

HELLO, FPGA

Document my FPGA learning journey

APRIL 2023, 4 BY ACKYE

Smart ZYNQ (SP&SL version) Project 3 Provide a clock for PL logic through the PS part (common method when doing engineering)

This article introduces a method of providing a logic clock for PL by sharing the clock through PS, which is more commonly used in engineering and can omit an active crystal oscillator

This article will also be the first PS and PL linkage project in the Tiny ZYNQ series of projects

(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)

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

Create a project

- 1) Create a new project, select **XC7Z020CLG484-1** for the chip model

Search: (8 matches)

| Part | I/O Pin Count | Available IOBs | LUT Elements | FlipFlops | Block RAMs | Ultra RAMs | DSPs | Gb Tr |
|------------------|---------------|----------------|--------------|-----------|------------|------------|------|-------|
| xc7z020clg400-3 | 400 | 125 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg400-2 | 400 | 125 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg400-1 | 400 | 125 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg484-3 | 484 | 200 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg484-2 | 484 | 200 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg484-1 | 484 | 200 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg400-1L | 400 | 125 | 53200 | 106400 | 140 | 0 | 220 | 0 |
| xc7z020clg484-1L | 484 | 200 | 53200 | 106400 | 140 | 0 | 220 | 0 |

< >

? < Back Next > 3 Finish Cancel

2) Create a BLOCK design

a. 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

Create Block Design

Please specify name of block design.

Design name: ZYNQ_CORE

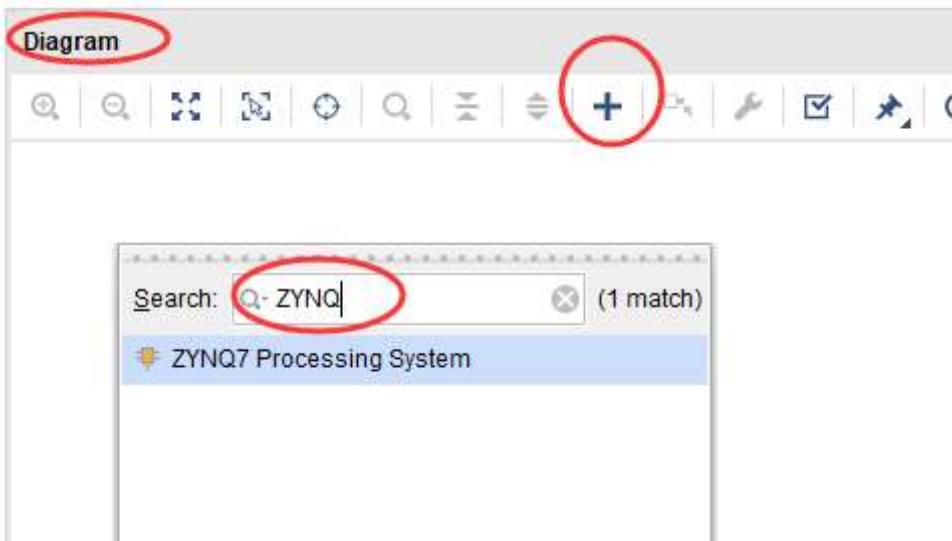
Directory: <Local to Project>

Specify source set: Design Sources

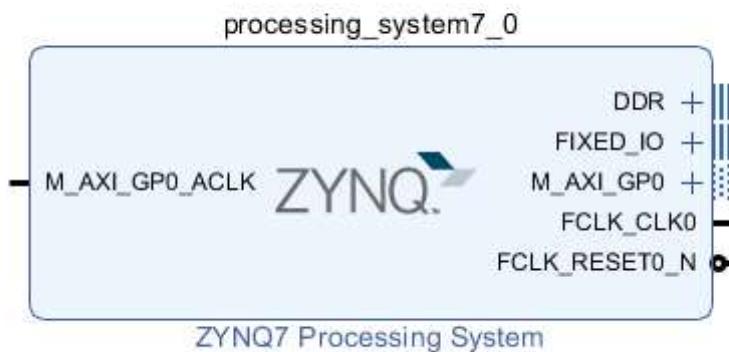
OK Cancel

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) 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

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator

- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration**
- DDR Configuration
- SMC Timing Calculation
- Interrupts

Clock Configuration

Basic Clocking Advanced Clocking

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

| Component | Clock Source | Requested Freq... | Actual Frequency... | Range(MHz) |
|---|--------------|-------------------|---------------------|-----------------------|
| > Processor/Memory Clocks | | | | |
| > IO Peripheral Clocks | | | | |
| PL Fabric Clocks | | | | |
| <input checked="" type="checkbox"/> FCLK_CLK0 | IO PLL | 50 | 50.000000 | 0.100000 : 250.000000 |
| <input type="checkbox"/> FCLK_CLK1 | IO PLL | 50 | 10.000000 | 0.100000 : 250.000000 |
| <input type="checkbox"/> FCLK_CLK2 | IO PLL | 50 | 10.000000 | 0.100000 : 250.000000 |
| <input type="checkbox"/> FCLK_CLK3 | IO PLL | 50 | 10.000000 | 0.100000 : 250.000000 |
| > System Debug Clocks | | | | |
| > Timers | | | | |

5) Set up DDR function in zynq:

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.

Peripheral I/O Pins

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

Search: Q-

| Name | Select | Description |
|-------------------------------------|--------------------------|--|
| DDR Controller Configuration | | |
| Memory Type | DDR 3 | Type of memory interface. Refer to UG! |
| Memory Part | MT41K256M16 R... | Memory component part number. For u |
| Effective DRAM Bus Width | 16 Bit | Data width of DDR interface, not includ |
| ECC | Disabled | Enables error correction code support. |
| Burst Length | 8 | Minimum number of data beats the cor |
| DDR | 533.333333 | Memory clock frequency. The allowed f |
| Internal Vref | <input type="checkbox"/> | Enables internal voltage reference sou |
| Junction Temperature (C) | Normal (0-85) | Intended operating temperature range. |
| > Memory Part Configuration | | |
| > Training/Board Details | User Input | |
| Additive Latency (cycles) | 0 | Additive Latency (cycles). Increases the |
| > Enable Advanced options | <input type="checkbox"/> | Enable Advanced DDR OnS settings |

5) After completing the above operations, click "Run Block Automation" as shown in the figure below



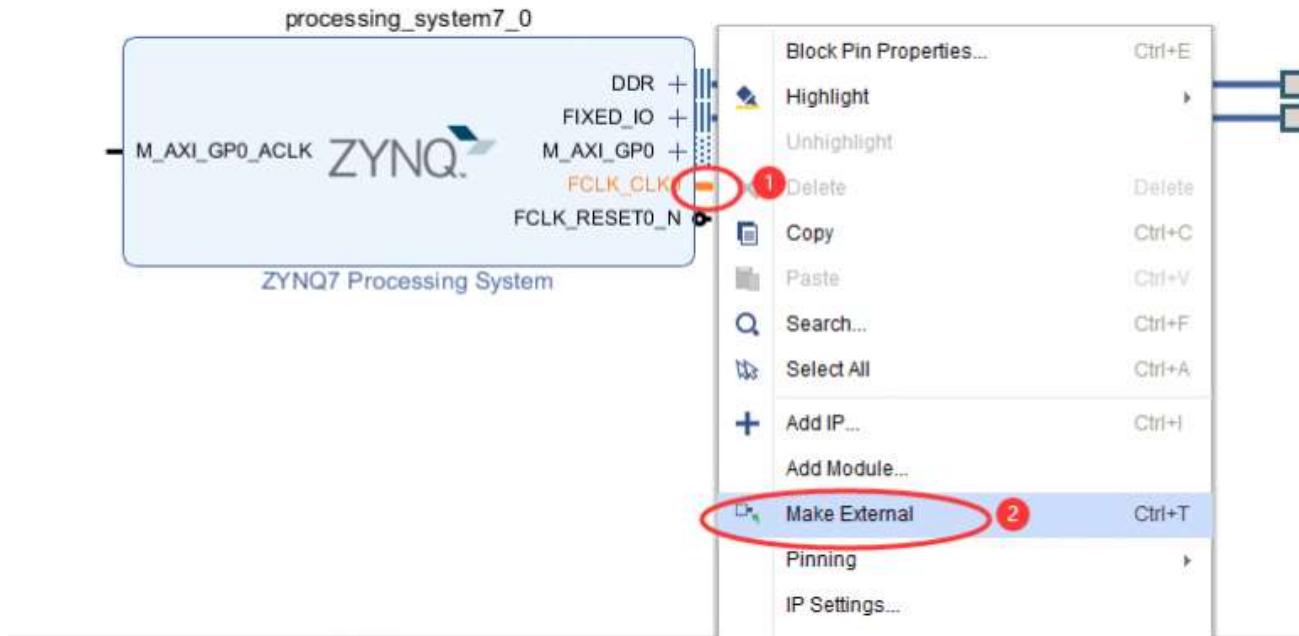
* Designer Assistance available. Run Block Automation



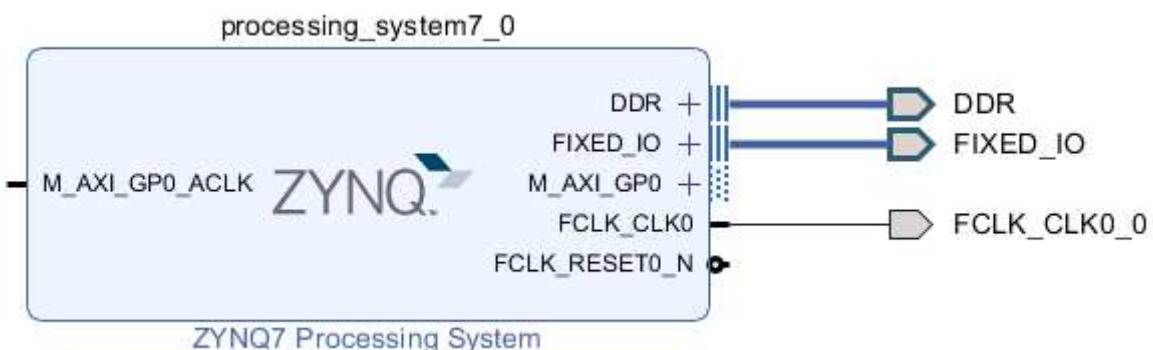
Keep the default in the pop-up options, click "OK", and you can complete the configuration of the ZYNQ7 Processing System, as shown below



6) Right-click FCLK_CLK0 the right line and click Make External to pull out the clock of PS as below screenshot shown

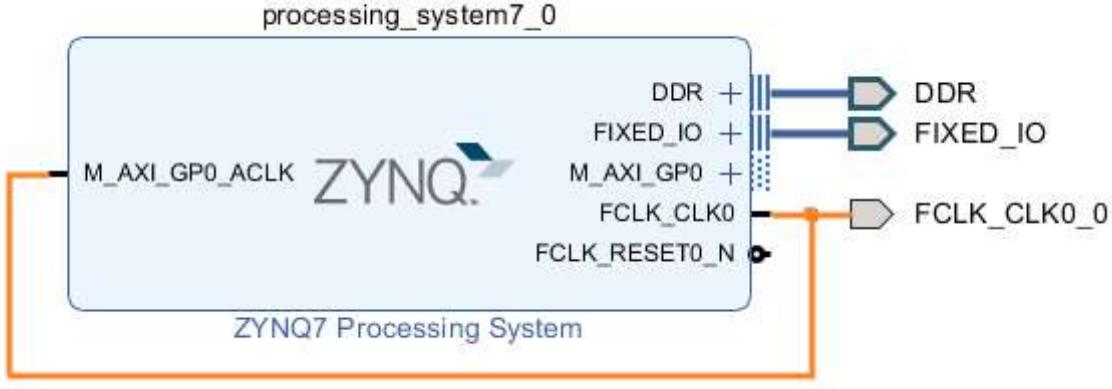


After elicitation as shown in the figure below



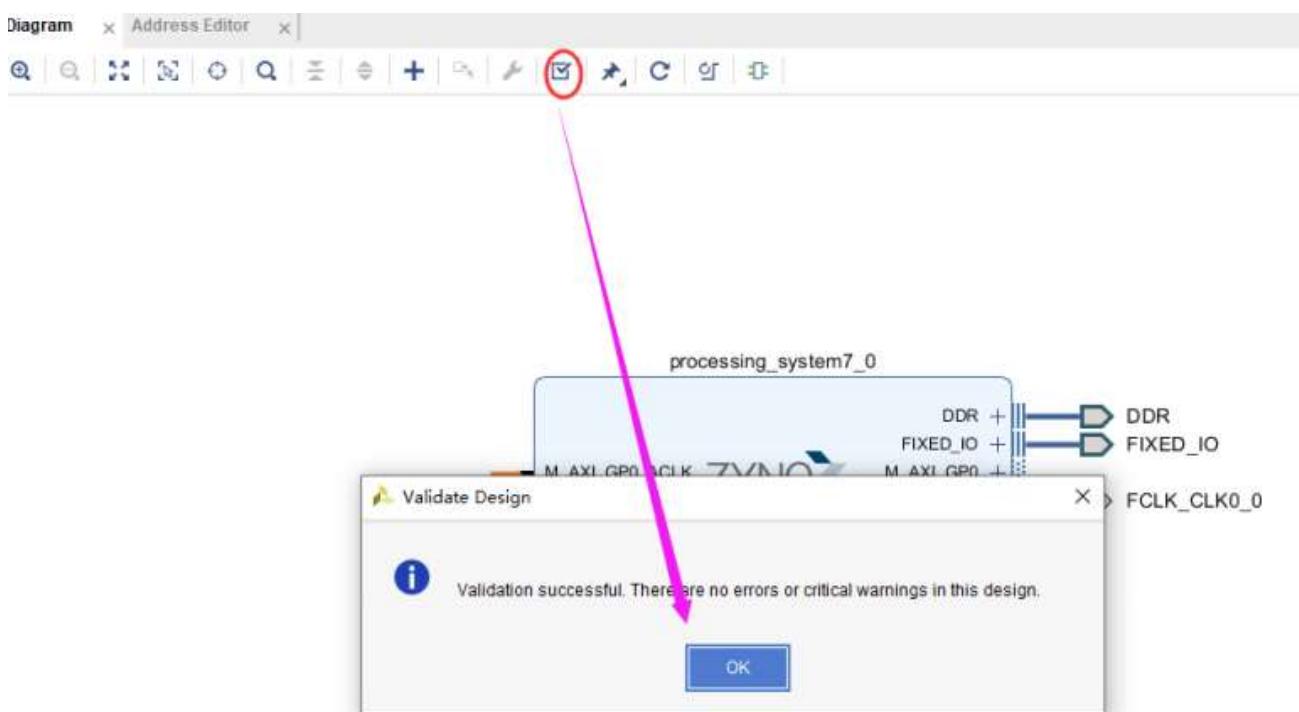
- 7) Connect a wire between the FCLK_CLK0 and the M_AXI_GP0_ACLK to provide a clock for the PS (if you don't connect, you will report an error, you can also disable the AXI interface in the ZYNQ setting interface, so you don't need to connect)

The final result is shown in the following figure

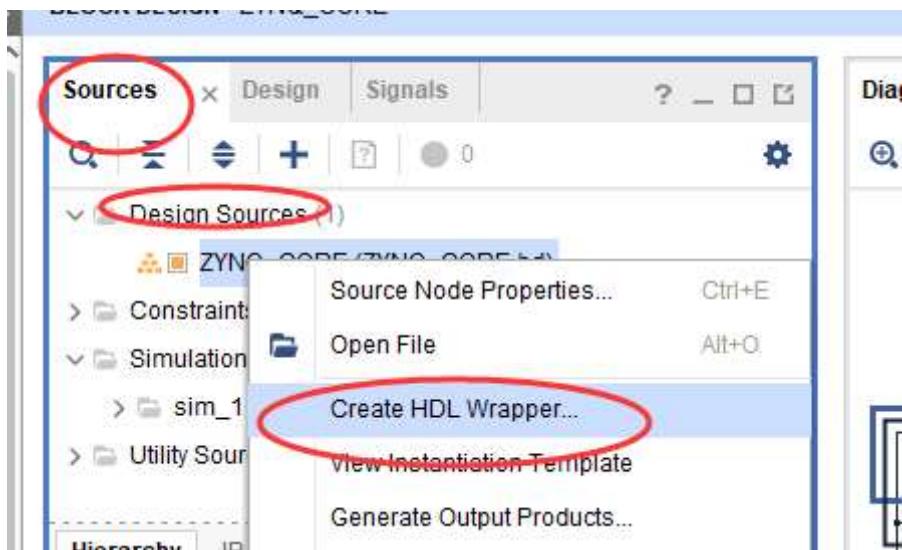


The above has completed the setup of the ZYNQ PS part, and then design your own PL module code and connect with the ZYNQ PS part

At this point, you can check whether the block design has a design error, and if there is no error, the positive design is correct



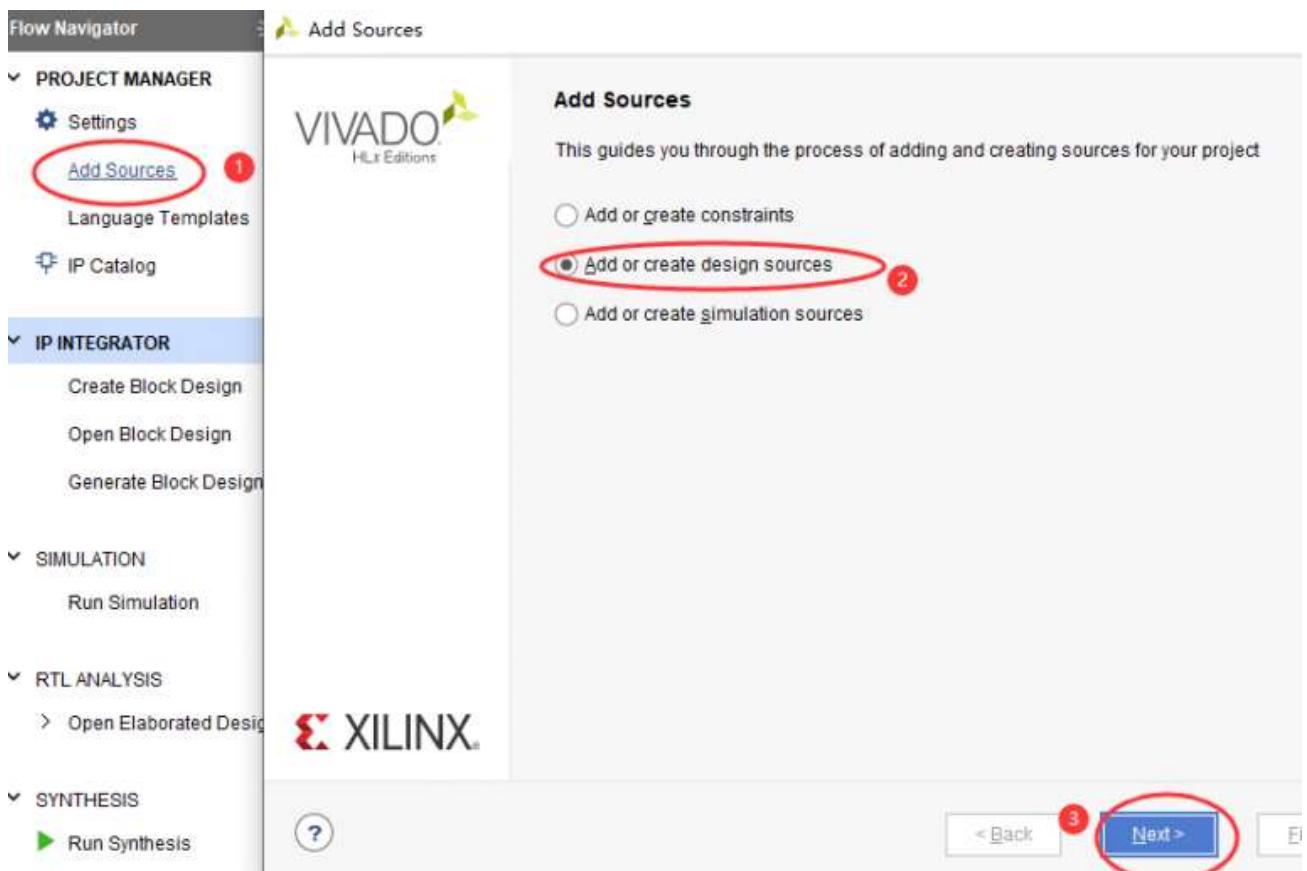
7) Create hardware description, source→Design Source, right-click the BLOCK project we created, click create HDL wrapper as shown in the figure below (this step is equivalent to the function of converting the drawing into the corresponding hardware description language)

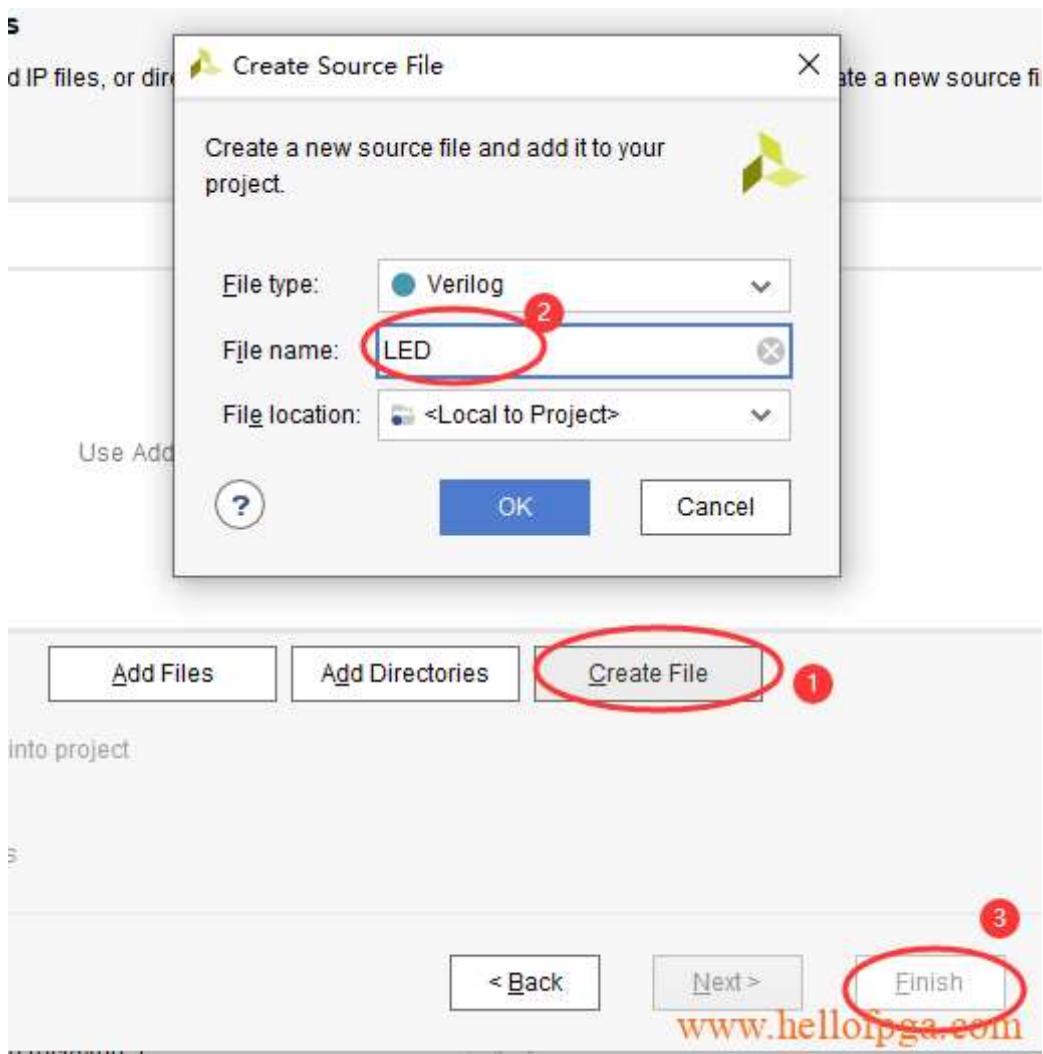


Keep the default in the pop-up dialog

The generated wrapper file is automatically set to the top level

7) Create a new .v file, this article is named LED and copy the code in Project <> (see Project <> for details)





Copy the following code to LED. V

```

module LED(
    input clk,
    output led
);
parameter T1MS = 26'd50_000_000 ; //50M晶振时钟
reg [25:0]time_count=26'd0;//时钟计数器
reg led_r=1'b0;
always@ (posedge clk)
    if(time_count>=T1MS)begin
        time_count<=26'd0;
        led_r<=~led_r;
    end
    else time_count<=time_count+1'b1;
assign led=led_r;
endmodule

```

8) Now that we have the clock (ZYNQ's PS module), we also have the FPGA LED module, next to combine the two modules, here we can create a top-level TOP.V module and then instantiate the ZYNQ and LED modules in the top-level module (the way to create TOP.v is the same as the way to create LED.V above)

Copy the following code to TOP. V

```
`timescale 1ns / 1ps
module TOP(
    output led
);

wire clk;
LED u_1(
    .clk(clk),
    .led(led)
);

ZYNQ_CORE_wrapper u_2(
    .FCLK_CLK0_0(clk)
);

endmodule
```

A closer look reveals that in addition to instantiating the LED and ZYNQ modules, the code also connects the clock of the LED module and the clock output of the ZYNQ module through the line CLK

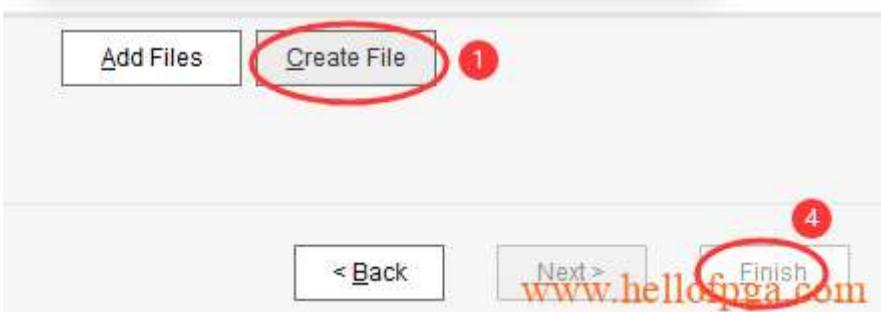
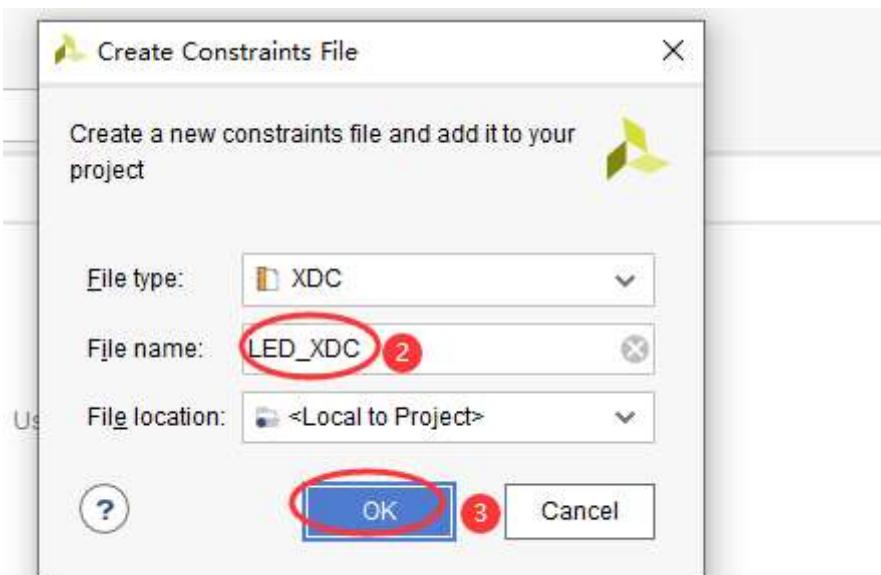
9) Synthesis of procedures



After synthesis, you can see that the topology of the program has changed, and the TOP has become a top-level module

10) In the Source window, create a constraint file





Open the constraint file you just created

LED_XDC.xdc *

```
1: set_property PACKAGE_PIN P20 [get_ports led]
2: set_property IOSTANDARD LVCMS33 [get_ports led]
```

And add the following code to assign the LED pins

```
set_property PACKAGE_PIN P20 [get_ports led]
set_property IOSTANDARD LVCMS33 [get_ports led]
```

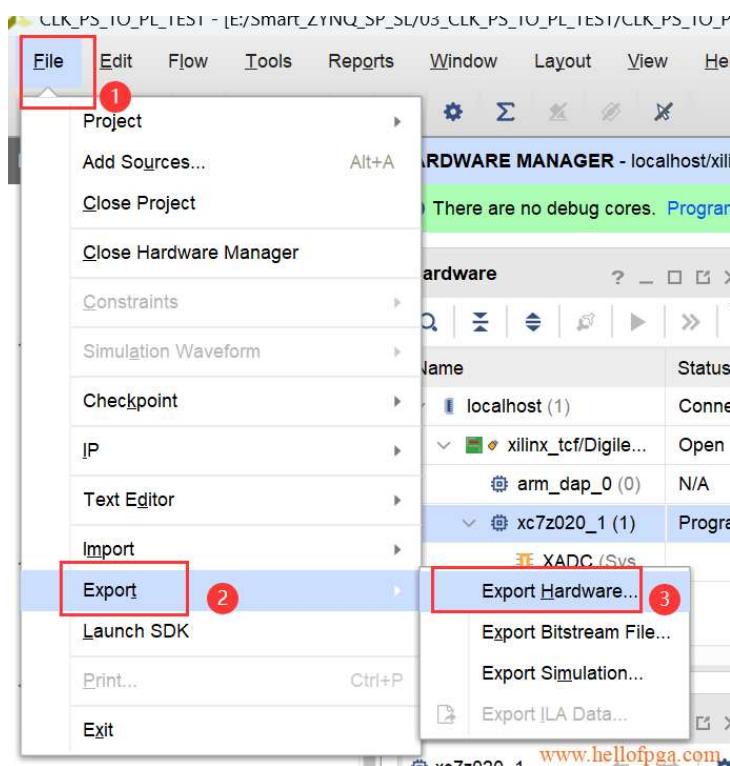
12) After completion, click Generate Bitstream (routing and generating binary files) (it will remind you whether to save the constraint file, just click Confirm)

After the above operations, the hardware PL and the settings and layout of PS have been completed, but because the clock is generated by the PS side, the ARM operation of the PS side can provide the clock signal to the PL. **Directly download the bit stream file to ZYNQ, the PS side will not work, and the clock will not be output to the PL**

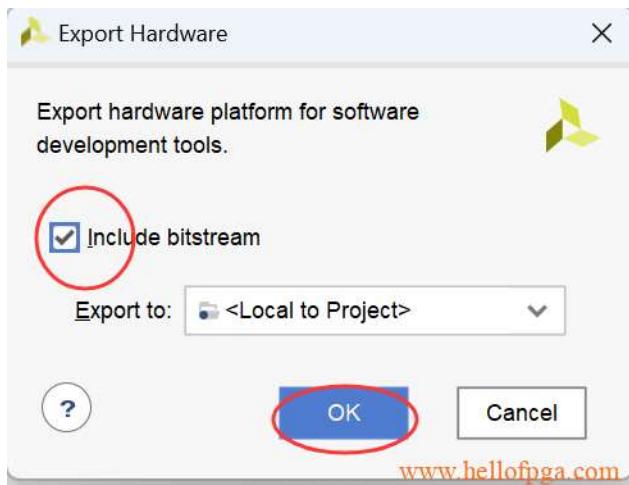
If you want the PS side to work, you need to create a project in the SDK, even if it is an empty project

The following is the process of creating a PS empty project, and the detailed graphic tutorial can refer to Project 6

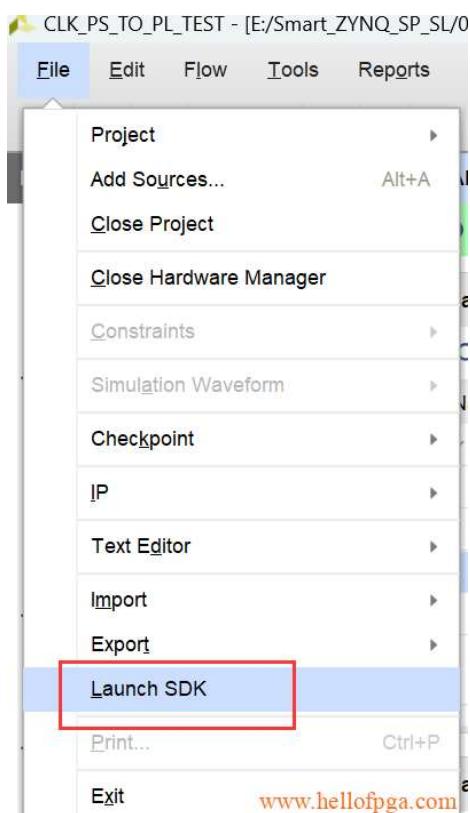
1) vivado Select File→Export→Export hardware..., check "include bitstream" in the pop-up dialog box, and click "OK" to confirm



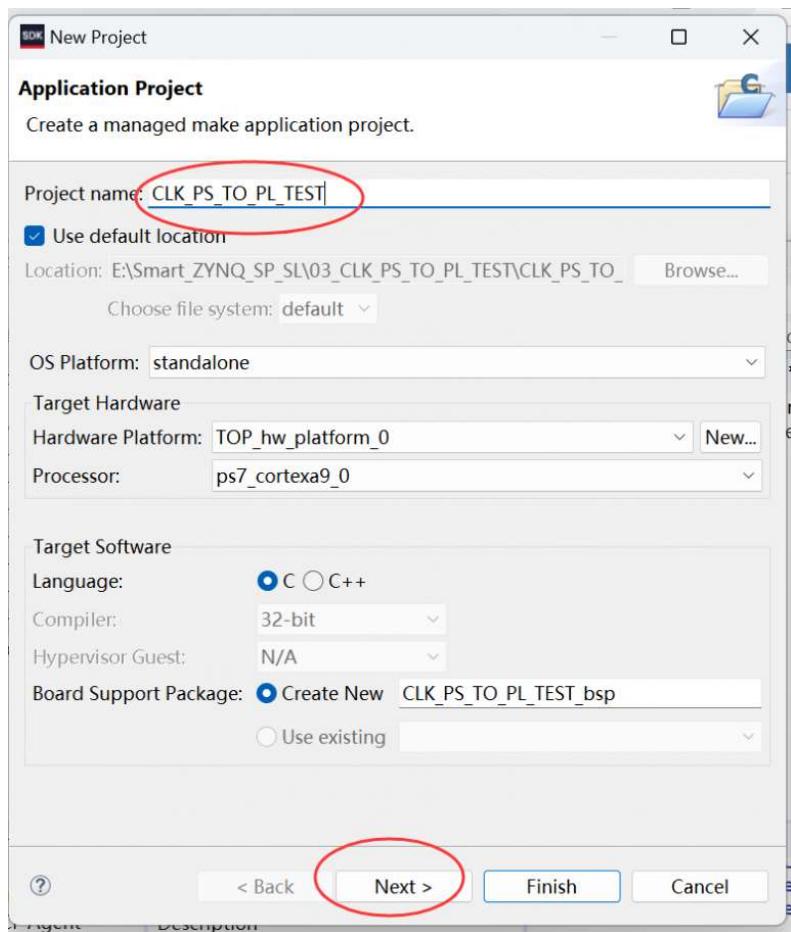
Check "Include Bitstream"



2) File→ Launch SDK, in the pop-up dialog box, save the default, click "OK", the system will open the SDK project



3) Create a new project file→new→Application Project, create a new "Application Project", enter your project name in the new project, and select Empty Application



SDK New Project

Templates

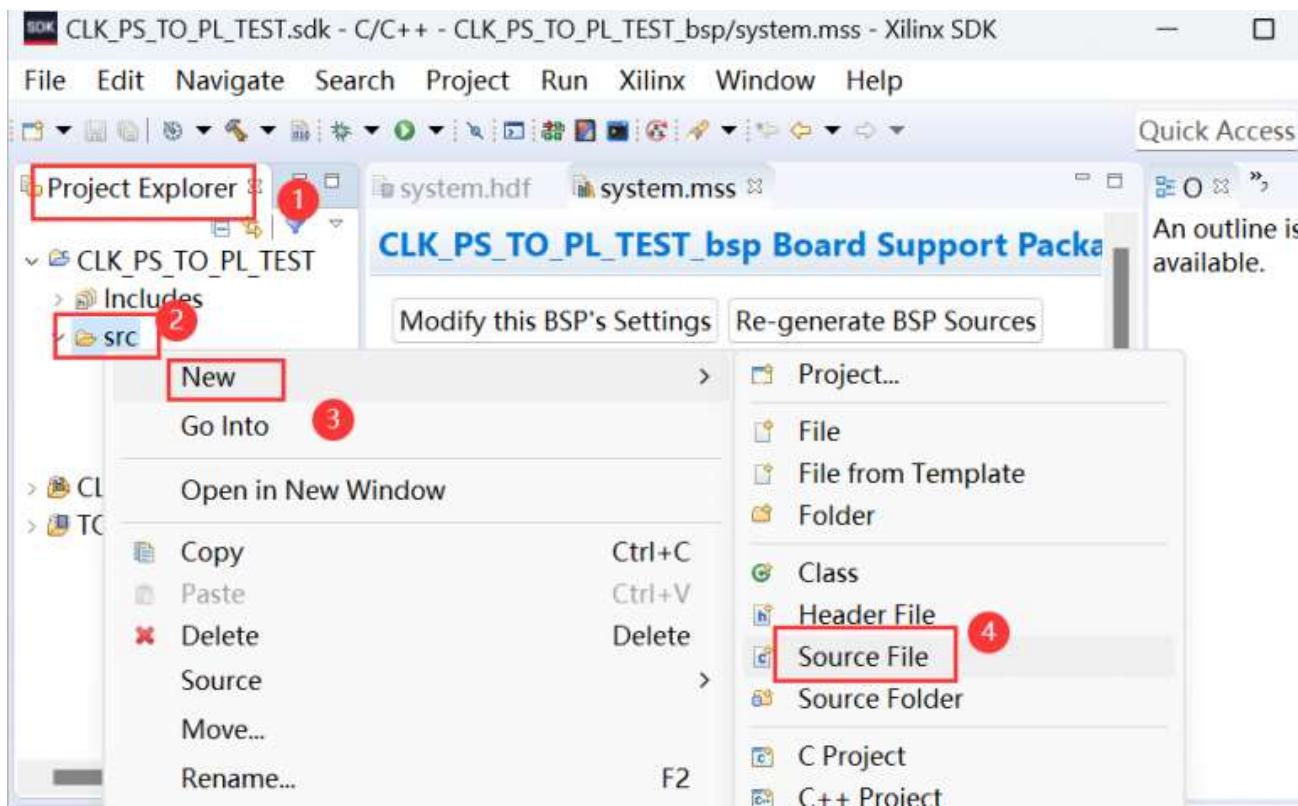
Create one of the available templates to generate a fully-functioning application project.

Available Templates:

- Dhrystone
- Empty Application**
- Hello World
- lwIP Echo Server
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

www.hellofpga.com

And create a main.c file in an empty application



Source File

Create a new source file.

| | |
|----------------|---------------------------|
| Source folder: | LED_TEST/src |
| Source file: | main.c |
| Template: | Default C source template |

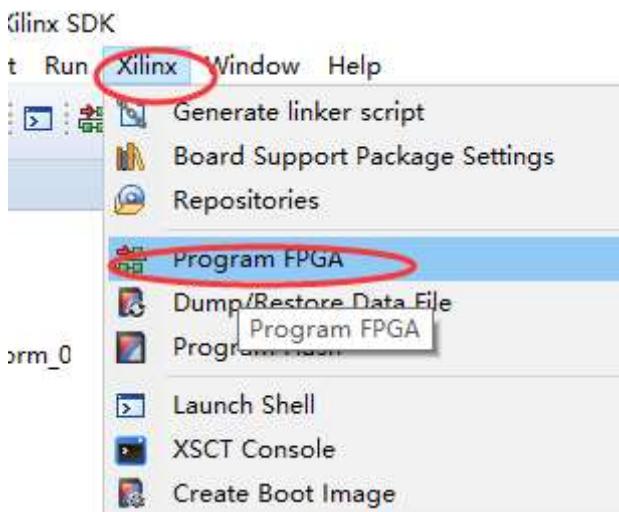
And copy the following code inside (equivalent to creating an empty program)

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

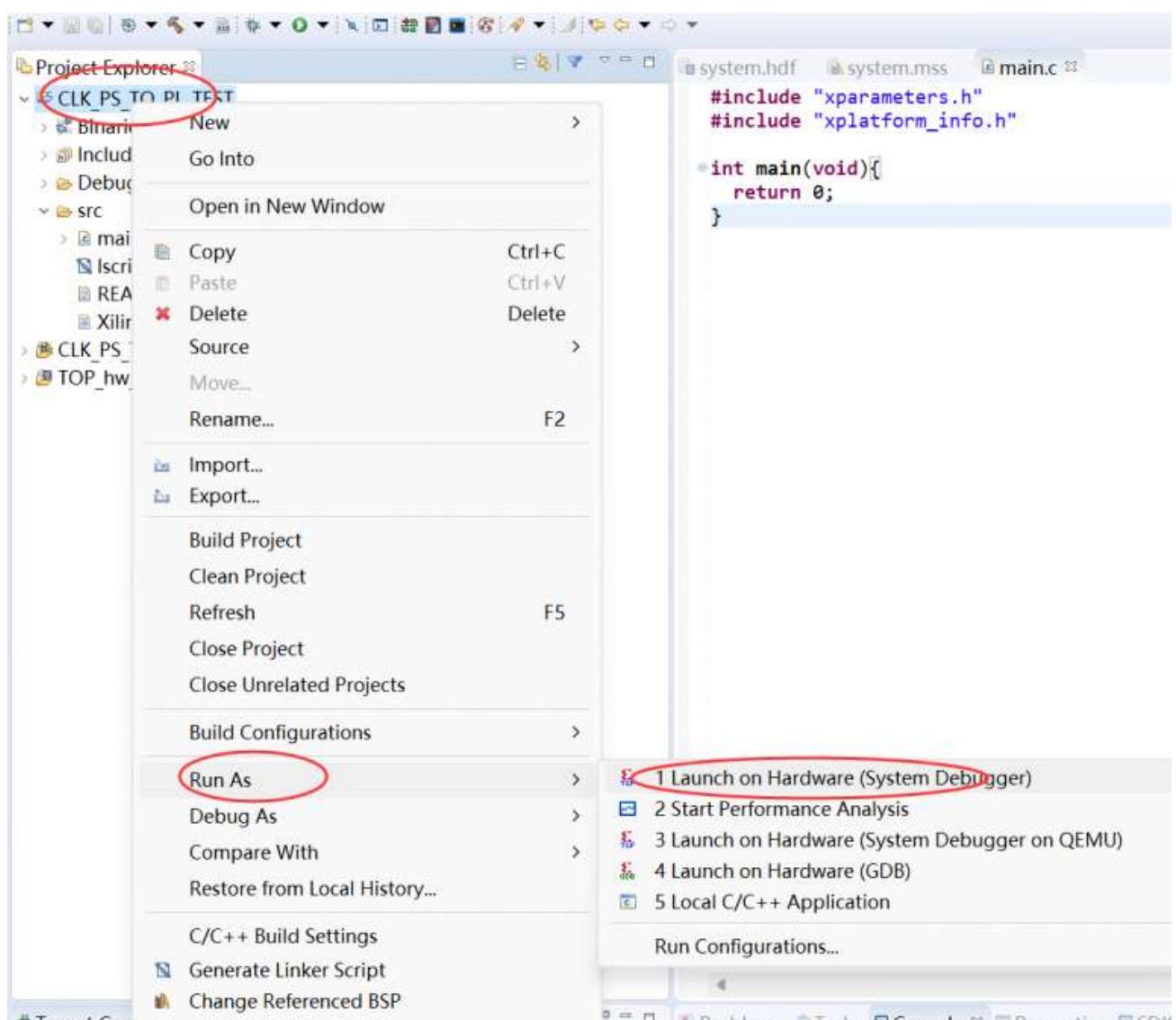
int main(void) {
    return 0;
}
```

Then click Save.

- 4) Program the FPGA with the previously generated binary, -> and click "Program" (at this time, the DONE light is lit, indicating that the FPGA is configured)
- Xilinx Tools
Program
FPGA



5) When the FPGA is successfully programmed, we need to initialize the processor in zynq, right-click on the empty project just created, and select -> or Launch on Hardware (GDB).Run AsLaunch on Hardware (System Debugger)



After the above operation, you can see that the LED light is flashing regularly, proving that our PS has output the clock to the PL normally

Complete Project:

03 CLK PS TO PL TEST XC7Z020

Download



SMART ZYNQ SP & SL