APRIL 2023, 4 BY ACKYE

Smart ZYNQ (SP&SL Version) Project 7 Use ZYNQ's PS to light up the LED light EMIO method connected to the PL end (Recommended use)

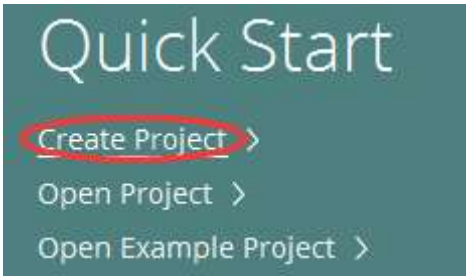
The previous project 6 uses AXI GPIO to let PS control the LED light on the PL side (equivalent to the PL side needs to generate the corresponding circuit of AXI GPIO, occupying resources), this article uses EMIO to map the LED on the PL end to the GPIO on the PS side. This method is more commonly used in the project, **(the resource occupation of the PL side is very small, it can be simply understood as the PS end pulls a wire to the corresponding IO port of the PL side)** In addition to GPIO can EMIO mapping, SPI I2C UART can be mapped in the same way, which greatly increases the scalability of the system

This article is demonstrated on vivado2018.3, please research for other versions

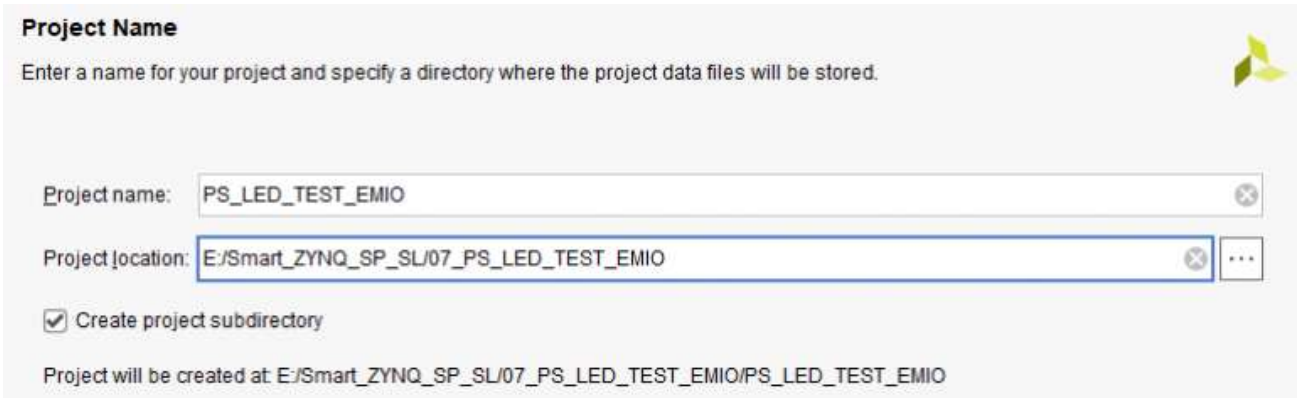
(Note: The content of this section applies to the boards of Smart ZYNQ SP and SL Edition, if it is Smart ZYNQ Standard Edition, please refer to the corresponding board directory)

1. Create a Vivado project

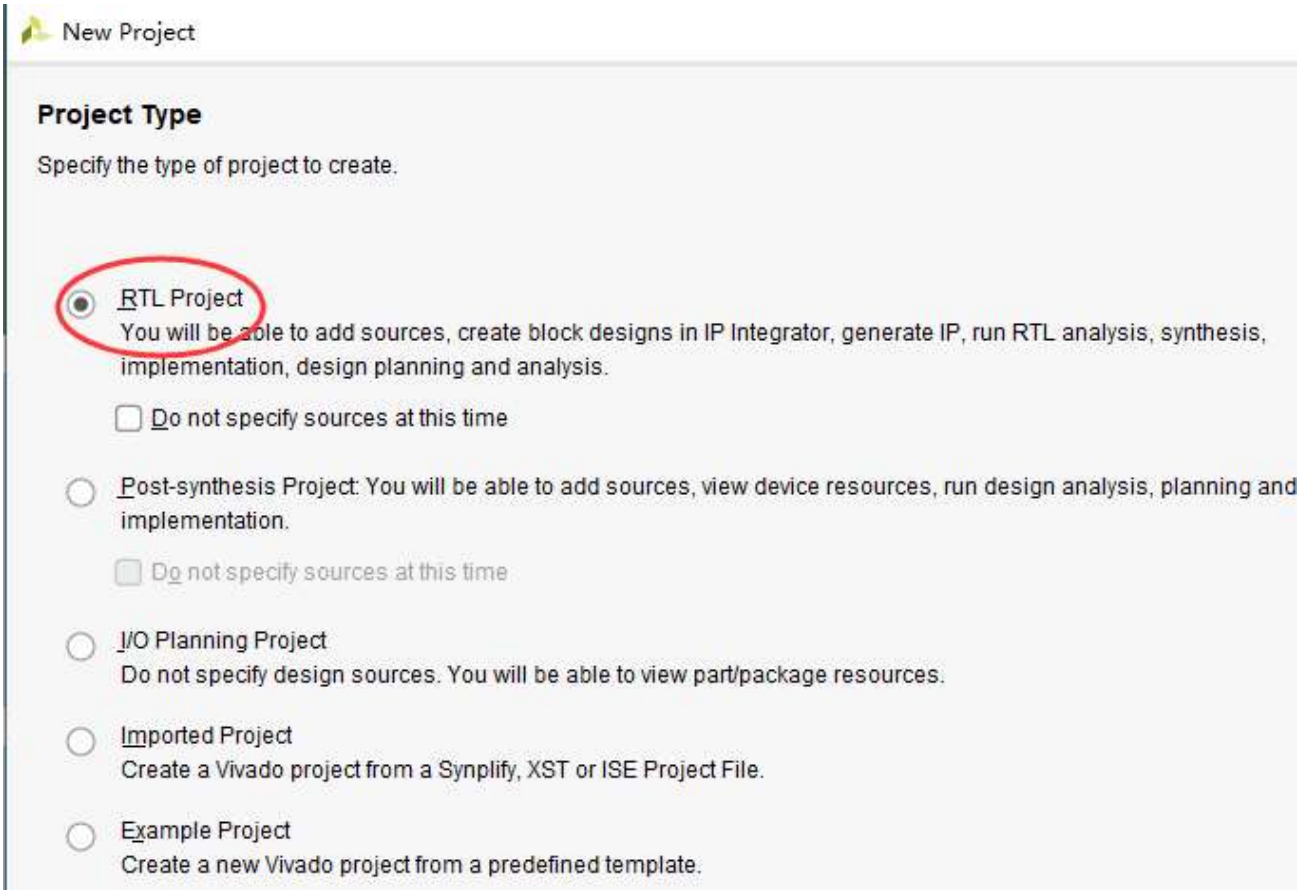
1) Specific steps to create a VIVADO project, open the software and select Create Project, as shown in the following figure



2) Click NEXT and enter the project name in the second dialog box "Project name" that appears; Select the save path in Project location; Check "Create project subdirectory" and click "Next" **Note, all paths cannot appear Chinese name**



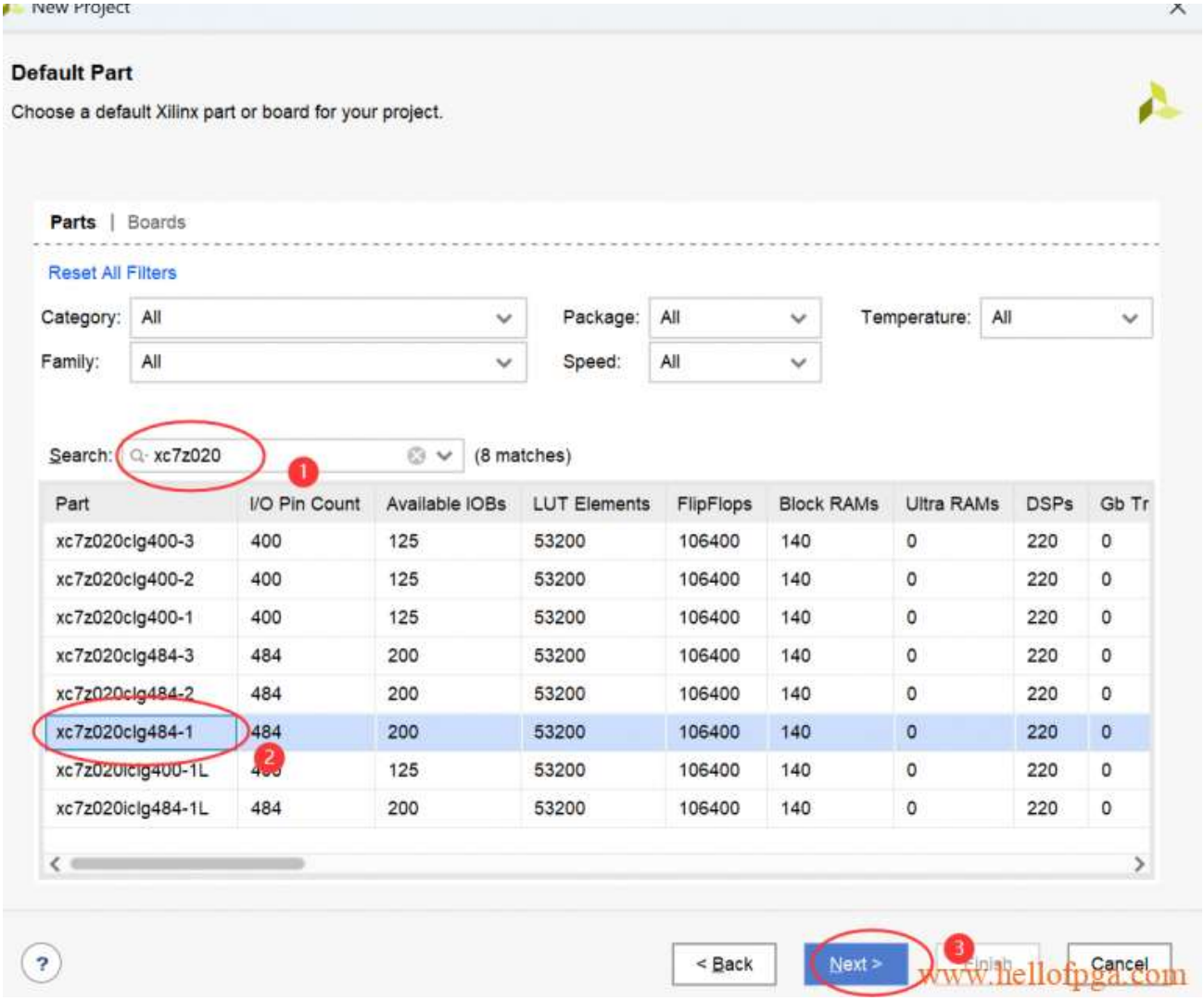
3) Click the RTL PROJECT option and click NEXT



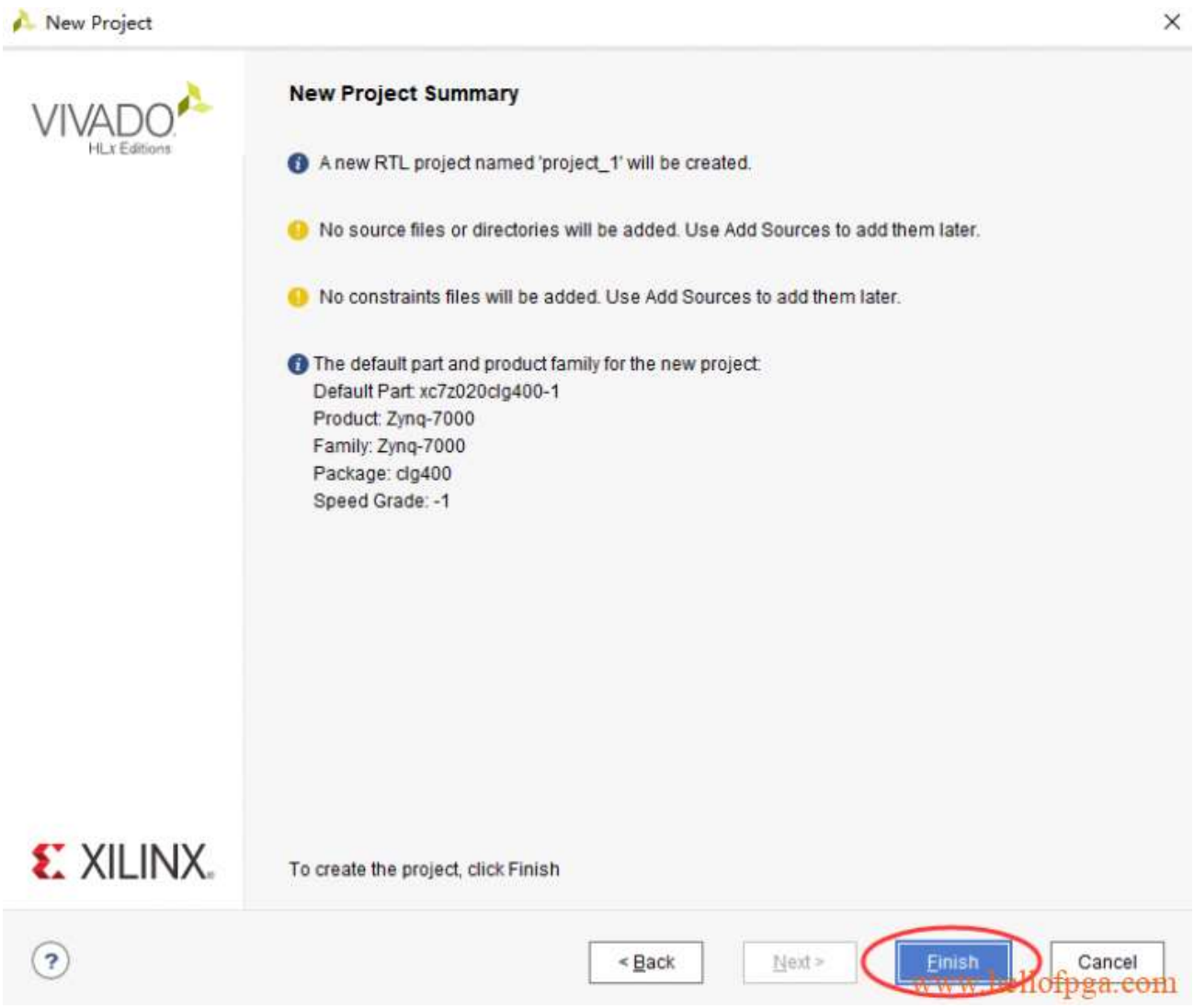
4) Step <>: The Add Sources option is left blank, NEXT

5) Step <>: The Add Constraints option is left blank, NEXT

6) Select the chip model XC7Z020CLG484-1

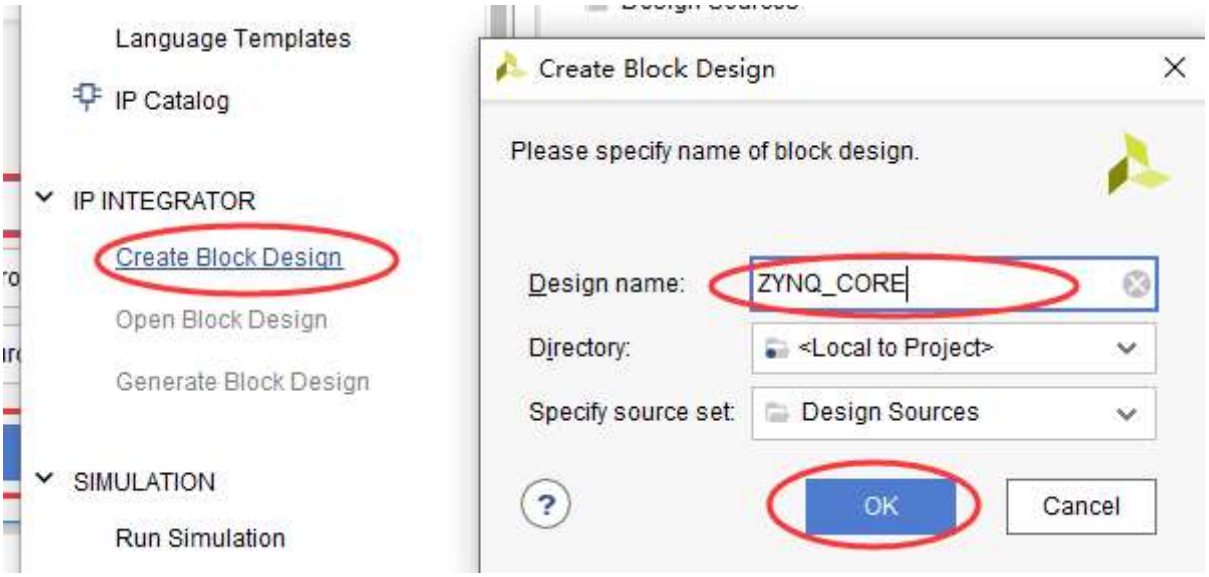


7）确认所选信息 点击“Finish”，完成vivado的工程创建

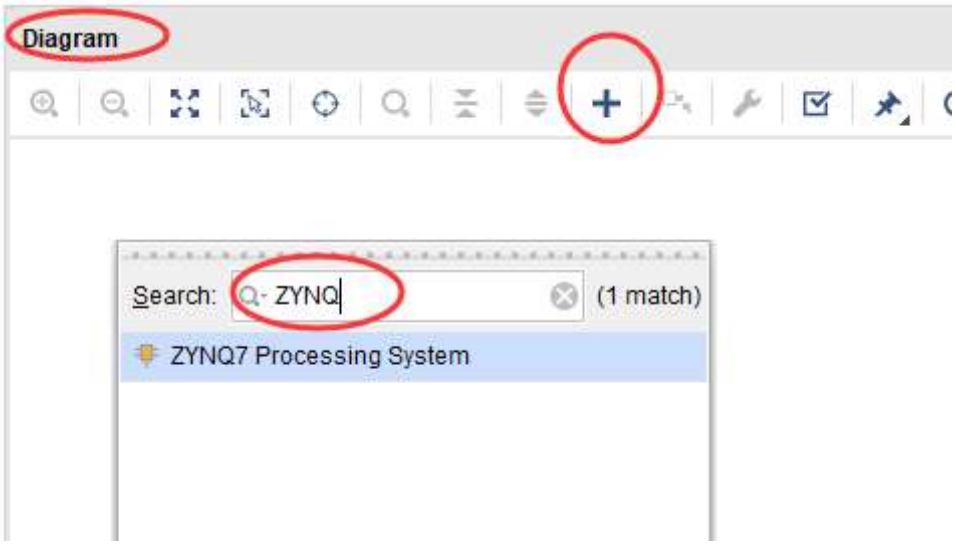


2 Create a BLOCK design

1) IP INTEGRATOR→Create Block Design, enter the design name in the pop-up dialog box, and finally click "OK", as shown in the figure below

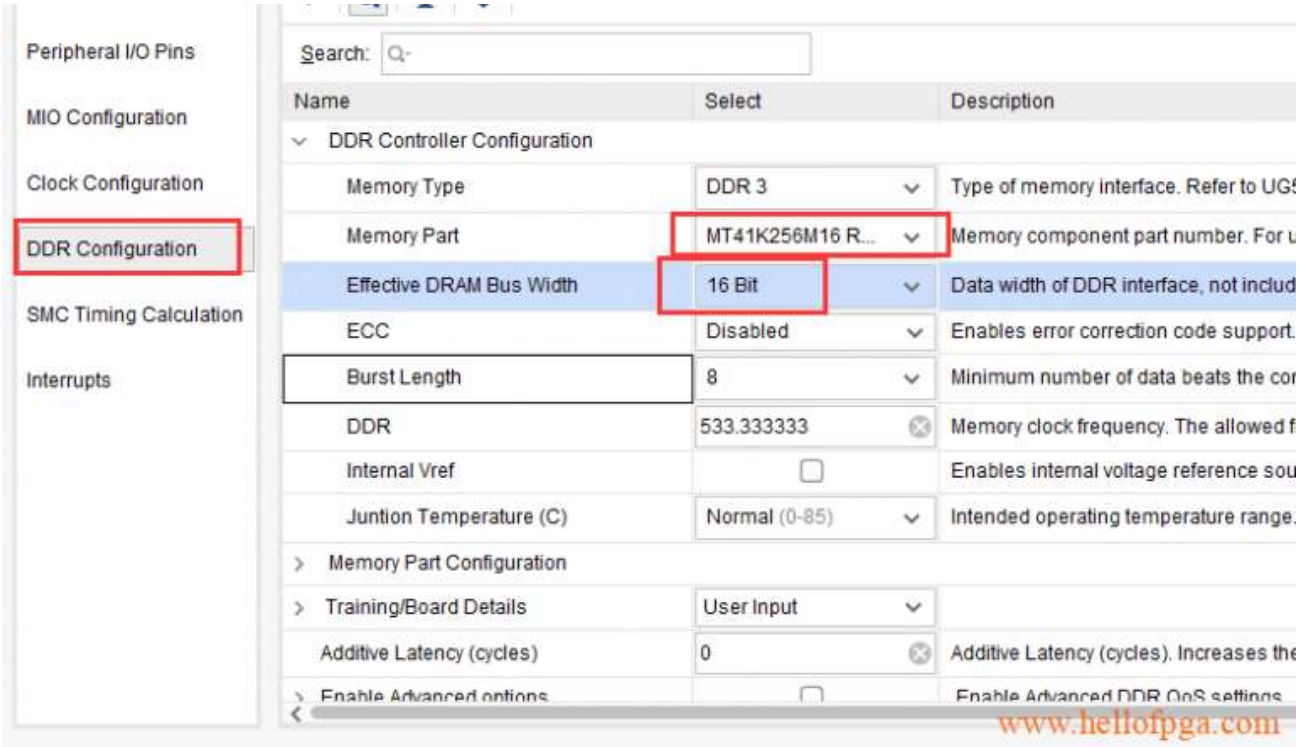


2) IN THE WINDOW ON THE RIGHT, CLICK THE PLUS SIGN, SEARCH FOR ZYNQ IN THE SELECTION BOX, AND FIND ZYNQ7 PROCESSING SYSTEM, DOUBLE-CLICK AND OPEN

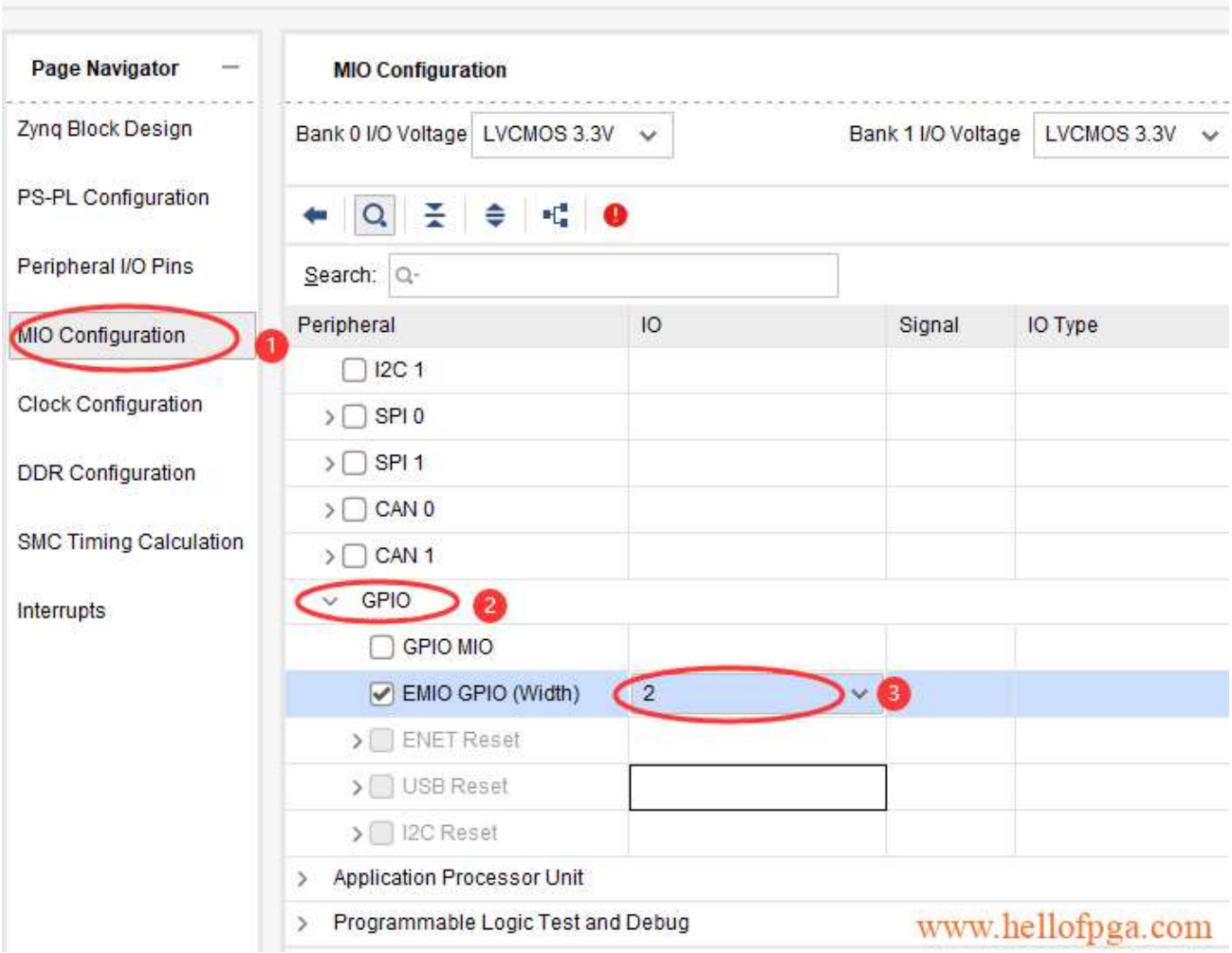


3) The software automatically generates a zynq block as shown in the figure below, next to do some corresponding settings, double-click the ZYNQ core in the figure below

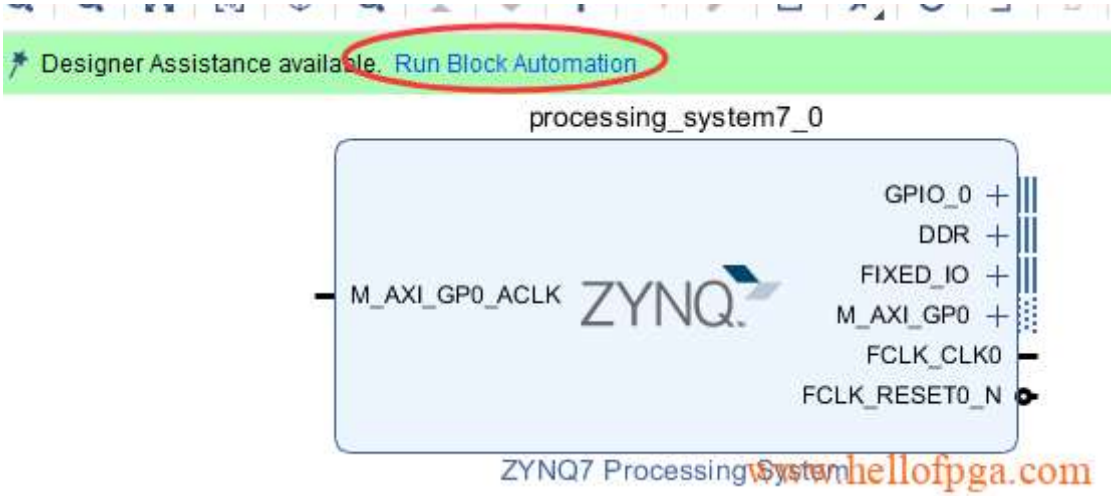
4) Find DDR Configuration →DDR Controller Configuration →DDR3 in the pop-up window in turn, select the corresponding DDR3 according to the DDR on your board in the Memory Part drop-down menu, the model used in this experiment: MT41K256M16RE-125, select 16bit of data width and finally click "OK", as shown in the figure below.



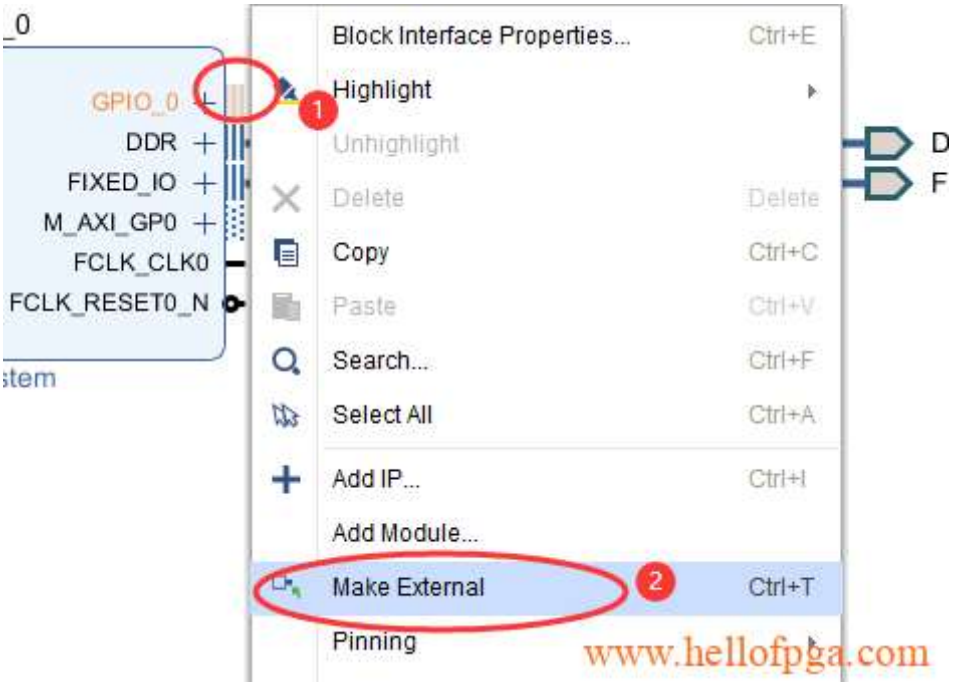
5) **Key step Different from the sixth chapter, in the GPIO column of the MIO configuration option of PS, add two EMIOs (because this test is two, if you need to add buttons or other IO here can be adjusted accordingly).**



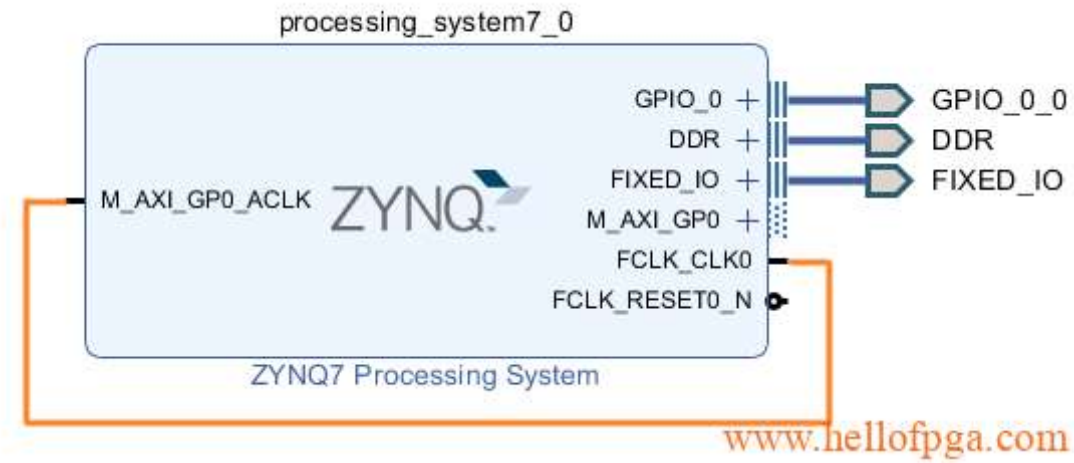
6) Finally, click "Run Block Automation" as shown in the image below. Keep the default in the pop-up options and click "OK" to complete the configuration of the ZYNQ7 Processing System



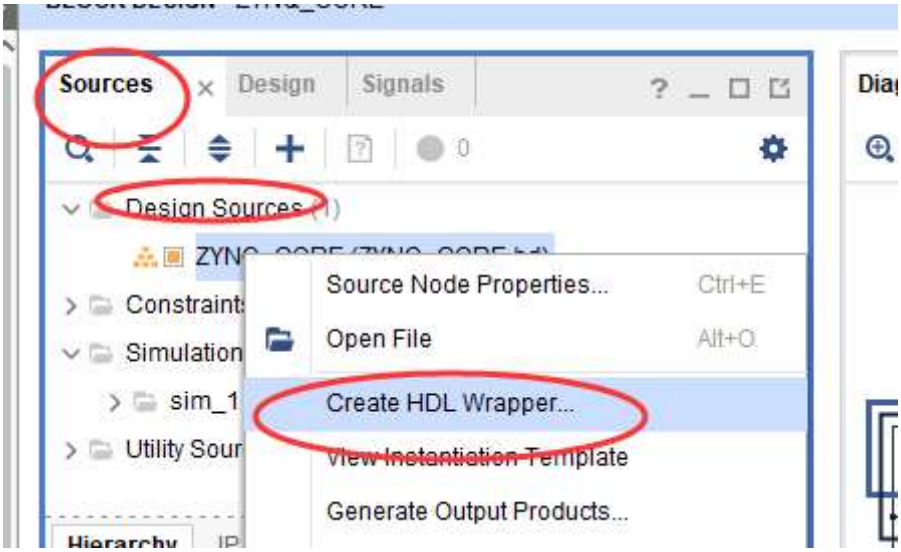
7) Right-click the GPIO_0—>Make External



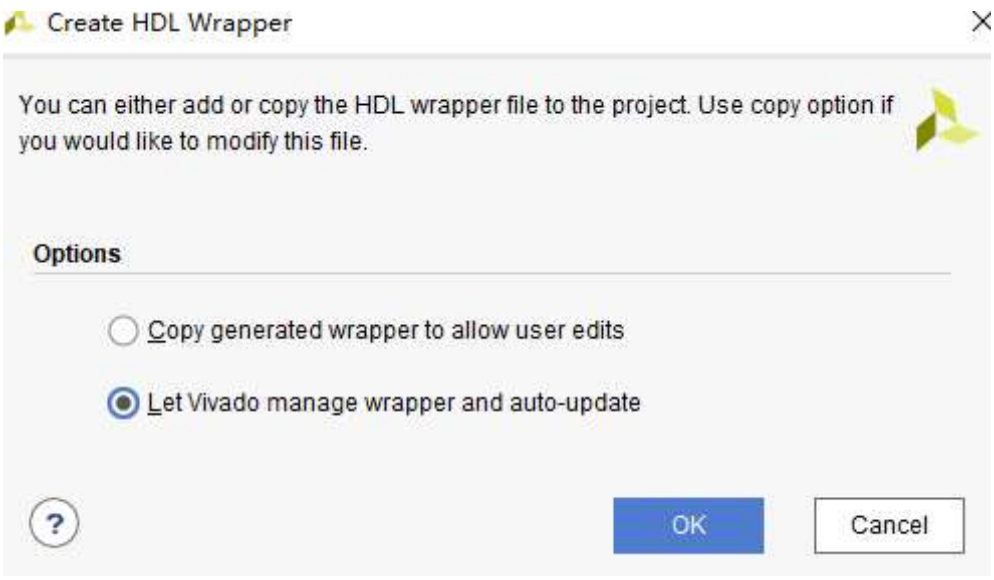
8) Connect the M_AXI_GP0_ACLK with the FCLK_CLK0 with a wire



9) source→ Design Source, right-click on the BLOCK project we created, and click create HDL wrapper as shown in the figure below.



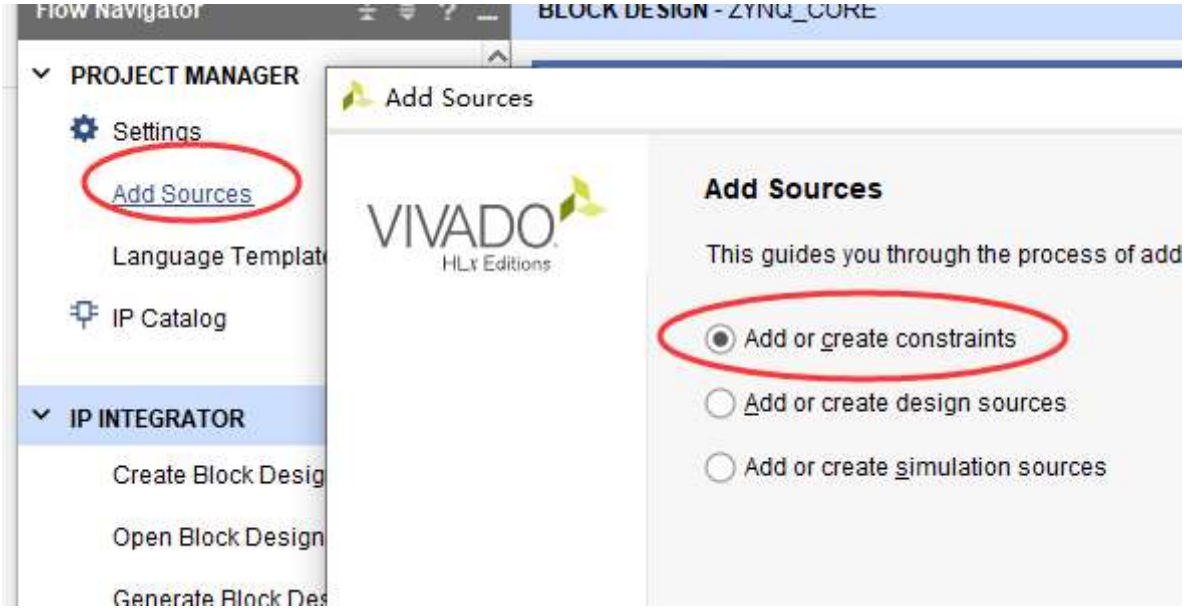
Keep the default in the pop-up dialog



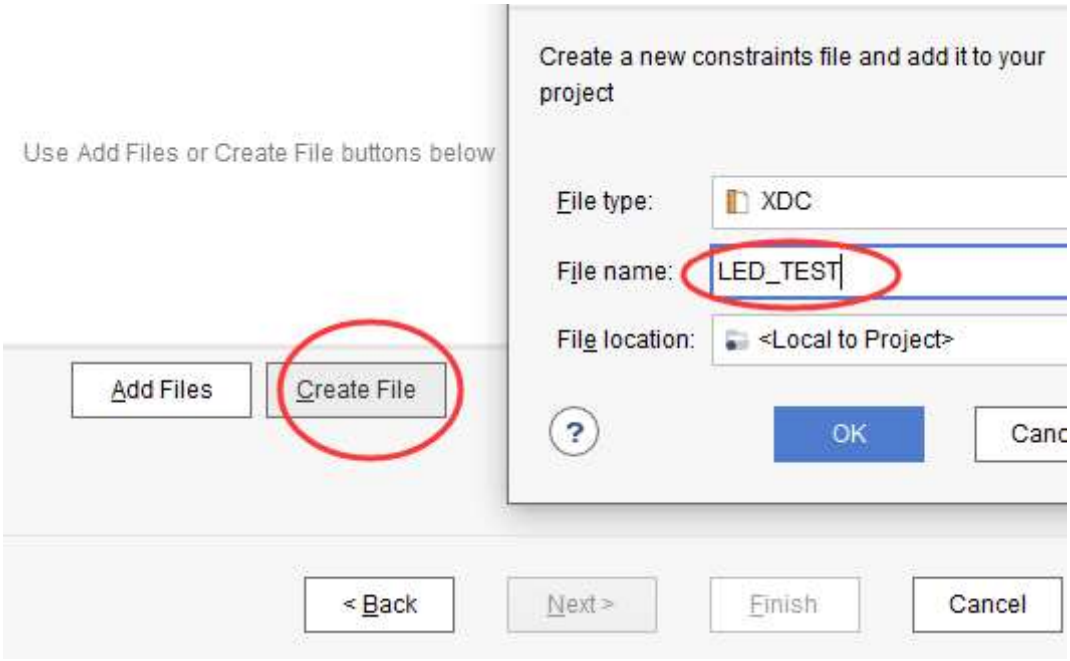
The software automatically generates HDL files for us

3. Create a constraint file and define the pins

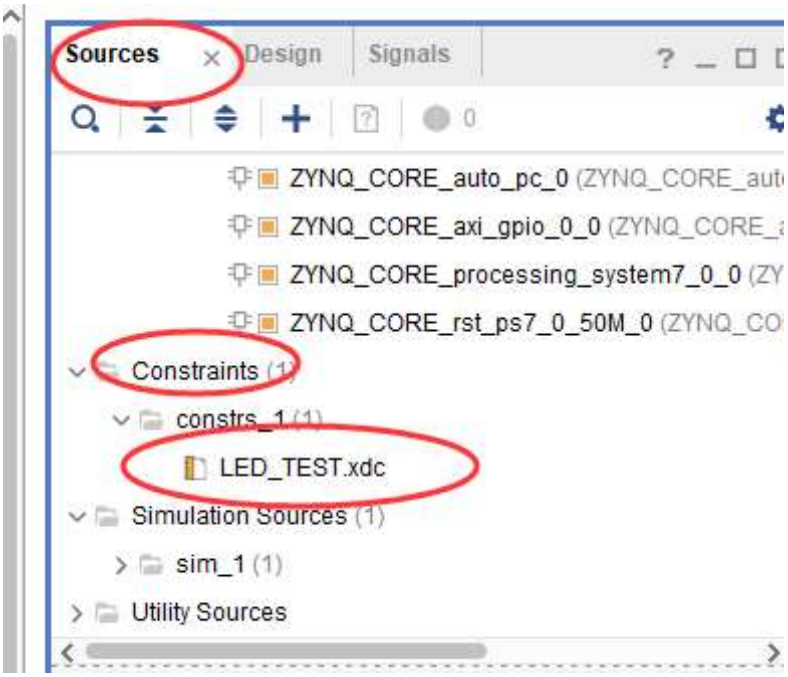
1）Add Source → Add or create constraints 点Next



Since no constraint file has been created for this project, create a constraint file here, set the name of the constraint file in the file name, and click FINISH to complete the creation of the constraint file



2）Sources → Constraints 里找到刚才创建的约束文件 双击并打开该XDC约束文件



Copy the following code in the constraint file to pin the output GPIO (all pin adapter boards have actual annotations corresponding IO)

```
set_property IOSTANDARD LVCMOS33 [get_ports GPIO_0_0_tri_io[0]]
set_property IOSTANDARD LVCMOS33 [get_ports GPIO_0_0_tri_io[1]]

set_property PACKAGE_PIN P20 [get_ports GPIO_0_0_tri_io[0]]
set_property PACKAGE_PIN P21 [get_ports GPIO_0_0_tri_io[1]]
```

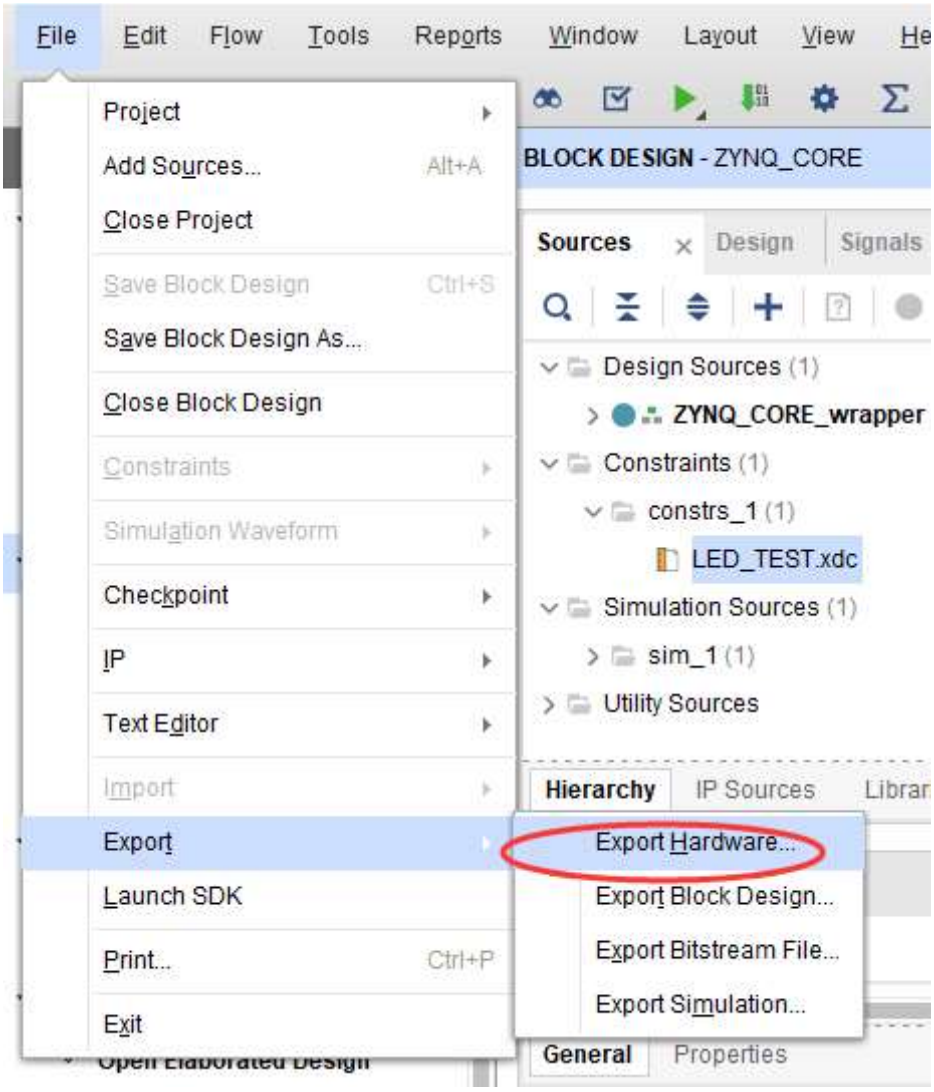
4. Generate a bit file

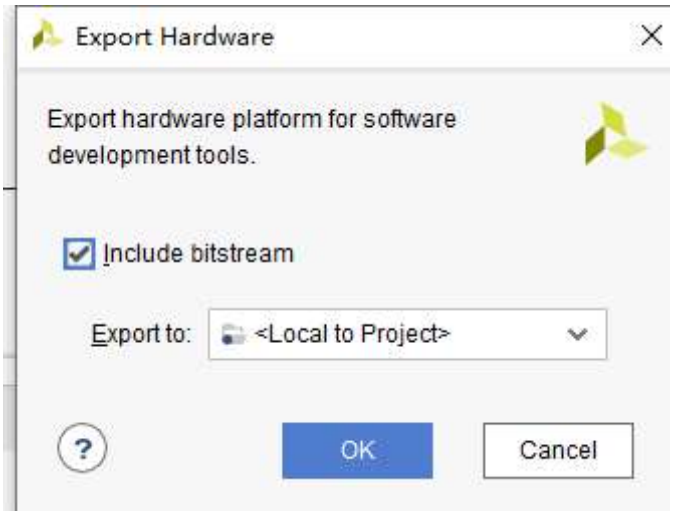
Press the green arrow to compile the project

Press Generate Bitstream to complete synthesis and generate the bit file

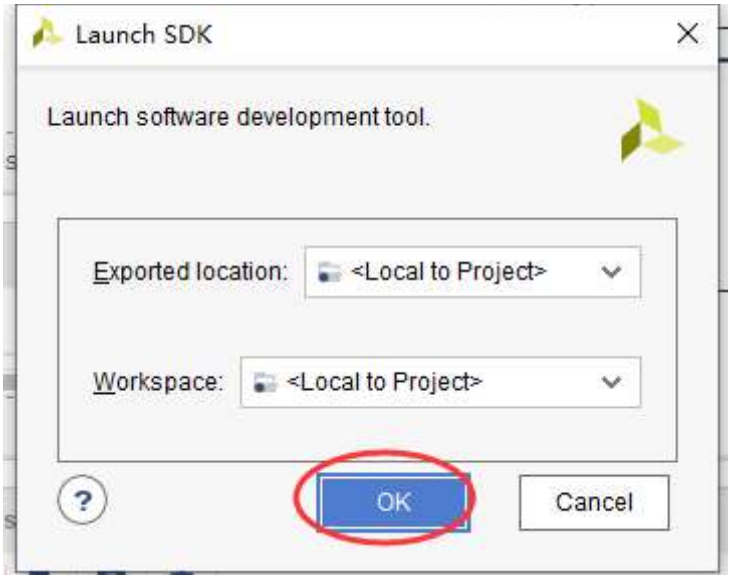
5. SDK program writing

1) File→Export→Export hardware..., check "include bitstream" in the pop-up dialog box, and click "OK" to confirm, as shown in the figure below.

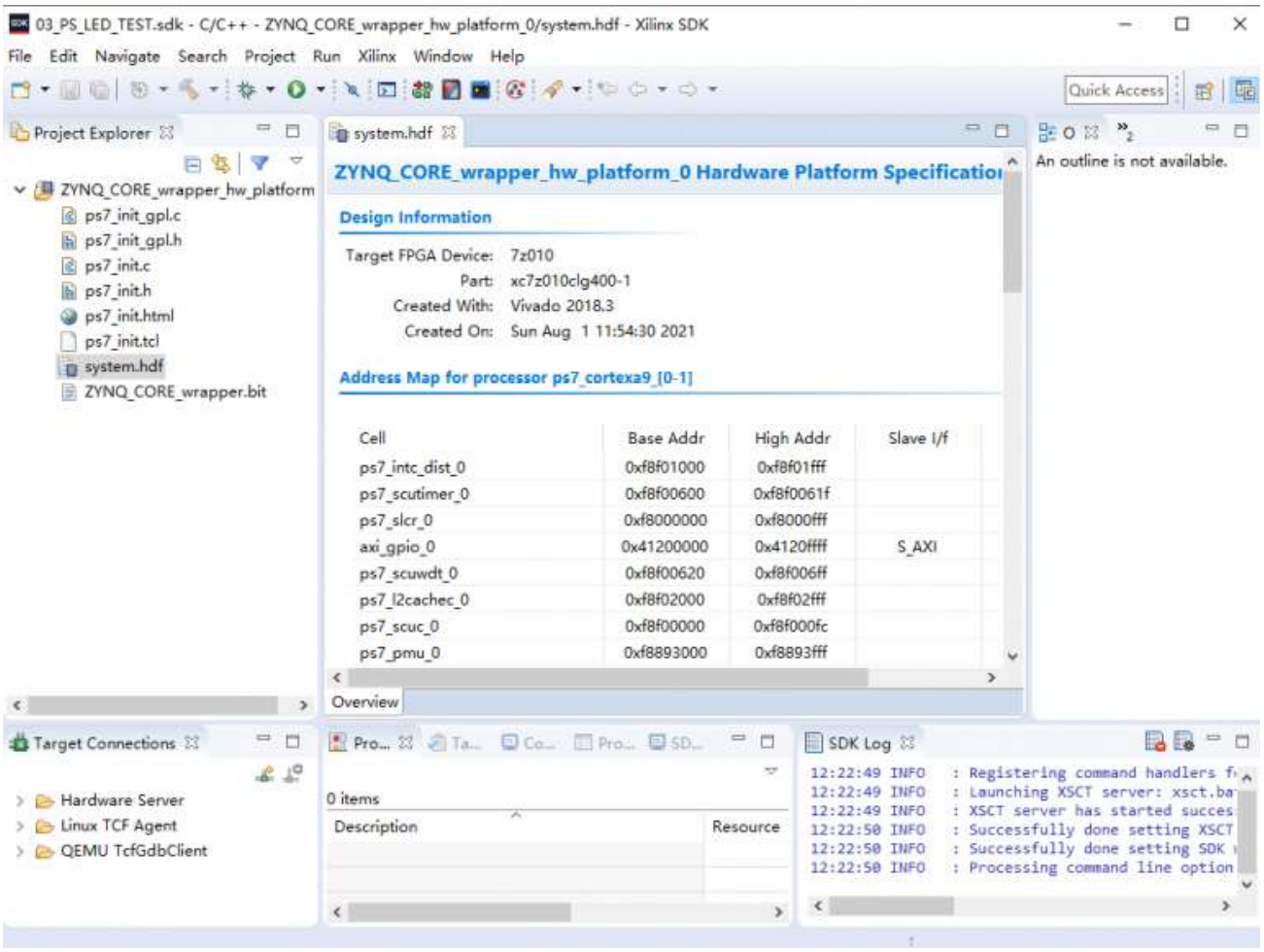




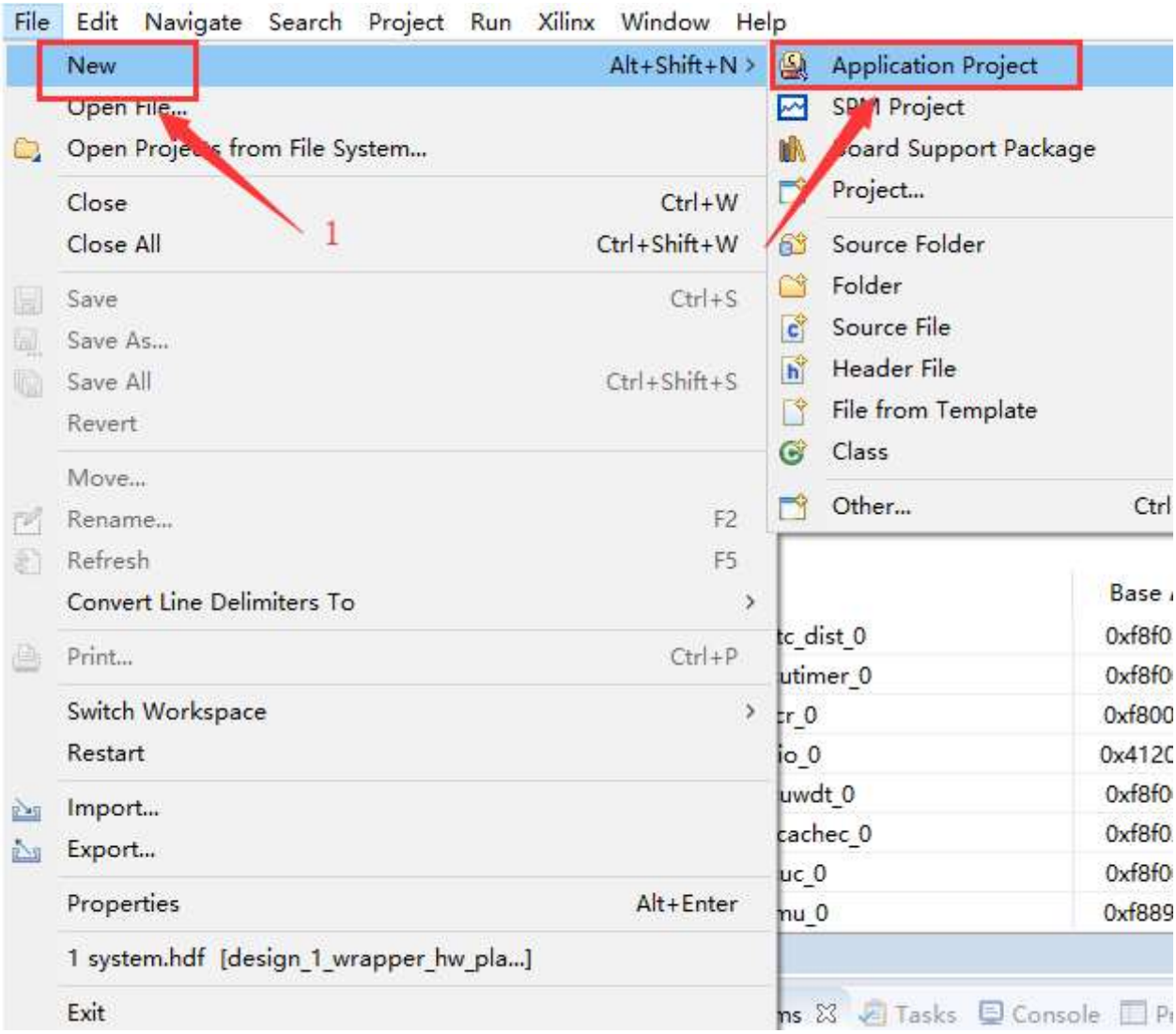
2) File→ Lauch SDK, in the pop-up dialog box, save the default, click "OK", as shown in the figure below.



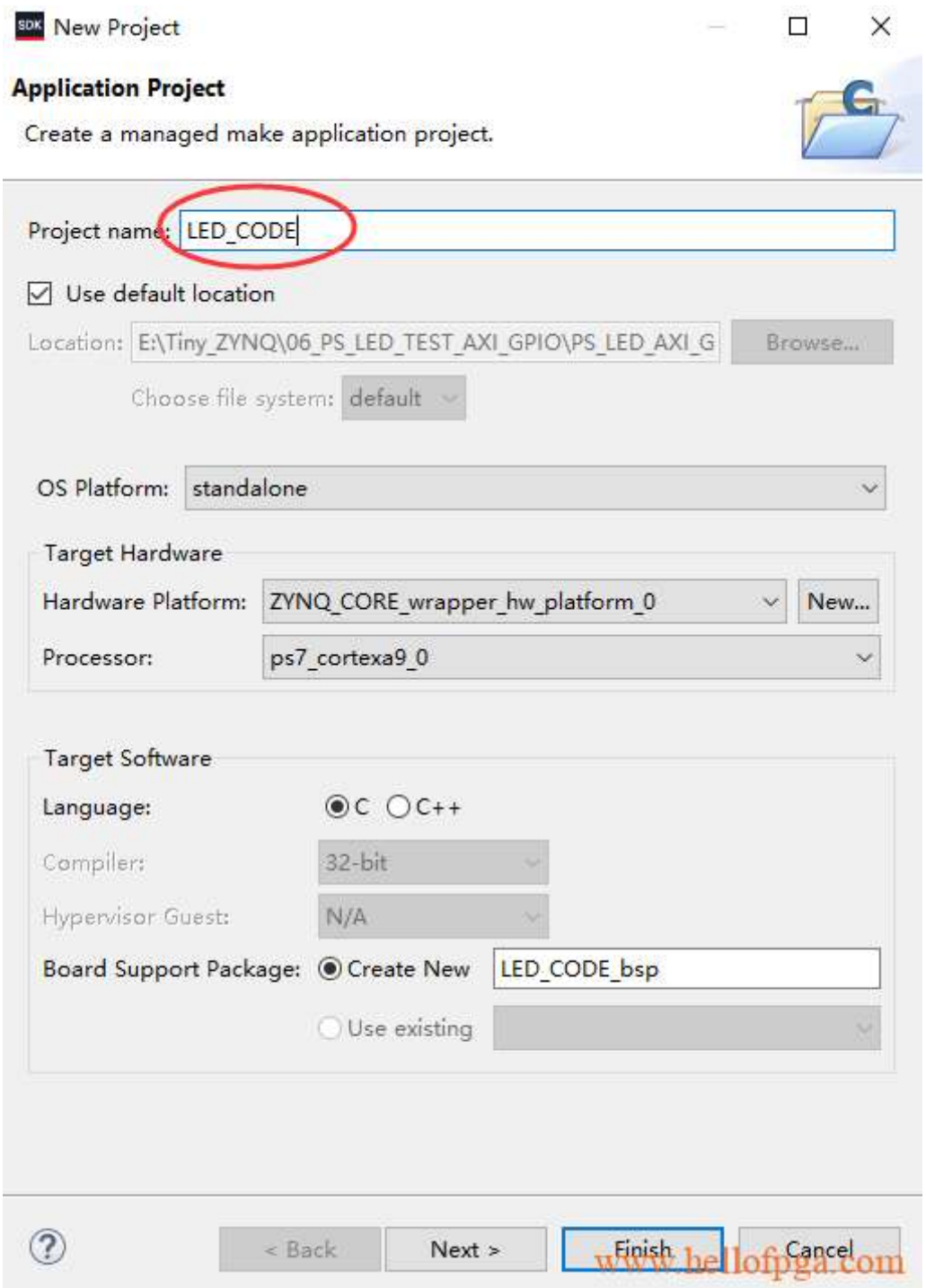
The SDK development environment will automatically open



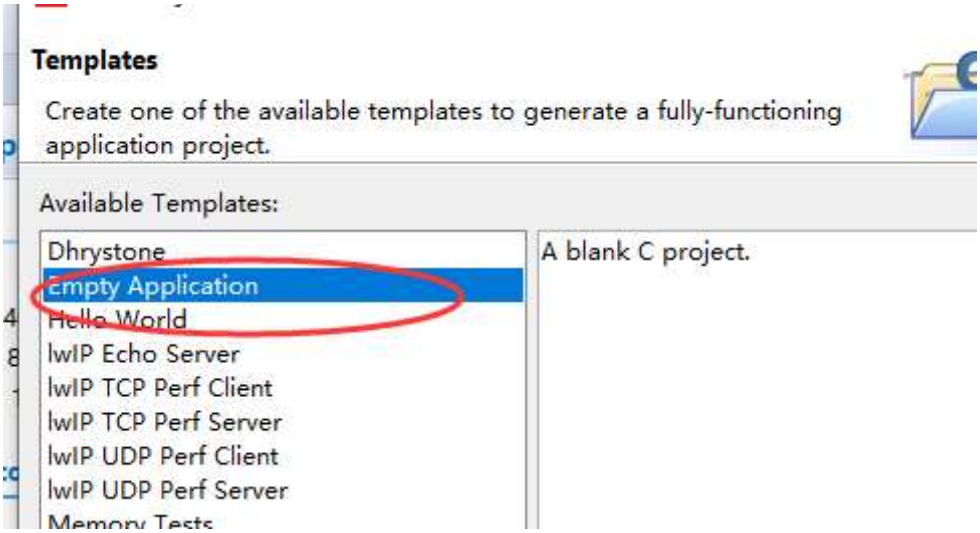
3) Create a new project file→new→Application Project to create an "Application Project", as shown in the figure below.



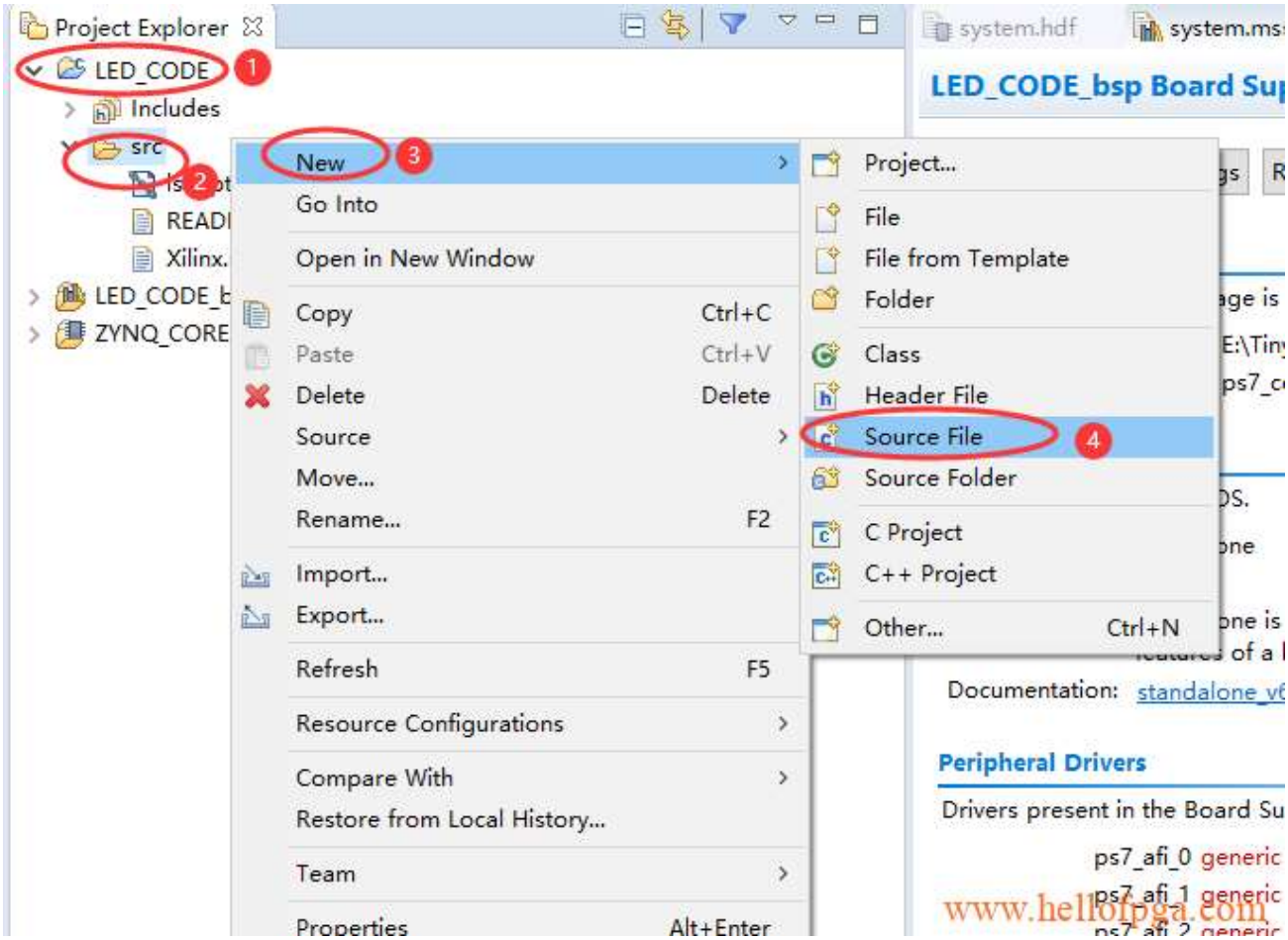
4) Enter your own project name in New Project Name and click NEXT



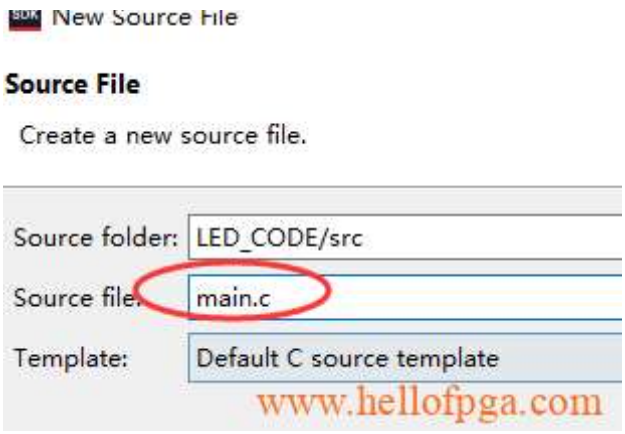
5) Select the empty project and click Finish



6) Add main.c file to the project src—>New—>Source File as shown in the figure below



7）在弹出的窗口中填入main.c 并且保存



8). Open the main.c you just created

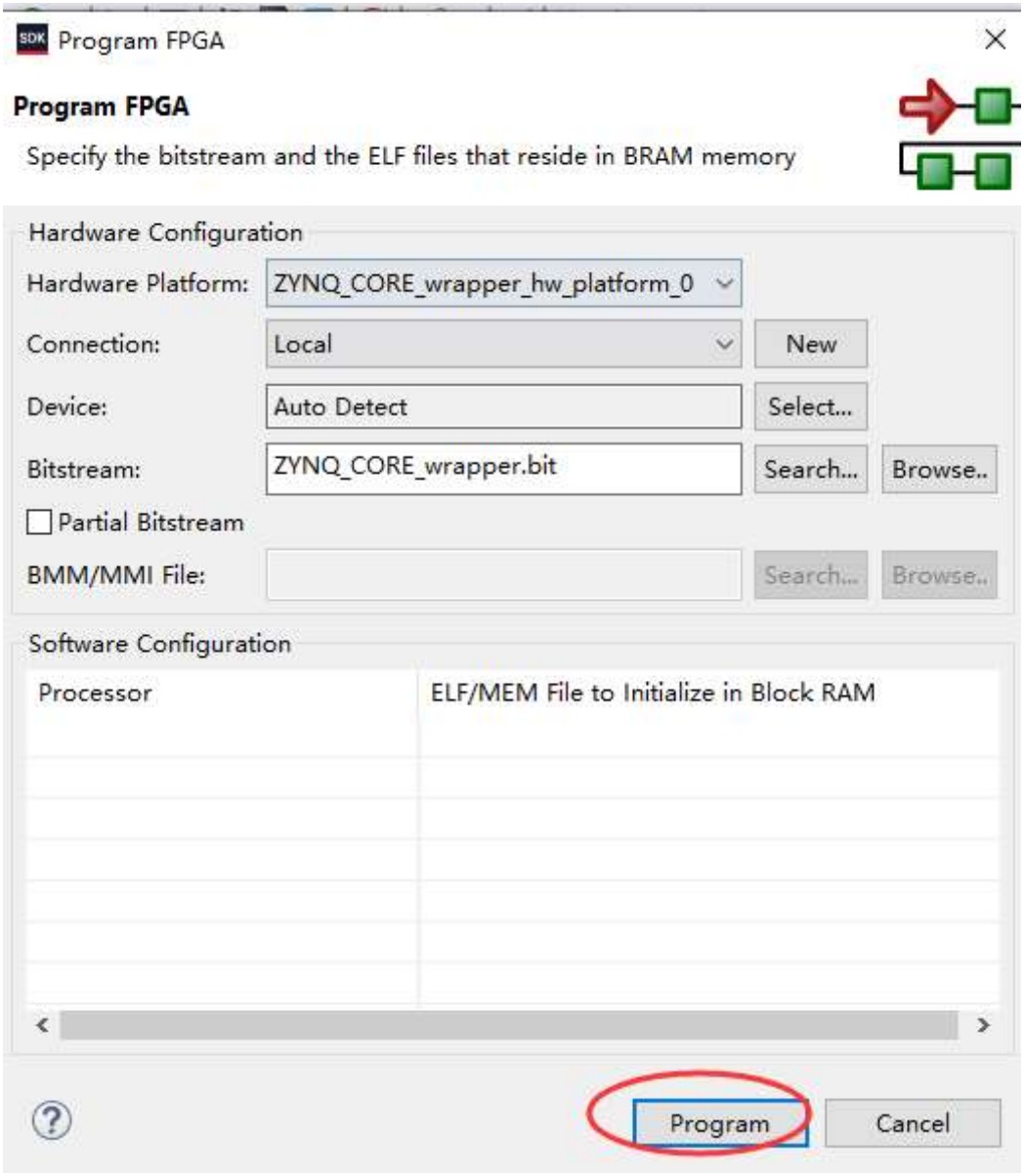
Then write the following code (the code is condensed on the basis of routines) There is a place worth noting that the IO port number of EMIO starts from 54, that is, the EMIO port created under my VIVADO, and the PS side is sorted from 54-55-56 (tips, less than 54 is MIO, that is, the hardware IO port of chip PS)

```
#include "xparameters.h"
#include "xgpiops.h"
#include "xstatus.h"
#include "xplatform_info.h"

#define LED1            54
#define LED2            55

#define GPIO_DEVICE_ID          XPAR_XGPIOPS_0_DEVICE_ID
XGpioPs Gpio;


void Gpio_Init(void){
        XGpioPs_Config *ConfigPtr;

        ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
        XGpioPs_CfgInitialize(&Gpio, ConfigPtr,ConfigPtr-
>BaseAddr);

        XGpioPs_SetDirectionPin(&Gpio, LED1, 1);
        XGpioPs_SetOutputEnablePin(&Gpio, LED1, 1);

        XGpioPs_SetDirectionPin(&Gpio, LED2, 1);
        XGpioPs_SetOutputEnablePin(&Gpio, LED2, 1);
```

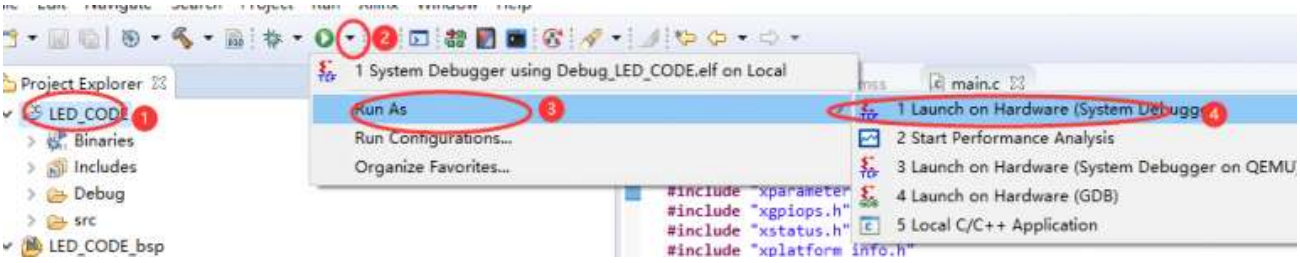
```
        XGpioPs_WritePin(&Gpio, LED1, 0);
        XGpioPs_WritePin(&Gpio, LED2, 0);
}


#define LED_DELAY     10000000
volatile int Delay;

int main(void)
{
        Gpio_Init();

        while(1){

                XGpioPs_WritePin(&Gpio, LED1, 0);
                XGpioPs_WritePin(&Gpio, LED2, 1);

                for (Delay = 0; Delay < LED_DELAY; Delay++);

                XGpioPs_WritePin(&Gpio, LED1, 1);
                XGpioPs_WritePin(&Gpio, LED2, 0);

                for (Delay = 0; Delay < LED_DELAY; Delay++);

        };

        return 0;
}
```

6. Download to the board for verification

Select the hardware platform in the project, right-click → Program FPGA, select Default in the pop-up dialog box, and click "program" to complete the Program work in the FPGA PL part

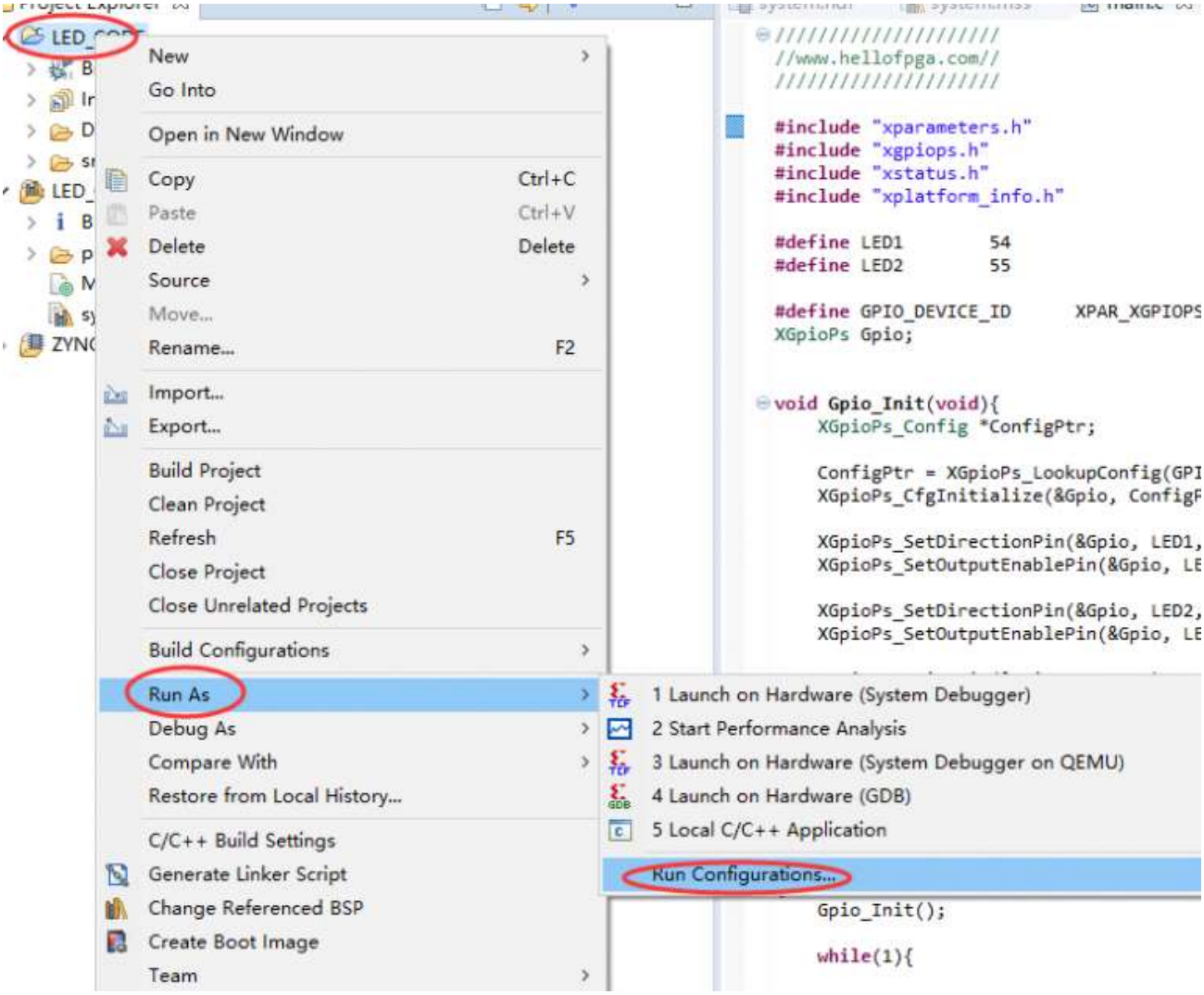


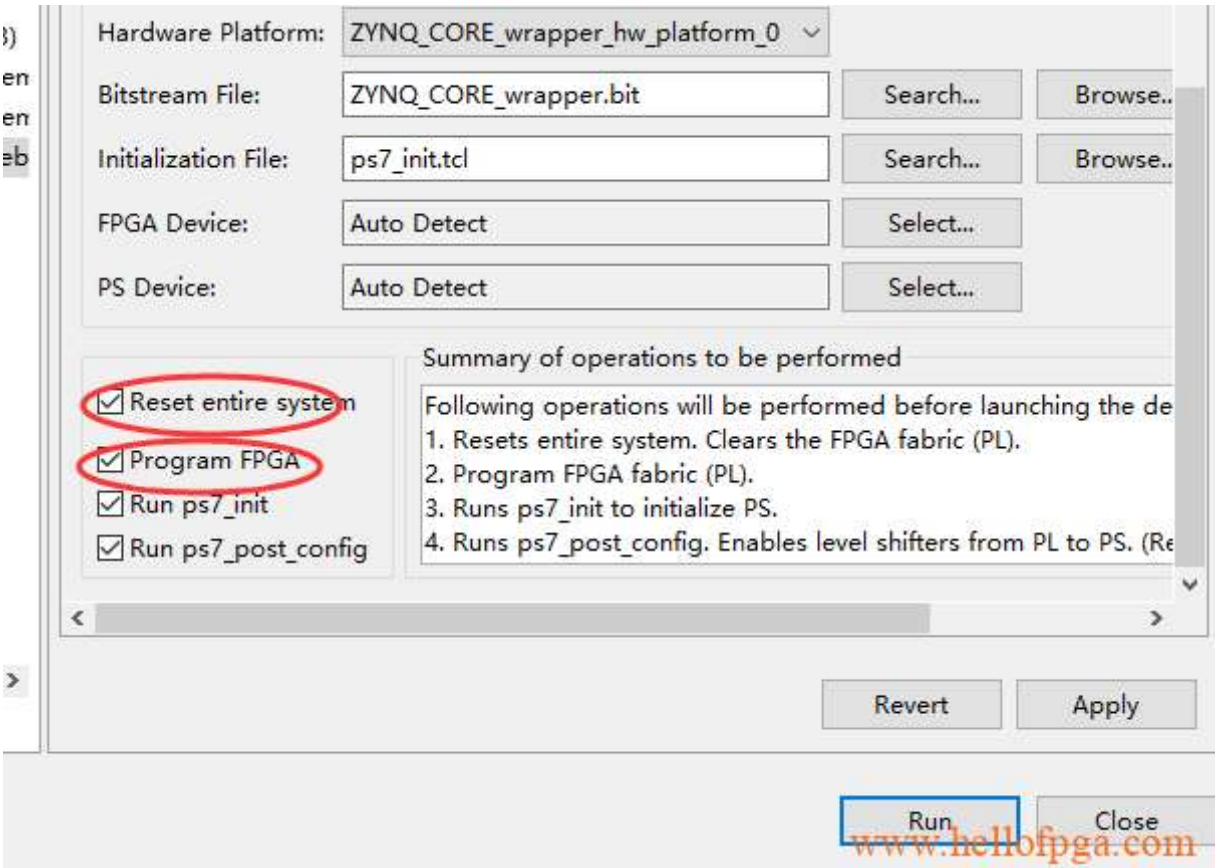
2) Select the GPIO project we generated, expand the icon to the right of the green arrow (RUN), and select Run As→1 Launch on Hardware (System Debugger)



You can see the LED1 light on the board blinking

Note: If an error pops up when RUN, you can follow the following operations to set up and then DEBUG


New
Go Into
Open in New Window
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Configurations
Run As
Debug As
Compare With
Restore from Local History...
C/C++ Build Settings
Generate Linker Script
Change Referenced BSP
Create Boot Image
Team
1 Launch on Hardware (System Debugger)
2 Start Performance Analysis
3 Launch on Hardware (System Debugger on QEMU)
4 Launch on Hardware (GDB)
5 Local C/C++ Application
Run Configurations...
//www.hellofpga.com//
#include "xparameters.h"
#include "xgpiops.h"
#include "xstatus.h"
#include "xplatform_info.h"
#define LED1 54
#define LED2 55
#define GPIO_DEVICE_ID XPAR_XGPIOPS
XGpioPs Gpio;
void Gpio_Init(void){
XGpioPs_Config *ConfigPtr;
ConfigPtr = XGpioPs_LookupConfig(GPI
XGpioPs_CfgInitialize(&Gpio, ConfigP
XGpioPs_SetDirectionPin(&Gpio, LED1,
XGpioPs_SetOutputEnablePin(&Gpio, LE
XGpioPs_SetDirectionPin(&Gpio, LED2,
XGpioPs_SetOutputEnablePin(&Gpio, LE
Gpio_Init();
while(1){


Hardware Platform: ZYNQ_CORE_wrapper_hw_platform_0
Bitstream File: ZYNQ_CORE_wrapper.bit
Search...
Browse...
Initialization File: ps7_init.tcl
Search...
Browse...
FPGA Device: Auto Detect
Select...
PS Device: Auto Detect
Select...
Summary of operations to be performed
Reset entire system
Program FPGA
Run ps7_init
Run ps7_post_config
Following operations will be performed before launching the de
1. Resets entire system. Clears the FPGA fabric (PL).
2. Program FPGA fabric (PL).
3. Runs ps7_init to initialize PS.
4. Runs ps7_post_config. Enables level shifters from PL to PS. (Re
Revert
Apply
Run
Close
www.hellofpga.com

Then click APPLY and then select Run As→1 Launch on Hardware (System Debugger) to see if the download is successful (if it still doesn't work, please power off the board and try again, generally this problem occurs because of conflicts with previously run programs when debugging)

Code interpretation

#define LED1 54 Defines LED 1 as pin 54 of PS (i.e. EIO pin 0) #define LED2 55 defines LED 2 as pin 55 of PS (i.e. EIO pin 1)

Remarks (EMIO is from the 54th foot)

```
XGpio_SetDataDirection 设置GPIO为输入/输出
XGpioPs_SetOutputEnablePin(&Gpio, LED, 1);//使能 LED 对应的GPIO的输出功能
XGpioPs_WritePin(&Gpio, LED, 0); //拉低 LED 对应的GPIO
XGpioPs_WritePin(&Gpio, LED, 1); //拉高 LED 对应的GPIO

for (Delay = 0; Delay < LED_DELAY; Delay++); 这是一个耗费系统资源的delay函数的简写
```

所以整个程序的效果就是LED 不停的点亮熄灭，反复循环。

完整工程如下：

07_PS_LED_TEST_EMIO_XC7Z020 **Download**

In fact, whether it is the EMIO method or the AXI GPIO method, or the code of the reference design or my condensed code are all the same, as long as you know what function you want to achieve, and then study the code of the reference design, you can write the function you want

In addition, the following address is the difference between the three Mio emio axigpio compiled by yourself, if you are interested, you can take a look

http://www.hellofpga.com/index.php/2021/08/08/zynq_emio_mio_axi_gpio/

SMART ZYNQ SP & SL