www.digilentchina.com
25A, No.1369,Dongfang Road, Shanghai
Tel:+86-021-58893151 Fax: +86-021-58894598

Lab4: 基于自定义 IP 核的 LED 显示实验（软件控制部分）

*基于 **Nexys 4 FPGA** 平台*

Lab4: 基于自定义 IP 核的 LED 显示实验（软件控制部分）

实验简介

本实验与 Lab3 为配套实验。旨在使读者进一步熟悉 Xilinx 的 XPS 和 SDK 工具的使用，并初步掌握添加自定义 IP 核的步骤，最终完成一个通过串口输入数据来控制流水灯方向的简单程序。其中，本实验侧重于通过软件来操作硬件设备，从而达到操控 LED 的显示效果。

实验目标

在完成本实验后，您将学会：

- 如何在 SDK 中通过 C 语言实现流水灯效果，并通过串口对其进行控制。

实验过程

本实验旨在指导读者使用 Xilinx 的 XPS 工具，调用 GPIO 与 UART 的 IP 核，并将导入到 SDK，调用它们，通过在串口输入数据并控制流水灯方向，然后在 Nexys 4 平台上进行测试验证。

实验由以下步骤组成：

1. 将工程导入到 SDK
2. 在 SDK 中添加 c 语言源程序
3. 在 Nexys 4 上进行测试验证

实验环境

◆ 硬件环境

1. PC 机
2. Nexys 4 FPGA 平台

◆ 软件环境

Xilinx ISE Design Suite 14.3（FPGA 开发工具）

第一步 将工程导入 SDK

1-1. 打开 XPS 工程

1-1-1. 在 Lab3 工程存储路径中，双击后缀名为“.xmp”的工程文件，如图 1 所示。

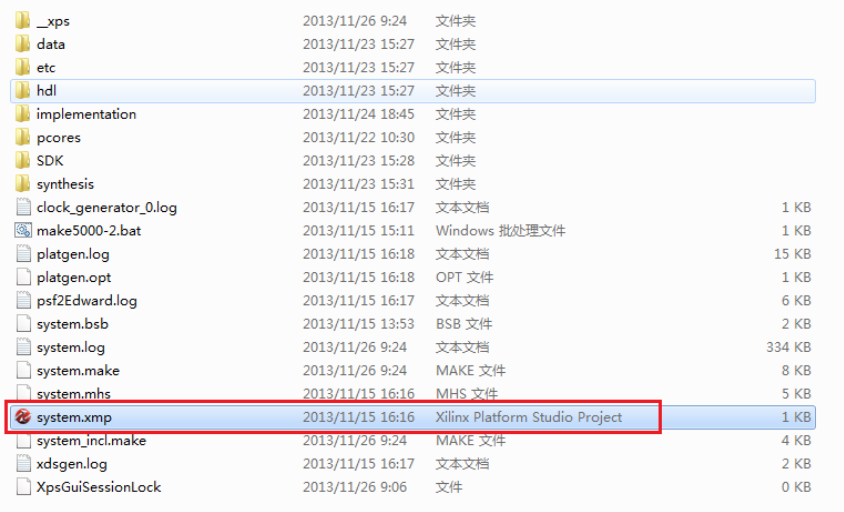


图 1：XPS 工程文件

1-1-2. 将工程导入到 SDK

在页面左边，点击 **Export Design**。

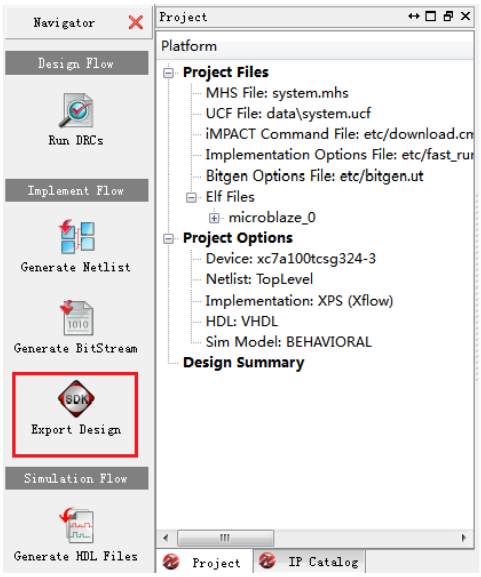


图 2：将工程导入到 SDK

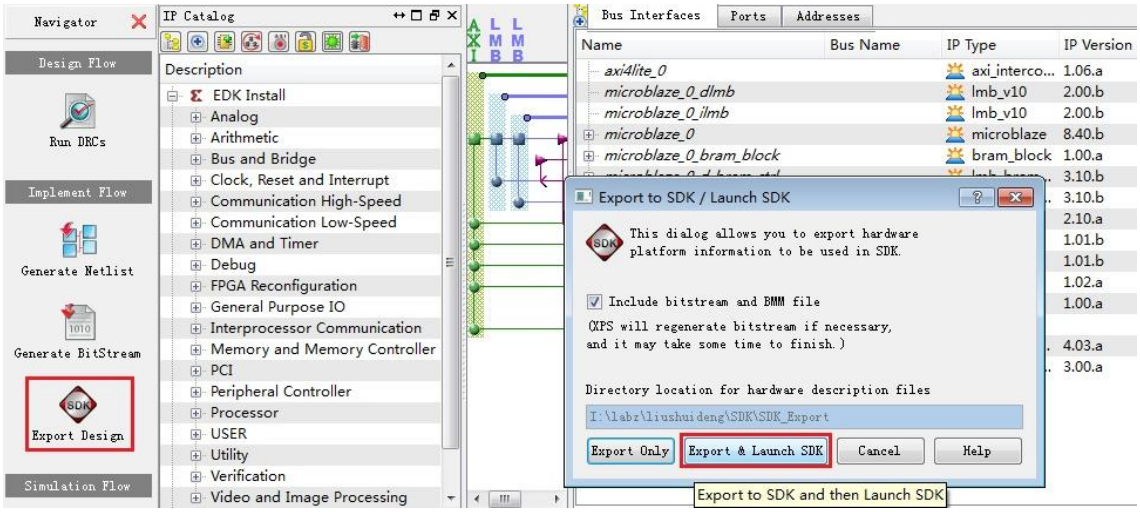


图 3：在弹出的对话框中选择 Export & launch sdk

1-1-3．选择 SDK 导入路径

注意要具体到..\sdk\sdk_export

点击 ok

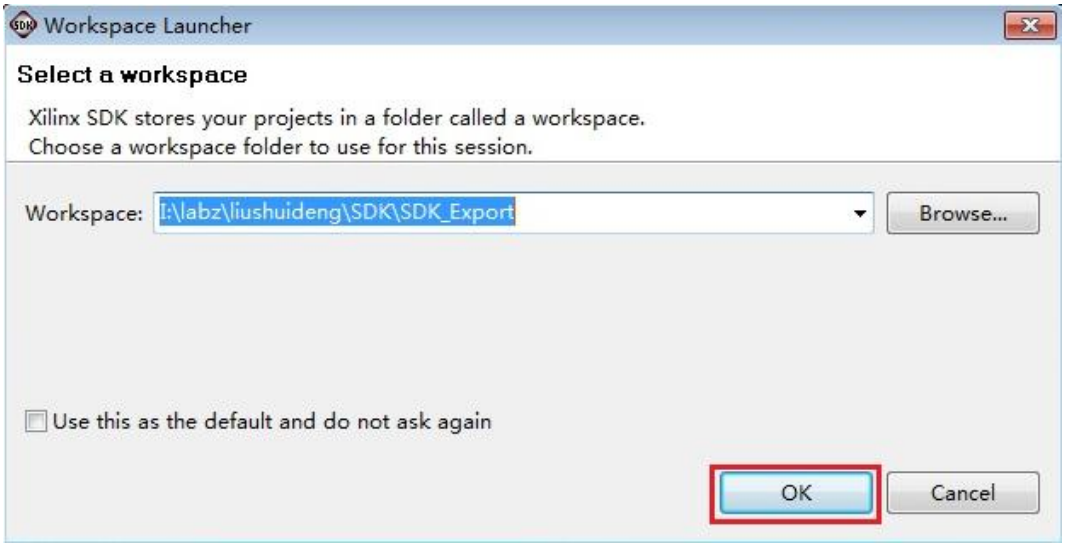


图 4：选择 SDK 导入路径

第二步 添加 C 语言源程序

2-1. 添加新的工程，编写软件程序。

2-1-1. 在 SDK 的用户界面中，选择 file—new—application project。

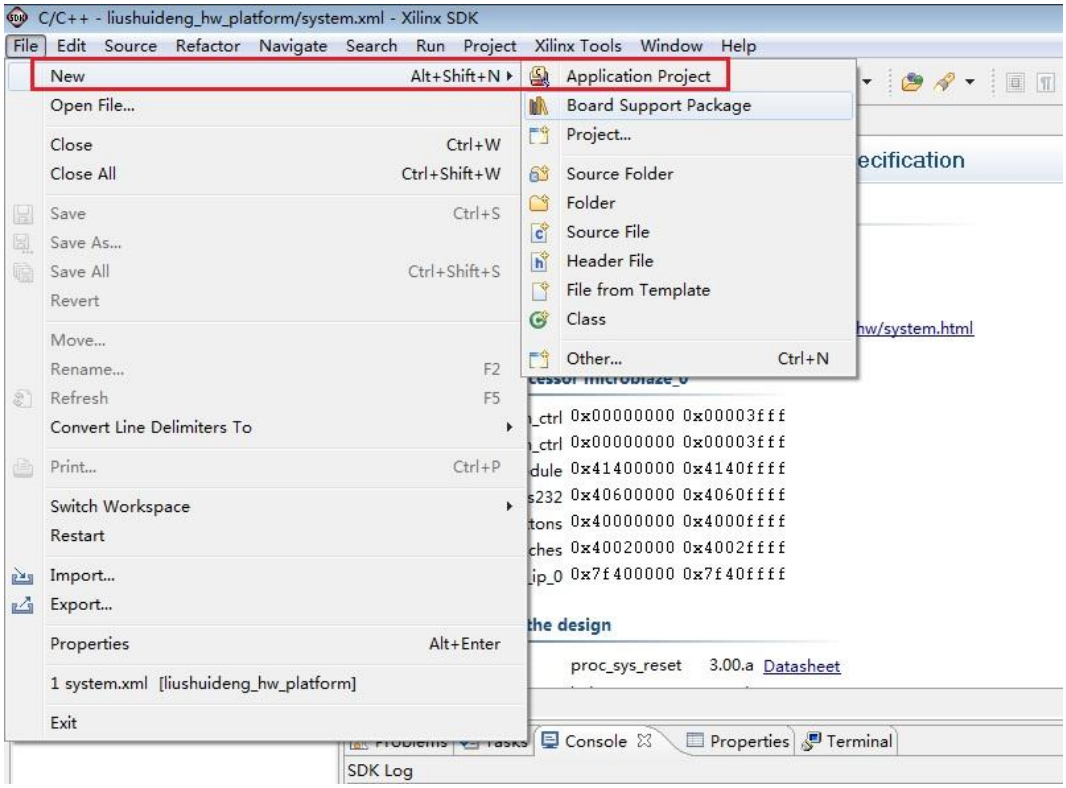


图 5：建立新的工程

2-1-2. 输入工程的名称，这里使用 test，同样不要包含空格和中文，点击 next

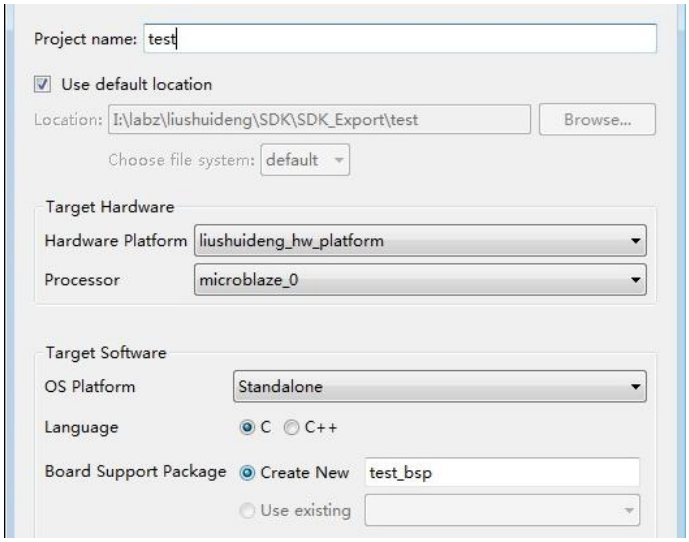


图 6：配置软件工程

2-1-3. 在下一步弹出的对话框中选择 Empty Application，然后点击 finish。

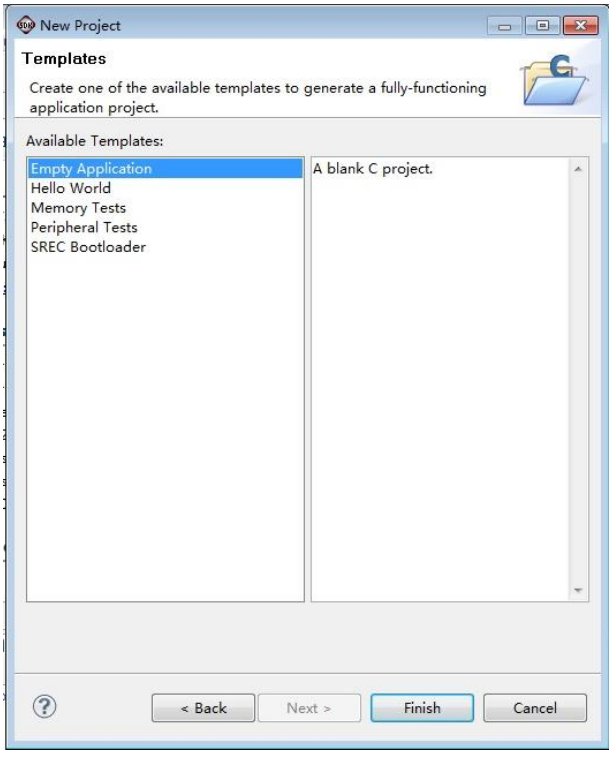


图 7：选择软件工程的模板

2-1-4. 为新建的工程添加 Source File 文件。

点击鼠标展开 **test** 工程，右键点击 **Src** 一项，并在弹出的菜单中选择 **New** 一栏中的 **Source File** 选项。

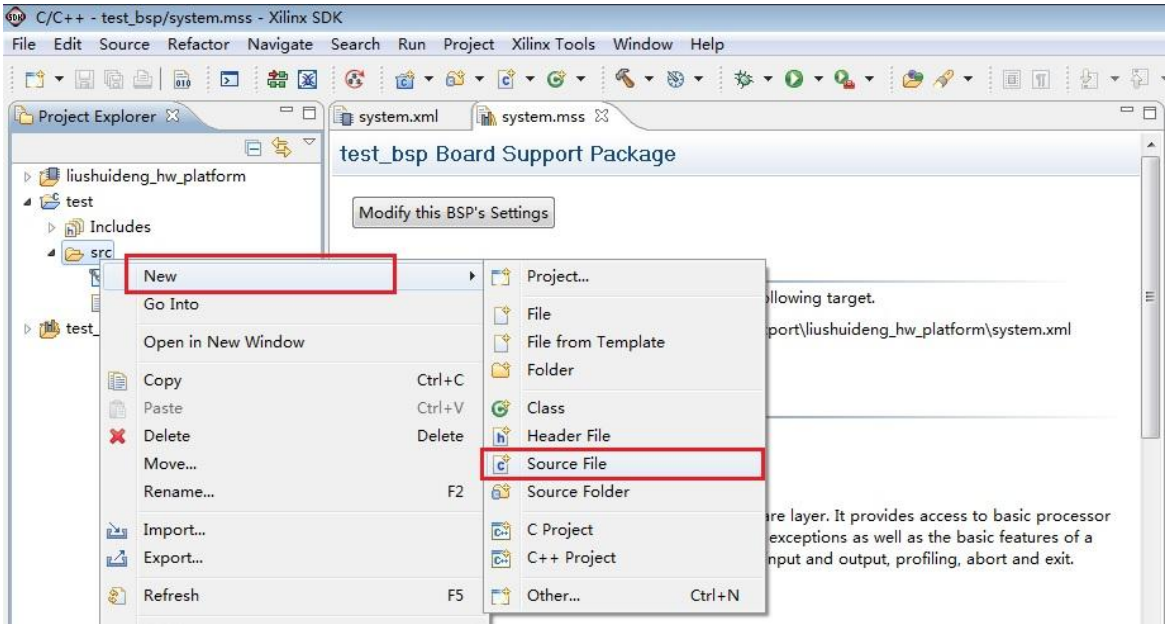


图 8：添加 Source File 文件

随后在 **Source File** 选项中输入源文件的名称，注意一定要添加**“.c”后缀**，否则会报错。

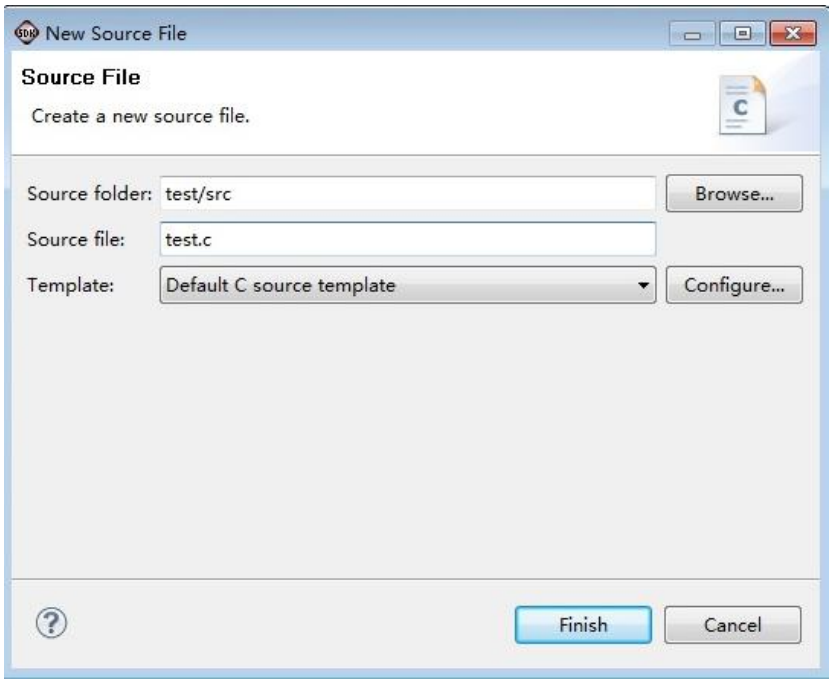


图 9：输入 Source File 文件的名称

2-1-5. 修改 selftest 文件。

点击屏幕左上角的 **File** 选项，打开 **Open File** 选项，我们需要打开在 **liushuideng（即工程目录）\ drivers \ led_ip_v1_00_a \ src** 路径下的 **led_ip_selftest.c** 文件。

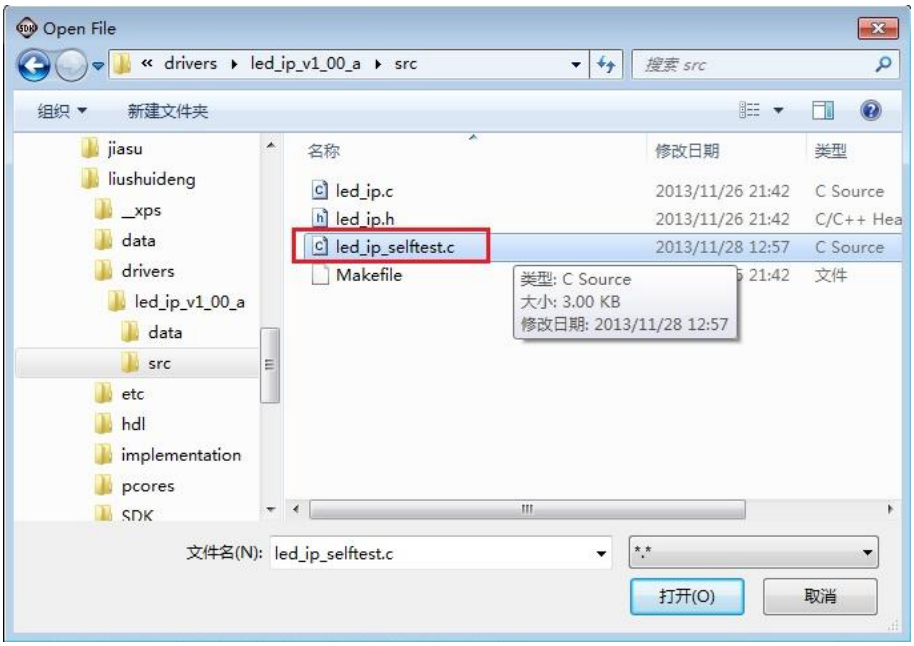


图 10：打开 selftest 文件

接下来，我们如图所示添加一句声明：

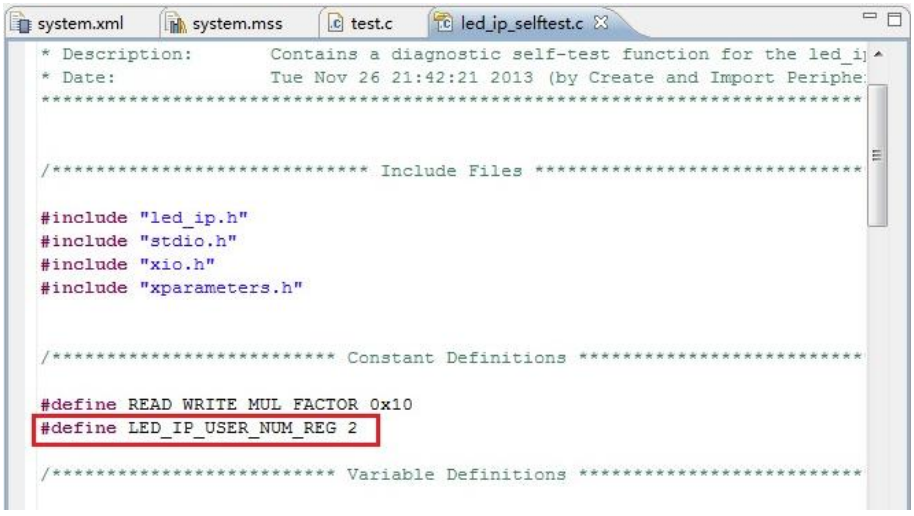


图 11：修改 selftest 文件

2-1-6. 点击屏幕上方的“Xilinx Tools”一项，打开 Repositories 选项。

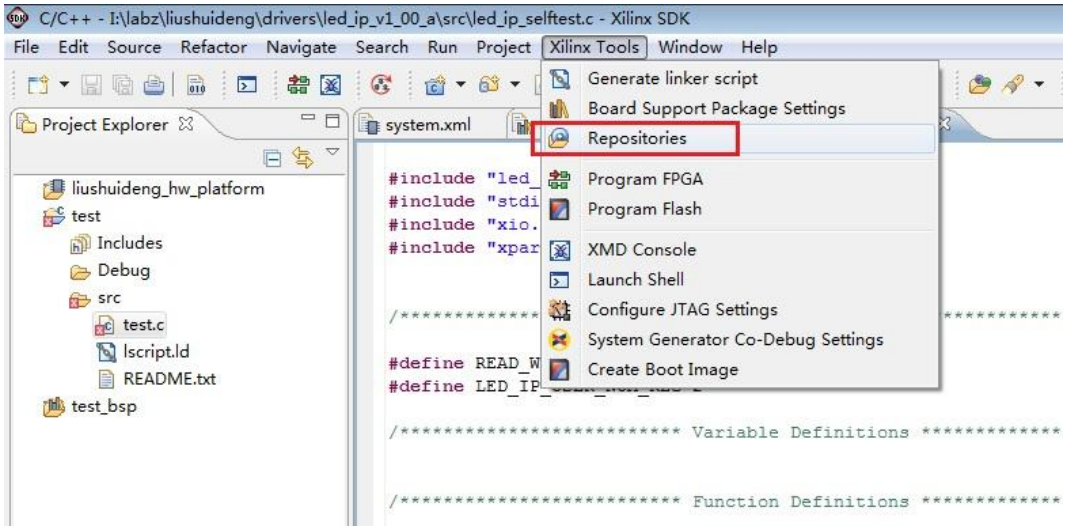


图 12：打开 Repositories 选项

在弹出的窗口中选择 **Repositories**，并在屏幕上方 **Local Repositories 窗口**的右边点击 **New 按钮**，随后选择自己的工程路径，并点击 **OK**。

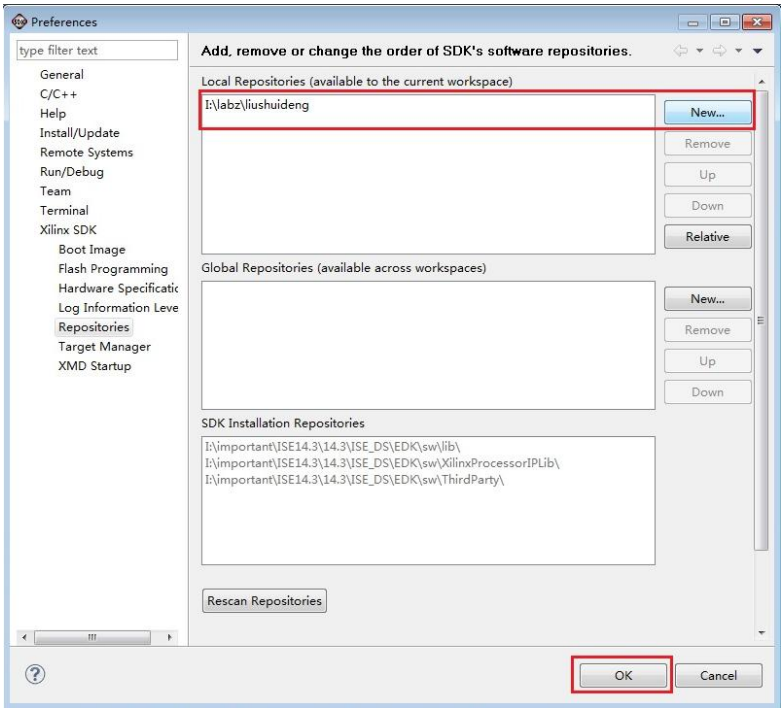


图 13：在 Repositories 窗口内添加工程路径。

2-1-7. 设置版级支持包。

点击屏幕上方的**“Xilinx Tools”**一项，打开 **Board Support Package Settings** 选项。

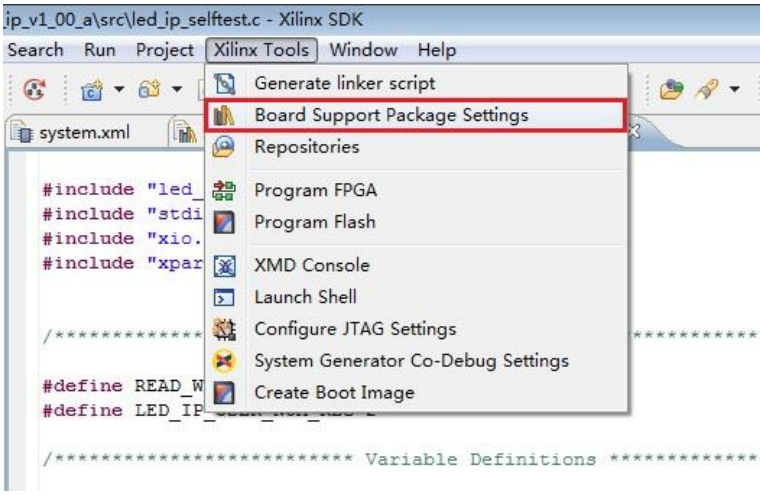


图 14：打开版级支持包设置选项

点击工程所需的 bsp 文件，并单击 OK。

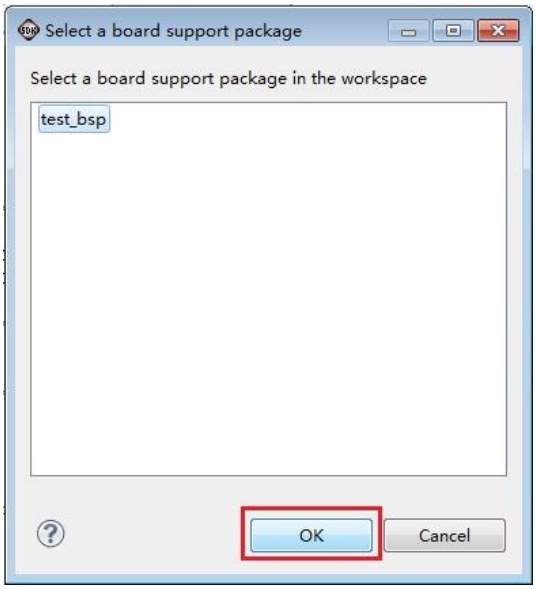


图 15：选择工程所需的版级支持包

在弹出窗口的左侧点击 **drivers** 一栏，并在 **led_ip_0** 这一行中的第三项从默认的 **generic** 改为 **led_ip**。

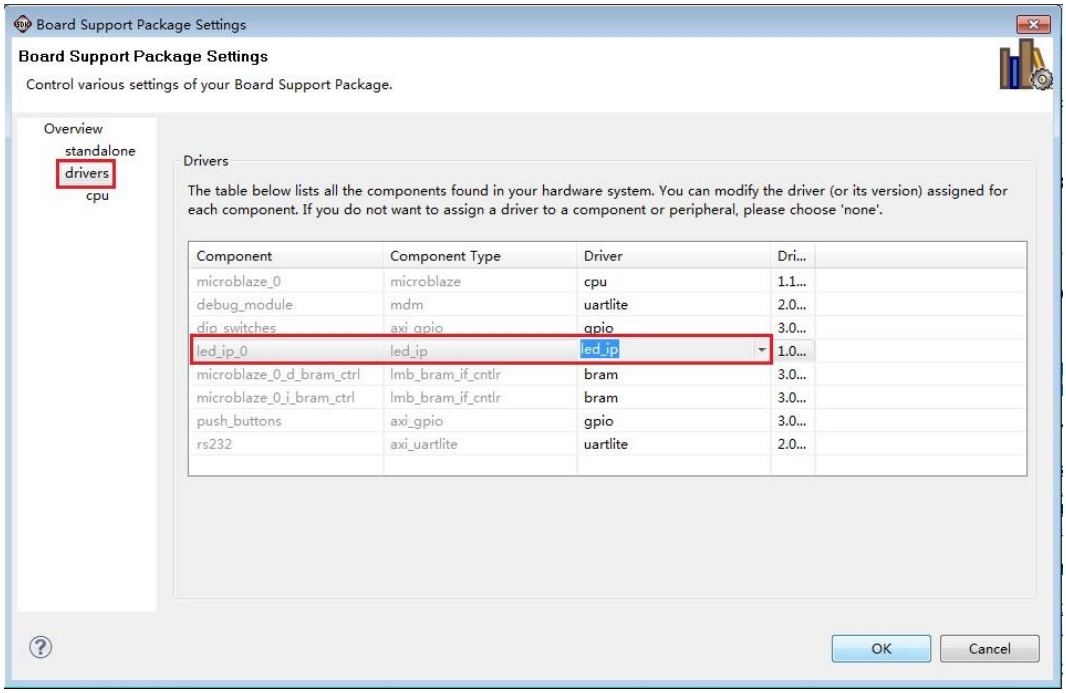


图 16：设置 led_ip

2-1-8.　现在我们在 test.c 中编写一个简单的测试程序，下面给出详细的源代码及说明：

```
#include "xparameters.h"
#include "xgpio.h"
#include "xutil.h"
#include "led_ip.h"

int main (void)
{
    XGpio dip, push,led;
    int i, psb_check, dip_check,Status;

    xil_printf("-- Start of the Program --\r\n");

    XGpio_Initialize(&dip, XPAR_DIP_SWITCHES_DEVICE_ID);
    XGpio_SetDataDirection(&dip, 1, 0);

    XGpio_Initialize(&push, XPAR_PUSH_BUTTONS_DEVICE_ID);
    XGpio_SetDataDirection(&push, 1, 0);

    Status = XGpio_Initialize(&led, XPAR_LED_IP_0_DEVICE_ID);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    XGpio_SetDataDirection(&led, 1, 0);
```

```
    while (1){
      psb_check = XGpio_DiscreteRead(&push, 1);
      xil_printf("Push Buttons Status %x\r\n", psb_check);
      dip_check = XGpio_DiscreteRead(&dip, 1);
      xil_printf("DIP Switch Status %x\r\n", dip_check);

      // output dip switches value on LED_ip device
      LED_IP_mWriteReg(XPAR_LED_IP_0_BASEADDR, 0, dip_check);

      for (i=0; i<999999; i++);
    }
}
```

图 17：LED 灯测试代码

程序主要功能：

1、 通过 Xgpio_Initialize()函数分别初始化 Switch 开关、Button 按钮和 LED 灯等多项设备；
2、 通过 Xgpio_SetDataDirection()函数设置 GPIO 数据传输方向；
3、 通过 LED_IP_mWriteReg()函数将 Switch 开关的操作值赋给 LED 寄存器中，从而实现通过 Switch 开关控制 LED 灯亮灭的效果。由于 Lab3 中我们只定义了 8 个 Switch 开关，所以现在能控制 8 个 LED 灯的亮灭；
4、 如果想调用自定义 LED 的 IP 核的相关函数，需要添加头文件“led_ip.h”，并通过阅读“led_ip.h”了解更多函数的用法及功能。
5、 若想了解更多 Xgpio 的函数功能，大家可以阅读“xgpio.h”。

2-1-9. 按照和刚才完全相同的步骤（selftest 文件就不用再修改了），我们可以同样添加流水灯实现的工程，并编写流水灯代码，下面给出详细的源代码及注释内容。

```
/***************************** Include Files *********************************/
#include "xparameters.h"
#include "xgpio.h"
#include "led_ip.h"
#include "xstatus.h"


/************************** Constant Definitions *****************************/

int main(void)
{
    XGpio led;
    int Status;

    Status = XGpio_Initialize(&led, XPAR_LED_IP_0_DEVICE_ID);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    XGpio_SetDataDirection(&led, 1, 0);

    int i,j,k;
    int t;                //周期计数器(范围0~199)

    int duty_led[16];     //LED的占空(范围0~200;数值越大,LED越亮;下同)
    int duty_led_buf[16];//缓冲

    int num[16];
    int reg=0b0000000000000001;
```

```
for(i=0;i<16;i++){
    if(i==0)
        num[i]=reg;
    else{
        reg=reg<<1;
        num[i]=reg;
    }
}

char a;

while(1){
    print("please make the choice!\n\r");
    a=inbyte();        //接受来自串口的数据

    for(i=0;i<16;i++)
       duty_led[i]=0;

    duty_led_buf[0] = 80;
    duty_led_buf[1] = 150;
    duty_led_buf[2] = 300;
    duty_led_buf[3] = 600;
    duty_led_buf[4] = 300;
    duty_led_buf[5] = 150;
    duty_led_buf[6] = 80;
    duty_led_buf[7] = 0;
    duty_led_buf[8] = 0;
    duty_led_buf[9] = 0;
    duty_led_buf[10] = 0;
    duty_led_buf[11] = 0;
    duty_led_buf[12] = 0;
    duty_led_buf[13] = 0;
    duty_led_buf[14] = 0;
    duty_led_buf[15] = 0;

   for( j = 0 ; j<50 ; j++ ){          //调节流水灯间隔时间
      for( i = 0 ; i<20 ; i++ ){       //调节流水灯速度
         for( t = 0 ; t<200  ; t++ ){ //根据占空比,控制每个LED的亮度
              if(a=='1')          //设置串口输入为1流水灯从左往右流动
              {
                for(k=0;k<16;k++){
                  if(t<duty_led[k])
                    LED_IP_mWriteReg(XPAR_LED_IP_0_BASEADDR,0,num[k]);
                  else       //0代表灯灭，1代表灯亮
                    LED_IP_mWriteReg(XPAR_LED_IP_0_BASEADDR,0,0b0000000000000000);
                 }
              }
              else              //设置串口输入为其他时，流水灯从右往左流动
              {
                for(k=0;k<16;k++){
                  if(t<duty_led[k])
                     LED_IP_mWriteReg(XPAR_LED_IP_0_BASEADDR,0,num[15-k]);
                  else
                      LED_IP_mWriteReg(XPAR_LED_IP_0_BASEADDR,0,0b0000000000000000);
                  }
              }
          }
    }

  //占空队列移动
    for(i=0;i<16;i++){
        if(i==15)
            duty_led[i]=duty_led_buf[0];
        else
```

```
            else
                duty_led[i]=duty_led[i+1];
        }

        for(i=0;i<16;i++){
           if(i==15)
               duty_led_buf[i]=0;
           else
               duty_led_buf[i]=duty_led_buf[i+1];
        }
    }
  }
}
```

图 18：流水灯完整代码

程序主要功能：

1、 通过 Xgpio_Initialize()函数初始化 LED；
2、 通过 Xgpio_SetDataDirection()函数设置 GPIO 数据传输方向；
3、 用户输入的数据会通过 inbyte()函数赋值给变量 a，程序根据变量 a 的值设置流水灯运行方向。while 循环会重复执行这段过程，从而反复接收用户的选择，多次运行这段程序。

第三步 上板验证

3-1. 将 Nexys4 与 PC 的 USB 接口连接

3-2. 查看端口号：

右键**“我的电脑”-“属性”**，在页面左侧选择**“设备管理器”。**
发现与 com4 端口相连：（不同电脑可能有所区别，同一电脑每次连接也有可能有所区别）。

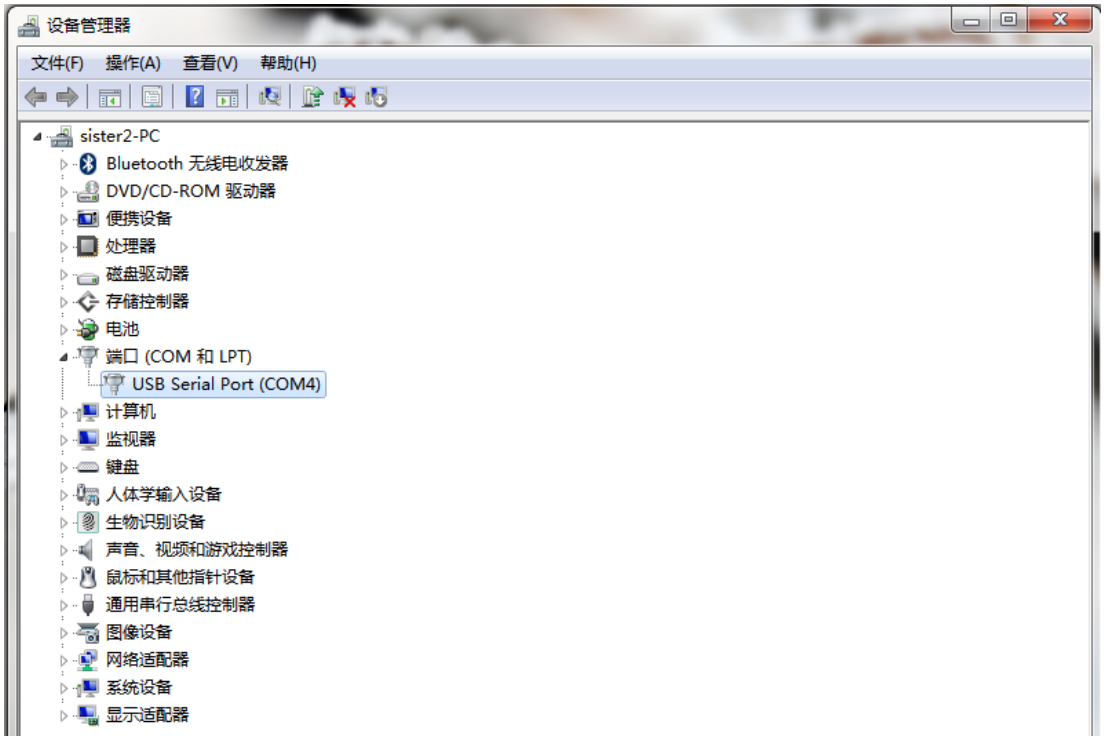


图 19：查看端口号

3-3. 在 SDK 中打开串口：

3-3-1. 在下面的在页面下方找到 terminal 选项卡，然后点击绿色的连接按钮。

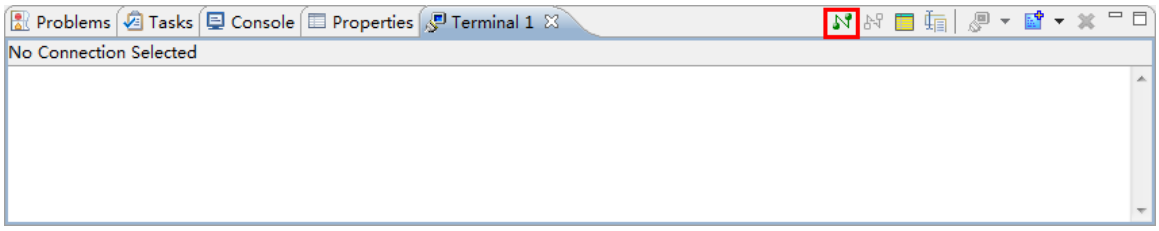


图 20：连接端口

3-3-2. 按照端口号和 XPS 中的波特率（baud rate）进行如下设置（要和配置硬件工程时为串口 IP 核设置的波特率一样）：

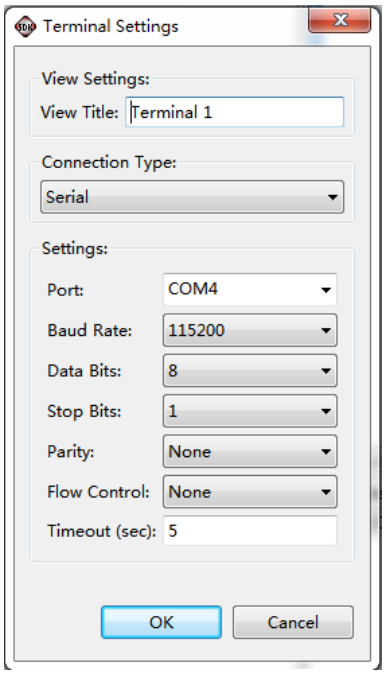
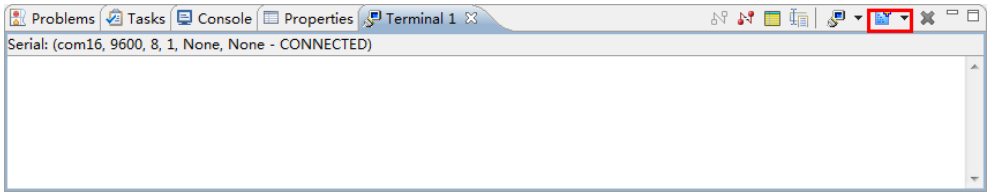


图 21：设置端口

如果报了“no such port”的错误，可以通过新建串口，更改串口号：



3-4. 将程序下载到板子上并运行

3-4-1. 在页面上方，xilinx tools 下拉菜单中选择“program fpga”一项。

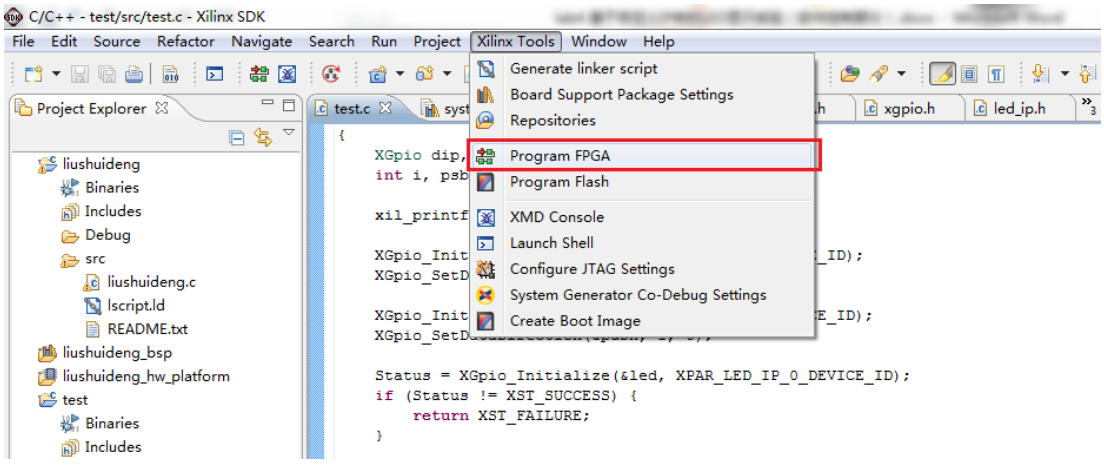


图 22：将 C 语言程序下载到开发板中

3-5-2.由于我们编写了两套软件，所以要注意选择正确的 elf 文件，并点击 program 运行程序：

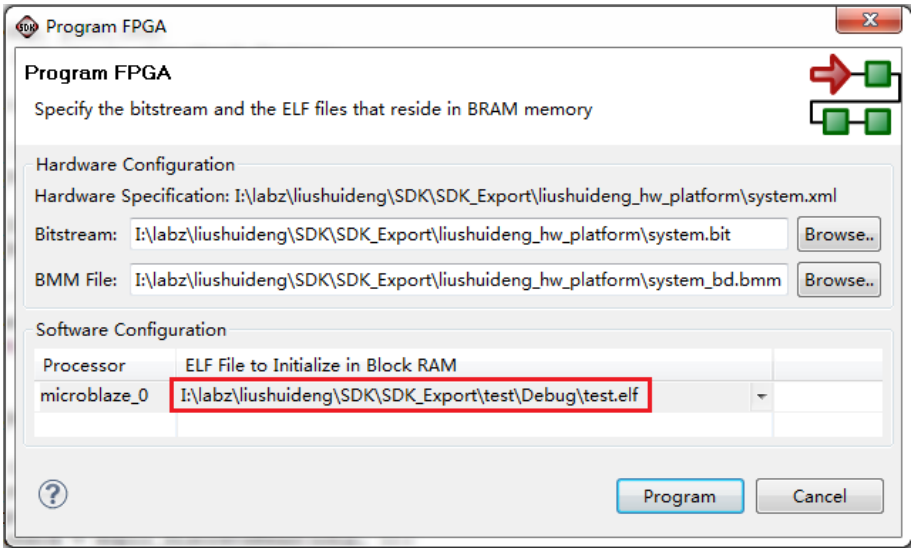


图 23：选择 test.c 的 elf 文件

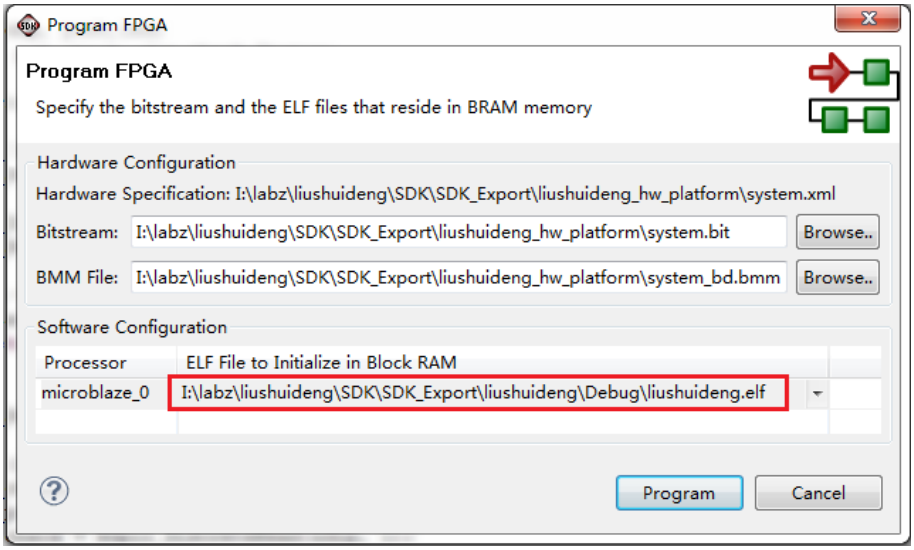


图 24：选择 liushuideng.c 的 elf 文件

3-5. 实验效果：

3-5-1. 运行 test.c 的实验效果

如果我们选择前 8 个开关中的第 1、3、5、7 个开关上拨，那么串口会显示以下效果：

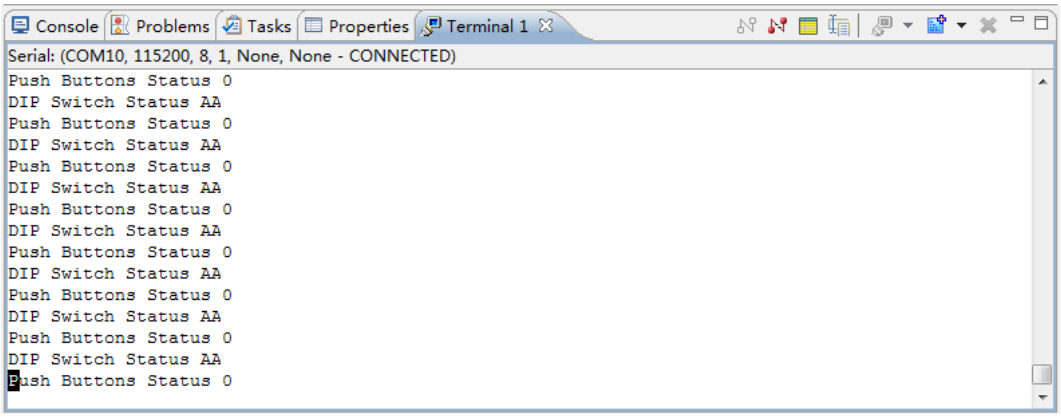


图 25：test.c 代码的串口显示效果

可以看到，在开发板的前 8 个 switch 开关中，如果开关上拨，LED 灯就会发亮。

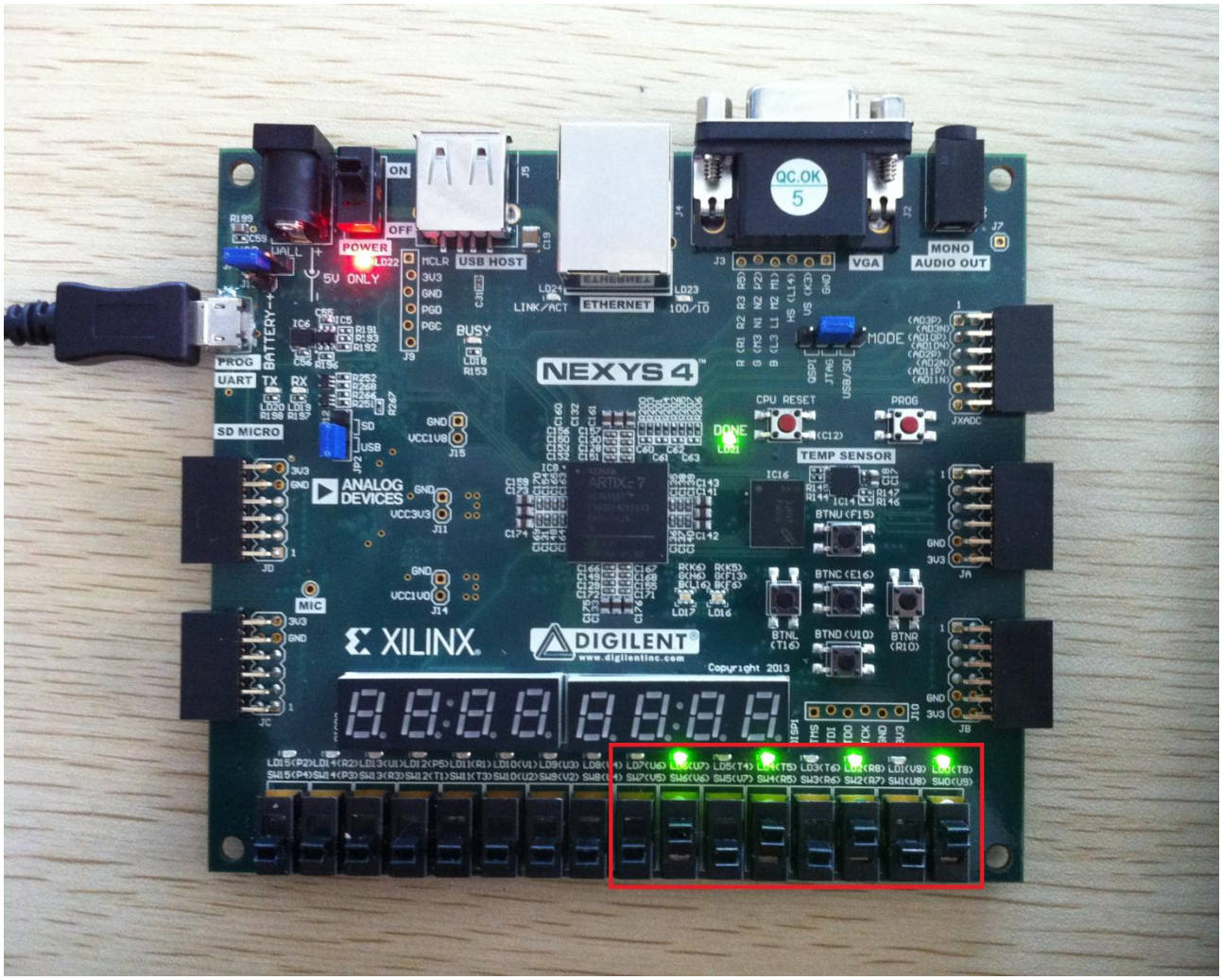


图 26：test.c 代码的板上运行效果

3-5-2. 运行 liushuideng.c 的实验效果

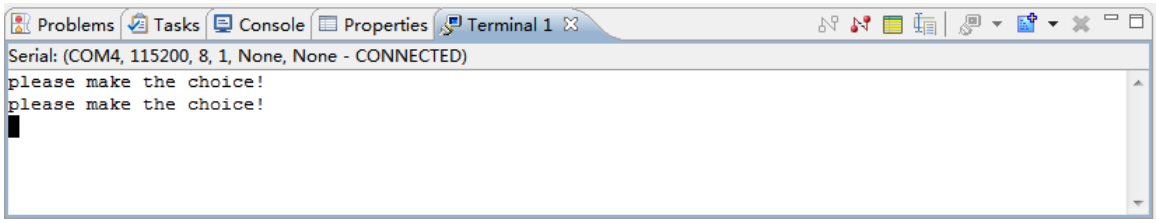
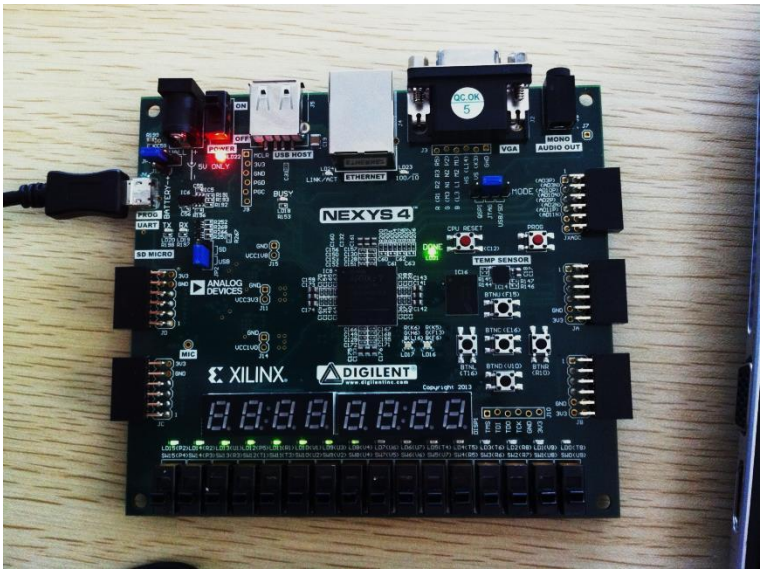
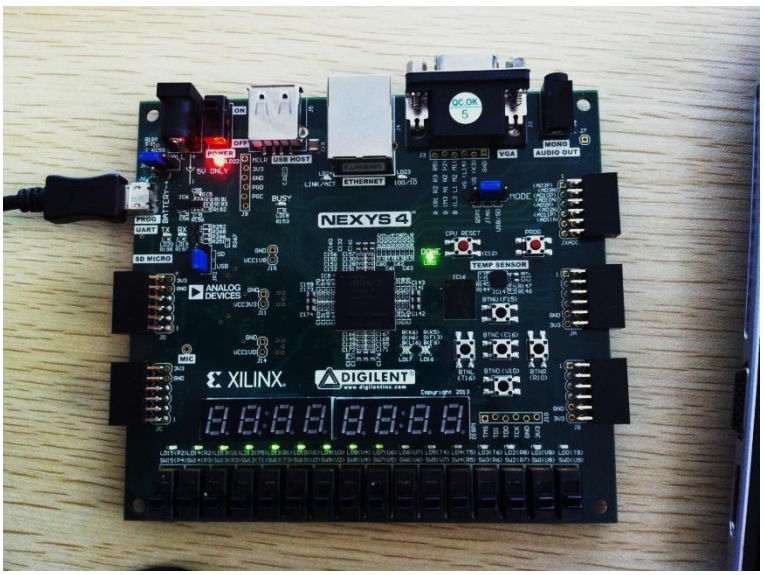


图 27：流水灯代码的串口显示效果

经过下载后，每次开发板的 LED 灯上运行一次流水灯程序后，串口都会出现“please make the choice”字样以提示使用者选择下一次流动方向。而且我们可以通过开发板观察到，每次亮起来的这些 LED 灯的亮度其实是不同的，有着渐明渐暗的效果。





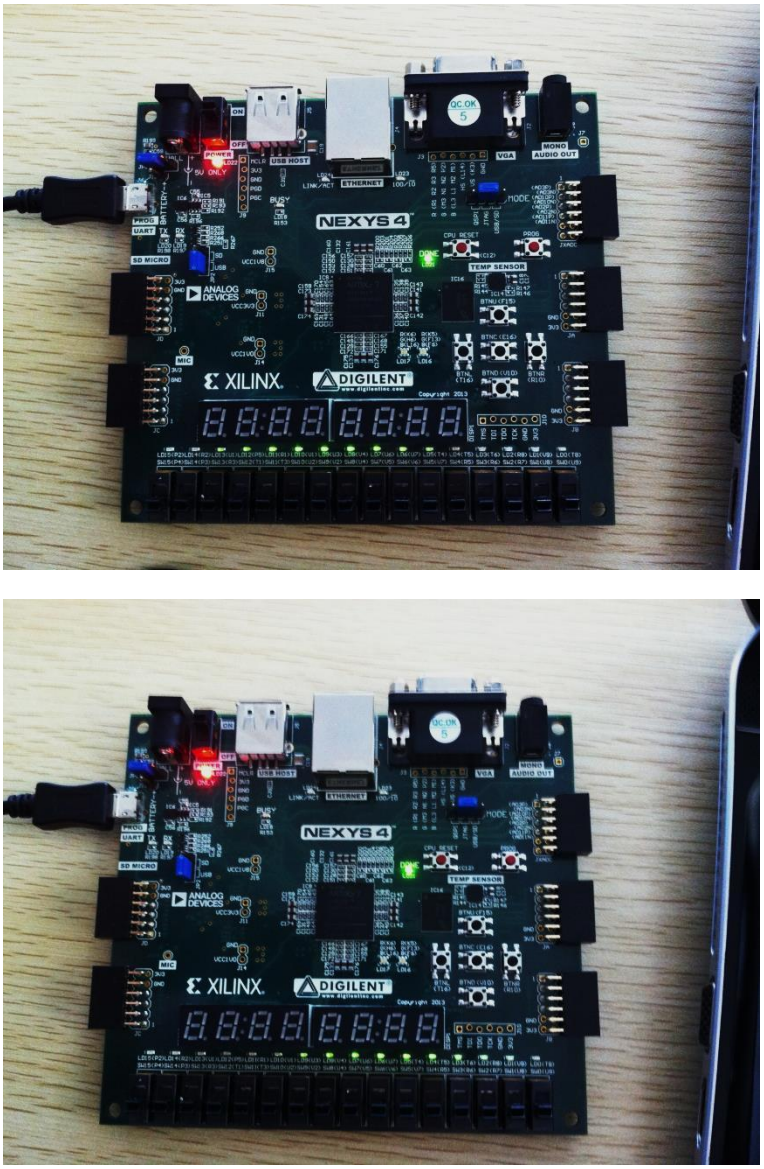


图 28：流水灯代码的板上运行效果