Zynq7000-PS-GPIO之LED点亮

# 基础知识

开始本小节之前，希望你已经阅读了第0，1，2小节，了解了Zynq7000的大致架构，搭建好了完整的开发环境，掌握了Vivado+SDK开发套件的基本使用方法。接下来，我们开始一步步发掘Zynq7000的魅力吧！

# 实验：通过PS的GPIO点亮LED灯

## 实验环境

### Software

Windows 10 64bit

Vivado 2017.4

Xilinx SDK 2017.4

### Hardware

Zybo-Z7-20

## 实验成果和目标

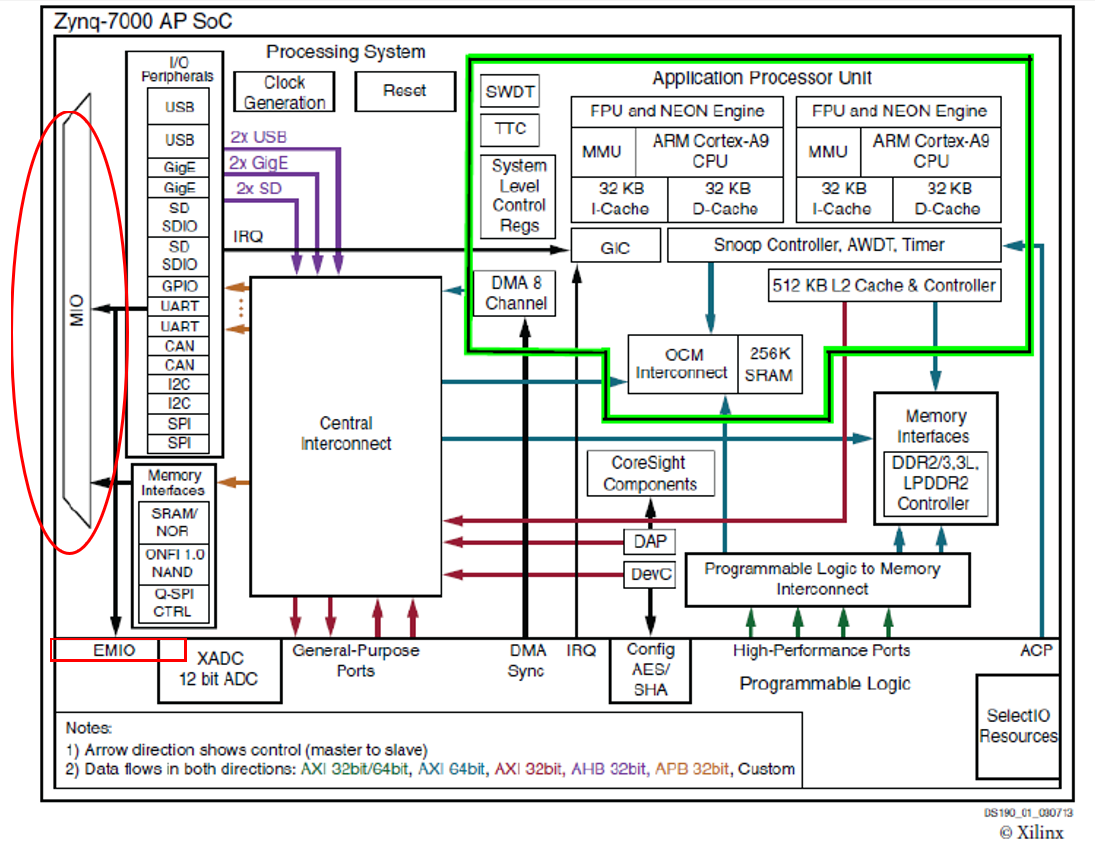
成果：分别用两种方式，直接访问寄存器和API Driver，点亮Zybo-Z7-20的LED4

目标：掌握了解Zynq7000和Zybo-Z7-20的技术手册使用方法，并了解API文档的查阅使用。

## 预备知识

#### MIO & EMIO

Zynq是由PS和PL两部分组成，PS部分的CPU配有许多外围I/O接口，可以根据不同协议与外围设备通信，比如UART,I2C,SPI,GPIO等等。我们通过多路复用模块MIO（Multiplexed I/O）来实现通信。注意，MIO仅仅供PS部分使用。除此以外，当PS部分的MIO引脚不够用时，可以从PL部分“借用”一些引脚来供PS使用，我们称这些引脚为EMIO（Extended Multiplexed I/O）。



图表 1：MIO与EMIO

## 实验步骤

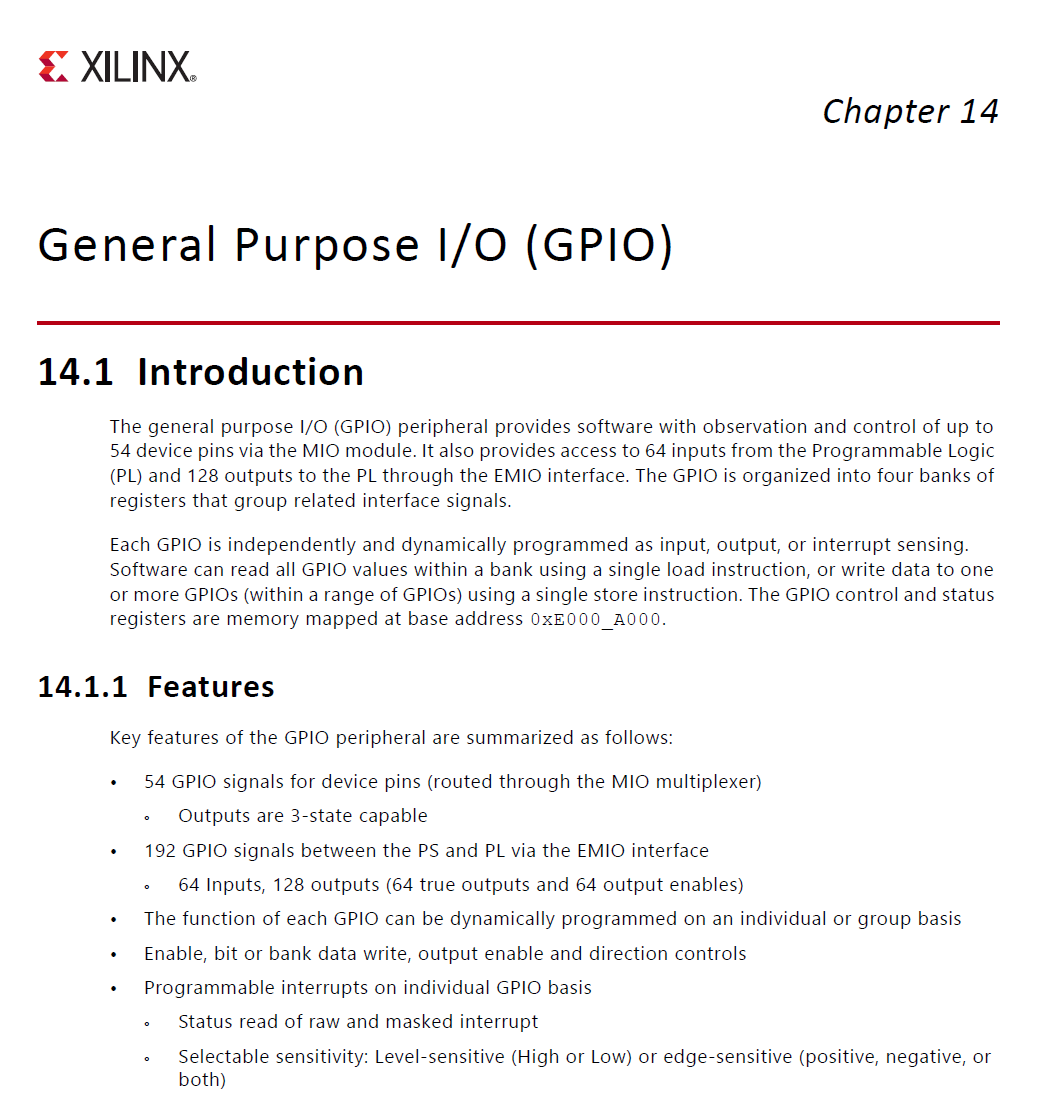
在本次实验中，我们主要采用两种方式对PS中GPIO接口进行控制。

方式1：直接访问寄存器；方式2：利用Xilinx SDK Driver中的API

### 方式1：直接访问寄存器

首先，我们需要查阅Zynq-7000-TRM（Technology Reference Manual）技术手册，获取PS部GPIO接口的使用说明。打开TRM的pdf文件，Ctrl+F输入“GPIO”，查找对应的章节如下。

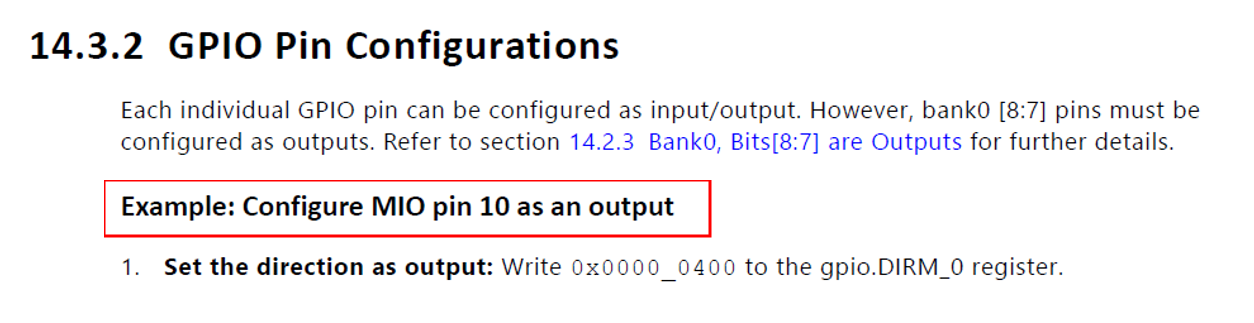
在综述部分，我们发现GPIO有一系列功能供我们来开发使用，比如通过MIO模块引出54个独立可动态编程的引脚，来供我们任意配置GPIO为输入，输出，中断接口，采用各种通信协议（UART,I2C,SPI等等）来与外围设备通信。或者借助PL部的EMIO模块，拓展更多的外围引脚。

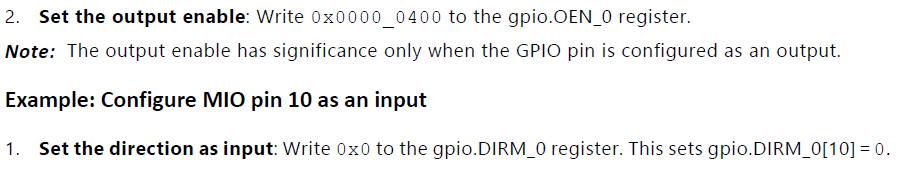


图表 2：GPIO综述

#### 原理分析

在官方手册中，给我们提供了一个如何配置MIO-pin10的例子，如下图所示：





图表 3：如何配置GPIO（例MIO-PIN10）

从例子中可以看出，若要将一个MIO-PIN10配置成为输出，第一步，将0x0000\_0400写入DIRM\_0寄存器，第二步，将0x0000\_0400写入OEN\_0寄存器。那么，我们好奇为什么把这个值写入对应的寄存器就可以将MIO-PIN10配置为输出GPIO呢？

第一步：16进制0x0000\_0400，低16位的值0x0400转化为2进制0000\_0100\_0000\_0000，不难发现，从低位到高位，第10位的值恰好为1，对应PIN10。

第二步：同理，对应PIN10位的值，置1。

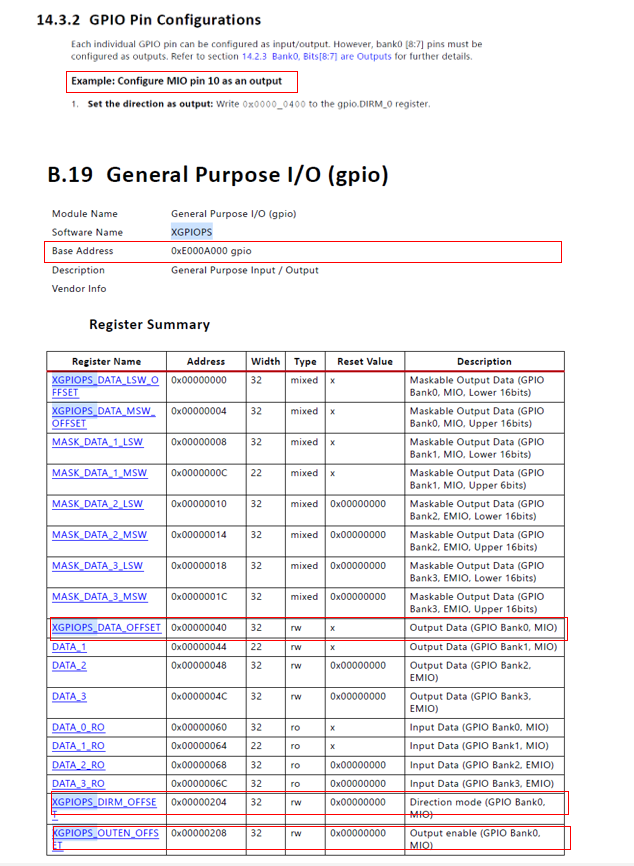
然后，我们查阅TRM附录B中对应的GPIO（B.19）小节，如图5所示，GPIO的基地址为0xE000A000，主要相关偏移量有三个：

XGPIOPS\_DATA\_OFFSET：输出数据

XGPIOPS\_DIRM\_OFFSET：GPIO的通信模式

XGOPOPS\_OUTEN\_OFFSET：输出使能信号

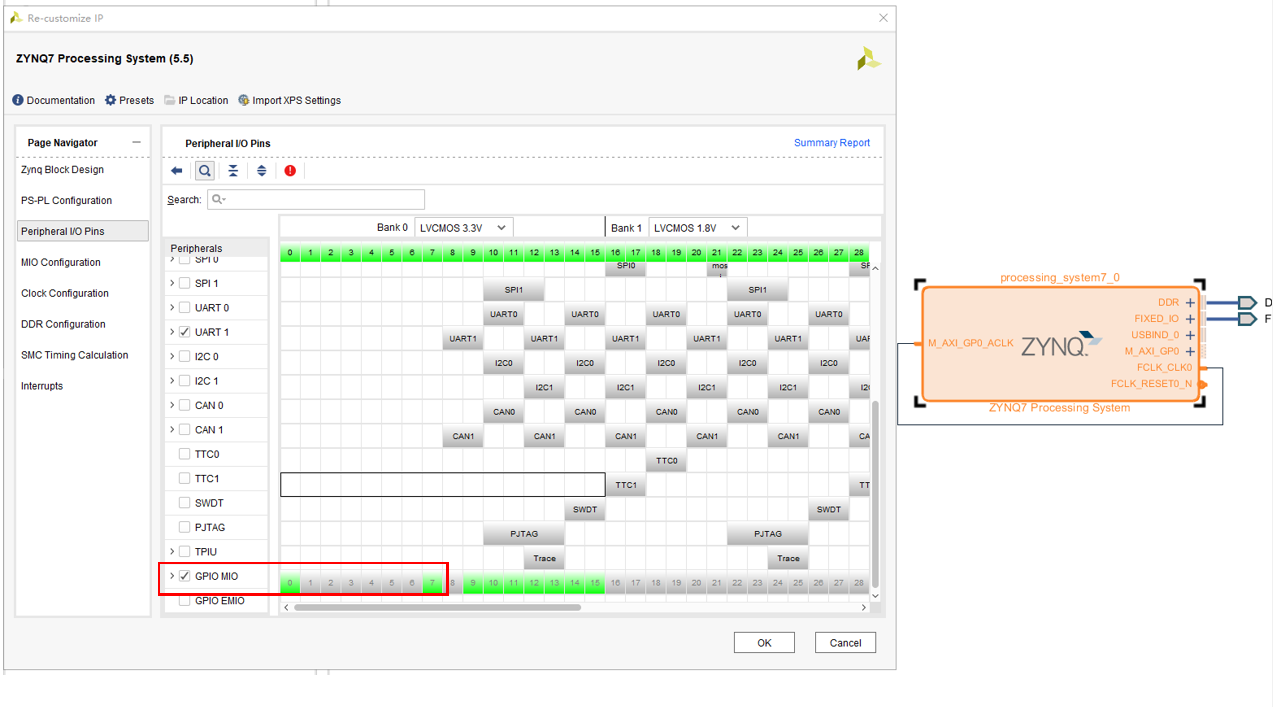
掌握了以上地址信息，我们便可以配置GPIO，并通过其读取数据了。



图表 4：GPIO 寄存器详细配置

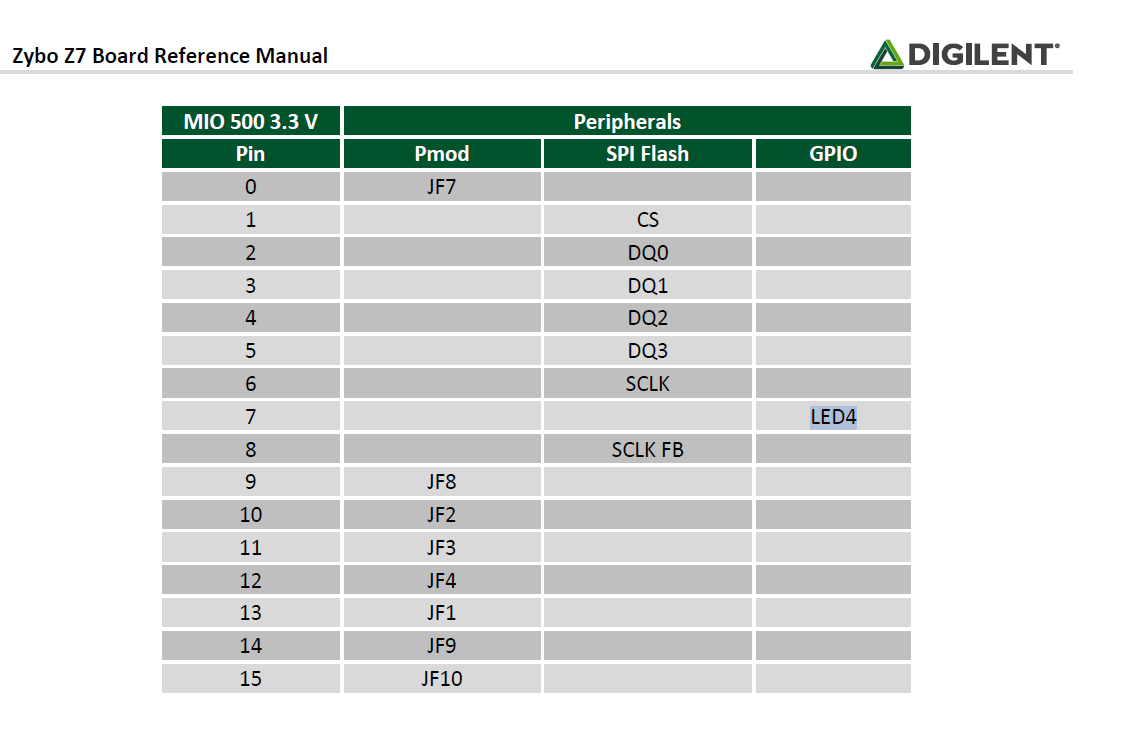
#### 实际操作

**STEP1：**由于我们不需要更改硬件配置信息，本节继续沿用上一节的HelloWord工程。打开上一节的HelloWord工程，确认Zybo-Z7-20的外围引脚配置信息。MIO模块的0号和7号引脚，分配为GPIO接口使用。



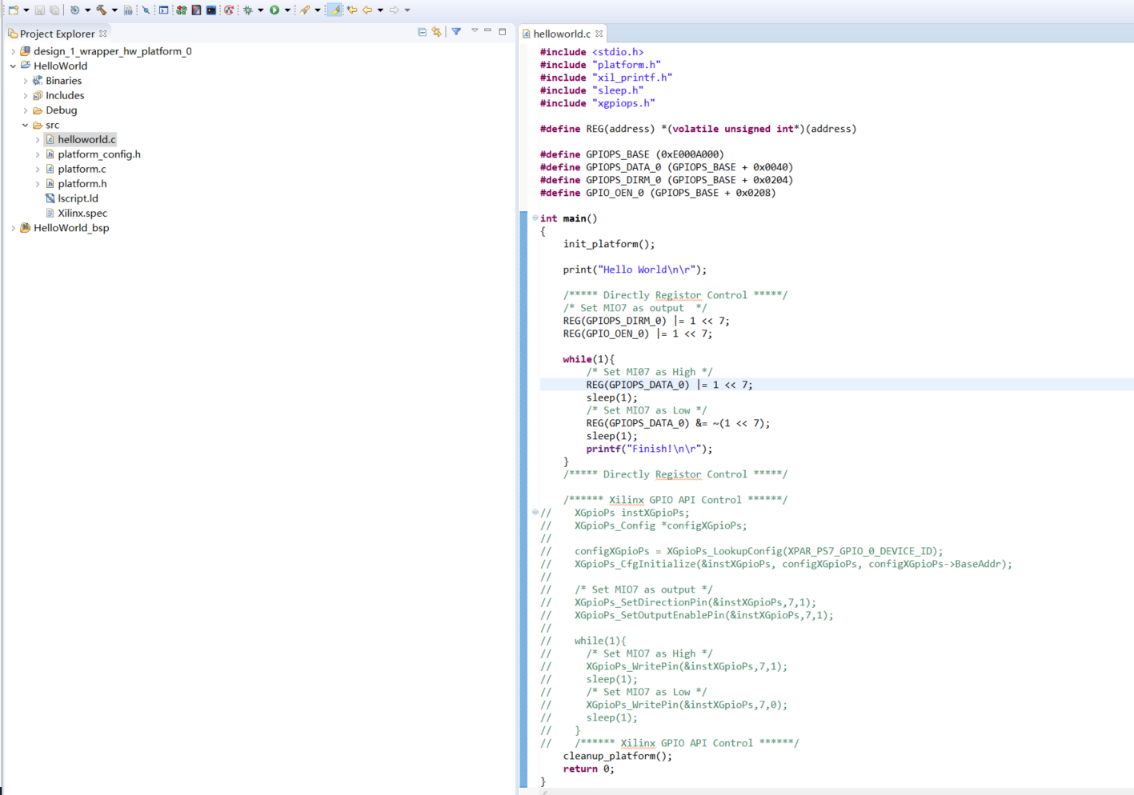
图表 5：Vivado中查看GPIO MIO引脚配置

**STEP2：**查阅Zybo-Z7-20的TRM，确认Zynq7000处理器的MIO7引脚在Zybo-Z7-20开发板中，与哪一个LED灯链接。不难发现，Zybo开发板中，MIO7引脚与板上的LED4对应。



图表 6：Zybo-Z7-20的MIO引脚分配

**STEP3：**确认好需要控制的电子器件以后，就要进行实际的代码编辑阶段了。因为Zynq7000的PS部分，由2个ARM处理构成，我们可以直接通过C语言进行控制。同上一节的HelloWord工程一样，直接在Xilinx SDK的HelloWord.c中进行代码更改即可。



图表 7：直接寄存器访问代码

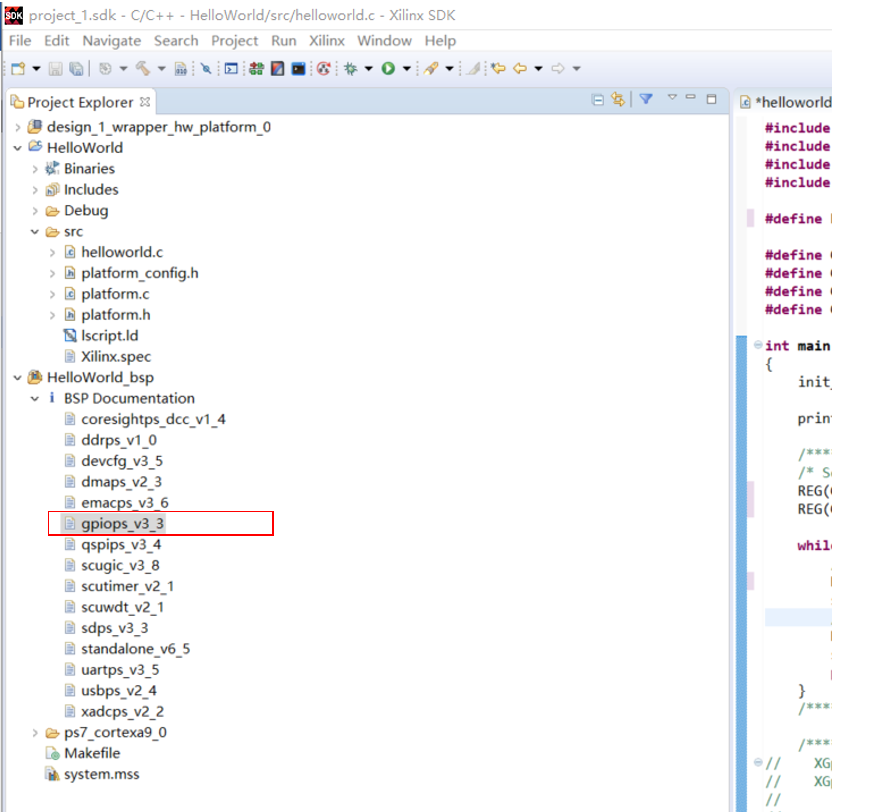
**STEP4：**完成实际代码编译后，同第二小节的步骤，Program FPGA Board -> Run as -> Launch on Hardware (System Debugger) -> LED4间隔一秒进行闪烁。

### 方式2：Xilinx SDK API

直接调用寄存器的方法可以很好的帮助我们看清本质，搞清楚事情的来龙去脉。我们会觉得费了九牛二虎之力仅仅点亮了一个LED灯，难免心中有些失落。不过，Xilinx SDK中为我们提供了一套健全的API，这大大的提高了我们的开发效率，实际工作中大规模项目的开发，离不开原厂API的使用。接下来，我们一起用新的方式去点亮LED4吧！

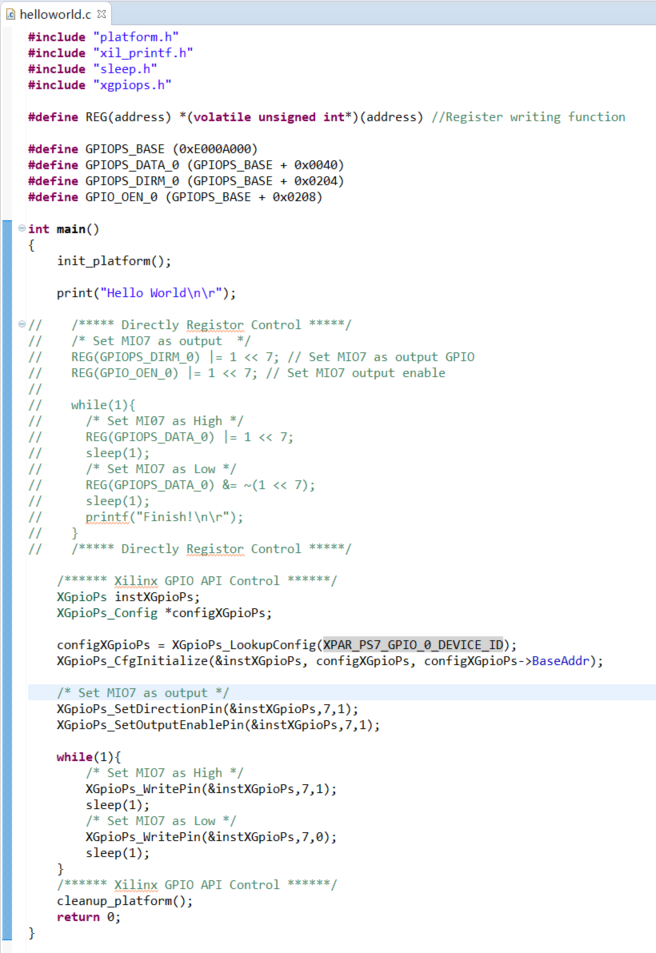
#### 阅读Xilinx SDK Driver API的文档

在Xilinx SDK的Project Explorer中，此项目的Board Support Package（BSP）文件中，可以找到Xilinx提供的API文档。本小节中，我们只涉及的GPIO的控制，查阅gpiops\_v3\_3文档，我们便可以知道所有的相关API信息。



图表 8：API 文档

读者可以根据自身需要参阅API文档，此小节点亮LED4的代码如下。按照同方式1的步骤，我们就可以顺利使LED4闪烁啦！



图表9：API LED4 Control

## 反思

参考图表1中Zynq7000的系统架构，除了GPIO，与PS直接相连的外围I/O接口（Peripherals I/O）还有I2C,SPI,UART,USB,CAN,Giga-Enthernet，既然我们已经掌握了GPIO的配置方法，那么同理，我们是不是同样可以举一反三的操作其它的Peripherals I/O了呢？

# 参考文献

[1] Xilinx, Zynq7000 SoC Technical Reference Manual.

[2] Digilent, Zybo Z7 Board Reference Manual.

[3] Xilinx, The Zynq Book.

[4] iwatake2222, ZYBO (Zynq) 初心者ガイド (3) PSのGPIOでLチカ,

<https://qiita.com/iwatake2222/items/39d2ed00fa88a1204fce>