# 微

# 机

# 原

# 理

# 实

# 验

# 报

# 告

# 学院：电子信息与通信学院

# 班级：提高2101班

# 姓名：杨筠松

# 学号：U202115980

# 实验时间：2023年4月11日

并行IO接口实验

**1.实验任务**

嵌入式计算机系统将独立按键以及独立开关作为输入设备，LED 灯作为输出设备。修改实验示例程序代码，实现以下功能：

1） 点击BTNC 按键时，计算机读入一组16 位独立开关状态作为第一个输入的二进制数据，并即时显示输入的二进制数到16 位LED 灯上。（没有按下BTNC按键时，开关拨动不读入数据）

2） 点击BTNR 按键时，计算机读入另一组16 位独立开关状态作为第二个输入的二进制数据，并即时显示输入的二进制数到16 位LED 灯上。（没有按下BTNR按键时，开关拨动不读入数据）

3） 点击BTNU 按键时，将保存的2 组二进制数据做无符号加法运算，并将运算结果输出到LED 灯对应位。

4） 点击BTND 按键时，将保存的2 组二进制数据做无符号乘法运算，并将运算结果输出到LED 灯对应位。

程序控制方式提示：循环读取按键键值，根据按键的值读取开关状态，并做相应处理。

**2**.**实验目的**

（1）掌握GPIO IP核的工作原理和使用方法

（2）掌握中断控制方式的I0接口设计原理

（3）掌握中断程序设计方法

（4）掌握IO接口程序控制方法：

* 查询方式
* 中断方式

**3.实验环境**

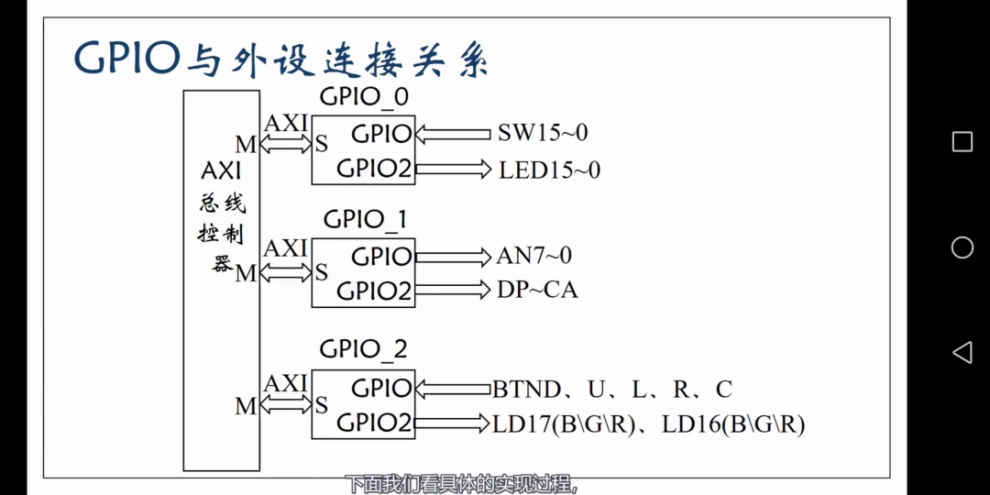
Windows 7 以上操作系统；

编辑工具：Vivado，SDK；

实验开发板：Nexys 4 DDR。

**4.设计方案**

硬件框图：



Nexys 4 DDR开发板的16个独立开关SW接GPIO\_0的第一个端口，16个LED发光二极管接GPIO\_0的第二个端口，数码显示管的位选信号AN7~0接GPIO\_1的第一个端口，数码显示管的片选信号DP~CA接GPIO\_1的第二个端口，5个独立按键BTND.BTNU.BTNL.BTNR.BTNC接GPIO\_2的第一个端口。

设计思路：

查询方式： 使用while(1)循环不断读取button的状态直到发生变化，根据不同的按钮被按下，设置对应的数据情况，最后端口输出即可。

中断方式：首先设置中断源的状态，并且注册总的中断程序，当中断发生时，根据不同的Button情况，设置对应的数据，最后端口输出即可

**5. 源代码**

**#include** "xintc\_l.h"

**#include** "xtmrctr\_l.h"

**#include** "xgpio\_l.h"

**#include** "xparameters.h"

**#include** "xio.h"

**#include** "xil\_exception.h"

**int** **main**(){

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set SW as input

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI2\_OFFSET, 0x0); // set LED as output

Xil\_Out16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set BTND, U, L, R, C as input

**unsigned** **short** data1 = 0, data2 = 0, data3 = 0, data4 = 0;

**while** (1) {

**unsigned** **short** status = 0;

status = Xil\_In16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_DATA\_OFFSET) & 0x1f;

**if** (status & 0x1) {

data1 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data1);

}

**if** (status & 0x8) {

data2 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data2);

}

**if** (status & 0x10) {

data3 = data1 \* data2;

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data3);

}

**if** (status & 0x2) {

data4 = data1 + data2;

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data4);

}

}

}

**#include** "xintc\_l.h"

**#include** "xtmrctr\_l.h"

**#include** "xgpio\_l.h"

**#include** "xparameters.h"

**#include** "xio.h"

**#include** "xil\_exception.h"

**int** interrupt = 0;

**void** **interrupt\_hub**(**void**) **\_\_attribute\_\_**((interrupt\_handler));

**void** **setup\_GPIO\_interrupt**(**void**);

**int** **main**(){

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set SW as input

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI2\_OFFSET, 0x0); // set LED as output

Xil\_Out16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set BTND, U, L, R, C as input

setup\_GPIO\_interrupt();

**return** 0;

}

**unsigned** **short** data1 = 0, data2 = 0, data3 = 0, data4 = 0;

**void** **interrupt\_hub**(**void**) {

xil\_printf("this is in the interrupt\n");

**unsigned** **short** status = 0;

status = Xil\_In16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_DATA\_OFFSET) & 0x1f;

**if** (status & 0x1) {

data1 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data1);

}

**if** (status & 0x8) {

data2 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data2);

}

**if** (status & 0x10) {

data3 = data1 \* data2;

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data3);

}

**if** (status & 0x2) {

data4 = data1 + data2;

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data4);

}

Xil\_Out32(XPAR\_GPIO\_2\_BASEADDR + XGPIO\_ISR\_OFFSET, XGPIO\_IR\_CH1\_MASK);

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IAR\_OFFSET, status); // clear AXI IAR status.

}

**void** **setup\_GPIO\_interrupt**(**void**) {

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IAR\_OFFSET, XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK); //clear AXI IAR status

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IER\_OFFSET, XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK); //set IER

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_MER\_OFFSET, XIN\_INT\_MASTER\_ENABLE\_MASK | XIN\_INT\_HARDWARE\_ENABLE\_MASK); // set the interrupt can be generated by software and hardware

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_IER\_OFFSET, XGPIO\_IR\_CH1\_MASK); //enable the channel 1 interrupt

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_ISR\_OFFSET, XGPIO\_IR\_CH1\_MASK); //clear the channel 1 interrupt status

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_GIE\_OFFSET, XGPIO\_GIE\_GINTR\_ENABLE\_MASK); //enable GPIO interrupt and output

microblaze\_enable\_interrupts();

}

**#include** "xintc\_l.h"

**#include** "xtmrctr\_l.h"

**#include** "xgpio\_l.h"

**#include** "xparameters.h"

**#include** "xio.h"

**#include** "xil\_exception.h"

**int** interrupt = 0;

**void** **interrupt\_hub**(**void**) **\_\_attribute\_\_**((fast\_interrupt));

**void** **setup\_GPIO\_interrupt**(**void**);

**int** **main**(){

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set SW as input

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI2\_OFFSET, 0x0); // set LED as output

Xil\_Out16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_TRI\_OFFSET, 0xffff); // set BTND, U, L, R, C as input

setup\_GPIO\_interrupt();

**return** 0;

}

**unsigned** **short** data1 = 0, data2 = 0, data3 = 0, data4 = 0;

**void** **interrupt\_hub**(**void**) {

xil\_printf("this is in the interrupt\n");

**unsigned** **short** status = 0;

status = Xil\_In16(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_DATA\_OFFSET) & 0x1f;

**if** (status & 0x1) {

data1 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data1);

}

**if** (status & 0x8) {

data2 = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data2);

}

**if** (status & 0x10) {

data3 = data1 \* data2;

xil\_printf("this is times case %x\n", data3);

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data3);

}

**if** (status & 0x2) {

data4 = data1 + data2;

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA2\_OFFSET, data4);

}

Xil\_Out32(XPAR\_GPIO\_2\_BASEADDR + XGPIO\_ISR\_OFFSET, XGPIO\_IR\_CH1\_MASK);

}

**void** **setup\_GPIO\_interrupt**(**void**) {

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_IER\_OFFSET, XGPIO\_IR\_CH1\_MASK); //enable the channel 1 interrupt

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_ISR\_OFFSET, XGPIO\_IR\_CH1\_MASK); //clear the channel 1 interrupt status

Xil\_Out32(XPAR\_AXI\_GPIO\_2\_BASEADDR + XGPIO\_GIE\_OFFSET, XGPIO\_GIE\_GINTR\_ENABLE\_MASK); //enable GPIO interrupt and output

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IAR\_OFFSET, XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK);

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IER\_OFFSET, XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK);

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_MER\_OFFSET, XIN\_INT\_MASTER\_ENABLE\_MASK | XIN\_INT\_HARDWARE\_ENABLE\_MASK);

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IMR\_OFFSET, XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK); // set the IMR as the fast interrupt

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IVAR\_OFFSET +

4 \* XPAR\_INTC\_0\_GPIO\_2\_VEC\_ID, (**unsigned** **int**)interrupt\_hub); // make the address into the IVAR

microblaze\_enable\_interrupts();

}

**6.实验总结**

通过这个实验，我学会了基本IO读写函数的使用方法，更加熟练了Vivado和SDK的使用方法以及加深了对NEXYS 4 DDR 开发板认识也对接口的利用，认识上有了更多的了解，加深了对中断模式的印象。