APRIL 2023, 4 BY ACKYE

Smart ZYNQ (SP&SL Edition) Project XVI Demonstration of UART functionality in the PS part based on ZYNQ

This article demonstrates the hello world routine of the serial port on the motherboard in the way of MIO (UART resources for the type C port of the motherboard)

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)

(Remarks The silkscreen TX and RX silkscreen logos on the V1.0 board are reversed, and the schematic diagram prevails)

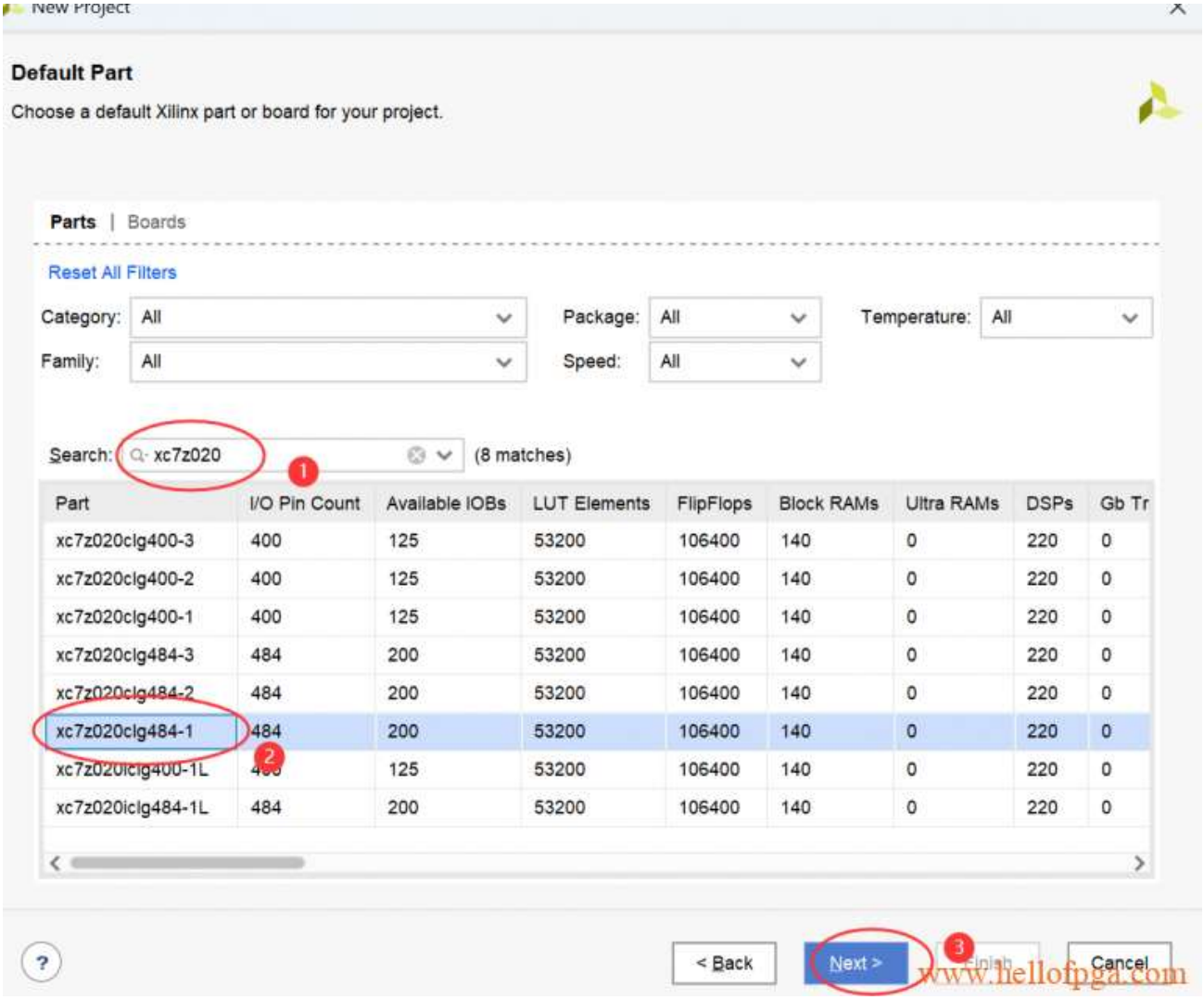
1. Introduction of hardware resources

There are two kinds of mapping of the serial port of the PS end, one way is to directly pass through the IO port of the PS dedicated serial port of ZYNQ. Method 2 The UART on the PS side is mapped to any PL port for eviction through EMIO, and the effect of the two methods is exactly the same (the method can be extended to any IO port on the PL side). (Because our serial port is connected to the PL side, here we use method two to map the UART to the PL IO through EMIO)

As can be seen from the figure below, the UART function of the motherboard is connected to the L17 pin and M17 pin on the ZYNQ PL side, that is, the TX of ZYNQ is connected to L17, and the RX of ZYNQ is connected to M17 **(Note: The silkscreen TX and RX on the V1.0 board are reversed, subject to the schematic).**

2. Create a project

1) Create a new project, select XC7Z020CLG484-1 according to the chip model selection

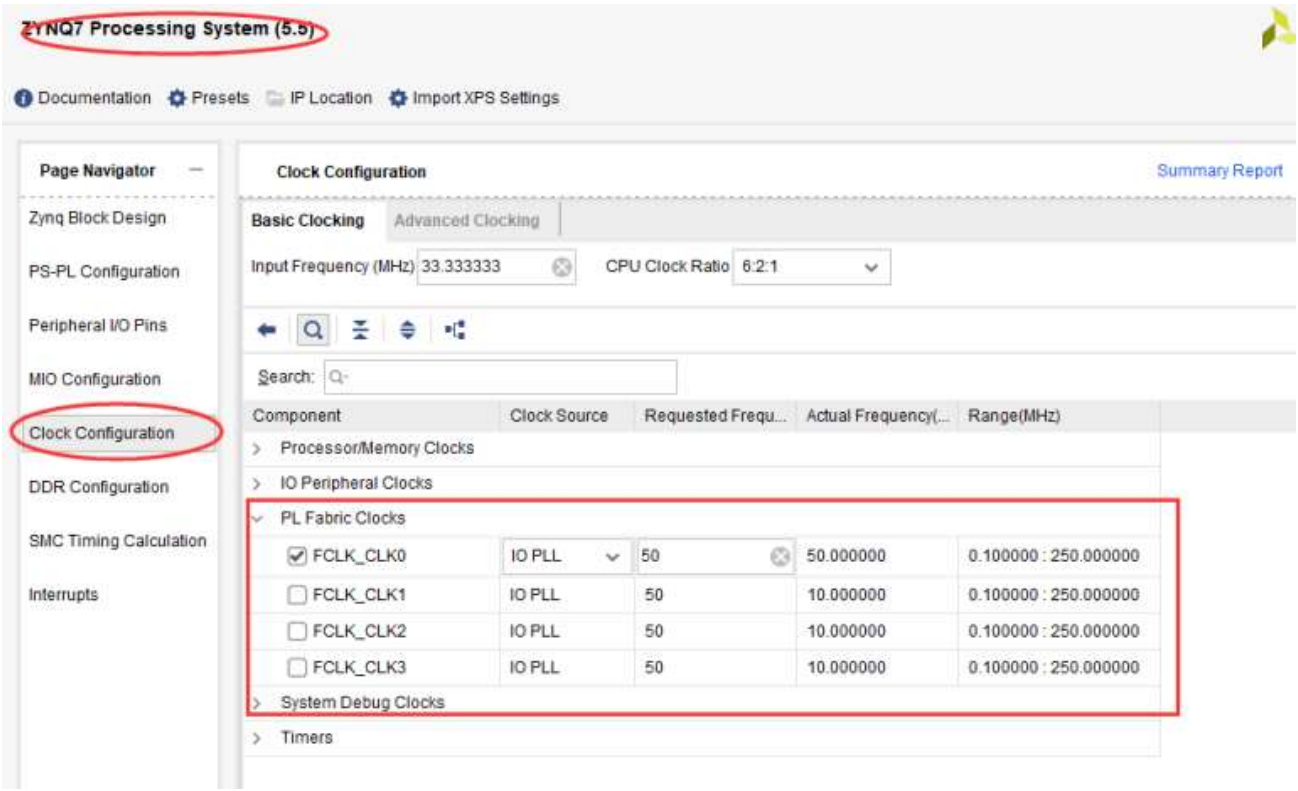


2) Create a block design, and add ZYNQ7 PROCESSING SYSTEM module, the software automatically generates a zynq block as shown in the figure below, next to do some

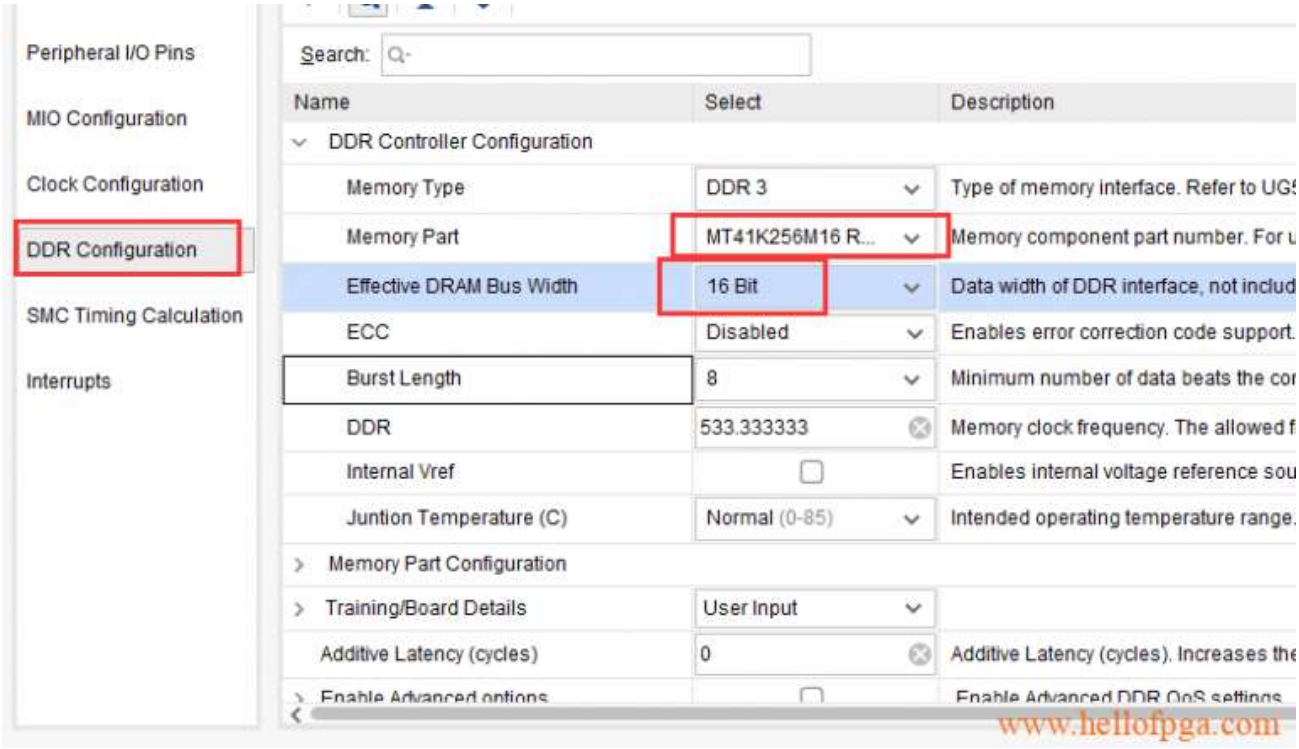
corresponding settings, double-click the ZYNQ core in the figure below

3) Set the clock function in ZYNQ:

Find the Clock Configuration option in the Settings project, set your desired clock frequency in PL Fabric Clocks, there are a total of 4 frequencies that can be set similar to our PLL function. Here we set the 50M clock

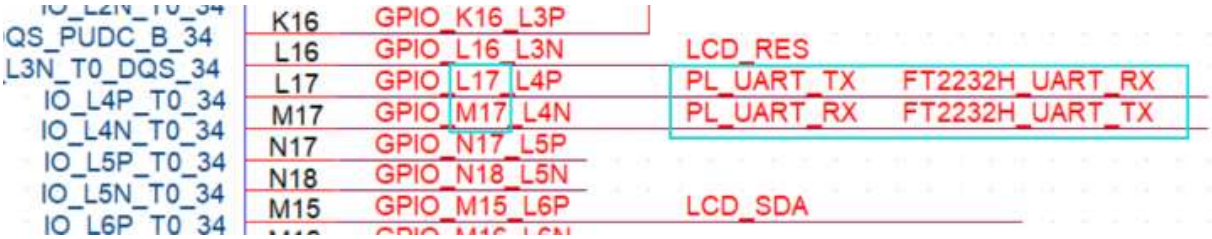


4) Find DDR Configuration →DDR Controller Configuration →DDR3 in the pop-up window in turn, select the corresponding DDR3 according to the DDR on your board in the Memory Part drop-down menu, the model used in this experiment: MT41K256M16RE-125, select 16bit of data width and finally click "OK", as shown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

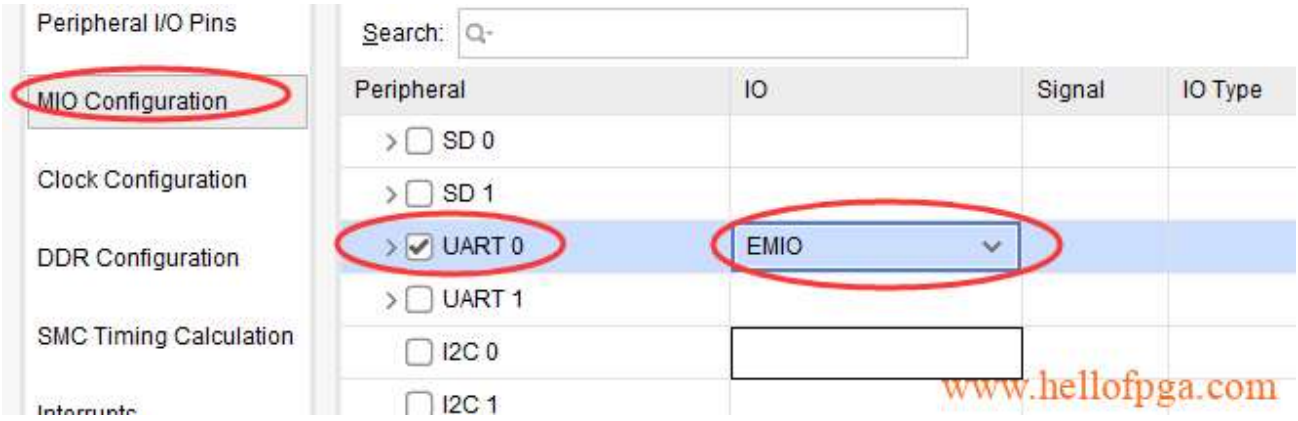


5) Add UART functionality

As can be seen from the figure below, the UART function of the motherboard is connected to the L17 pin and M17 pin on the ZYNQ PL side, that is, the TX of ZYNQ is connected to L17, and the RX of ZYNQ is connected to M17 **(Note: The silkscreen TX and RX on the V1.0 board are reversed, subject to the schematic).**

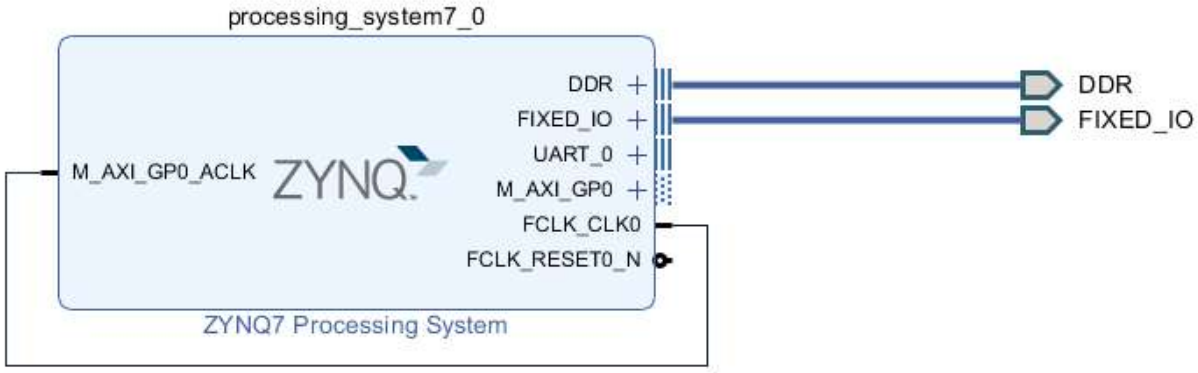


Enable UART 0 and select the EMIO mode in the IO option

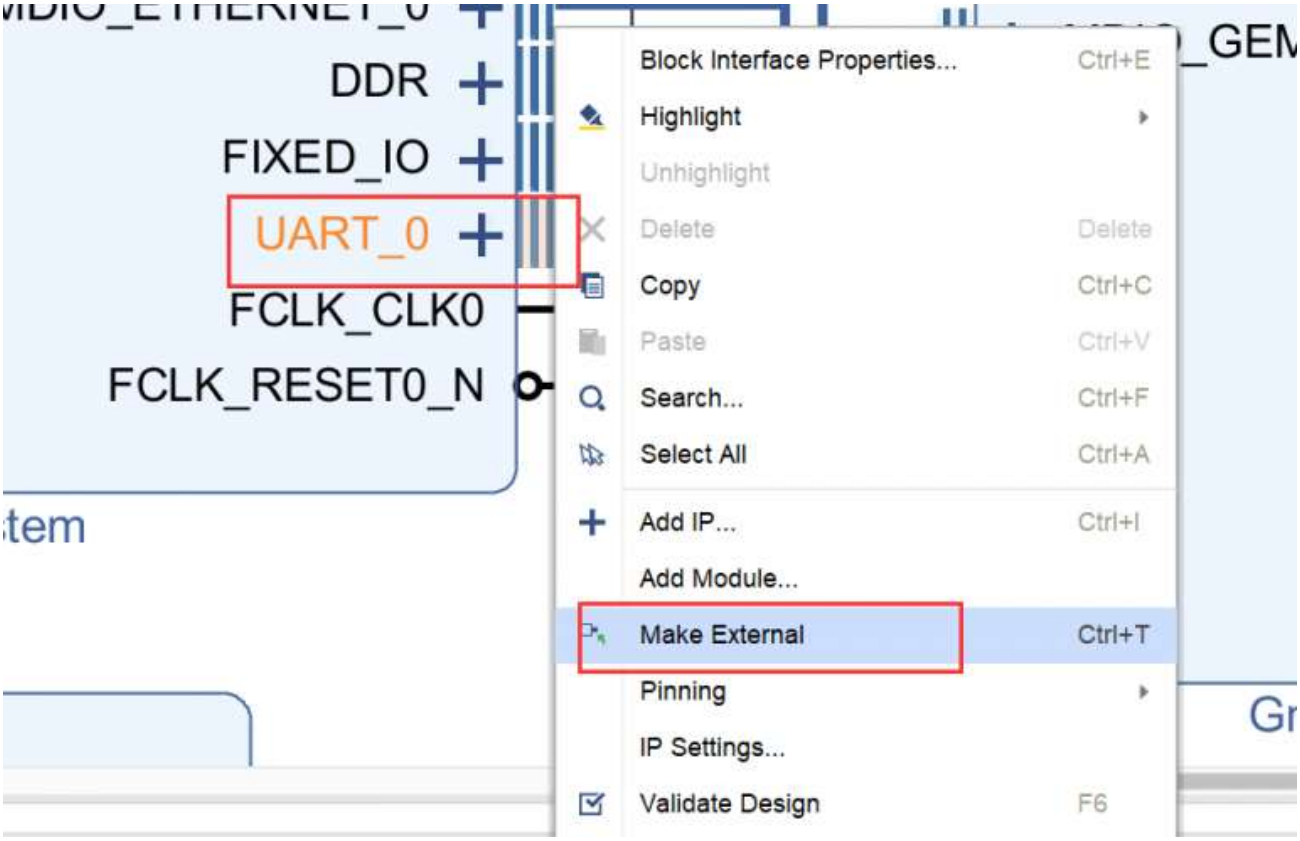


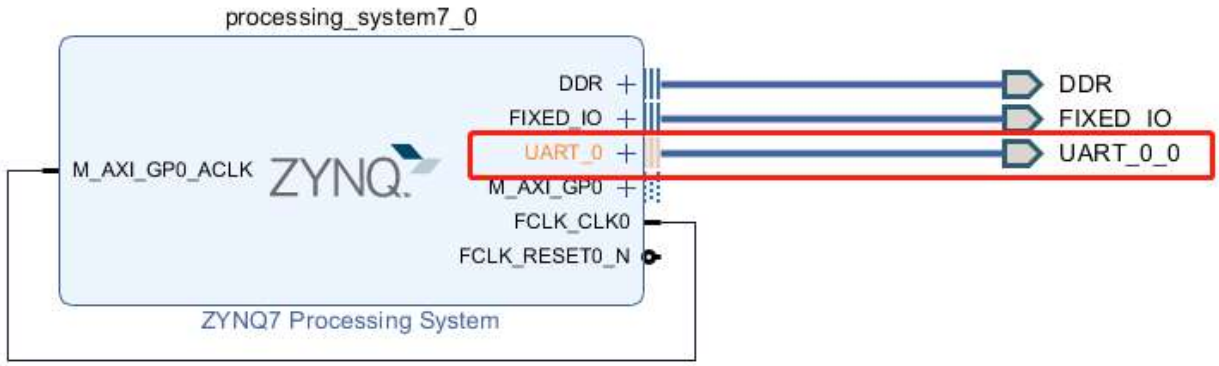
6) After completing the above operations, click "Run Block Automation" as shown in the figure below. Keep the default in the pop-up options, click "OK", you can complete the

configuration of the ZYNQ7 Processing System, and connect the FCLK_CLK and M_AXI_GP0_ACLK with the mouse to get the following figure



Because we have added the UART for EMIO, here we need to lead the UART pinout by right-clicking the UART_0 of the ZYNQ module, and then selecting Make External

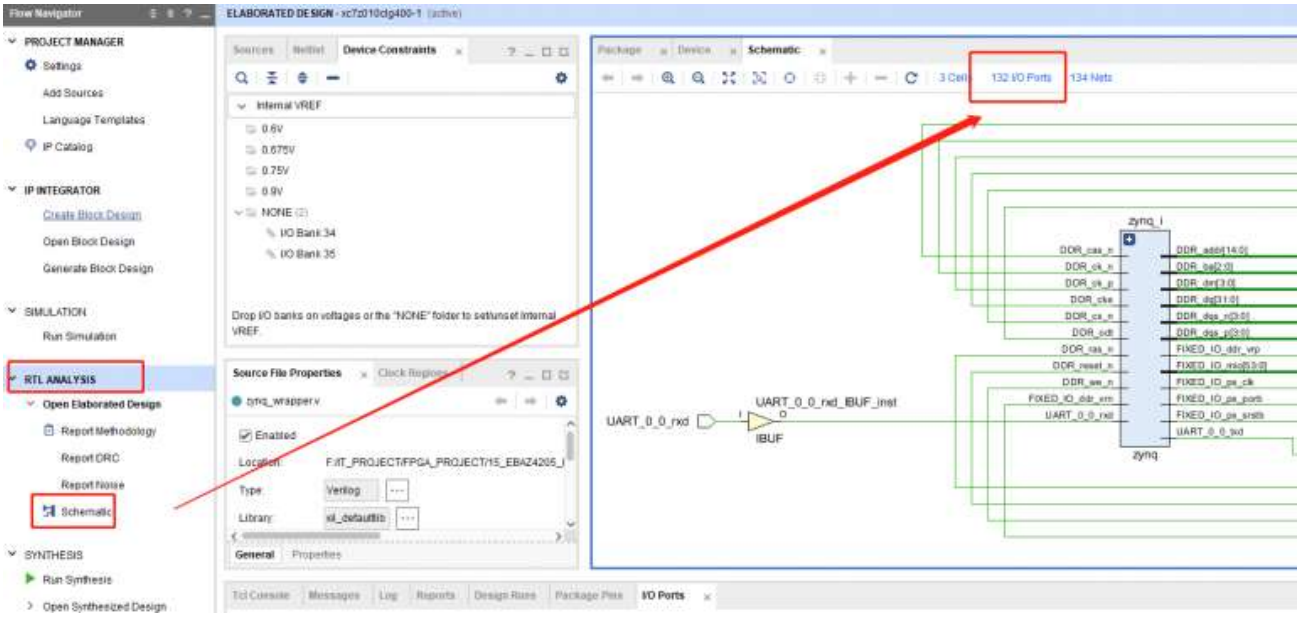




7) source→ Design Source, right-click the BLOCK project we created, click create HDL wrapper, package the BLOCK file and generate the .v code

8) Click the green arrow RUN to compile the code

9) Click SCHEMATIC in RTL and select IO Ports on the right to add the pin definition of the UART 0 EMIO section (this step can also be defined in the constraint file, see the previous example)

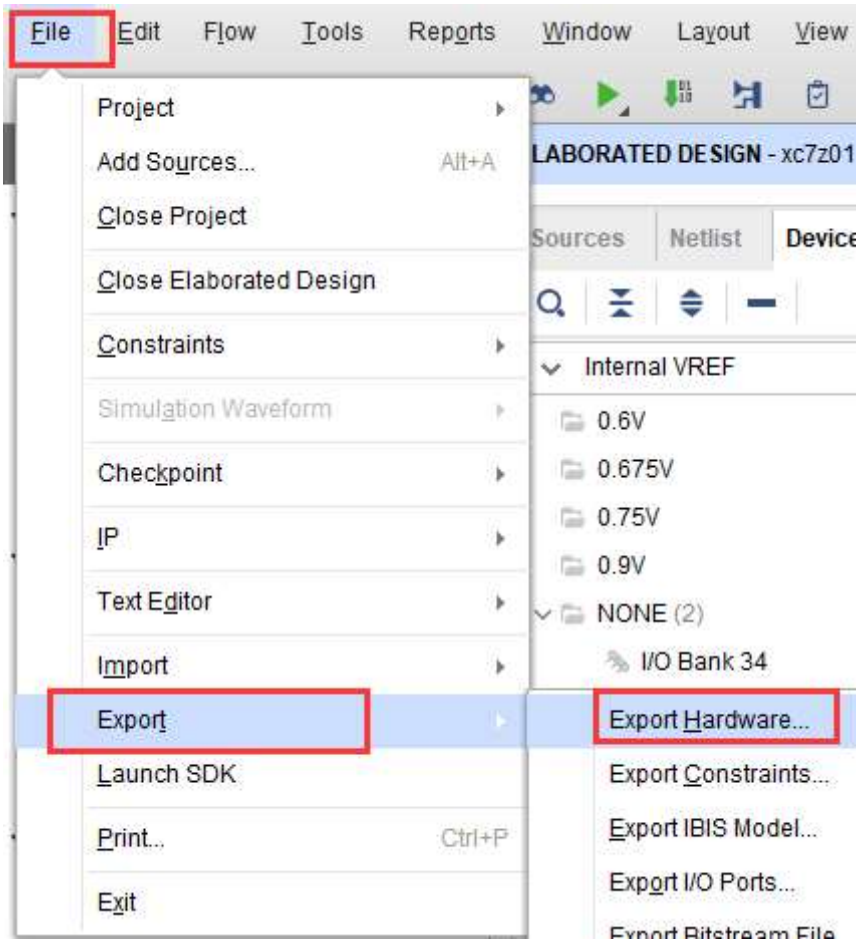


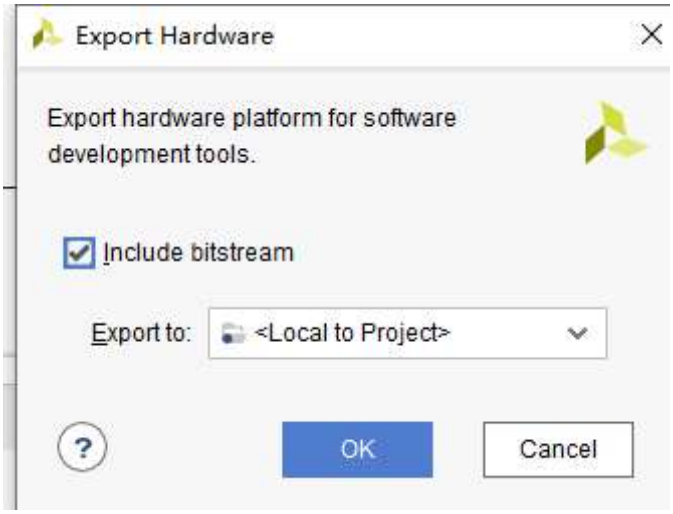
Set TX RX in the UART section to L17 and M17 voltage property to LVCMOS33, and then save it

10) Generate bit file: Press Generate Bitstream to complete synthesis and generate bit file (a pop-up window will ask you to save the constraint rule file)

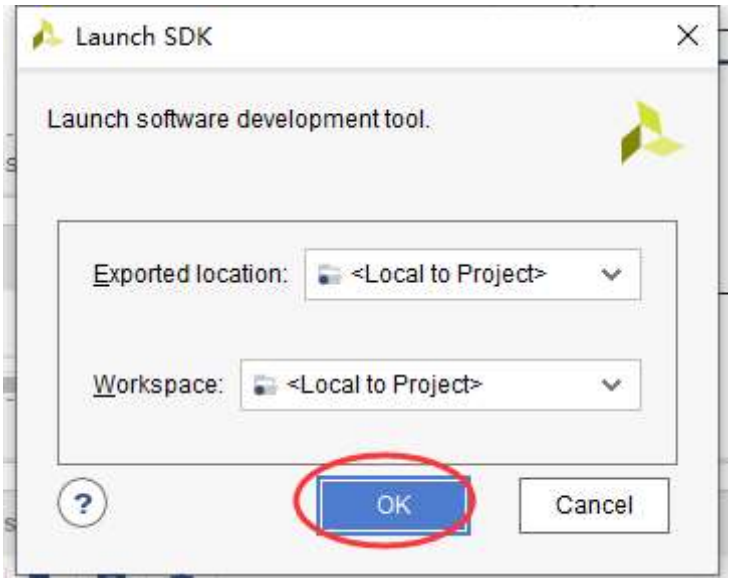
5. SDK program writing

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.

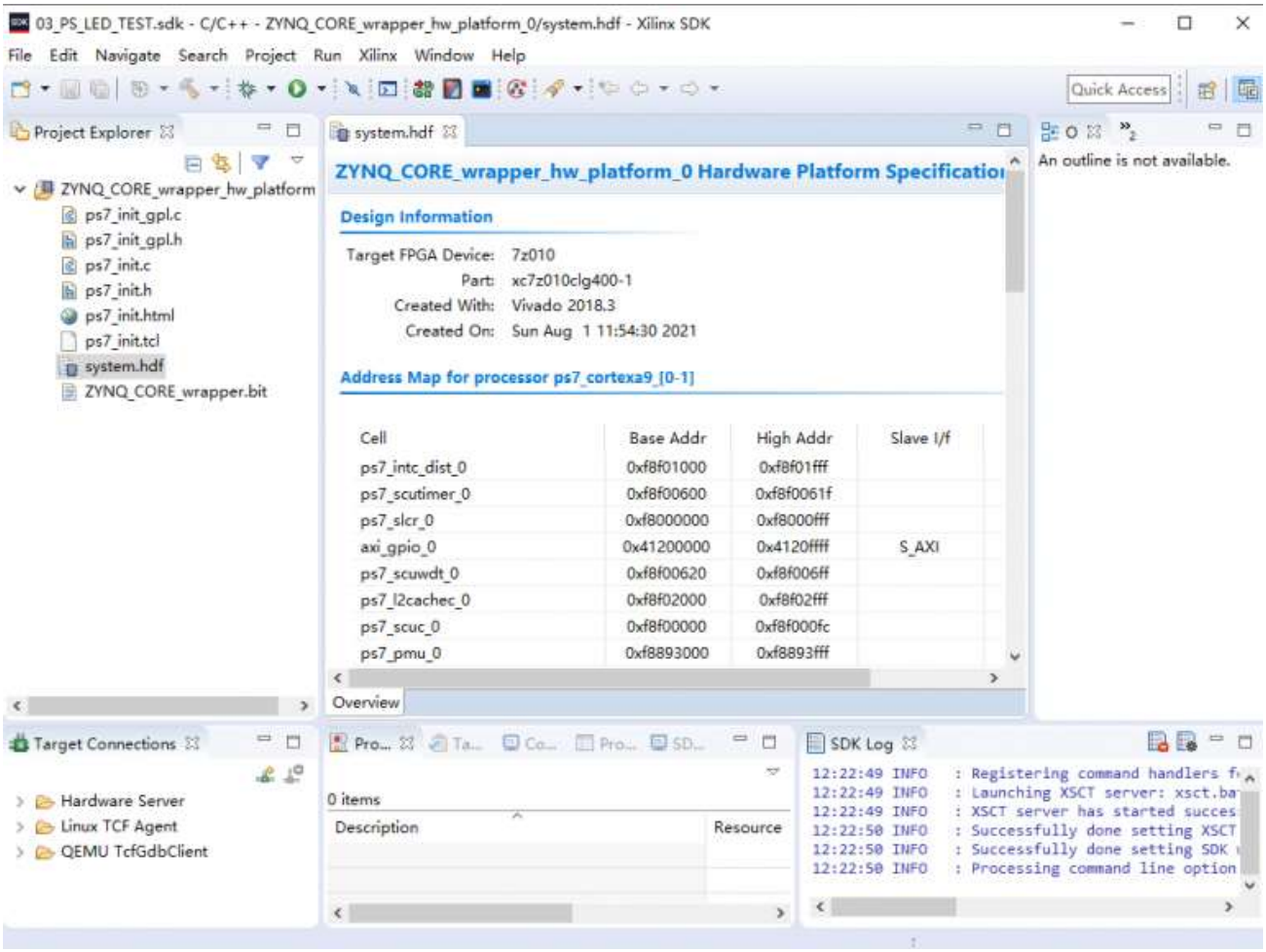




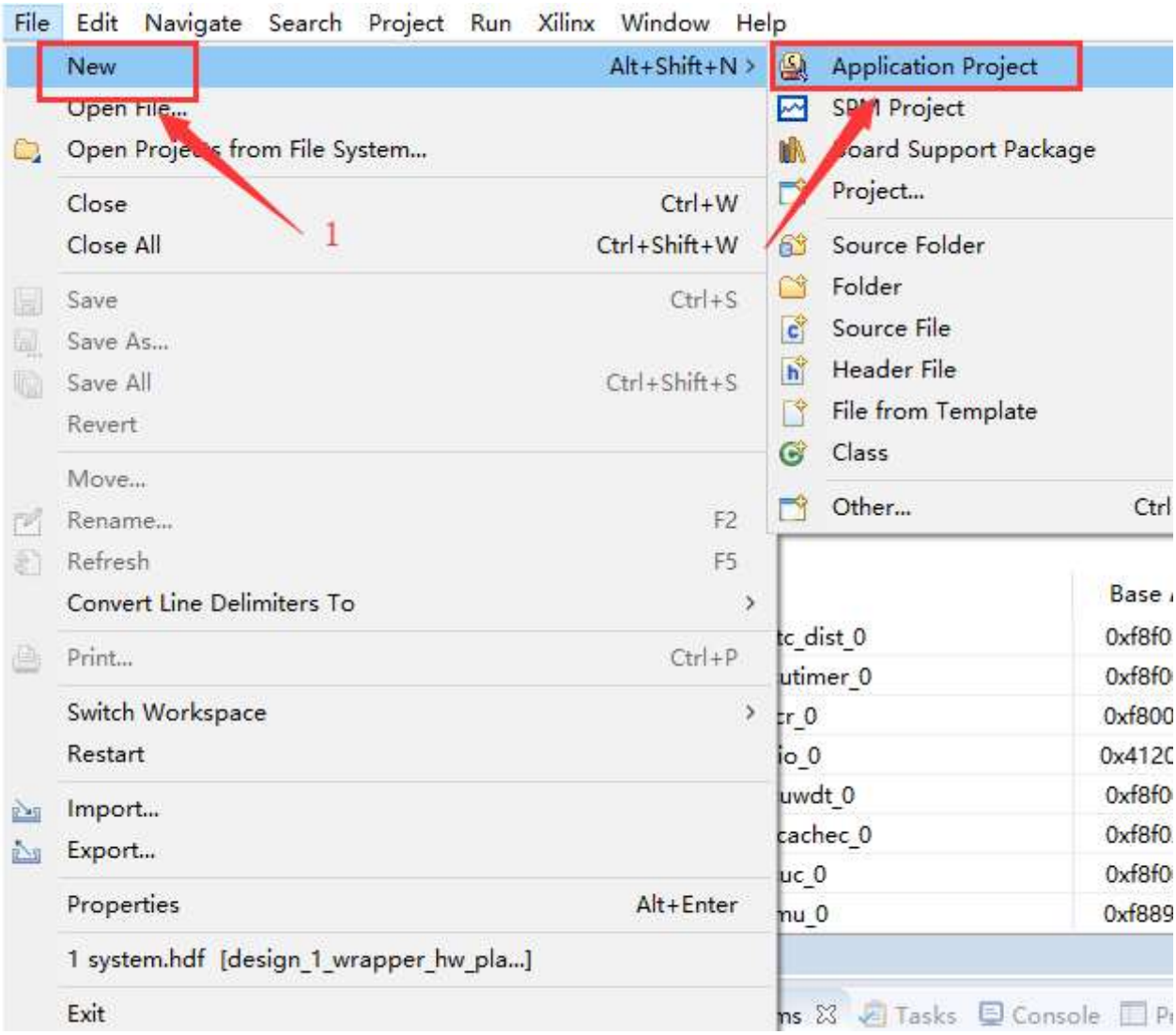
2) File→ Lauch SDK, in the pop-up dialog box, save the default, click "OK", as shown in the figure below.



系统将自动打开SDK开发环境

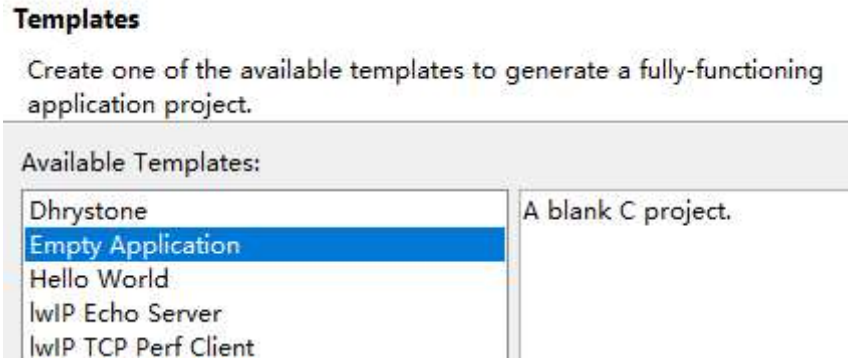


3) Create a new project file→new→Application Project to create an "Application Project", as shown in the figure below.



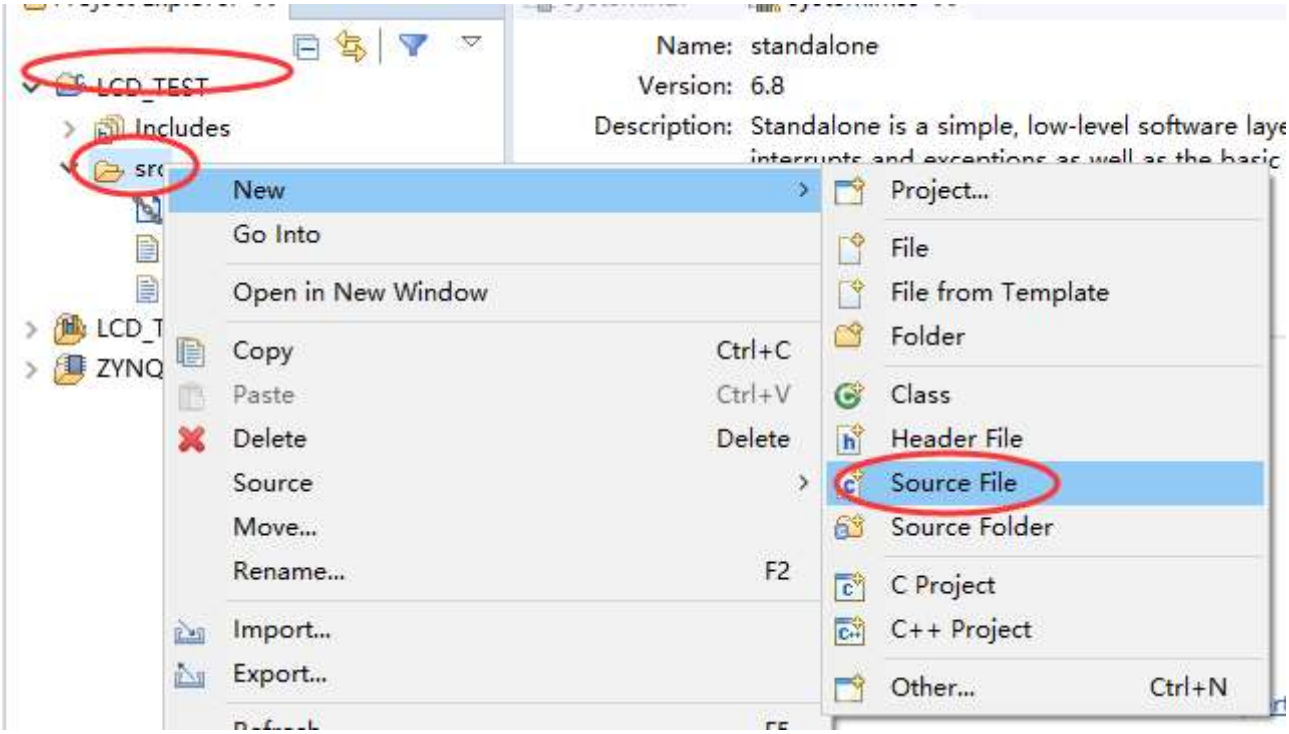
4) Enter your own project name (UART) in New Project Name and click NEXT

5) Select an empty project and click Done

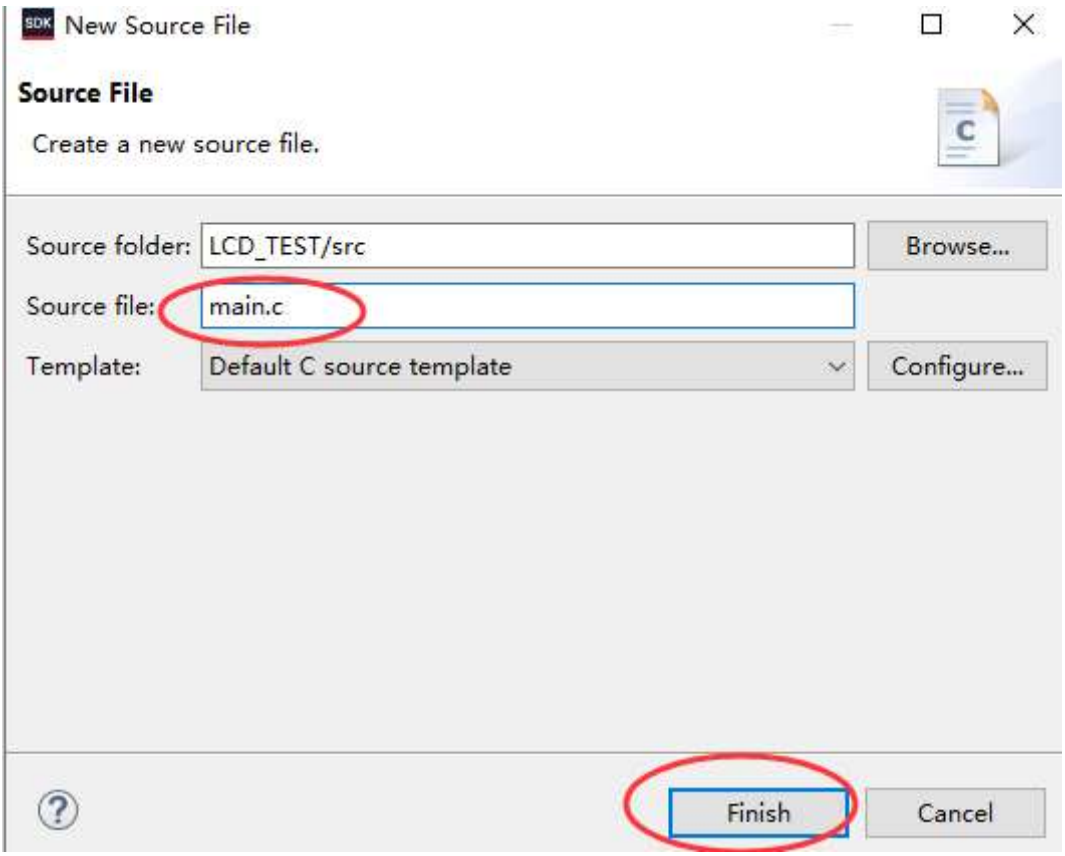


6) Create our own code in an empty project

Expand the project we created, right-click on the src directory, and select New->Source File, as shown in the following figure:



Create a main.c file in the pop-up window

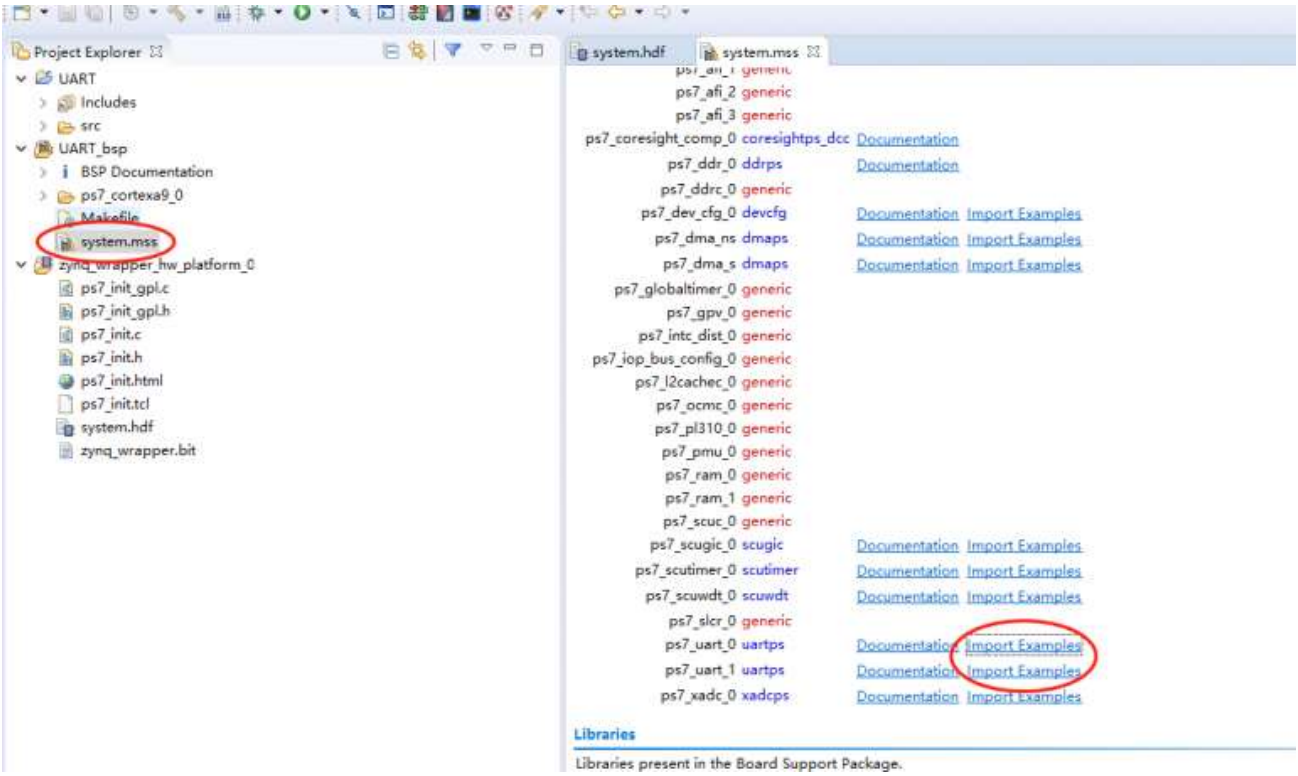


7) Write your own code

7.1 **Tips:**

When writing the PS part of the code for the first time, such as GPIO code and UART code, if you don't know how to write it, you can actually open the system.mss file under the BSP

project, and then import the required reference routine on the right, and then modify the part of the routine you need (such as GPIO initialization, or the UART write and read code COPY into the main function of your project).



7.2 Copy the following code to main.c

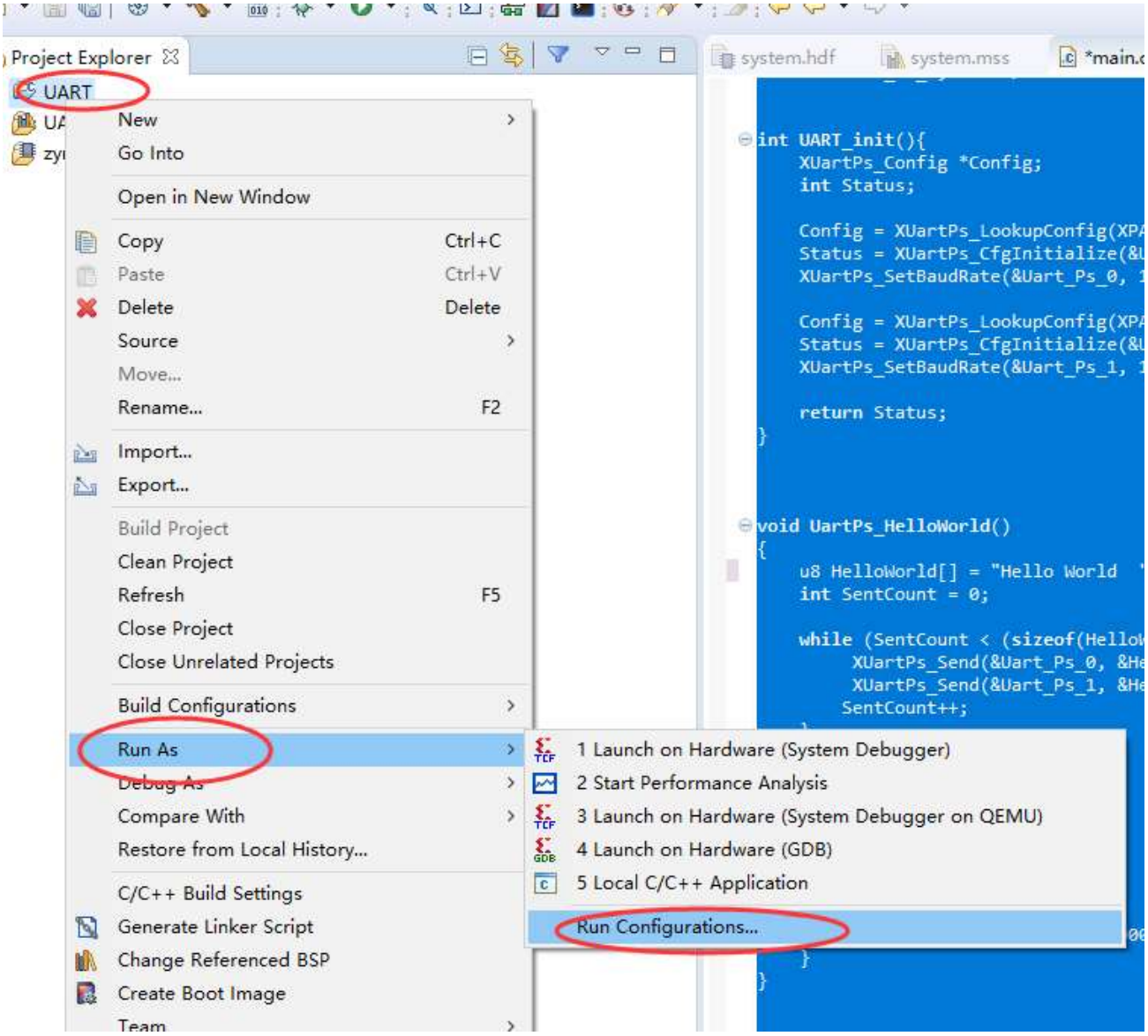
```
#include "xparameters.h"
#include "xuartps.h"
#include "xil_printf.h"

XUartPs Uart_Ps_0;              /* The instance of the UART Driver
*/

int UART_init(){
        XUartPs_Config *Config;
        int Status;

        Config = XUartPs_LookupConfig(XPAR_XUARTPS_0_DEVICE_ID);
        Status = XUartPs_CfgInitialize(&Uart_Ps_0, Config, Config-
>BaseAddress);
        XUartPs_SetBaudRate(&Uart_Ps_0, 115200);

        return Status;
}
```

```
void UartPs_HelloWorld()
{
        u8 HelloWorld[] = "Hello World  ";
        int SentCount = 0;

        while (SentCount < (sizeof(HelloWorld) - 1)) {
                 XUartPs_Send(&Uart_Ps_0, &HelloWorld[SentCount],
1);
                SentCount++;
        }
}

int main(void)
{
        int Delay;
        UART_init();
        while(1){
                UartPs_HelloWorld();
                for (Delay = 0; Delay < 100000000; Delay++);
        }
}
```

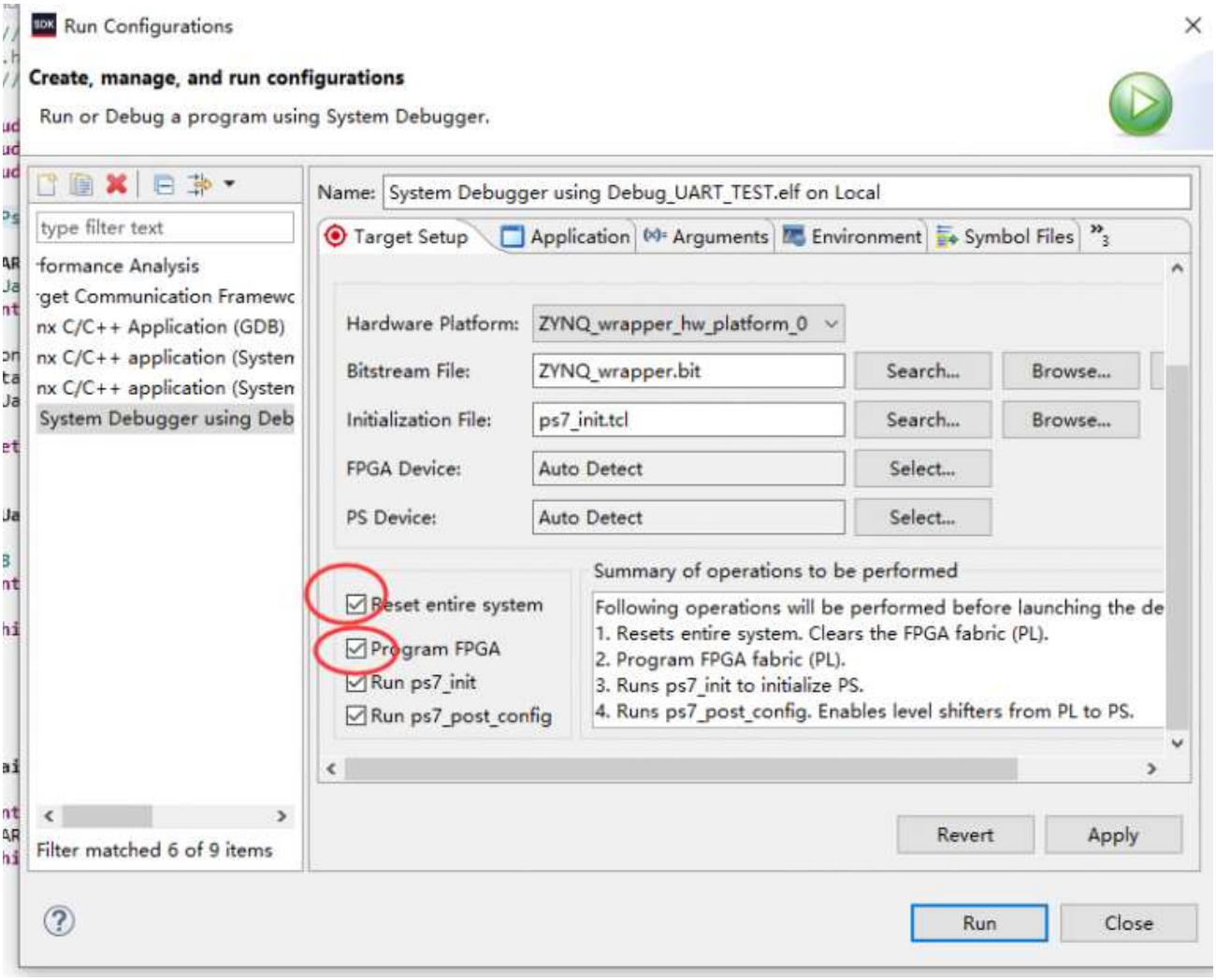
I referred to the official code here, and made certain deletions (removing unnecessary initialization verification, etc.), initialized UART 0, and in the code of helloworld, the port performs helloworld's string output, looping repeatedly.

8. Code download verification

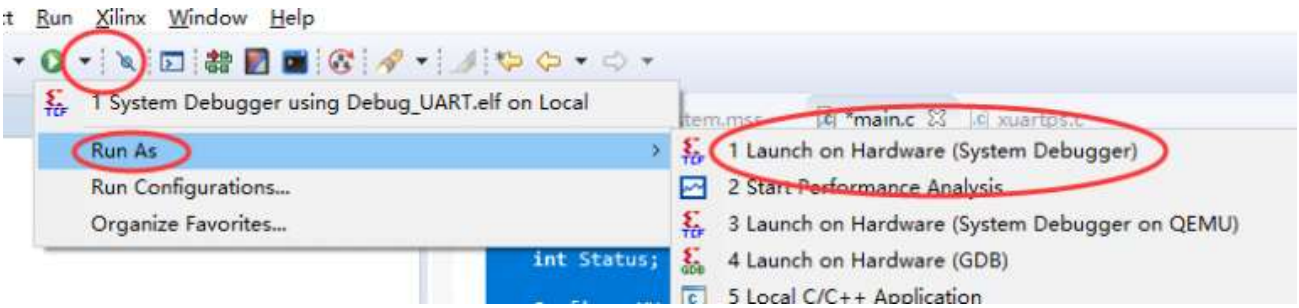
Configure the environment for RUN AS first



In the pop-up window, check all 4 boxes to ensure that the system will be restarted every time it runs, and the FPGA will be reprogrammed, so that there will be no problem that the probability does not work when downloading the program debug (if it is not checked, when the board itself runs the program, and then re-debug will probably, there will be a download error) PS does not appear The following figure dialog box appears, you can first click the green arrow DEBUG , and then re-click RUN in the above figure Options for CONFIGURATIONS



Once set, select -> or Launch on Hardware (GDB).`Run AsLaunch on Hardware (System Debugger)`



After the above operations, the serial port will output data normally, at this time use the serial port assistant to view, you can see that the system is constantly sending hello world string (serial port baud rate 115200)

The following is the complete project mentioned in this article, for reference only

16_PS_UART_TEST_XC7Z020 **Download**

Note ZYNQ chip function is very powerful, the above is only a simple helloworld demonstration, the actual serial port function can also call interrupt, call FIFO, etc., this is their own learning attempt

NOTE IF YOU USE YOUR OWN TYPE C CHARGING CABLE, PLEASE NOTE THAT SOME TYPE C CHARGING CABLES ON THE MARKET ARE CHARGE-ONLY, WITHOUT DATA TRANSMISSION FUNCTION, AND NEED TO USE TYPE C DATA CABLE TO ACHIEVE COMMUNICATION

SMART ZYNQ SP & SL