



**汇编与接口 课 程 设 计**

团队 实验报告

|  |  |
| --- | --- |
| 学 院 | 计算机学院 |
| 专 业 | 计算机科学与技术 |
| 指导老师 | 王娟 |
| 组 长 | 邹知秋 |
| 组 员 | 郑子帆、张驰 |
| 组长联系方式 | 18801364976 |

二O二三 年 9 月

# 目录

[第一章 项目简述 1](#_Toc146673250)

[第二章 组员分工 1](#_Toc146673251)

[第三章 设计目的 1](#_Toc146673252)

[第四章 设计环境 1](#_Toc146673253)

[第五章 设计原理及内容 2](#_Toc146673254)

[5.1 实验原理 2](#_Toc146673255)

[5.1.1 串行通信 2](#_Toc146673256)

[5.1.2 LCD字库设置 3](#_Toc146673257)

[5.1.3 Bmp格式图片 4](#_Toc146673258)

[5.2 项目整体架构 5](#_Toc146673259)

[5.2.1 AXI总线封装的蓝牙模块 5](#_Toc146673260)

[5.2.2 GPIO模块 6](#_Toc146673261)

[第六章 设计与实现 6](#_Toc146673262)

[第七章 测试 15](#_Toc146673263)

[第八章 问题及解决方法 16](#_Toc146673264)

[第九章 心得体会及总结 17](#_Toc146673265)

[第十章 参考文献有价值的资源推荐 17](#_Toc146673266)

# 项目简述

在计算机组成原理课设设计的CPU基础上，学习与熟悉Microblaze软核，基于精工实验开发板，设计并且实现了UART、蓝牙、LCD接口控制器，测试并形成了IP核，正确综合、实现、生成比特流并在精工板上实验正常。

本项目的特色为可以通过手机、PC端通过蓝牙向精工板发送数据并在板上的LCD屏显示。另外，通过ILA(Integrated Logic Analyzer)进行了对FPGA内部数字信号波形的实时监测，帮助调试项目。

# 组员分工

小组分工见表2.1 小组分工。

|  |  |
| --- | --- |
| **组员** | **分工任务** |
| 邹知秋 | * 项目配置、上板测试 |
| 郑子帆 | * AXI蓝牙模块编写与封装、录制视频 |
| 张驰 | * MicroBlaze软核编程、接口调试 |

表2.1 小组分工

# 设计目的

根据精工板资源，完成蓝牙接口模块和LCD接口模块的编写，通过MicroBlaze软核编程实现移动端通过蓝牙向精工板发送文字、图片消息并在板上的LCD屏上显示。

# 设计环境

|  |  |
| --- | --- |
| 操作系统 | Windows10 家庭版 |
| 编程语言 | Verilog HDL |
| EDA工具 | Vivado 2019.2、Vitis 2019 |
| 汇编语言 | MIPS |
| 汇编程序编辑器 | Mars4\_5 |

# 

# 设计原理及内容

## 实验原理

## 串行通信

A diagram of a computer program

Description automatically generated with medium confidence本实验设计到了数据通信方面的内容，我们需要在不同的地方使用串行通信协议以传输相关数据。这里我们主要用到了UART和SPI。

图表 1 UART协议通信方式

UART，通用异步收发器，是一个异步全双工通信协议，如上图1。在我们从主机向从机传输比特流数据时，高电平表示1，低电平表示0，每8位为一个单位进行传输，并从低位开始传。在每一段比特流前先从高电平到低电平表示起始位，传输8比特后，再从低电平转高电平表示停止位。这样主机和从机不一定需要同步工作即可传输数据。

而SPI，串行外围设备接口协议，是一个同步全双工通信协议。当主机向从机发送数据时，二者的时钟是共用的、相同的，如下图2。从主机向从机发送数据时，通过MOSI信号传输；从从机向主机发送数据时，通过MISO信号传输。而时钟SCK是二者共用的信号，表示同一个始终频率。和UART相比，UART只有TXD和RXD两个信号引脚，其中TXD表示发送端（要向外发送的数据），RXD表示接受段（从外面接收到的数据）。

A diagram of a building

Description automatically generated

图表 2 SPI协议通信方式

## LCD字库设置

本实验所用的精工板的LCD模块由LCD驱动IC ST7571、字库IC和背光组成。其中核心部分为字库IC，如下图3。

A black and white text on a white background

Description automatically generated

图表 3 字库IC

精工板可以从字库IC中读出内置的字库的点阵数据写入到LCD驱动IC 中，以达到显示汉字的目的。该模块最重要的引脚为ROM-IN、ROM-OUT、ROM-SCK、ROM-CS。其中ROM-IN用于串行数据输入，对应字库IC接口SI；ROM-OUT用于串行数据输出，对应字库IC接口SO；ROM-SCK用于串行时钟输入，对应字库IC接口SCLK；ROM-CS用于片选输入，对应字库IC接口CS#。

对于字库的调用，我们在本实验中使用了两种字符排列方式，分别为15x16点汉字排列和5x7点ASCII字符排列格式，具体的排列图示例如下图4和5。A diagram of a computer

Description automatically generatedA diagram of a number of digits

Description automatically generated with medium confidence

图表 4 15x16点汉字排列格式

对于中文汉字的显示，我们采用前者；而英文字母和数字的显示我们采用后者。值得注意的是显示的时候每竖着的8个为一个字节编码。

## Bmp格式图片

对于图片的输出显示，我们需要将bmp格式的图片转化成01点阵，其中0表示该像素不涂黑，1表示该像素块涂黑。

A screen shot of a computer program

Description automatically generated我们实验使用的是128\*128的BMP格式的图片。其中前63个字节为元信息数据，不需要处理，后面每个字节包含8个像素块的信息，但是需要延x轴进行翻转。最终的BMP图片转化点阵信息的python代码如下所示。

## 项目整体架构

在本次汇编与接口课程设计中，我们设计了蓝牙模块和LCD模块的接口，使得EES-338可以通过蓝牙模块接收到移动端发送过来的数据（包括文本内容和图片内容），并将其通过LCD模块内置的字库获得点阵信息，结合我们烧录的由MicroBlaze软核编写的硬件逻辑、Vitis编写的软件逻辑程序，将对应的字/图输出到LCD屏幕上。

对于区分发送的信息是文字还是图片，我们进行了如下指令的设定，如表。

表 1 功能与数据发送格式对应表

|  |  |
| --- | --- |
| **功能** | **发送形式** |
| 显示文字 | # {GB2312十六进制编码} # |
| 显示bmp图片 | \* {图片128x128十六进制编码} \* |

这样在软件设计的时候我们可以通过数据流首个字符是\*还是#以判断我们将要在LCD屏上显示什么内容。

总的设计图如图5，其中最主要的两个模块分别为蓝牙模块和GPIO模块。

A diagram of a computer

Description automatically generated

图表 5 总Block Design设计图

### AXI总线封装的蓝牙模块

本次实验我们使用的精工版的蓝牙模块出厂默认配置为通过串口协议与FPGA进行通信，用户无需研究蓝牙相关协议与标准，只需要按照 UART 串口通信协议来处理发送与接收的数据即可。

在我们设计的蓝牙模块中，主要需要包含三部分内容，分别为UART接收、命令处理（包括命令解析和响应）以及UART发送。

在UART接收模块中，我们通过UART串口通信协议得到了传送过来的命令，在命令处理模块中我们需要从AXI总线中的预设的寄存器取出URAT接收模块得到的数据，然后判断数据类型是文字数据还是图片数据，根据对应的数据格式调用不同的LCD功能函数进行显示。

在UART发送模块，即向数据发送端传输类似成功接收到数据的确认信息。

### GPIO模块

本实验中用到的GPIO模块用于模拟SPI时序，输出接口分别连接到LCD驱动和字库IC上。字库IC用于对特定的字符码，找到对应的点阵信息所在的位置；LCD驱动则是通过高低电平的置位和保持完成每一个像素块内的内容清除、内容显示等等。

# 设计与实现

这部分内容我们侧重于软件设计与实现，本实验的软件代码编写均在Vitis2019上完成。

对于AXI总线封装的蓝牙模块，核心部分的代码bt\_top.v如下：

`timescale 1ns/1ps

*module* bt\_top(

input wire clk\_pin,

input wire rst\_pin,

// Synchronous clk

output wire clk\_rx, // Receive clock

output wire clk\_tx, // Transmit clock

// Synchronous rst

output wire rst\_clk\_rx, // Reset, synchronized to clk\_rx

output wire rst\_clk\_tx, // Reset, synchronized to clk\_tx

// RS232 signals

input wire rxd\_pin, // RS232 RXD pin

output wire txd\_pin, // RS232 TXD pin

// Received data

output wire[7:0] rx\_data,

output wire rx\_buf\_rdy,

// Send data

input wire[7:0] tx\_data,

input wire tx\_buf\_en,

output wire tx\_buf\_full

);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Parameter definitions

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

parameter BAUD\_RATE = 9600;

parameter CLOCK\_RATE\_RX = 100\_000\_000;

parameter CLOCK\_RATE\_TX = 100\_000\_000;

wire rst\_i;

wire rxd\_i;

wire txd\_o;

// From Clock Generator

wire clock\_locked; // Locked signal from clk\_core

// From the character FIFO

wire [7:0] char\_fifo\_dout; // Character to be popped from the FIFO

wire char\_fifo\_empty; // The character FIFO is full

// From the UART transmitter

wire char\_fifo\_rd\_en; // Pop signal to the char FIFO

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Code

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Instantiate input/output buffers

IBUF IBUF\_rst\_i0 (.I (rst\_pin), .O (rst\_i));

IBUF IBUF\_rxd\_i0 (.I (rxd\_pin), .O (rxd\_i));

OBUF OBUF\_txd (.I(txd\_o), .O(txd\_pin));

// Instantiate the clock generator

clk\_gen clk\_gen\_i0 (

.clk\_pin (clk\_pin), // Input clock pin - IBUFG is in core

.rst\_i (rst\_i), // Asynchronous input from IBUF

.rst\_clk\_tx (rst\_clk\_tx), // For clock divider

.pre\_clk\_tx (), // Current divider

.clk\_rx (clk\_rx), // Receive clock

.clk\_tx (clk\_tx), // Transmit clock6

.clk\_samp (), // Sample clock

.en\_clk\_samp (), // Enable for clk\_samp

.clock\_locked (clock\_locked) // Locked signal from clk\_core

);

// Instantiate the reset generator

rst\_gen rst\_gen\_i0 (

.clk\_rx (clk\_rx), // Receive clock

.clk\_tx (clk\_tx), // Transmit clock

.clk\_samp (), // Sample clock

.rst\_i (rst\_i), // Asynchronous input - from IBUF

.clock\_locked (clock\_locked), // Locked signal from clk\_core

.rst\_clk\_rx (rst\_clk\_rx), // Reset, synchronized to clk\_rx

.rst\_clk\_tx (rst\_clk\_tx), // Reset, synchronized to clk\_tx

.rst\_clk\_samp () // Reset, synchronized to clk\_samp

);

// Instantiate the UART receiver

uart\_rx #(

.BAUD\_RATE (BAUD\_RATE),

.CLOCK\_RATE (CLOCK\_RATE\_RX)

) uart\_rx\_i0 (

.clk\_rx (clk\_rx), // Receive clock

.rst\_clk\_rx (rst\_clk\_rx), // Reset, synchronized to clk\_rx

.rxd\_i (rxd\_i), // RS232 receive pin

.rxd\_clk\_rx (), // RXD pin after sync to clk\_rx(unused)

.rx\_data\_rdy (rx\_buf\_rdy), // New character is ready

.rx\_data (rx\_data), // New character

.frm\_err () // Framing error (unused)

);

// Instantiate the Character FIFO - Core generator module

char\_fifo char\_fifo\_i0 (

.din (tx\_data), // Bus [7 : 0]

.rd\_clk (clk\_tx),

.rd\_en (char\_fifo\_rd\_en),

.rst (rst\_i), // ASYNCHRONOUS reset - to both sides

.wr\_clk (clk\_rx),

.wr\_en (tx\_buf\_en),

.dout (char\_fifo\_dout), // Bus [7 : 0]

.empty (char\_fifo\_empty),

.full (tx\_buf\_full)

);

// Instantiate the UART transmitter

uart\_tx #(

.BAUD\_RATE (BAUD\_RATE),

.CLOCK\_RATE (CLOCK\_RATE\_TX)

) uart\_tx\_i0 (

.clk\_tx (clk\_tx), // Clock input

.rst\_clk\_tx (rst\_clk\_tx), // Reset - synchronous to clk\_tx

.char\_fifo\_empty (char\_fifo\_empty), // Empty signal from char FIFO (FWFT)

.char\_fifo\_dout (char\_fifo\_dout), // Data from the char FIFO

.char\_fifo\_rd\_en (char\_fifo\_rd\_en), // Pop signal to the char FIFO

.txd\_tx (txd\_o) // The transmit serial signal

);

*endmodule*

LCD屏幕显示的相关函数代码lcd\_func.c：

*void* get\_and\_write\_16x16(ulong *fontaddr*,uchar *page*,uchar

*column*)

{

uchar i,j,disp\_data;

rom\_cs\_L;

send\_command\_to\_ROM(0x03);

send\_command\_to\_ROM((fontaddr&0xff0000)>>16);

send\_command\_to\_ROM((fontaddr&0xff00)>>8);

send\_command\_to\_ROM(fontaddr&0xff);

for(j=0;j<2;j++)

{

lcd\_address(page+j,column);

for(i=0; i<16; i++ )

{

disp\_data=get\_data\_from\_ROM();

transfer\_data(disp\_data);

transfer\_data(disp\_data);

}

}

rom\_cs\_H;

}

*void* get\_and\_write\_8x16(ulong *fontaddr*,uchar *page*,uchar *column*)

{

uchar i,j,disp\_data;

rom\_cs\_L;

send\_command\_to\_ROM(0x03);

send\_command\_to\_ROM((fontaddr&0xff0000)>>16);

send\_command\_to\_ROM((fontaddr&0xff00)>>8);

send\_command\_to\_ROM(fontaddr&0xff);

for(j=0;j<2;j++)

{

lcd\_address(page+j,column);

for(i=0; i<8; i++ )

{

disp\_data=get\_data\_from\_ROM();

transfer\_data(disp\_data);

transfer\_data(disp\_data);

}

}

rom\_cs\_H;

}

*void* display\_GB2312\_string(uchar *page*,uchar *column*, uchar \**text*)

{

uchar i= 0;

ulong fontaddr=0;

while(((uchar)text[i]>0x00))

{

if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))

{

fontaddr = (text[i]- 0xb0)\*94;

fontaddr += (text[i+1]-0xa1)+846;

fontaddr = (ulong)(fontaddr\*32);

get\_and\_write\_16x16(fontaddr,page,column);

i+=2;

column+=16;

}

else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))

{

fontaddr = (text[i]- 0xa1)\*94;

fontaddr += (text[i+1]-0xa1);

fontaddr = (ulong)(fontaddr\*32);

get\_and\_write\_16x16(fontaddr,page,column);

i+=2;

column+=16;

}

else if((text[i]>=0x20) &&(text[i]<=0x7e))

{

fontaddr = (text[i]- 0x20);

fontaddr = (ulong)(fontaddr\*16);

fontaddr = (ulong)(fontaddr+0x3cf80);

get\_and\_write\_8x16(fontaddr,page,column);

i+=1;

column+=8;

}

else

i++;

}

}

命令处理主程序main.c:

#include <stdio.h>

#include "xgpio.h"

#include "gpio\_initial.h"

#include "lcd\_func.h"

#include "bluetooth\_top.h"

uchar text\_buf[16 \* 128] = {0};

uchar row\_buf[20] = {0};

XGpio Gpio;

*int* main()

{

u32 rx\_data32 = 0, cnt = 0, mode = 0;

u32 now\_row = 1;

u8 rx\_data8 = 0;

gpio\_initial(&Gpio);

initial\_lcd();

clear\_screen();

while(1) {

usleep(100);

rx\_data32 = BLUETOOTH\_TOP\_mReadReg((*void* \*)XPAR\_BLUETOOTH\_TOP\_0\_S00\_AXI\_BASEADDR, 0);

rx\_data8 = (uchar)(rx\_data32 & 0xff);

if(rx\_data32 & 0x00000100) {

if(rx\_data8 == '#') {

if(cnt == 0) {

mode = 0;

cnt = 1;

}

else {

clear\_screen();

now\_row = 2;

for(*int* i = 0; i < cnt-1; i ++) {

row\_buf[i%14] = text\_buf[i];

if(i % 14 == 13) {

row\_buf[14] = '\0';

display\_GB2312\_string(now\_row, 0, row\_buf);

now\_row += 2;

if(now\_row == 14) now\_row = 2;

}

}

row\_buf[(cnt-2) % 14 + 1] = '\0';

display\_GB2312\_string(now\_row, 0, row\_buf);

now\_row += 2;

if(now\_row == 13) now\_row = 3;

cnt = 0;

mode = 10;

}

}

else if(rx\_data8 == '\*') {

if(cnt == 0) {

mode = 1;

cnt = 1;

}

else {

clear\_screen();

display\_graphic(bmp1);

cnt = 0;

mode = 10;

}

}

else {

text\_buf[cnt-1] = rx\_data8;

cnt ++;

}

}

}

return 0;

}

# 测试

因为我们本实验的内容的特殊性，不方便进行仿真测试，故直接采用下板测试的方法，验证了我们完成的项目的正确性、完整性，达到了实验设计时的预期。在下面的测试中，我们用到的移动端为搭载安卓系统的红米8手机，通过BLE调试助手APP连接到精工板完成测试。

1. **中、英文文字测试**

对于中、英文文字测试，我们准备了这样一段文字：此为北京理工大学邹知秋、郑子帆、张驰小组的汇编课设展示，from BITCS with love. 对于这样一段问题，首先我们再对应的GB2312编码网站将其转化为十六进制编码，为B4CB CEAA B1B1 BEA9 C0ED B9A4 B4F3 D1A7 D7DE D6AA C7EF A1A2 D6A3 D7D3 B7AB A1A2 D5C5 B3DB D0A1 D7E9 B5C4 BBE3 B1E0 BFCE C9E8 D5B9 CABE A3AC 66 72 6F 6D 20 42 49 54 43 53 20 77 69 74 68 20 6C 6F 76 65 2E。

在蓝牙成功连接后，将\*号从手机端发送到精工板上，表示即将发送文字信息，然后将上面得到的十六进制编码作为待传输内容，发送，最后再将\*号发送，表示一条完整的待输出信息。

这时观察LCD屏，应该显示如下图6的内容。

A screen with text on it

Description automatically generated

图表 6 中、英文显示测试结果

最终我们的测试结果和图6完全一样，测试成功。

1. **BMP图片测试**

对于BMP图片的测试，我们在网上找了图7这张图片并制成128\*128的BMP格式图片。

A black circle with white lines in it

Description automatically generated

图表 7 用于图片显示测试的图片

然后我们用5.1.3中的代码将得到的BMP图片转化成128\*128的点阵信息，这里注意得到的点阵信息为每列的8行为一个字节。

在蓝牙成功连接后，将#号从手机端发送到精工板上，表示即将发送图片信息，然后将上面得到的十六进制编码作为待传输内容，发送，最后再将#号发送，表示一条完整的待输出图片信息。

这时观察LCD屏，应该显示如下图8的内容。

A close up of a game

Description automatically generated

图表 8 图片显示测试结果

最终我们的测试结果和图8完全一样，测试成功。

# 问题及解决方法

在本次实验中，我们也并不是一帆风顺的，中途也遇到了一些小困难。如最开始移动端连接不上精工板到蓝牙模块，再多次调试和进一步查阅相关文档后，我们发现自己的设计中并没有涵盖多个开关sw\_pin的引脚配置，导致无法将精工板的蓝牙模块置成SLAVE模式，从而导致无法成功连接。在加入开关相关的引脚配置后，这一问题得以解决。

另外，在设计阶段我们没有考虑到区分文字和图片信息的相关事宜，使得在软件设计部分不知道该如何区分二者，而后制定了如表1的格式限定，解决的该问题。

再如，中间下板测试时遇到了通信失败的问题，即精工板接收不到手机发送的内容。在排查了很久之后发现，AXI总线的蓝牙模块封装的有问题，比如我们没有将本来作为输出接口的数据传给AXI总线临时的寄存器，导致在软件部分读取寄存器的内容为空，所以LCD屏没有输出。

当然，在整个实验的过程中，我们还会遇到一些比较杂的小问题，比如引脚约束配置错误、IP的配置出现差错等等，这些问题都在我们一遍又一遍的精读各个说明文档后得以解决。所以，从另一方面来讲，这个实验遇到的最大的一方面问题则是文档过于繁杂，难以找到自己需要的、有疑问的内容部分。

# 心得体会及总结

通过本次的实验，我们阅读了大量的Xilinx官方的文档，同时也从乐学上多份文档学习了很多有关本实验用到的接口的下至底层电路原理，上至应用代码编写的诸多知识。这次实验也是一个非常好的机会让我们能够自己动手实践完成接口设计和上板操作，结合大三学年学习过的数字逻辑、计算机组成与体系结构和汇编语言与接口设计等相关课程的理论知识，对整个计算机底层逻辑和接口设计都有了更深的理解和更全方面的认知。

当然，这次的实验也是小组所有人合作共同完成，小组中的每个成员都有自己主要负责的部分。如何将自己的部分更好的完成以及将多个部分充分融合、对接，小组成员的责任意识以及相互之间的交流便显得至关重要。阅读大量的文档增长了我们的理论基础和实践能力；组员间相互讨论、完善又丰富了我们多人合作完成项目的经历，给未来合作完成更复杂的项目打下坚实的基础。

# 参考文献有价值的资源推荐

1. 官网：Microblaze Quick Start Quide landing page
2. 官网：Xilinx Artix-7 FPGA landing page
3. JLX128128G-81202-PC中文说明书.pdf
4. EES-338\_UserManual\_v1.0.pdf
5. lab09\_Microblaze嵌入式软核实验.pdf
6. lab10\_LCD\_display.pdf