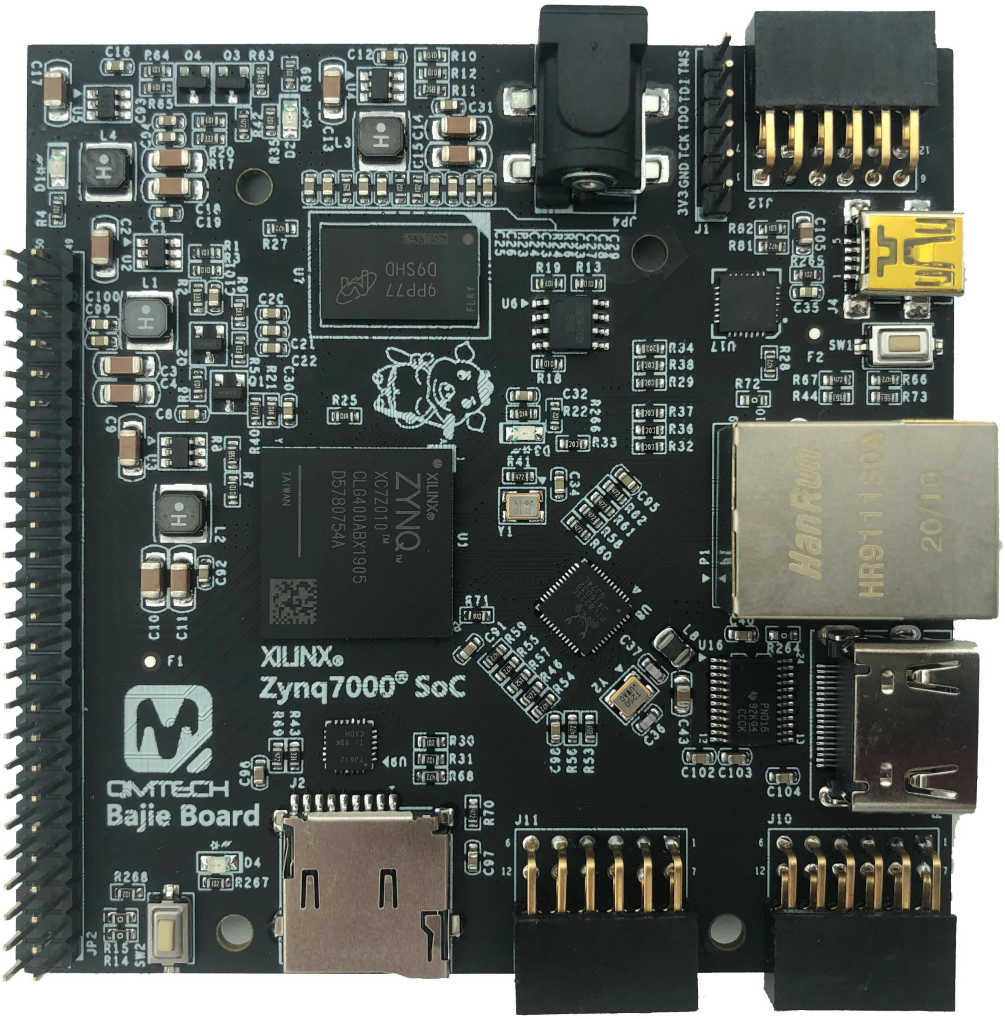
QMTECH ZYNQ7000 BAJIE BOARD

USER MANUAL



Preface

The QMTech® ZYNQ7000 Bajie Board uses Xilinx Zynq®-7000 device which integrates the software programmability of an ARM®-based processor with the hardware programmability of an FPGA, enabling key analytics and hardware acceleration while integrating CPU, DSP, ASSP, and mixed signal functionality on a single device. Consisting of single-core Zynq-7000S and dual-core Zynq-7000 devices, the Zynq-7000 family is the best price to performance-per-watt, fully scalable SoC platform for your unique application requirements.

For more information, updates and useful links, please visit QMTECH Official Website:

http://www.chinaqmtech.com

Table of Contents

1. Introduction

1.1 Document Scope

This demo user manual introduces the non-Linux part test examples that running on the QMTECH ZYNQ7000 Bajie Board. Those examples are all running with Xilinx Vivado 2018.3 environment. So the prerequisites before working with the examples are shown as below:

1. Users have already installed the Vivado 2018.3 in the Windows OS.
2. Users have the basic knowledge about the usage of the Vivado environment. At least know how to synthesis, implement and generate bitstream, etc.

1.2 Test Examples

Below diagram shows the main parts that these test examples cover:

- PS side MIO(Multipurpose Input Output);
- PL side HDMI Display;
- PL side EMIO(Extendable Multipurpose Input Output);
- PS side UART;
- PS side RGMII ethernet Interface;
- PS side DDR3 Memory Controller;

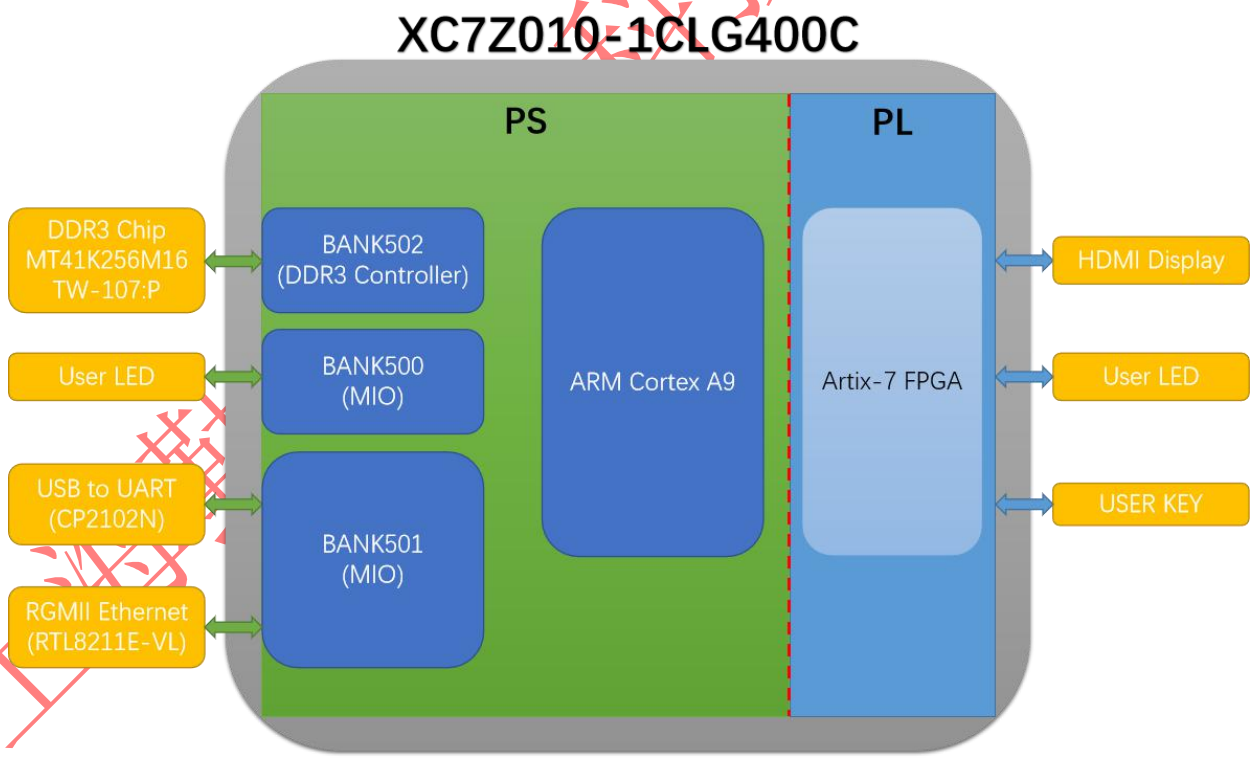


Figure 1-1. Tested Peripherals

1.3 Bajie Board Setup

Before start to test the Zynq7000 Bajie Board, users need to prepare the hardware setup shown as in below image. In default, the factory binary test images are already stored in the micro SD card and the D3 LED will be periodically blinking.

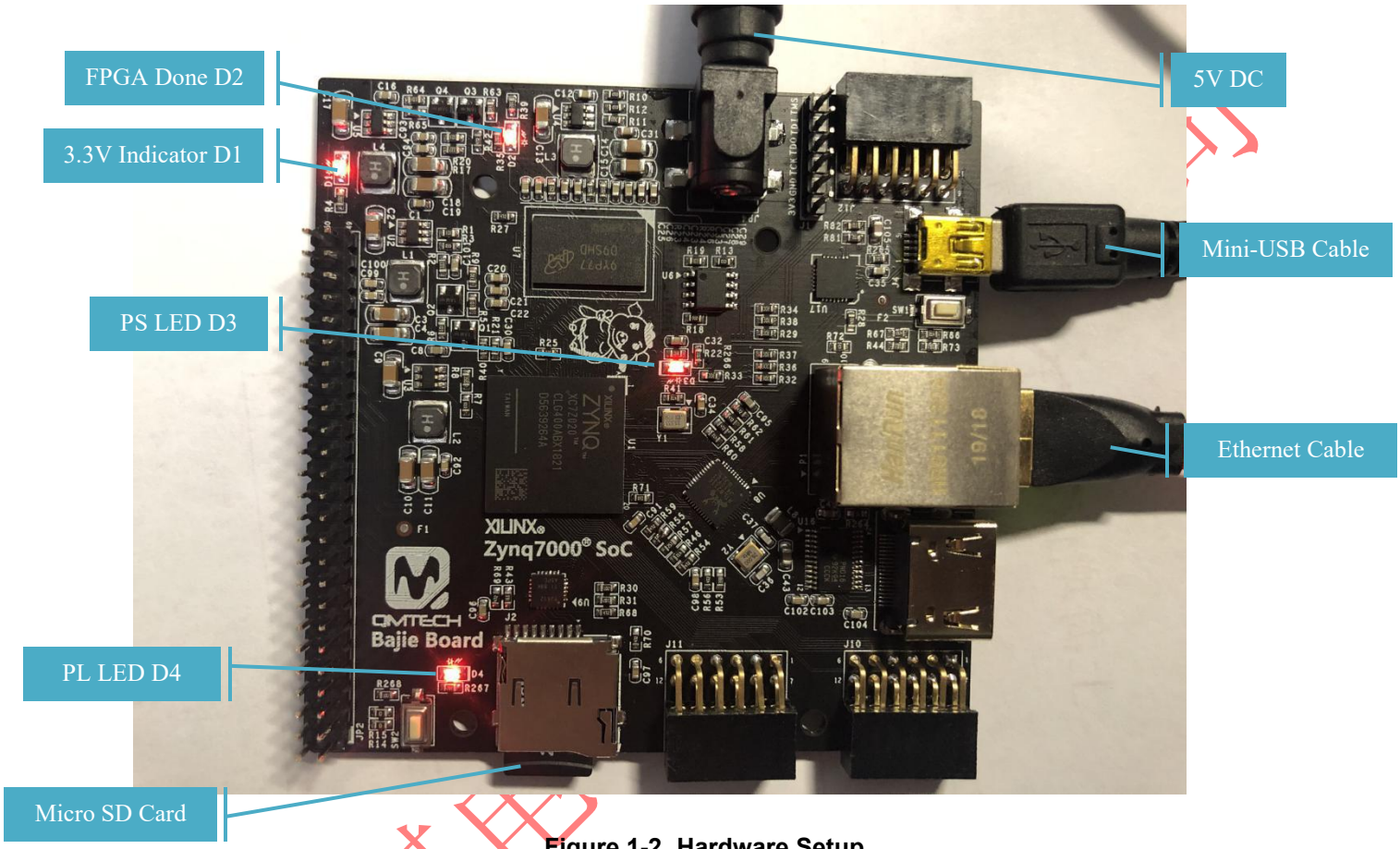


Figure 1-2. Hardware Setup

2. Getting Started

This chapter describes the detailed steps to create a customized ZYNQ Processing System. Comparing to the existing ZYNQ development board e.g. Xilinx ZC702, there are many differentiations in the QMTECH ZYNQ7000 Bajia Board. For example, there's only one 16bit width DDR3 memory chip connected to PS ARM core.

2.1 Steps to Customize the ZYNQ Processing System

2.1.1 Step 1: Create New Project

Open Vivado 2018.3, then click【Create Project】shown in below figure.

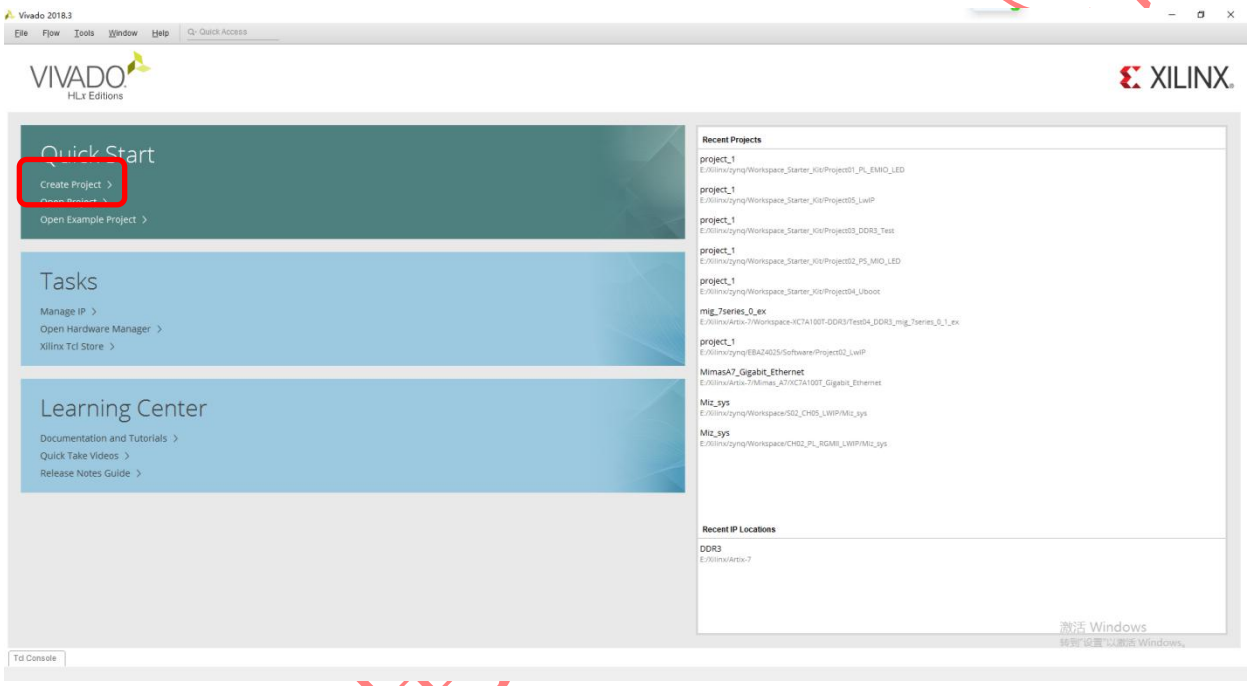


Figure 2-1. Vivado 2018.3

Below image will be shown and click【NEXT】button:

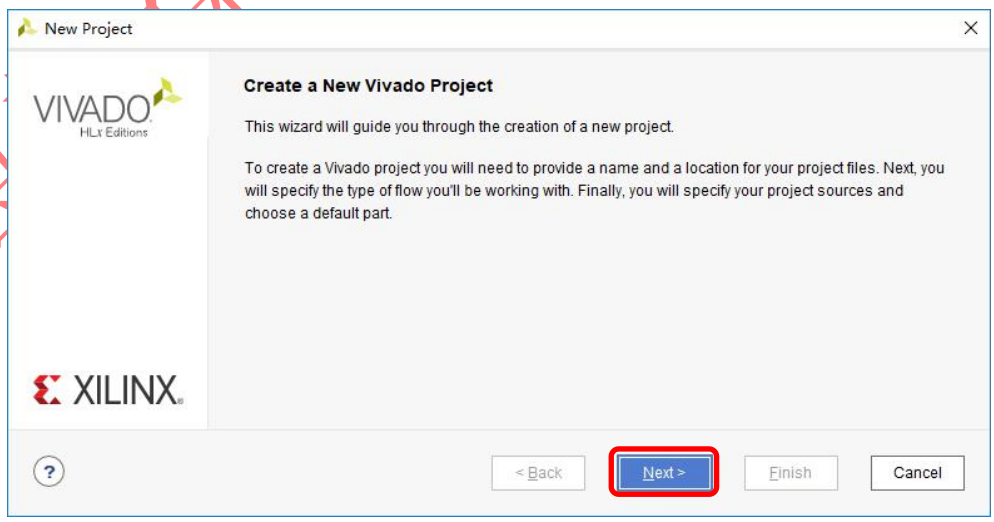


Figure 2-2. Create New Project

Set the Project name and the project location:

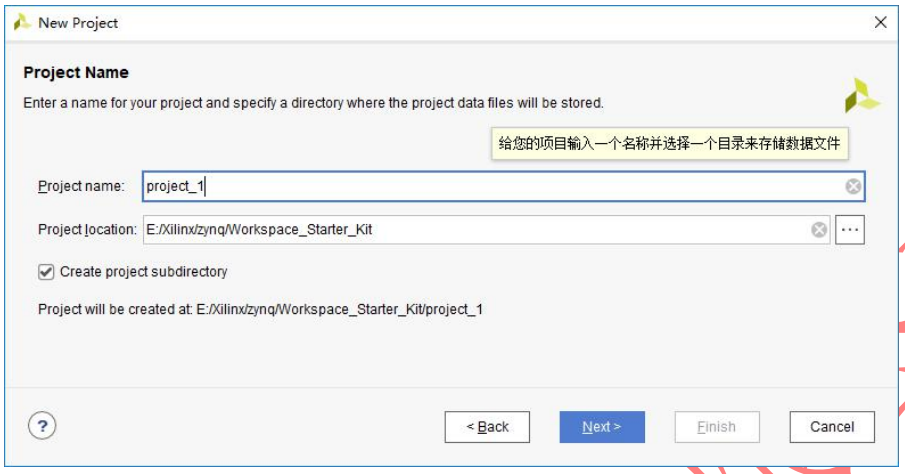


Figure 2-3. Set Project Name and Location

Click【Next】button in below image:

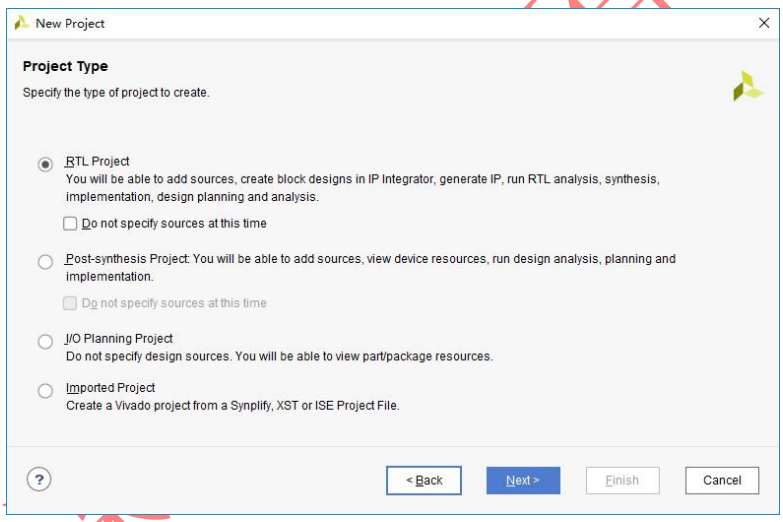
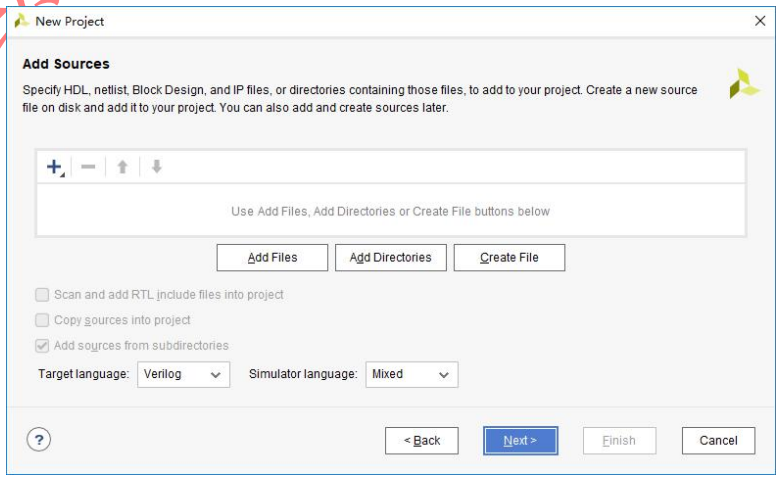


Figure 2-4. Click Next

Click【Next】button in below image if there's no source file existing:



Click 【Next】 button in below image if there's no constraint file existing:

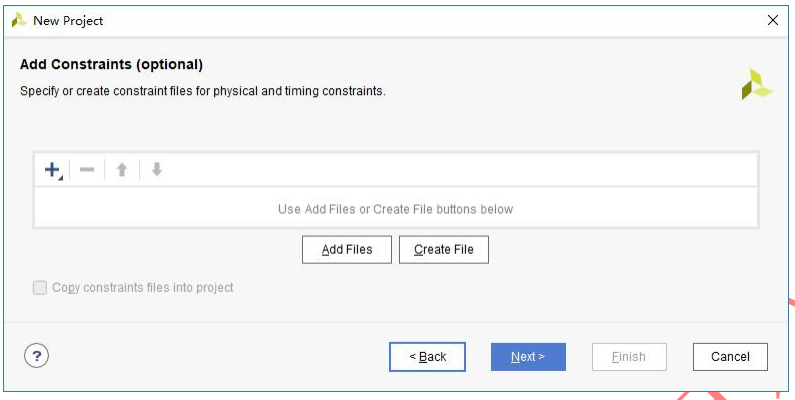


Figure 2-5. Click Next

Select the device consistent to the chip mounted on QMTECH ZYNQ xc7z010clg400-1:

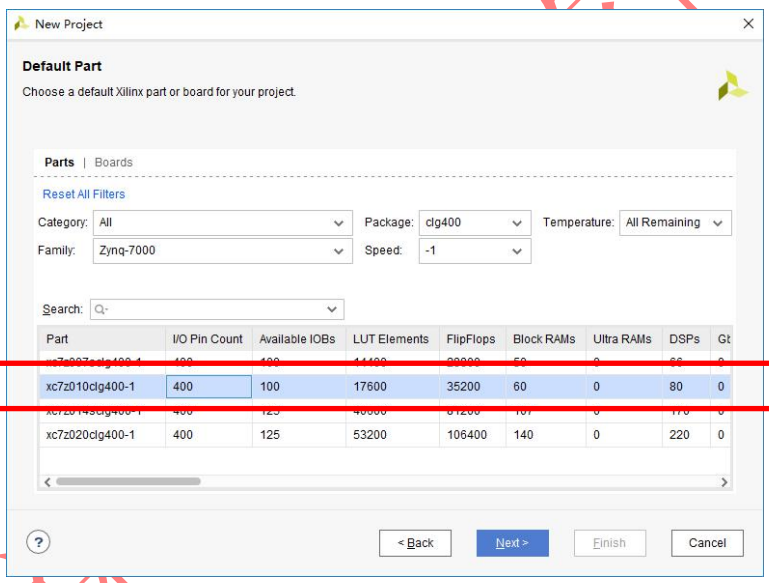
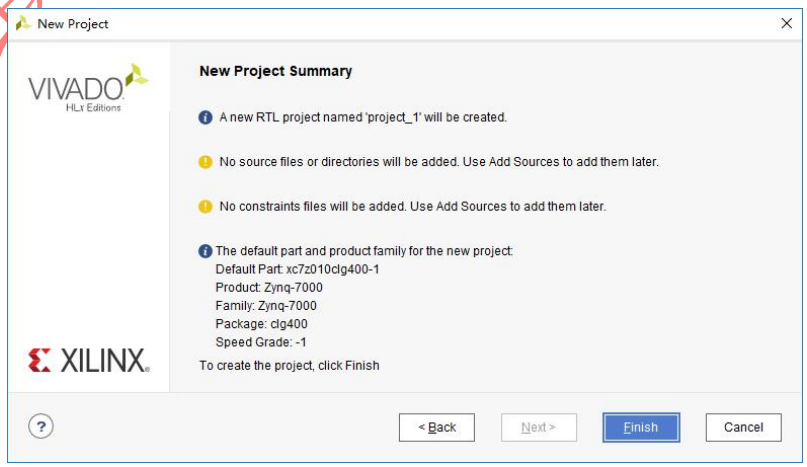


Figure 2-6. Select Device

Click 【Finish】if there's nothing needs to be changed.



2.1.2 Step 2: Add Zynq Processing System

After the project successfully created, users can start to add the Zynq PS to the design by click【Create Block Design】in the IP INTEGRATOR. Type example name 'system' in the edit box shown in below image.

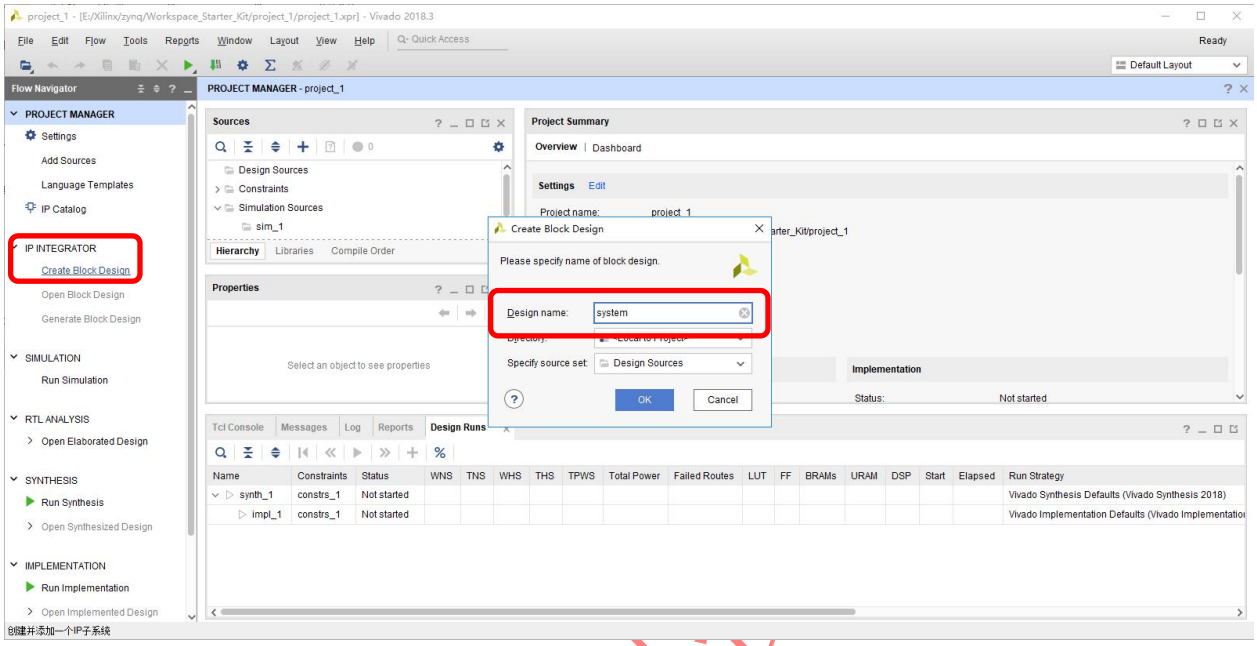


Figure 2-7. Create Block Design

Click the 【+】button shown in below image and Search the keyword ZYNQ in the edit box. If the ZYNQ7 Processing system can be found, users could double click it and add it in the design.

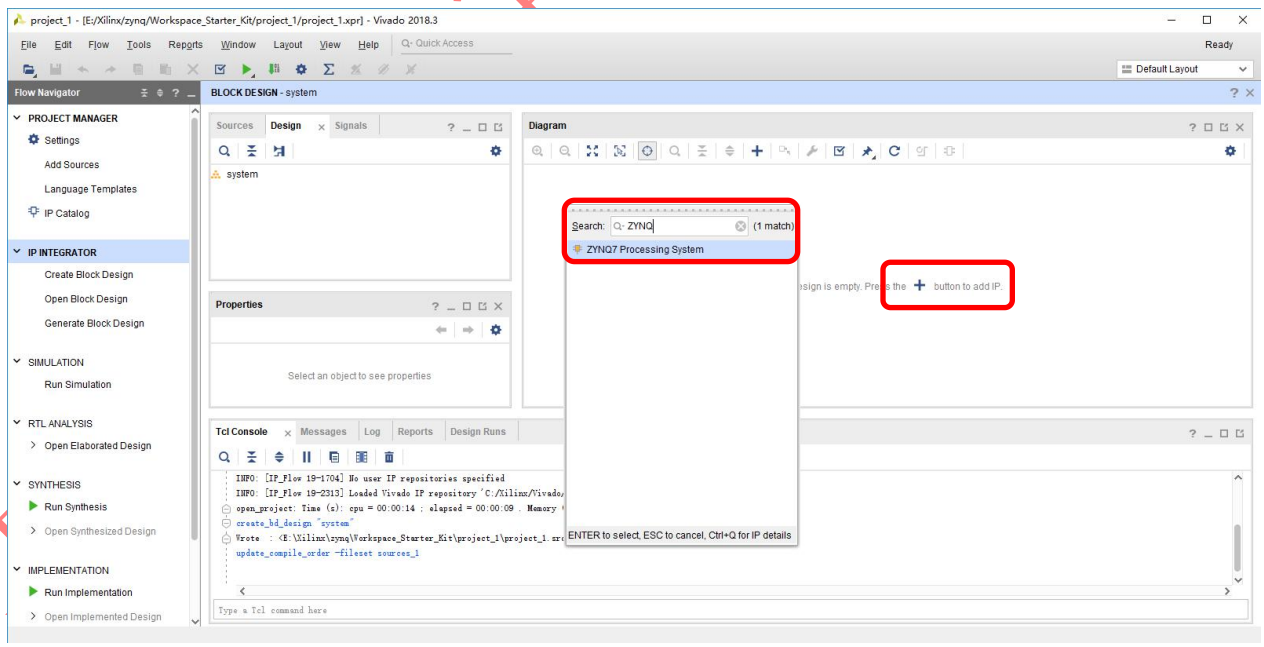


Figure 2-8. Add ZYNQ7 PS

The below image will be displayed once the ZYNQ7 is correctly added.

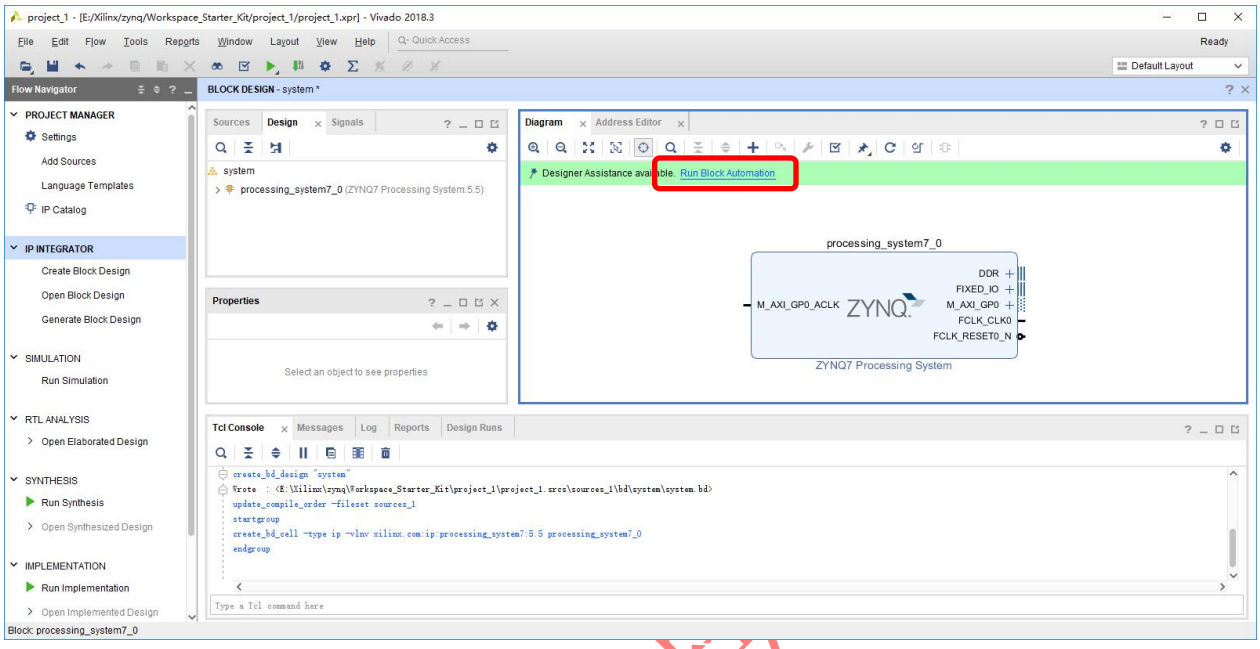


Figure 2-9. Added ZYNQ7 PS

Users could click the 【Run Block Automation】. And then connect PS clock 【FCLK_CLK0】to 【M_AXI_GP0_ACLK】.

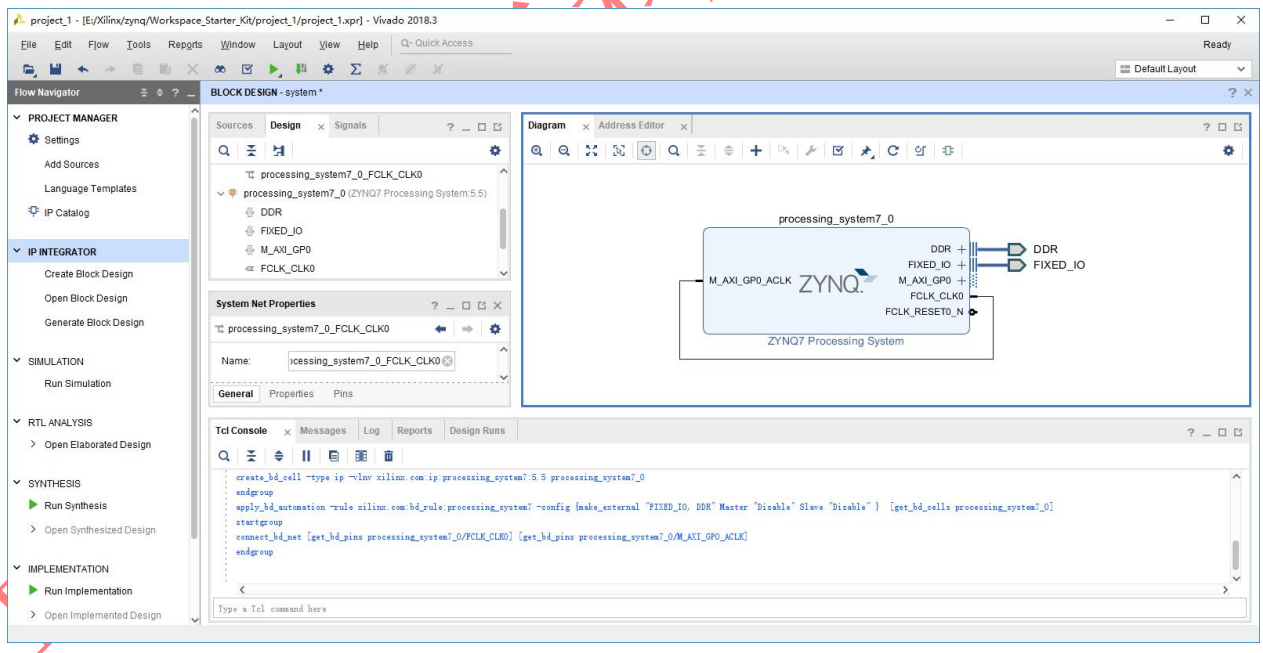


Figure 2-10. Block Automation

2.1.3 Step 3: Customize Zynq Processing System

Double click the ZYNQ IP shown in the above image to implement the detailed customizations. The architecture block diagram will be shown as below. All the configurable parts are all listed in the Page Navigator.

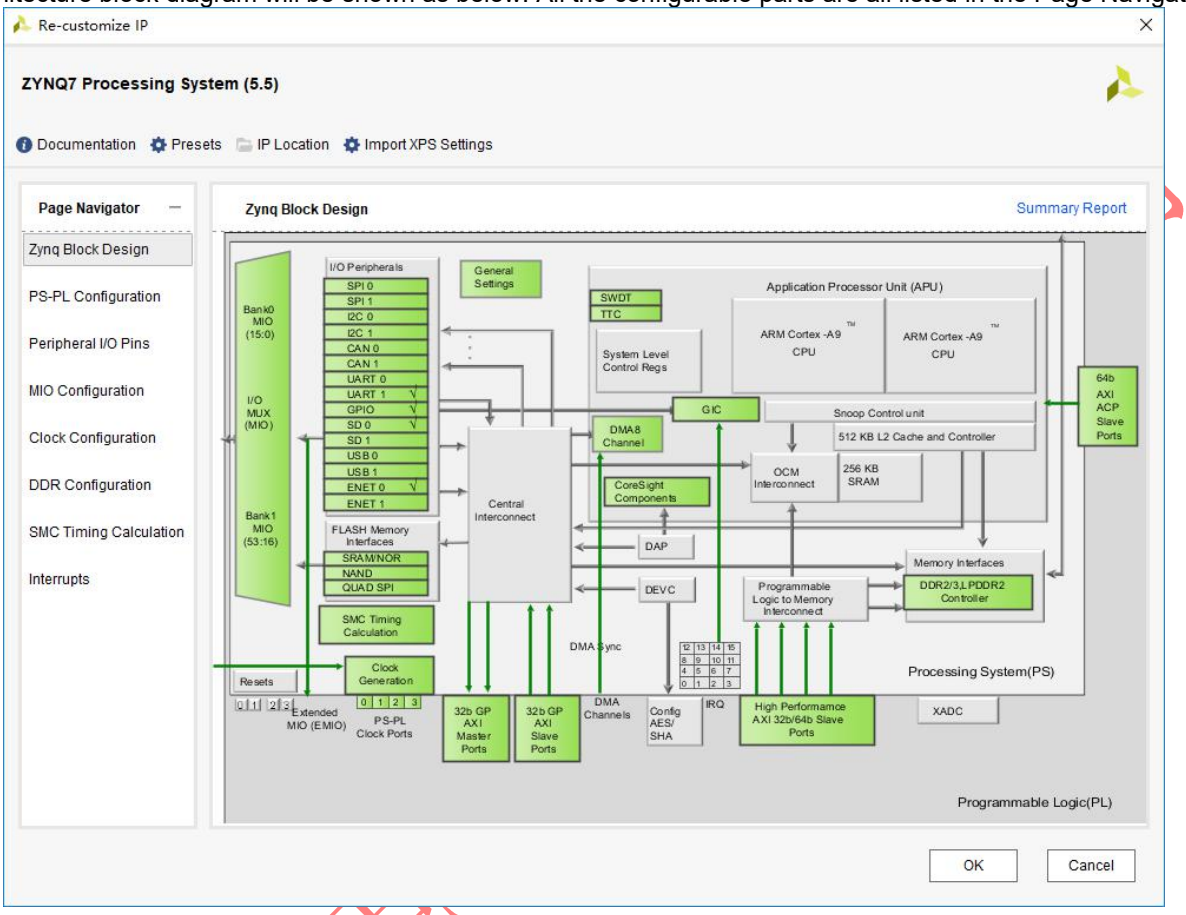


Figure 2-11. ZYNQ Architecture

Click the **PS-PL Configuration** page, nothing needs to be changed here.

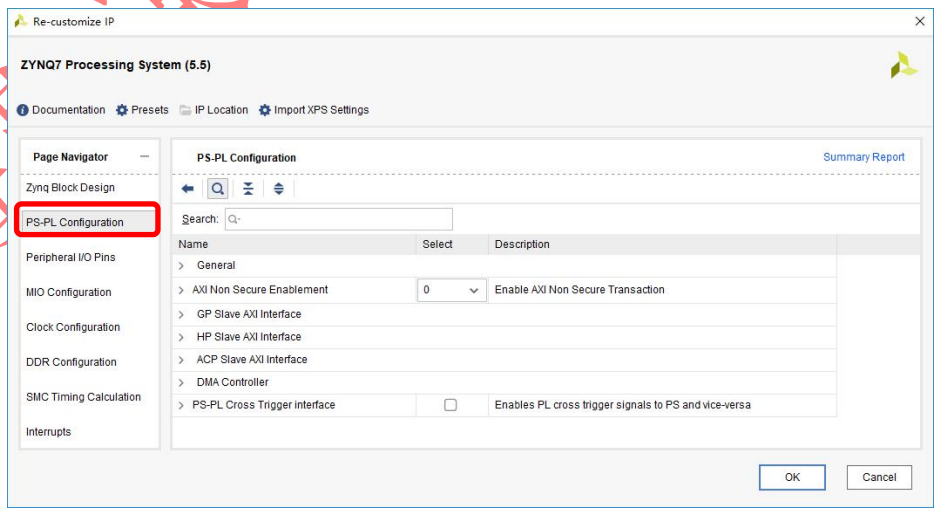


Figure 2-12. ZYNQ Architecture

Click the **Peripheral I/O Pins** page. Configure the Ethernet 0 shown as below. Assign the Ethernet 0 I/Os on the MIO[16:27]and MDIO I/Os on MIO[52:53].

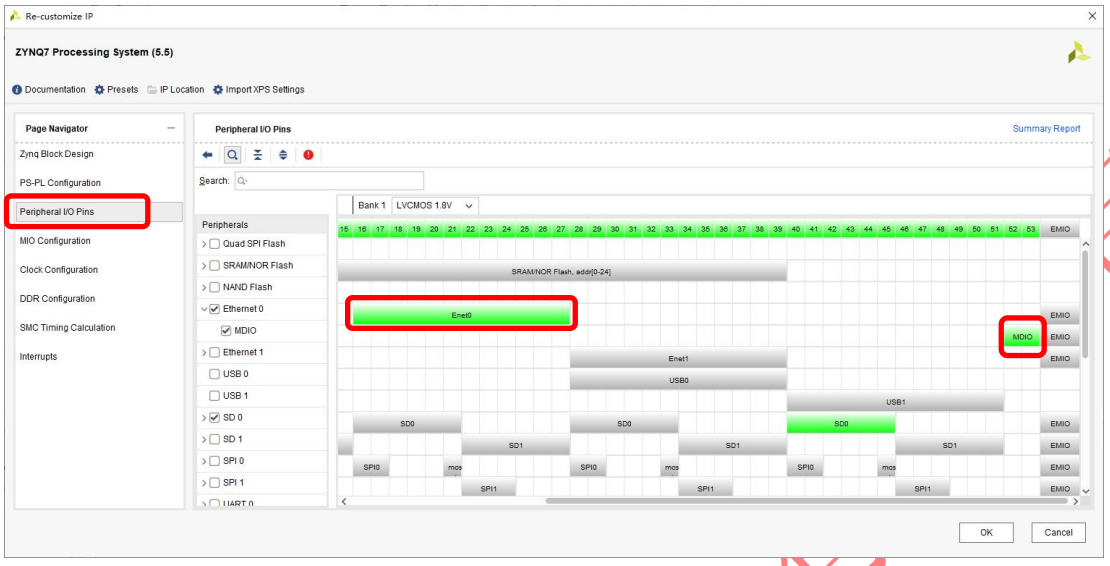


Figure 2-13. Configure Ethernet 0

Configure the SD 0 shown as in below image. The SD card slot is connected to MIO[40:45] and the SD detect signal is connected to MIO47.

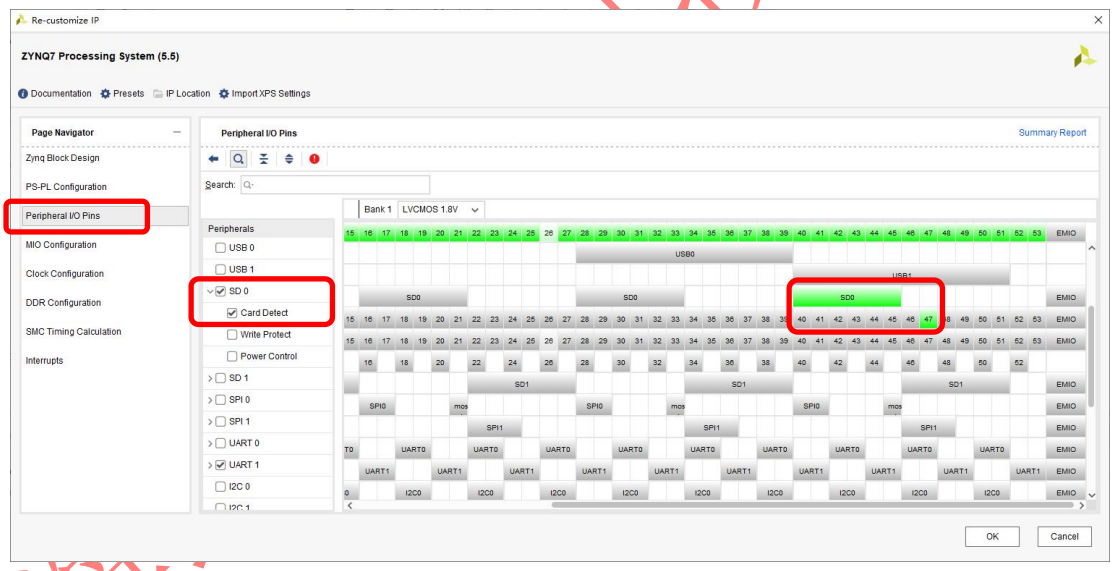
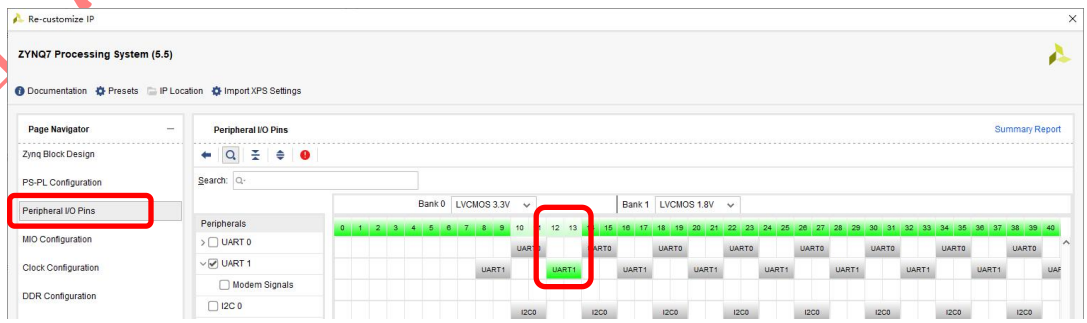


Figure 2-14. Configure SD 0

Below image shows the UART 1 port is connected to MIO12 and MIO13.



Click **MIO Configuration** page. Make sure all the MIO/EMIO assignment are same to the configurations shown in below image.

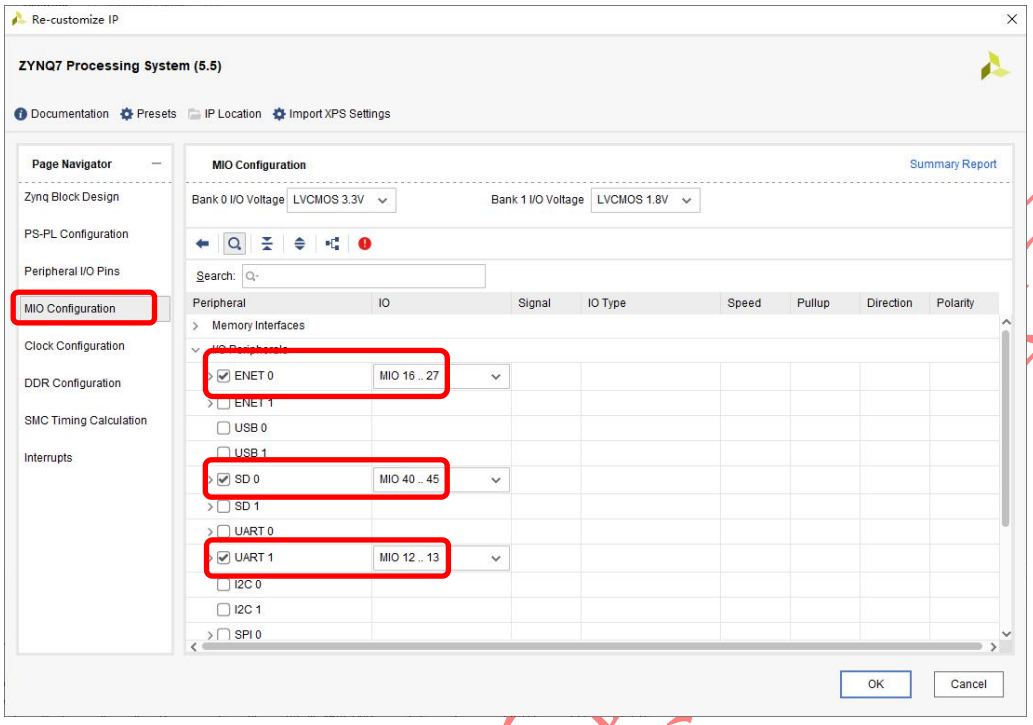
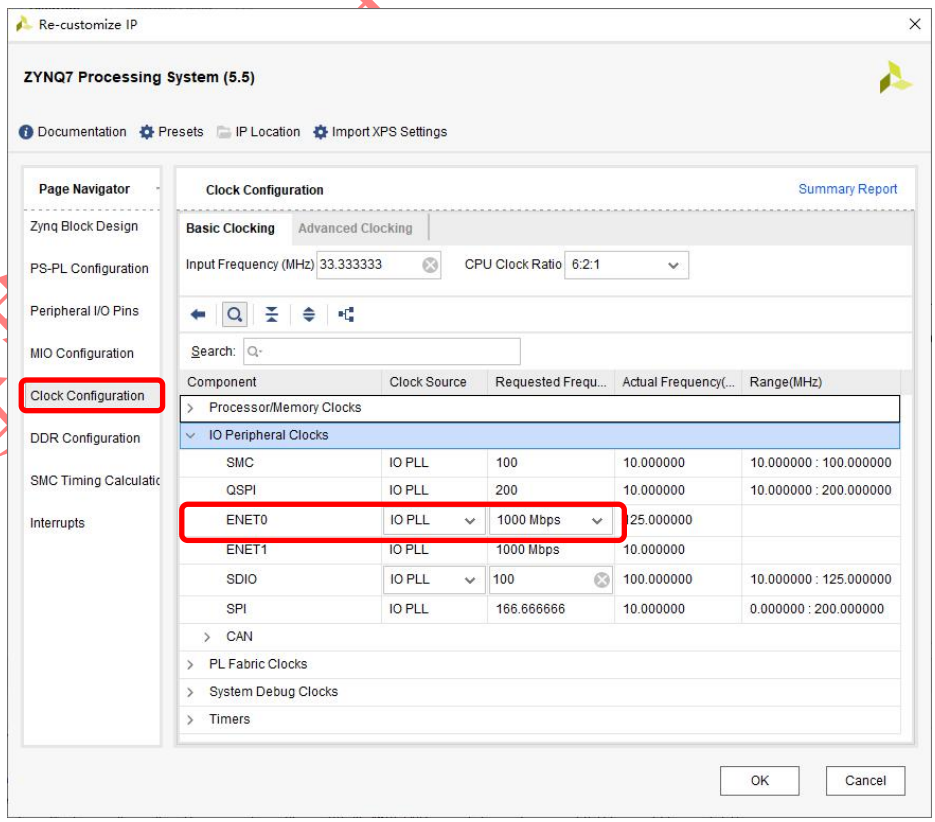


Figure 2-15. MIO/EMIO Assignment

Click **Clock Configuration** page. Configure the ENET0 clock with 1000Mbps because the RTL8211 supports RGMII 1000Mbps ethernet work mode.



Click **DDR Configuration** page. Change the Effective DRAM Bus Width into 16 Bit and select MT41K256M16 RE-125 as the memory part, though the actual part mounted on the ZYNQ Bajie Board is MT41K256M16TW-107:P. This is still workable because the clock frequency is only 400MHz and all the DDR3 memory chips are down speed compatible. If users still worry about the stability thing, then DDR3 memory customization also could be done in this page. And detailed parameters like the CAS latency, RAS to CAS Delay, etc. could be retrieved from Micron DDR3 data sheet and filled in this DDR Configuration page.

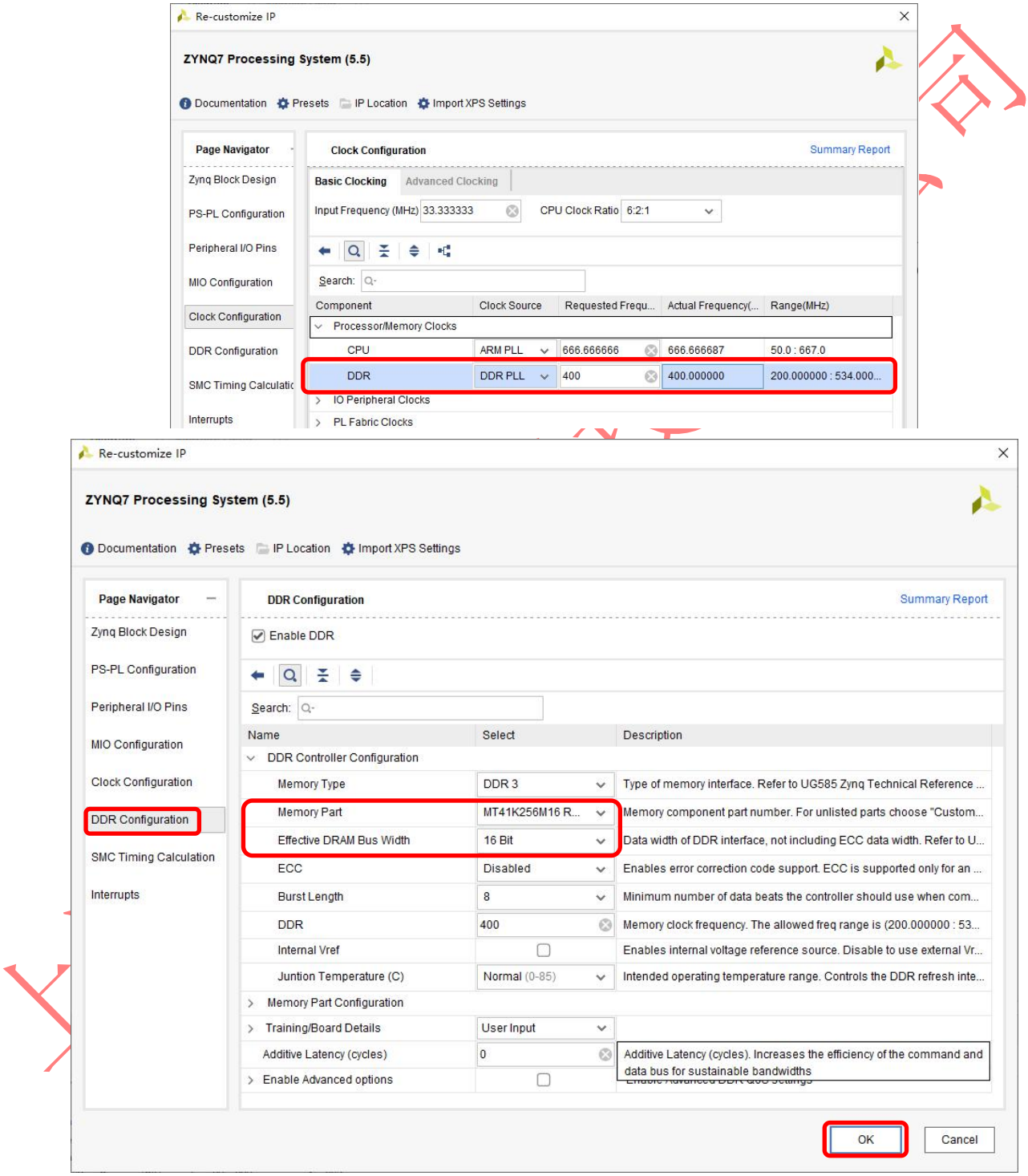


Figure 2-16. DDR3 Memory Configuration

For the **SMC Timing Calculation** and **Interrupts** pages, nothing needs to be changed here. And then click the OK button to finish the whole ZYNQ Processing System customization.

2.1.4 Step 4: Generate Output Products

After the ZYNQ PS customization finished, users still need to make the PINs external.

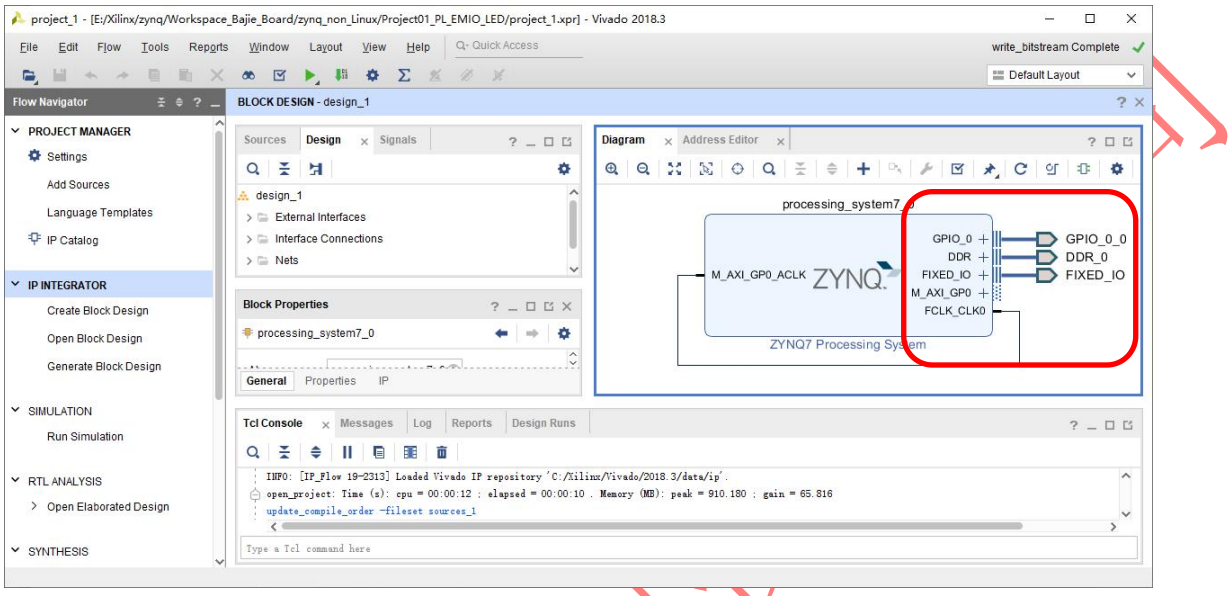
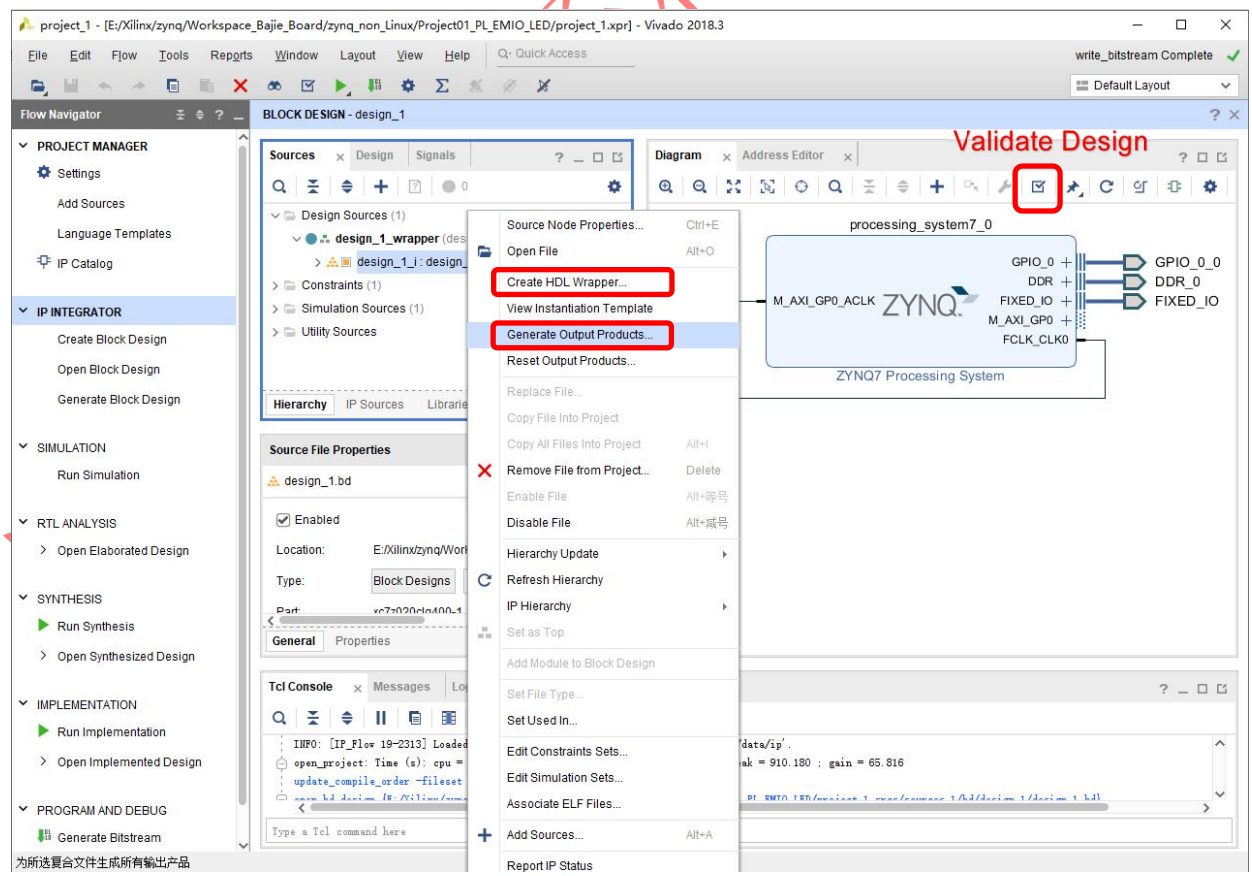


Figure 2-17. Make External

Then **Validate Design**, **Generate Output Products** and **Create HDL Wrapper** to finish the whole procedure.



3. Experiment 1: EMIO User LED

Before start to test the user LED connected to EMIO, users need to make sure the designer_1_wrapper is correctly generated in the previous step. And then execute the whole compilation procedure: **Run Synthesis**, **Run Implementation** and **Generate Bitstream**. Make sure there's no error generated in any of these three steps.

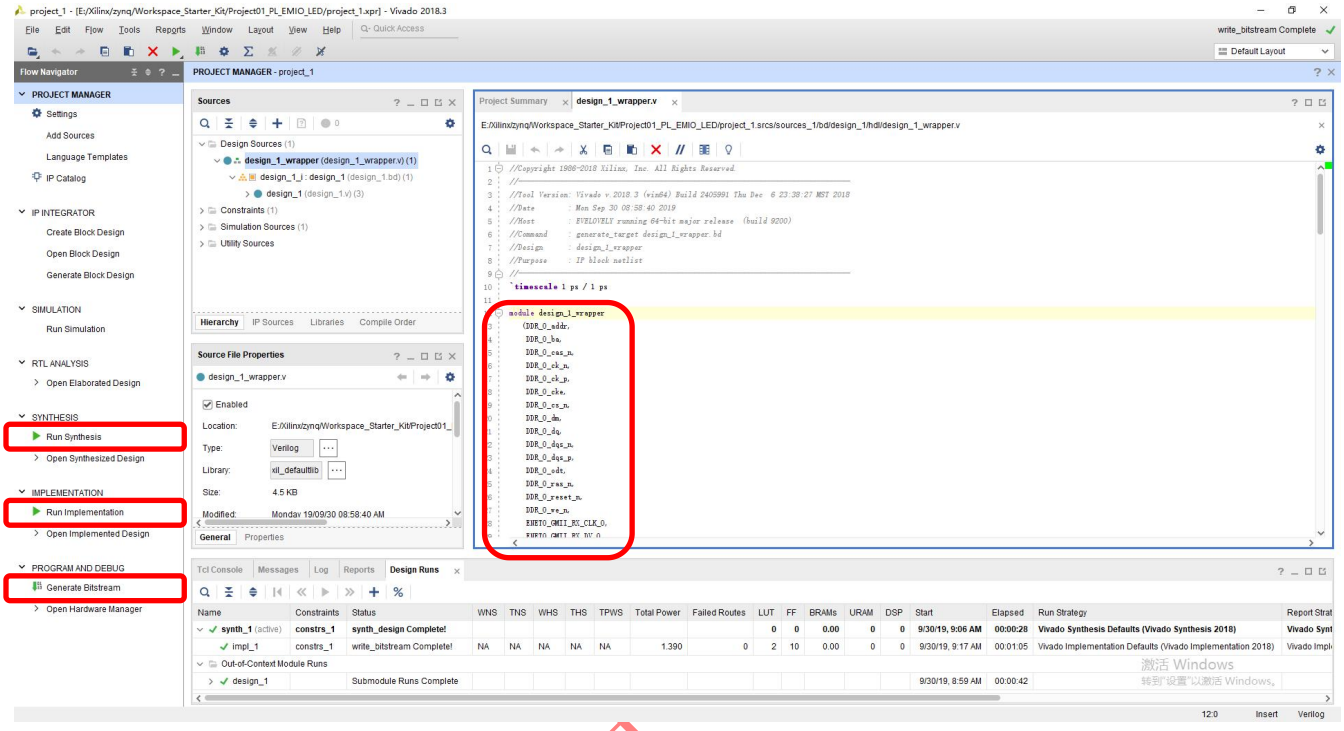
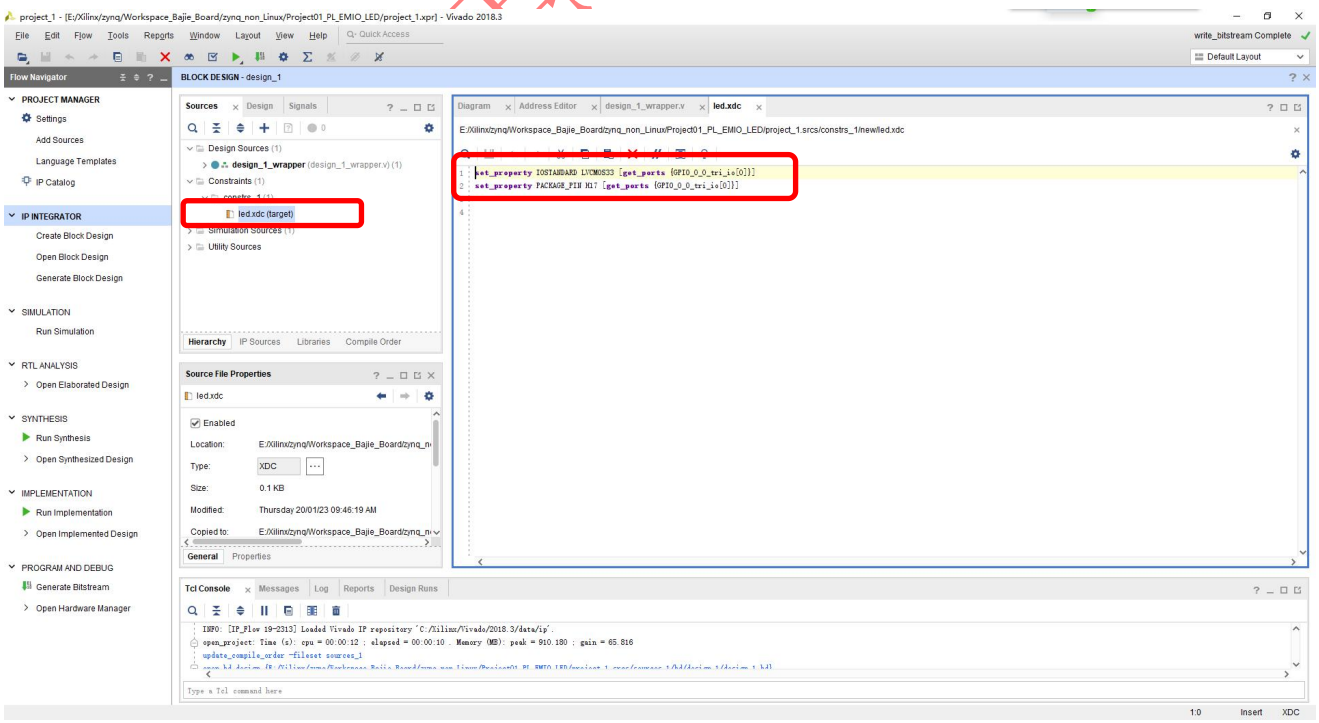


Figure 3-1. Generate Bitstream

Below image shows the constraint file of this project. Users may modify this constraint file if needed.



Then export hardware to the SDK by clicking【File】→【Export Hardware】, select the checkbox **Include bitstream** and click the OK button. The detailed procedure is shown as below:

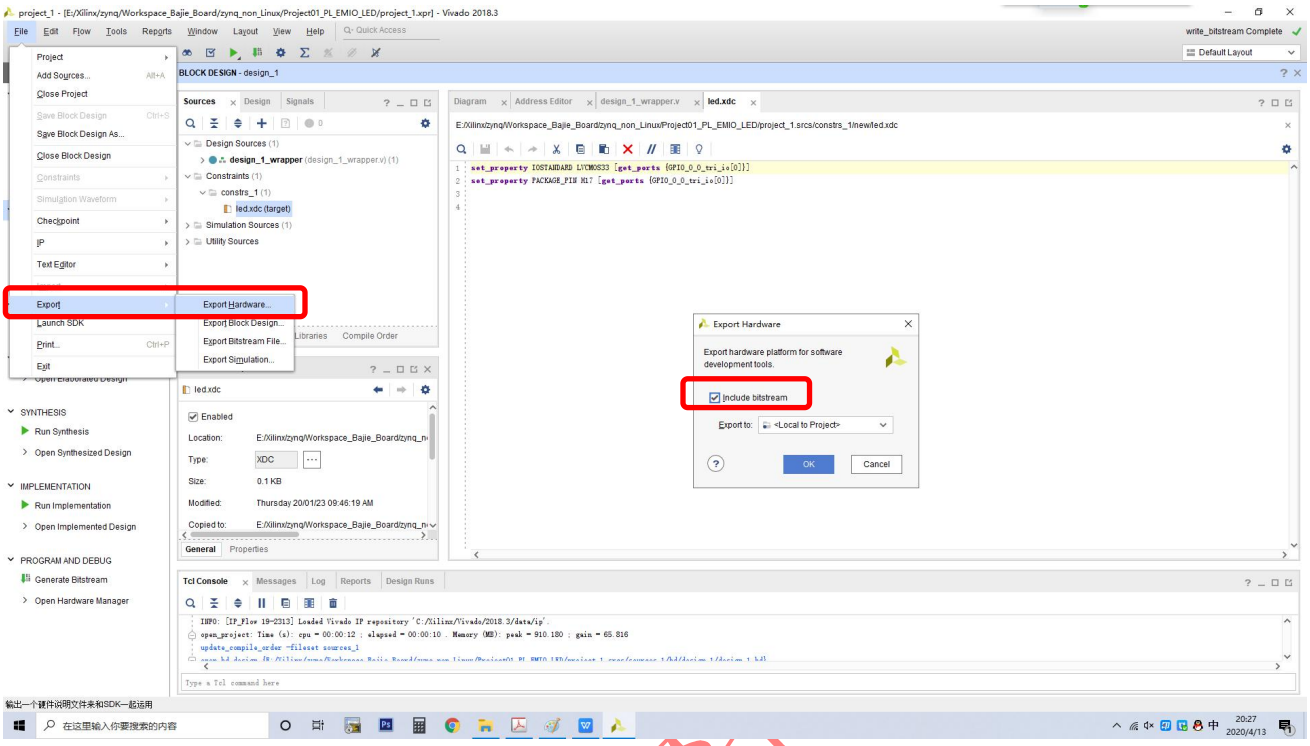


Figure 3-2. Export Hardware to SDK

Start the SDK by clicking 【File】→【Lunch SDK】and then clicking the OK button:

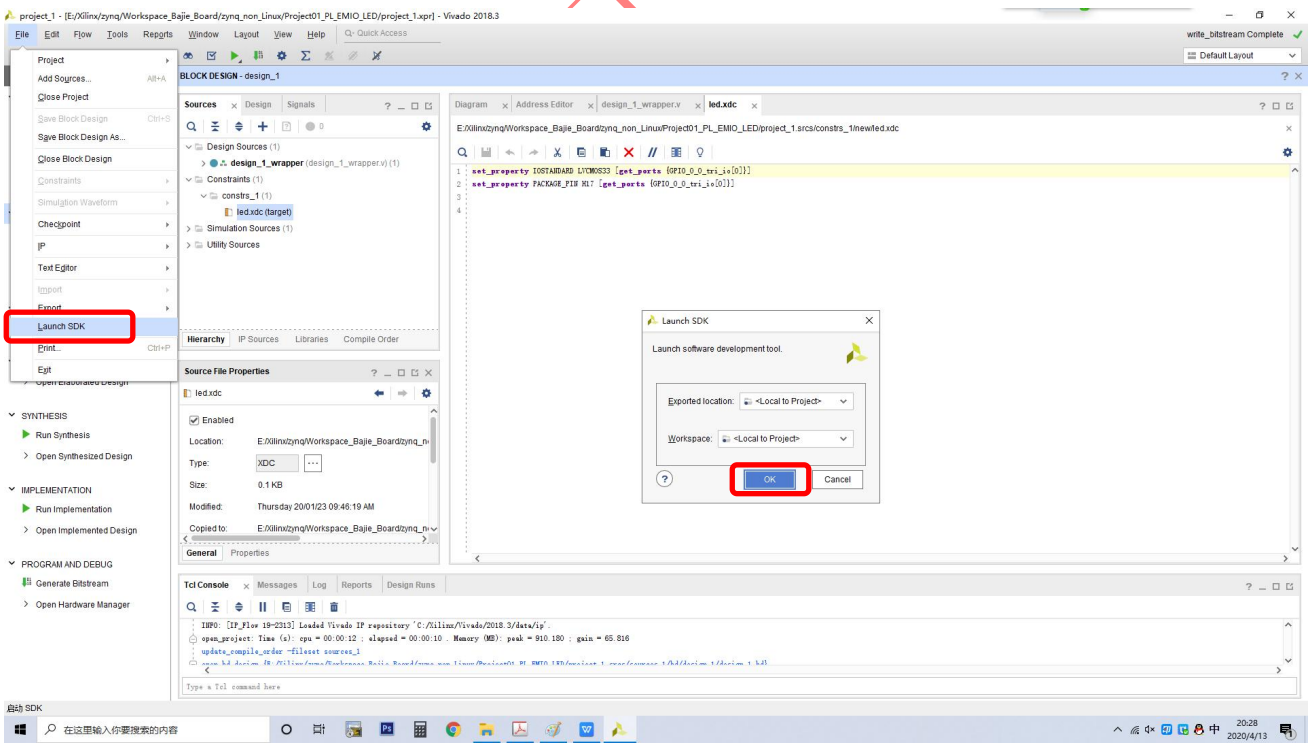


Figure 3-3. Start SDK

Setup a new project for testing the EMIO by clicking 【New】→【Application Project】, and type EMIO_Test in the project name.

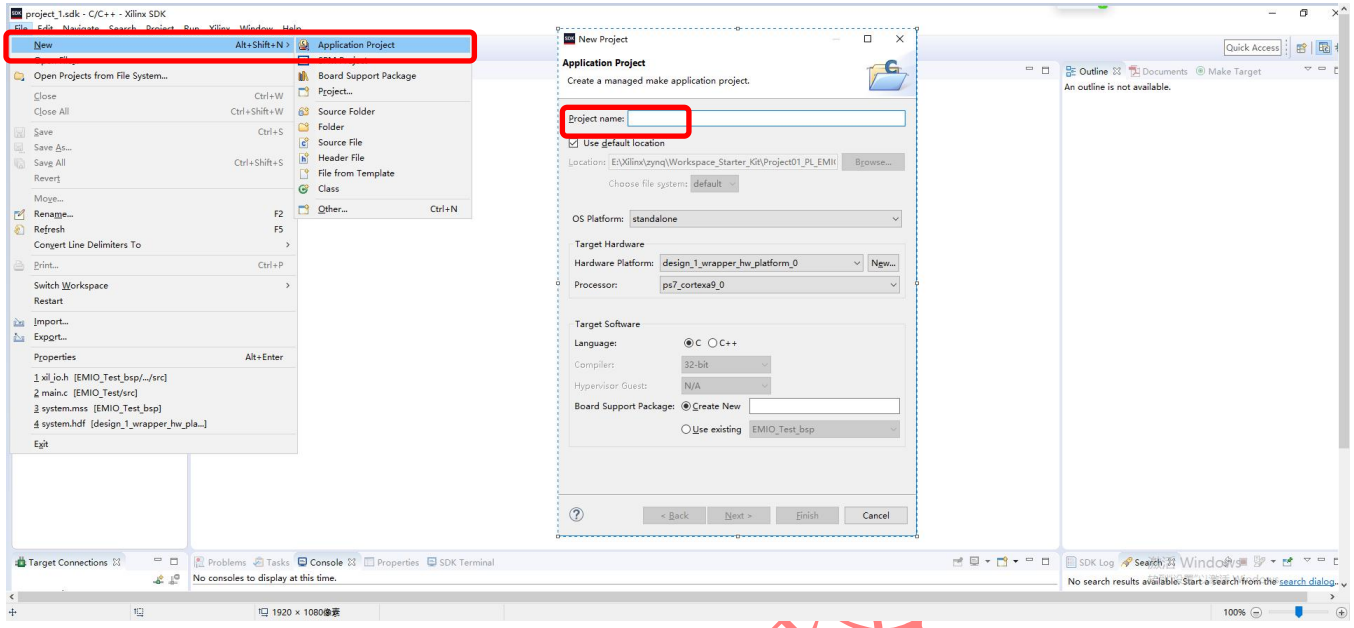
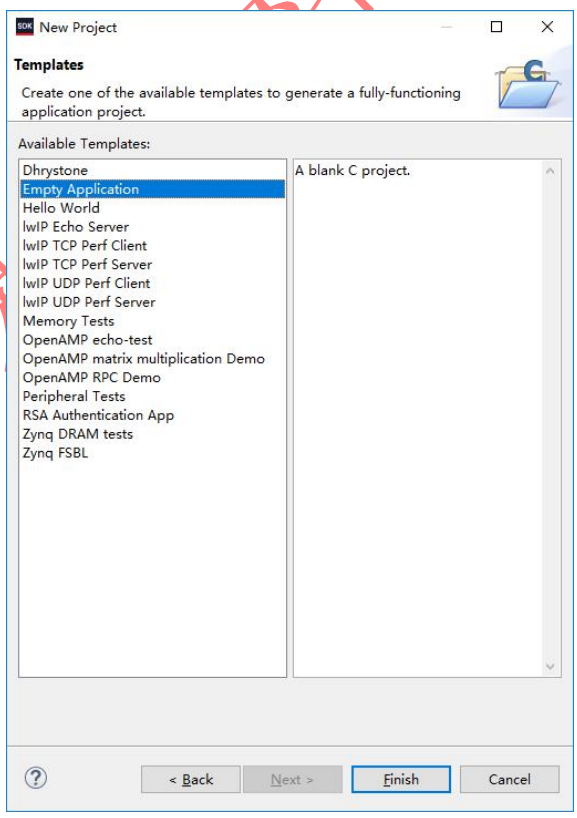


Figure 3-4. Create New Project

Choose Empty Application and click the OK button.



Add source file main.c in the EMIO_Test/src folder and the project will be automatically built by the Xilinx SDK environment. The below test routine configures the EMIO pin H17 as the output pin and enable it. After that it will periodically toggle the pin.

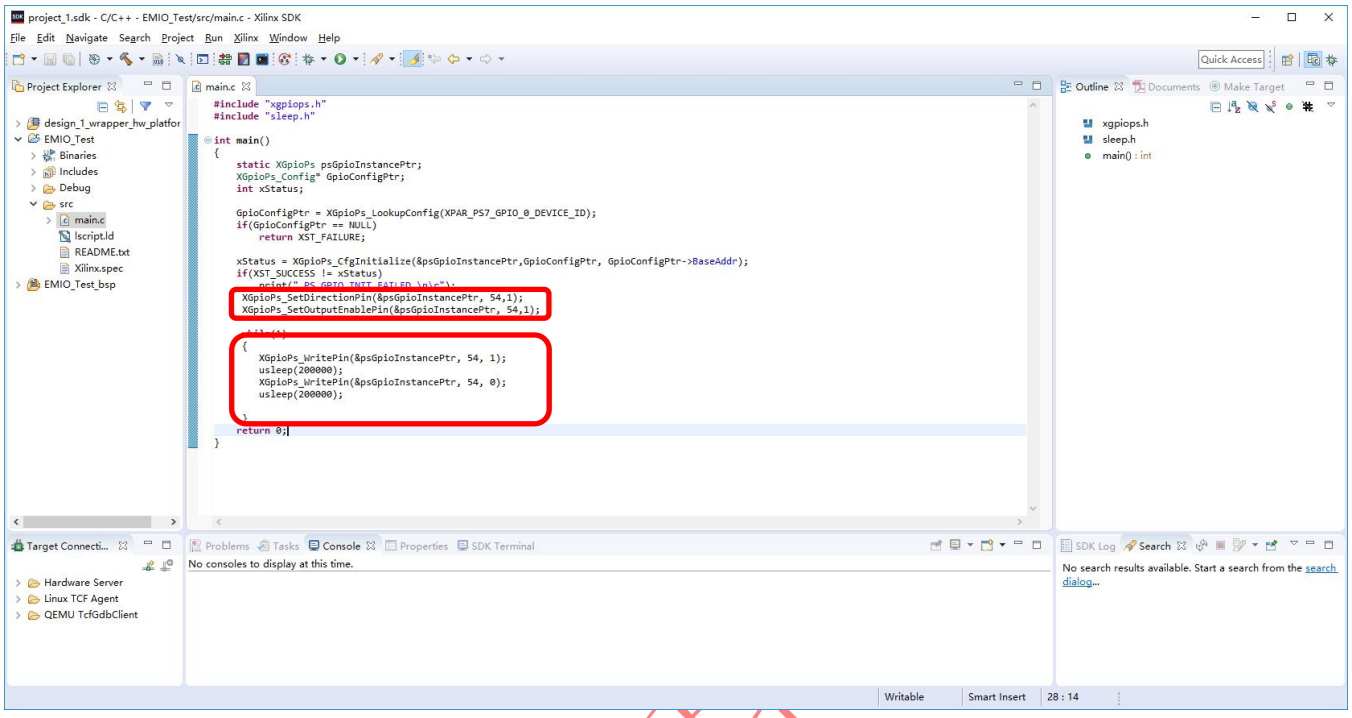


Figure 3-5. Toggle EMIO Pin H17

Right click the project folder EMIO_Test and then select **Run As→1 Launch on Hardware (System Debugger)**. And then users could see the D4 LED will be periodically blinking.

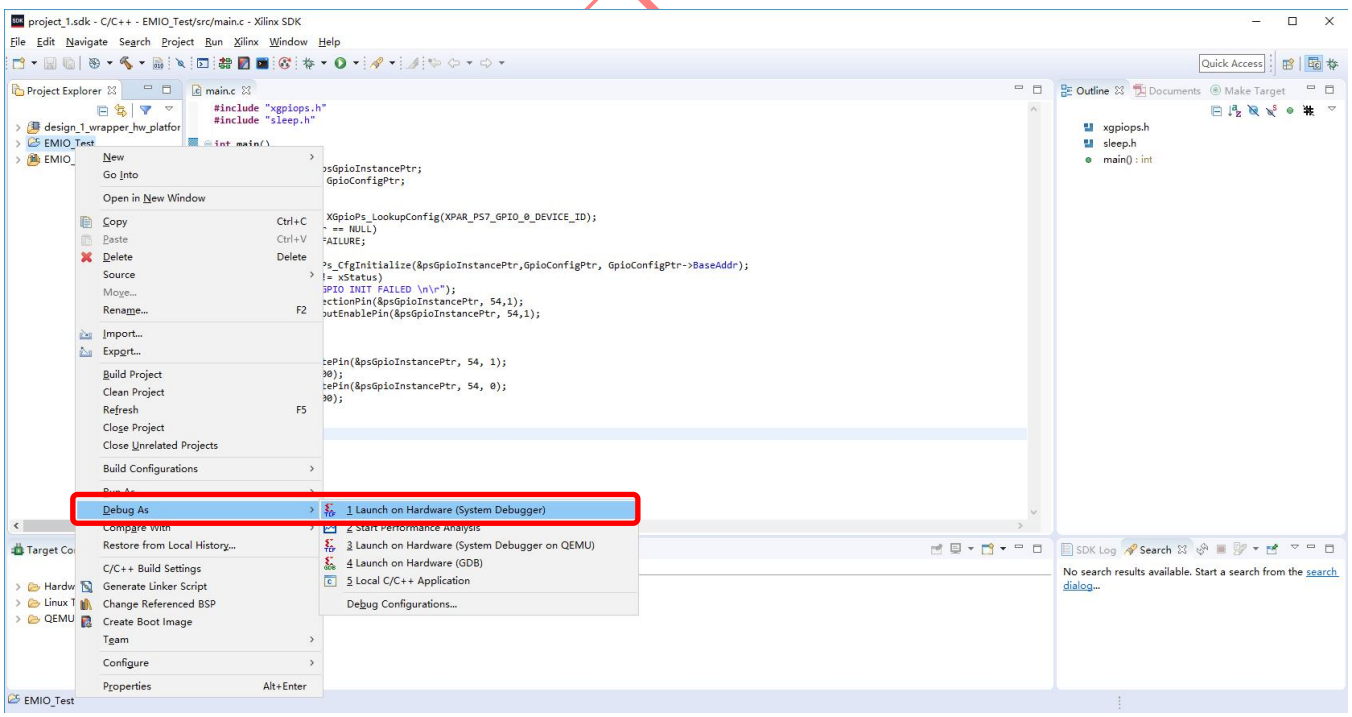


Figure 3-6. Start to Run

4. Experiment 2: MIO User LED

Before start to test the user LED connected to MIO10, users need to make sure the designer_1_wrapper is correctly generated in the previous step. And then execute the whole compilation procedure: **Run Synthesis**, **Run Implementation** and **Generate Bitstream**. Make sure there's no error generated in any of these three steps.

And then export hardware to the SDK and Lunch SDK.

In the SDK environment, create a new empty project named as MIO_Test.

Add the main.c file in MIO_Test/src folder and wait until the SDK project build finished. In the tet routine, it firstly configures the MIO9 as the output pin and enables it. In the main loop, it periodically toggles the MIO.

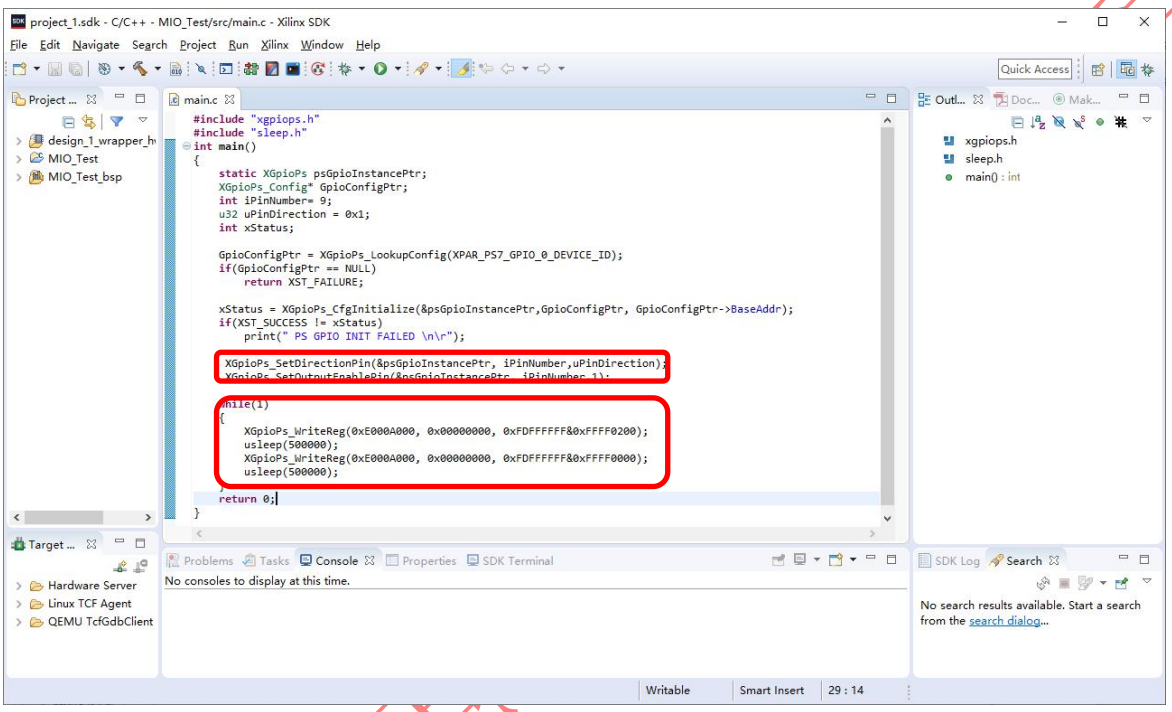


Figure 4-1. Toggle MIO9

Right click the project folder MIO_Test and then select **Run As→1 Launch on Hardware (System Debugger)**. And then users could see the D3 LED will be periodically blinking.

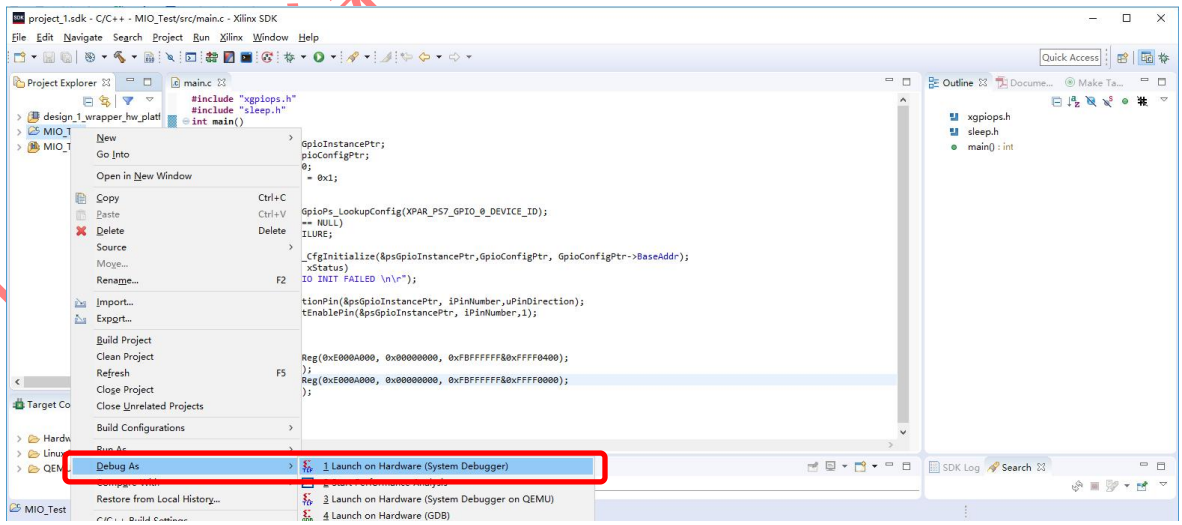


Figure 4-2. Start to Run

5. Experiment 3: DDR3 Test

Before start to test the DDR3 memory connected to PS side, users need to make sure the designer_1_wrapper is correctly generated in the previous step. And then execute the whole compilation procedure: **Run Synthesis**, **Run Implementation** and **Generate Bitstream**. Make sure there's no error generated in any of these three steps.

And then export hardware to the SDK and Lunch SDK.

In the SDK environment, create a new project named as DDR3_Test and click Next button.

Select Available Templates: Zynq DRAM tests and click Finish button.

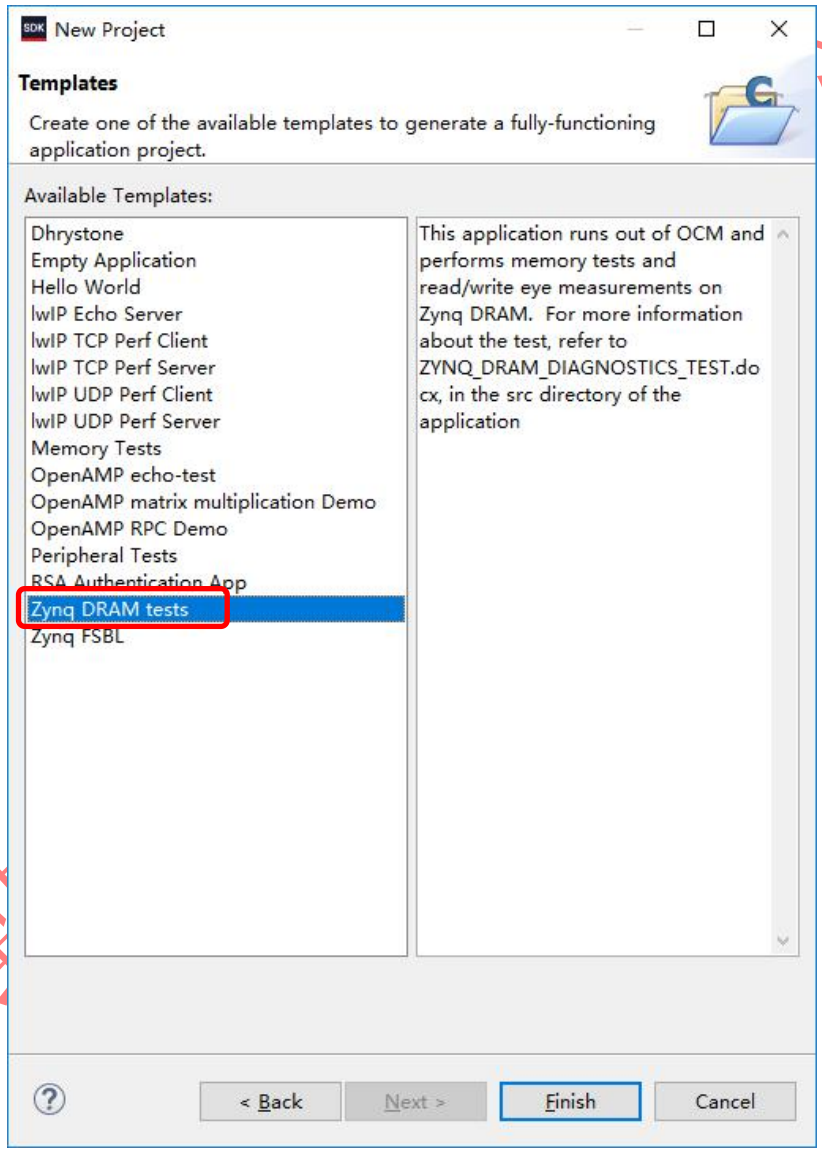


Figure 5-1. Create DDR3 Test Project

Below image shows the main function of the DDR3 test project.

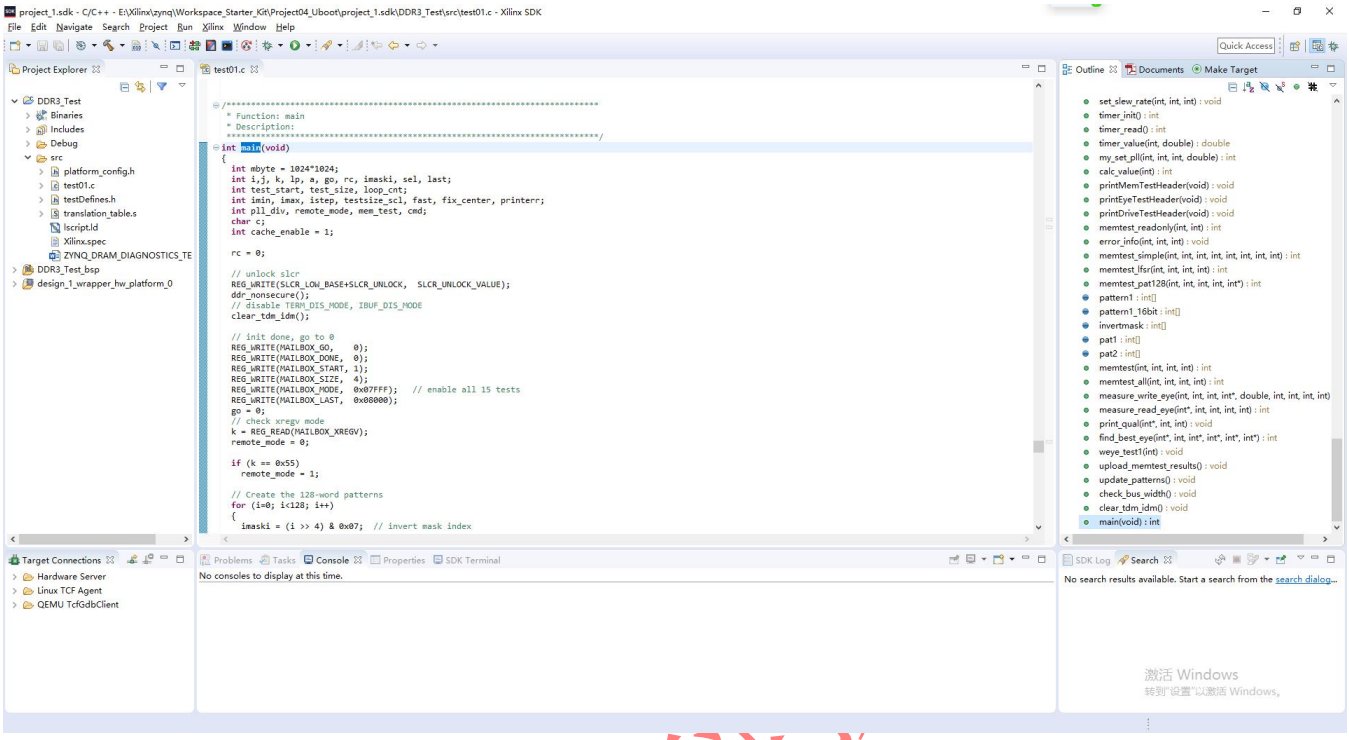


Figure 5-2. Main() Function

Right click the project folder DDR3_Test and then select **Run As→1 Launch on Hardware (System Debugger)**:

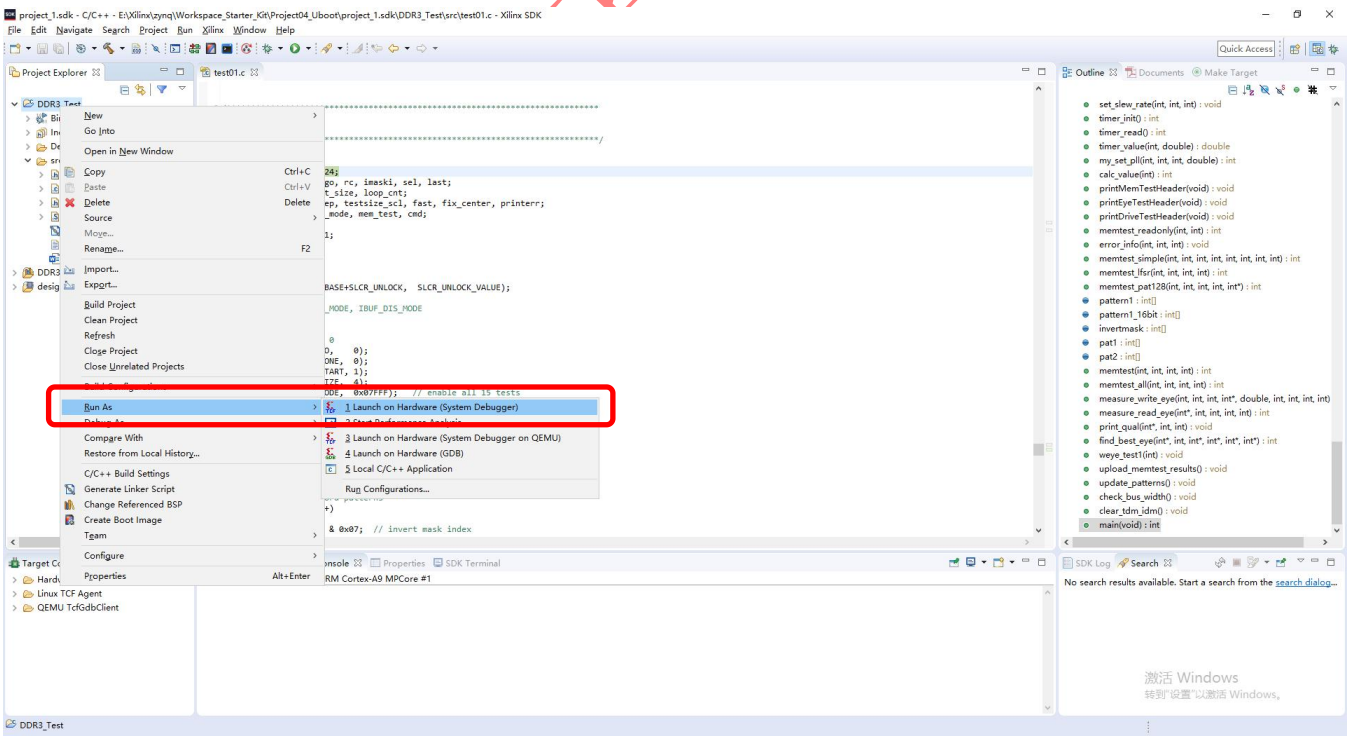


Figure 5-3. Launch Debugger

Use mini USB cable to connect the PC and the ZYNQ Bajie Board J4 connector. Below image shows the output log sent by the DDR3_Test example. And users could type '1', '2', etc. to select different test pattern. And the test results will also be displayed once the test finished.

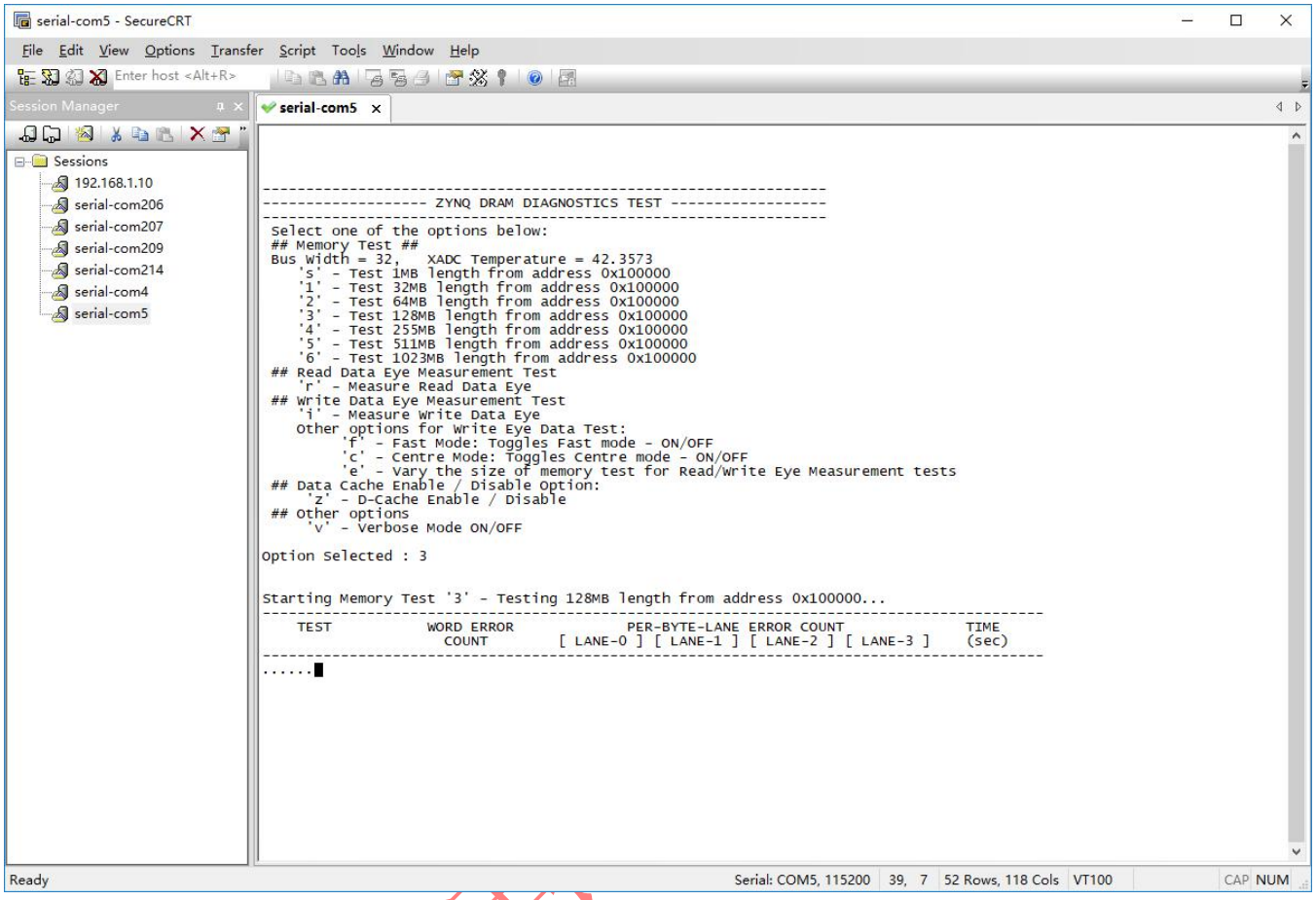


Figure 5-4. Test Pattern

6. Experiment 5: LwIP Test

Before start to test the LwIP running with the MII ethernet interface, users need to make sure the designer_1_wrapper is correctly generated in the previous step. And then execute the whole compilation procedure: **Run Synthesis**, **Run Implementation** and **Generate Bitstream**. Make sure there's no error generated in any of these three steps.

And then export hardware to the SDK and Lunch SDK.

In the SDK environment, create a new project named as LwIP_Test and click Next button.

Select Available Templates: LwIP Echo Server and click Finish button.

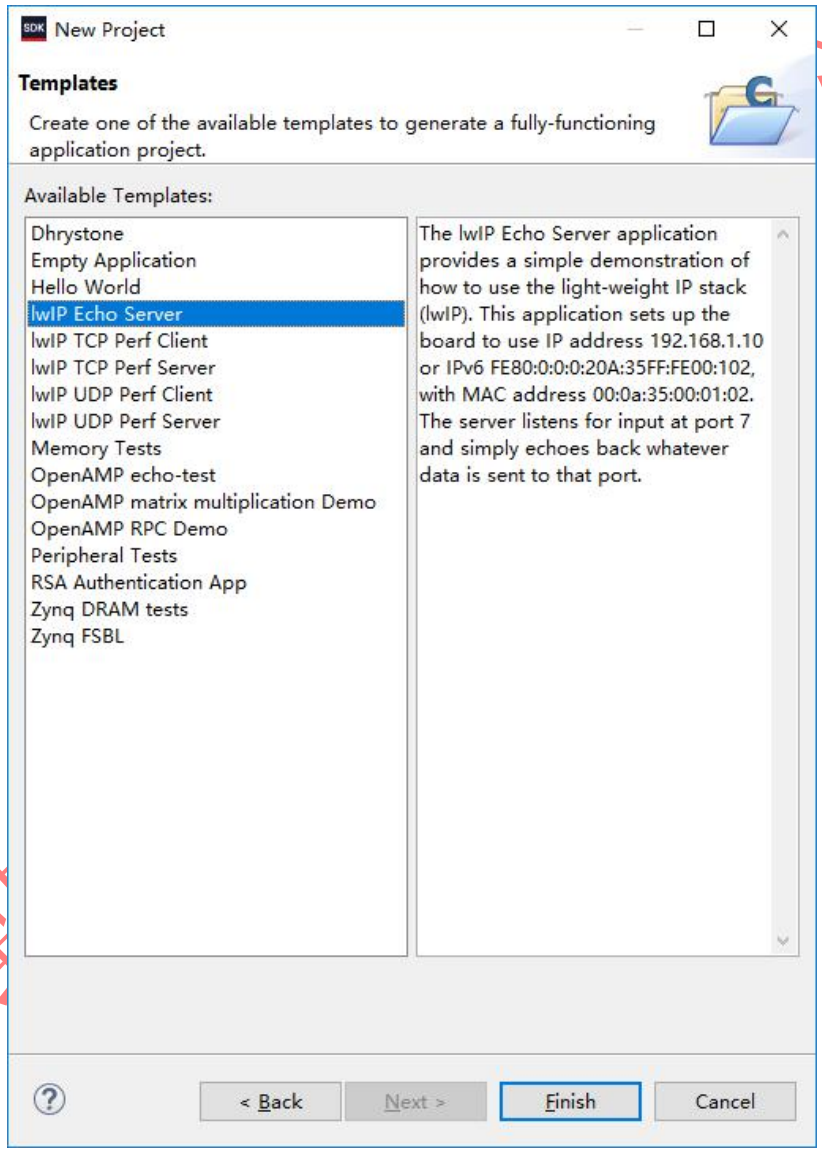


Figure 6-1. Create LwIP Test Project

Below image shows the main function of the LwIP test project. If DHCP server is not applicable, then set this static IP in the test routine directly.

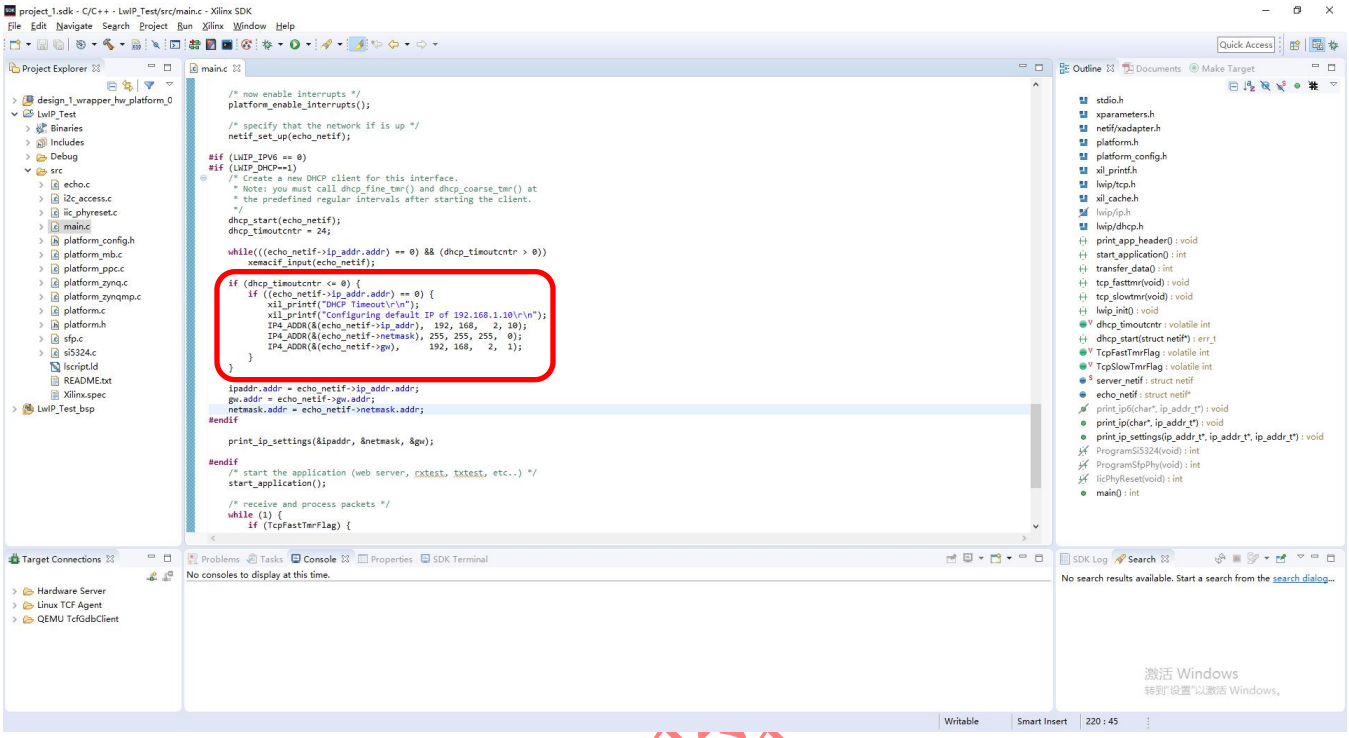


Figure 6-2. Main() Function

Right click the project folder DDR3_Test and then select **Run As→1 Launch on Hardware (System Debugger)**:

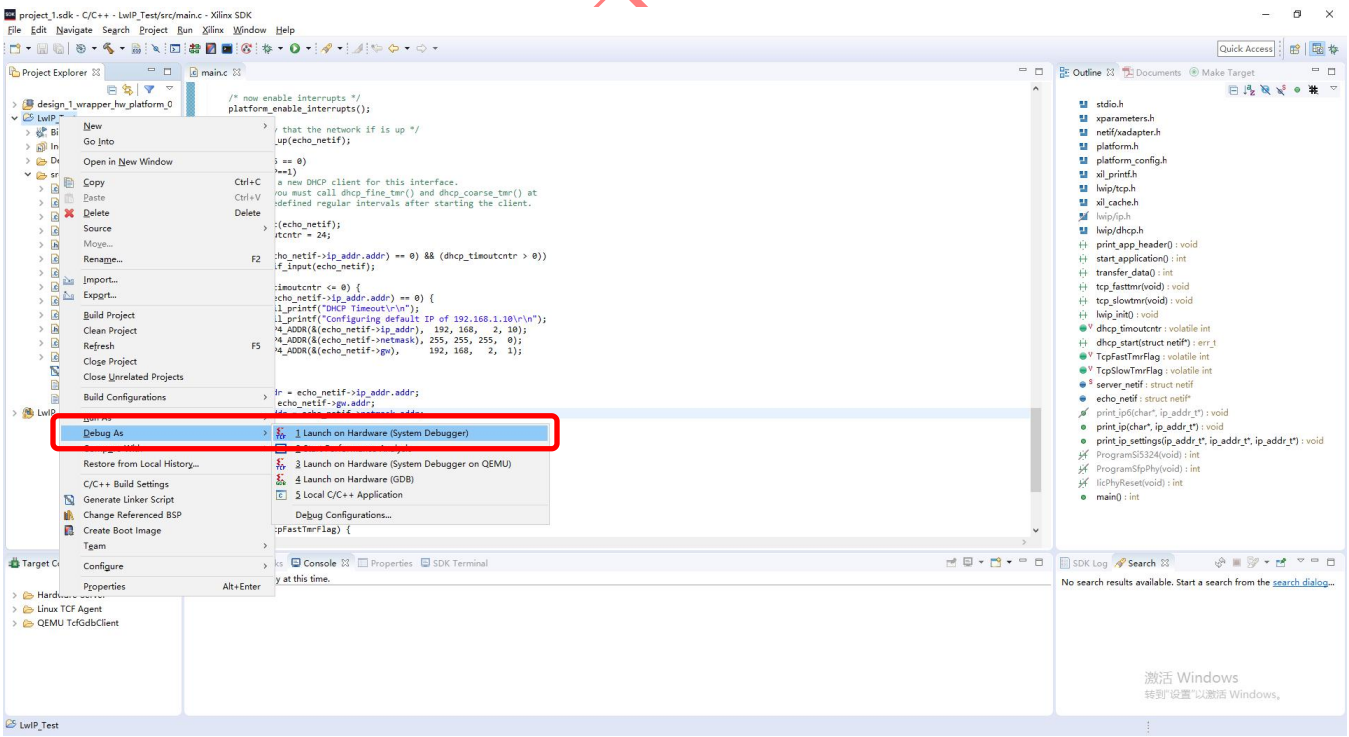


Figure 6-3. Launch Debugger

Use mini USB cable to connect the PC and the ZYNQ Bajie Board J4 connector. Below image shows the output log sent by the LwIP_Test example.

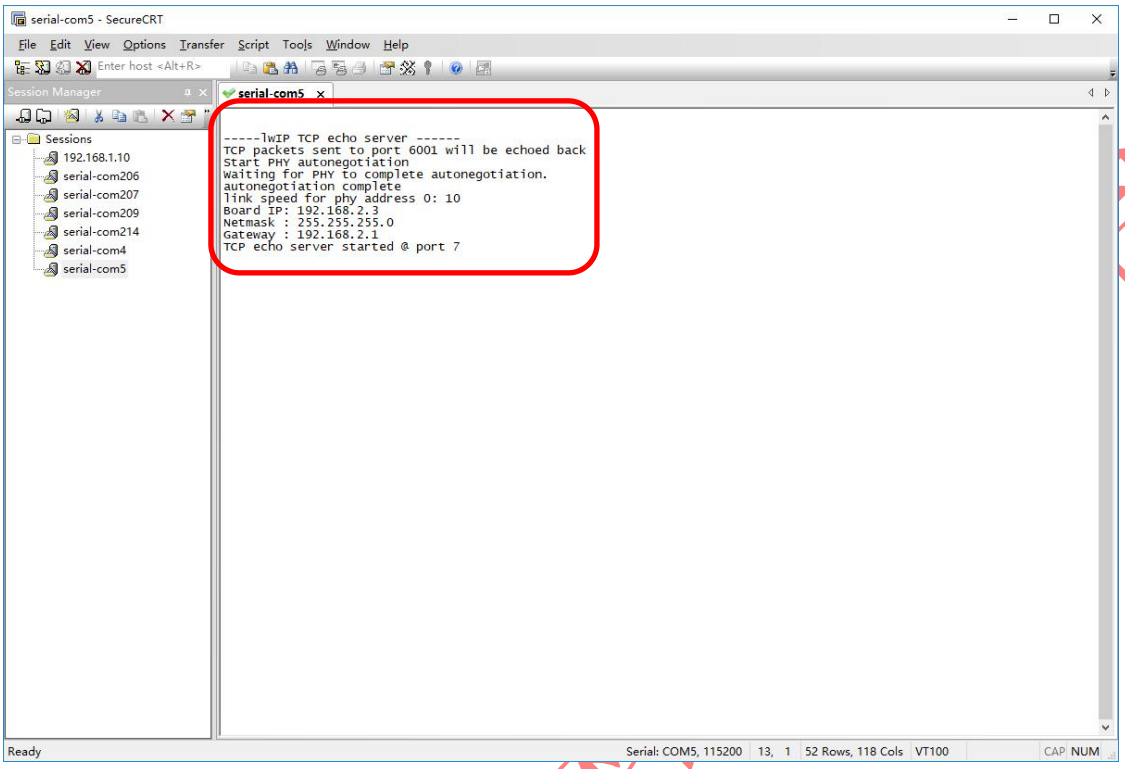


Figure 6-4. Log Info

Open ethernet test assistant and set the configurations as shown in below image. TCP Client@192.168.2.3:7 and connect to server. “http://www.cmsoft.cn” is the message will be echo back from the server side.

Figure 6-5. Echo Message

7. Reference

[1] ug585-Zynq-7000-TRM.pdf
[2] ds187-XC7Z010-XC7Z020-Data-Sheet.pdf
[3] ug865-Zynq-7000-Pkg-Pinout.pdf
[4] MT41K256M16TW-107:P.pdf
[5] tps563201.pdf
[6] RTL8211E-VL.PDF

8. Revision

Doc. Rev.	Date	Comments
0.1	28/03/2020	Initial Version.
1.0	13/04/2020	V1.0 Formal Release.