

**接口技术综合实验报告**

|  |
| --- |
| 姓 名：车春池 |
| 学 号：U201816030 |
| 小组成员：王耀 |
| 学 院：计算机科学与技术学院 |
| 专 业：计算机科学与技术 |
| 班 级：校交1801 |
| 指导教师：胡迪青 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2021年 06 月 15 日

目录

[**1.实验目的** 3](#_Toc72874590)

[**2.实验环境** 3](#_Toc72874591)

[**3.项目设计与实现** 3](#_Toc72874592)

[**3.1 总体设计与实现** 3](#_Toc72874593)

[**3.2 硬件实现** 3](#_Toc72874594)

[**3.3 软件实现** 3](#_Toc72874595)

[**4.系统运行结果** 3](#_Toc72874596)

[**5.总结** 4](#_Toc72874597)

[**6.源代码** 4](#_Toc72874598)

**1.实验目的**

1. 熟悉MIPSfpga的软硬件协同开发
2. 学会硬件调试方法

**2.实验环境**

操作系统：Windows 10

开发环境：

1. 硬件开发：Vivado 2018.2
2. 软件编译：MIPS工具链
3. GUI实现：Bevy游戏引擎

语言：

1. MIPSfpga软件：C
2. GUI实现：Rust

**3.项目设计与实现**

**3.1 总体设计与实现**

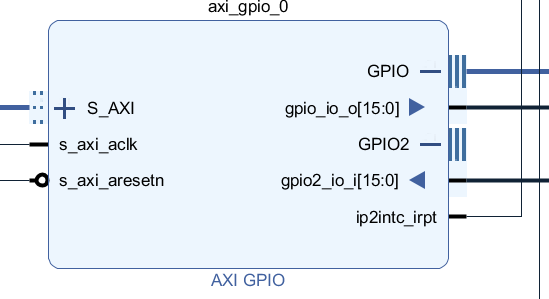
设计：通过FPGA上的串口功能，与X86主机进行信息交流，进而实现对游戏人物的移动控制。

实现：学校提供的FPGA实验平台有UART串口模块，通过串口转USB线可以实现和X86主机的信息收发，这点已经在前面的实验中实践过了。那么如何实现在程序中收到串口信息呢？很多编程语言都会有已经实现好的串口编程支持库，我们只需要调用API就行了。

**3.2 硬件实现**

硬件实现主要包括两部分，GPIO和UART模块。

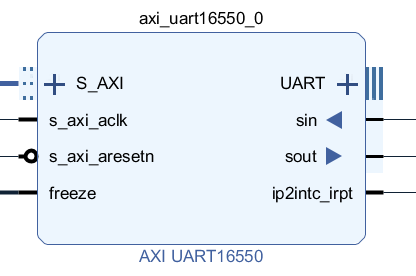
首先在vivado工程里面添加GPIO模块：



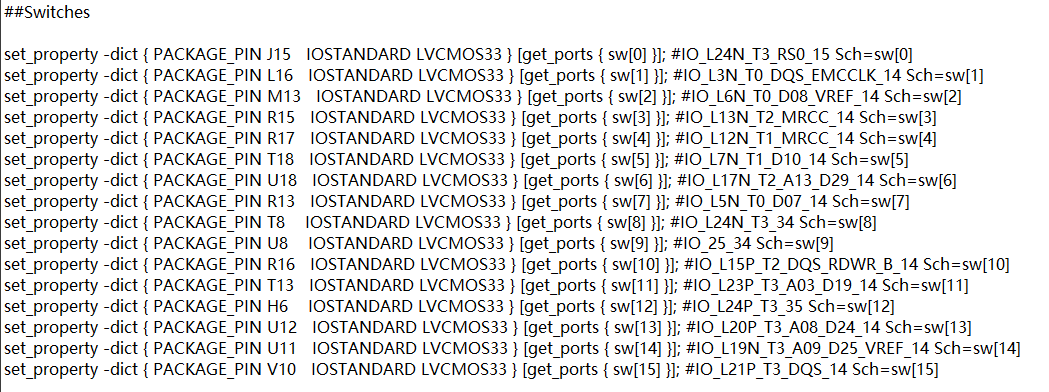
并且将中断源接到中断控制器7号端口：



然后添加UART模块并接到AXI总线上：



修改约束文件，将SWITCH相关的约束注释去掉：



最后综合，生成比特流烧到板子上。

**3.3 软件实现**

软件实现分两部分，第一部分是C语言程序，这部分程序将会被交叉编译为MIPS指令集的二进制文件烧到FPGA的内存中，然后在板子上运行。

在原先实验四代码的基础上，在中断处理函数\_mips\_handle\_irq里面添加对GPIO中断来源的判断，如果确认是来自GPIO的中断源，从GPIO里面读取值，然后通过UART发送到主机上。

代码：

void \_mips\_handle\_irq(void\* ctx, int reason) {

volatile unsigned int period;

volatile unsigned int rxData;

\*WRITE\_IO(IO\_LEDR) = 0xF00F; // Display 0xF00F on LEDs to indicate enter the interrupt

data\_received = 0x0;

if(reason & IS\_UART\_INTR && !(reason & IS\_TIME\_INTR)) {

/\* Read an input value from the console. \*/

rxData = \*READ\_IO(UART\_BASE + rbr);

data\_received = 0x1;

}

else if(reason & IS\_PWM\_INTR) {

// 设置 pwm 值为 0

\*WRITE\_IO(PWM\_BASE) = 0 \* 110000;

}

else if(reason & IS\_TIME\_INTR) {

asm volatile ("mtc0 $0, $11");

asm volatile ("li $9, 0x1");

asm volatile ("mtc0 $9, $9");

}

else if(reason & IS\_GPIO\_INTR) {

volatile unsigned int sw = 0;

sw = \*READ\_IO(SW\_LEDR);

uart\_outbyte((char)sw);

}

else {

uart\_print("Other interrupts occurred!\n\r");

uart\_print(my\_itoa(reason));

}

return;

}

然后GUI程序是使用了Rust语言编写，基于Rust语言开发的Bevy游戏引擎，我们实现了一款双人射击小游戏，两个角色分别通过键盘和FPGA板子进行控制。这里的关键问题是我们如何在Rust程序里面收到由FPGA发过来的信息呢？Rust社区里面有一个实现好的开源串口编程库，我们调用这个库的API即可较为方便地实现串口通信功能。这个库的github地址在这里：<https://github.com/SKTT1Ryze/serialport-rs-hotfix>。然后用Rust语言实现的小游戏代码的github地址在这里：<https://github.com/SKTT1Ryze/nexys4-player>。

软硬件结合起来的效果就是，拨动FPGA上的switch按键，会产生一个外部中断，这个中断在C程序里面的中断处理函数\_mips\_handle\_irq中被捕获，然后通过串口给主机发信息，主机收到信息后进行解析，然后角色进行相应的移动操作。这样子就实现了通过FPGA板子控制角色移动进行游戏。

**4.系统运行结果**

硬件连接：



游戏运行结果：



**5.总结**

通过本次实验，我把前面5个实验融会贯通，对软硬件协同更加熟悉，对硬件调试更加熟练了。这次实验充分激发了我对硬件的兴趣和编程欲望，学到了很多。希望以后如果有机会参与硬件相关的工作，这些知识可以派上用场。

**6.源代码**

C语言程序的main.c源代码：（Rust源代码太多了这里就不贴出来了）

/\*

\* main.c for microAptiv\_UP MIPS core running on Nexys4DDR

\* Prints \n\rMIPSfpga\n\r out via UART.

\* Default baudrate 115200,8n1.

\* Also display a shifting 0xf on the LEDs

\*/

#include "fpga.h"

#define inline\_assembly() asm("ori $0, $0, 0x1234")

#define CLK\_FREQ 50000000

#define UART\_BASE 0xB0401000 //With 1000 offset that axi\_uart16550 has

#define rbr 0\*4

#define ier 1\*4

#define fcr 2\*4

#define lcr 3\*4

#define mcr 4\*4

#define lsr 5\*4

#define msr 6\*4

#define scr 7\*4

#define thr rbr

#define iir fcr

#define dll rbr

#define dlm ier

#define PWM\_BASE 0xB0C00000

#define IS\_BTNU\_INTR (1<<10)

#define IS\_BTND\_INTR (1<<11)

#define IS\_BTNL\_INTR (1<<12)

#define IS\_BTNR\_INTR (1<<13)

#define IS\_BTNC\_INTR (1<<16)

#define IS\_GPIO\_INTR (1<<17)

#define IS\_UART\_INTR (1<<15)

#define IS\_PWM\_INTR (1<<14)

#define IS\_TIME\_INTR (1<<30)

void delay();

void uart\_outbyte(char c);

char uart\_inbyte(void);

void uart\_print(const char \*ptr);

char \*reverse(char \*s);

char \*my\_itoa(int n);

volatile unsigned int rxData = 0x0;

volatile unsigned int data\_received = 0x0;

unsigned char \*promt = "Select a Brightness between 0 and 9\n\r";

//------------------

// main()

//------------------

int main() {

volatile unsigned int count = 0xF;

volatile unsigned int j;

volatile unsigned int period;

\*WRITE\_IO(UART\_BASE + lcr) = 0x00000080; // LCR[7] is 1

delay();

\*WRITE\_IO(UART\_BASE + dll) = 27; // DLL msb. 115200 at 50MHz. Formula is Clk/16/baudrate. From axi\_uart manual.

delay();

\*WRITE\_IO(UART\_BASE + dlm) = 0x00000000; // DLL lsb.

delay();

\*WRITE\_IO(UART\_BASE + lcr) = 0x00000003; // LCR register. 8n1 parity disabled

delay();

\*WRITE\_IO(UART\_BASE + ier) = 0x00000000; // IER register. disable interrupts

delay();

\*WRITE\_IO(UART\_BASE + ier) = 0x00000001; // IER register. Enables Receiver Line Status and Received Data Interrupts

delay();

/\* Prompt the user to select a brightness value ranging from 0 to 9. \*/

// 故意触发 pwm 中断

// \*WRITE\_IO(PWM\_BASE) = 10 \* 110000;

volatile unsigned int sw = 0;

// uart\_print(promt);

while(1) {

\*WRITE\_IO(UART\_BASE + ier) = 0x00000001; // IER register. Enables Receiver Line Status and Received Data Interrupts

// LEDs display

\*WRITE\_IO(IO\_LEDR) = count;

count = count + 1;

for(j=0; j<1000; j++)

delay();

sw = \*READ\_IO(SW\_LEDR);

uart\_outbyte((char)sw);

delay();

// if(data\_received) {

// period = rxData - 0x30;

// \*WRITE\_IO(PWM\_BASE) = period \* 110000;

// data\_received = 0x0;

// uart\_print(promt);

// delay();

// }

}

return 0;

}

void delay() {

volatile unsigned int j;

// for (j = 0; j < (10000); j++) ; // delay

for (j = 0; j < (100); j++) ; // delay

}

void uart\_outbyte(char c) {

\*WRITE\_IO(UART\_BASE + thr) = (unsigned int) c;

delay( );

}

char uart\_inbyte(void) {

unsigned int RecievedByte;

while(!((\*READ\_IO(UART\_BASE + lsr) & 0x00000001)==0x00000001));

RecievedByte = \*READ\_IO(UART\_BASE + rbr);

return (char)RecievedByte;

}

void uart\_print(const char \*ptr)

{

while (\*ptr) {

uart\_outbyte (\*ptr);

ptr++;

}

}

//反转字符串

char \*reverse(char \*s)

{

char temp;

char \*p = s; //p指向s的头部

char \*q = s; //q指向s的尾部

while(\*q)

++q;

q--;

//交换移动指针，直到p和q交叉

while(q > p)

{

temp = \*p;

\*p++ = \*q;

\*q-- = temp;

}

return s;

}

/\*

\* 功能：整数转换为字符串

\* char s[] 的作用是存储整数的每一位

\*/

char \*my\_itoa(int n)

{

int i = 0,isNegative = 0;

static char s[100]; //必须为static变量，或者是全局变量

if((isNegative = n) < 0) //如果是负数，先转为正数

{

n = -n;

}

do //从各位开始变为字符，直到最高位，最后应该反转

{

s[i++] = n%10 + '0';

n = n/10;

}while(n > 0);

if(isNegative < 0) //如果是负数，补上负号

{

s[i++] = '-';

}

s[i] = '\0'; //最后加上字符串结束符

return reverse(s);

}

void \_mips\_handle\_irq(void\* ctx, int reason) {

volatile unsigned int period;

volatile unsigned int rxData;

\*WRITE\_IO(IO\_LEDR) = 0xF00F; // Display 0xF00F on LEDs to indicate enter the interrupt

data\_received = 0x0;

if(reason & IS\_UART\_INTR && !(reason & IS\_TIME\_INTR)) {

/\* Read an input value from the console. \*/

rxData = \*READ\_IO(UART\_BASE + rbr);

data\_received = 0x1;

}

else if(reason & IS\_PWM\_INTR) {

// 设置 pwm 值为 0

\*WRITE\_IO(PWM\_BASE) = 0 \* 110000;

}

else if(reason & IS\_TIME\_INTR) {

asm volatile ("mtc0 $0, $11");

asm volatile ("li $9, 0x1");

asm volatile ("mtc0 $9, $9");

}

else if(reason & IS\_GPIO\_INTR) {

volatile unsigned int sw = 0;

sw = \*READ\_IO(SW\_LEDR);

uart\_outbyte((char)sw);

}

else {

uart\_print("Other interrupts occurred!\n\r");

uart\_print(my\_itoa(reason));

}

return;

}