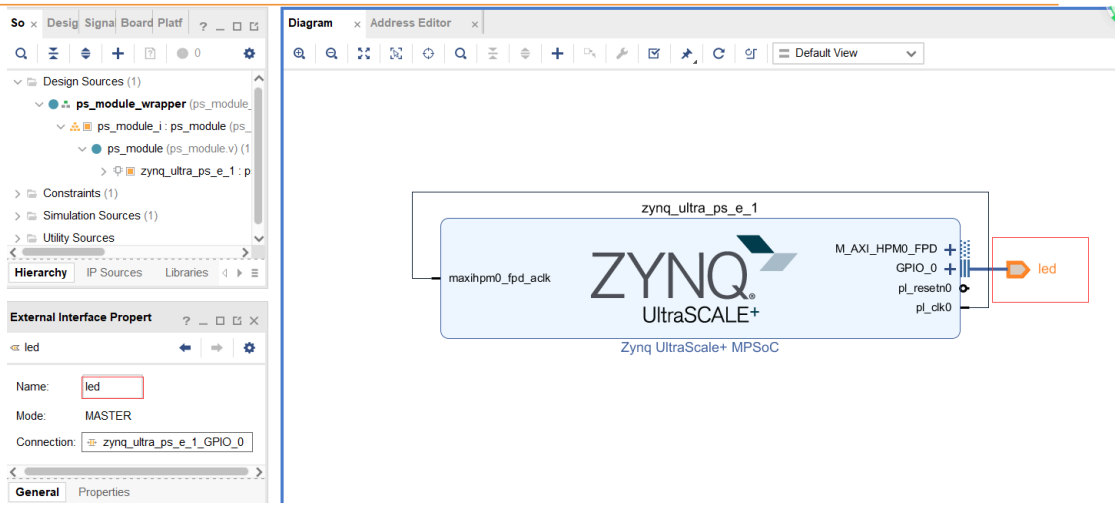
Open ZYNQ configuration and check GPIO EMIO. Due to one LED on the PL side, select the bit width of EMIO as 1 bit in MIO configuration. After configuration, click OK.



Click GPIO_ 0 port, Right click and select make external to export the port signal.

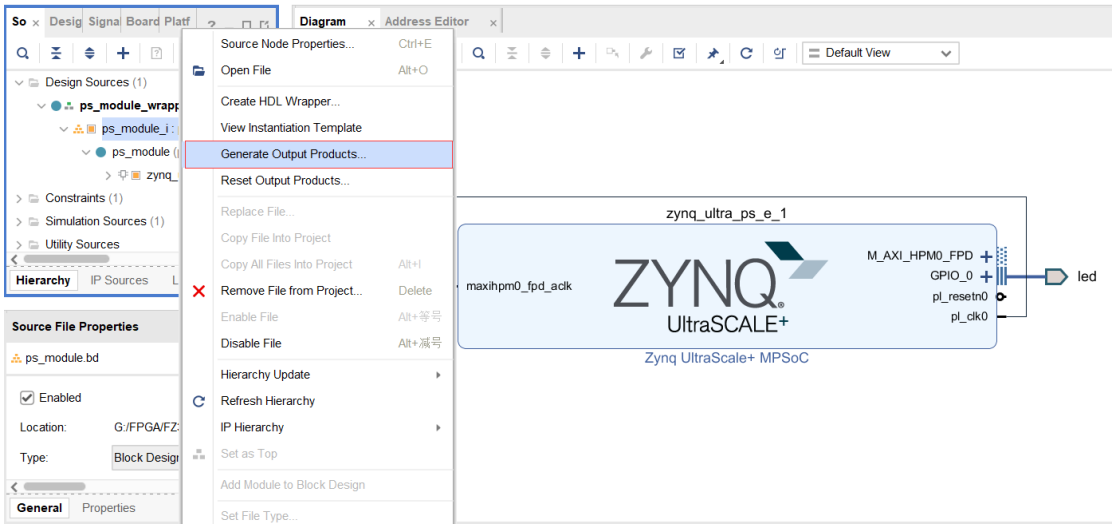


Click on the pin and change the pin name to LED, and save the design.

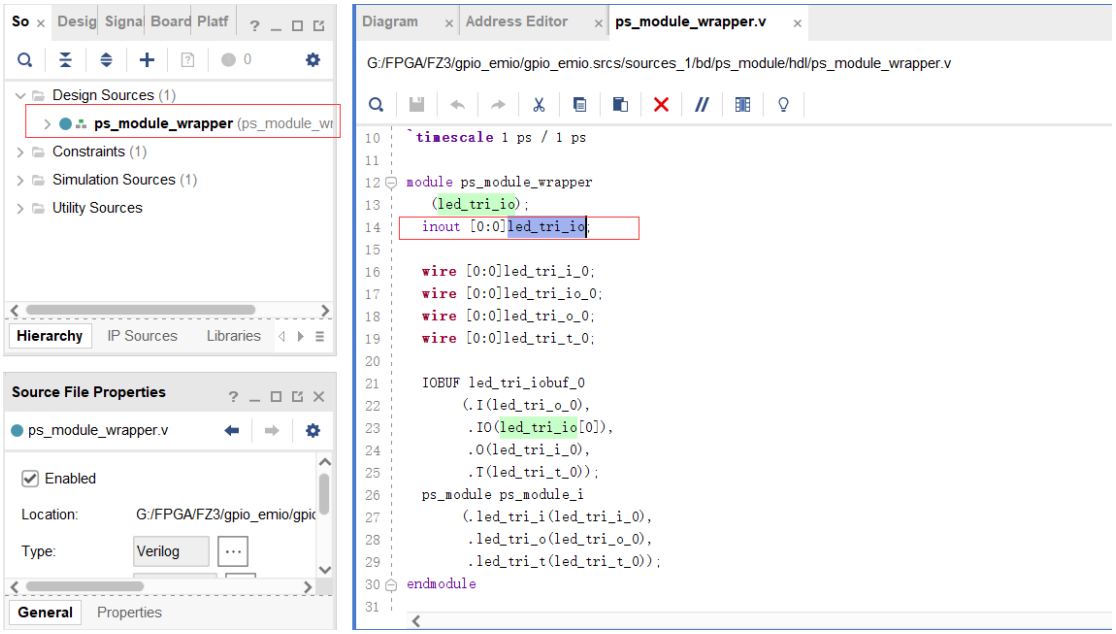


1.2 Generate synthesis files

Click ps_module.bd ,then right click and select Generate Output Products to regenerate the output file.

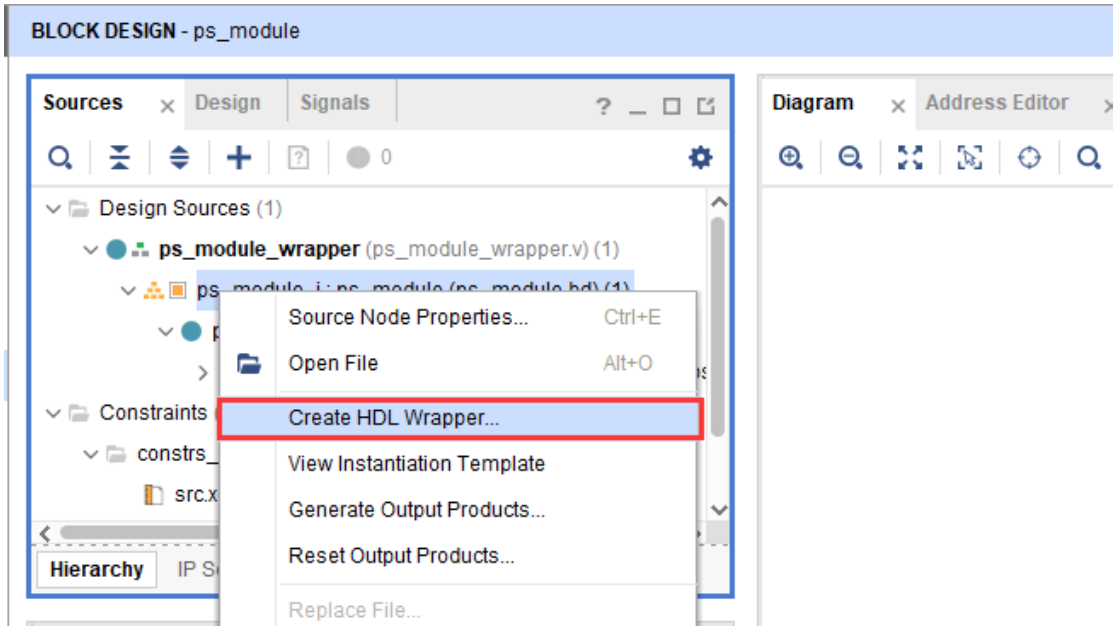


After the end, the top-level file will update the new pin, which needs to be pin bound.



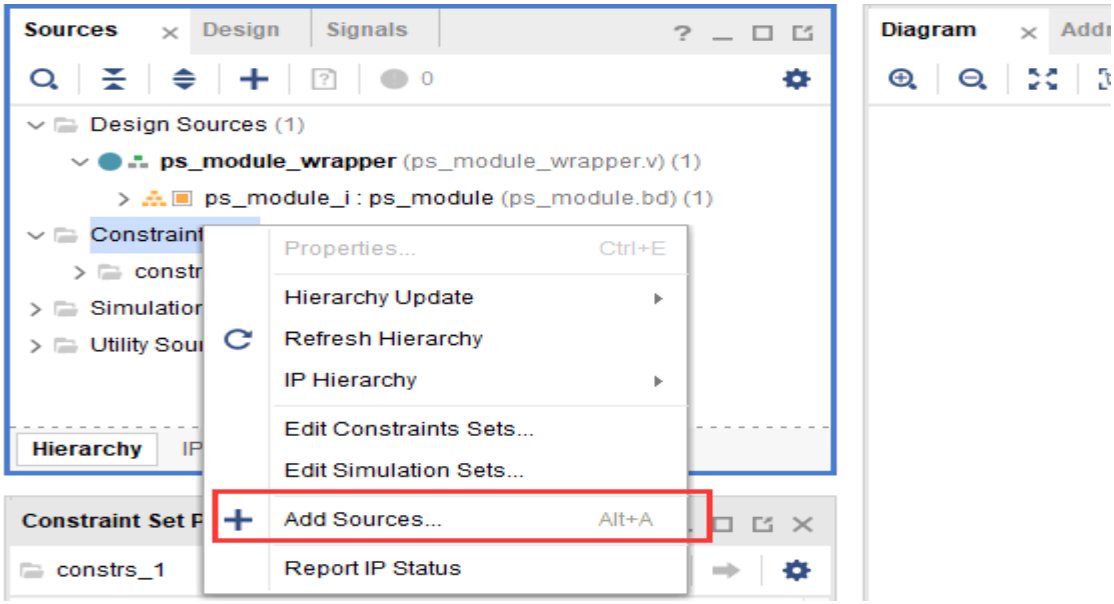
1.3 Generating top-level file of FPGA

Select ps_ module.bd. Right click -- Create HDL Wrapper to generate FPGA top-level file.

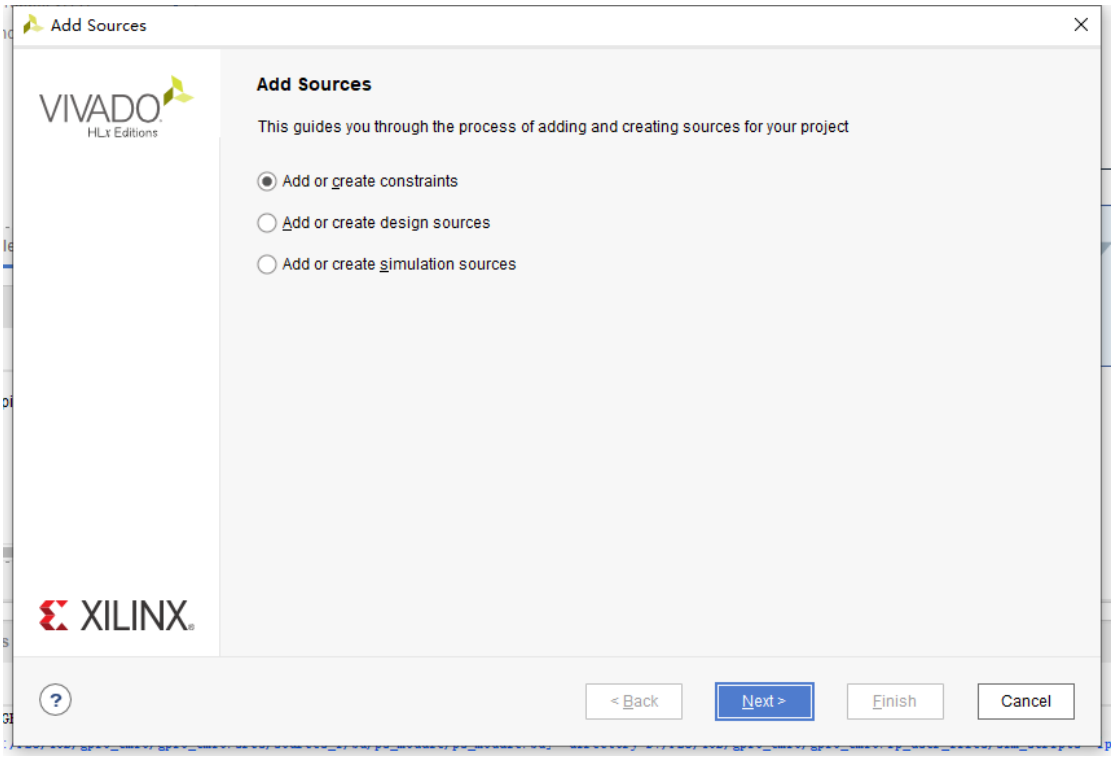


1.4 Adding xdc pin constraints

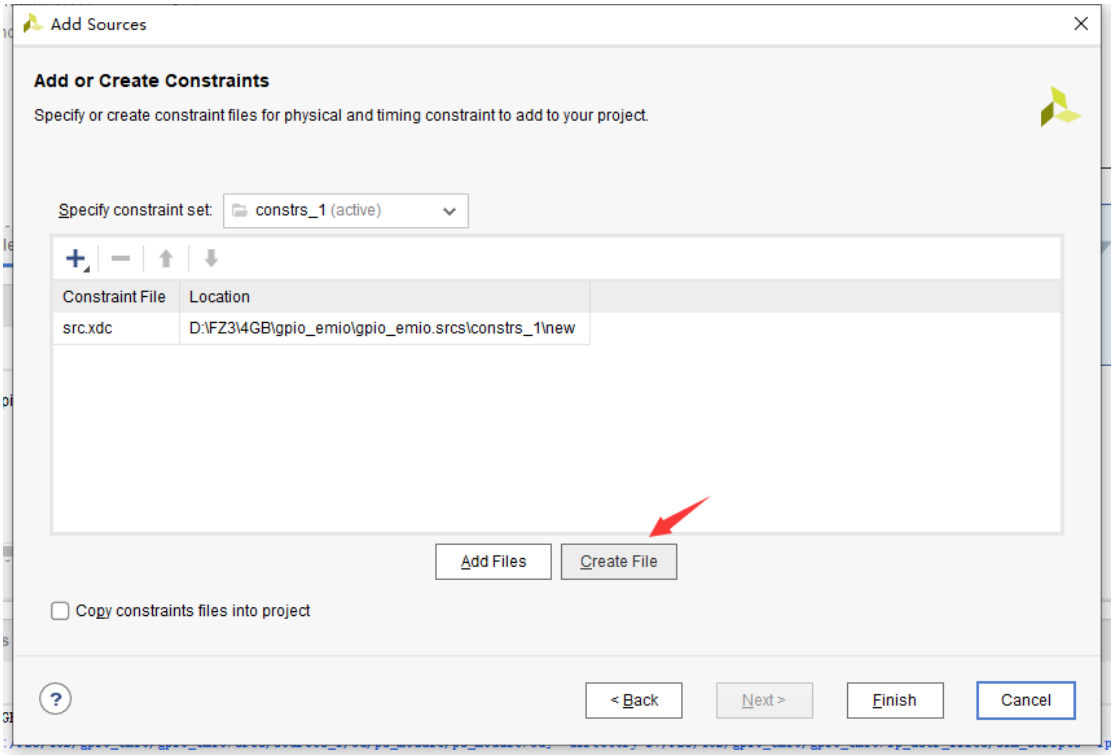
Left click to select Constraint， right click--Add Source.



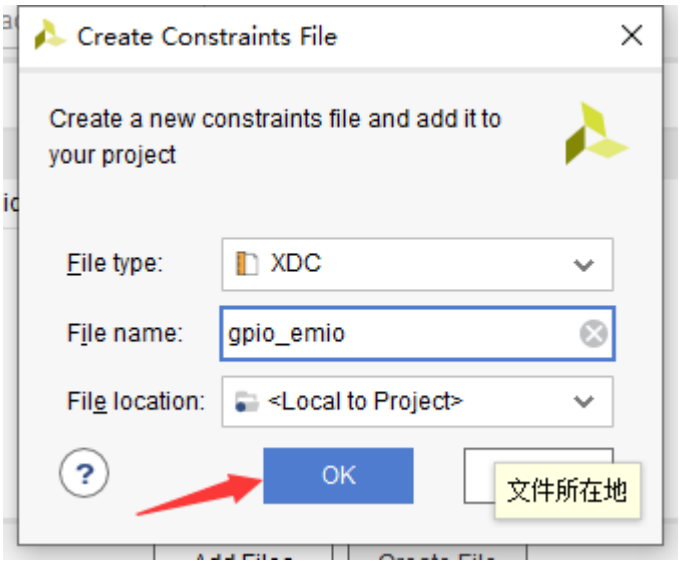
Select Add Create Constraints



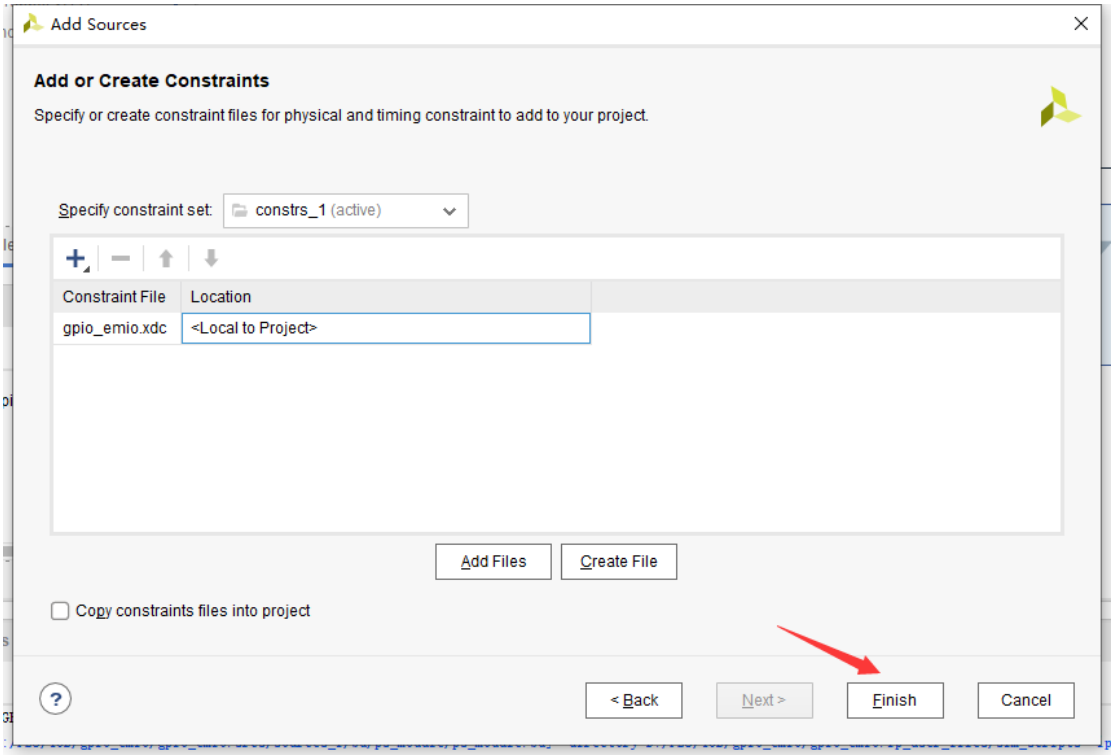
Select Create File



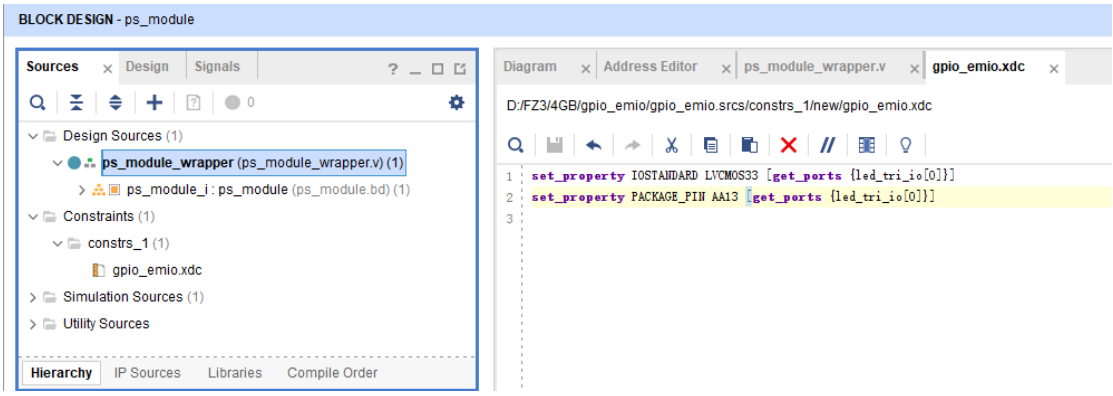
Create a src.xdc file



Click Finish

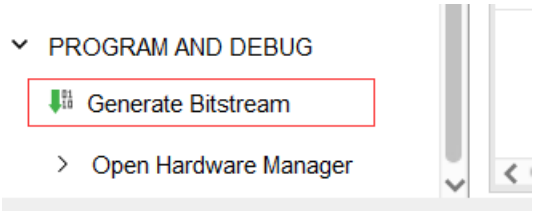


Add the following content to src.xdc, the port name must be consistent with the top-level file port.



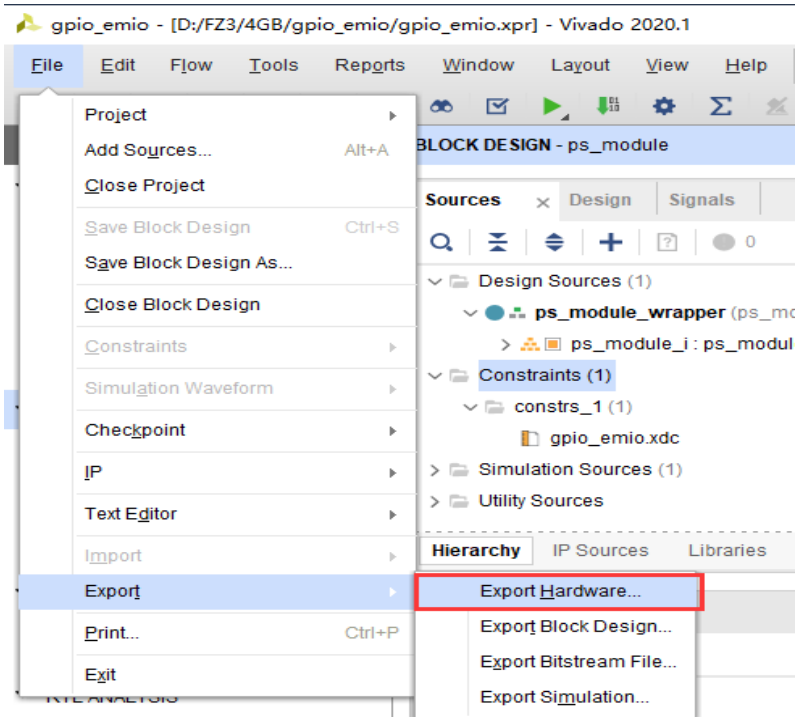
1.5 Generate bit files

Generate bit file

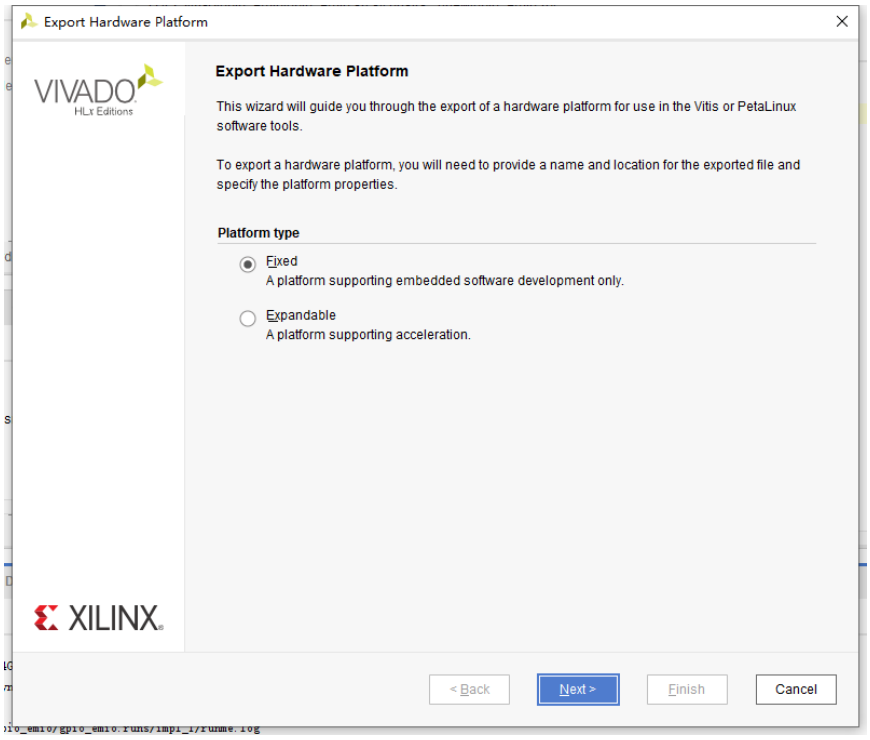


1.6 Export Hardware Profile

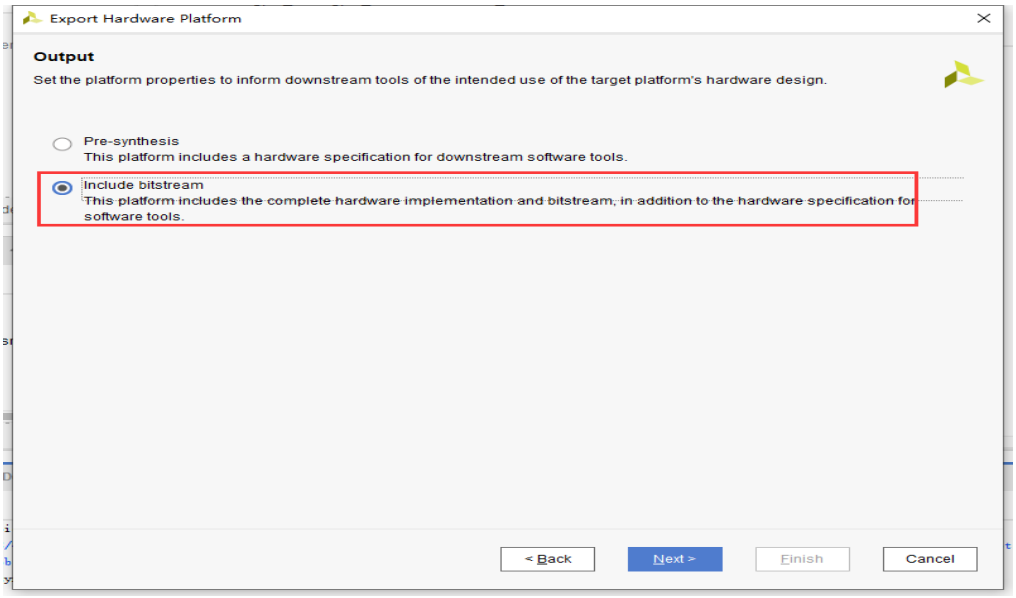
Click File > export > export hardware > OK on the menu bar to export the hardware configuration file.



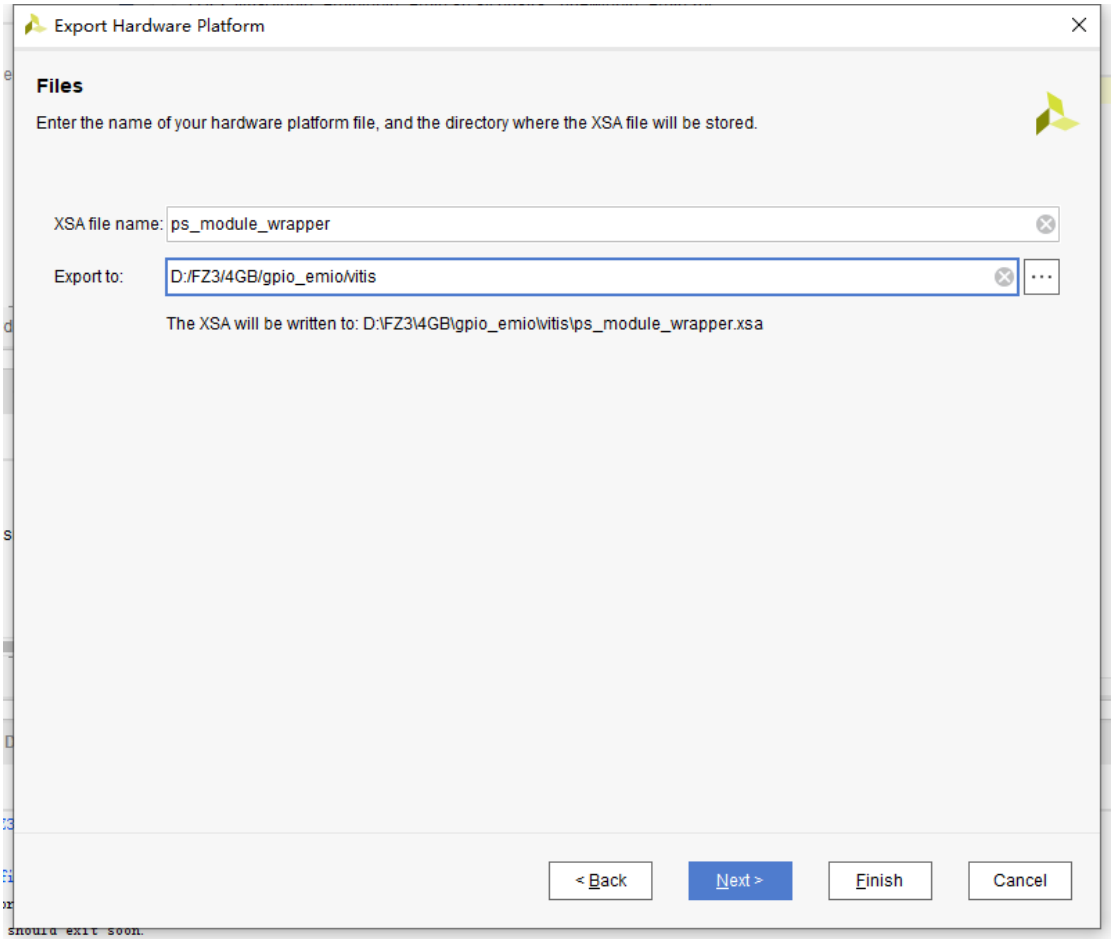
The platform type select Fixed



To export hardware, select "Include bitstream".

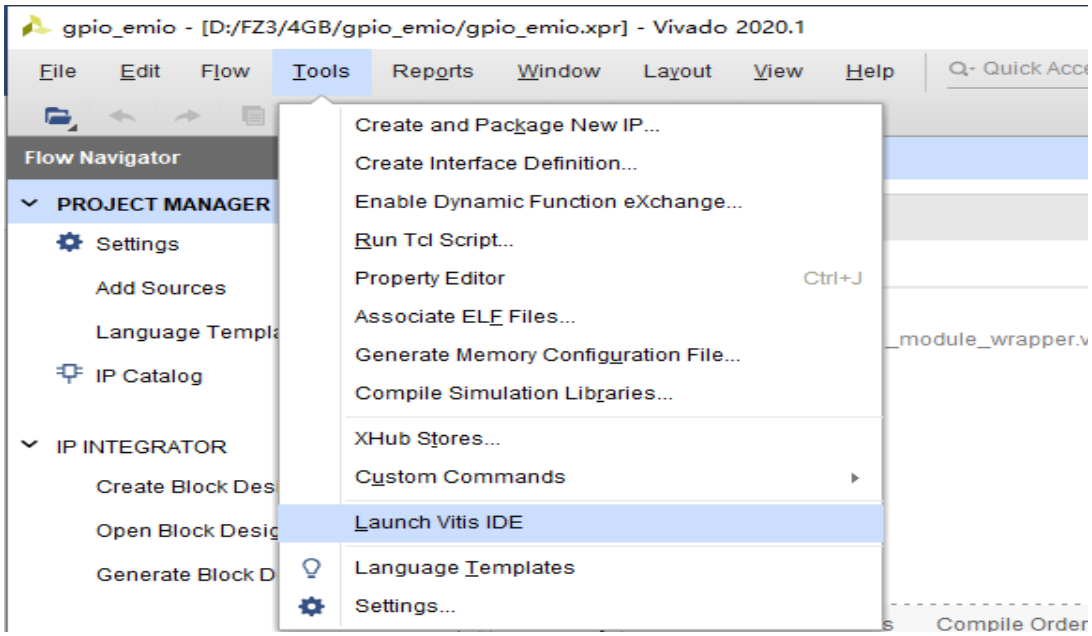


Select the export file name and export path. Here, the file name is selected by default, and the path is the newly created Vitis folder under the project file.

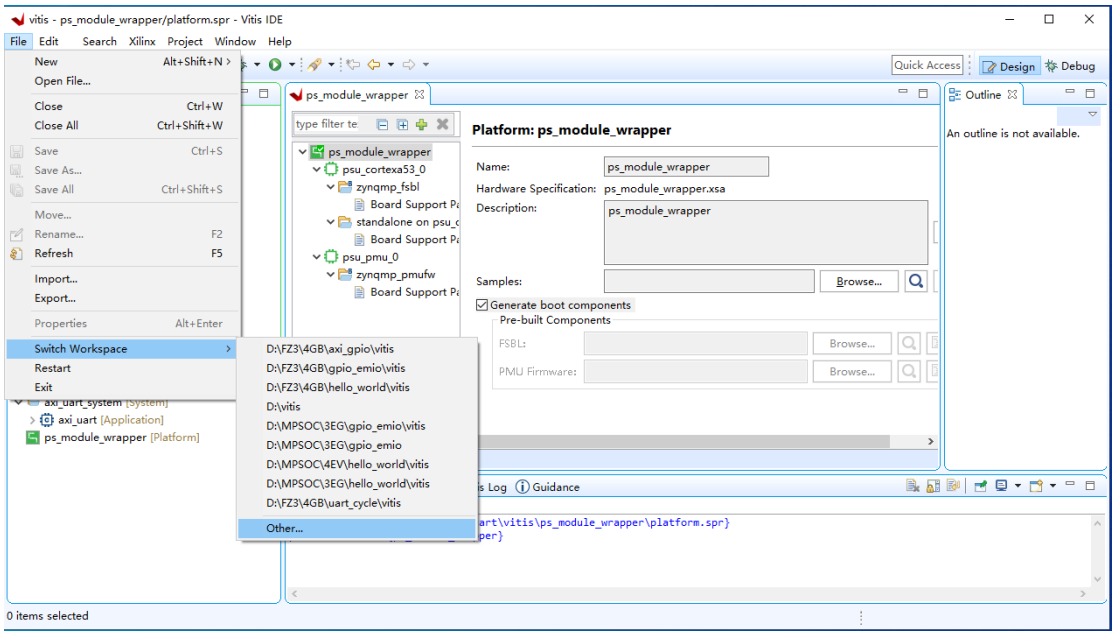


1.7 Launch Vitis and create new platform project

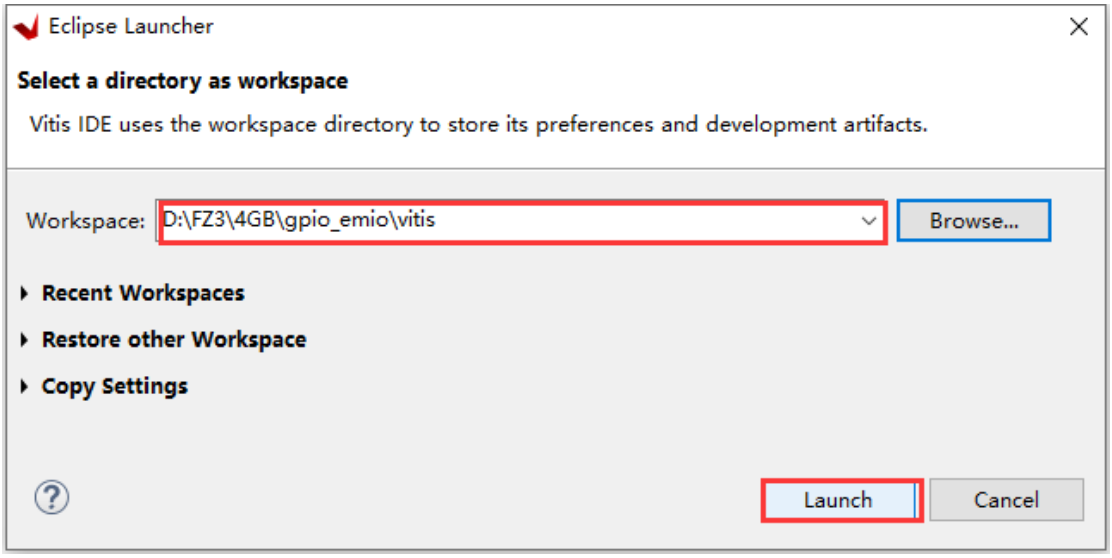
Click Tools > launch Vitis ide on the menu bar to launch Vitis



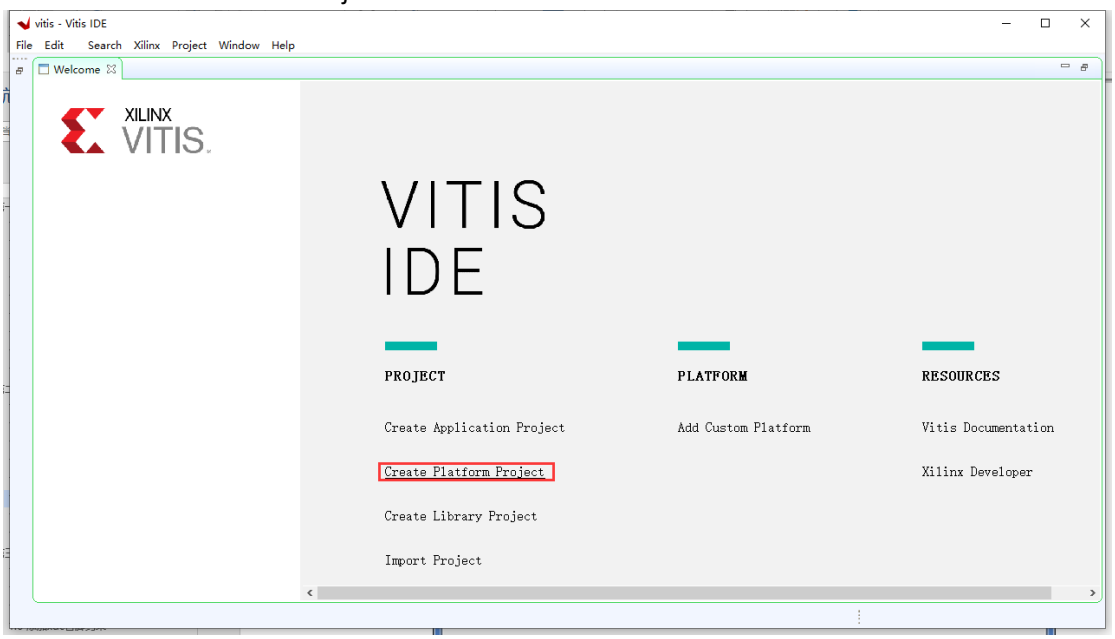
Click File→Switch Workspace→Other…,



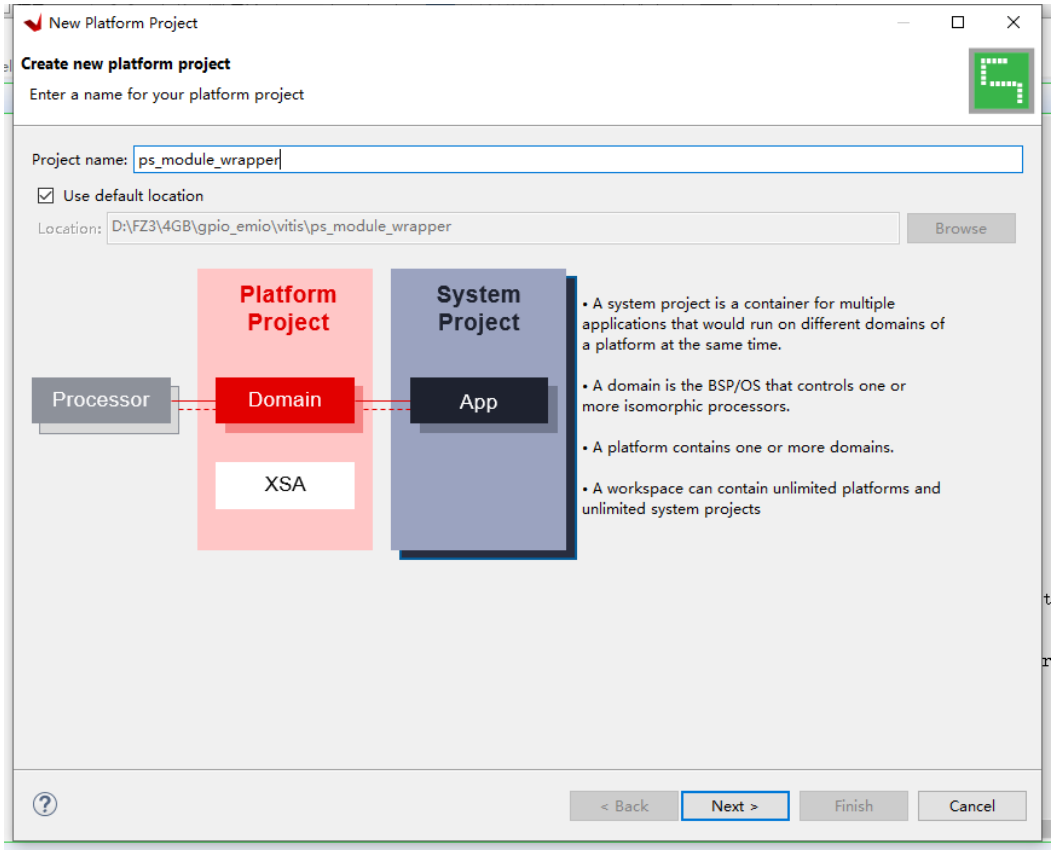
elect the path, select the vitis folder under gpio_emio project. Click launch



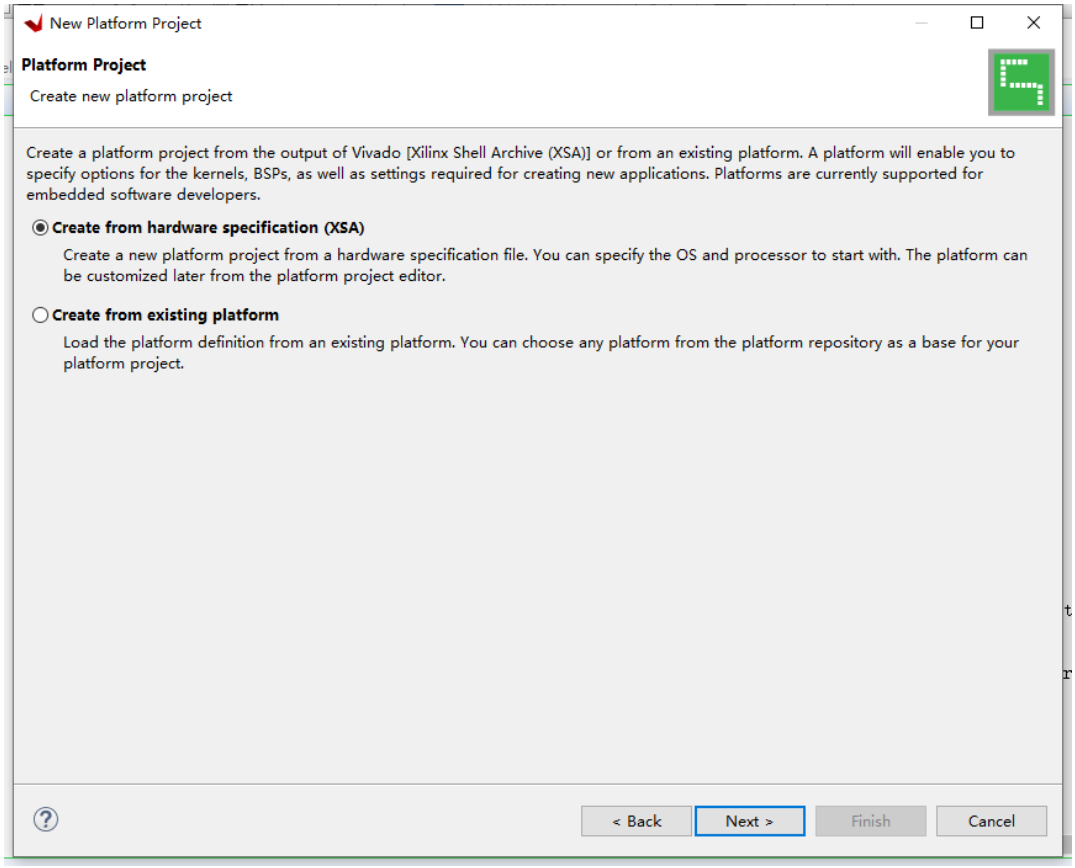
Click Create Platform Project



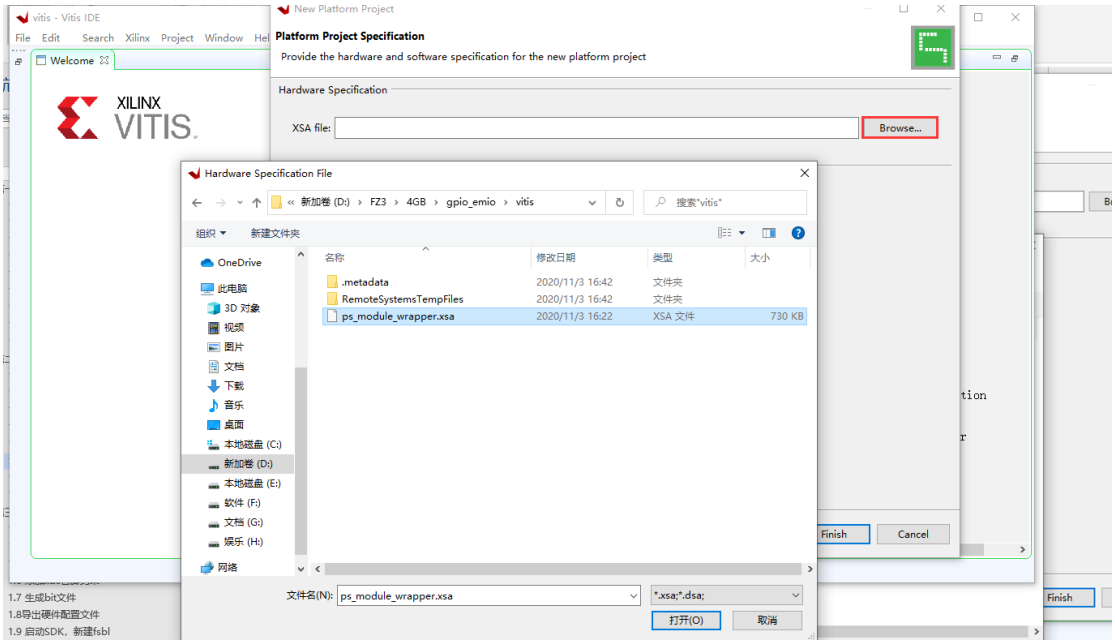
The project name uses the same name as the xsa file. Click next.



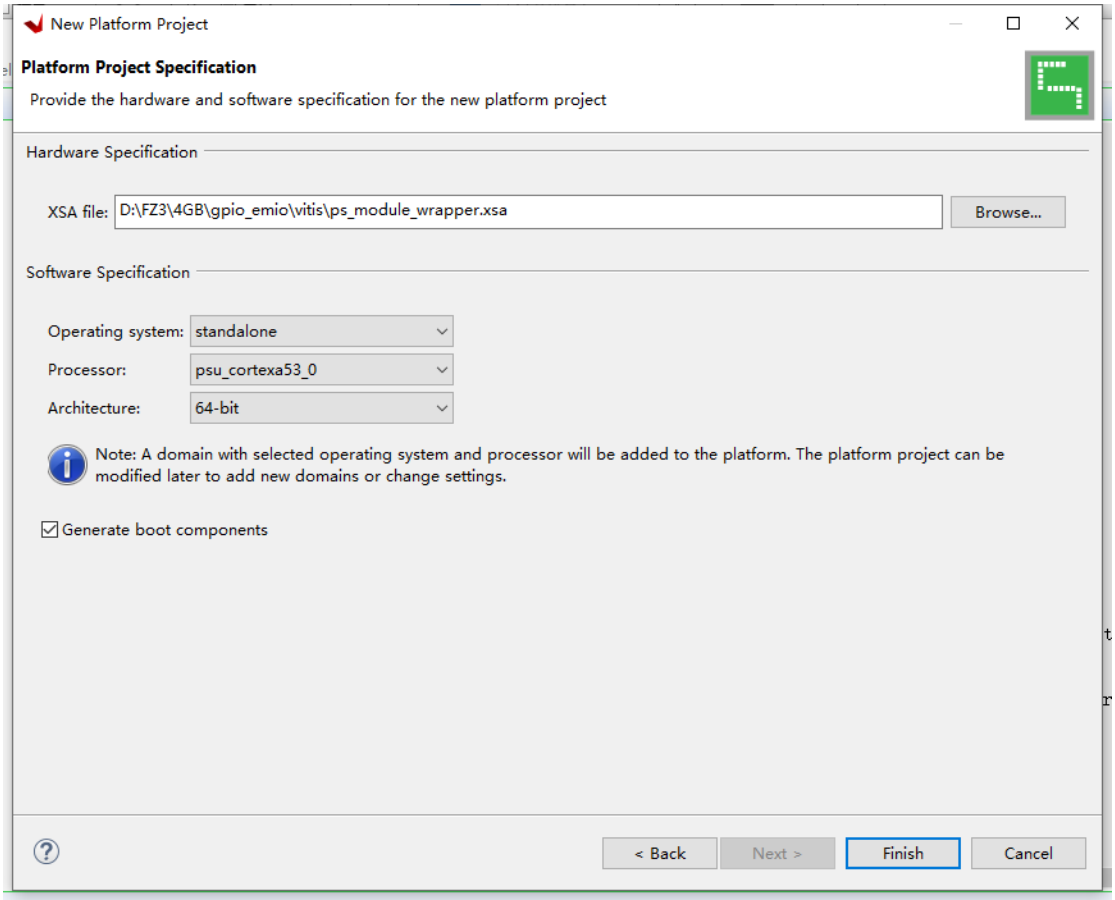
Select Create from hardware specification(XSA),click NEXT



Open Browse… , select the previously generated xsa file

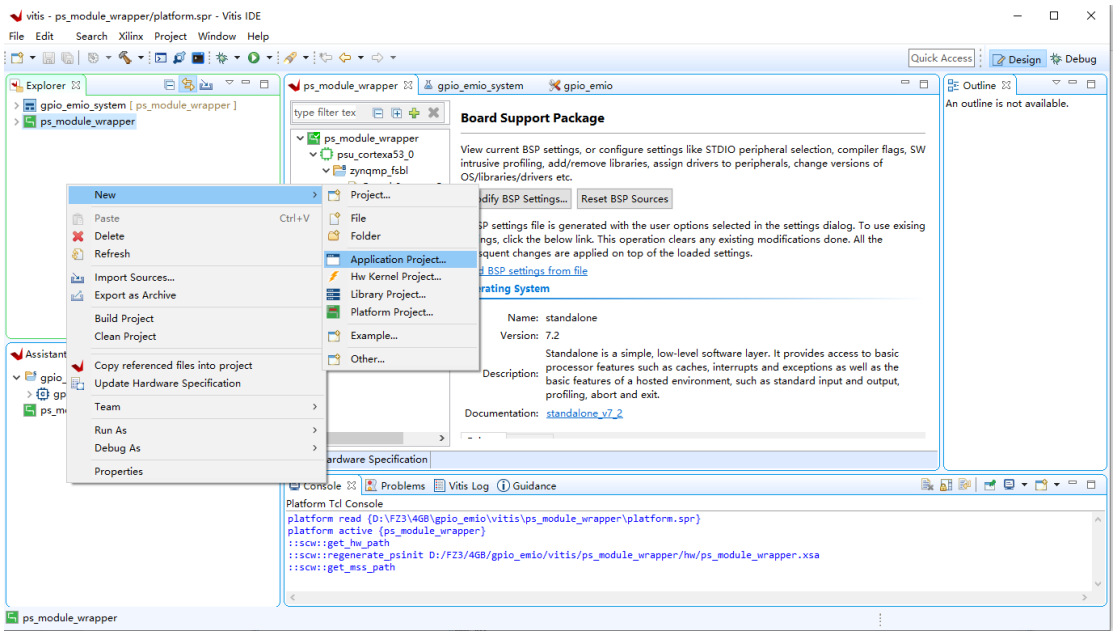


Leave the default and click finish.

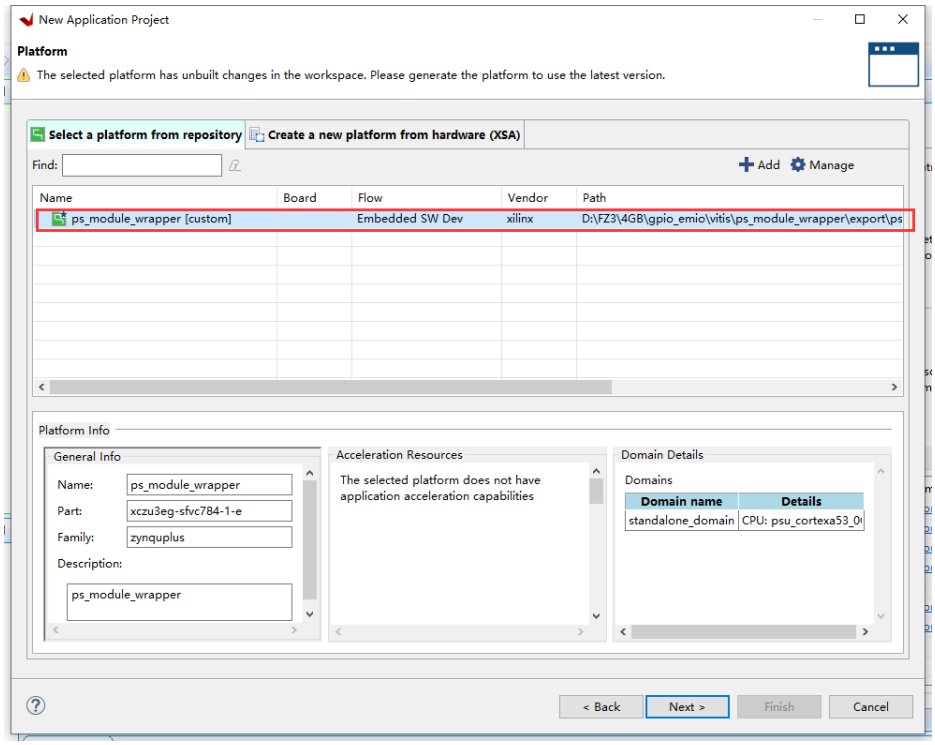


1.8 New gpio_emio Project

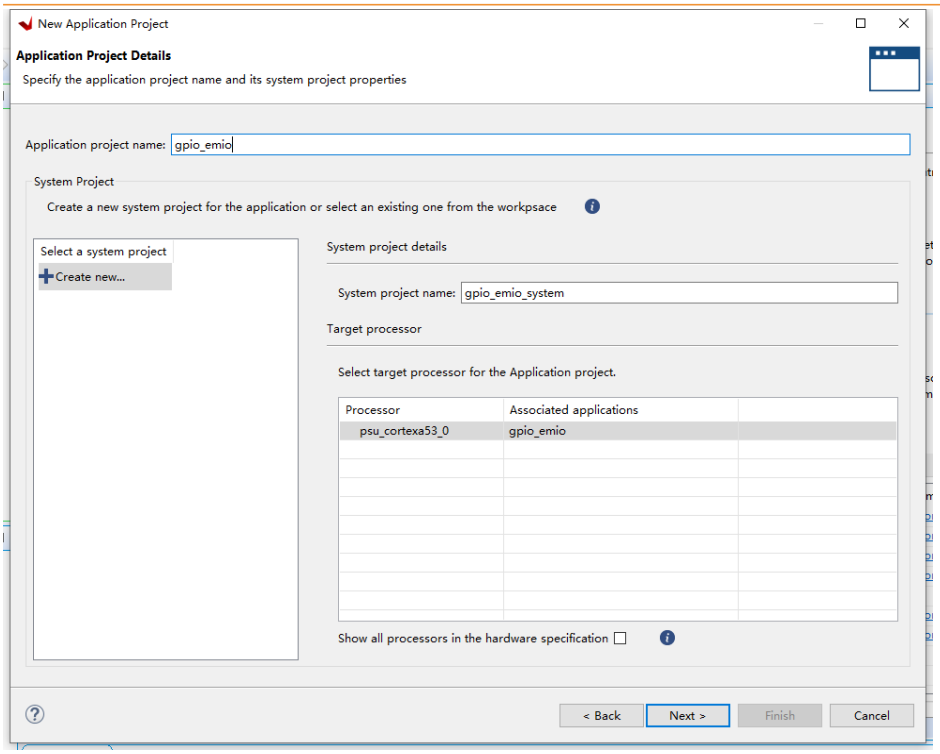
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



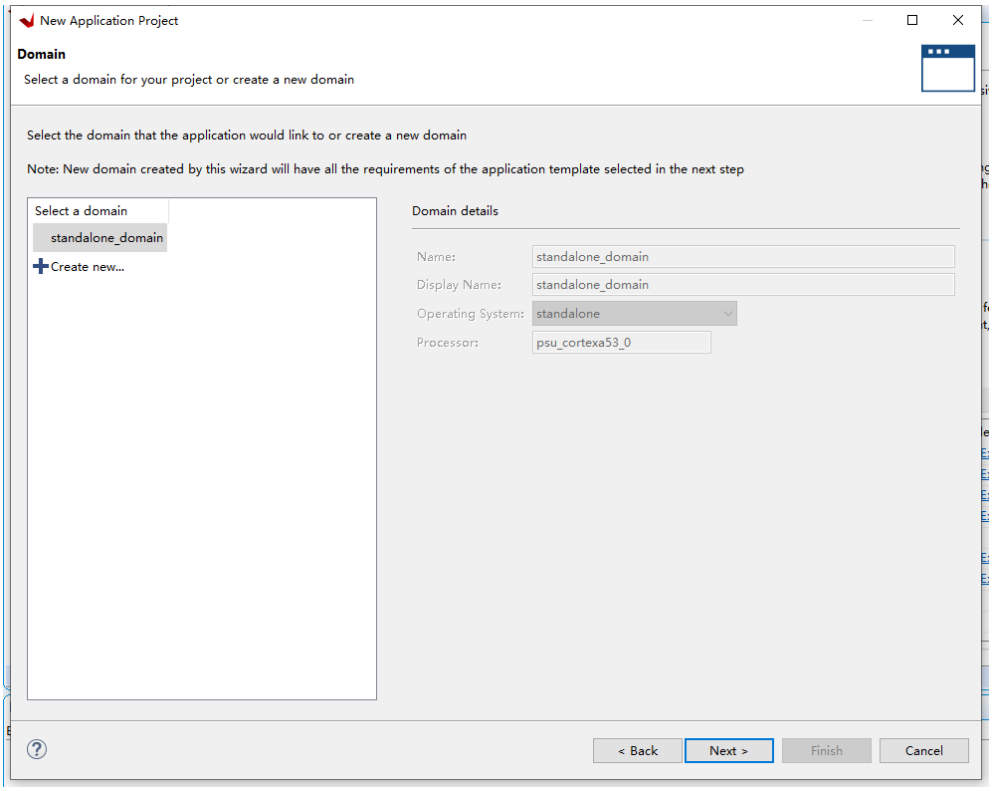
Skip to the second page and select the platform project created above (default), next



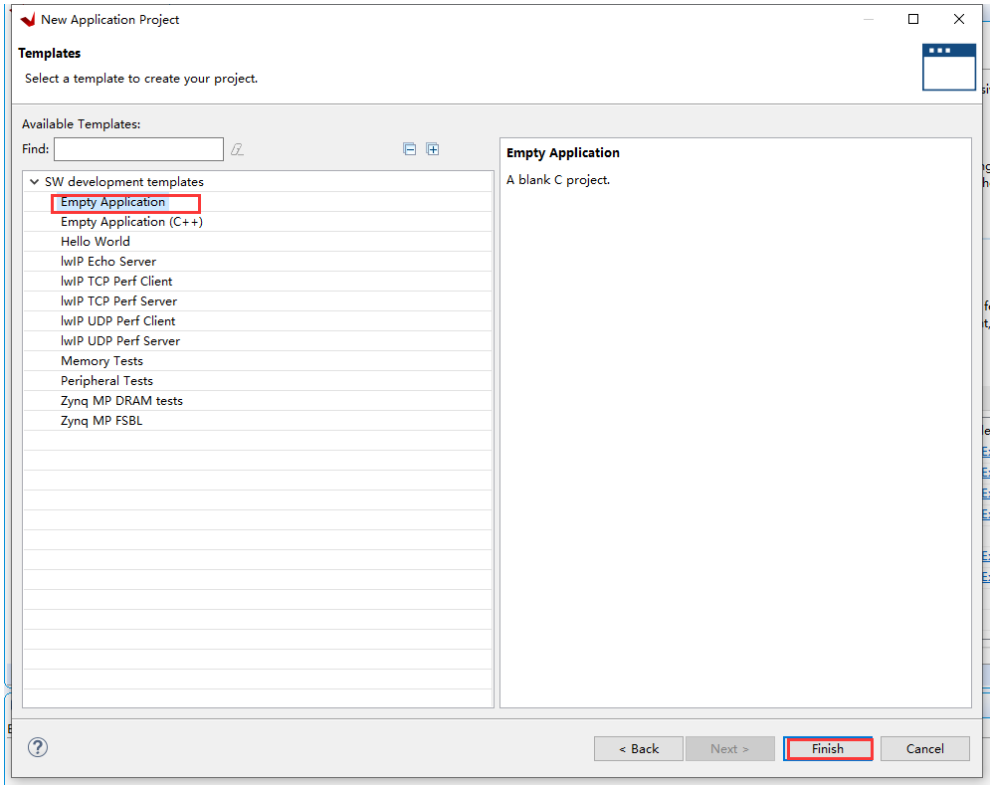
Project name enter gpio_ emio， Next



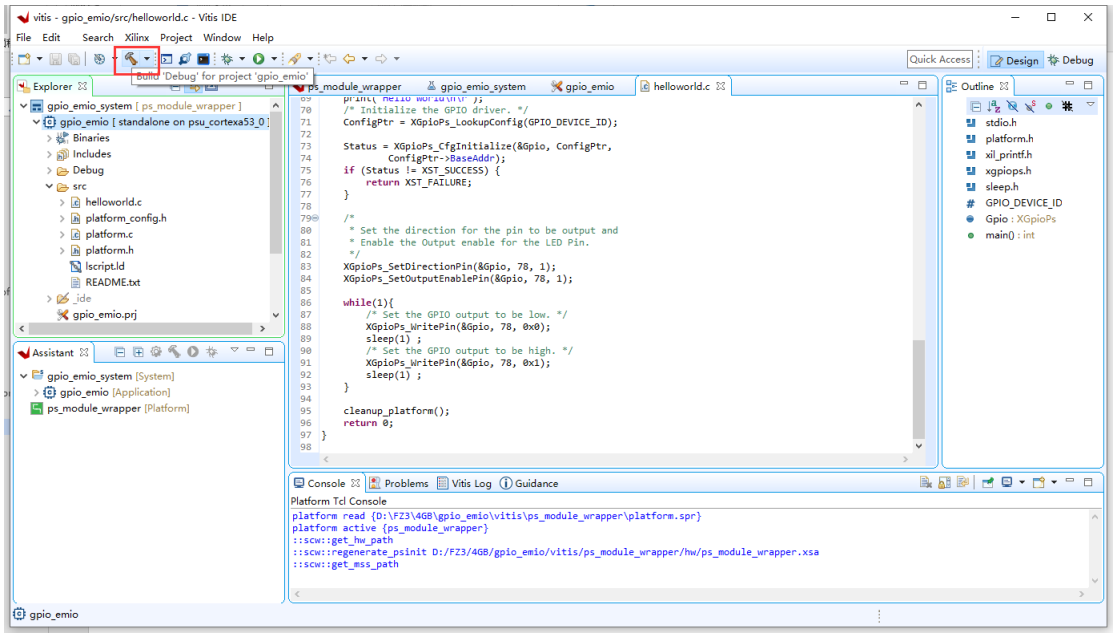
Next



Select Empty Application， Finish

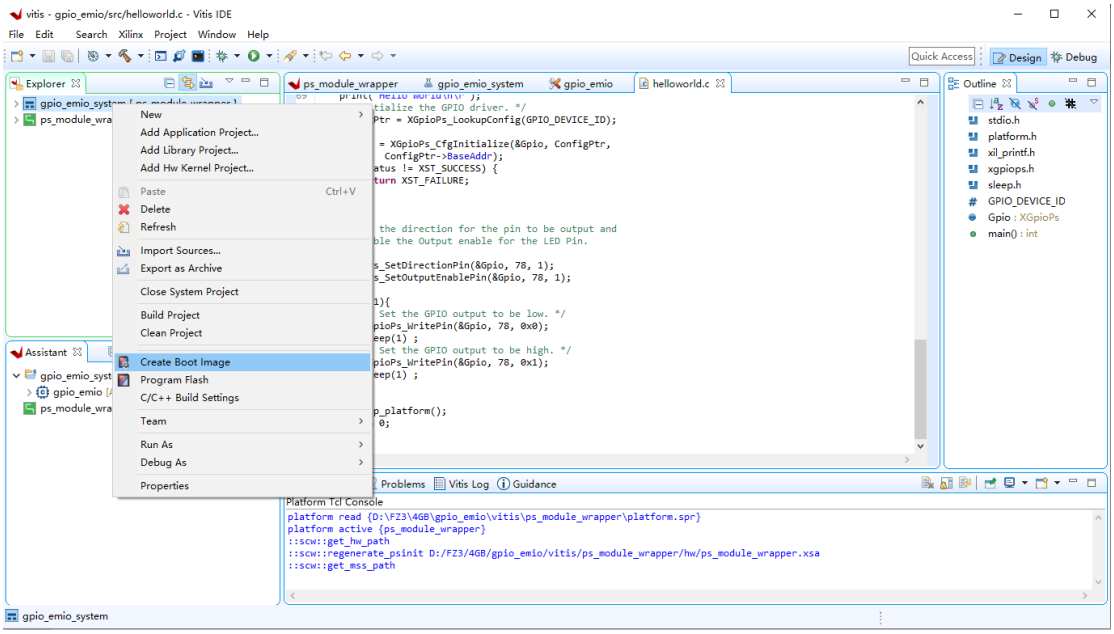


Copy the files in the example project to src, select the project, and click the compile buttom.



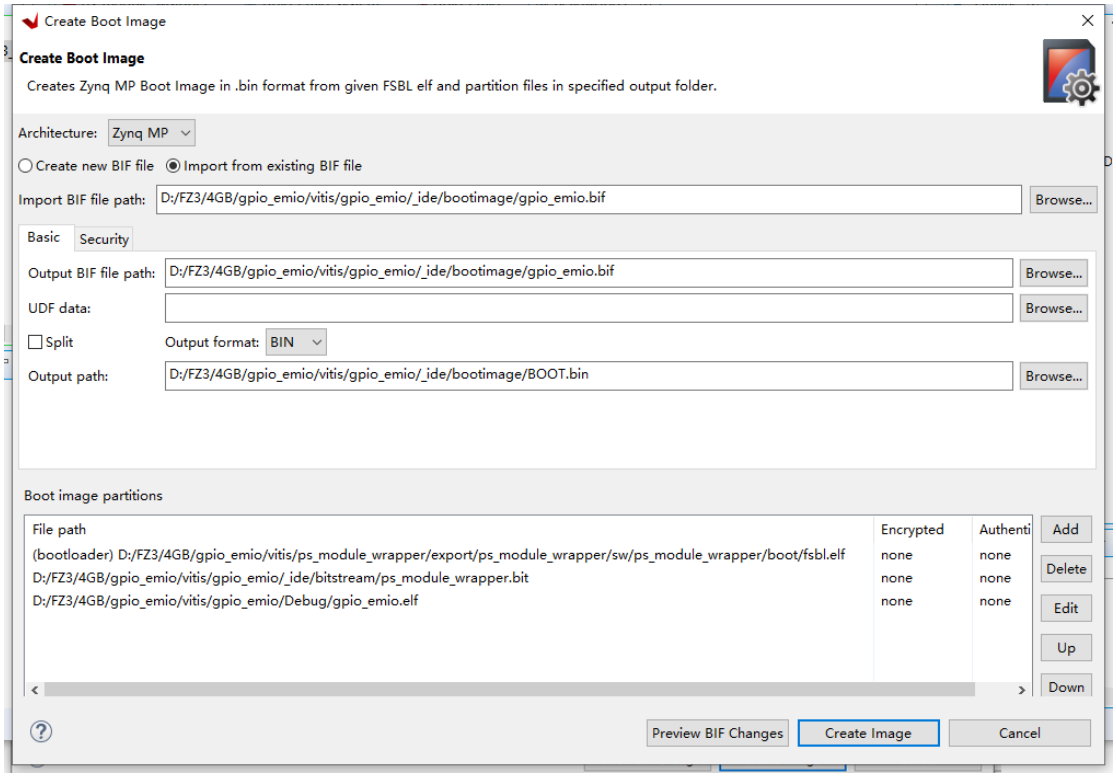
1.9 Generate BOOT.bin file

Right-click the system of APP project and select Create boot image.

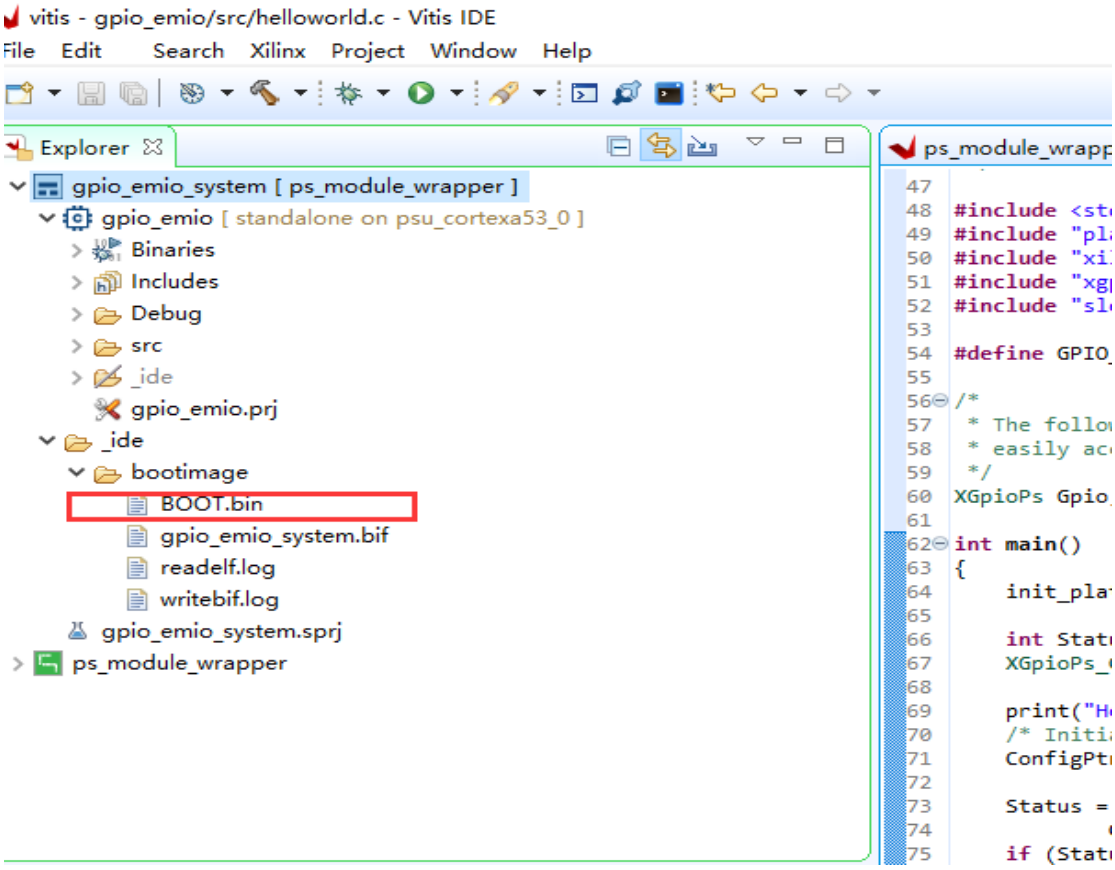


In the pop-up window, you can see the path of the generated BIF file and BOOT.bin file. The bif file is the configuration file for generating the BOOT file . The BOOT.bin file is the boot file we need. It can be put into SD card to start or burn to QSPI flash.

Click Create Image to generate BOOT.bin Startup file



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.

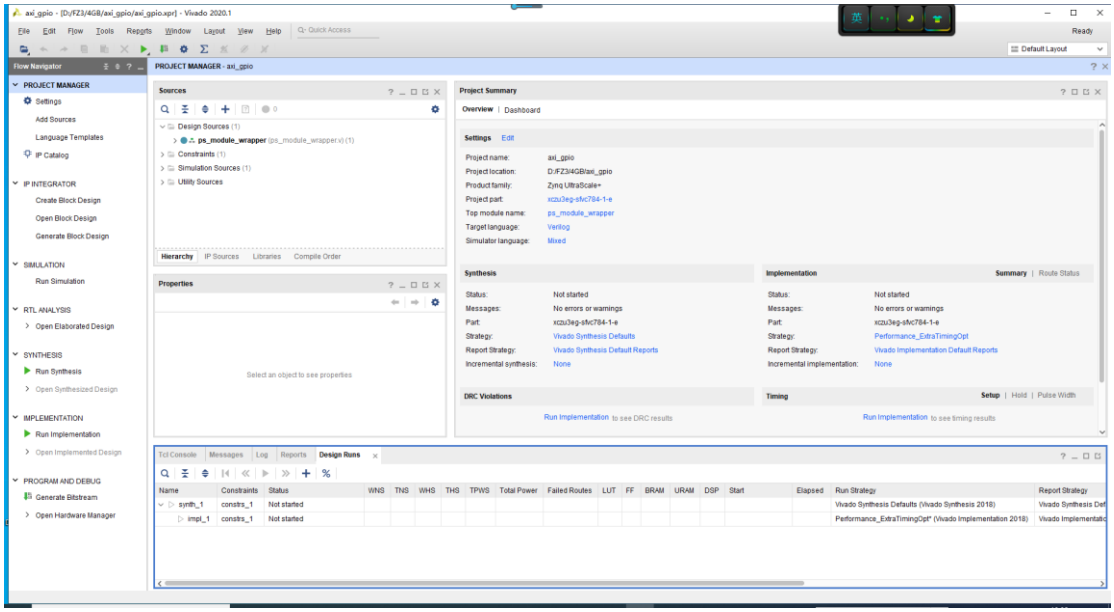

vitis - gpio_emio/src/helloworld.c - Vitis IDE
File Edit Search Xilinx Project Window Help
Explorer
gpio_emio_system [ps_module_wrapper]
gpio_emio [standalone on psu_cortexa53_0]
Binaries
Includes
Debug
src
_ide
gpio_emio.prj
_ide
bootimage
BOOT.bin
gpio_emio_system.bif
readelf.log
writebif.log
gpio_emio_system.sprj
ps_module_wrapper
ps_module_wrapp
47
48 #include <st
49 #include "pl
50 #include "xi
51 #include "xg
52 #include "sl
53
54 #define GPIO_
55
56 /*
57 * The follo
58 * easily ac
59 */
60 XGpioPs Gpio_
61
62 int main()
63 {
64 init_pla
65
66 int Stat
67 XGpioPs_
68
69 print("H
70 /* Initi
71 ConfigPt
72
73 Status =
74
75 if (Stat

Chapter 3 axi_gpio

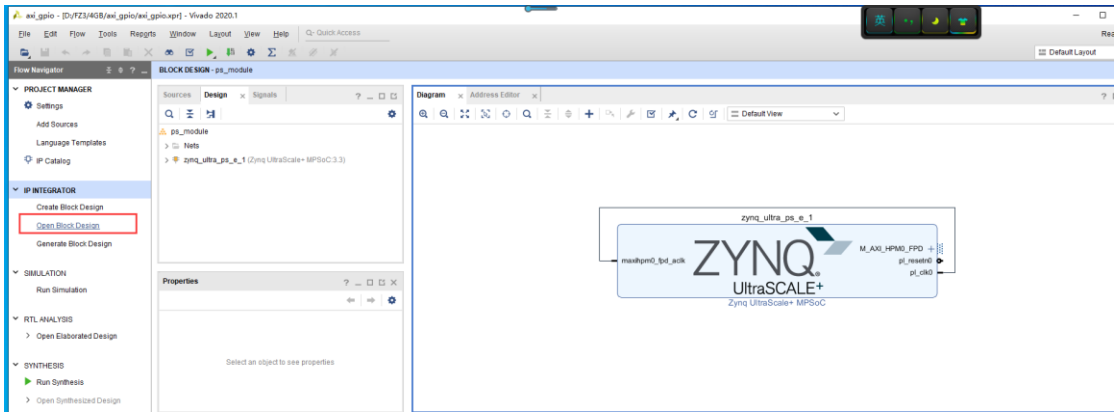
Vivado comes with many IP cores, such as CAN, UART, SPI, etc. if the CPU does not have enough resources, it can be extended on the PL side. Information about these IP cores can be found in DocNav. Docnav will be automatically installed when installing vivado software. DocNav also contains many other materials, such as the use of Vivado and SDK, and video tutorials. This chapter will introduce how to use Xilinx's GPIO core to implement a GPIO controller in PL to control LED.

1.1 Create project and configure IP core of PS

Based on "ps_hello" project, save as axi_gpio project.

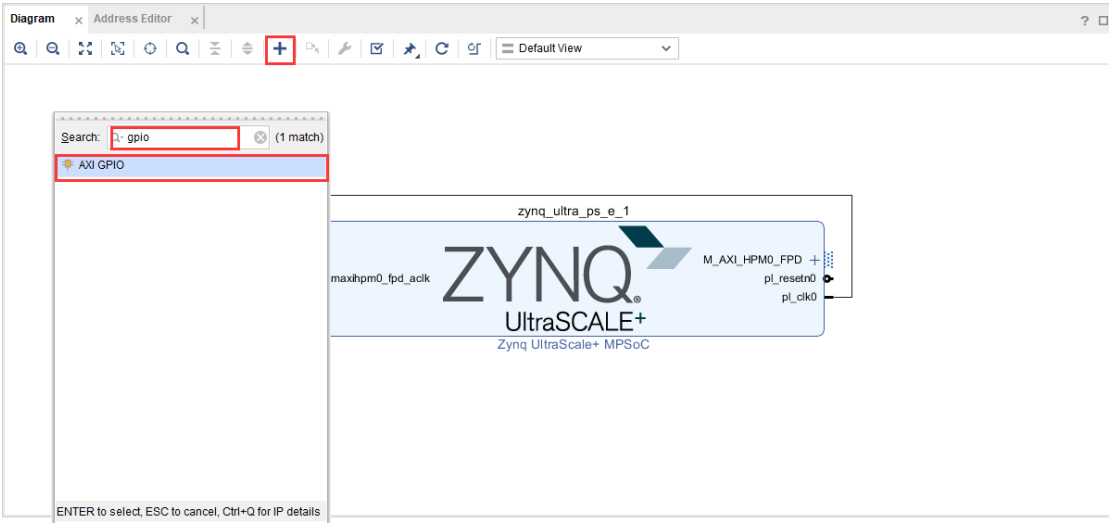


Click Open Block Design on the left navigation bar to open block design

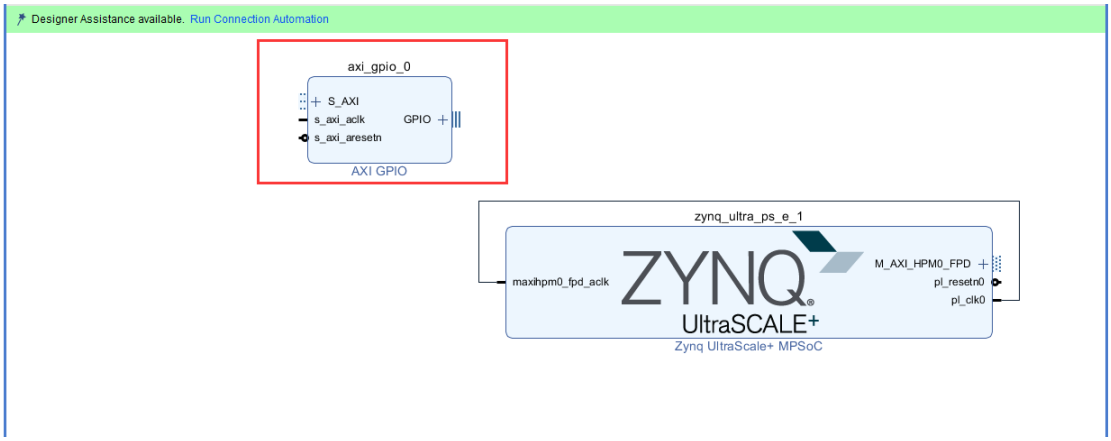


1.2 Add axi_gpio IP core and configure it

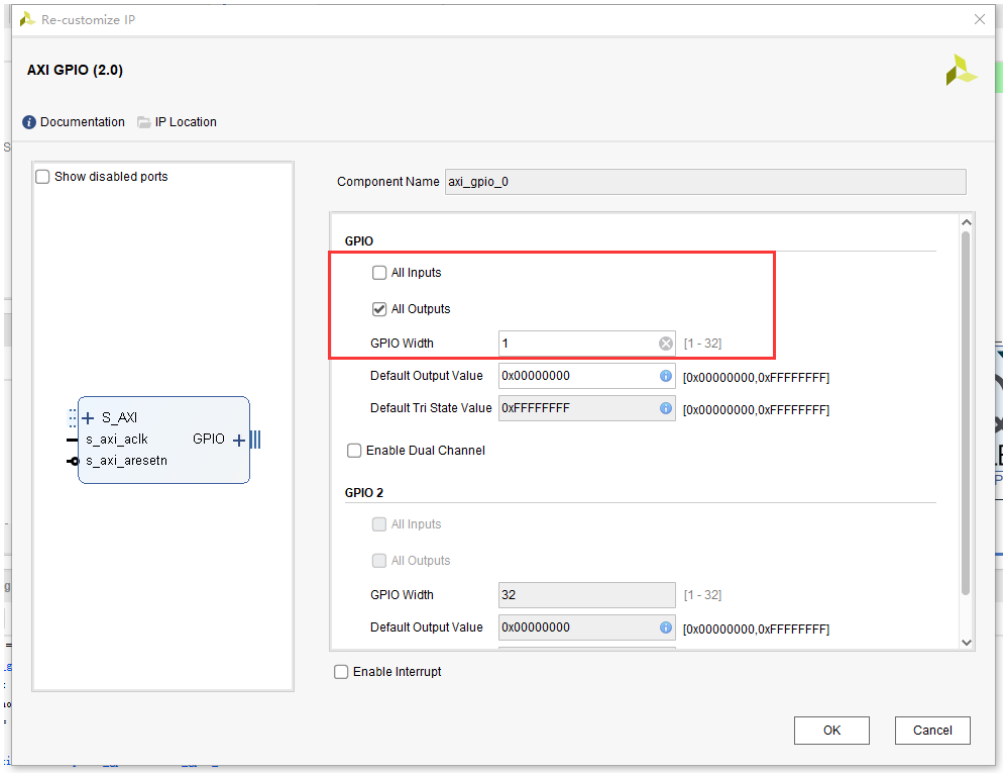
Click Add IP .Enter axi_gpio, then double-click AXI GPIO to add axi_gpio core.



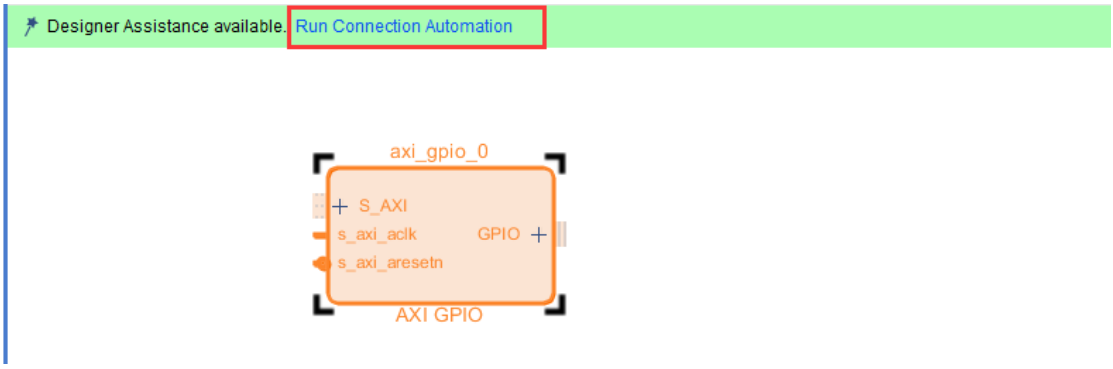
Double-click the axi_gpio core to set parameters.



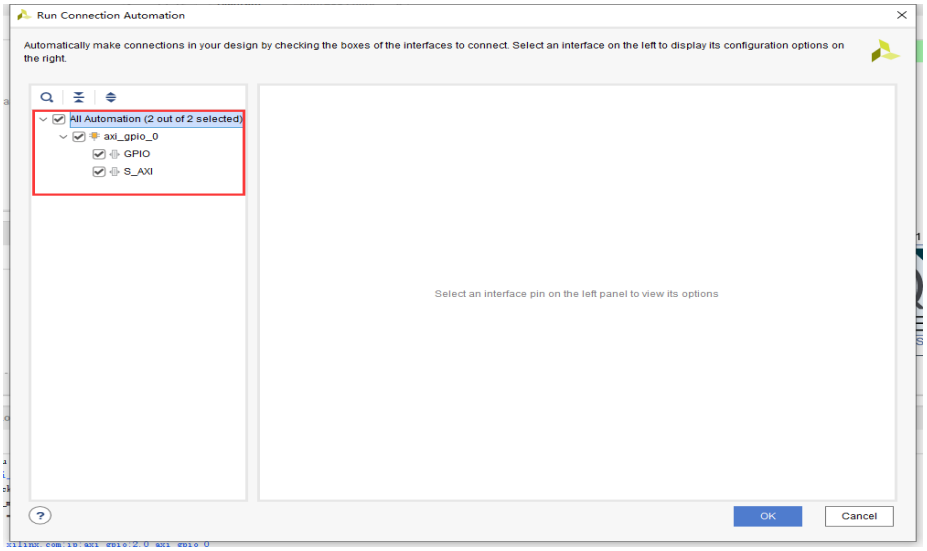
check all Outputs, GPIO Width to set to 1, and click OK.



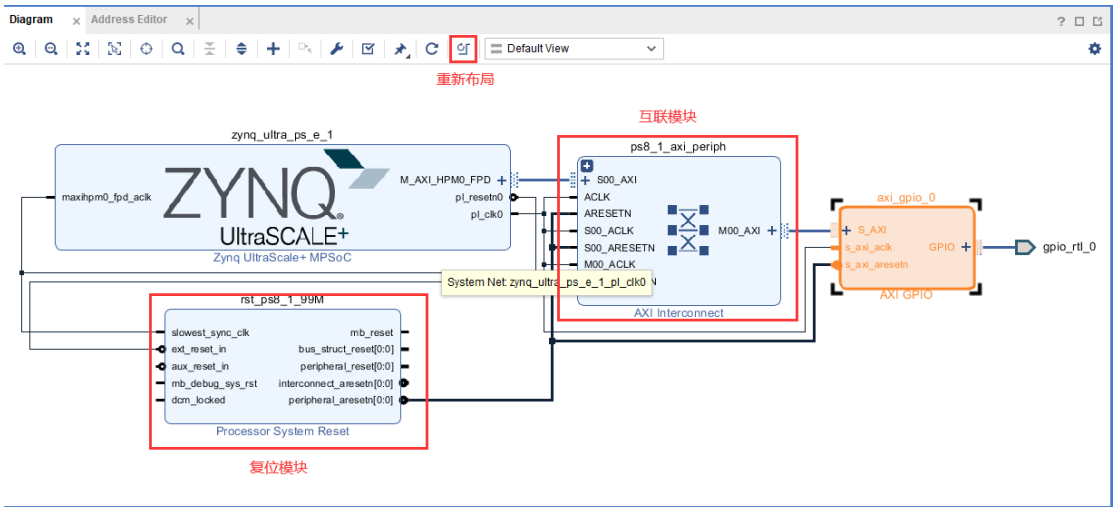
Click Run Block Automation - > OK to connect automatically



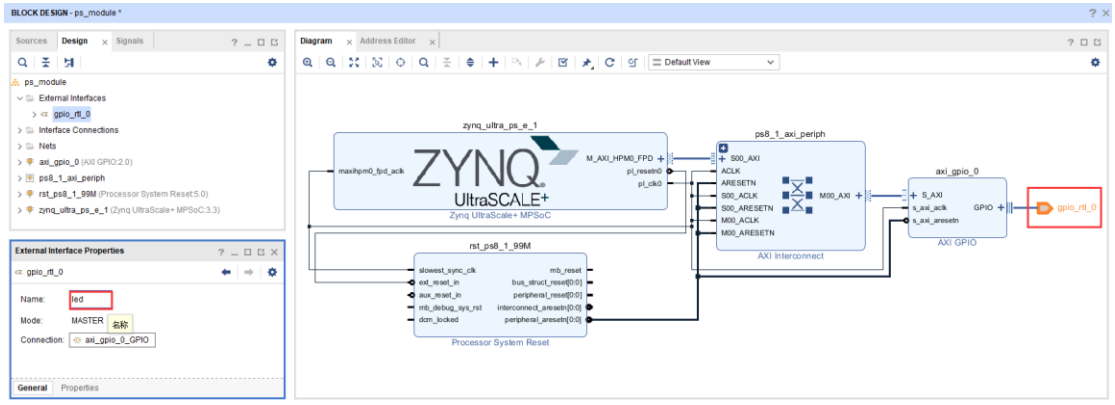
Check all the options and click OK



Click "optimize routing" to optimize the layout. At the same time, you can see that there are two more modules. One is the processor system reset module, which provides the reset signal of the same clock domain for the synchronous reset module. Axi interconnect module is an Axi bus interconnection module, which is used for the cross interconnection of Axi modules.

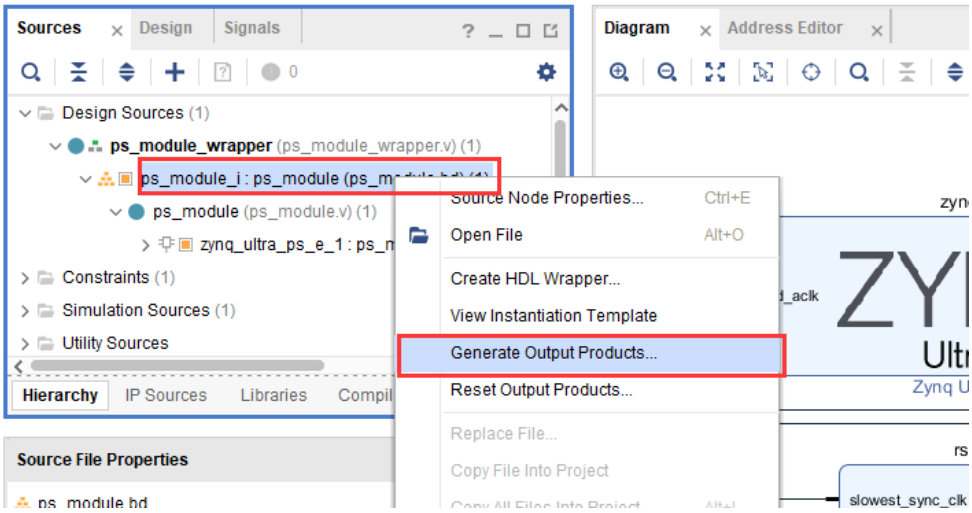


Change the name of GPIO port to led.

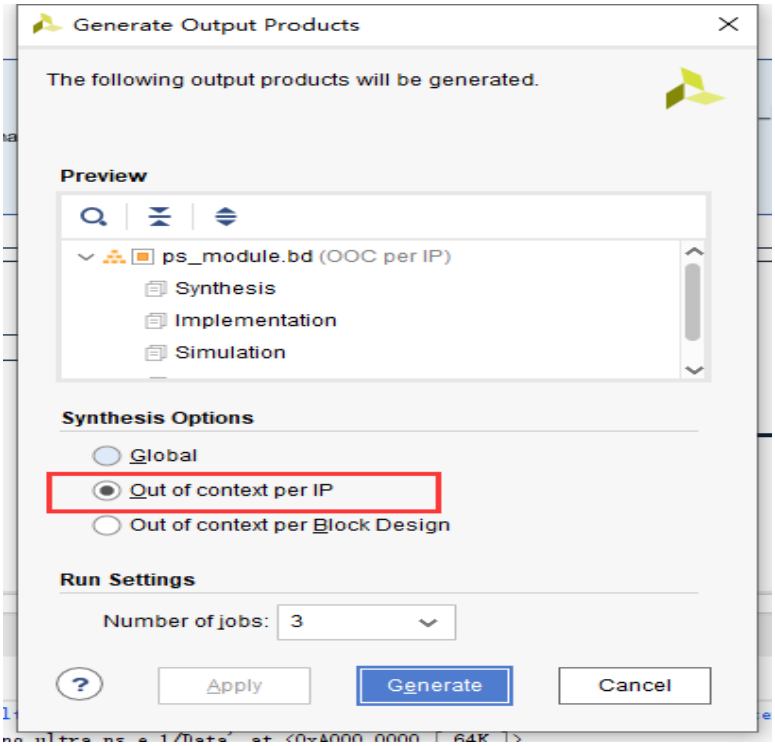


1.3 Generate synthesis files

Right click ps_module.bd ->Generate Ouput Products->Generate

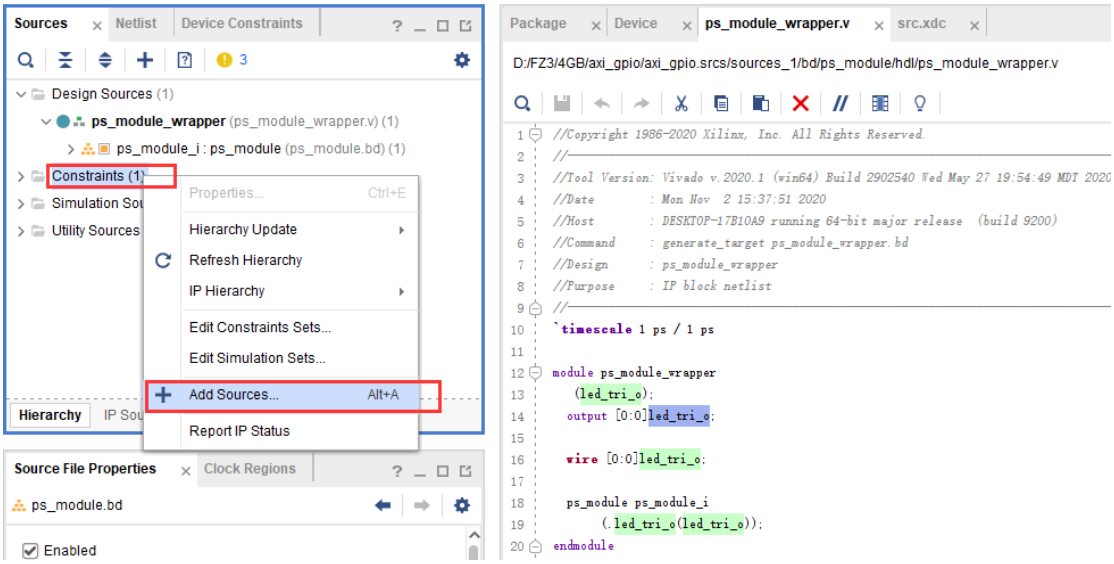


Select Out of context per IP and click generate

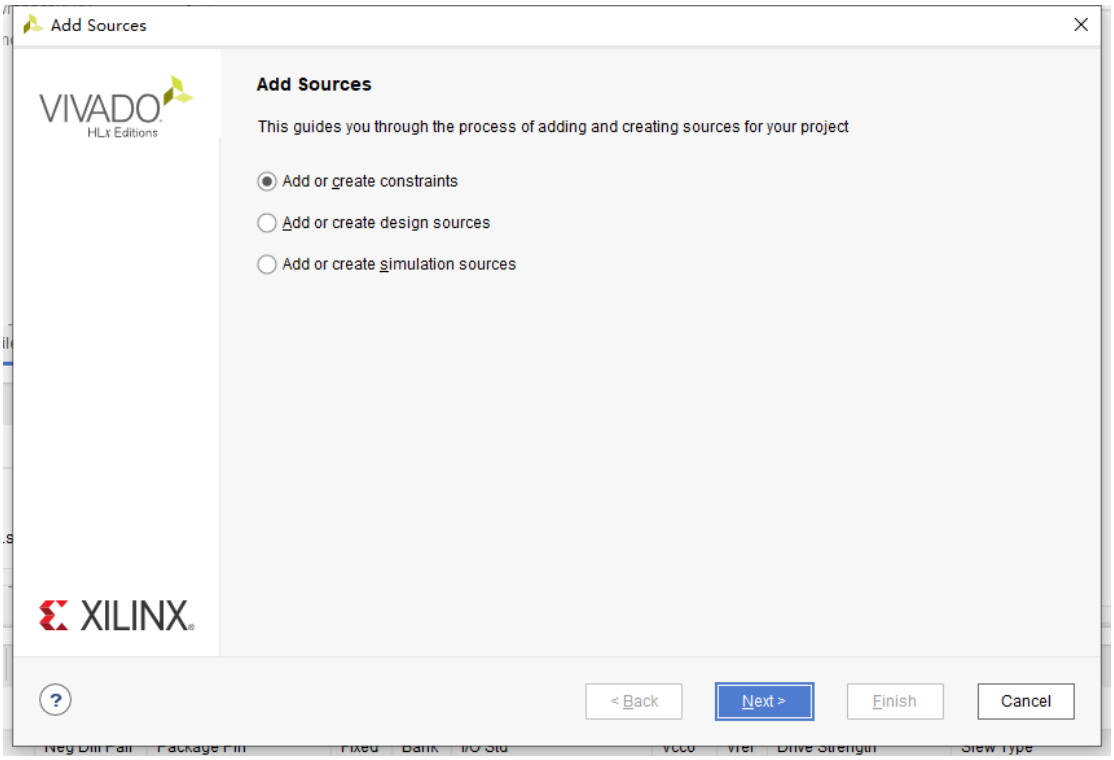


1.4 Adding xdc pin constraints

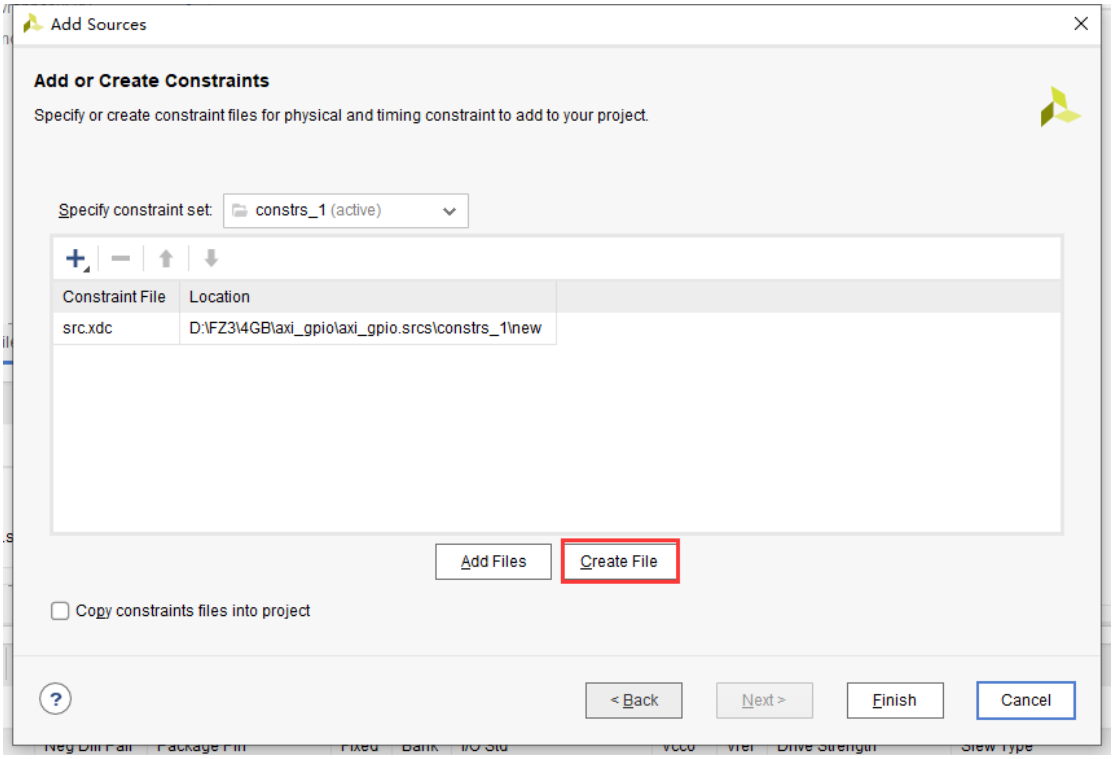
Right click Constraints-->Add Sources



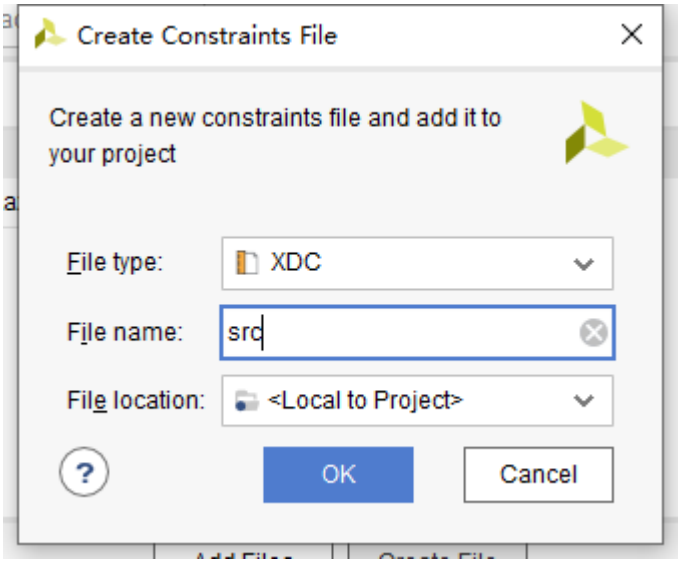
Select Add create constraints and click Next.



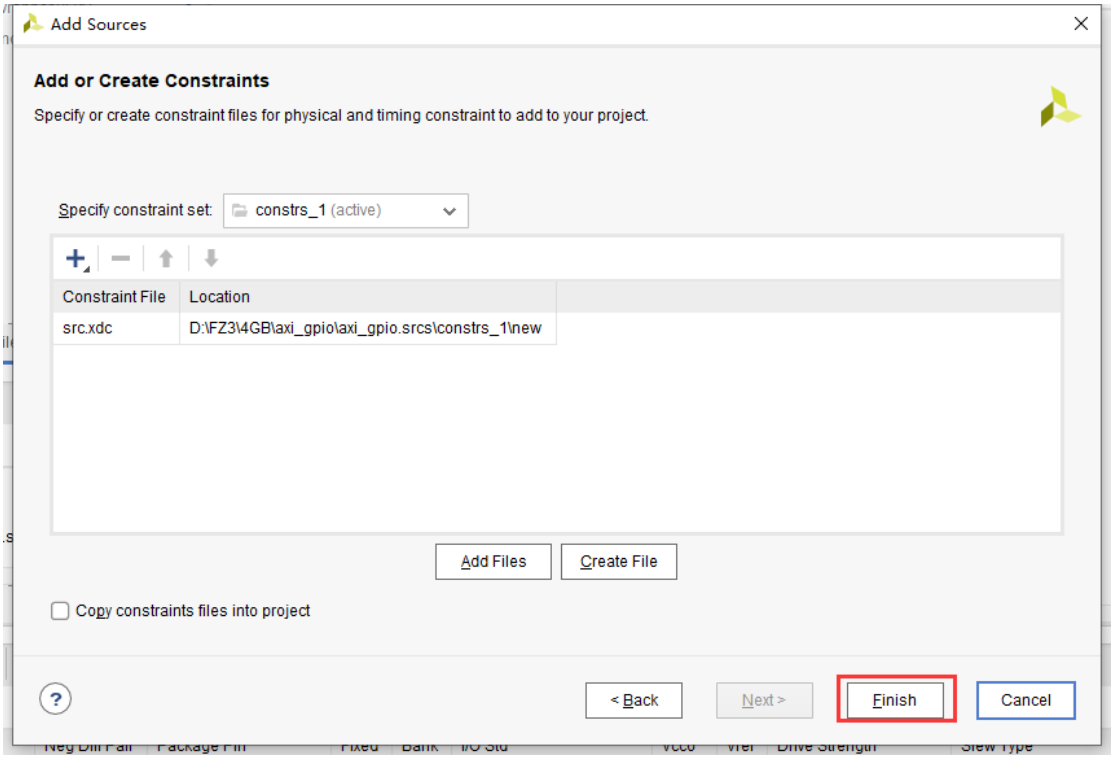
Click Create File



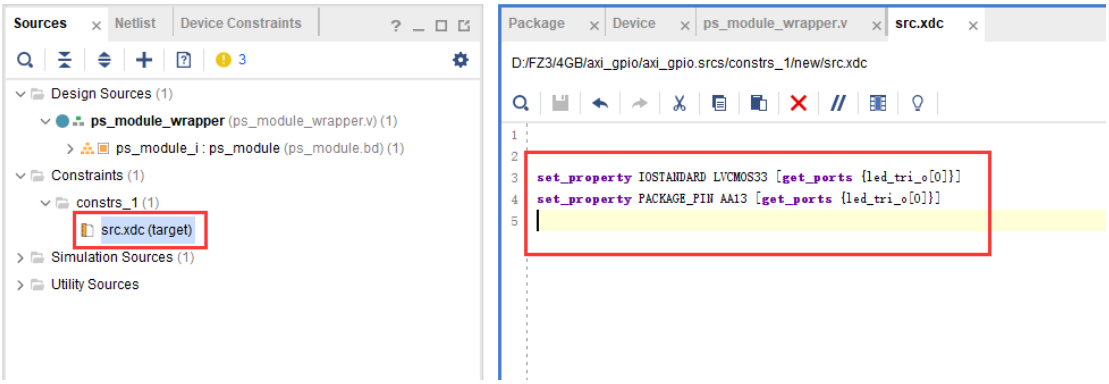
Enter the name of the new xdc file and click OK



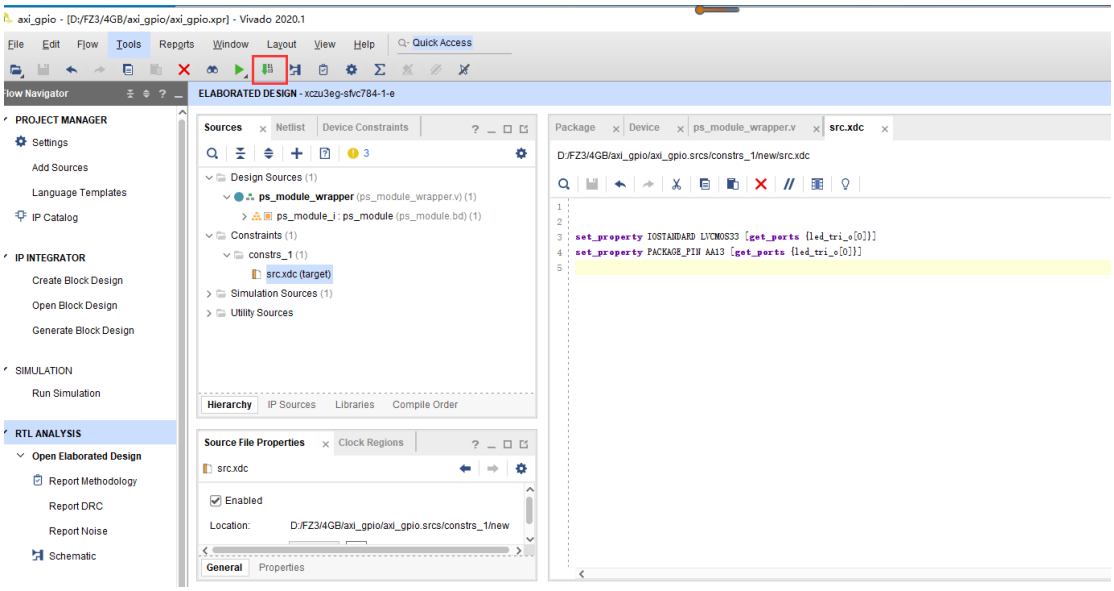
Click Finish


Add Sources
Add or Create Constraints
Specify or create constraint files for physical and timing constraint to add to your project.
Specify constraint set:
constrs_1 (active)
Constraint File
Location
src.xdc
D:\FZ3\4GB\axi_gpio\axi_gpio.srcs\constrs_1\new
Add Files
Create File
Copy constraints files into project
< Back
Next >
Finish
Cancel

Add xdc file

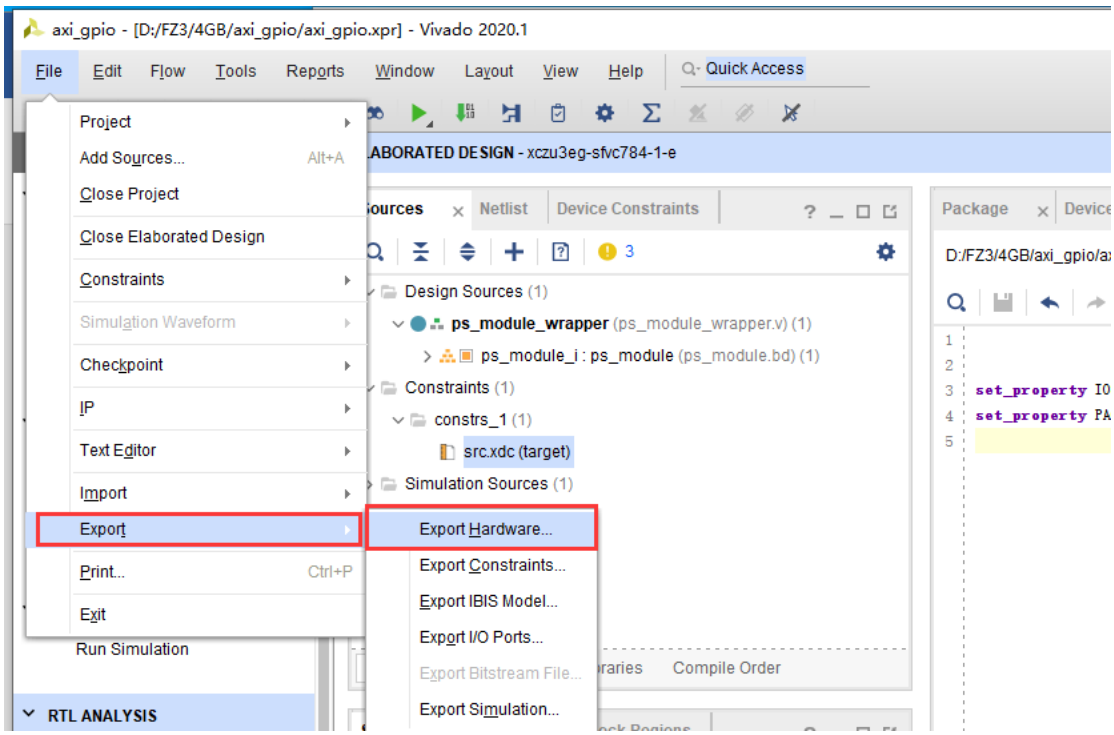

Sources
Netlist
Device Constraints
Design Sources (1)
ps_module_wrapper (ps_module_wrapper.v) (1)
ps_module_i : ps_module (ps_module.bd) (1)
Constraints (1)
constrs_1 (1)
src.xdc (target)
Simulation Sources (1)
Utility Sources
Package
Device
ps_module_wrapper.v
src.xdc
D:/FZ3/4GB/axi_gpio/axi_gpio.srcs/constrs_1/new/src.xdc
set_property IOSTANDARD LVCMOS33 [get_ports {led_tri_o[0]}]
set_property PACKAGE_PIN AA13 [get_ports {led_tri_o[0]}]

1.5 Generate bit files



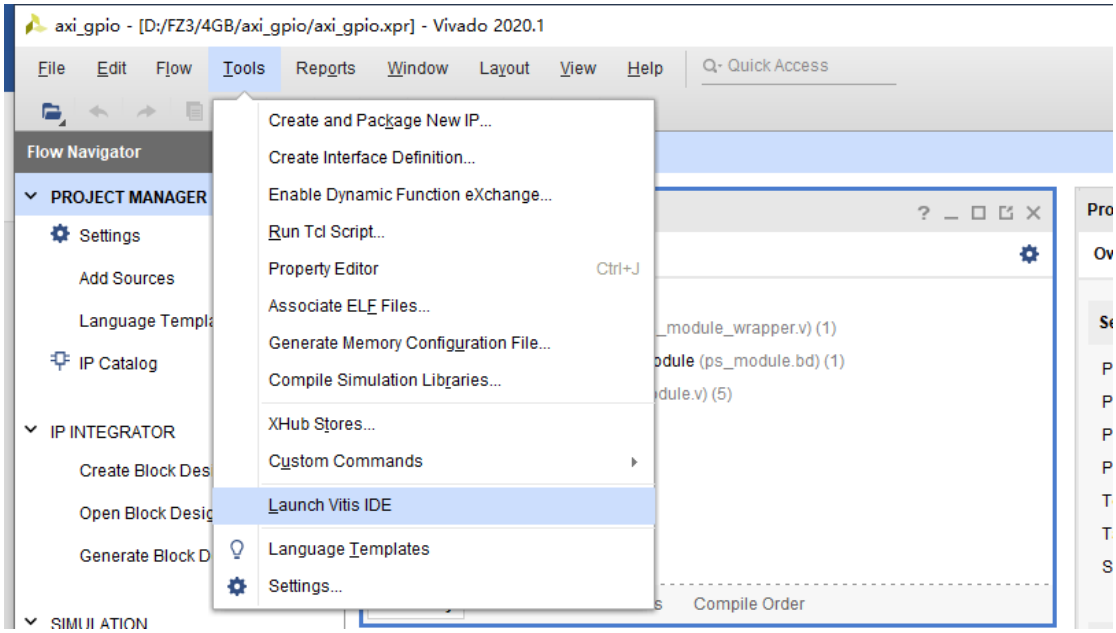
1.6 Export Hardware Profile

Click File - > Export - > Export Hardware - > OK on the menu bar to export the hardware configuration file

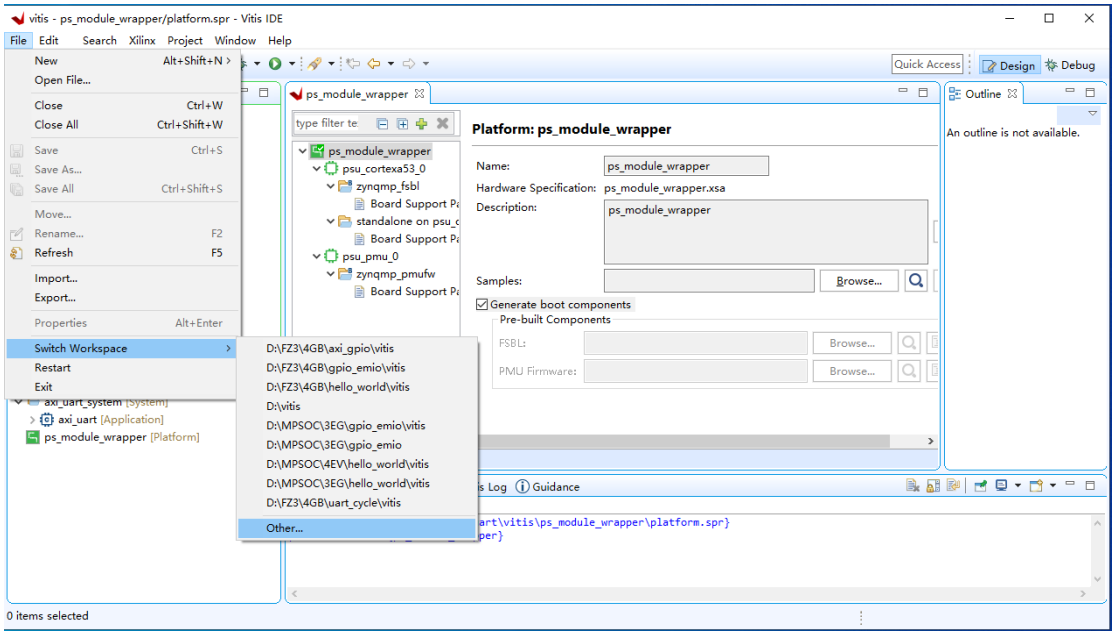


1.7 Launch Vitis and create new platform project

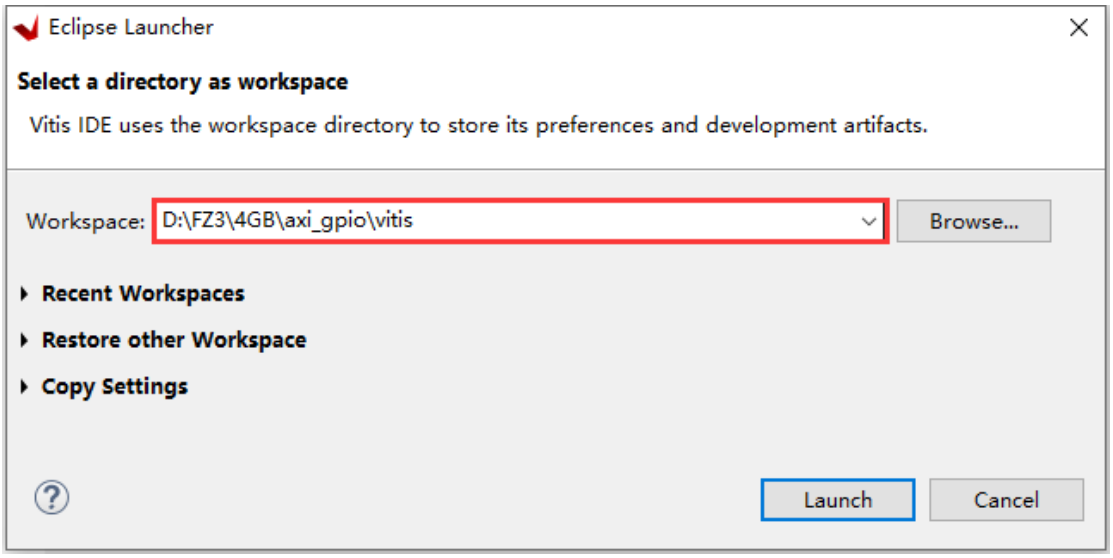
Click Tools > launch Vitis ide on the menu bar to launch Vitis



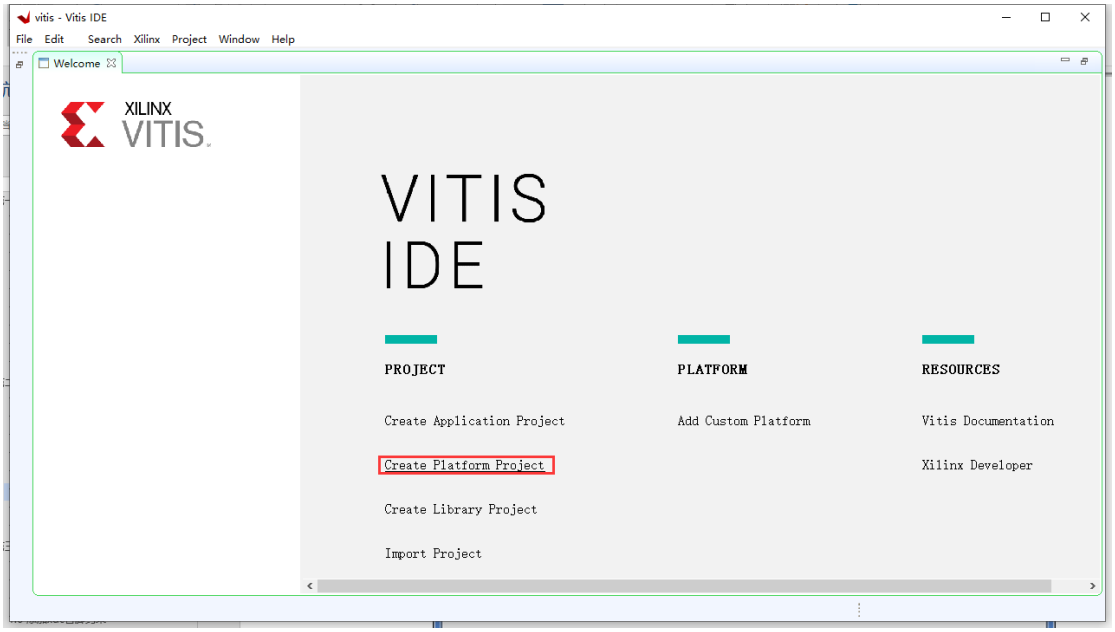
Click File→Switch Workspace→Other…,



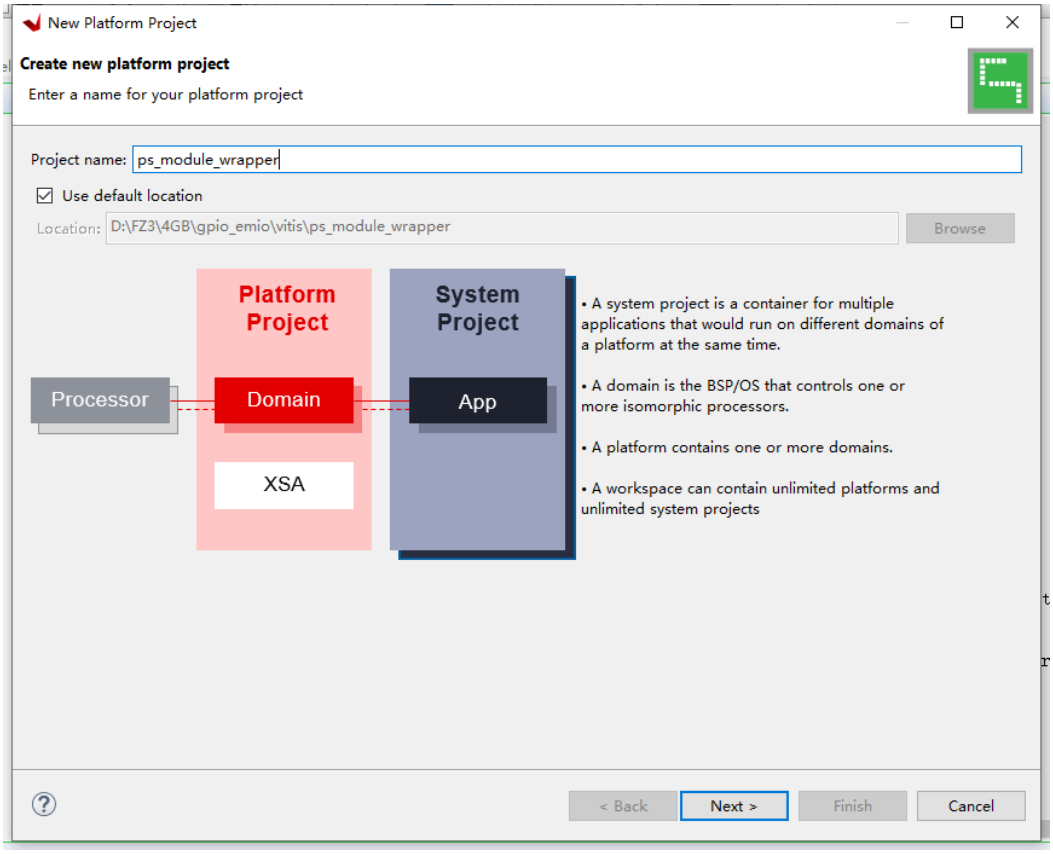
elect the path, select the vitis folder under gpio_emio project. Click launch.



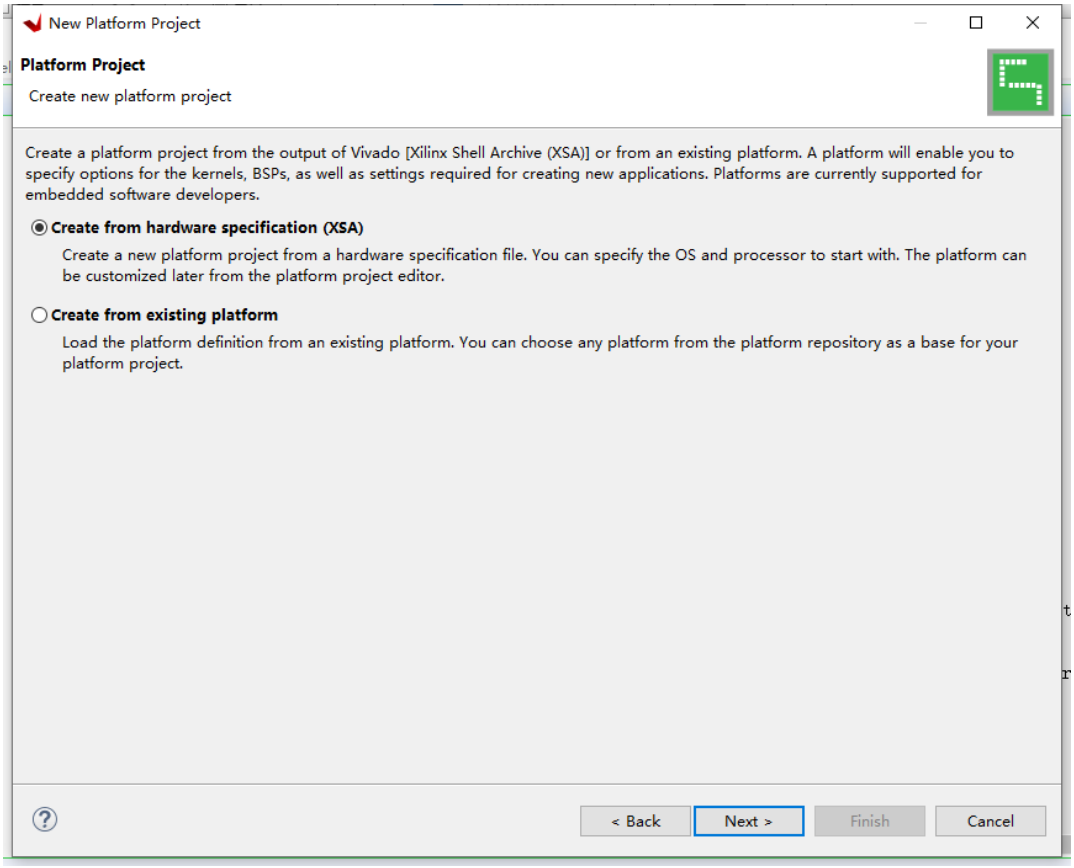
Click Create Platform Project



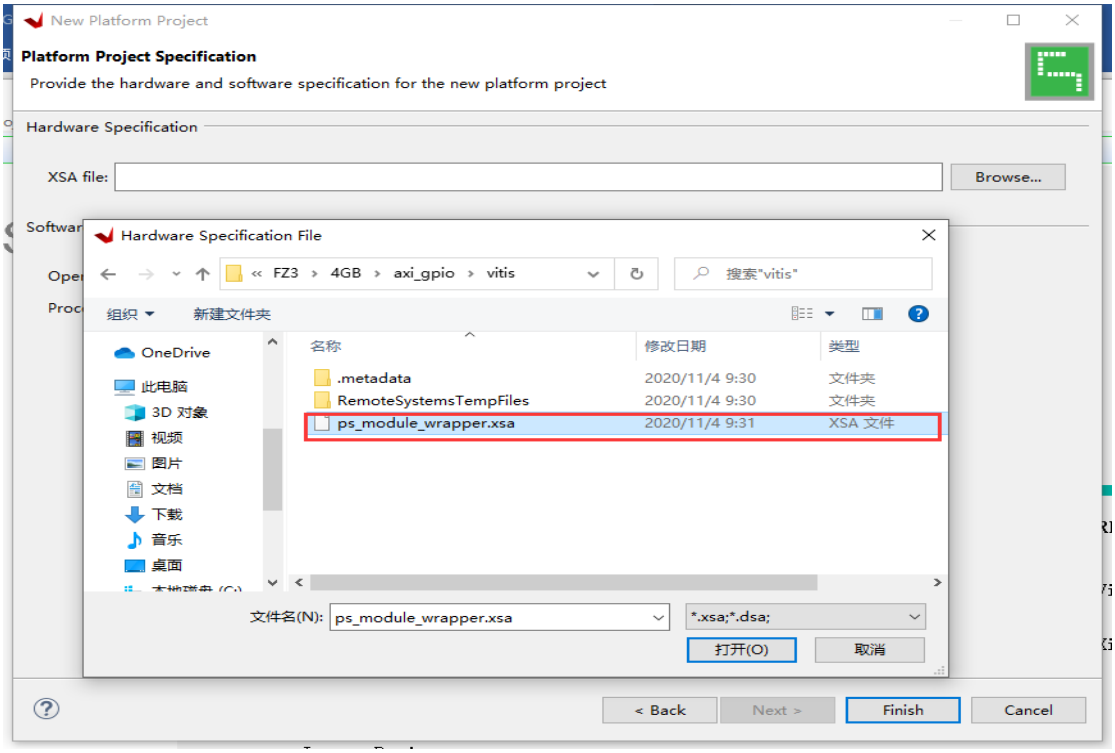
The project name uses the same name as the xsa file. Click next.



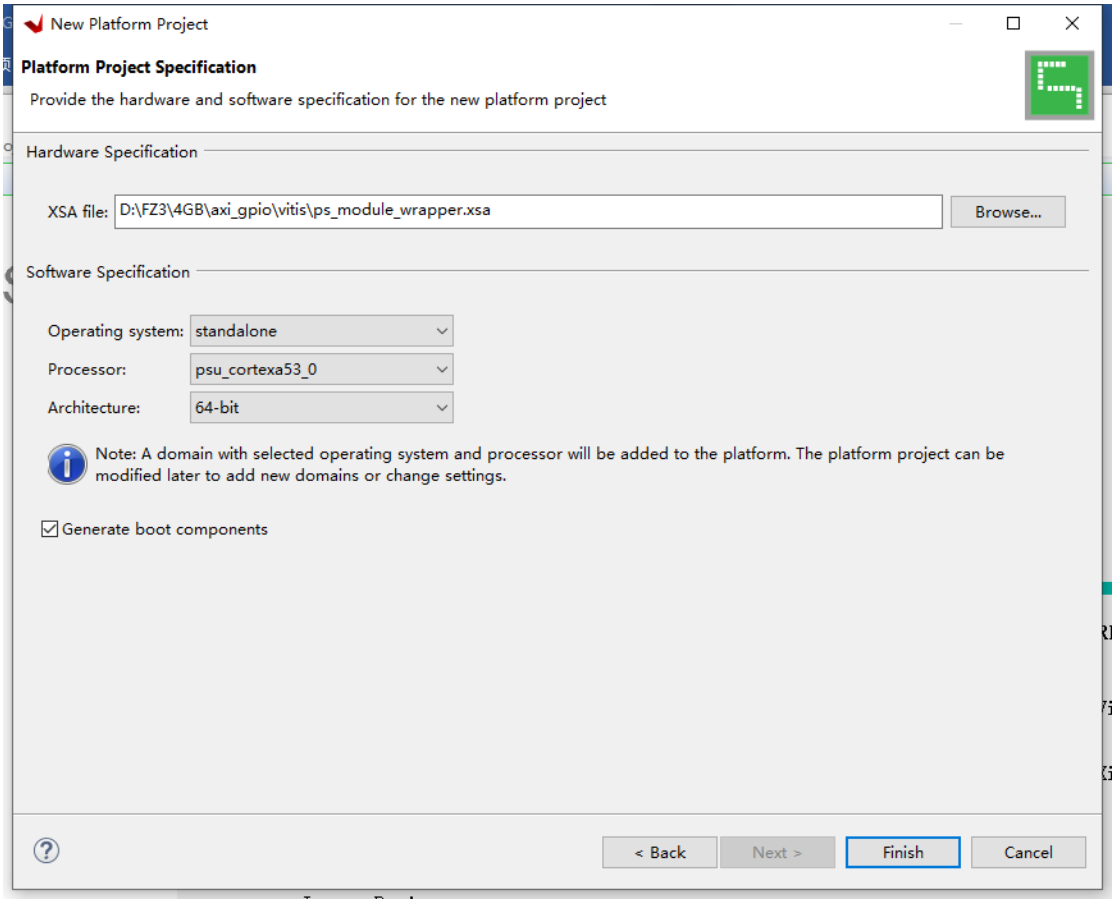
Select Create from hardware specification(XSA),click NEXT.



Open Browse… , select the previously generated xsa file.

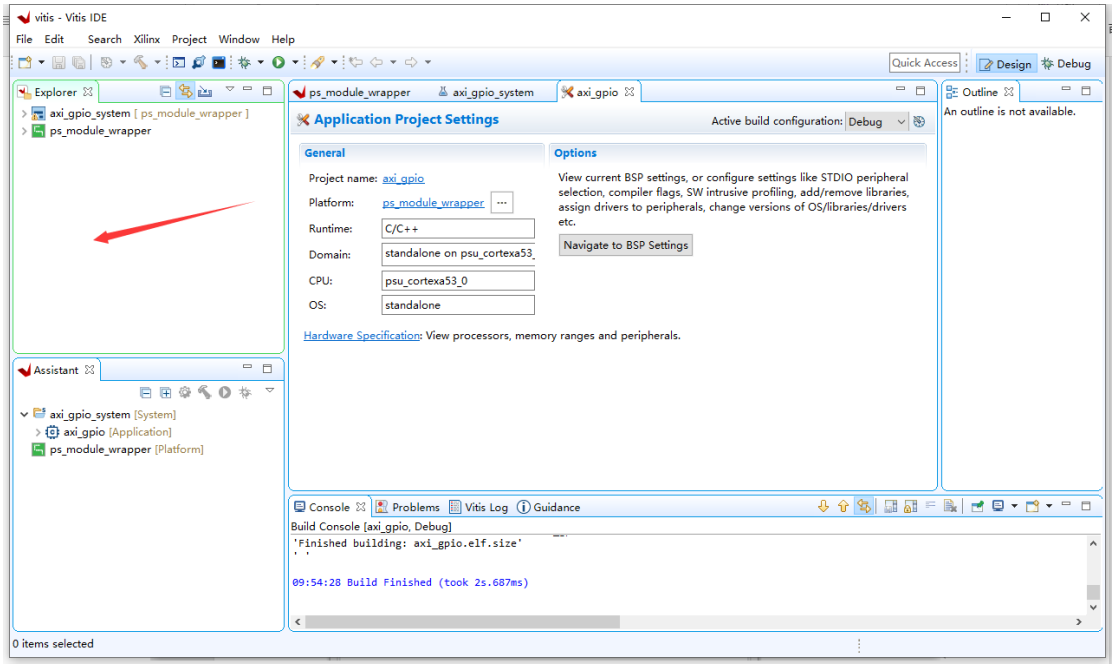


Leave the default and click finish.

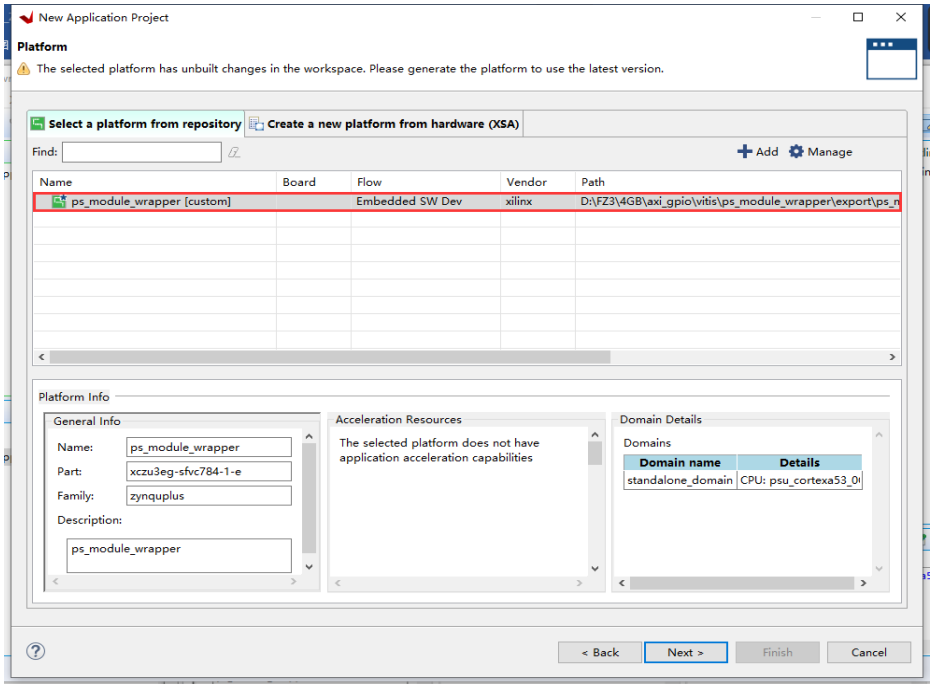


1.8 New axi_gpio Project

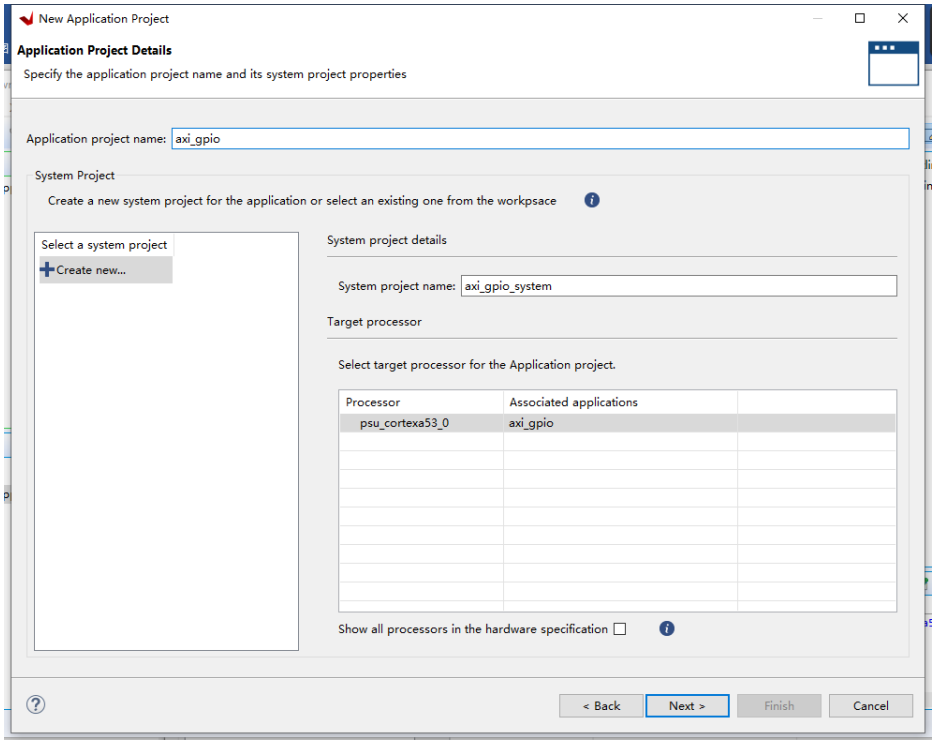
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



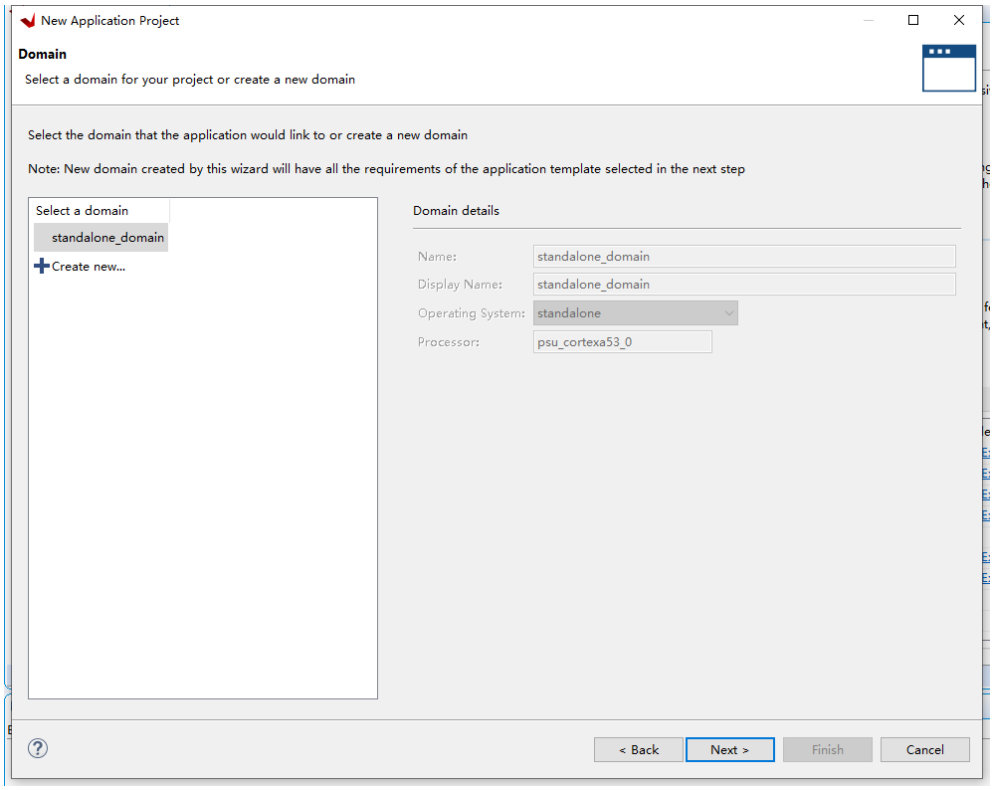
Skip to the second page and select the platform project created above (default), next.



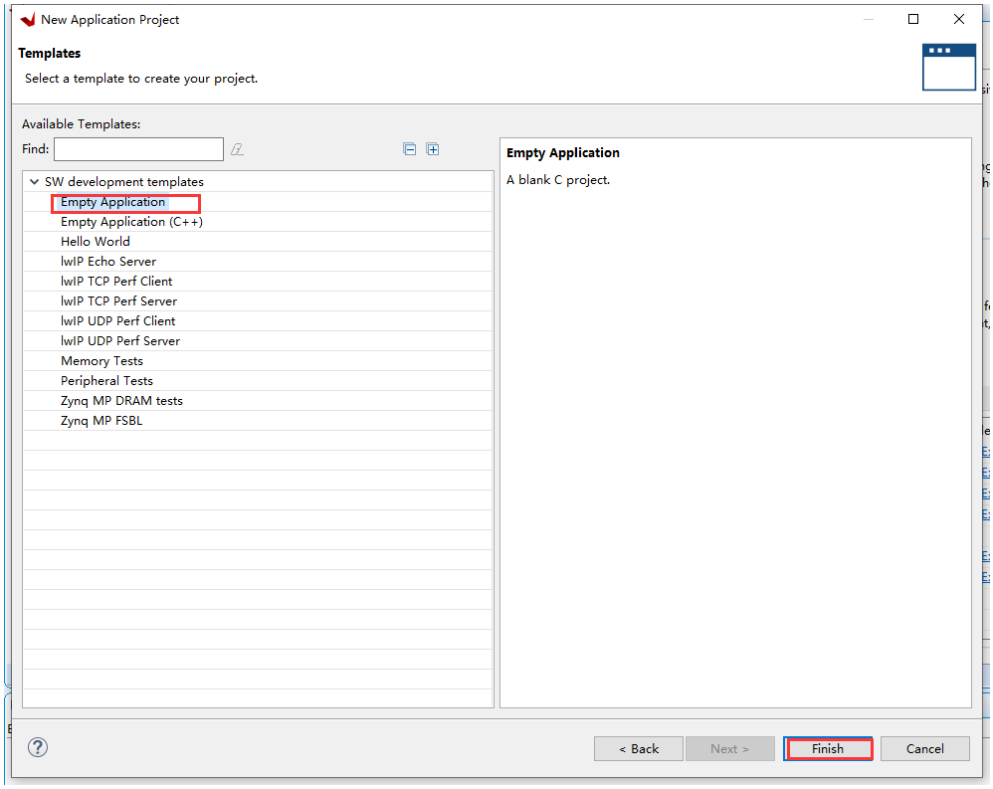
Project name enter axi_gpio， Next



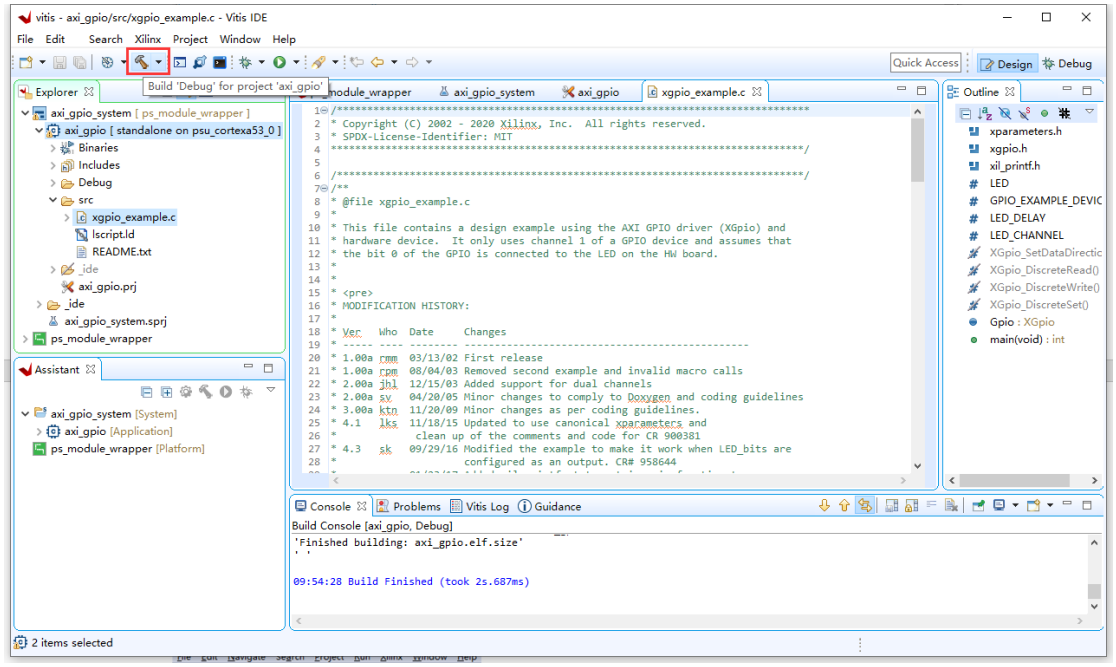
Next



Select Empty Application， Finish

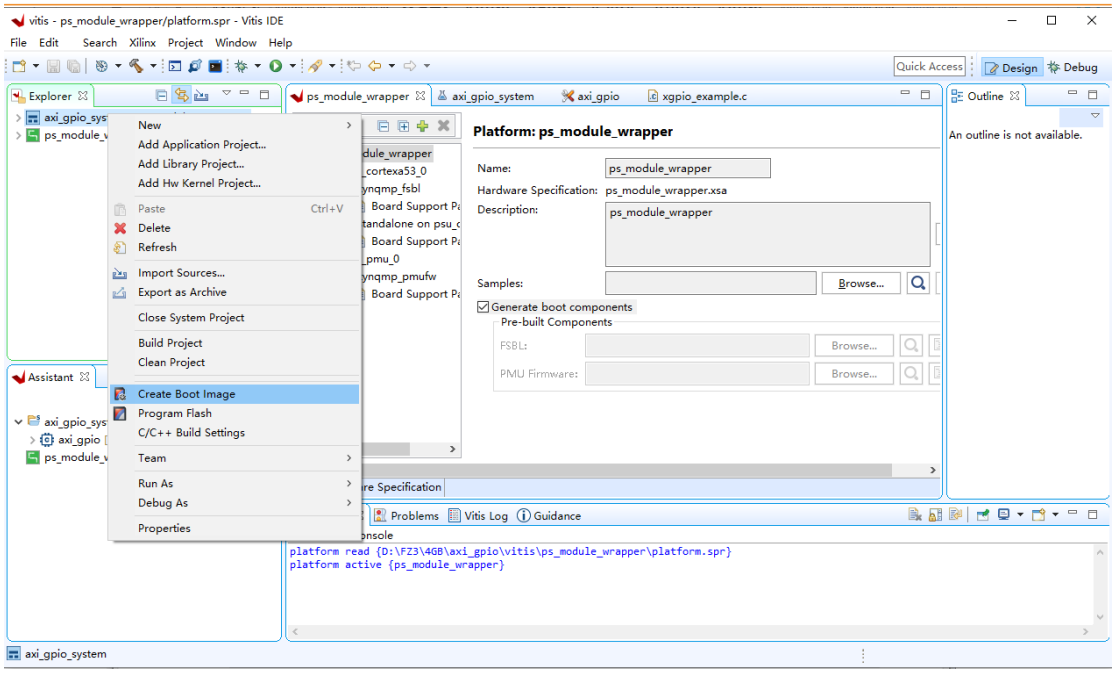


Copy the files in the example project to src, select the project, and click the compile buttom.

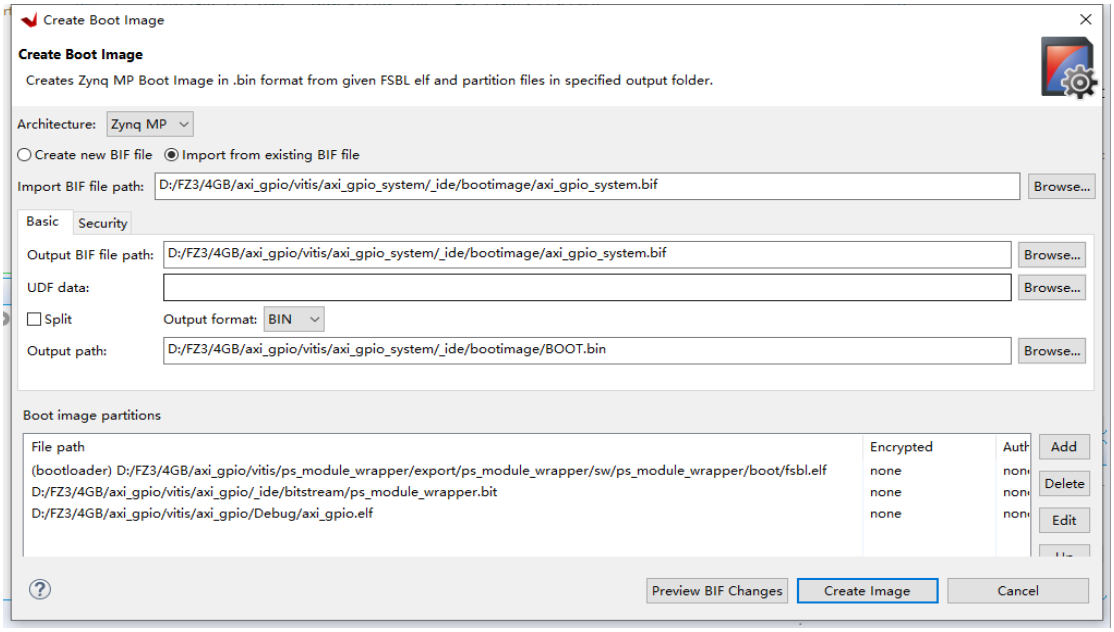


1.9 Generate BOOT.bin file

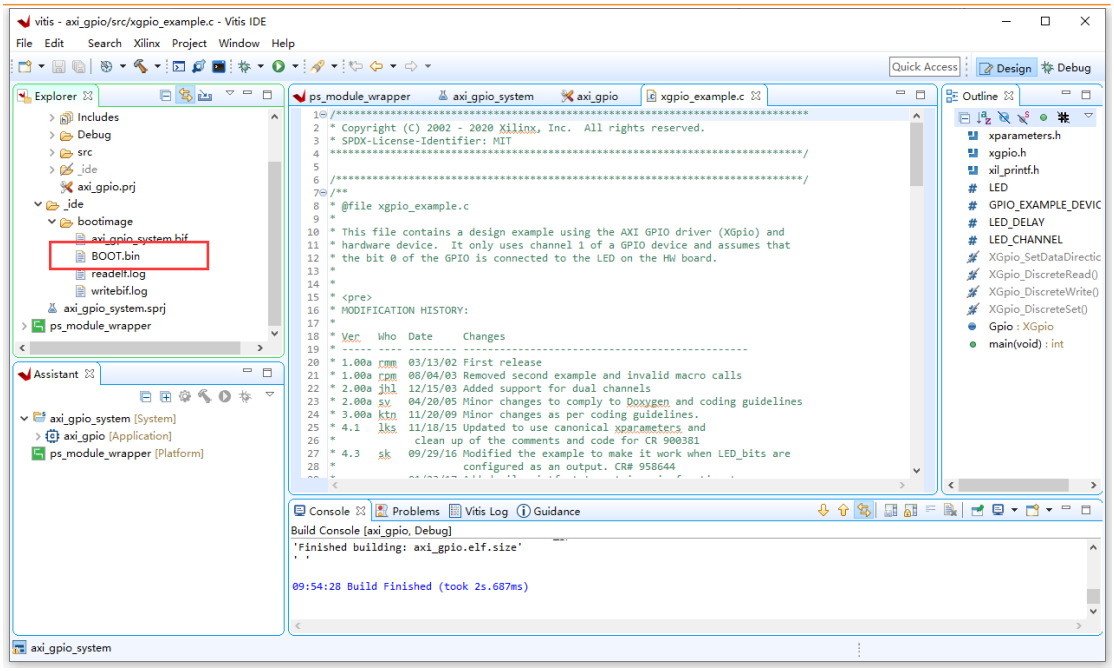
Right-click the system of APP project and select Create boot image.



Click Create Image to generate BOOT.bin Startup file.



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.


vitis - axi_gpio/src/xgpio_example.c - Vitis IDE
File Edit Search Xilinx Project Window Help
Quick Access
Design
Debug
Explorer
Includes
Debug
src
_ide
axi_gpio.prj
_ide
bootimage
axi_gpio_system.bif
BOOT.bin
readelf.log
writebif.log
axi_gpio_system.sprj
ps_module_wrapper
Assistant
axi_gpio_system [System]
axi_gpio [Application]
ps_module_wrapper [Platform]
ps_module_wrapper
axi_gpio_system
axi_gpio
xgpio_example.c
Outline
xparameters.h
xgpio.h
xil_printf.h
LED
GPIO_EXAMPLE_DEVIC
LED_DELAY
LED_CHANNEL
XGpio_SetDataDirectic
XGpio_DiscreteRead()
XGpio_DiscreteWrite()
XGpio_DiscreteSet()
Gpio : XGpio
main(void) : int
Console
Problems
Vitis Log
Guidance
Build Console [axi_gpio, Debug]
'Finished building: axi_gpio.elf.size'
09:54:28 Build Finished (took 2s.687ms)
axi_gpio_system

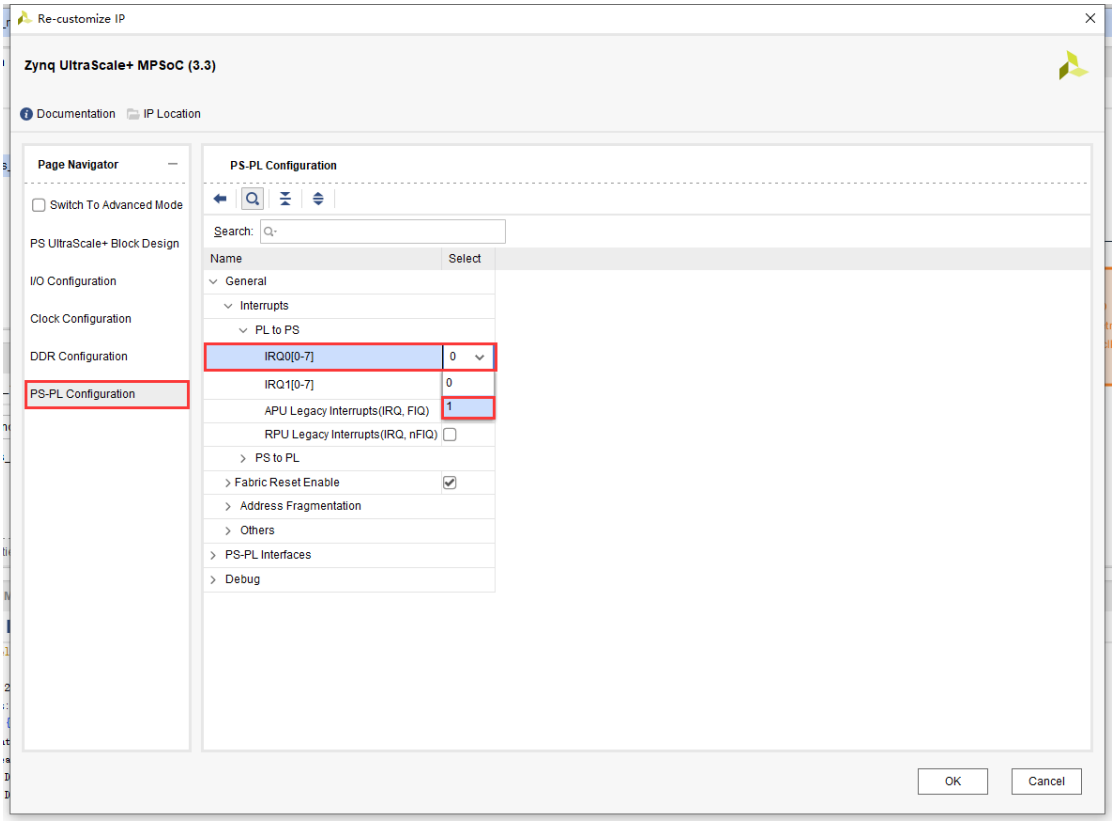
Chapter 4 axi_uart

This project mainly uses axi_uart core provided by Xilinx for rs485 data transmission.

1.1 Create project and configure IP core of PS

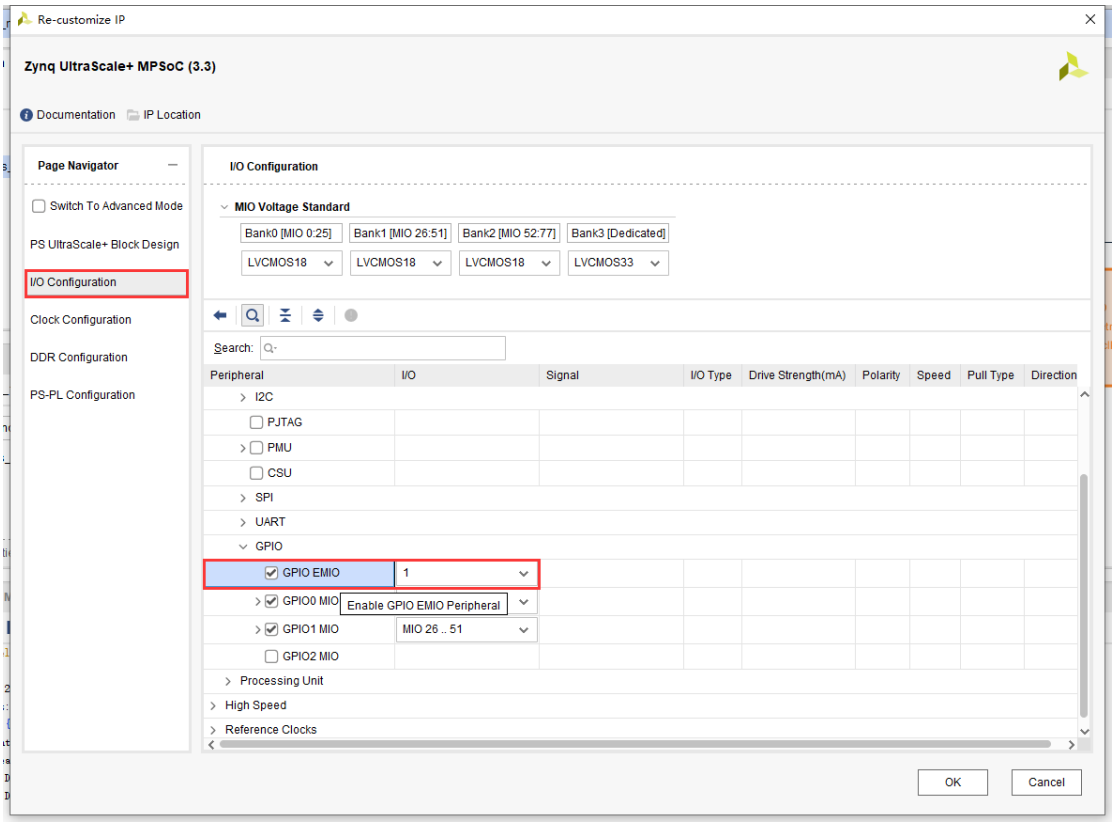
Based on "ps_hello" project, save as axi_ uart project.

Open PL interrupt.

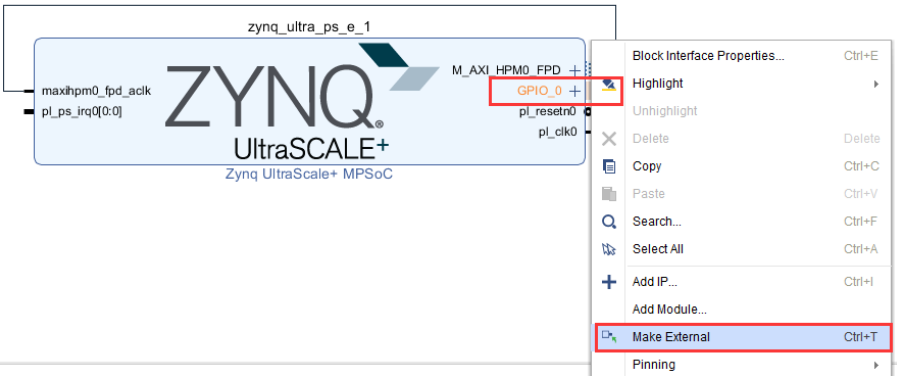


Check GPIO EMIO and select the bit width of EMIO as 1 bit in MIO configuration.

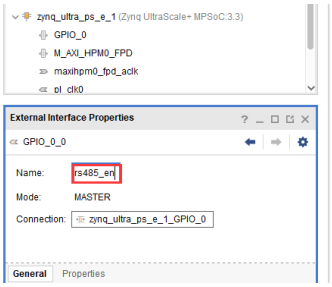
Click OK

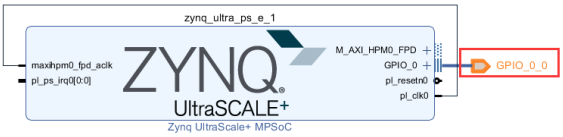


Click GPIO_ 0 port, Right click and select make external to export the port signal.



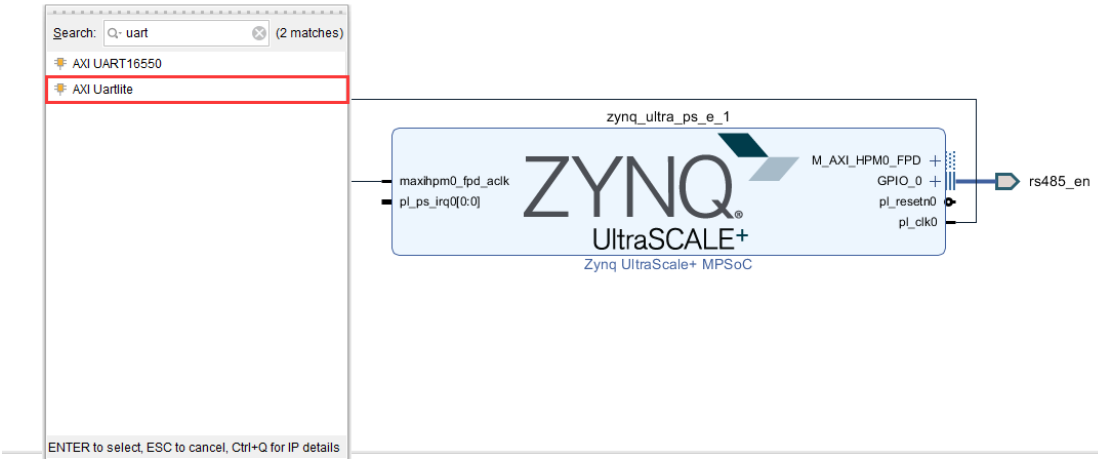
Click on the pin and change the pin name to rs485_en, and save the design.



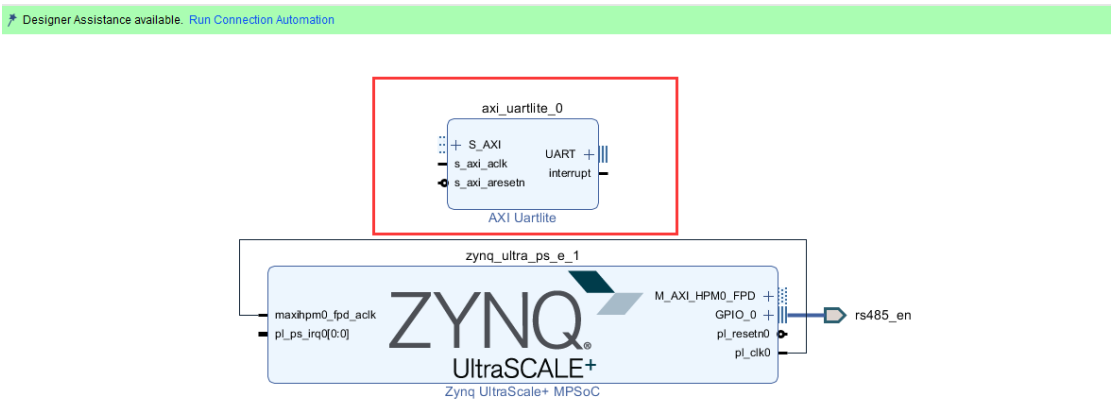


1.2 Add axi_uart IP core and configure it

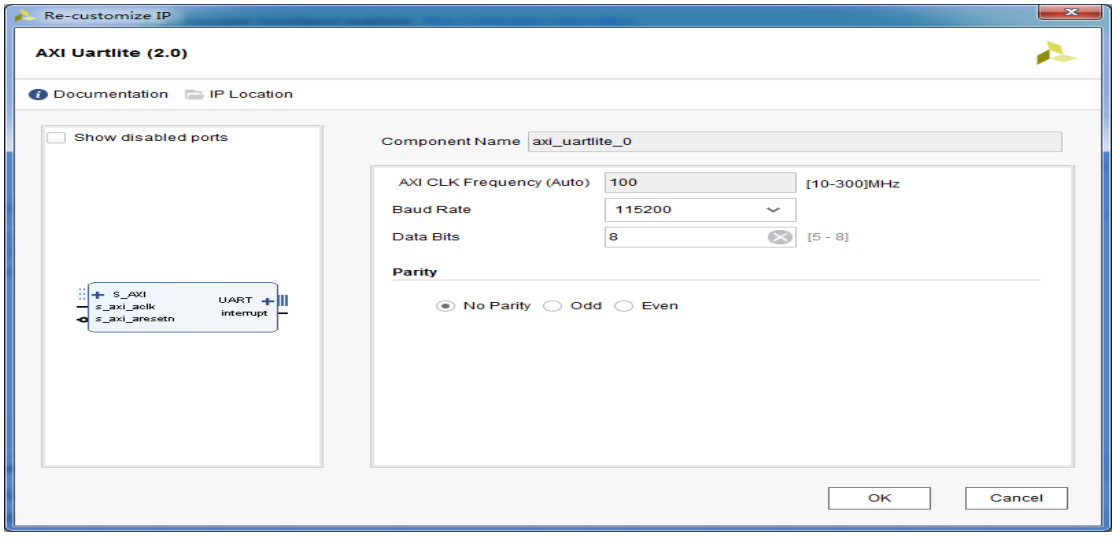
Enter UART in the search bar and double-click AXI Uartlite to add the axi_uart core.



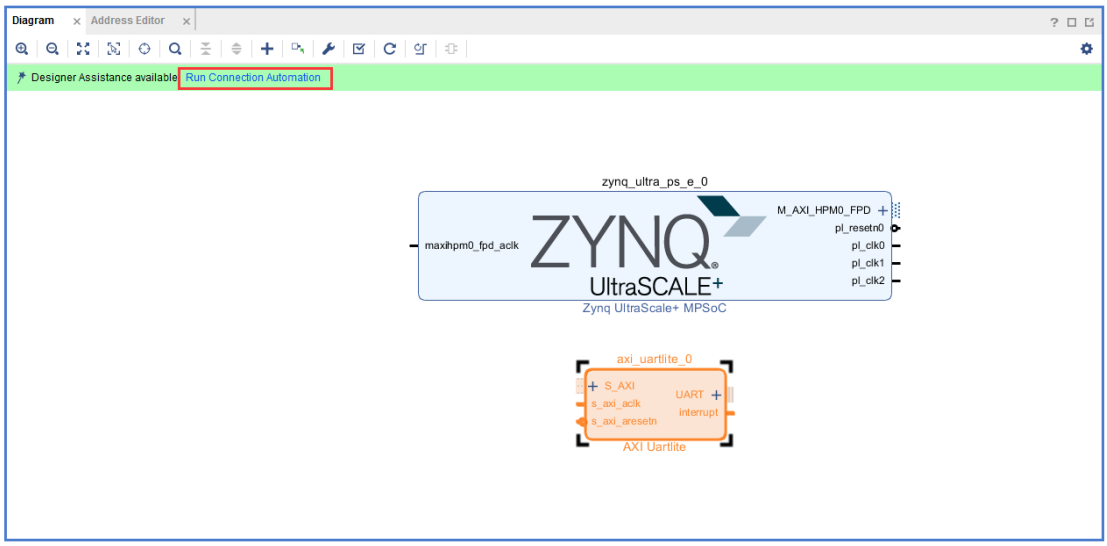
The added axi_uart core is shown in the following figure



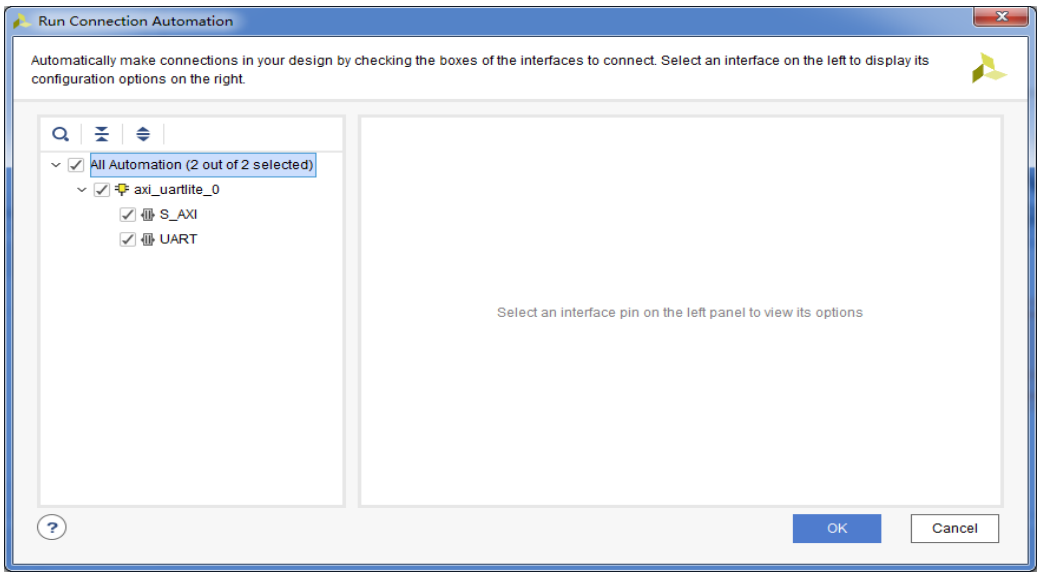
Double-click the axi_uart core to set the baud rate to 115200



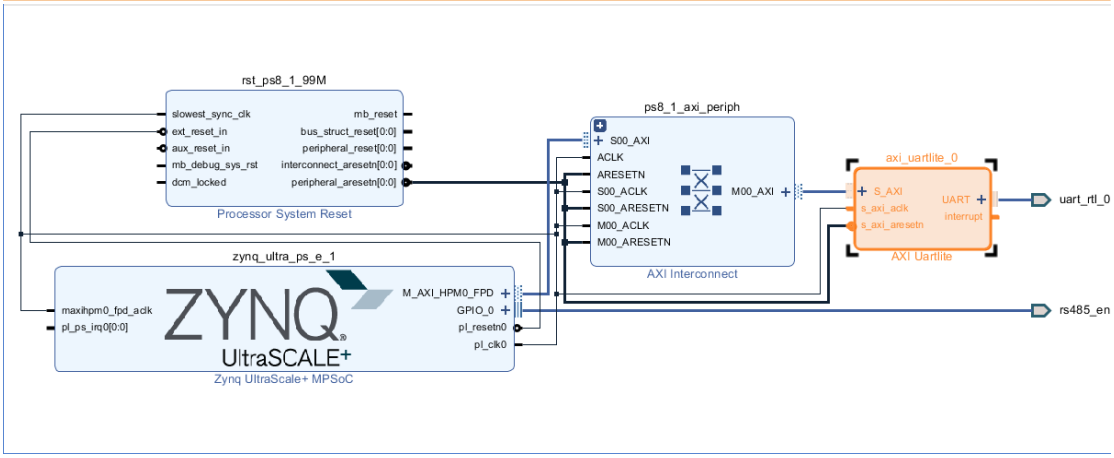
Click Run Block Automation - > OK to connect automatically



Check all the options and click OK

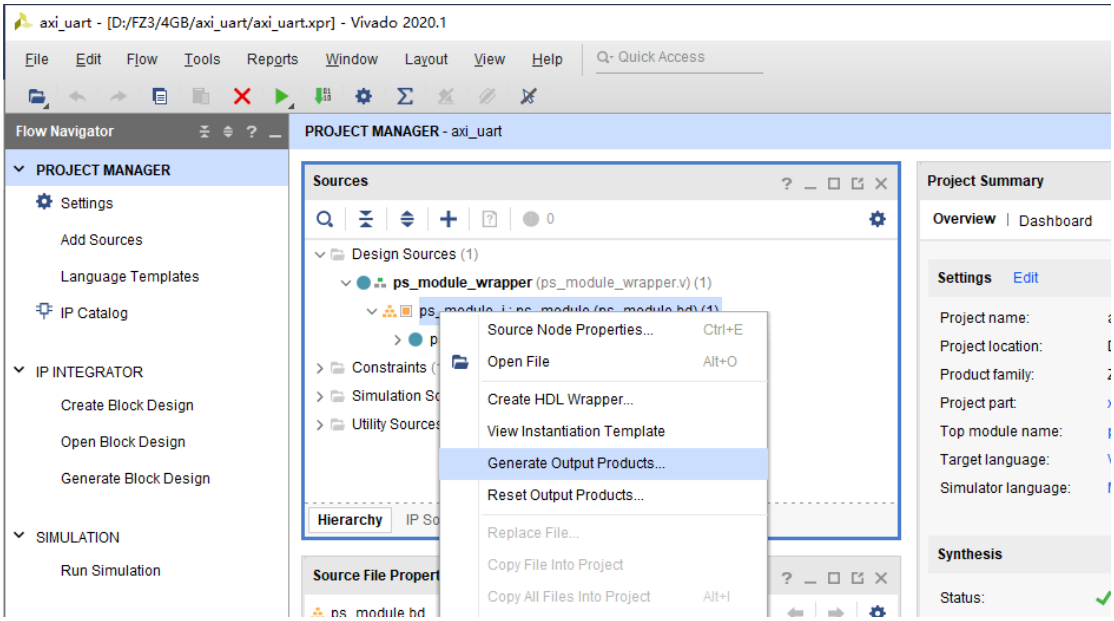


After the automatic connection is completed, the following figure is shown.

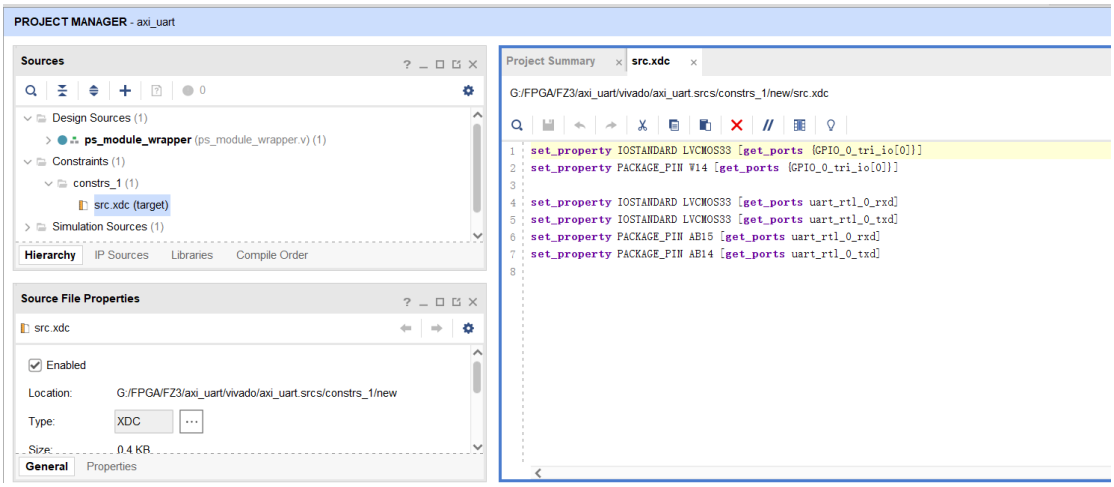


1.3 Generate synthesis files

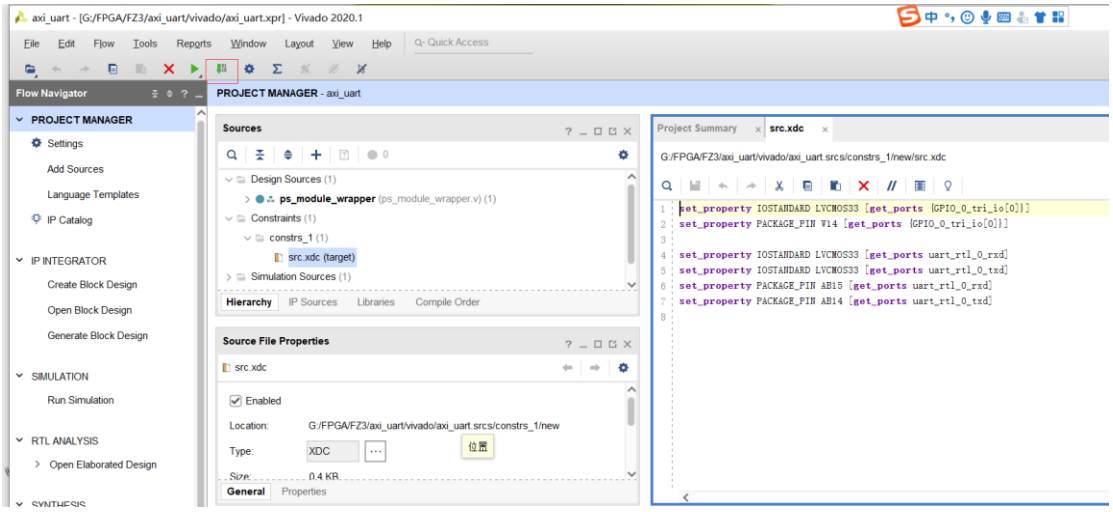
Right click ps_module.bd ->Generate Ouput Products->Generate



1.4 Adding xdc pin constraints

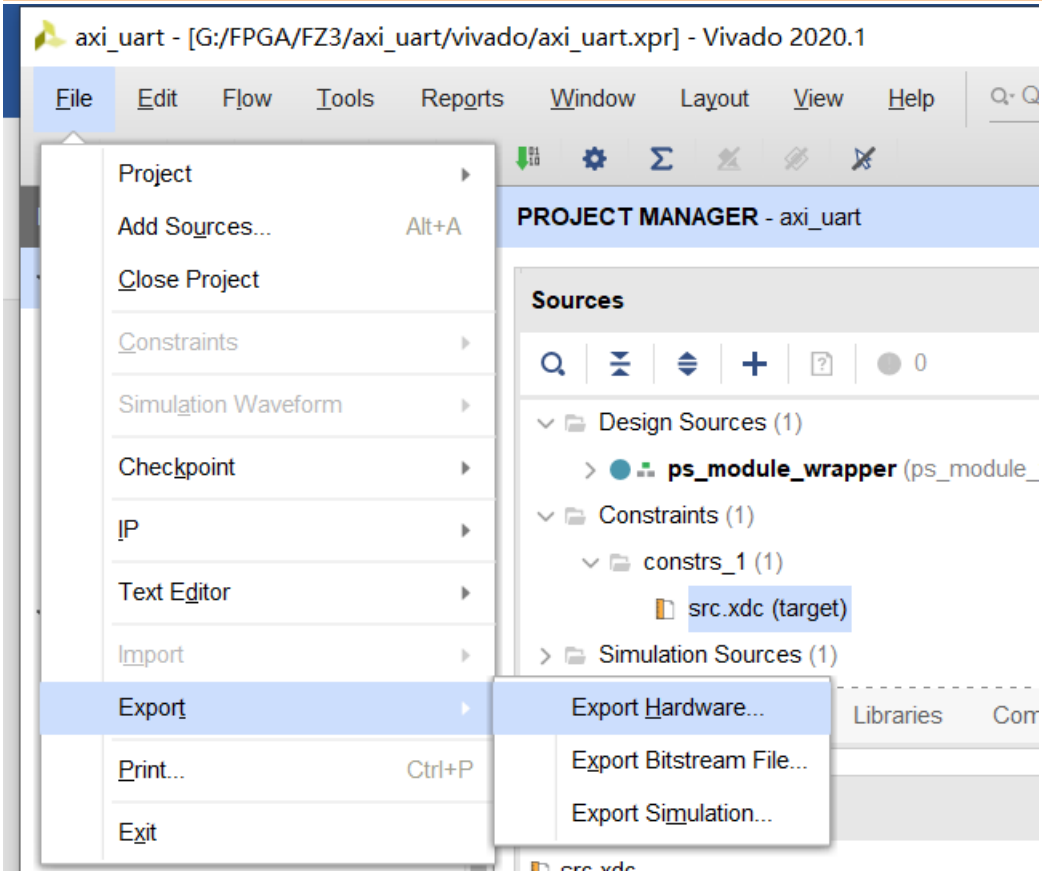


1.5 Generate bit files



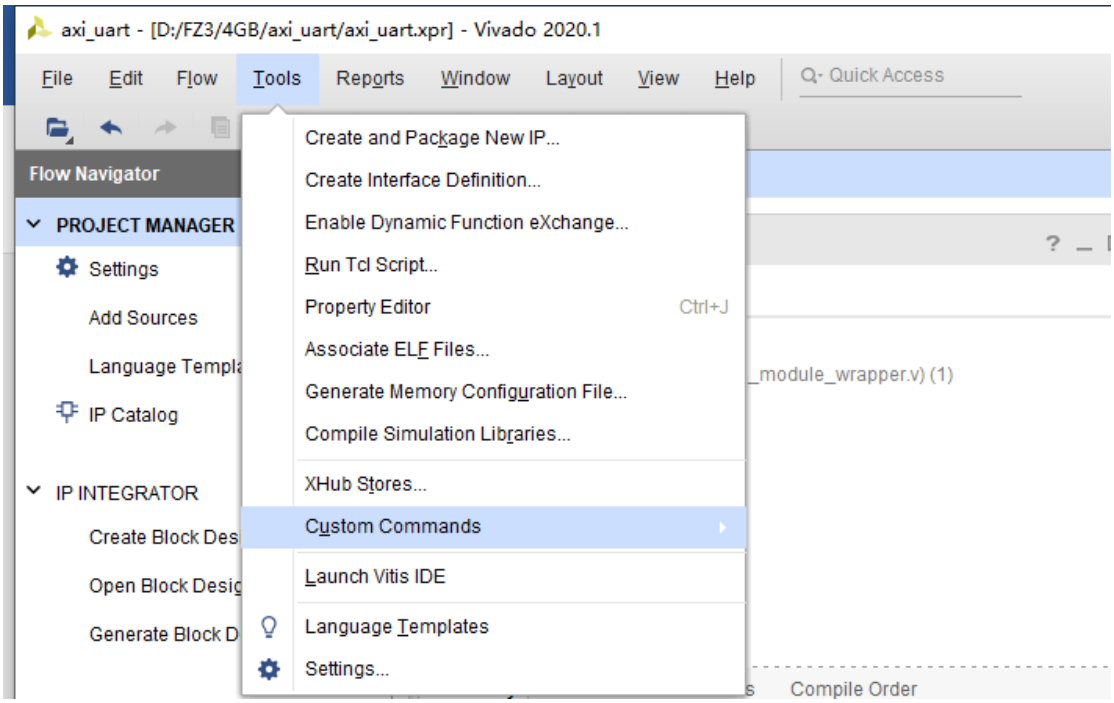
1.6 Export Hardware Profile

点击菜单栏上的 File->Export->Export Hardware->OK 导出硬件配置文件

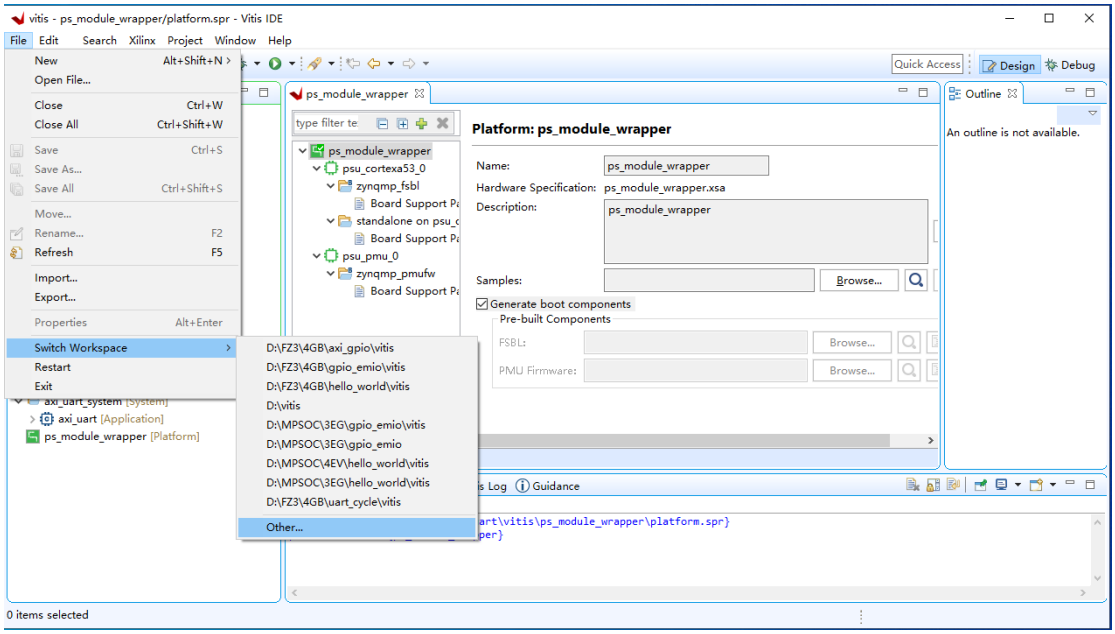


1.7 Launch Vitis and create new platform project

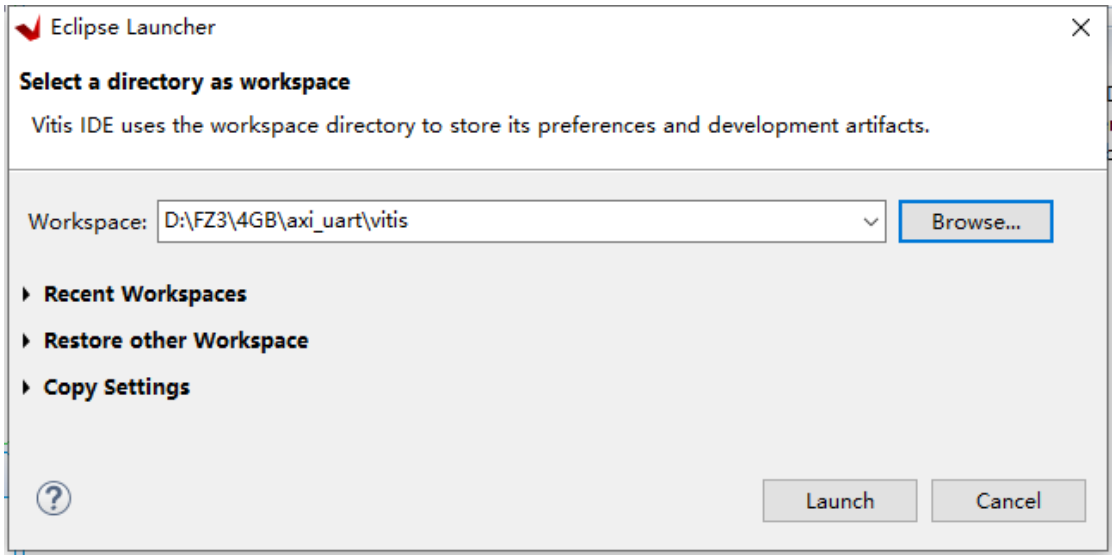
Click Tools > launch Vitis ide on the menu bar to launch Vitis



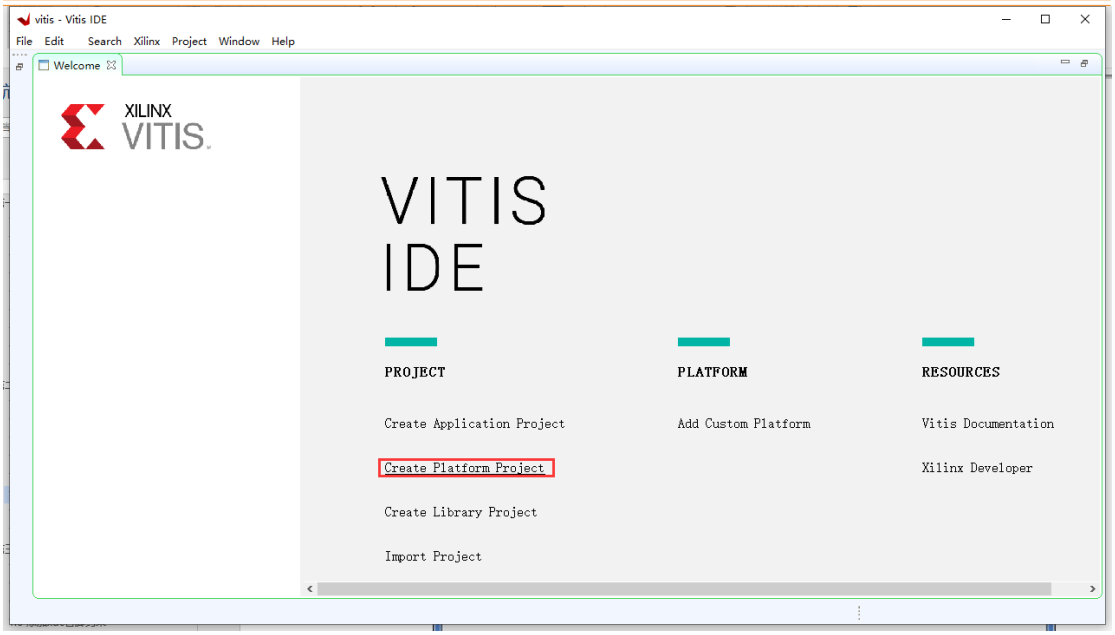
Click File→Switch Workspace→Other…,



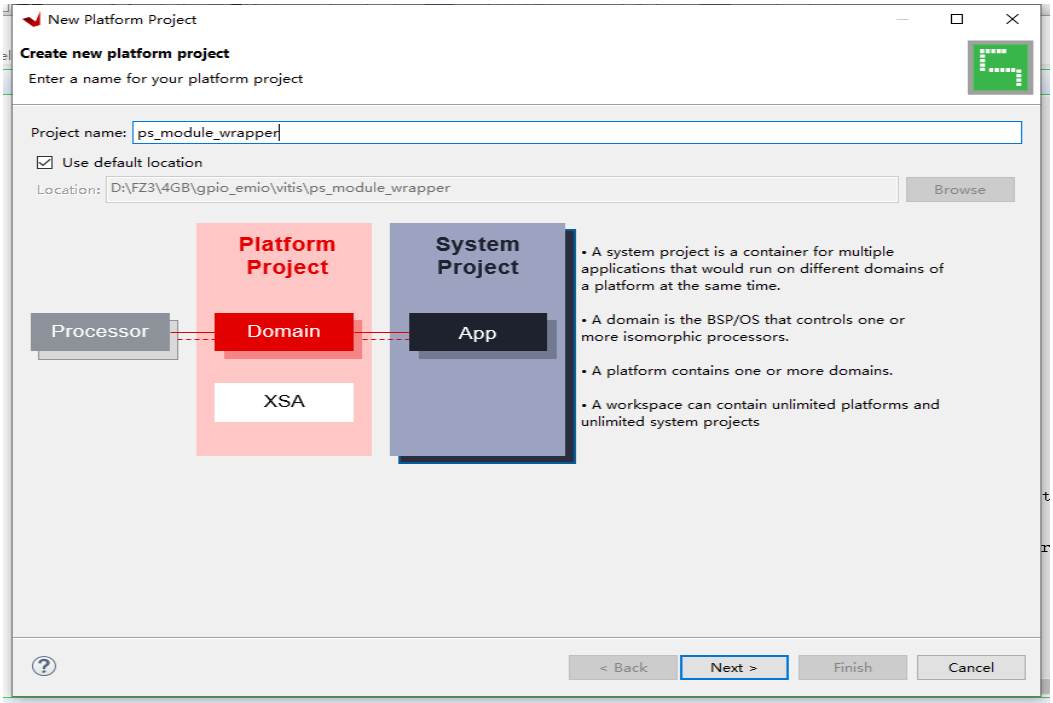
elect the path, select the vitis folder under axi_uart project. Click launch.



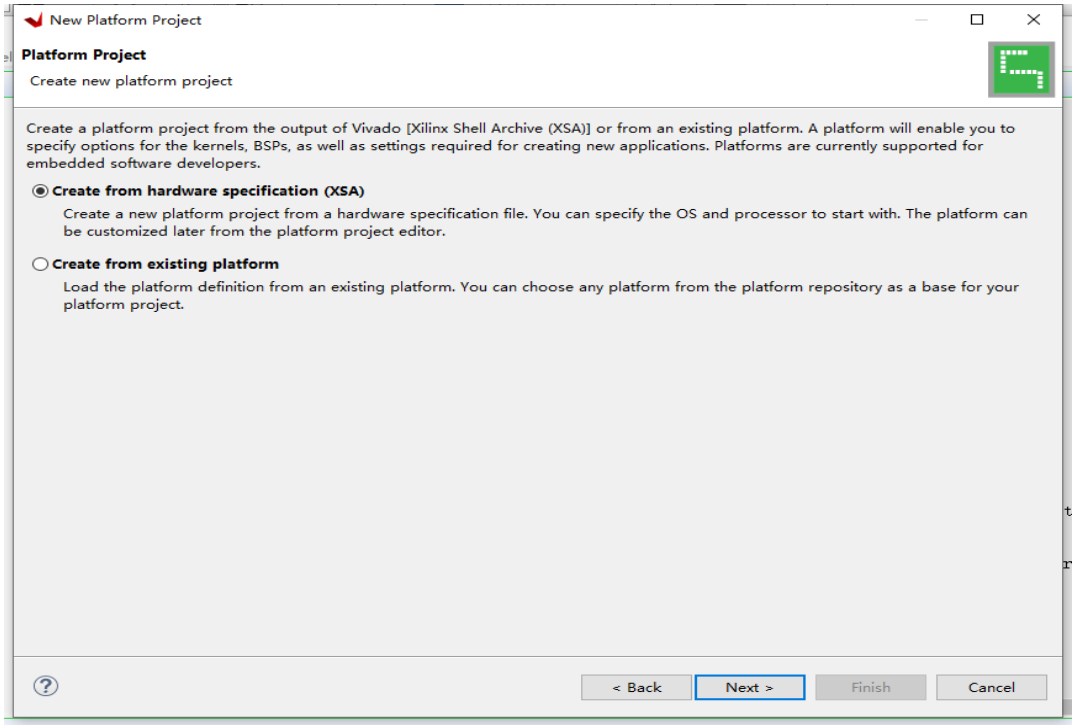
Click Create Platform Project



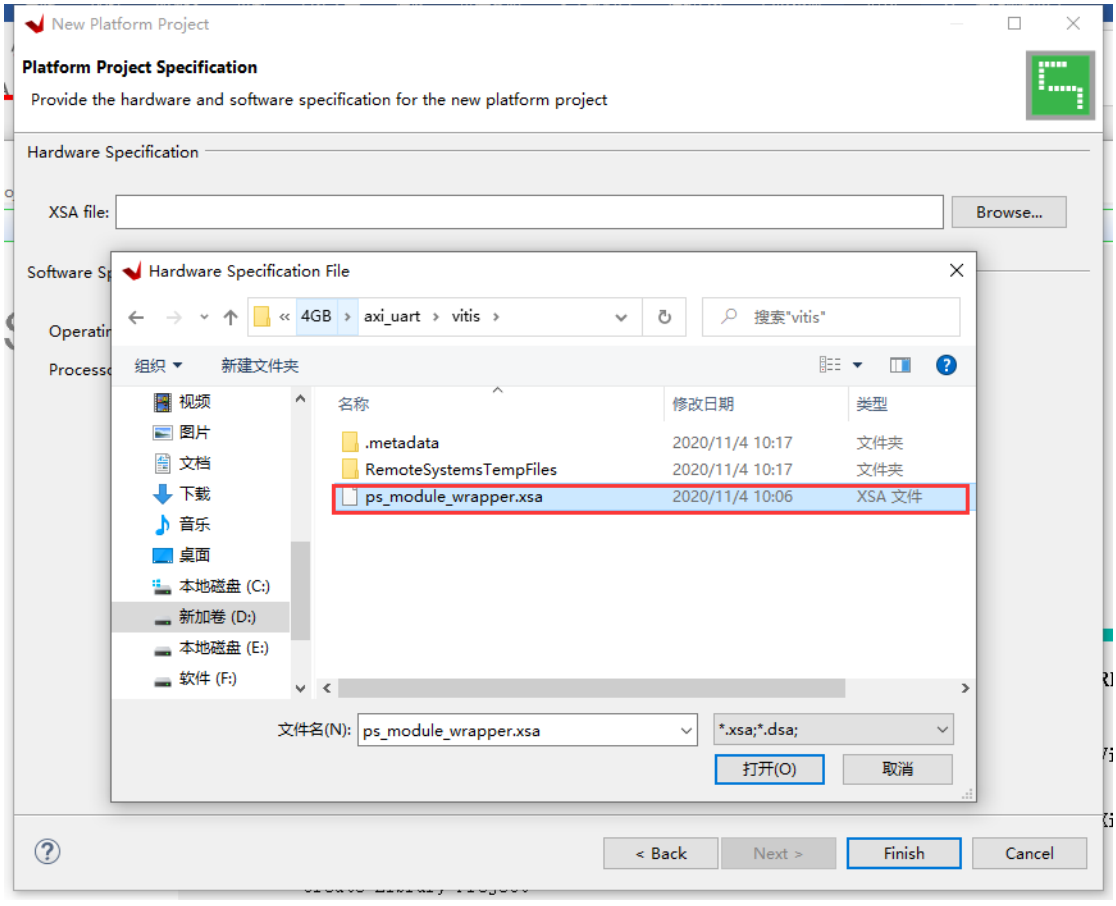
The project name uses the same name as the xsa file. Click next.



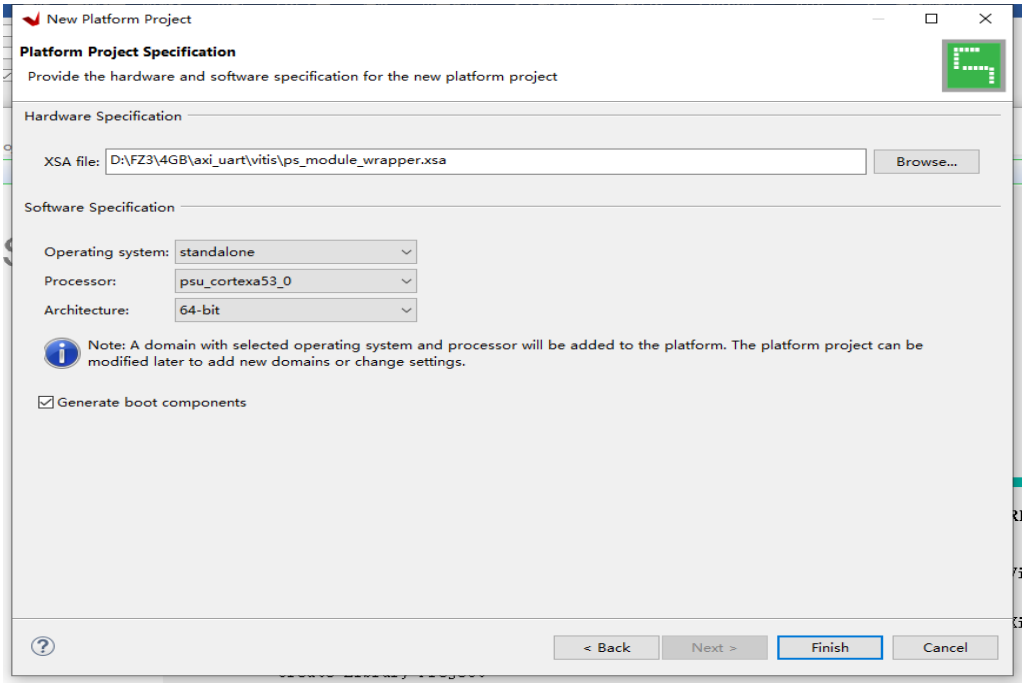
Select Create from hardware specification(XSA),click NEXT.



Open Browse… , select the previously generated xsa file.

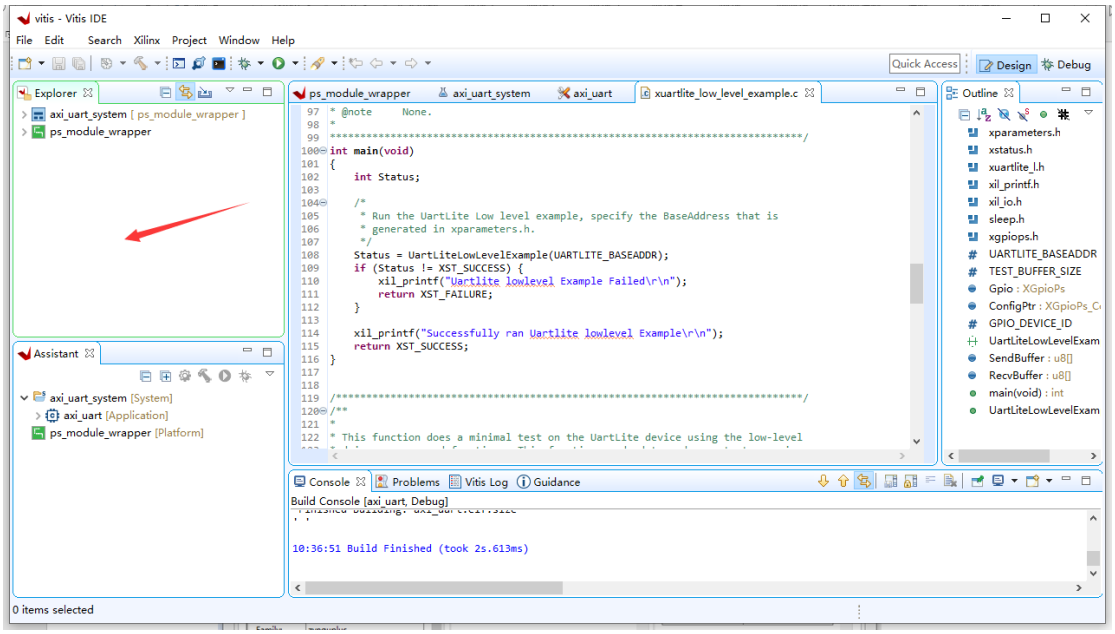


Leave the default and click finish.

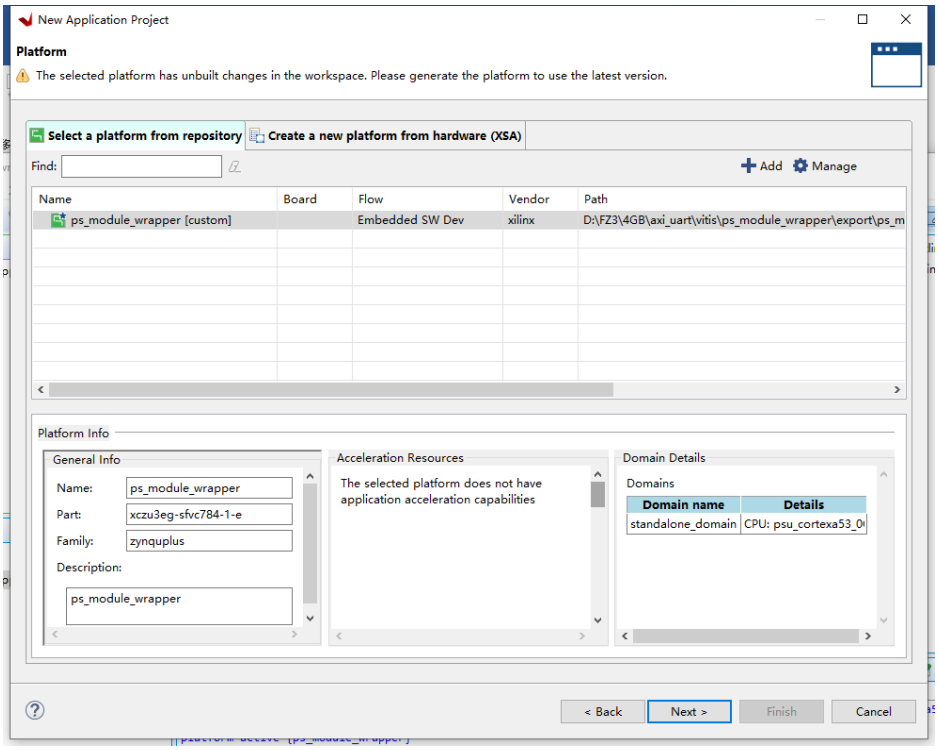


1.8 New axi_uart Project

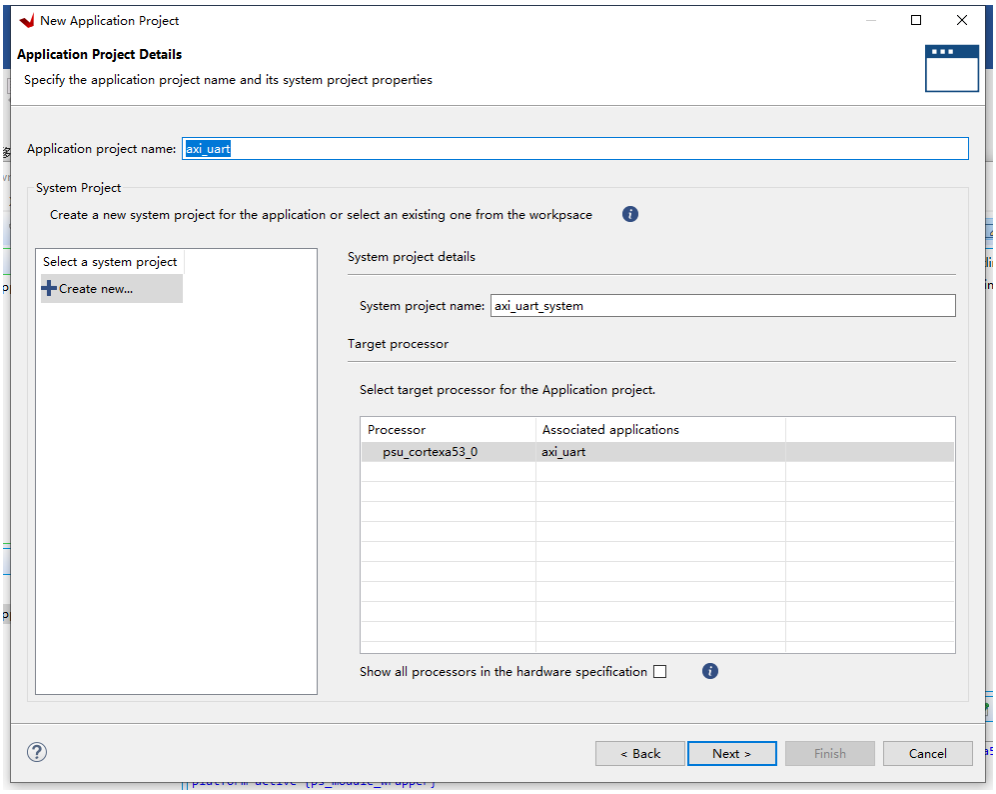
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



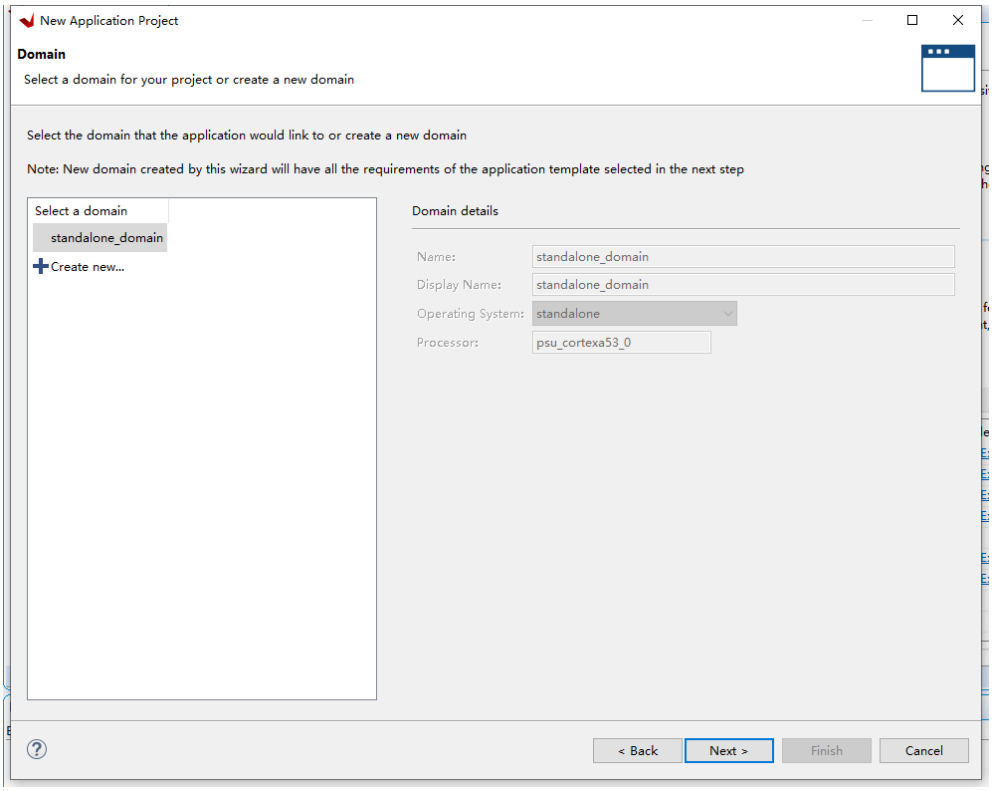
Skip to the second page and select the platform project created above (default), next



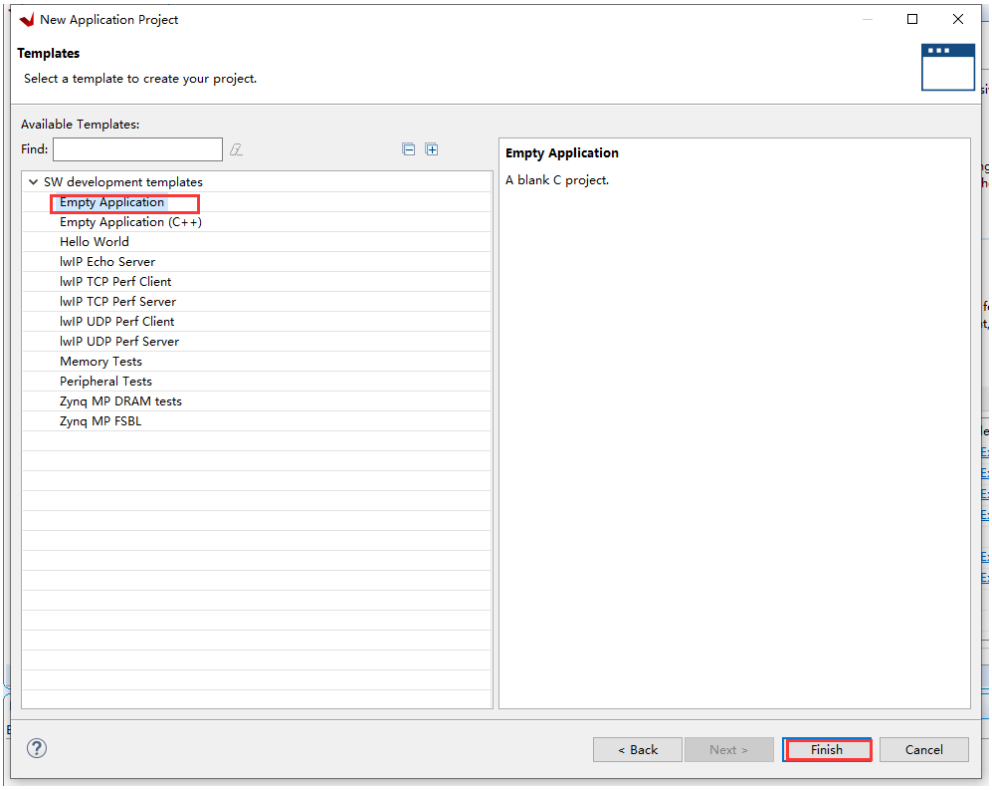
Project name enter axi_uart， Next



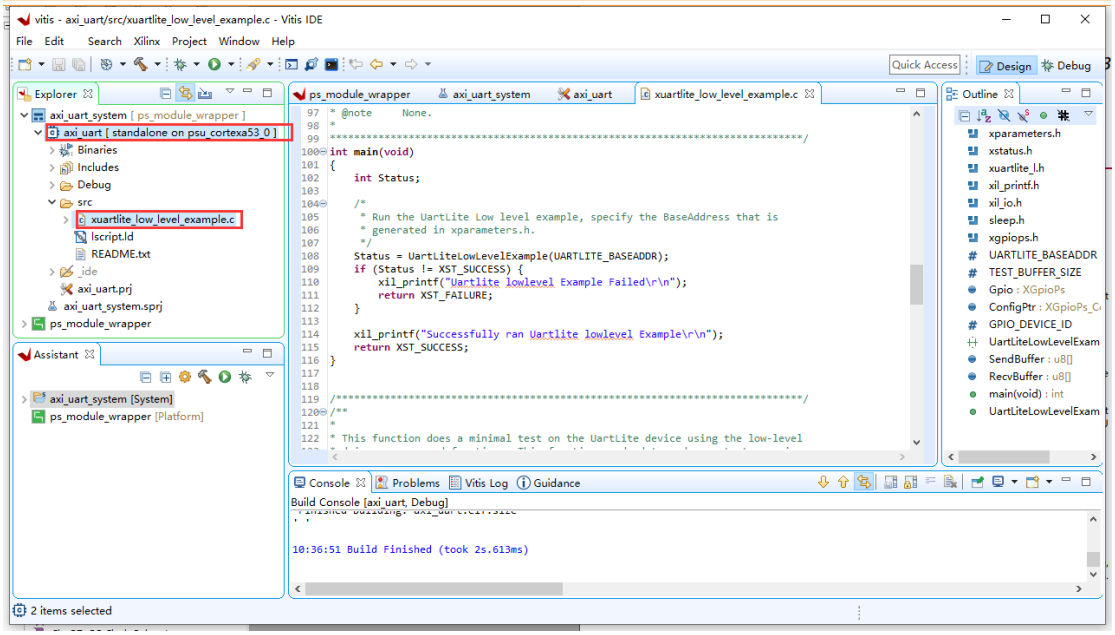
Next，



Select Empty Application， Finish

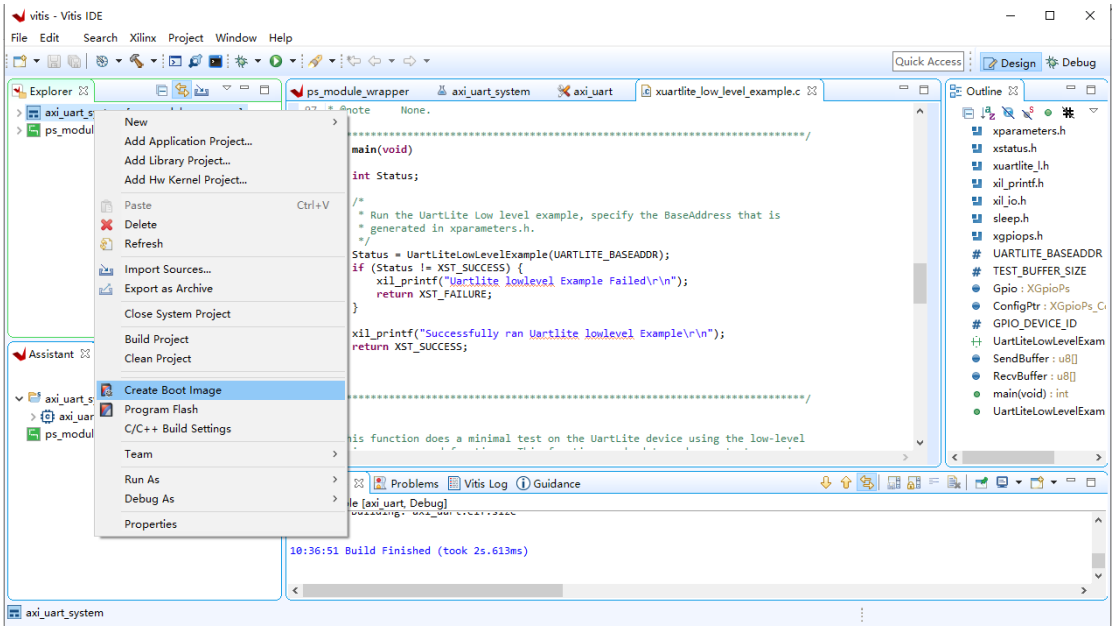


Copy the files in the example project to src, select the project, and click the compile buttom.

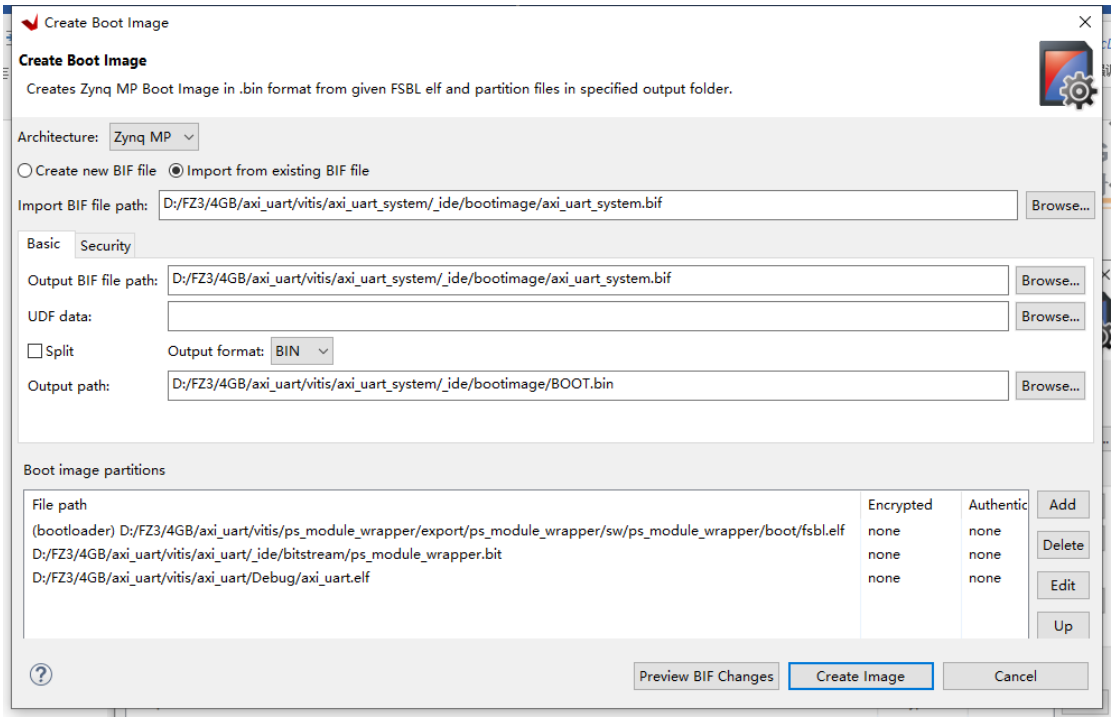


1.9 Generate BOOT.bin file

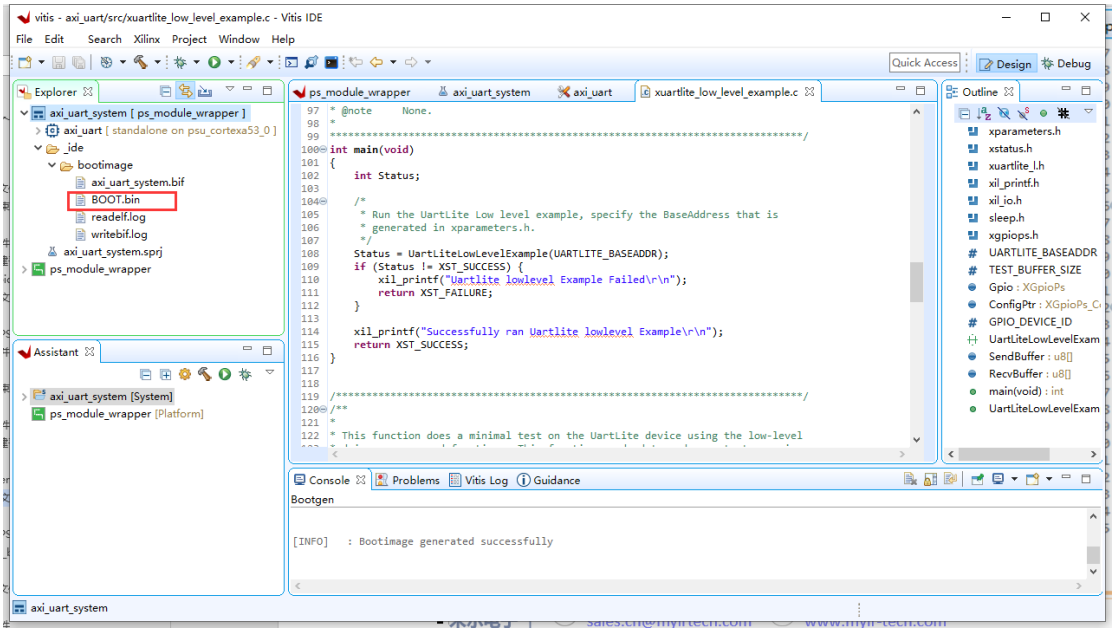
Right-click the system of APP project and select Create boot image.



Click Create Image to generate BOOT.bin Startup file.



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.



Chapter 5 bram_test

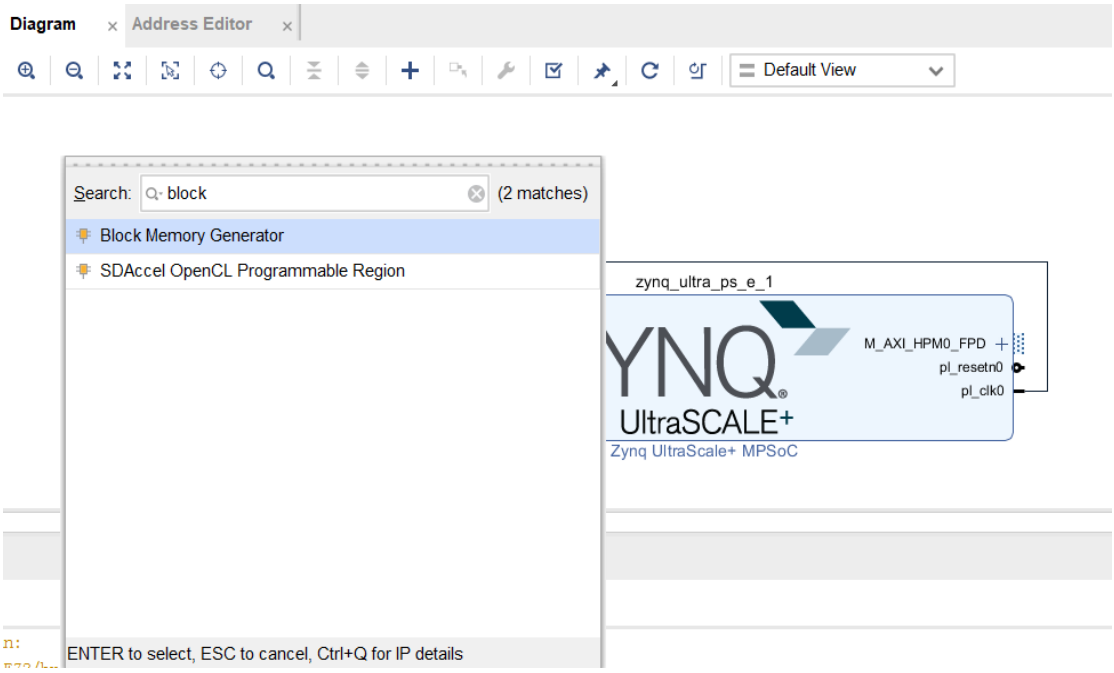
The purpose of this article is to use Block Memory for data exchange or data sharing between PS and PL, write data to BRAM through Master GP0 port of Zynq PS, read data through Mater GP1 of PS, print and output the results to serial terminal display.

1.1 Create project and configure IP core of PS

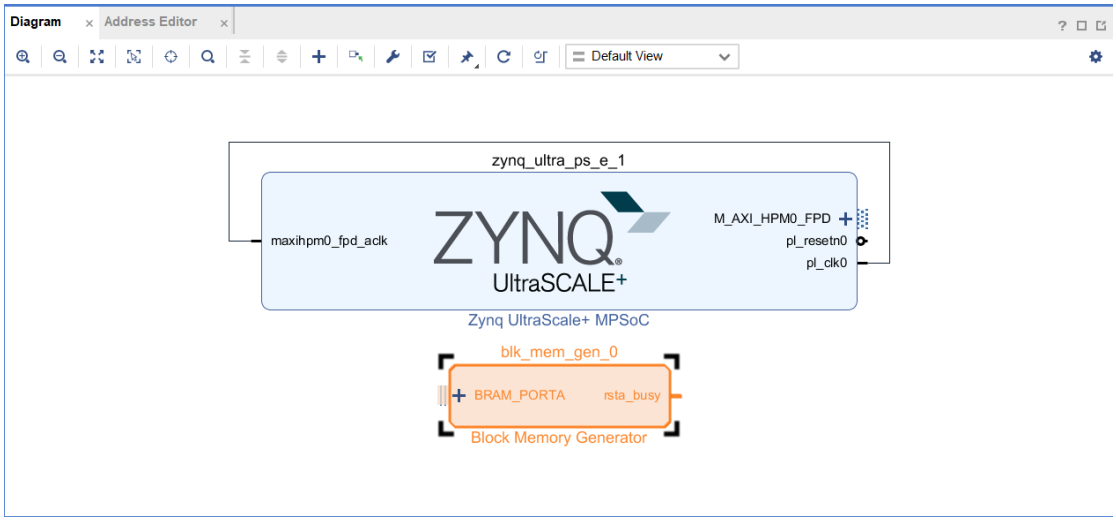
Based on "ps_hello" project, save as bram_test project.

1.2 Add Bram and axi_bram_controller cores and configure them

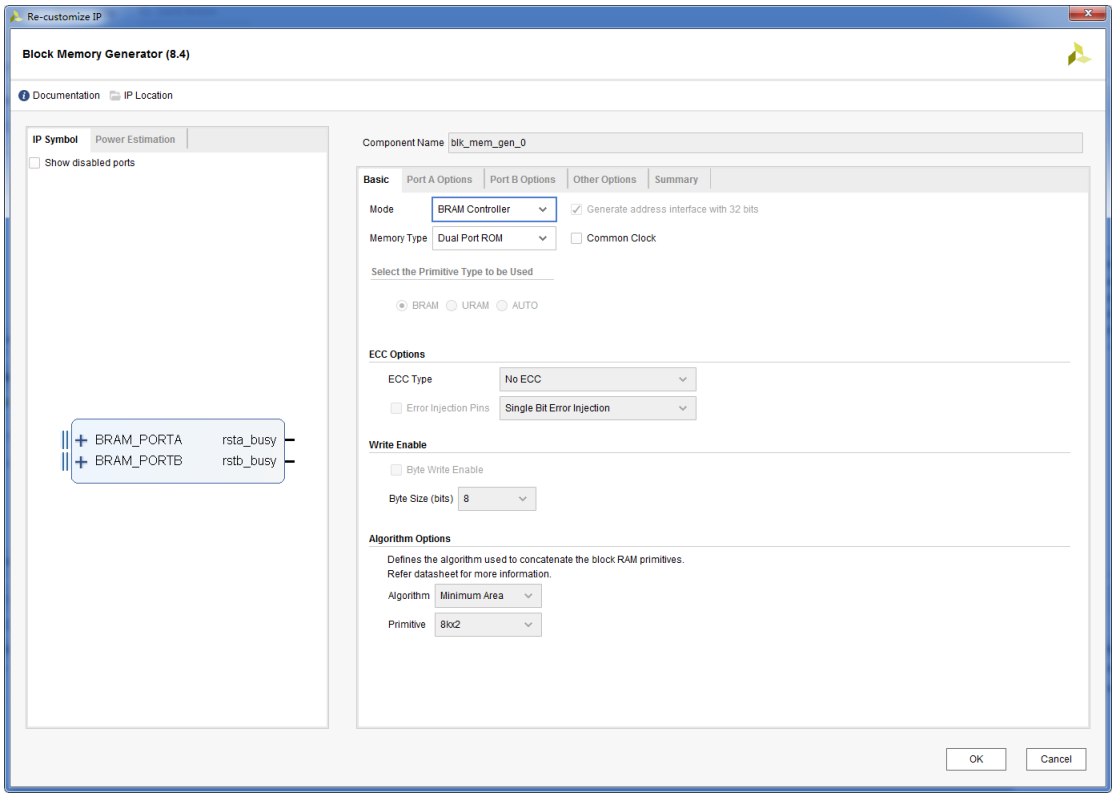
Enter Bram in the search bar and double-click Block Memory Generator to add the Bram core



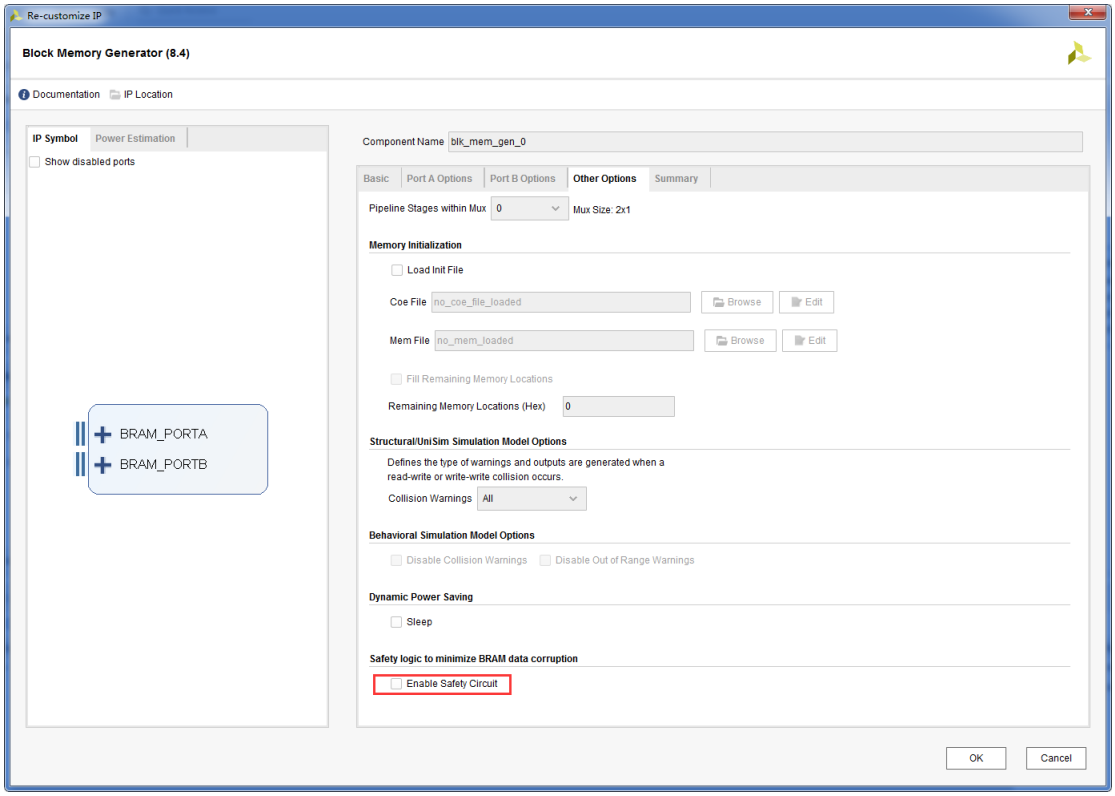
The Bram core is added, as shown in the following figure



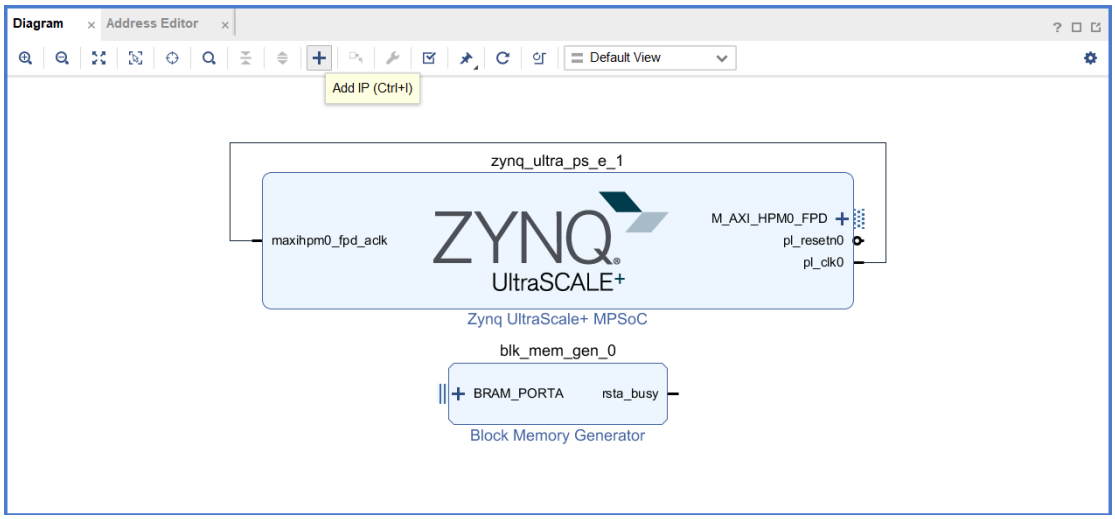
Double-click the Bram core to set Memory Type to Dual Port RAM



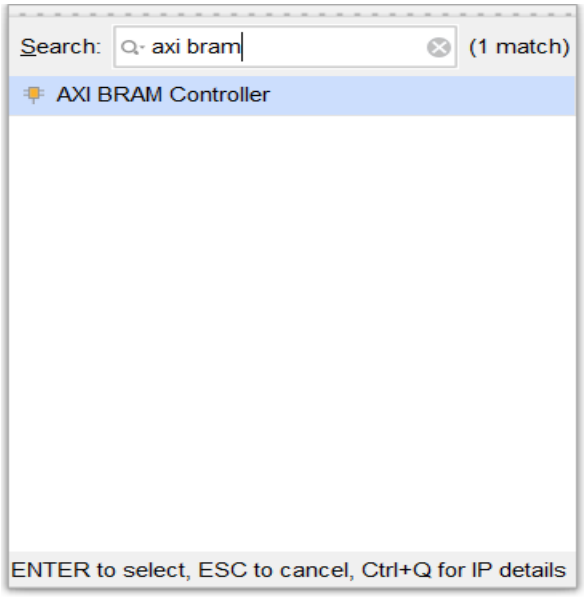
Cancel the Check of Enable Safety Circuit



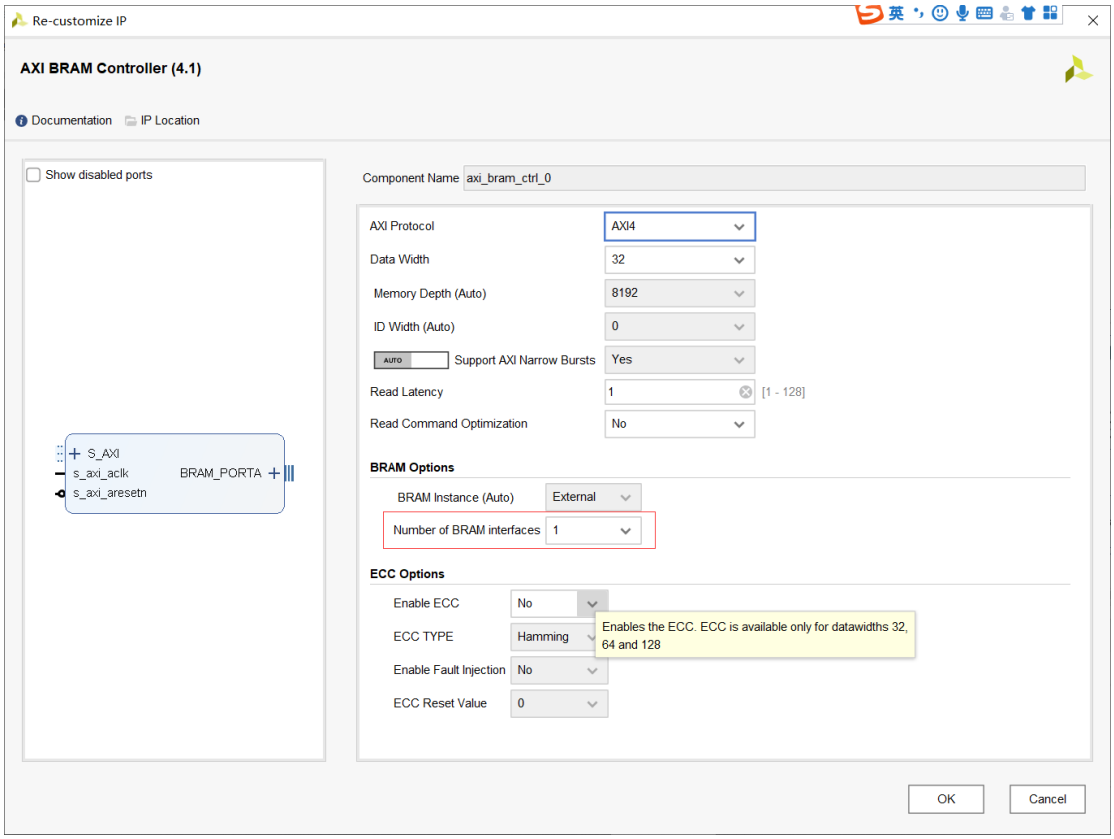
Click Add IP



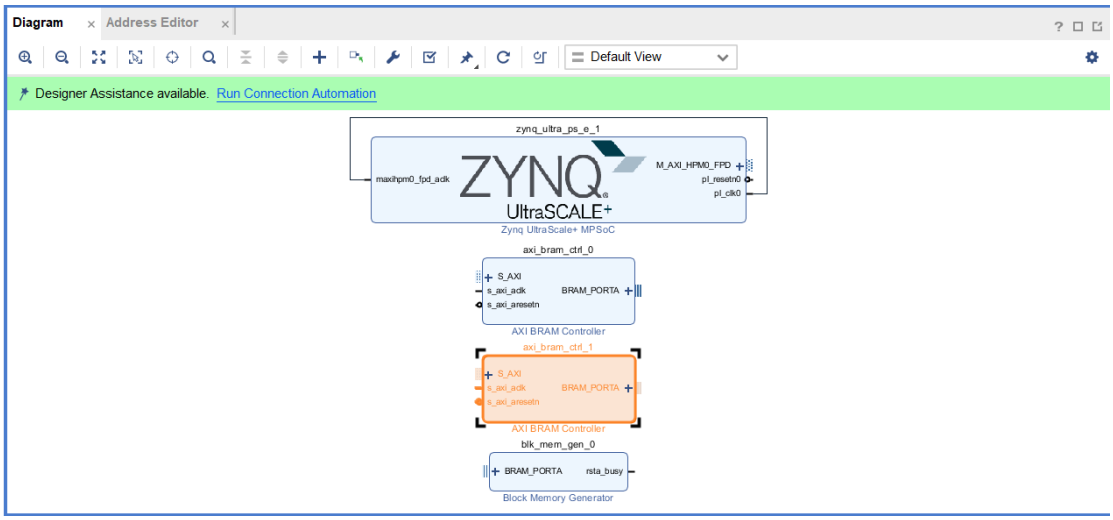
Enter axi_bram in the search bar and double-click AXI BRAM Controller to add the axi_bram_controller core



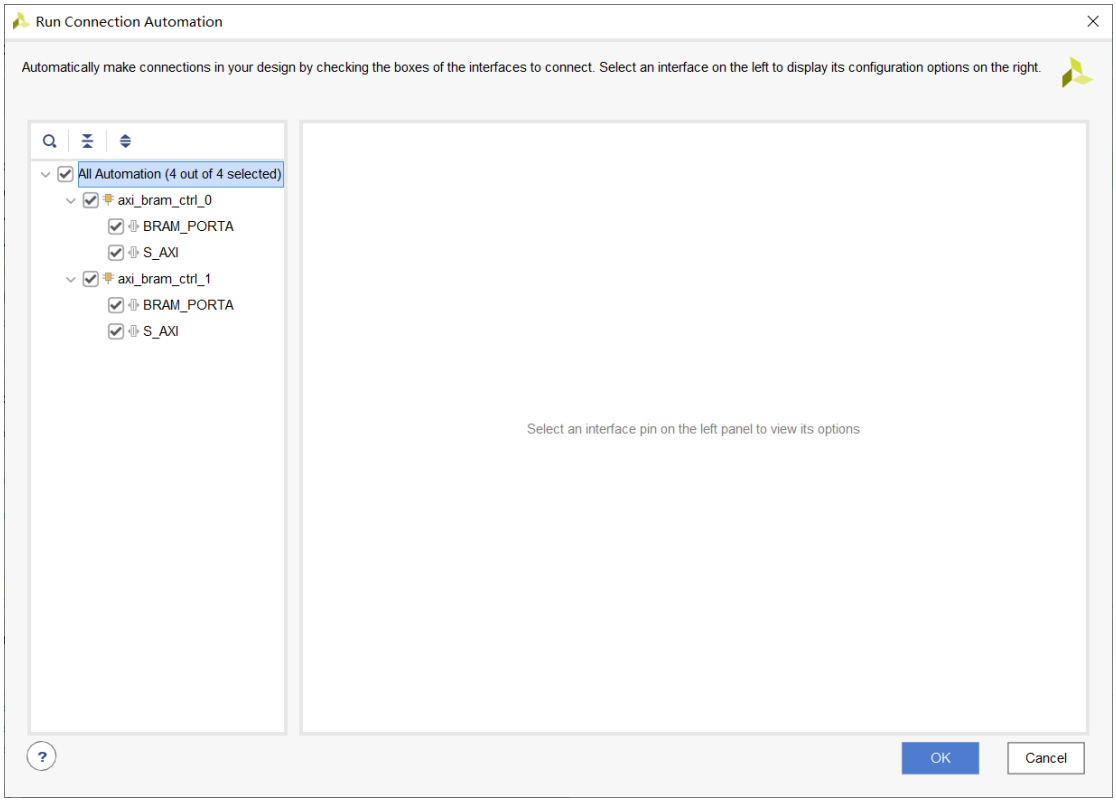
Double-click on axi_bram_controller to set parameters, and set Number of BRAM interfaces to 1, as does another axi_bram_controller.



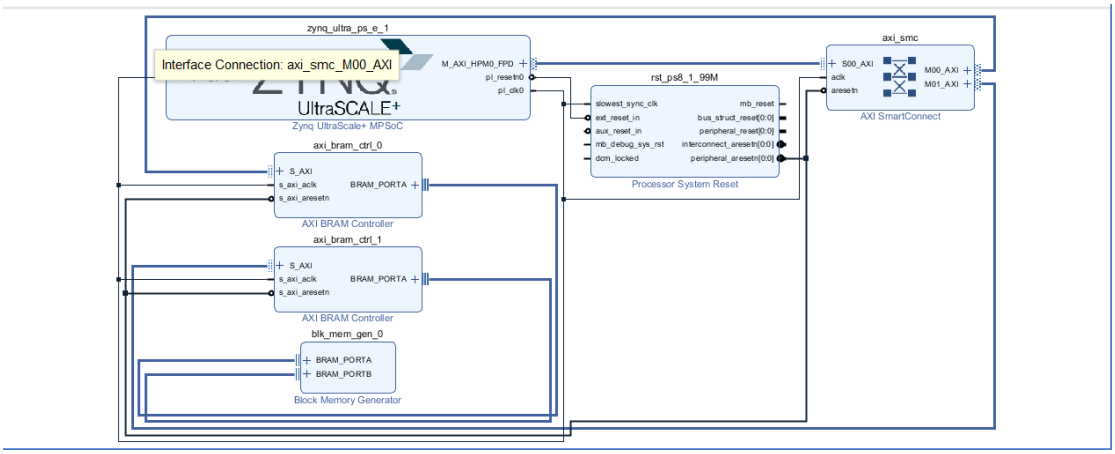
Click Run Block Automation - > OK to connect automatically



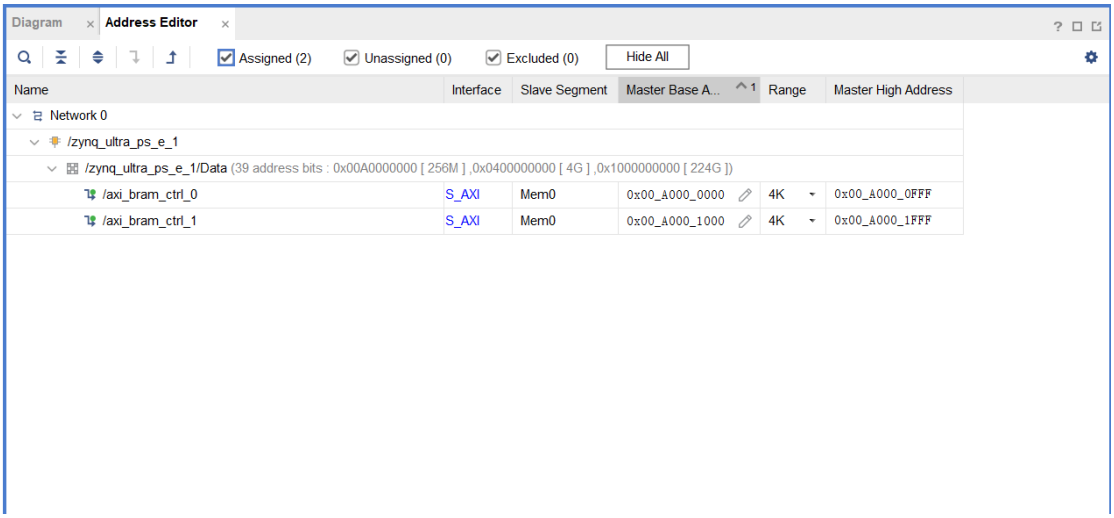
Check all the options and click OK



After the automatic connection is completed, the following figure is shown.

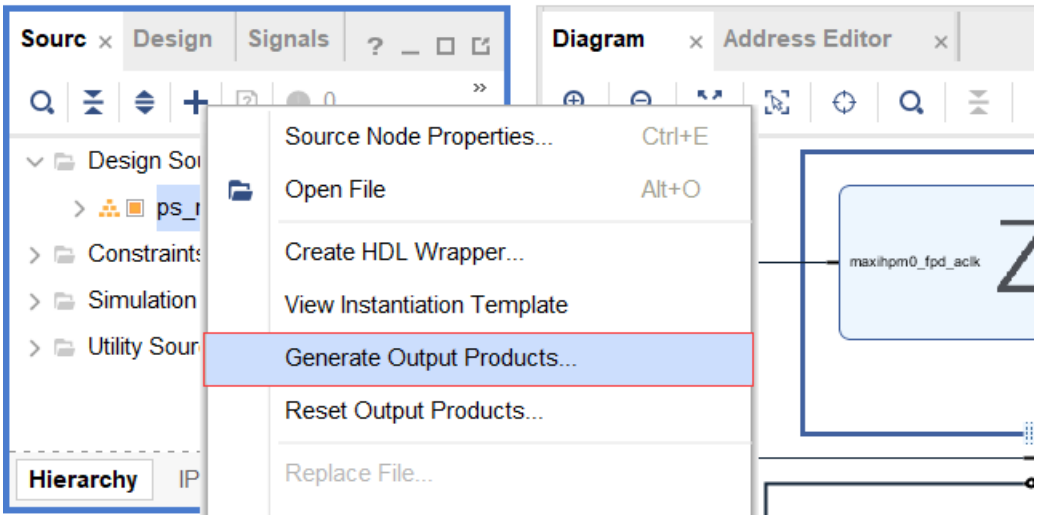


Setting Address

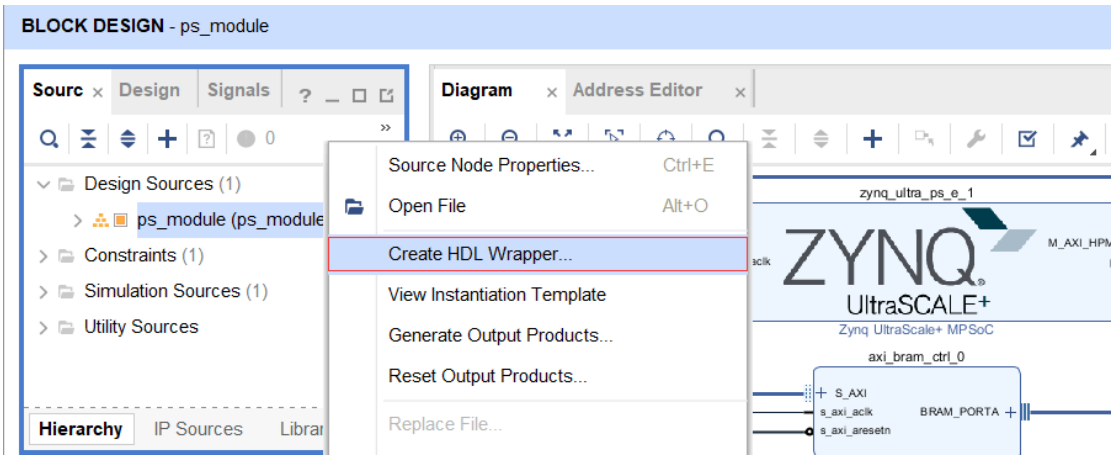


1.3 Generate synthesis files

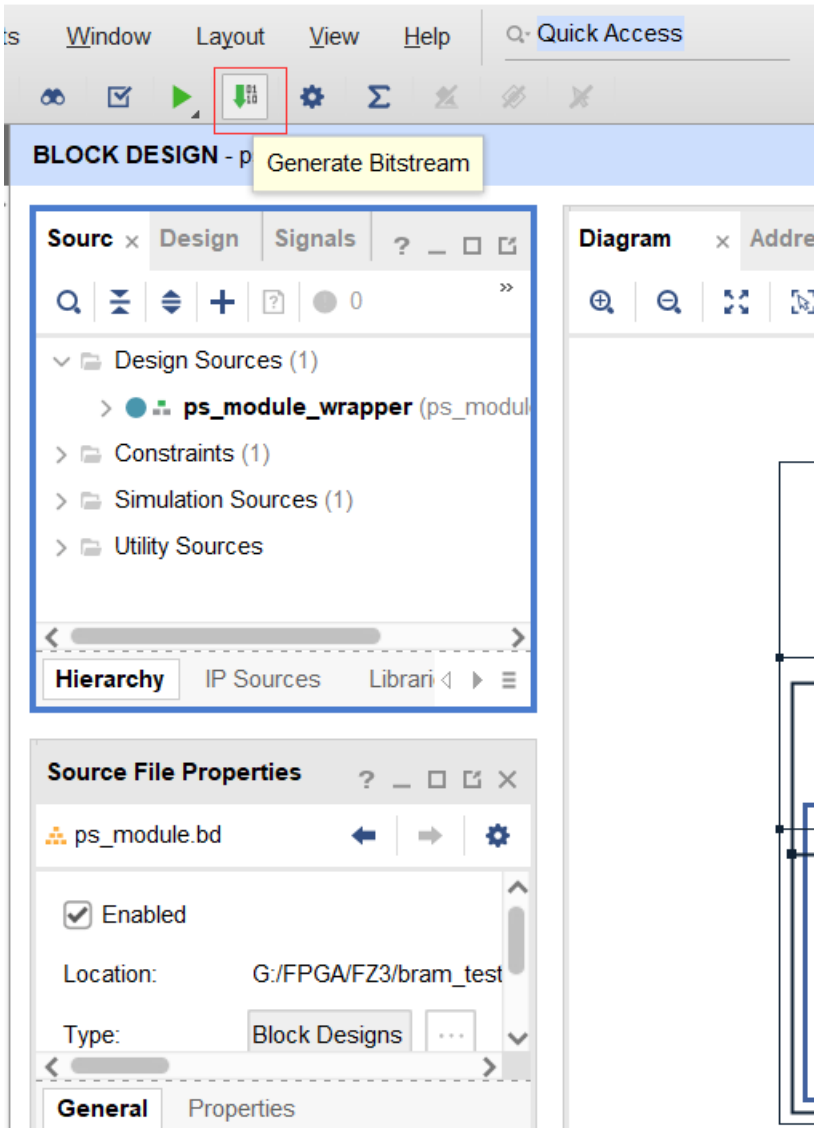
Right-click ps_module->Generate Ouput Products->Generate



1.4 Generating top-level file of FPGA

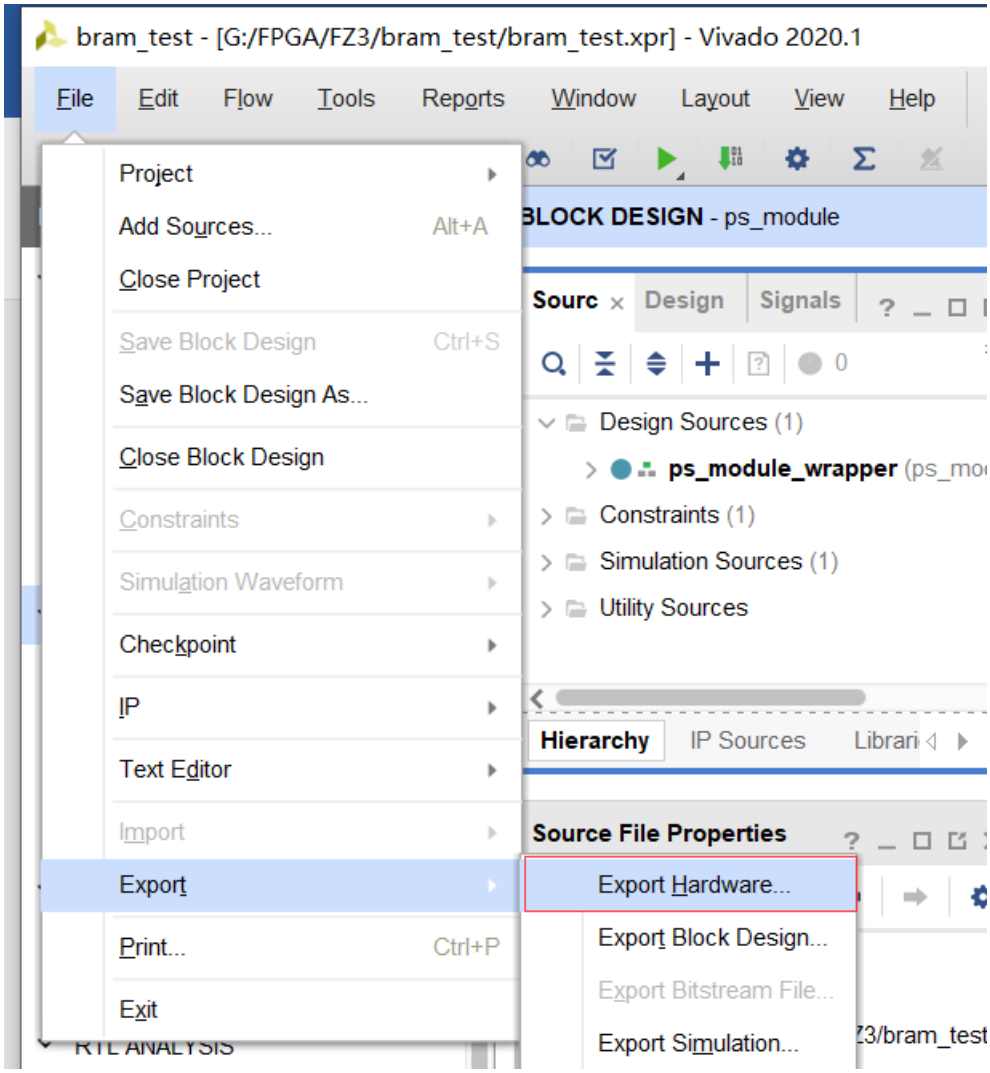


1.5 Generate bit files



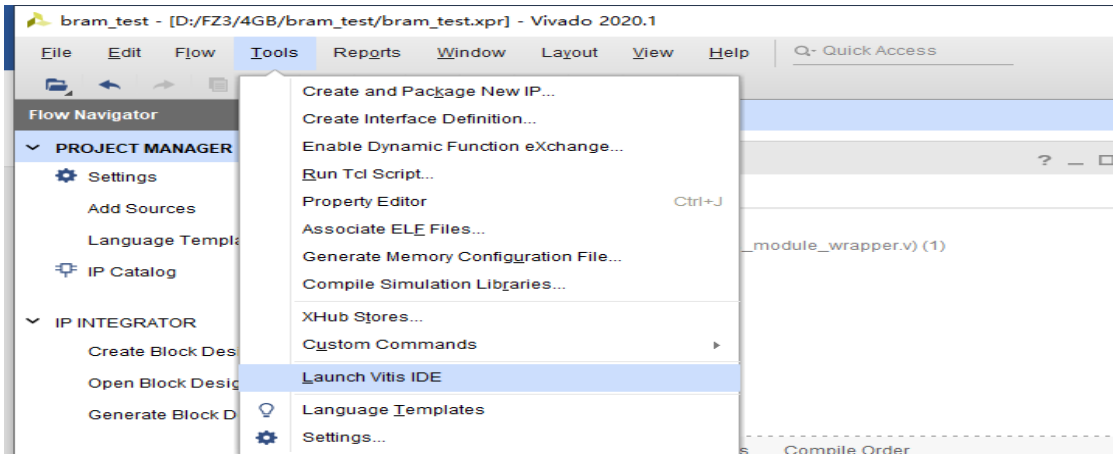
1.6 Export Hardware Profile

Click File - > Export - > Export Hardware - > OK on the menu bar to export the hardware configuration file

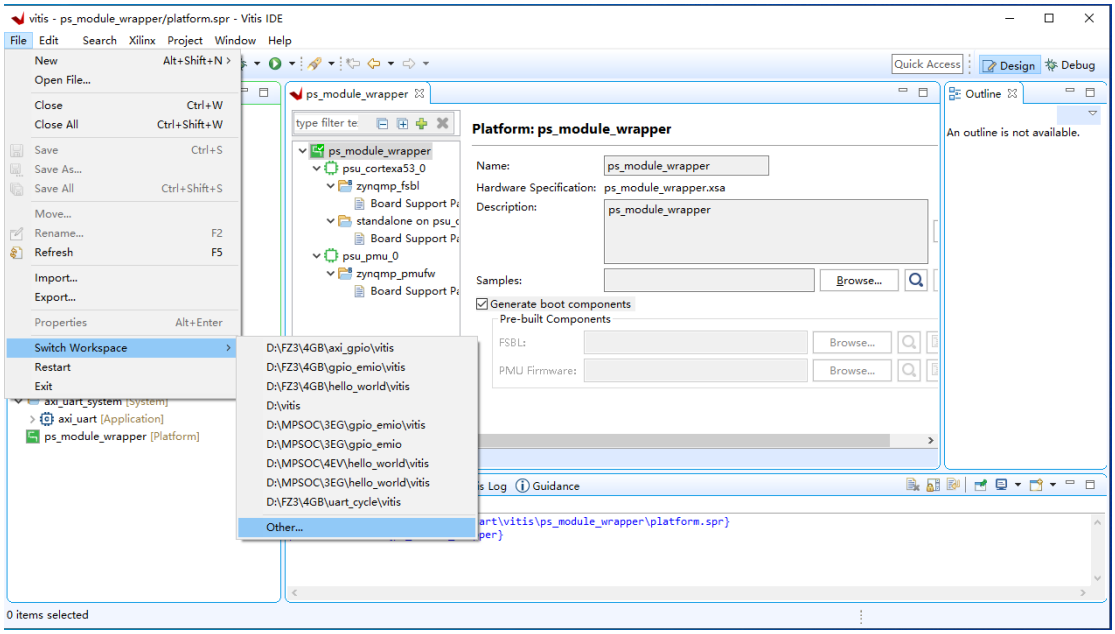


1.7 Launch Vitis and create new platform project

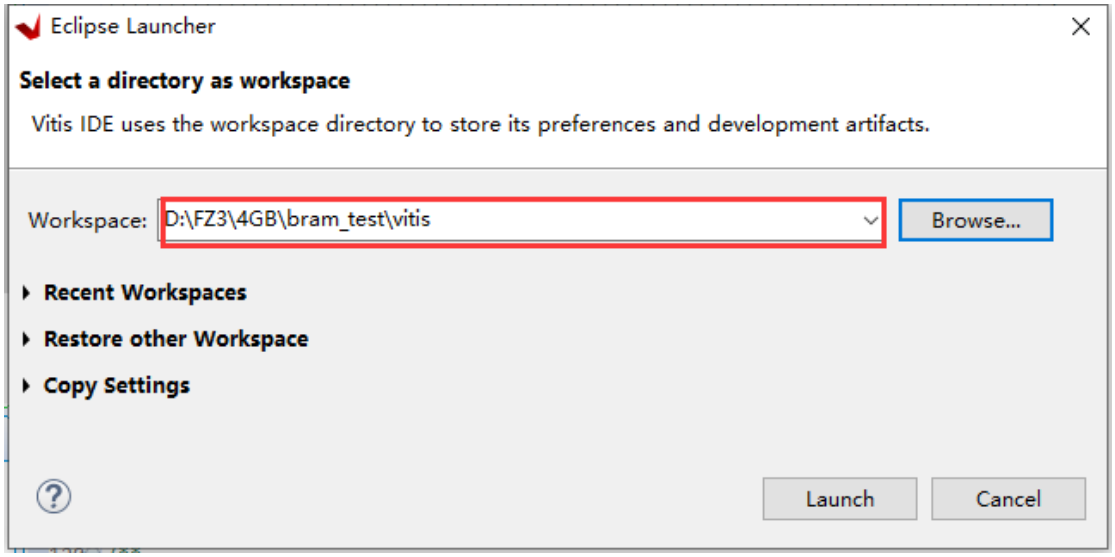
Click Tools > launch Vitis ide o n the menu bar to launch Vitis



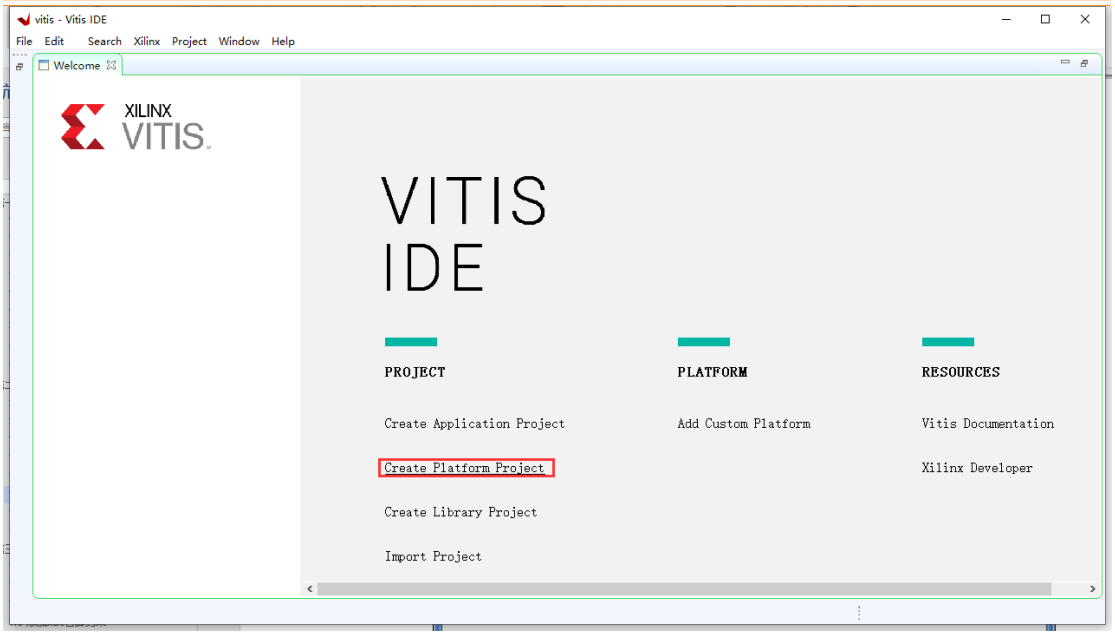
Click File→Switch Workspace→Other…,



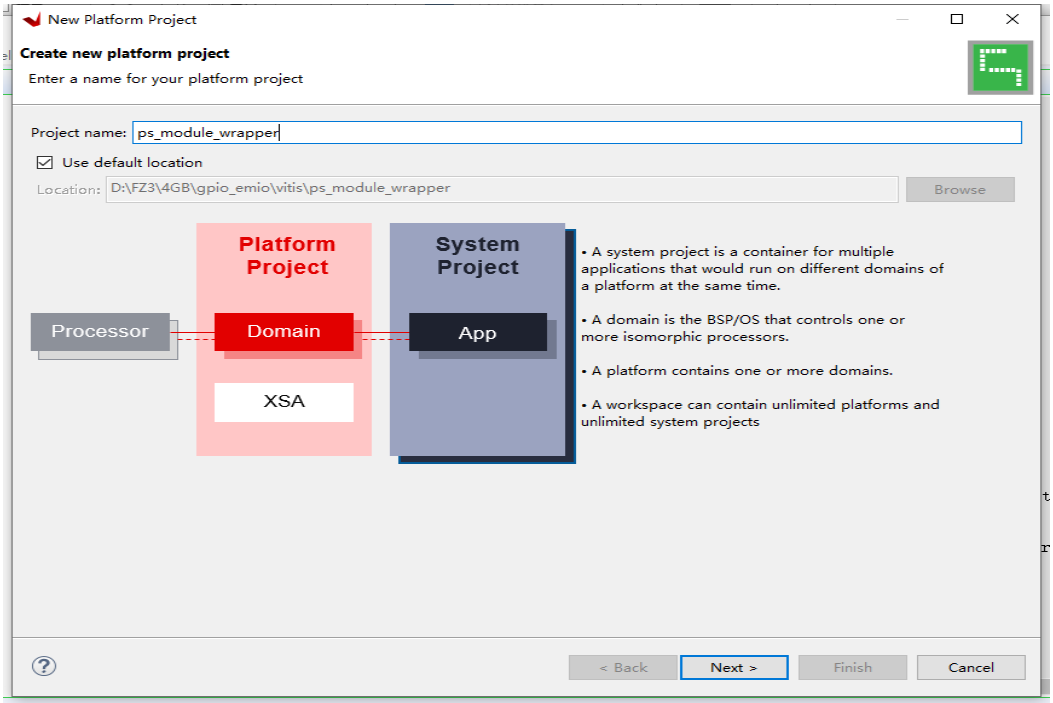
elect the path, select the vitis folder under bram_test project. Click launch.



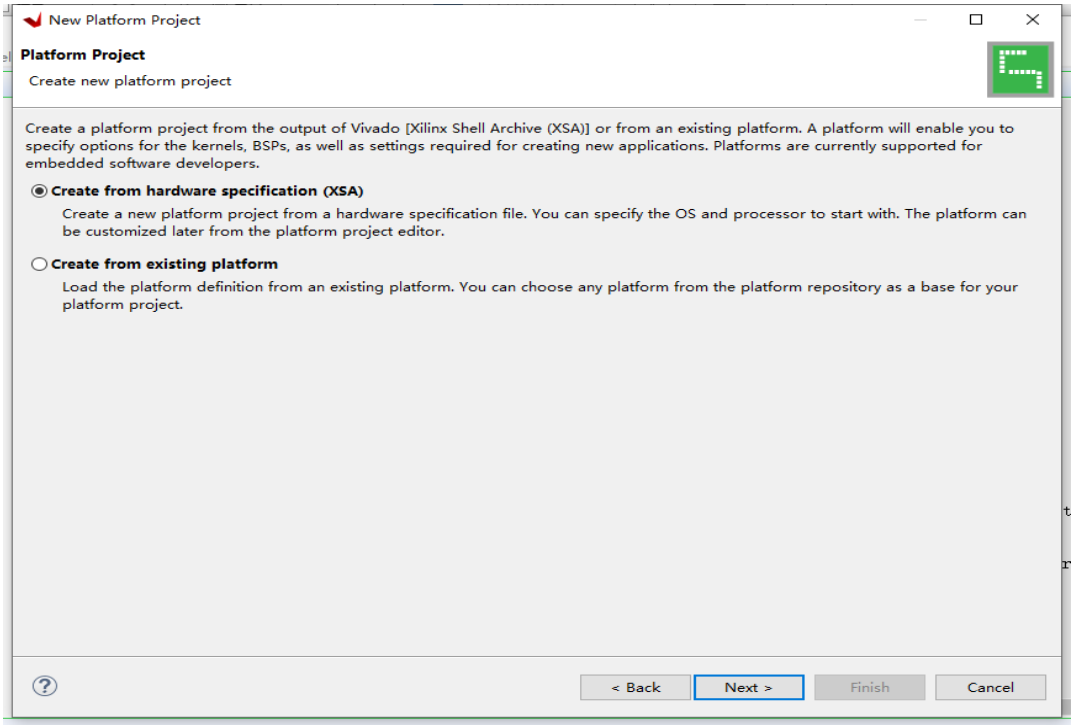
Click Create Platform Project



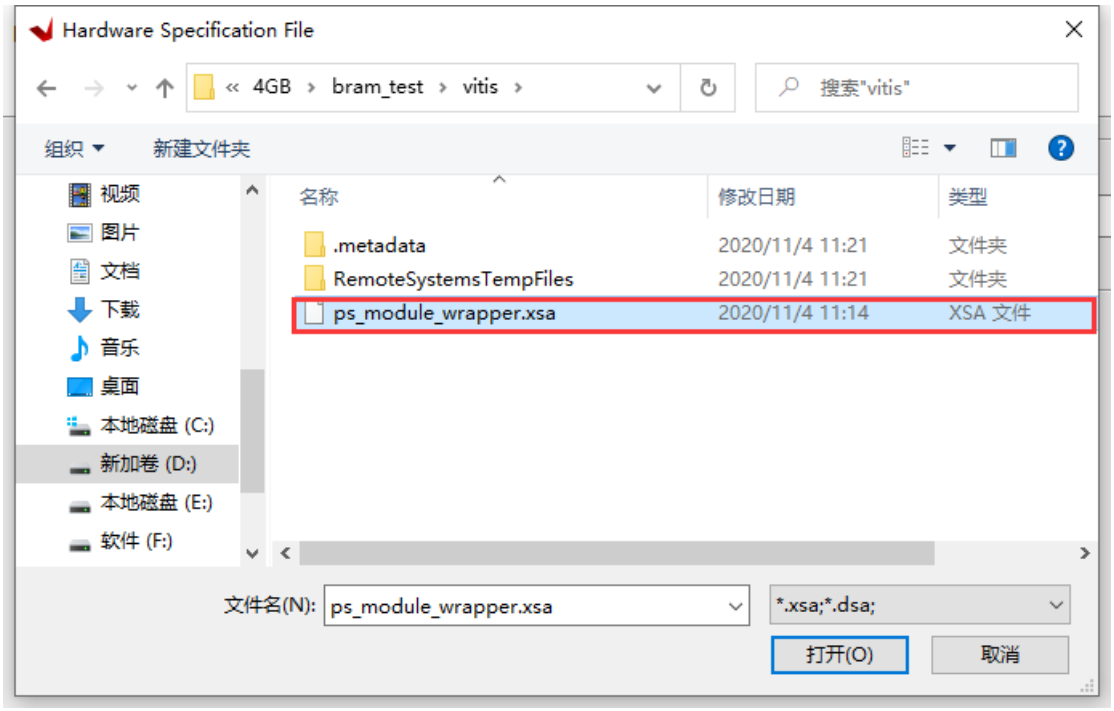
The project name uses the same name as the xsa file. Click next.



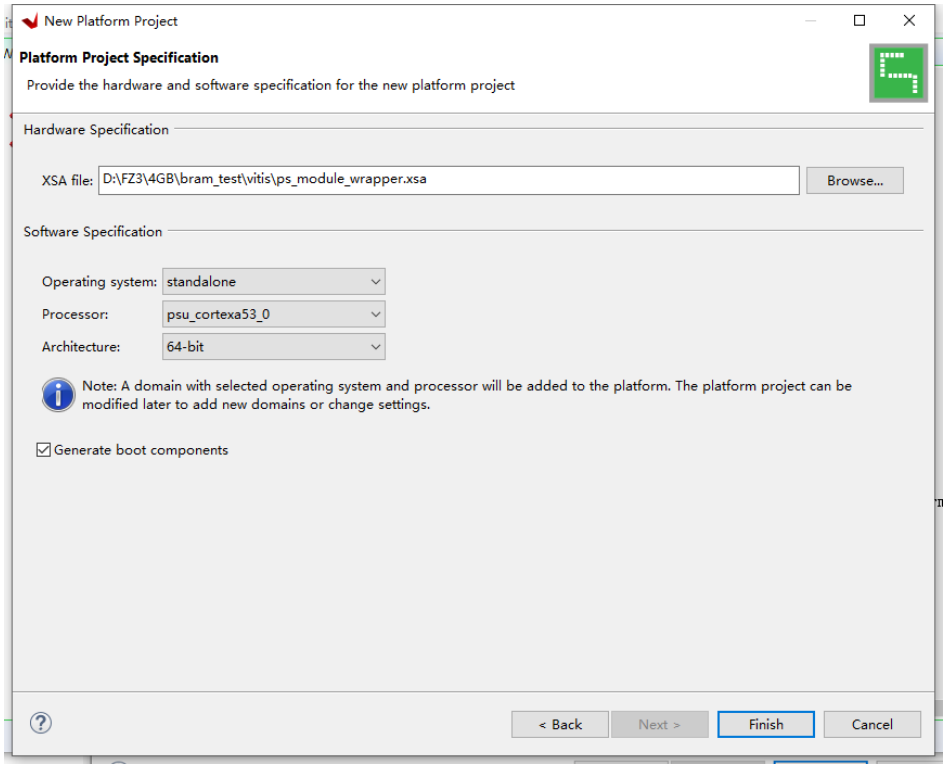
Select Create from hardware specification(XSA),click NEXT.



Open Browse… , select the previously generated xsa file.

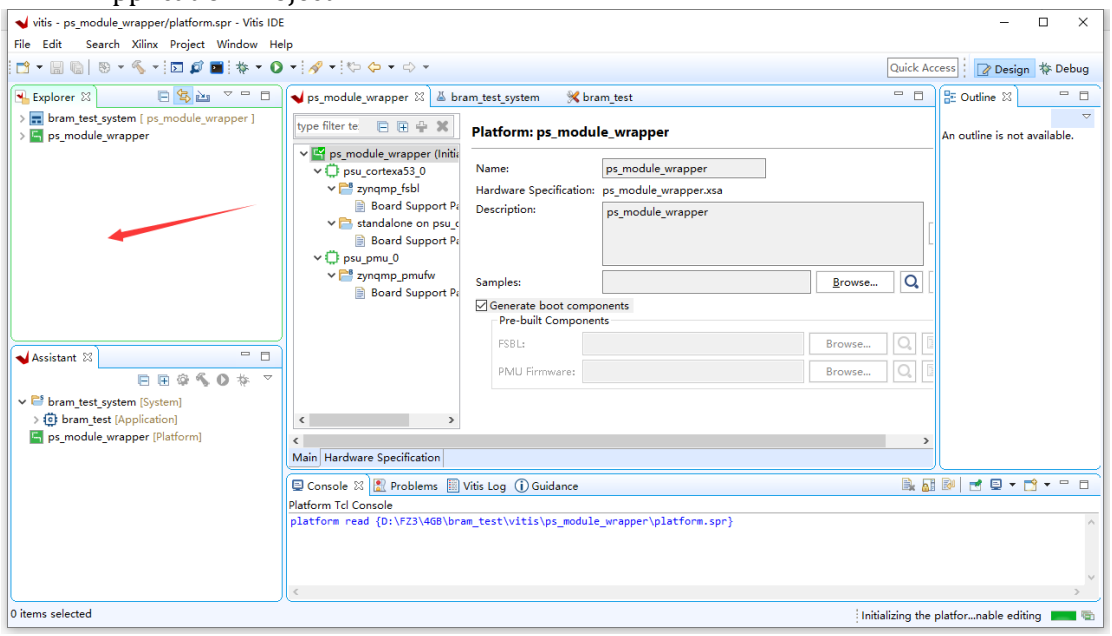


Leave the default and click finish.

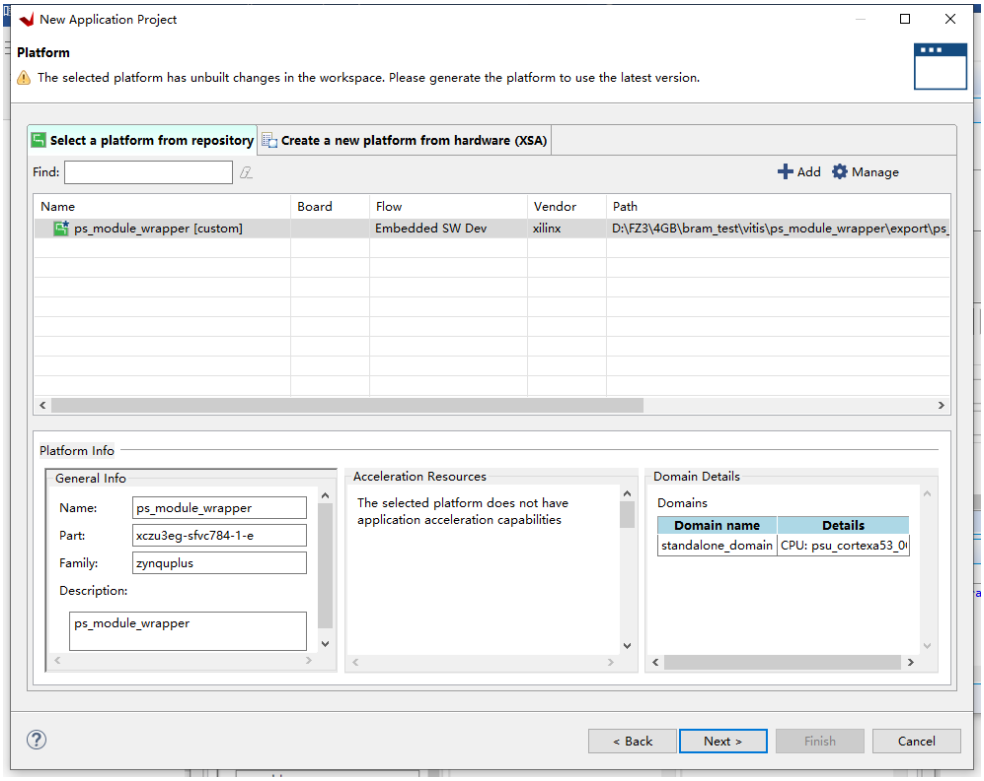


1.8 New bram_test Project

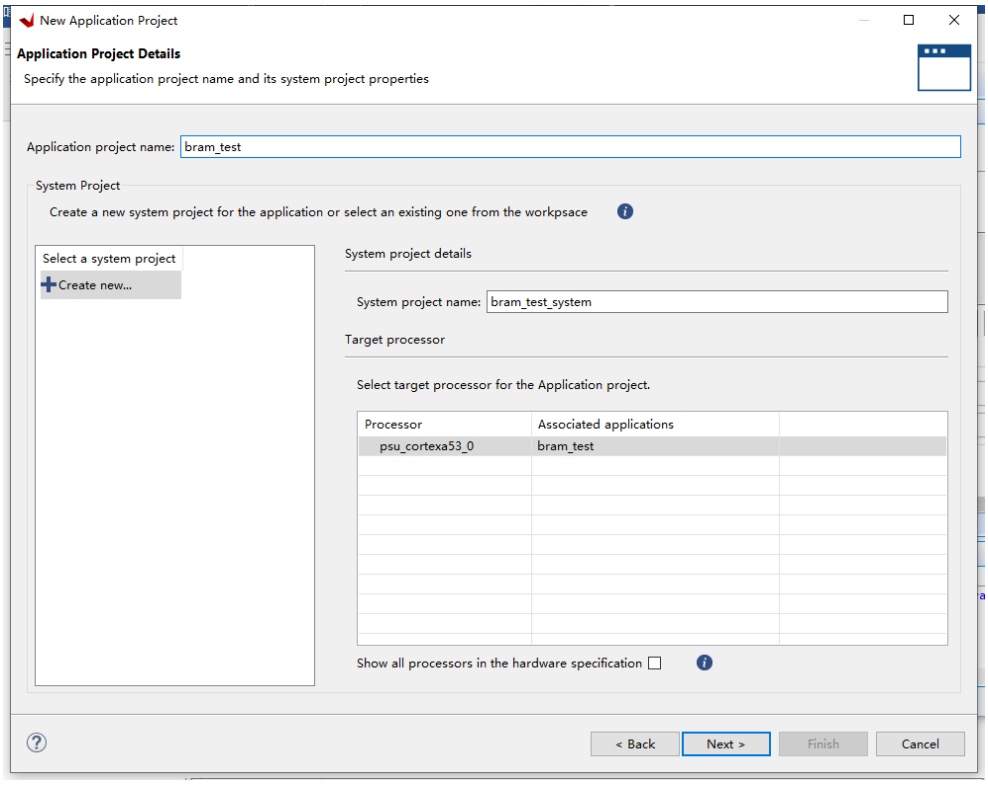
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



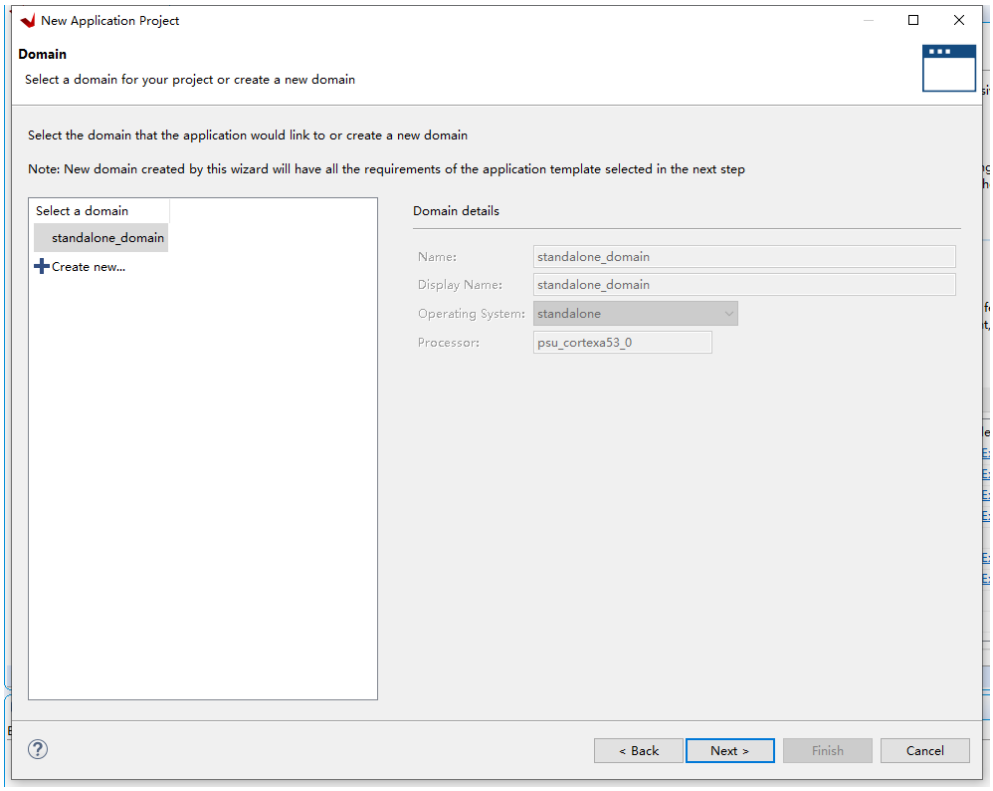
Skip to the second page and select the platform project created above (default), next



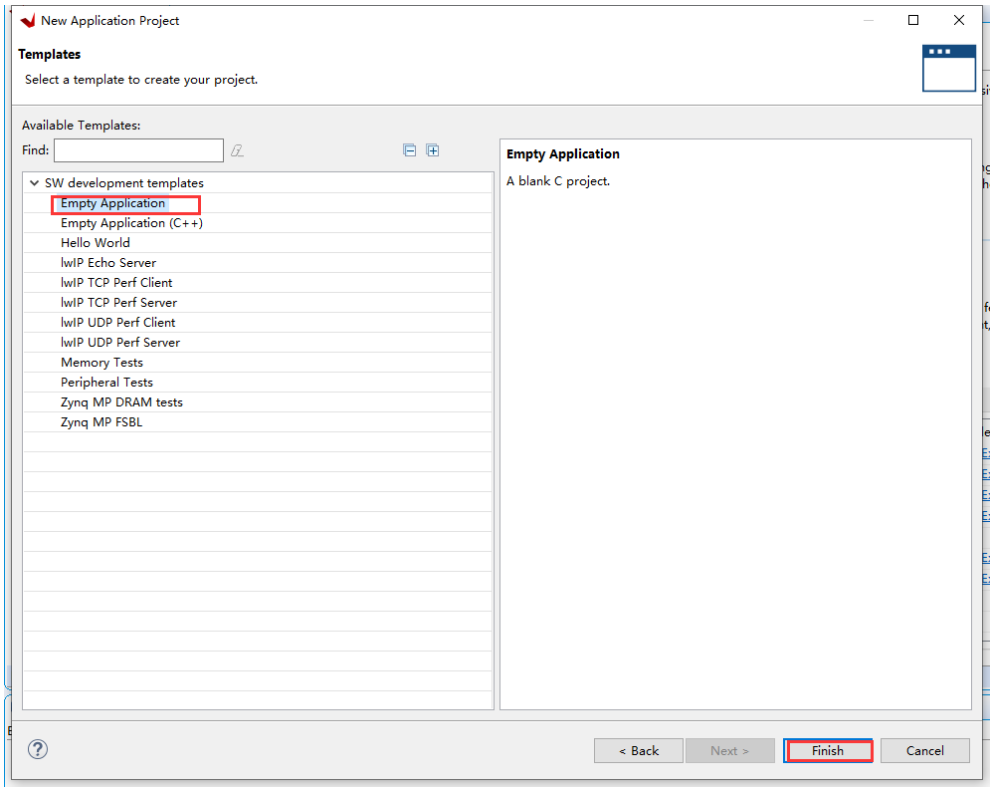
Project name enter bram_test， Next



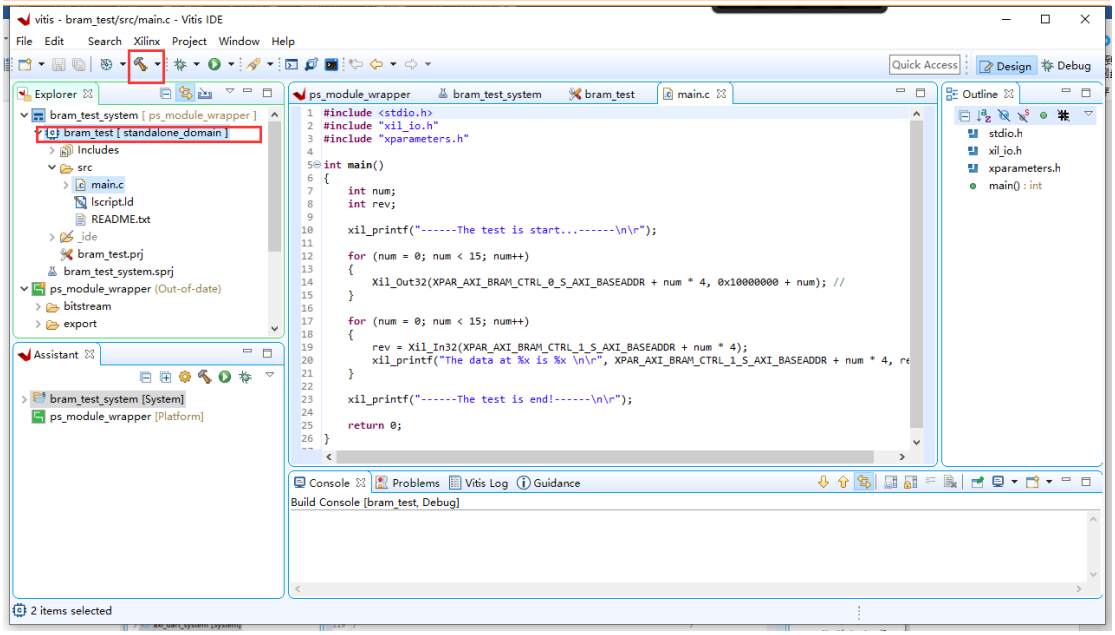
Next



Select Empty Application， Finish

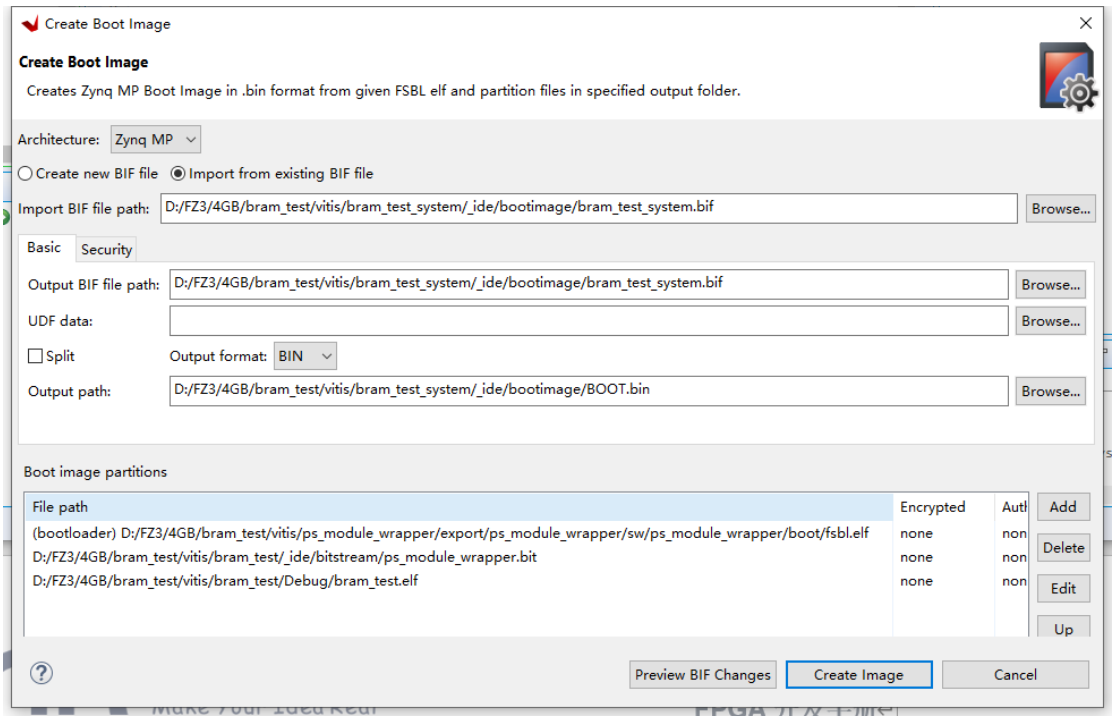


Copy the files in the example project to src, select the project, and click the compile buttom.

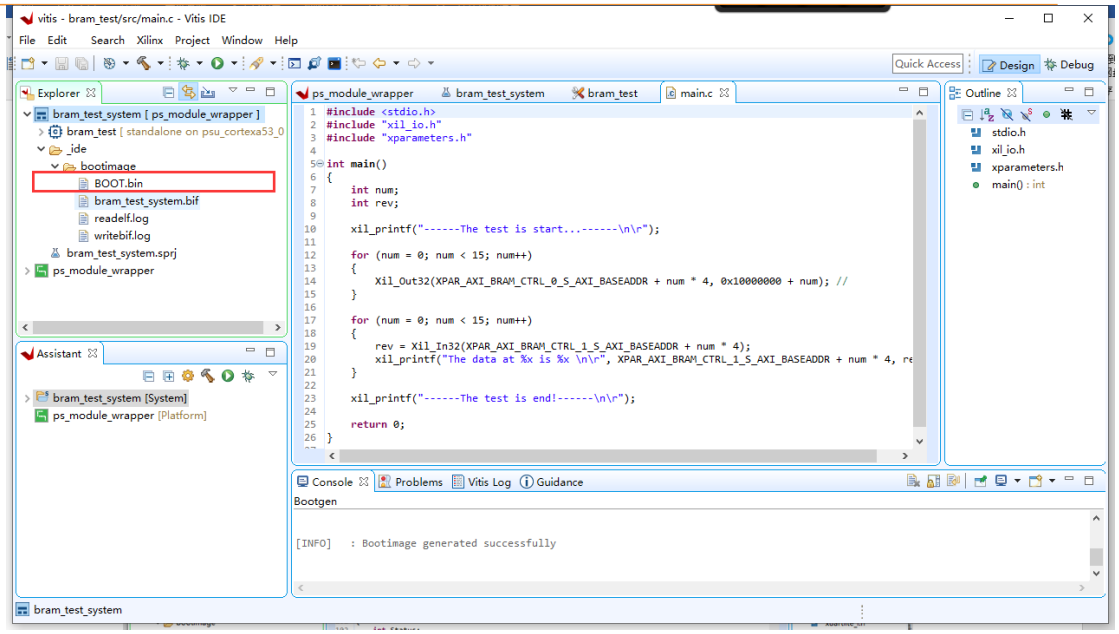


1.9 Generate BOOT.bin file

Right-click the system of APP project and select Create boot image.



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.


vitis - bram_test/src/main.c - Vitis IDE
File Edit Search Xilinx Project Window Help
Quick Access
Design
Debug
Explorer
bram_test_system [ps_module_wrapper]
bram_test [standalone on psu_cortexa53_0
_ide
bootimage
BOOT.bin
bram_test_system.bif
readelf.log
writebif.log
bram_test_system.sprj
ps_module_wrapper
Assistant
bram_test_system [System]
ps_module_wrapper [Platform]
ps_module_wrapper
bram_test_system
bram_test
main.c
#include <stdio.h>
#include "xil_io.h"
#include "xparameters.h"
int main()
{
int num;
int rev;
xil_printf("------The test is start...------\n\r");
for (num = 0; num < 15; num++)
{
Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + num * 4, 0x10000000 + num); //
}
for (num = 0; num < 15; num++)
{
rev = Xil_In32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + num * 4);
xil_printf("The data at %x is %x \n\r", XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + num * 4, re
}
xil_printf("------The test is end!------\n\r");
return 0;
}
Outline
stdio.h
xil_io.h
xparameters.h
main() : int
Console
Problems
Vitis Log
Guidance
Bootgen
[INFO] : Bootimage generated successfully

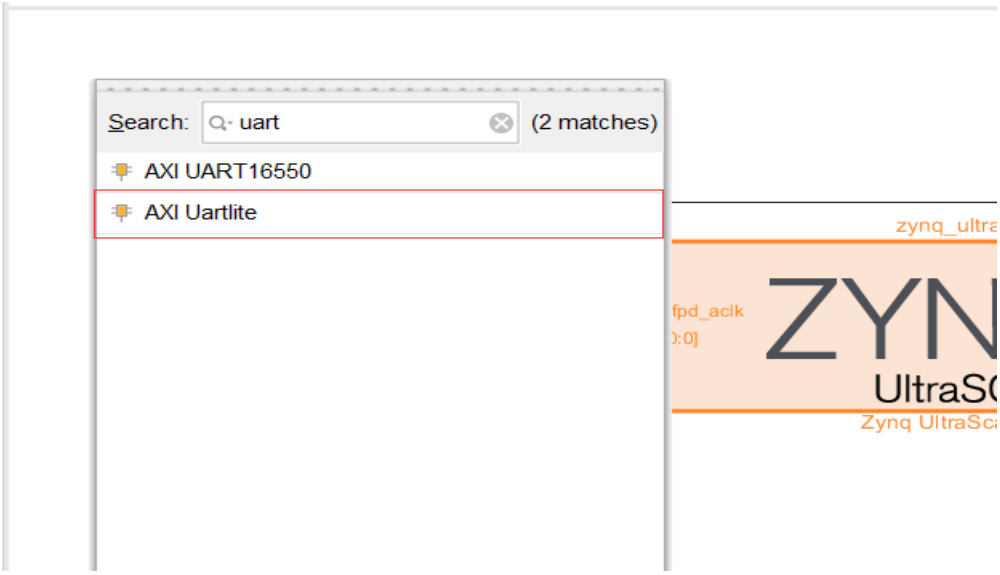
Chapter 6 uart_cycle

This section guides users to create a simple SOC design using vivado. By configuring PS and calling the axi_uartlite IP core on the PL side, the communication between PS and PL is realized through AXI GP Master.

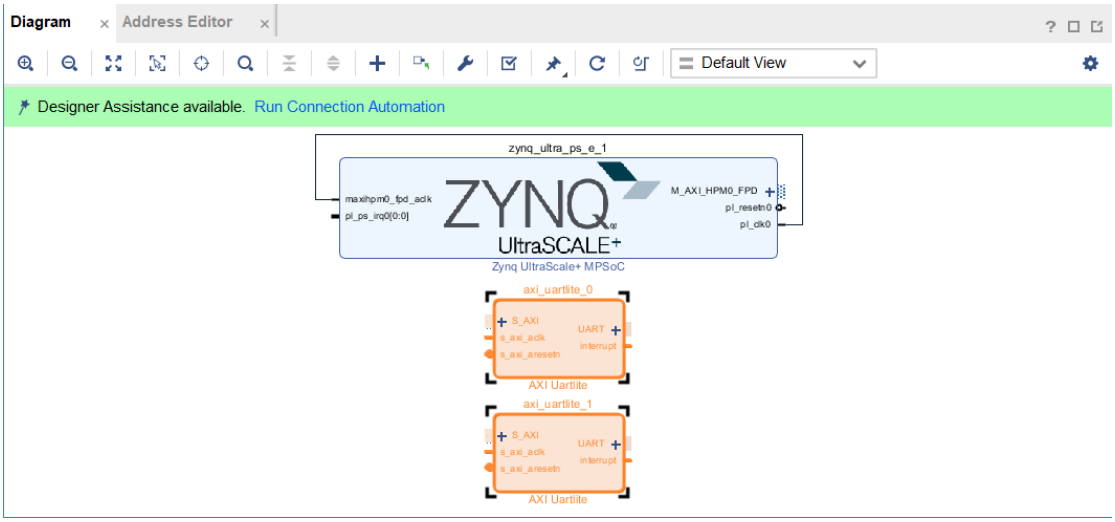
1.1 Create project and configure IP core of PS

Based on "ps_hello" project, save as uart_cycle project.

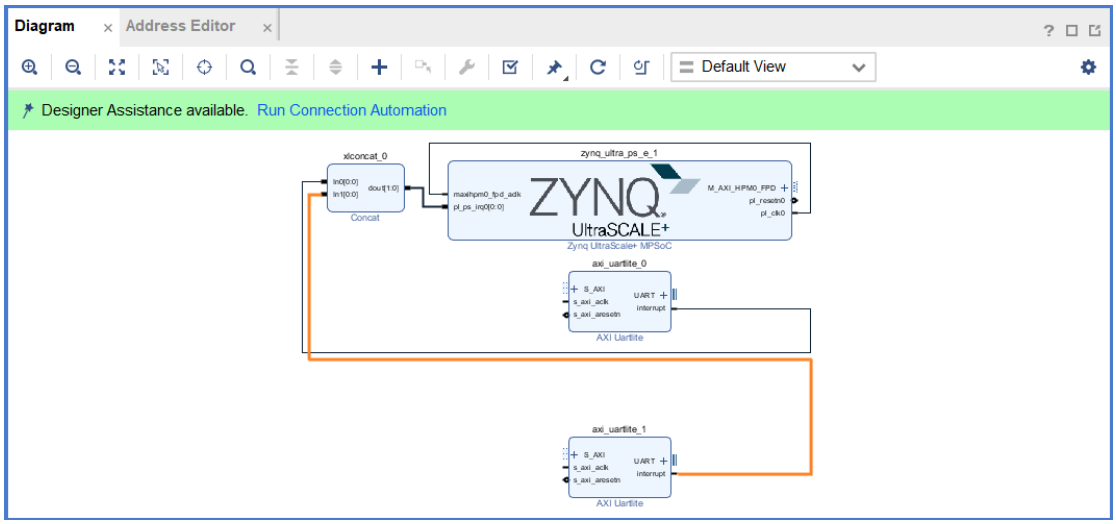
1.2 Add axi_uart IP core and configure them



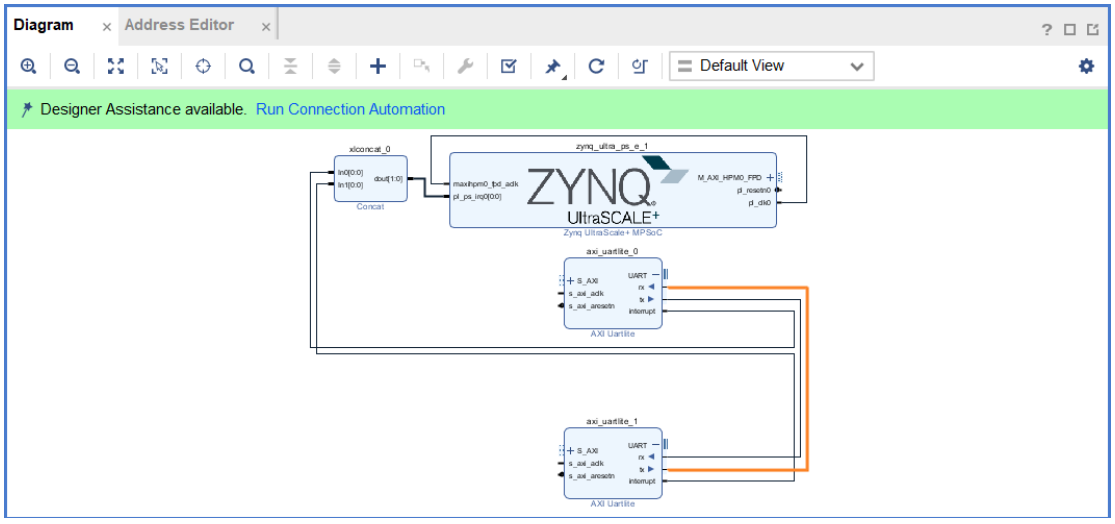
Add two uarts, where the configuration directly selects the default



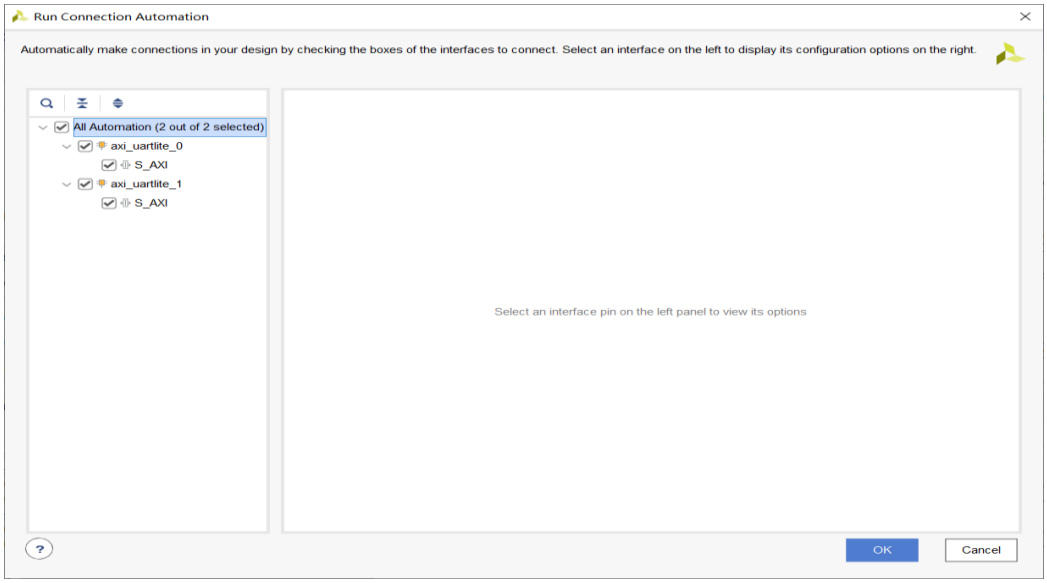
Add concat IP core and connect it to the interrupt ports.



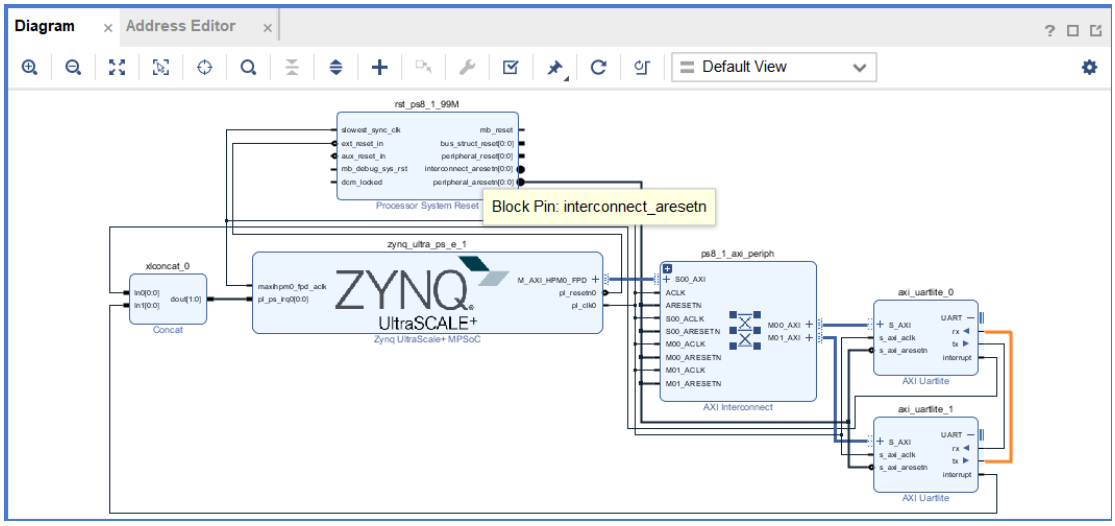
Connect the RX and TX of the two uarts, as shown in the following figure



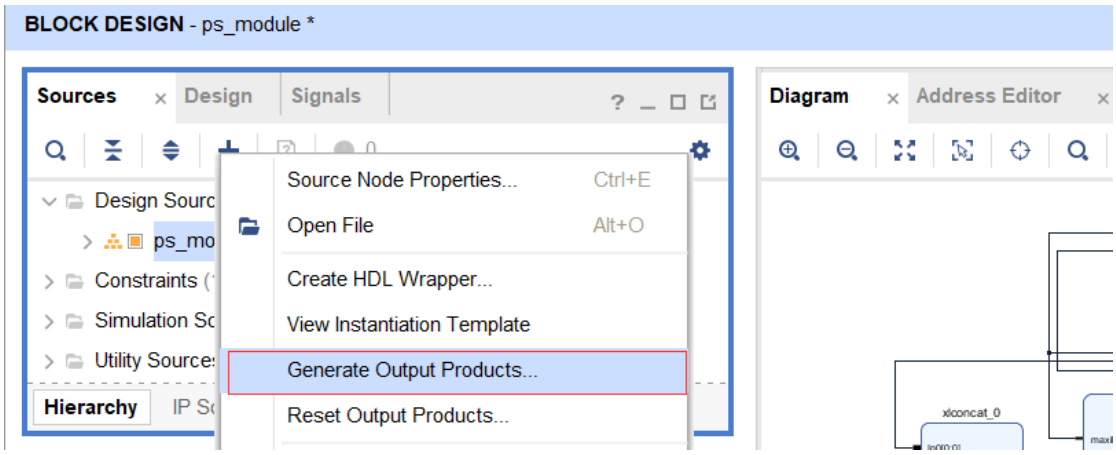
Click Run Connection Automation - > OK to connect automatically.



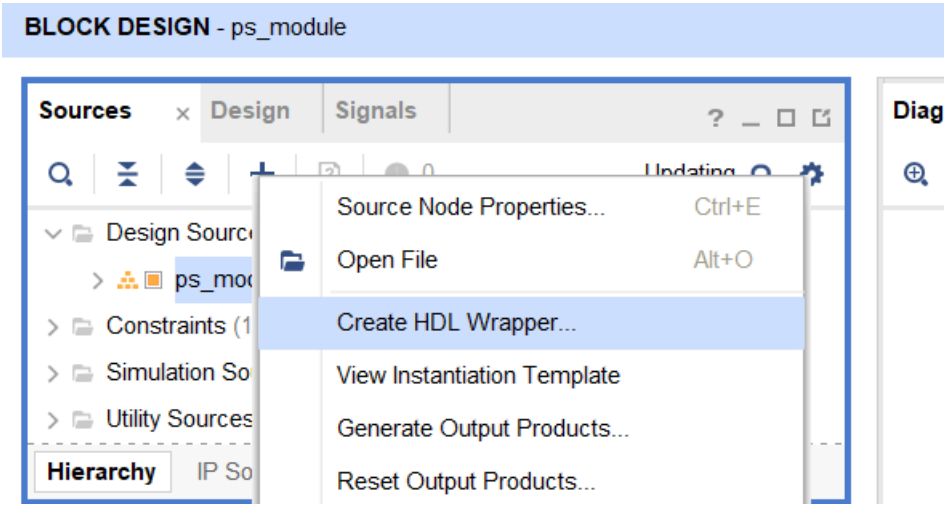
After the automatic connection is completed, the following figure is shown.



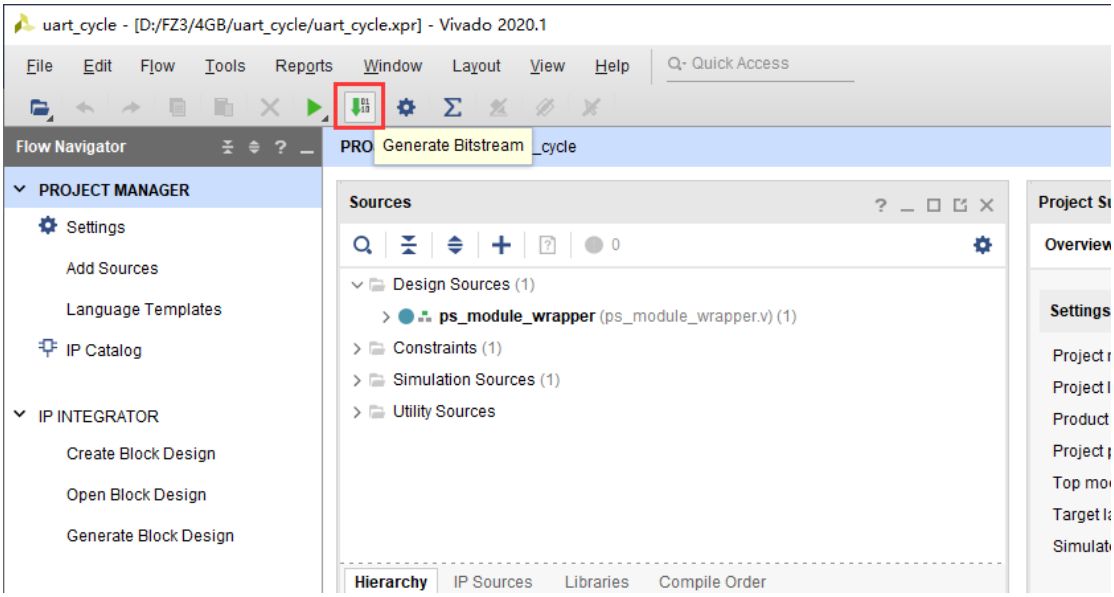
1.3 Generate synthesis files



1.4 Generating top-level file of FPGA

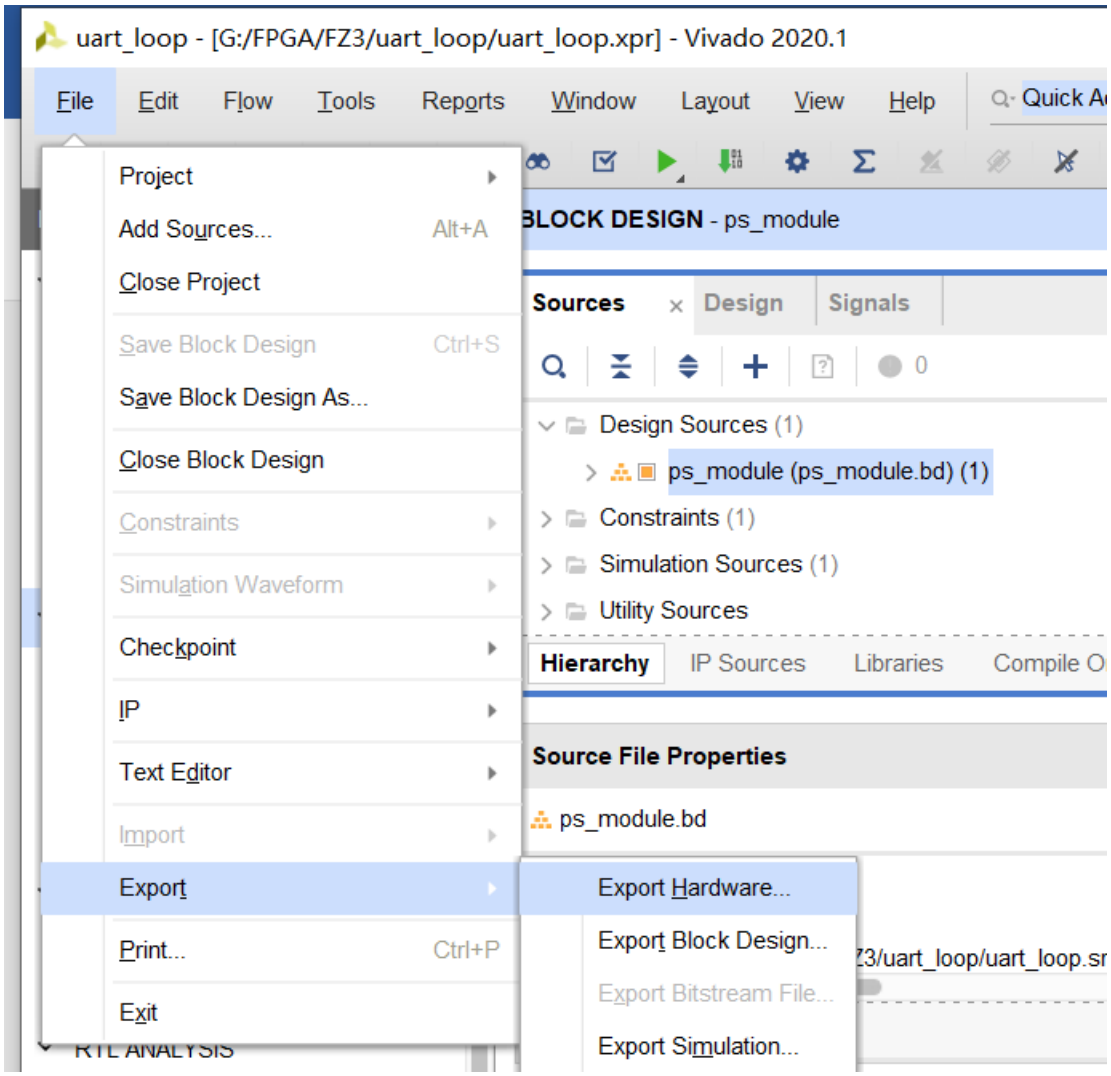


1.5 Generate bit files



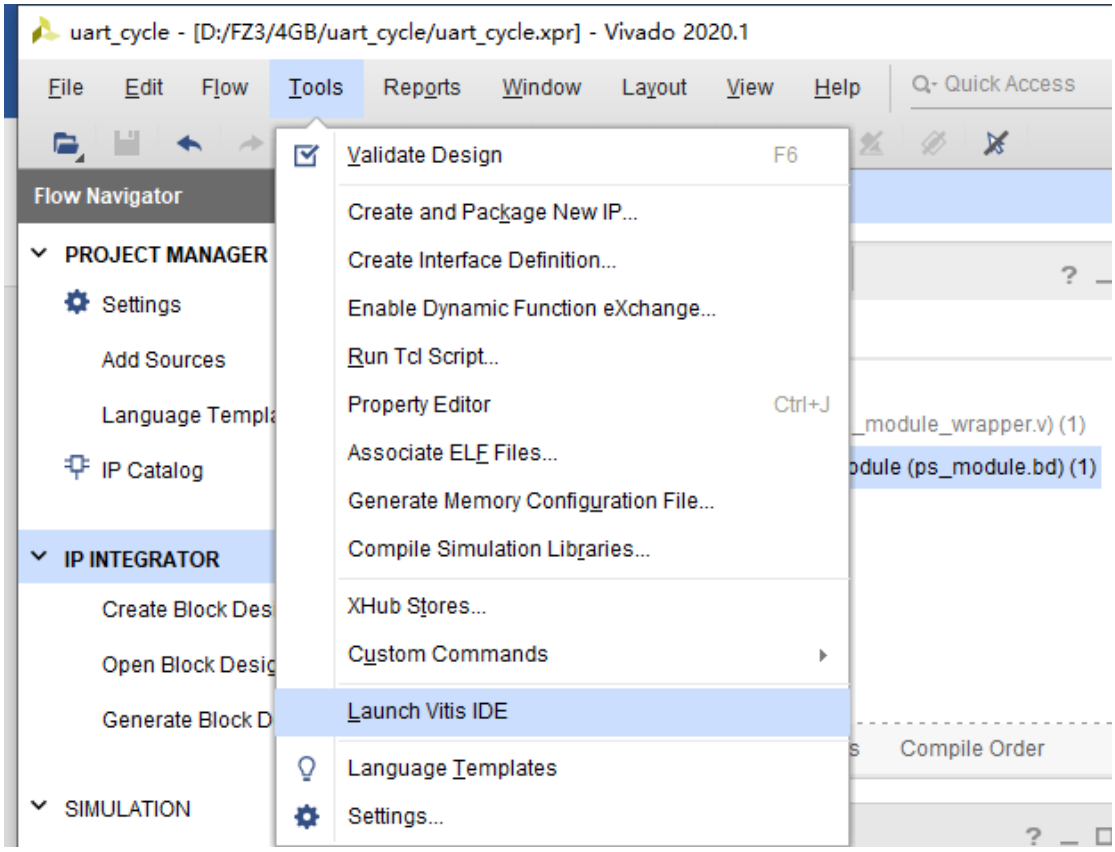
1.6 Export Hardware Profile

Click File - > Export - > Export Hardware - > OK on the menu bar to export the hardware configuration file

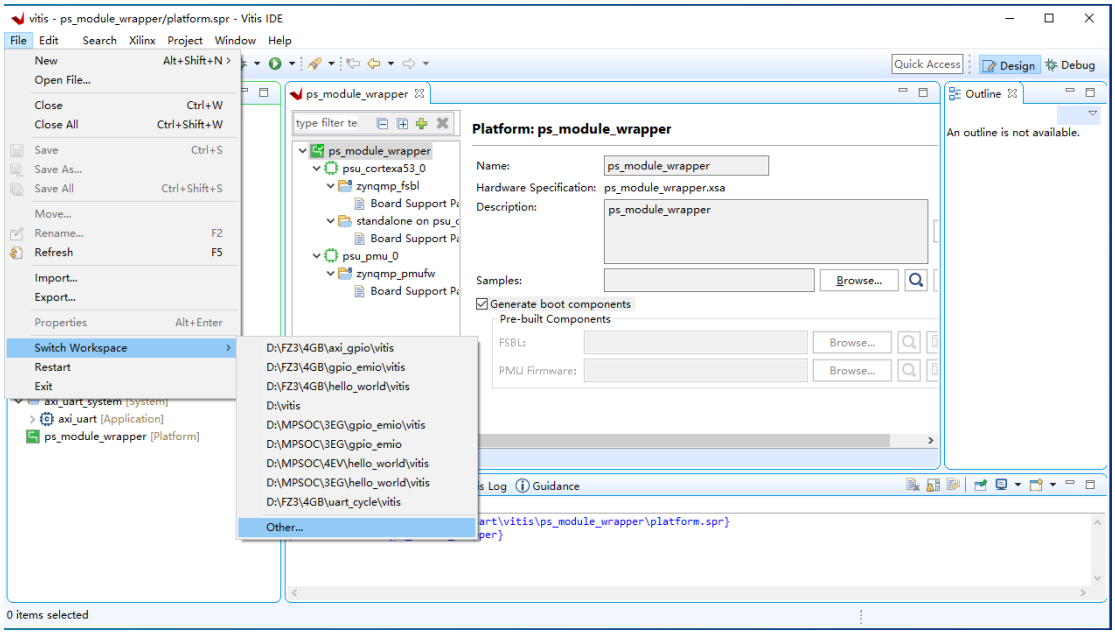

uart_loop - [G:/FPGA/FZ3/uart_loop/uart_loop.xpr] - Vivado 2020.1
File
Edit
Flow
Tools
Reports
Window
Layout
View
Help
Project
Add Sources... Alt+A
Close Project
Save Block Design Ctrl+S
Save Block Design As...
Close Block Design
Constraints
Simulation Waveform
Checkpoint
IP
Text Editor
Import
Export
Print... Ctrl+P
Exit
BLOCK DESIGN - ps_module
Sources
Design
Signals
Design Sources (1)
ps_module (ps_module.bd) (1)
Constraints (1)
Simulation Sources (1)
Utility Sources
Hierarchy
IP Sources
Libraries
Source File Properties
ps_module.bd
Export Hardware...
Export Block Design...
Export Bitstream File...
Export Simulation...

1.7 Launch Vitis and create new platform project

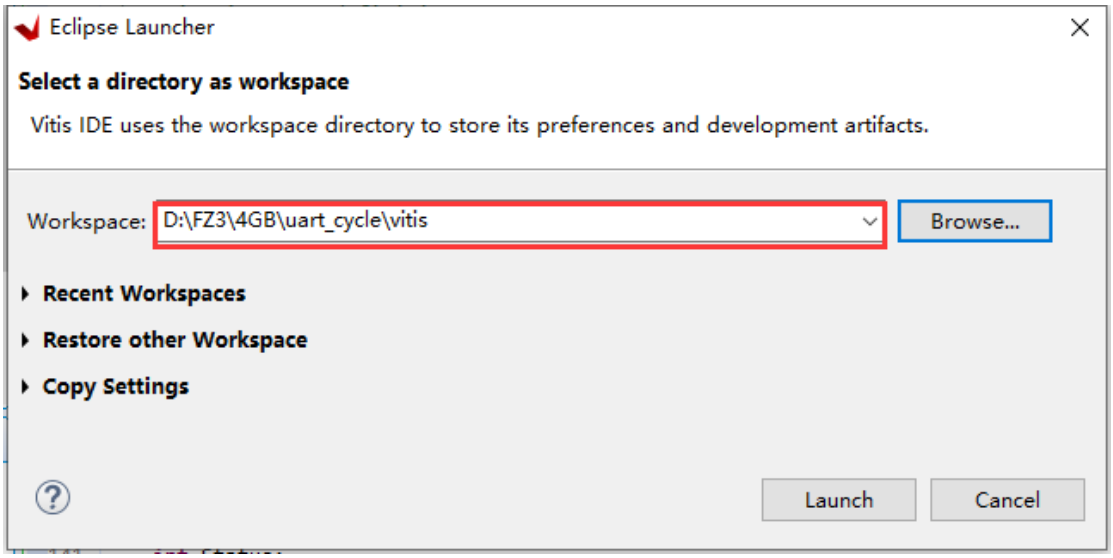
Click Tools > launch Vitis ide on the menu bar to launch Vitis.



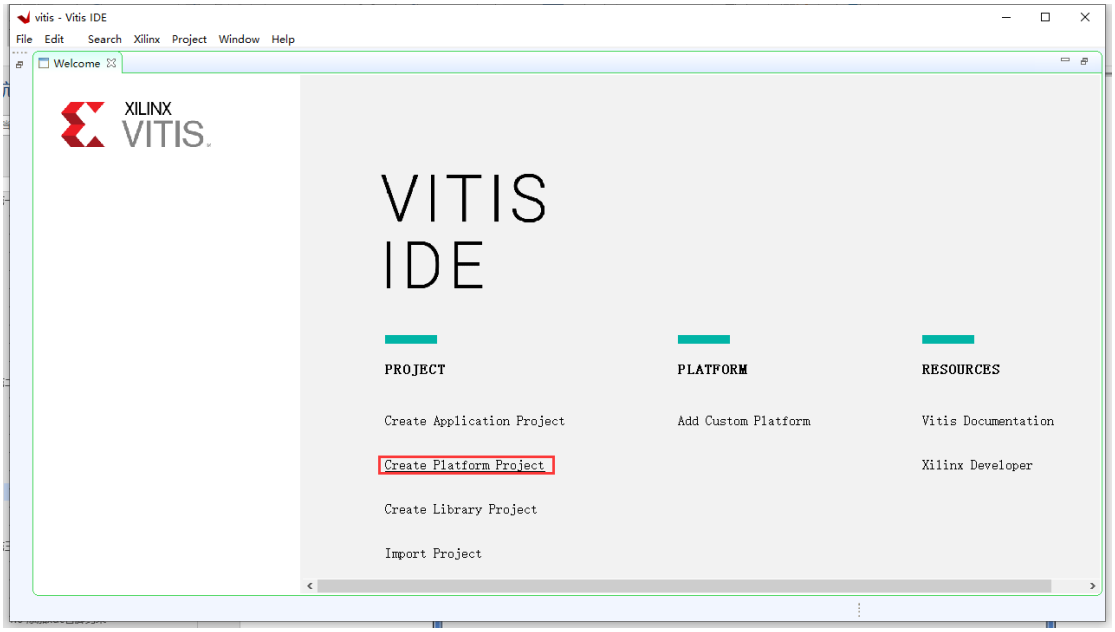
Click File→Switch Workspace→Other…,



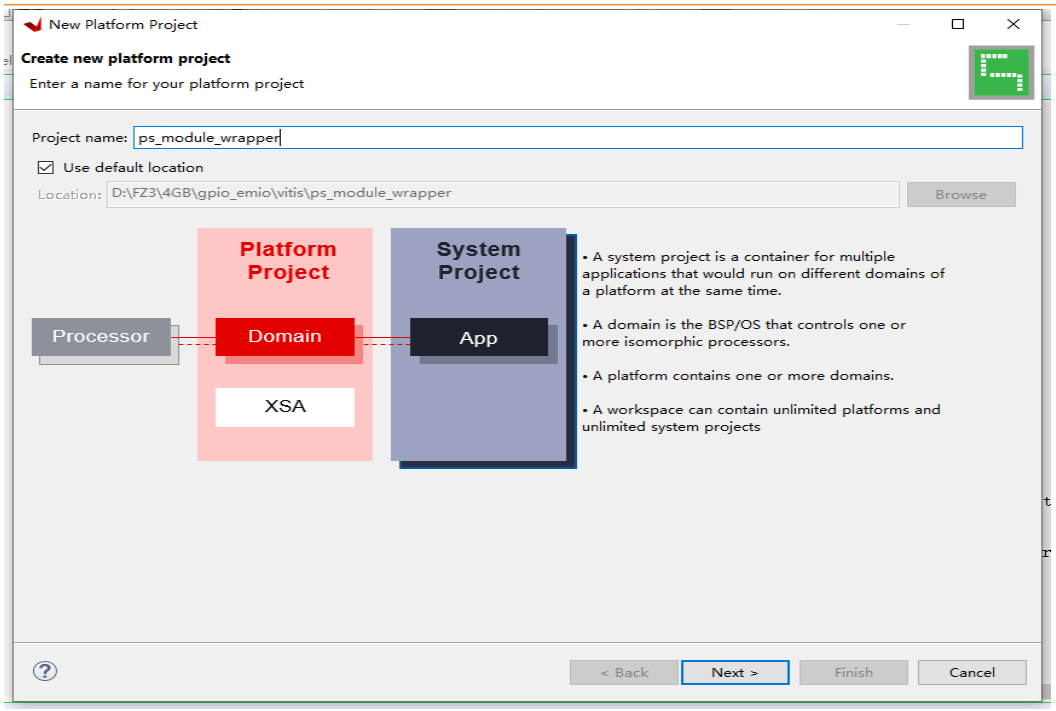
elect the path, select the vitis folder under uart_cycle project. Click launch.



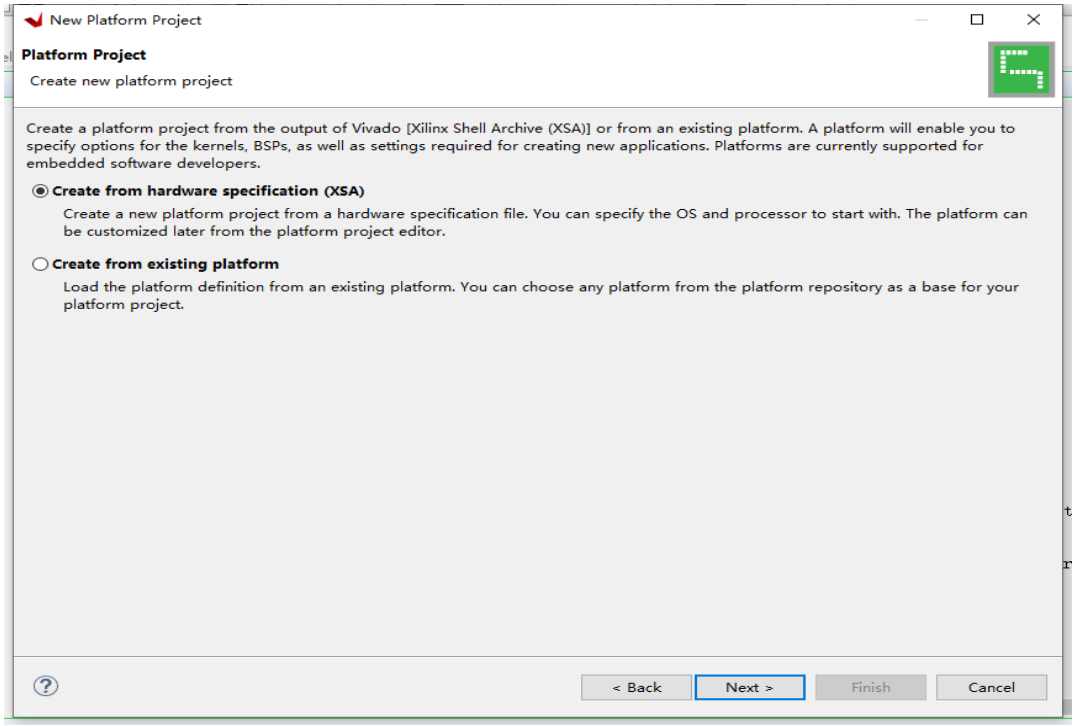
Click Create Platform Project



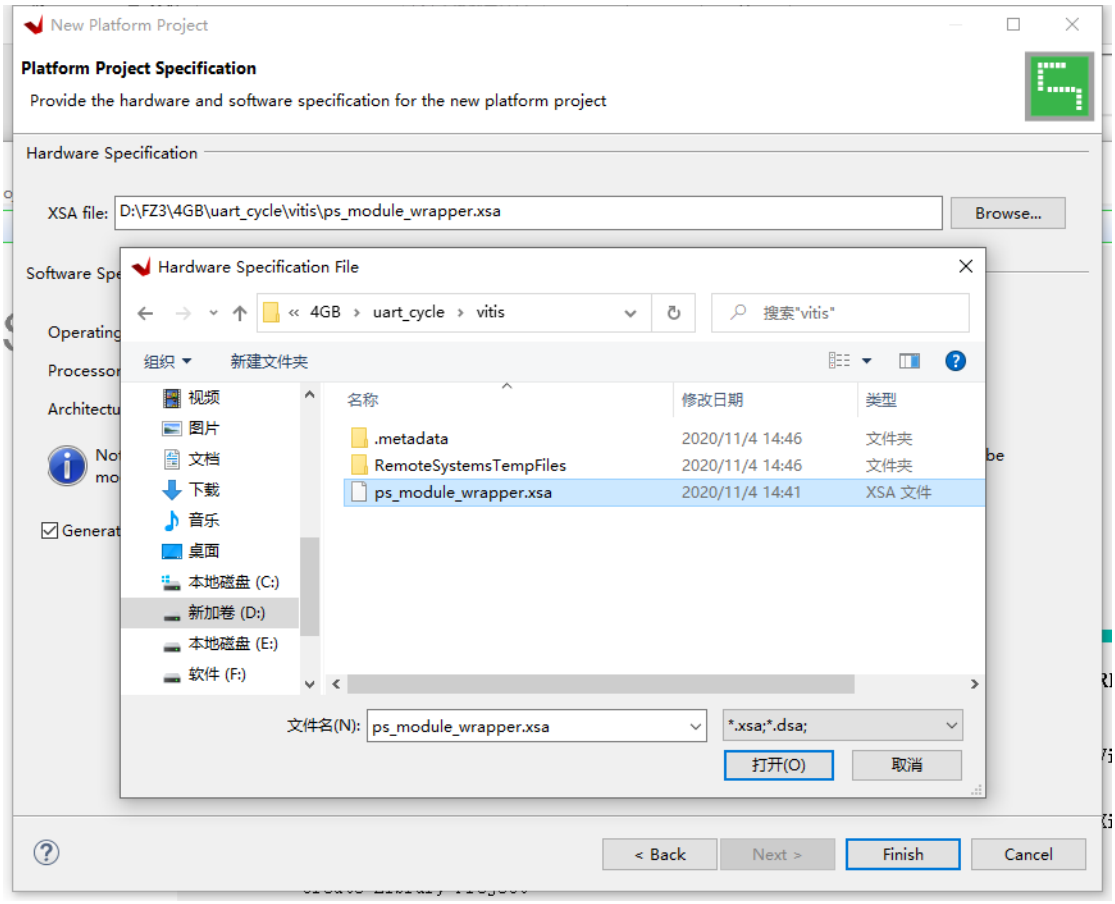
The project name uses the same name as the xsa file. Click next.



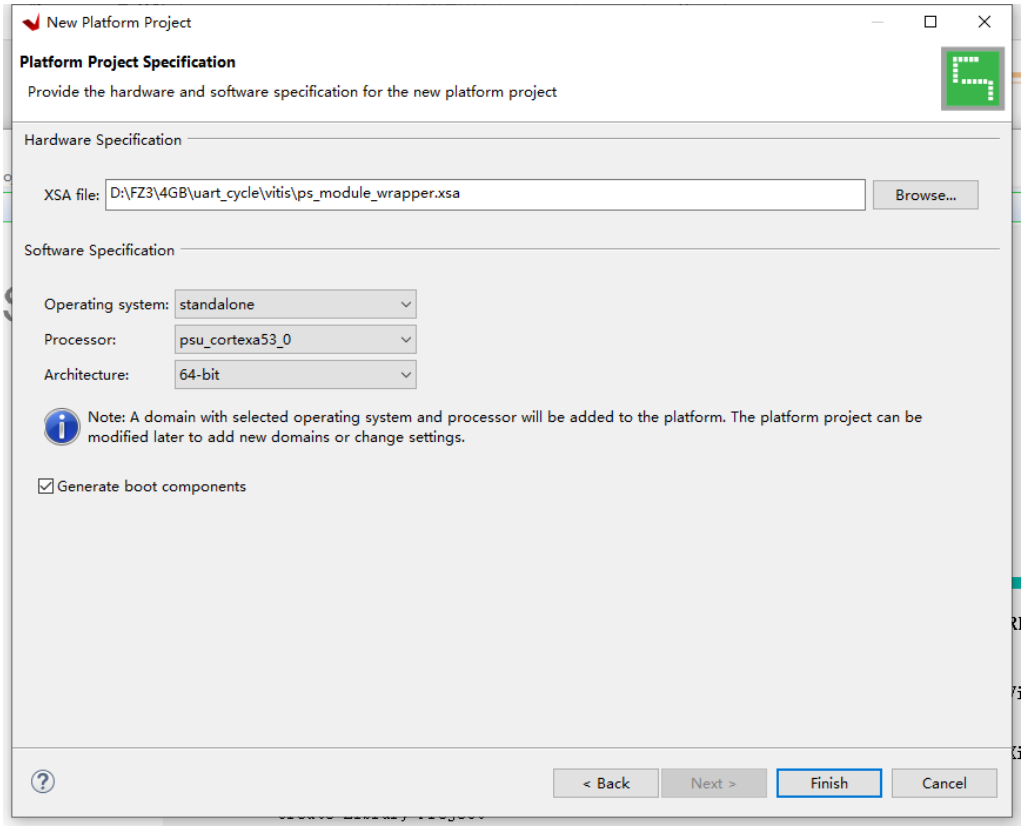
Select Create from hardware specification(XSA),click NEXT.



Open Browse… , select the previously generated xsa file.

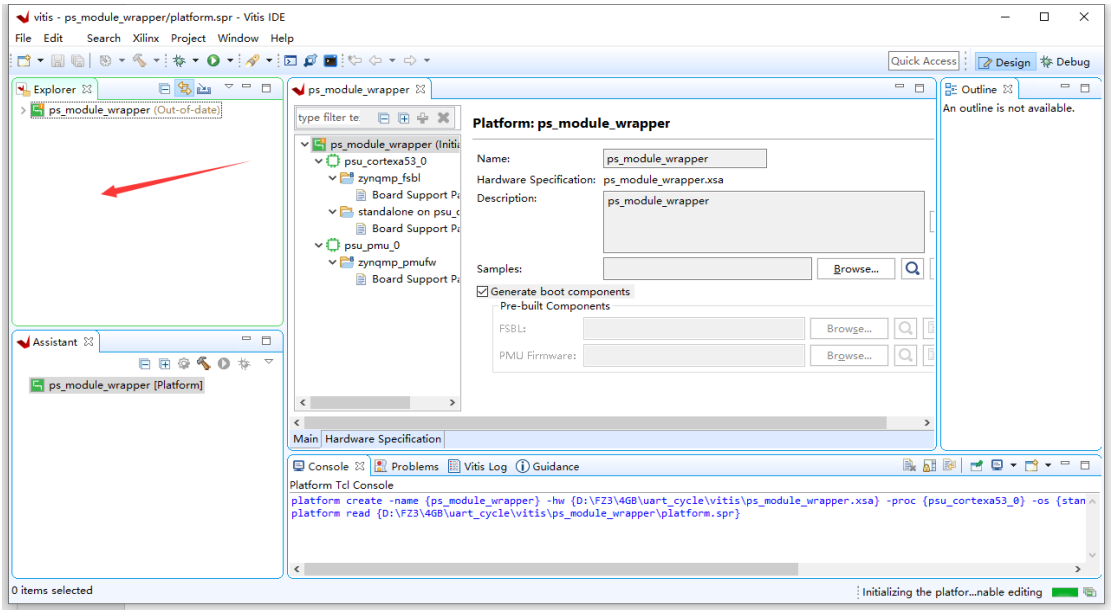


Leave the default and click finish.

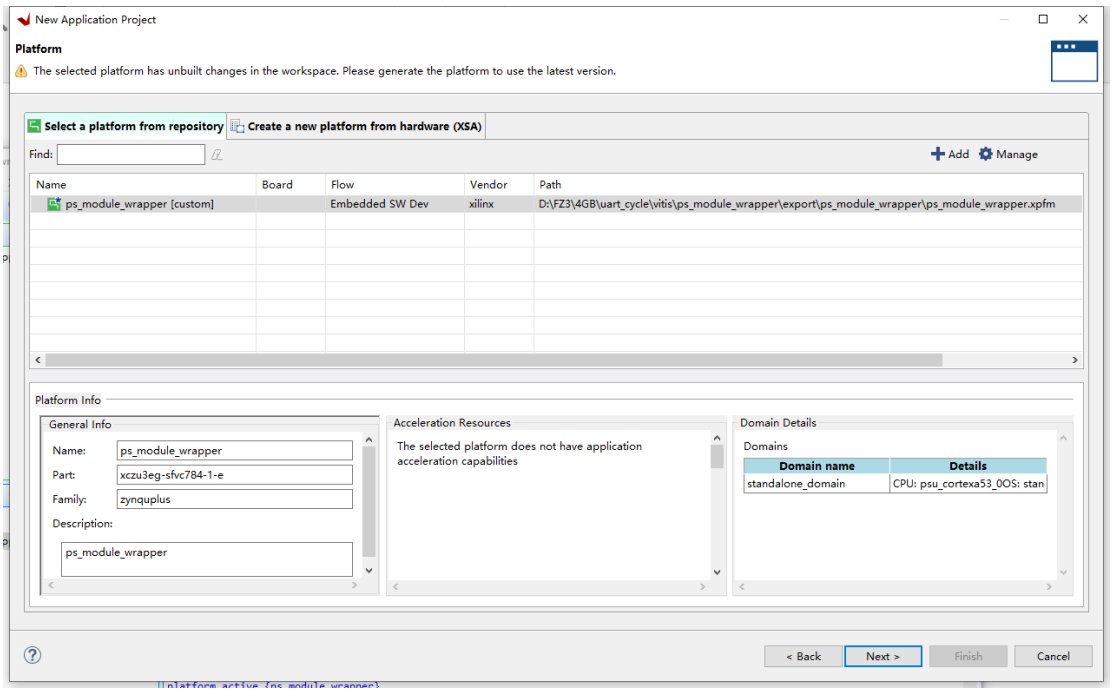


1.8 New uart_cycle Project

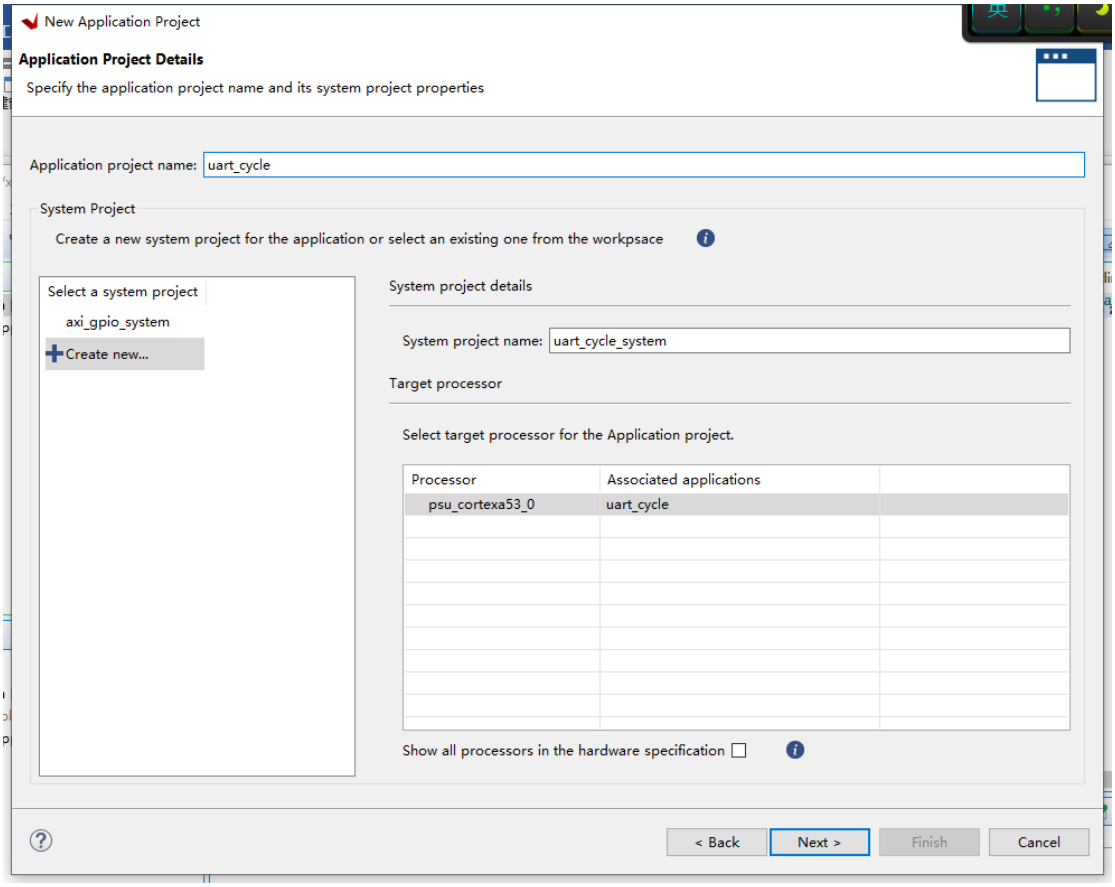
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



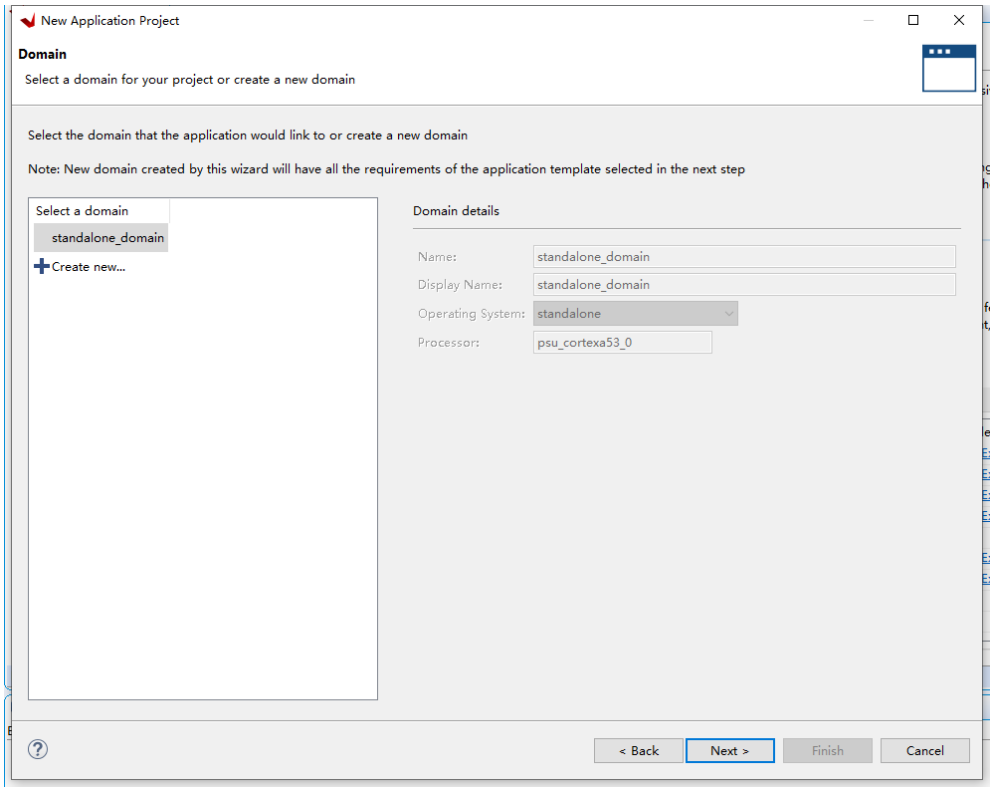
Skip to the second page and select the platform project created above (default), next.



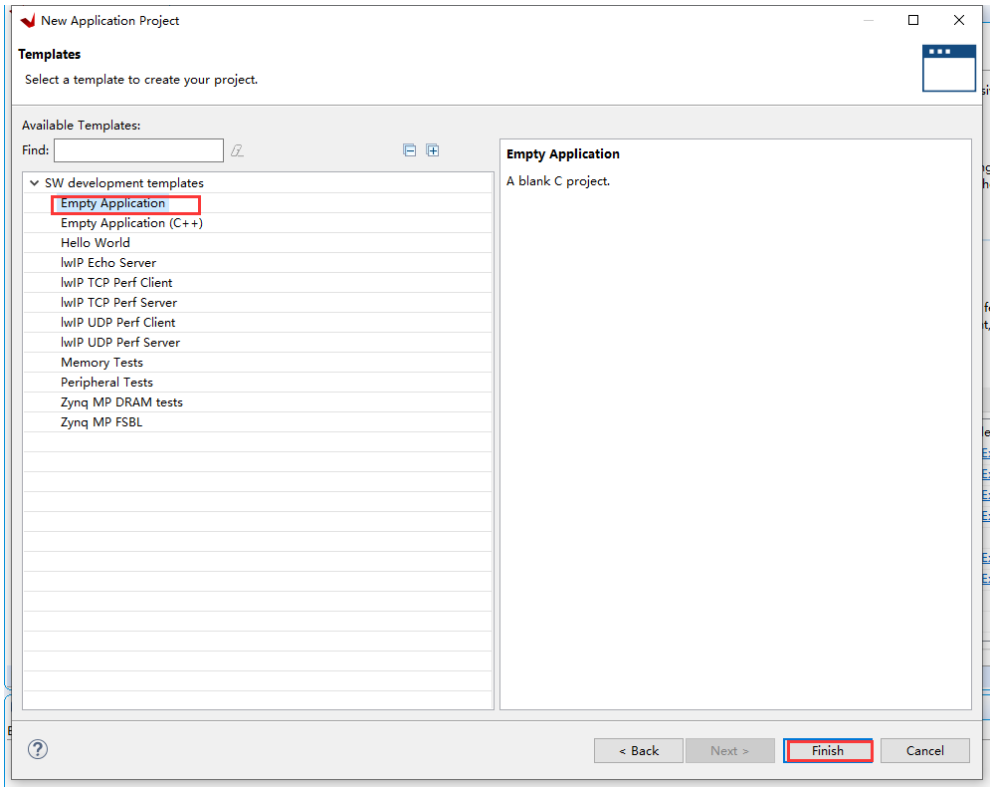
Project name enter uart_cycle，Next



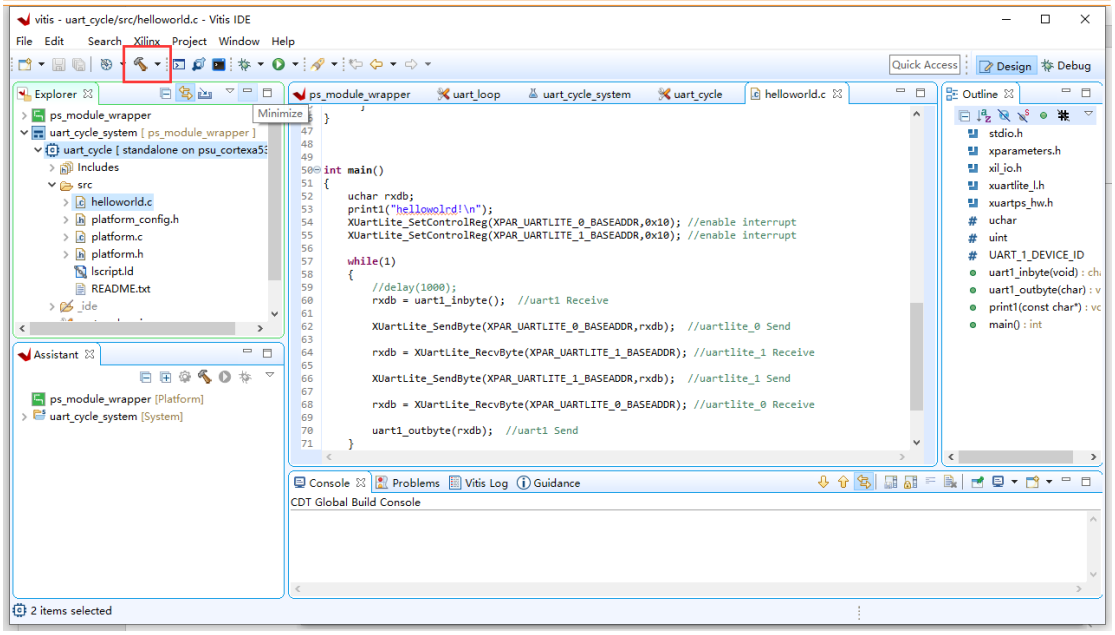
Next



Select Empty Application， Finish

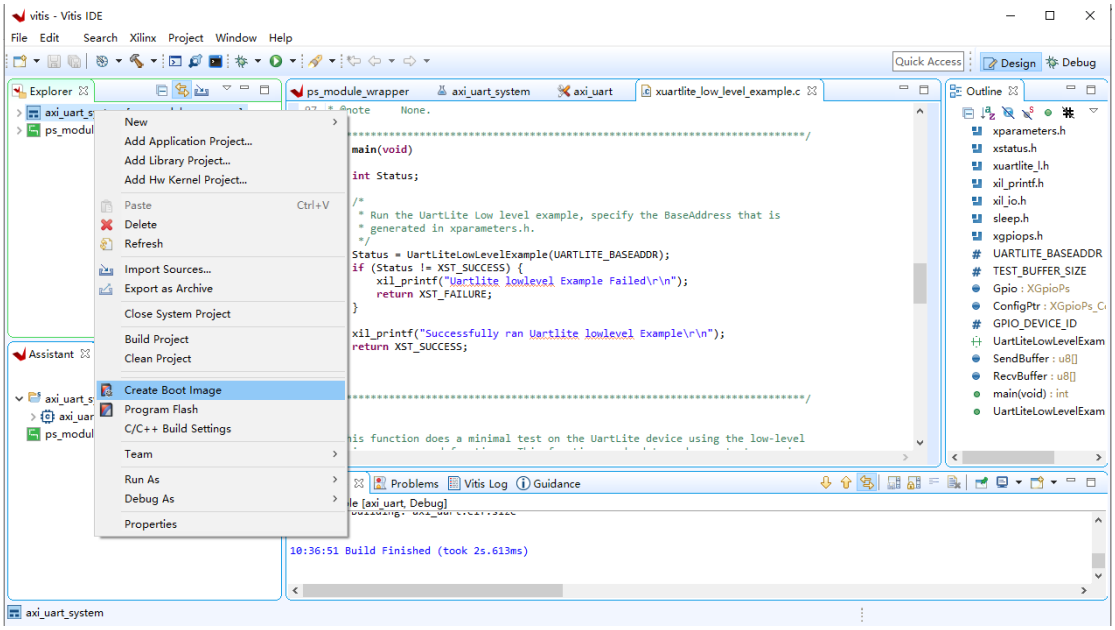


Copy the files in the example project to src, select the project, and click the compile buttom.

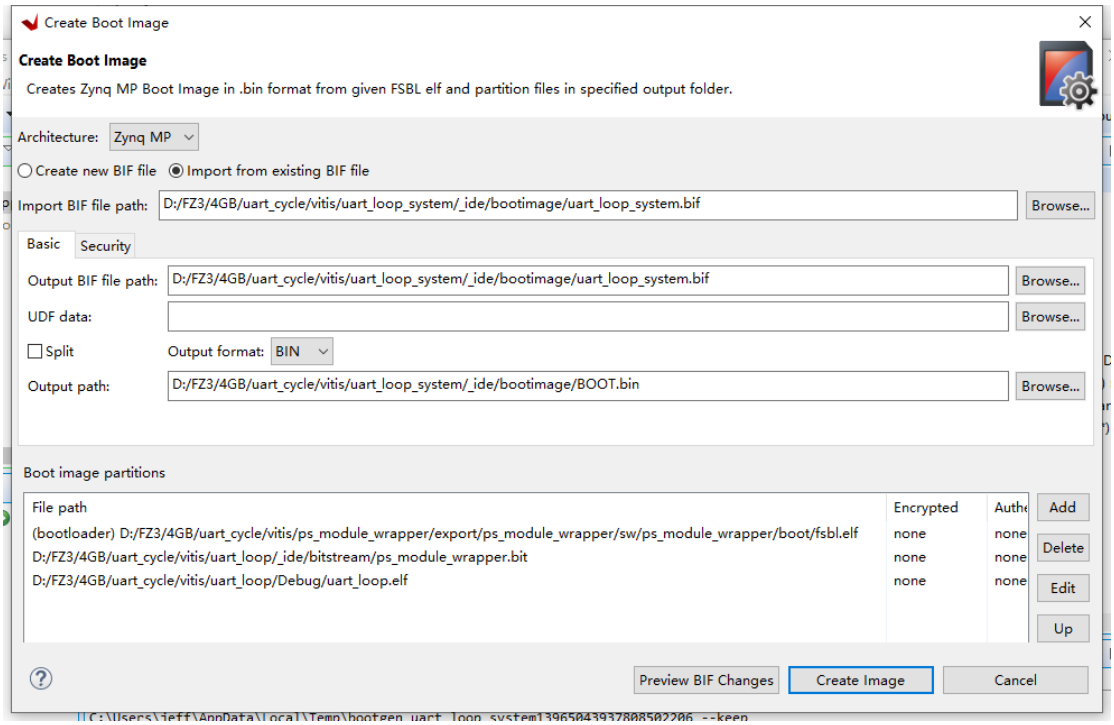


1.9 Generate BOOT.bin file

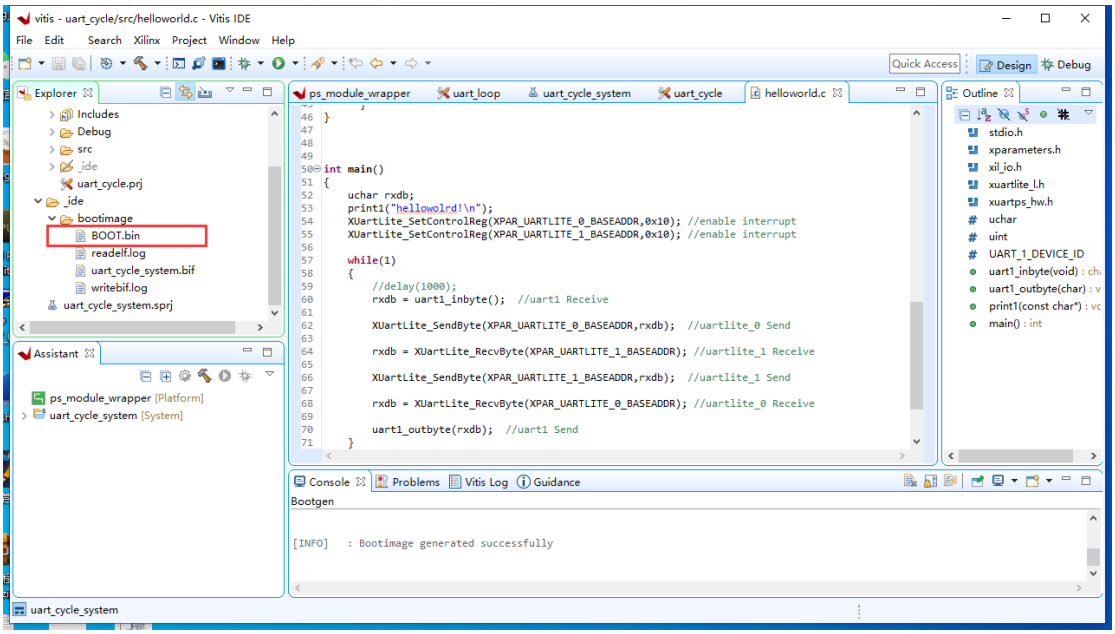
Right-click the system of APP project and select Create boot image.



Click Create Image to generate BOOT.bin Startup file.



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.



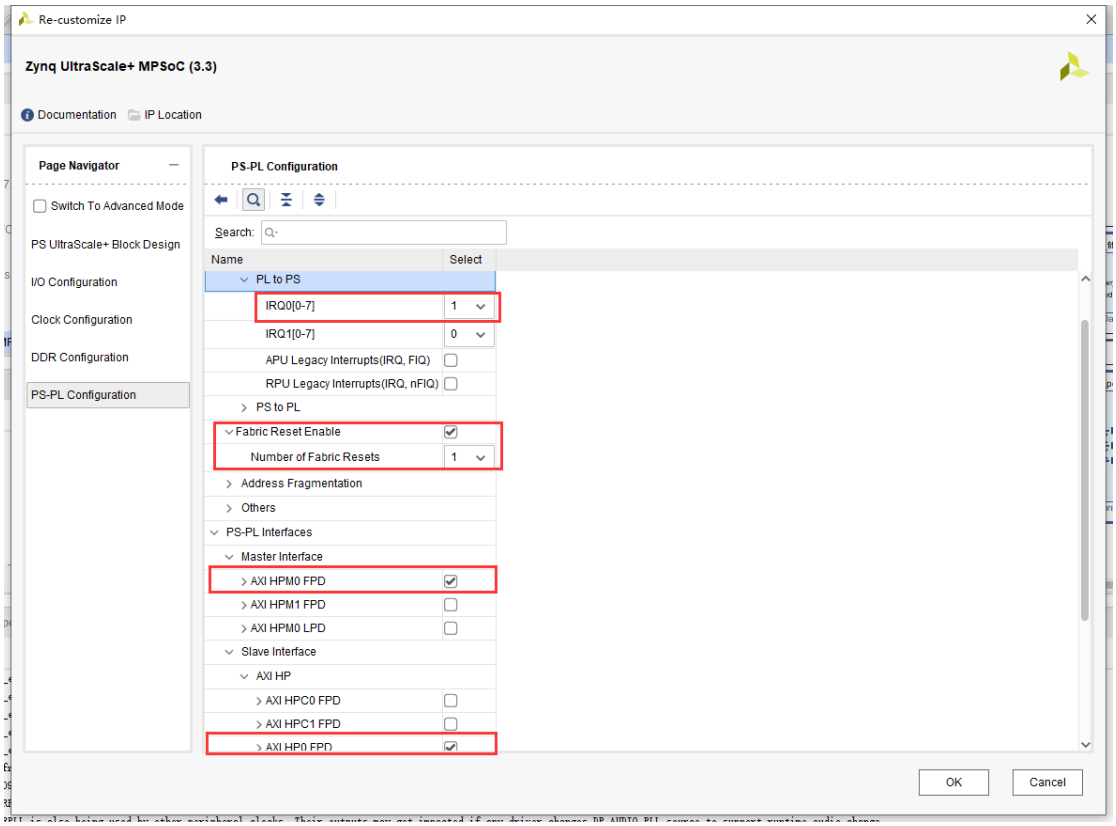
Chapter 7 dma_loop

This section guides users to create a DMA project using vivado. High-speed PL and PS through DMA Data transmission.

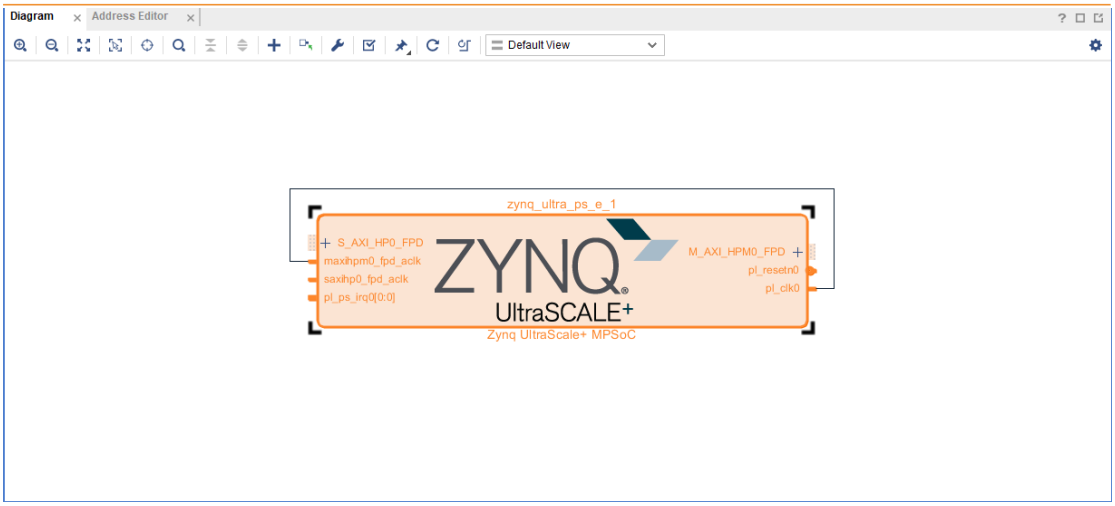
1.1 Add PS IP Core and Configure

Based on "ps_hello" project, save as dma_loop project.

Open reset,， HPM0,HP0， PL to PS interrupt IRQ0。

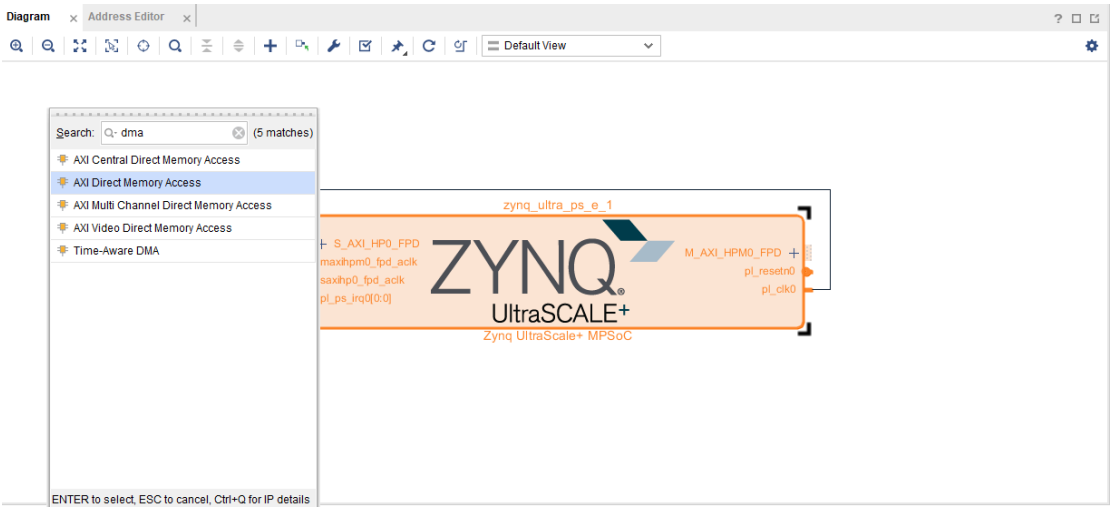


When the configuration is complete, the Zynq UltraScale + MPSoC core is shown in the following figure.

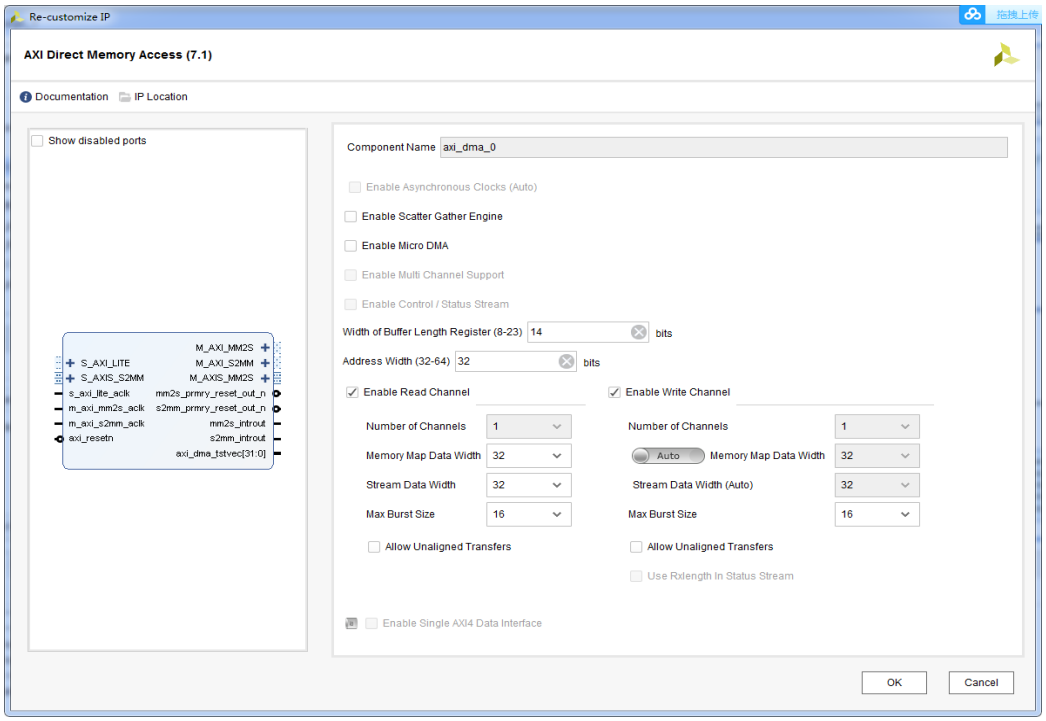


1.2 Add IP Core and Configure

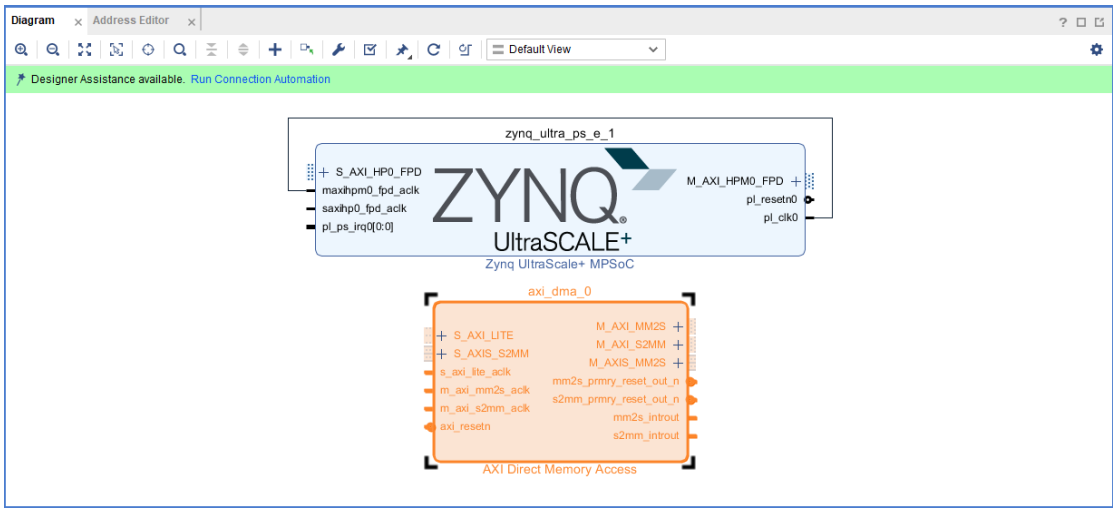
Calling DMA Core



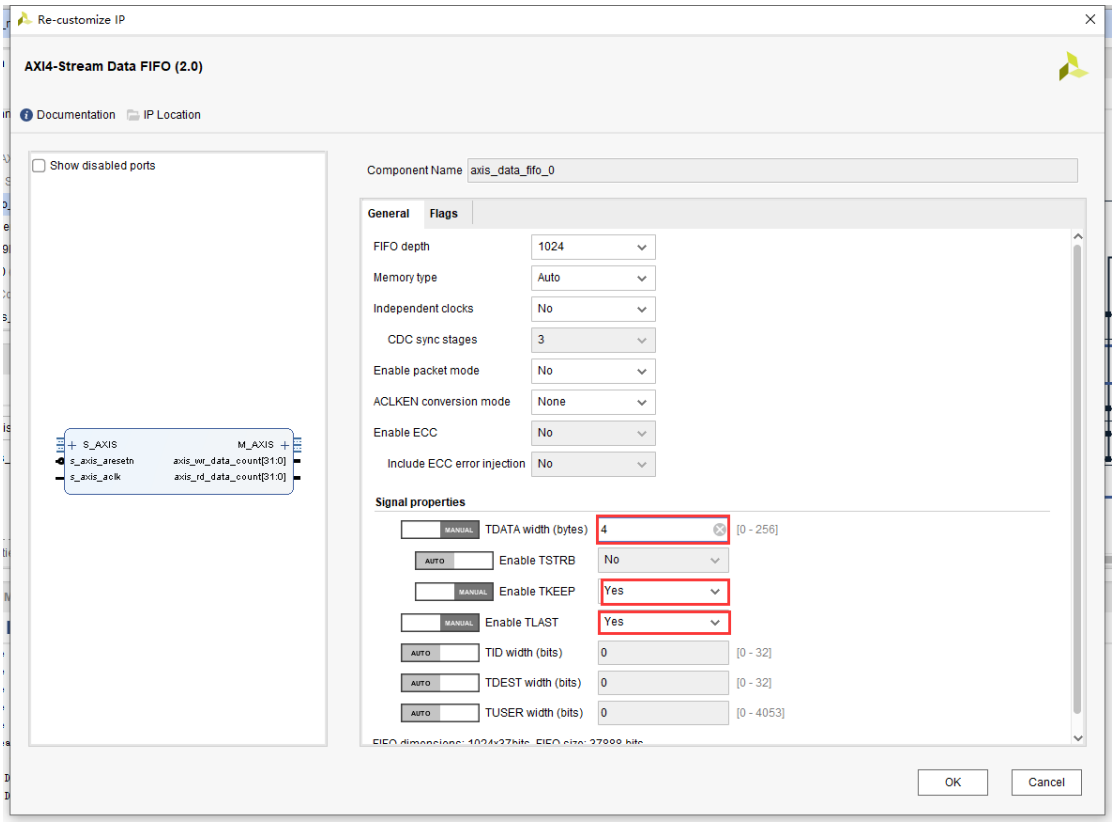
Double-click on the DMA IP core to set parameters.

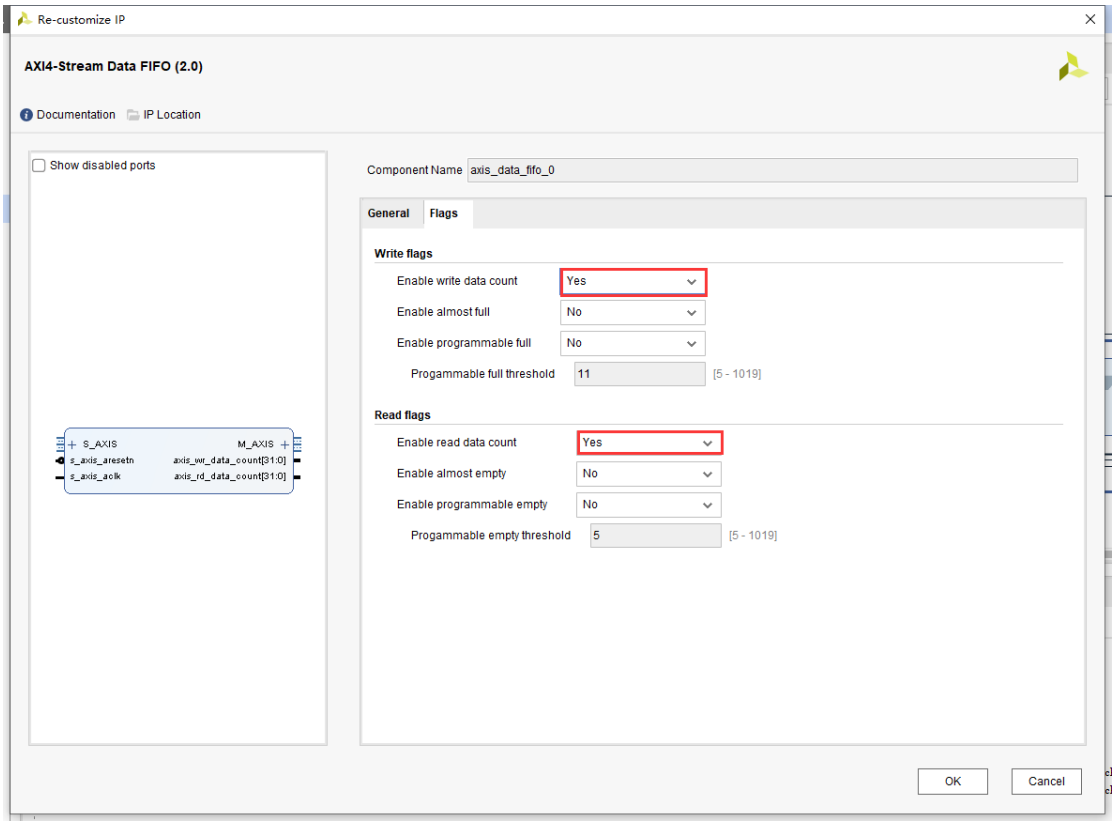


When the setup is complete, the following figure is shown.

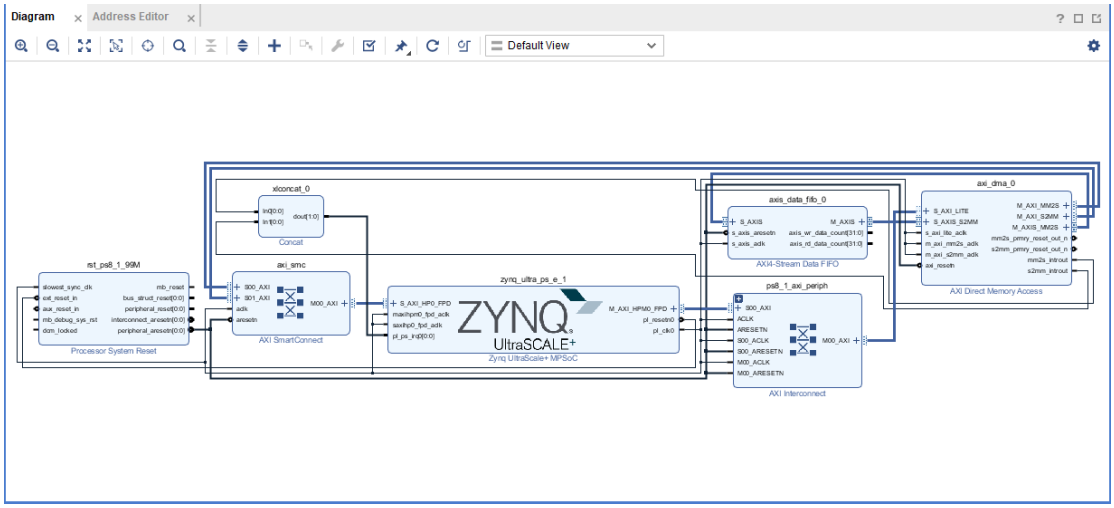


Add Axi Stream Data FIFO module and set it as follows: set the depth to 1024, Tdata width to 4 bytes, and open TKEEP and TLAST.

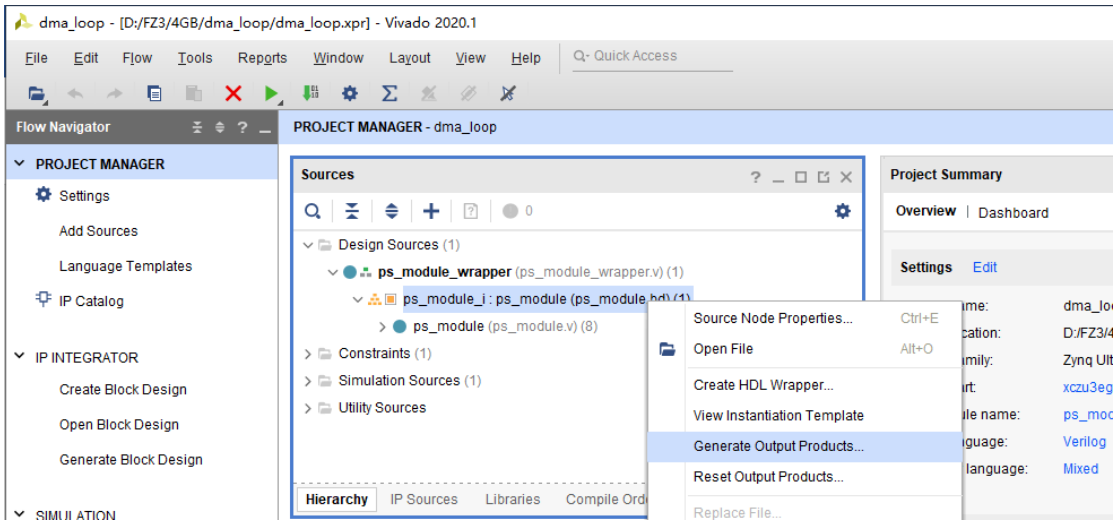




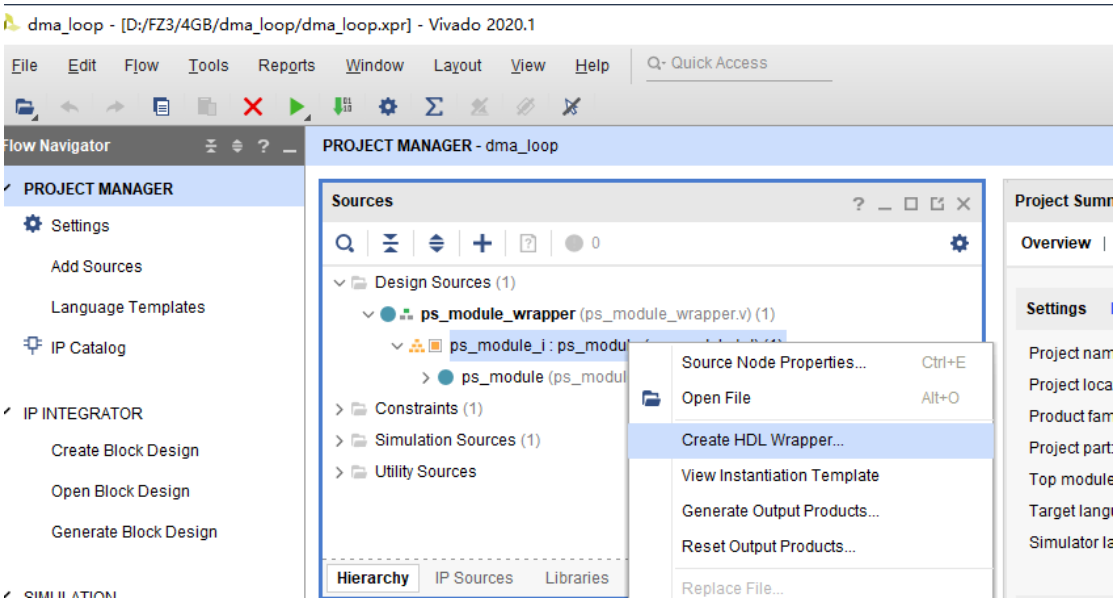
Automatic connection, and connect the S_AXIS and M_AXIS of FIFO to DMA, add concat, connect MM2S and S2MM, interrupt to pl_ps_irq the final result is shown in the figure below.



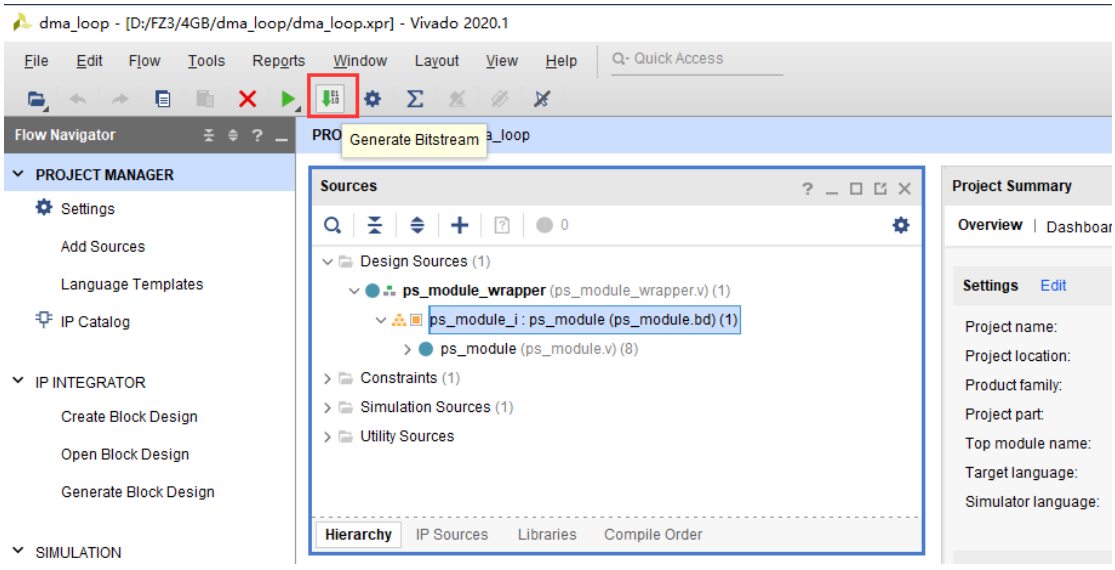
1.3 Generate synthesis files



1.4 Generating top-level file of FPGA

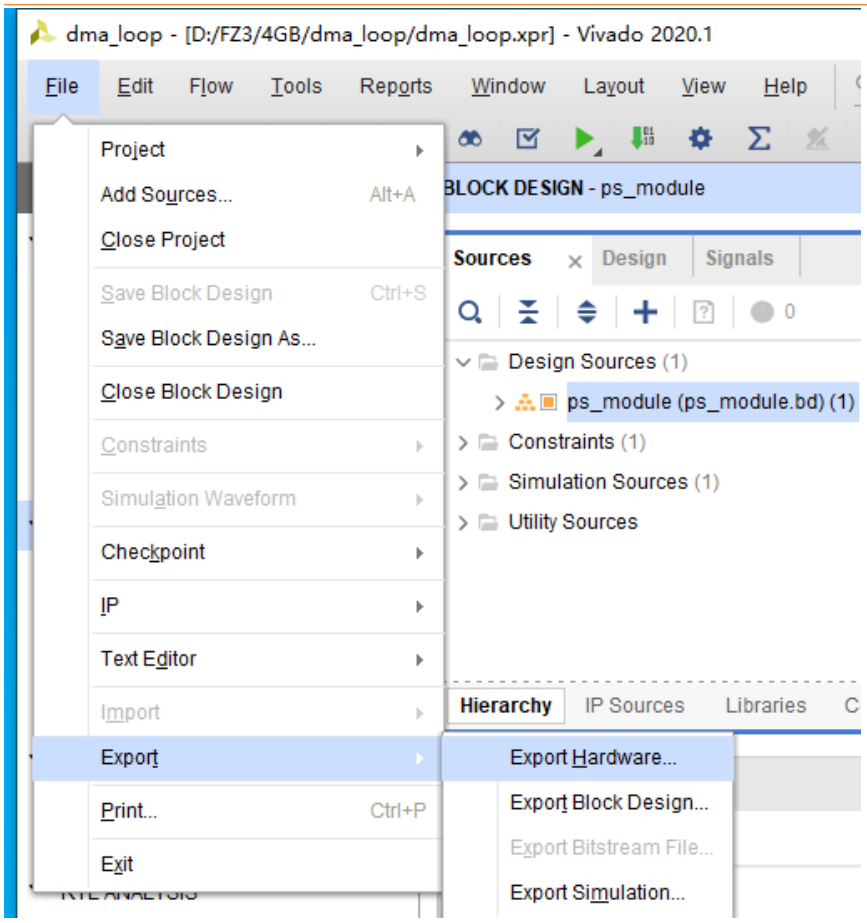


1.5 Generate bit files



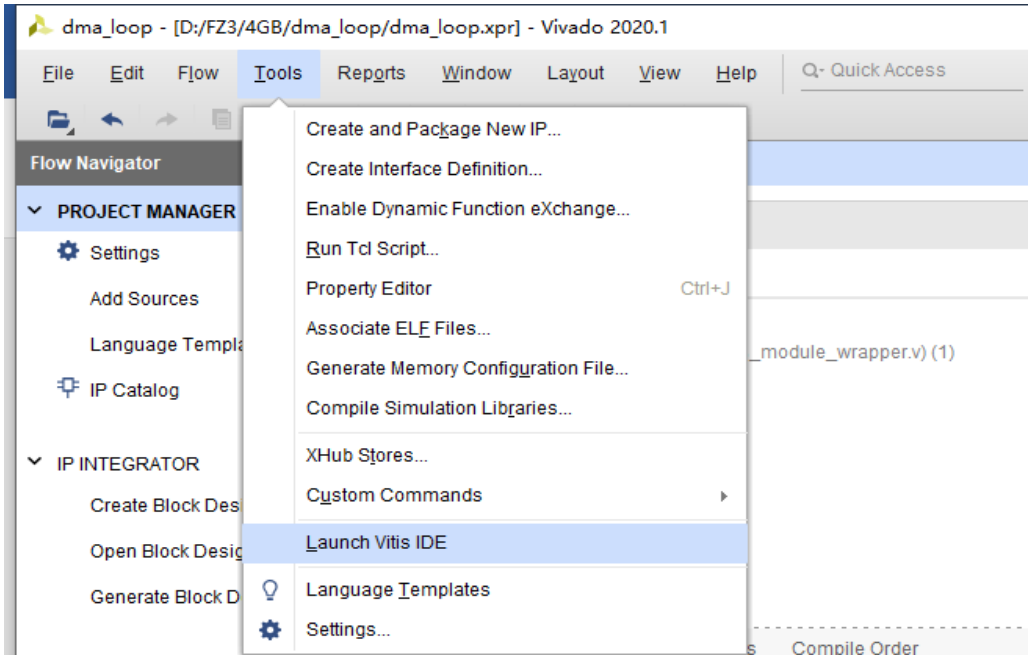
1.6 Export Hardware Profile

Click File - > Export - > Export Hardware - > OK on the menu bar to export the hardware configuration file

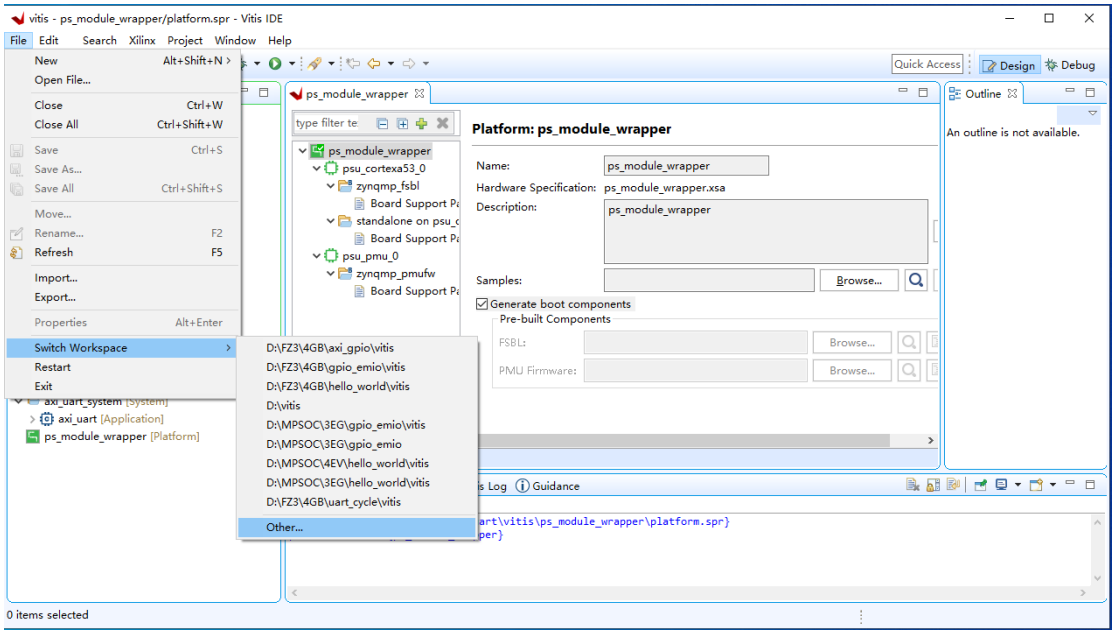


1.7 Launch Vitis and create new platform project

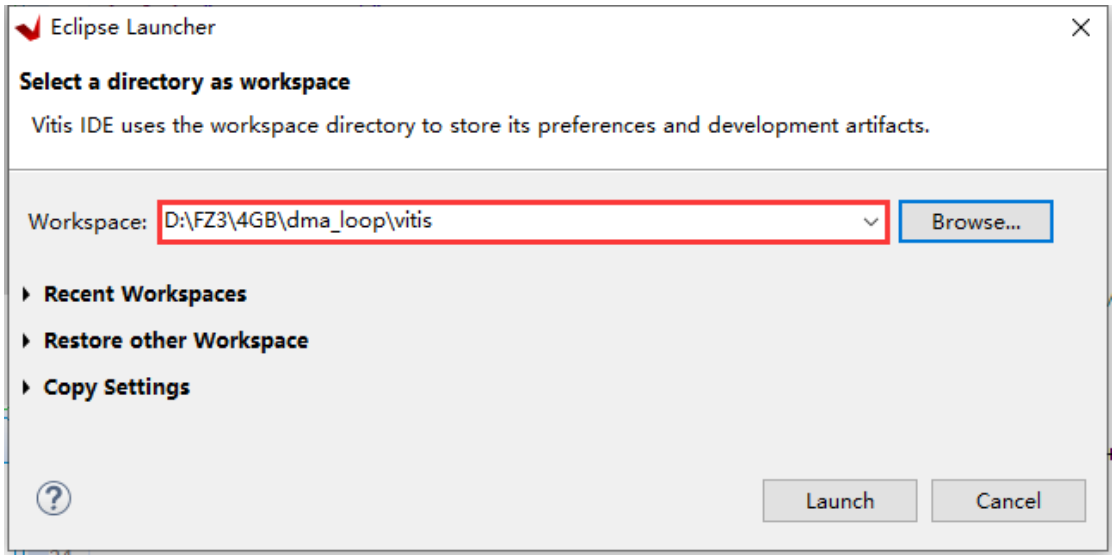
Click Tools > launch Vitis ide on the menu bar to launch Vitis.



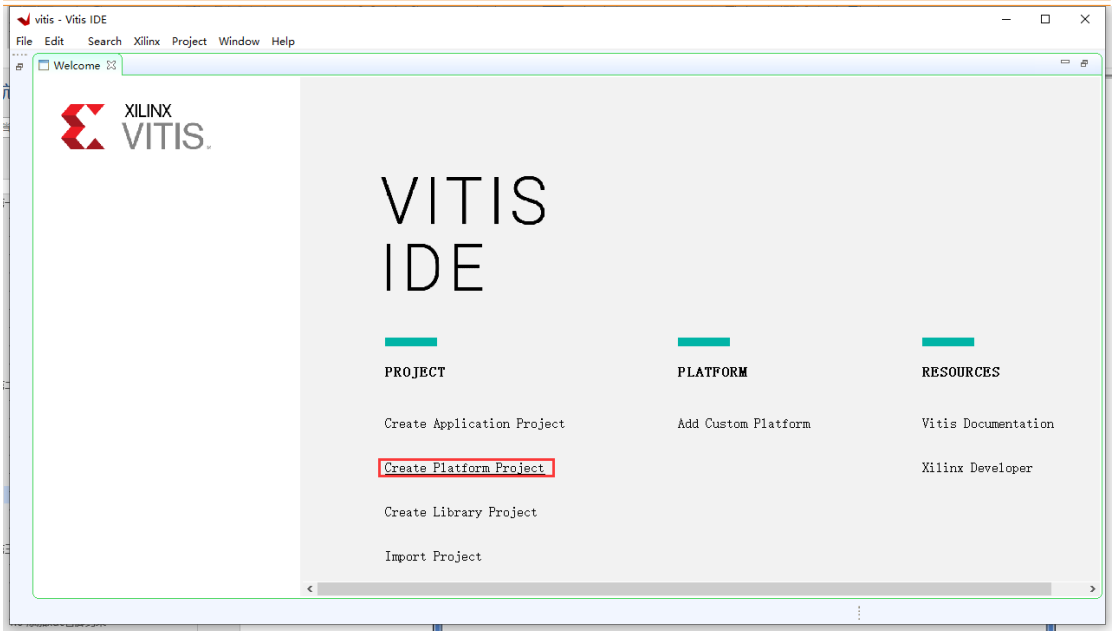
Click File→Switch Workspace→Other...,



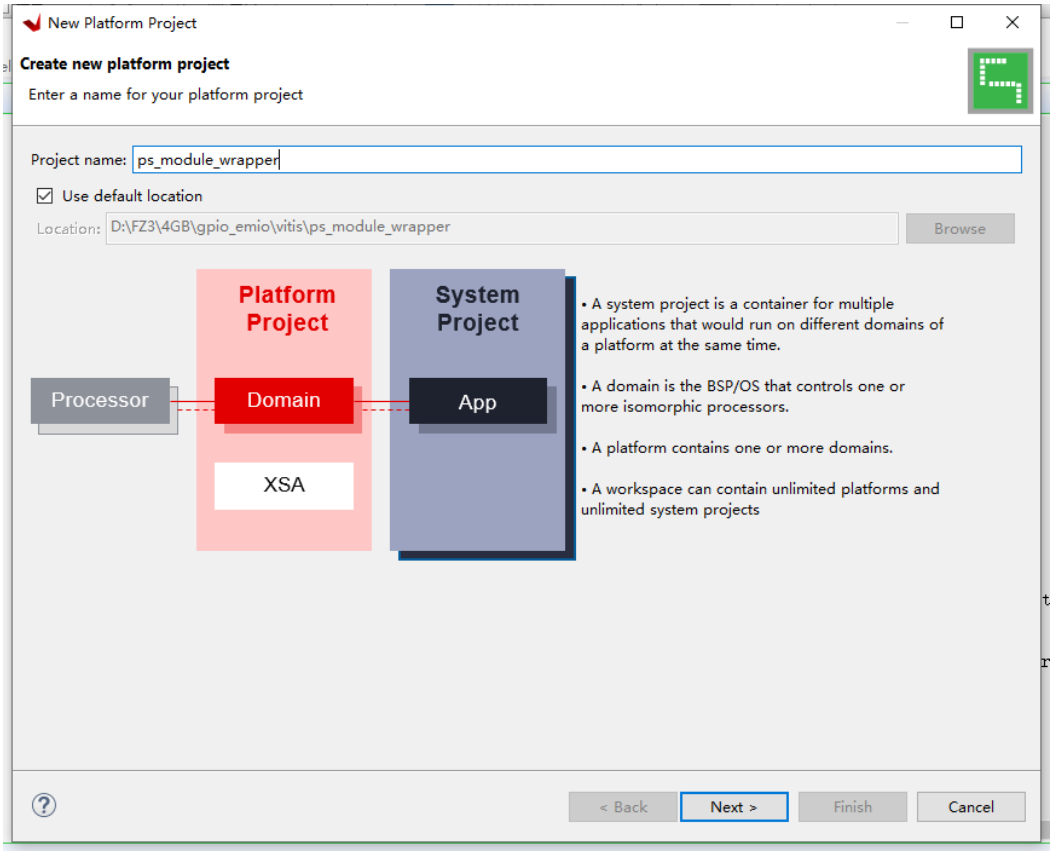
elect the path, select the vitis folder under bram_test project. Click launch.



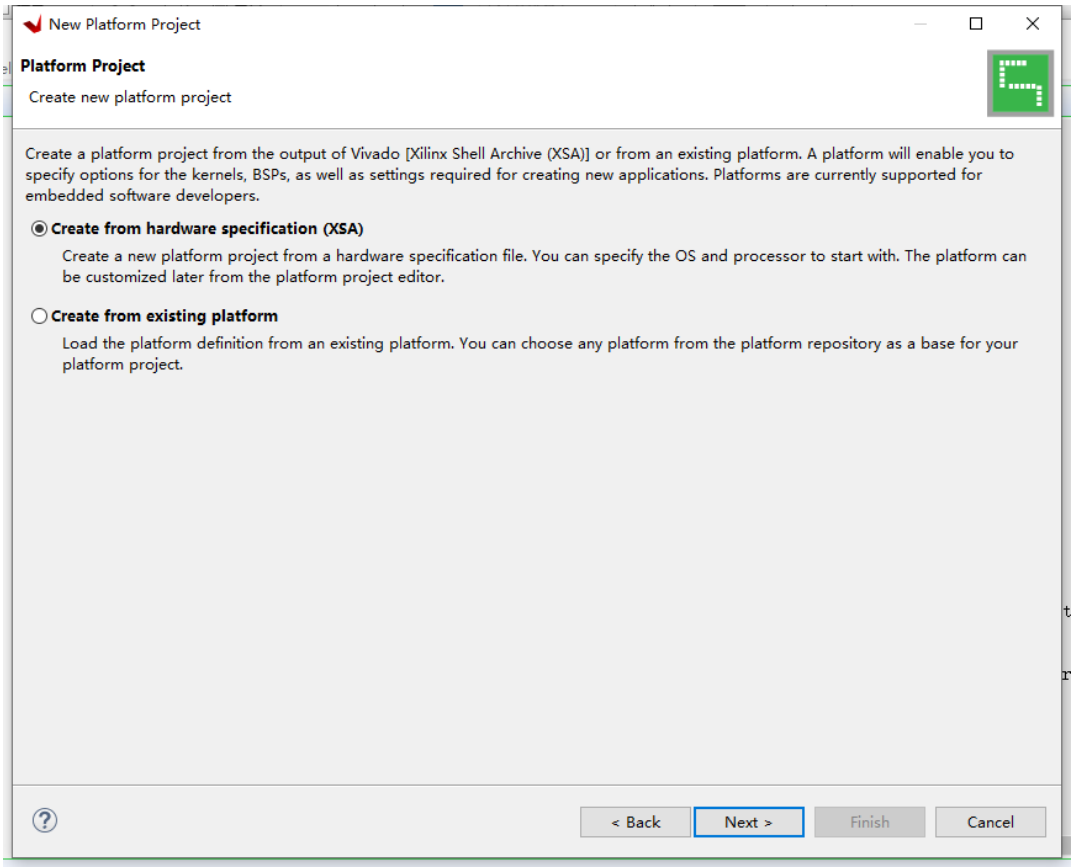
Click Create Platform Project



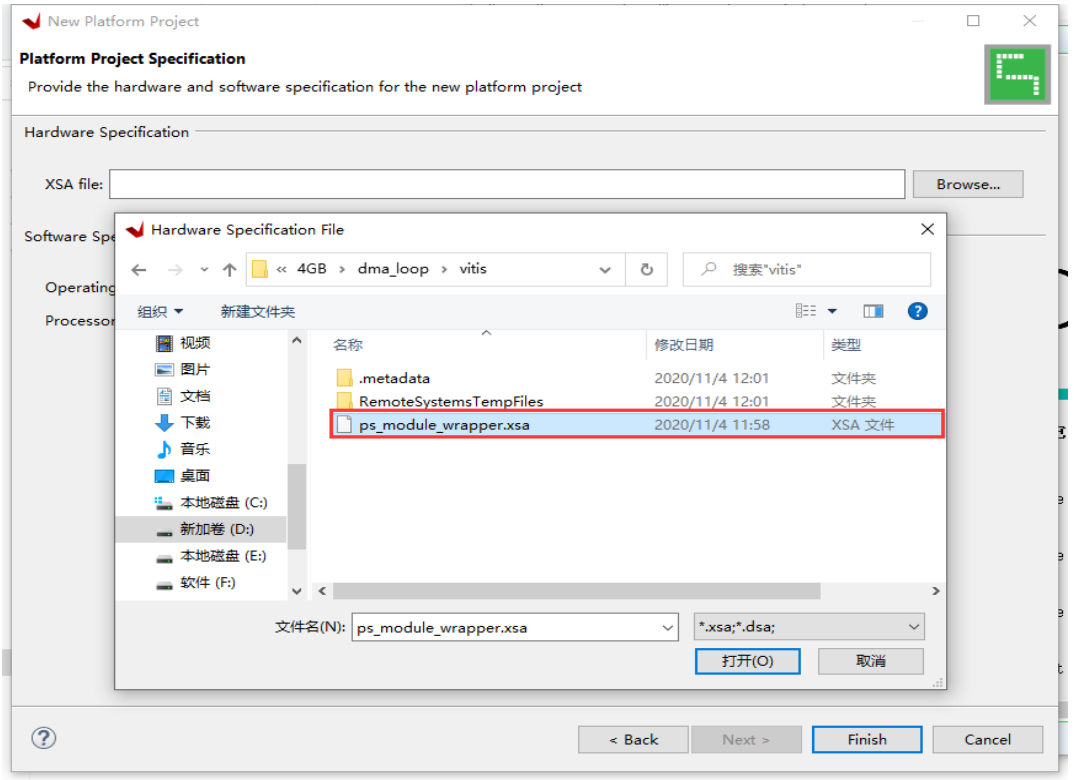
The project name uses the same name as the xsa file. Click next.



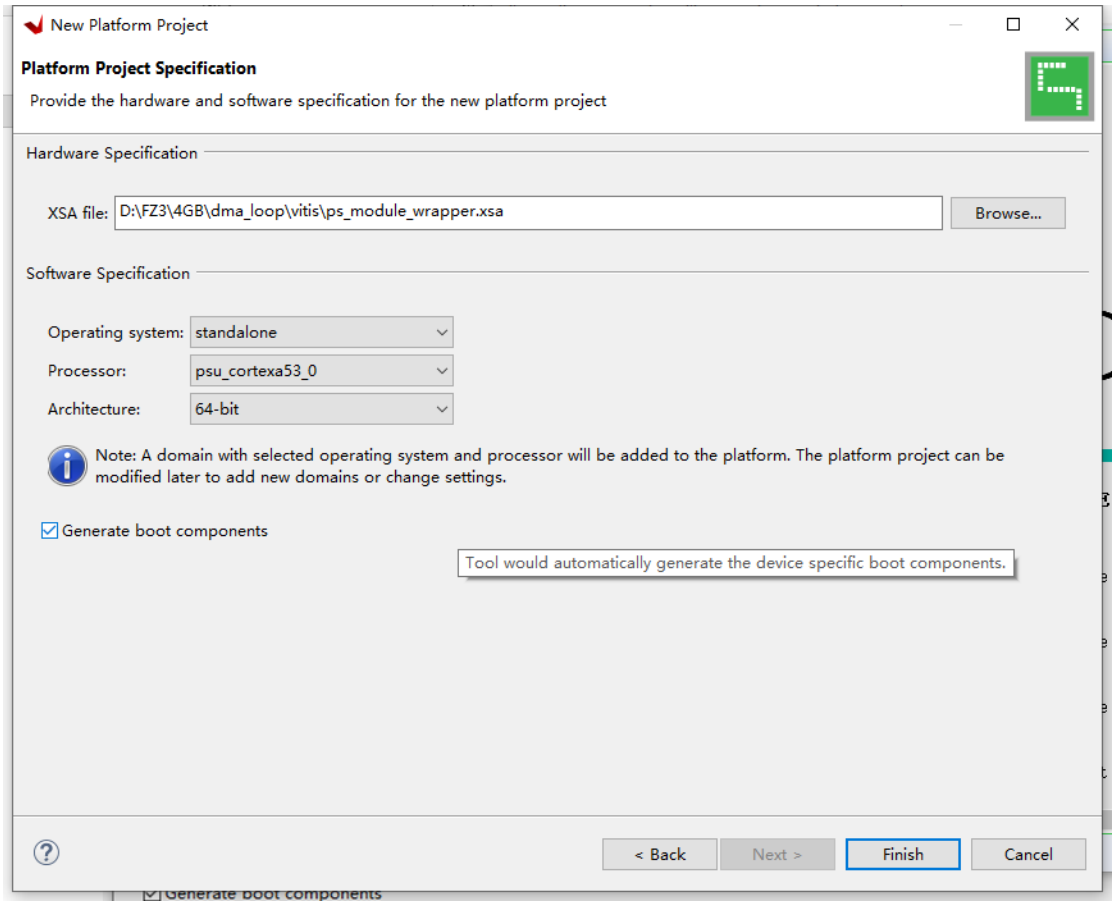
Select Create from hardware specification(XSA),click NEXT.



Open Browse… , select the previously generated xsa file.

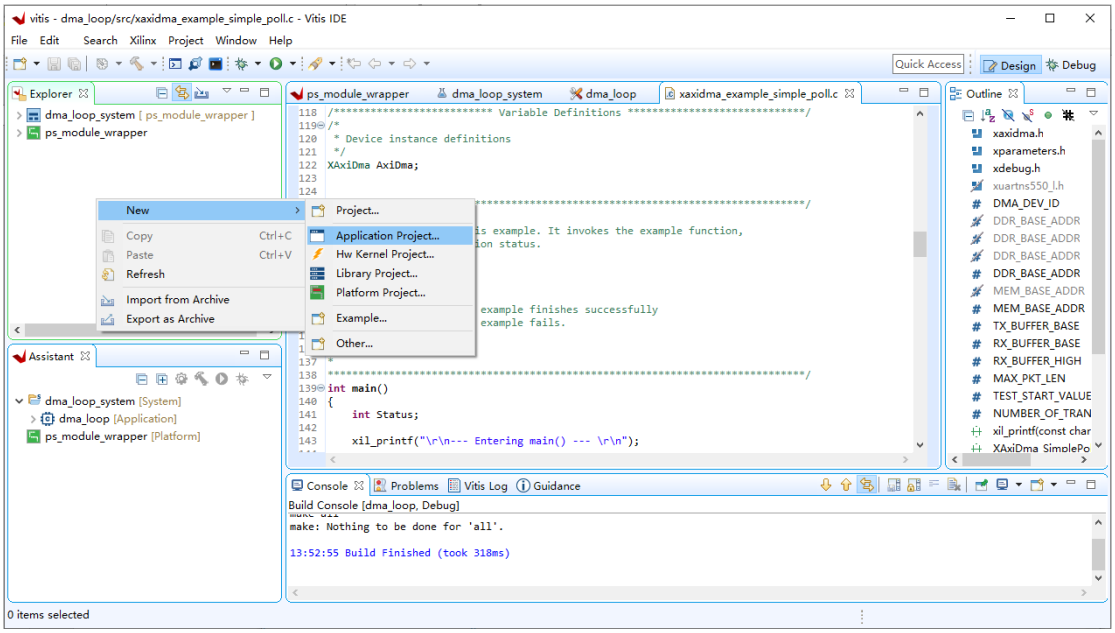


Leave the default and click finish.

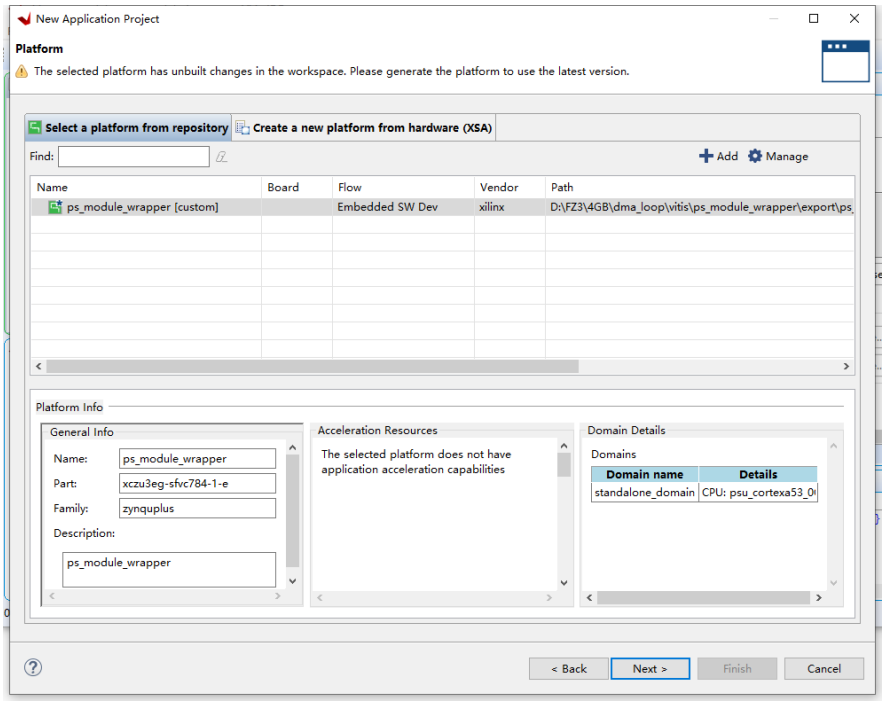


1.8 New dma_loop Project

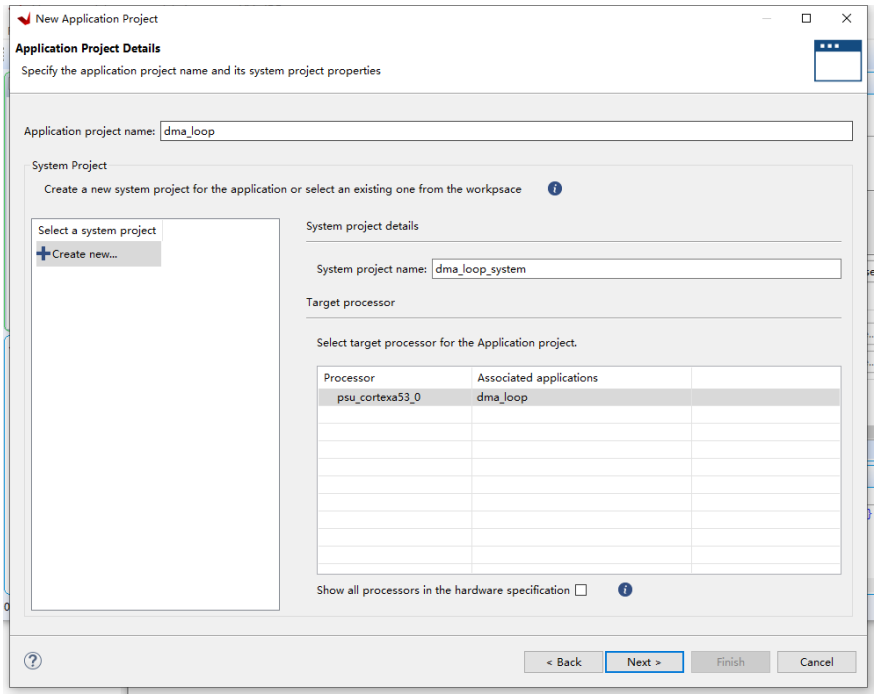
Right click the blank space of the project navigation bar on the left and select NEW→Application Project.



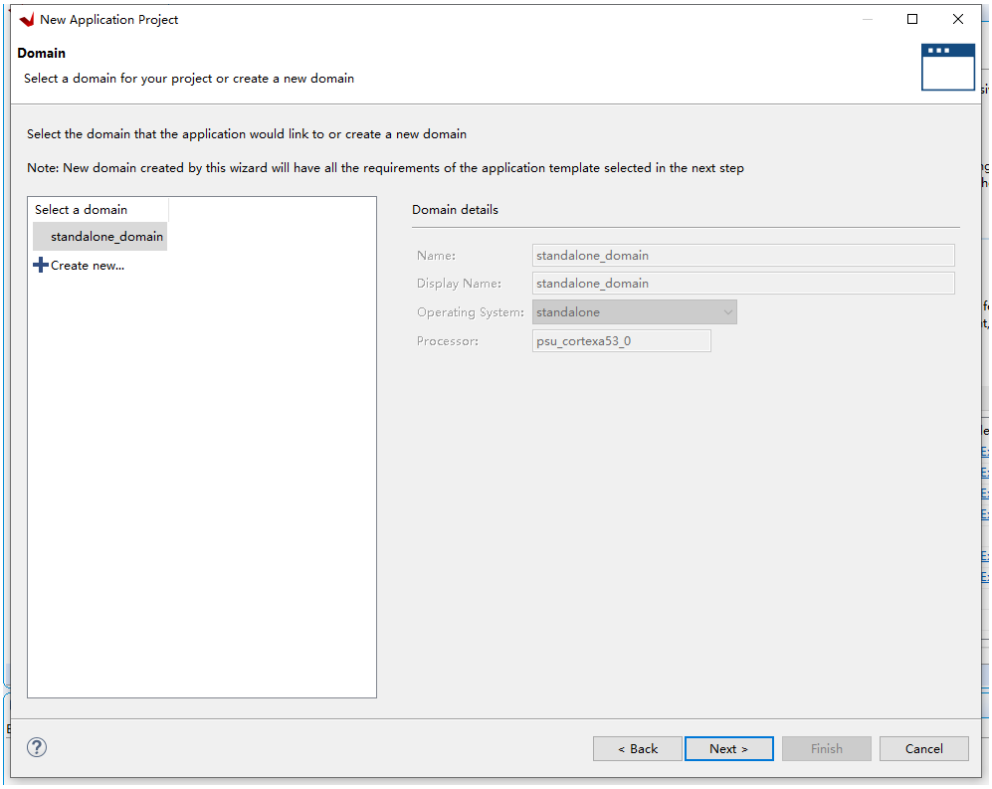
Skip to the second page and select the platform project created above (default), next



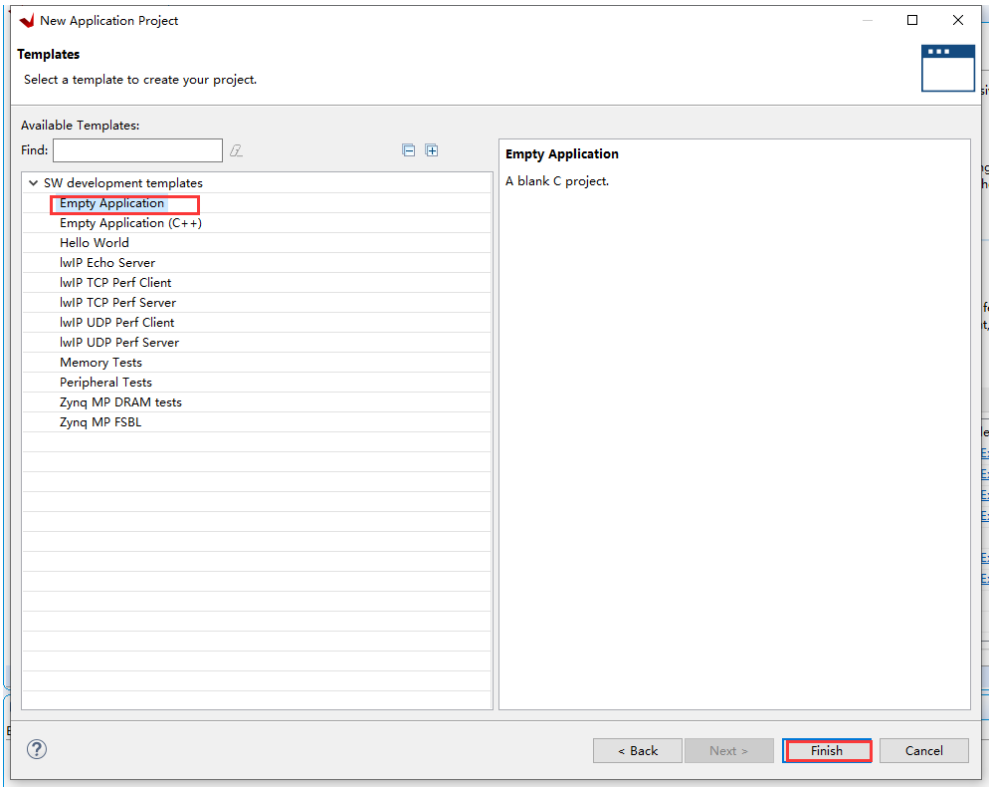
Project name enter dma_loop，Next



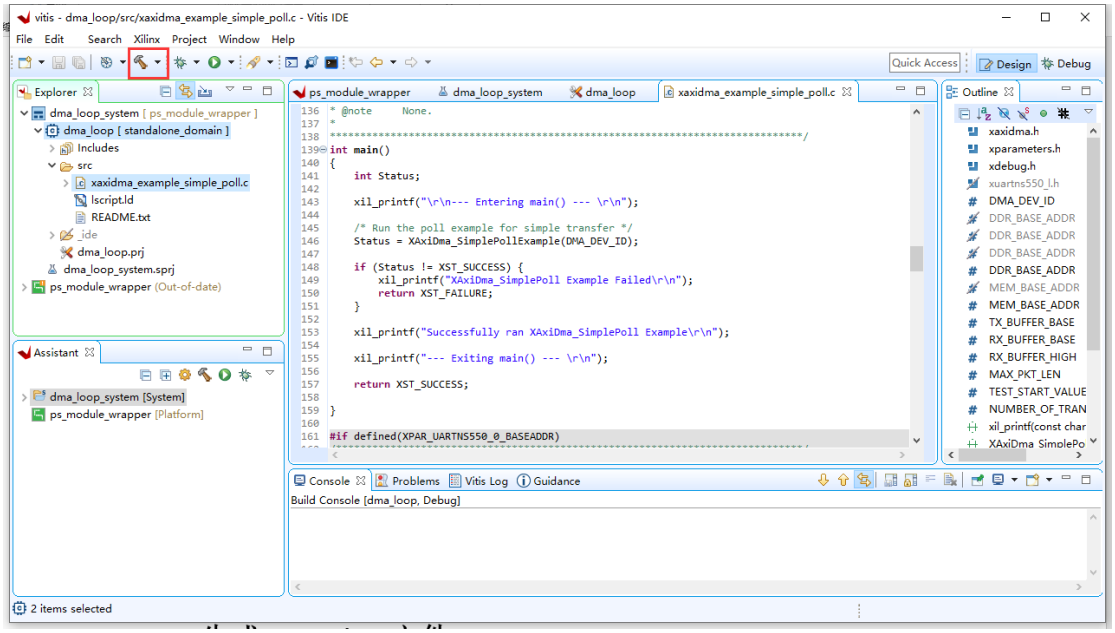
Next



Select Empty Application， Finish

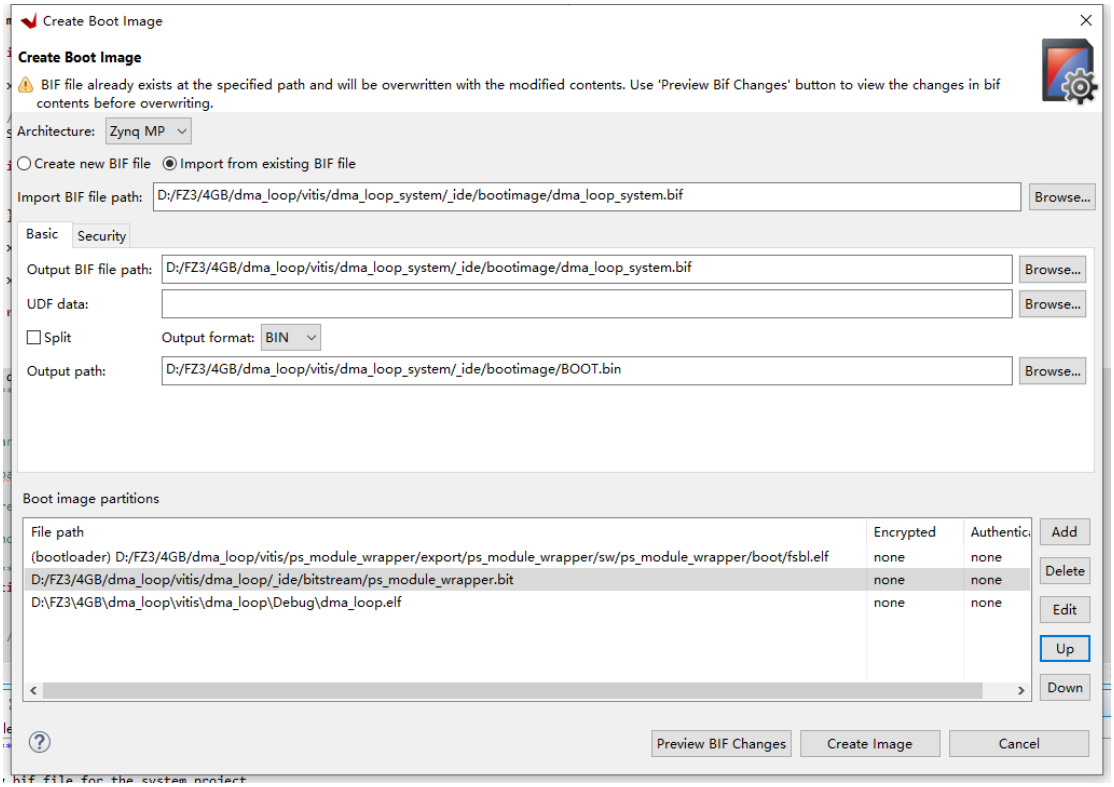


Copy the files in the example project to src, select the project, and click the compile buttom.

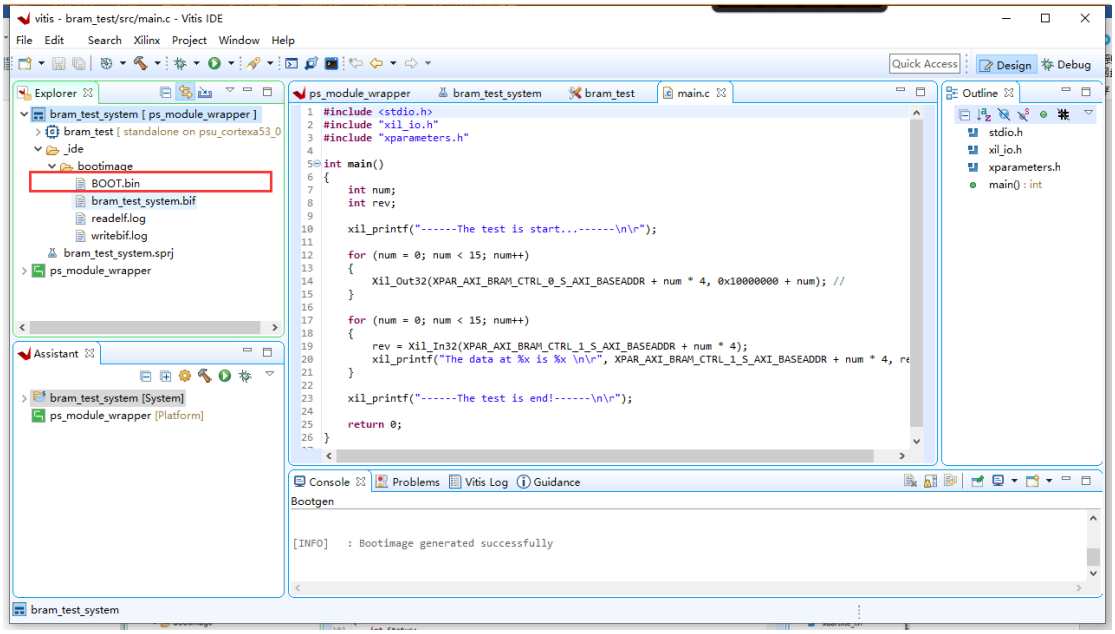


1.9 Generate BOOT.bin file

Right-click the system of APP project and select Create boot image.



Type the development board into SD card startup mode, then copy the BOOT.bin file to SD card and run on the development board.


vitis - bram_test/src/main.c - Vitis IDE
File Edit Search Xilinx Project Window Help
Quick Access
Design
Debug
Explorer
bram_test_system [ps_module_wrapper]
bram_test [standalone on psu_cortexa53_0
_ide
bootimage
BOOT.bin
bram_test_system.bif
readelf.log
writebif.log
bram_test_system.sprj
ps_module_wrapper
Assistant
bram_test_system [System]
ps_module_wrapper [Platform]
ps_module_wrapper
bram_test_system
bram_test
main.c
#include <stdio.h>
#include "xil_io.h"
#include "xparameters.h"
int main()
{
int num;
int rev;
xil_printf("------The test is start...------\n\r");
for (num = 0; num < 15; num++)
{
Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + num * 4, 0x10000000 + num); //
}
for (num = 0; num < 15; num++)
{
rev = Xil_In32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + num * 4);
xil_printf("The data at %x is %x \n\r", XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + num * 4, re
}
xil_printf("------The test is end!------\n\r");
return 0;
}
Outline
stdio.h
xil_io.h
xparameters.h
main() : int
Console
Problems
Vitis Log
Guidance
Bootgen
[INFO] : Bootimage generated successfully

Appendix 1 Warranty & Technical Support Services

MYIR Tech Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR's products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (**support@myirtech.com**), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:
1. Technical support service

a) MYIR offers technical support for the hardware and software materials which have provided to customers;
b) To help customers compile and run the source code we offer;
c) To help customers solve problems occurred during operations if users follow the user manual documents;
d) To judge whether the failure exists;
e) To provide free software upgrading service.

However, the following situations are not included in the scope of our free technical support service:

a) Hardware or software problems occurred during customers' own development;
b) Problems occurred when customers compile or run the OS which is tailored by themselves;
c) Problems occurred during customers' own applications development;
d) Problems occurred during the modification of MYIR's software source code.

2. After-sales maintenance service
The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

a) The warranty period is expired;
b) The customer cannot provide proof-of-purchase or the product has no serial number;
c) The customer has not followed the instruction of the manual which has caused the damage the product;

d) Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads the defects of appearance/function;

e) Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which have caused the damage of the products or defects of appearance;

f) Due to unauthorized weld or dismantle parts or repair the products which has caused the damage of the products or defects of appearance;

g) Due to unauthorized installation of the software, system or incorrect configuration or computer virus which has caused the damage of products.

Warm tips:

1) MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.

2) Please do not use finger nails or hard sharp object to touch the surface of the LCD.

3) MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.

4) Do not clean the surface of the screen with chemicals.

5) Please read through the product user manual before you using MYIR's products.

6) For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR's support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

3. Maintenance period and charges

a) MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.

b) For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.

4. Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

5. Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR's products, like serial port, USB, Ethernet, LCD, etc.
2. MYIR provides the services of OS porting, BSP drivers' development, API software development, etc.
3. MYIR provides other products supporting services like power adapter, LCD panel, etc.
4. ODM/OEM services.

MYIR Tech Limited

Room 1306, Wensheng Center, Wenjin Plaza,

North Wenjin Road, Luohu District, Shenzhen, China 518020

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836
Fax: +86-755-25532724

Website: www.myirtech.com