分类号：_____ 密级：_____

U D C ：_____ 编号：_____

工学硕士学位论文

# 基于 FPGA 的车路协同智能路侧系统的设计与实现


硕 士 研 究 生 ：李定洋

指 导 教 师 ：康维新 教授

学 科 、 专 业 ：电子科学与技术

论 文 主 审 人 ：黄丽莲 教授



哈尔滨工程大学

2014 年 12 月

分类号：_____ 密级：_____

U D C ：_____ 编号：_____

工学硕士学位论文

# 基于 FPGA 的车路协同智能路侧系统的设计与实现


**硕 士 研 究 生** ：李定洋

**指 导 教 师** ：康维新 教授

**学 位 级 别** ：工学硕士

**学 科 、 专 业** ：电子科学与技术

**所 在 单 位** ：信息与通信工程学院



**论文提交日期** ：2014 年 12 月

**论文答辩日期** ：2015 年 3 月

**学位授予单位** ：哈尔滨工程大学

Classified Index:
U.D.C:

A Dissertation for the Degree of M. Eng

# Design and Implementation of Intelligent Roadside System of Cooperative Vehicle Infrastructure based on FPGA

**Candidate:** Li Dingyang
**Supervisor:** Prof. Kang Weixin
**Academic Degree Applied for:** Master of Engineering
**Specialty:** Electronic Science and Technology
**Date of Submission:** Dec. 2014
**Date of Oral Examination:** Mar. 2015
**University:** Harbin Engineering University

# 哈尔滨工程大学
# 学位论文原创性声明

本人郑重声明：本论文的所有工作，是在导师的指导下，由作者本人独立完成的。有关观点、方法、数据和文献的引用已在文中指出，并与参考文献相对应。除文中已注明引用的内容外，本论文不包含任何其他个人或集体已经公开发表的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者（签字）：

日期：　　年　月　日

# 哈尔滨工程大学
# 学位论文授权使用声明

本人完全了解学校保护知识产权的有关规定，即研究生在校攻读学位期间论文工作的知识产权属于哈尔滨工程大学。哈尔滨工程大学有权保留并向国家有关部门或机构送交论文的复印件。本人允许哈尔滨工程大学将论文的部分或全部内容编入有关数据库进行检索，可采用影印、缩印或扫描等复制手段保存和汇编本学位论文，可以公布论文的全部内容。同时本人保证毕业后结合学位论文研究课题再撰写的论文一律注明作者第一署名单位为哈尔滨工程大学。涉密学位论文待解密后适用本声明。

本论文（□在授予学位后即可　□在授予学位 12 个月后　□解密后）由哈尔滨工程大学送交有关部门进行保存、汇编等。

作者（签字）：　　导师（签字）：

日期：　　年　月　日　　年　月　日

# 摘　　要

随着我国经济的迅速发展，人民生活水平不断提升，车辆的保有量迅速增加，交通问题也愈发严重，给交通监管部门带来了很多困扰。智能交通系统（Intelligent Transport System,简称 ITS）作为以后交通系统的发展方向，可以有效地利用现在的交通设施、减少了交通负荷和环境污染、保证了交通安全、提高了交通运输的效率，因而，越来越受到人们的重视。而车路协同作为智能交通系统的最新发展方向，有着极为广阔的发展前景。

本文针对车路协同的重要组成部分智能路侧系统进行分析，总结现有成果的基础上，设计了一种相对完善的路侧系统，安装在道路或者路口，旨在监控道路状况、交通堵塞、交通事故等信息，并引导车辆行驶，尤其着力于事故最为集中的道路交叉点，大大缓解了交通压力。

由于目前单片机功能相对简单、通信速率不高，已经无法担负起大规模智能路侧系统设计的重任，而 FPGA 以其内部集成锁相环运行速度快，丰富的 I/O 资源，高效率的并行执行机制，开发周期短，还可以构建软核进行二次开发等优势，逐渐成为嵌入式开发的主流平台和技术手段。所以，本文以 FPGA 为开发平台，使用 Verilog HDL 硬件编程语言编写摄像头采集显示模块和交通信号灯配时模块，同时采用 SOPC 技术构建一个 Nios 软核处理器，基于 TCP/IP 传输协议完成与控制中心和车载终端的通信，同时完成路侧引导 LCD 屏的显示工作。车载终端并非智能路侧系统的重点，所以采用了安卓手持设备模拟，利用设备自带 wifi 无线通信模块，将车辆信息发送给智能路侧系统；控制中心采用 Qt 界面编程技术，能够更加直观地观察到接收到的车辆信息，方便向智能路侧系统发布控制信息。

经过实际运行测试，本文设计的智能路侧系统性能稳定，数据的网络收发准确快速，交通信号灯能很好地根据配时方案运行，路侧引导 LCD 屏工作正常，视频采集的图像较为流畅，较好地满足了设计要求。

**关键词**：智能路侧系统；FPGA；交通控制；TCP/IP；Qt；视频采集与图像信息公告

# ABSTRACT

With the rapid development of our country's economy, people's living standards continue to improve, the vehicle ownership increases rapidly, traffic problems becomes more serious, which brings a lot of trouble to transportation regulators. As the future direction of the transportation system，Intelligent Transport System（ITS）can effectively utilize existing transportation facilities, reduce traffic load and environmental pollution and ensure traffic safety, improve transport efficiency, and therefore, it catches more people's attention. And as the latest development direction of intelligent transportation system, the collaborative carriageway has extremely broad prospects.

This article analyzes the smart roadside system which is the important part of collaborative carriageway. On the base of summarizing existing achievements, in order to monitor road conditions, traffic congestion, traffic accidents and other information, as well as to guide the vehicle, especially to focus on the most accident concentrated road intersection. This article designs a relatively complete system of roadside installed in the road or intersection, which greatly ease the traffic pressure.

Due to the current single-chip functionality is relatively simple, the communication speed is not high, has been unable to take on large-scale intelligent roadside system design task. But FPGA gradually becomes the mainstream of embedded development platform and technology with the advantages of its fast running internal integrated PLL, rich I/O resources, efficient parallel execution mechanisms, short development cycle and its ability to build a second soft-core development. Therefore, this article uses the FPGA as development platform, using Verilog HDL hardware programming language to write camera capture display module and traffic lights timing module, while using SOPC technology to build a Nois soft-core processor, based on the TCP/IP transport protocol to complete the communication between the control center and car terminals and complete the roadside guide LCD screen display work at the same time. As the intelligent vehicle terminal is not the focus of roadside systems, Android handheld device emulation is used, using device's own wifi wireless communication module to send vehicle information to smart roadside system. The control center uses Qt programming interface, which can more directly view received vehicle information and

makes it convenient to send control information to smart roadside system.

After the actual test run, the smart roadside system designed in this article performs stably, the network send and receive data quickly and accurately, traffic lights can run the program in accordance with good time, roadside guide LCD screen works well, video capture images relatively smooth, the system better meet the design requirement.

**Keywords**: Intelligent roadside system; FPGA; Traffic control; TCP/IP; Qt; Video capture and image information bulletin

# 目　　录

# 第 1 章　绪论

## 1.1　课题的研究背景与意义

随着经济的加速增长，我国交通运输业也快速发展起来，汽车已经变得越来越普遍，成为了人们生活当中不可或缺的一部分，道路交通也变得越来越便捷，但是车辆等交通工具作为交通网络的主要参与者，导致城市交通量与日俱增[1]。在道路交通设施有限的情况下，这就加大了车路矛盾，其中以下两个问题备受关注：

1.交通拥堵问题

据统计，2014 年 2 月我国机动车总量已到达 2.53 亿辆，其中汽车保有量就有 1.37 亿辆。对于我国的大中型城市来说，特别是北京、上海等一线城市，我国机动车保有量的增长速度远远超过各种道路里程数的增长速度。交通拥堵问题已成为影响交通的一个巨大难题。无论在我国经济发展方面还是在能源方面，严重的城市交通拥堵都会给我国造成巨大的损失。相关部门已采取提高道路里程、增加道路宽度等办法来缓解交通拥堵给城市交通带来的压力。但是面对城市人口数量激增和汽车保有量的飞速增长，利用土地资源加大交通道路规模以及加大交通道路上的资金投入等方法来解决交通拥堵问题，并不能从根本上解决交通拥堵给城市交通带来的难题。

2.交通安全问题

国内交通事故每年都会发生几十万起，造成很大的人员伤亡，交通事故的死亡率已经不容忽视。除了中国，汽车保有量较高的几个国家，如美国，日本，德国，俄罗斯因为交通事故而死亡的人数也一直居高不下。

仅仅依赖于传统的交通管理模式，棘手的交通问题并不能有效的解决。基于上述背景，智能交通系统（ITS）应运而生。ITS 将道路与车辆相结合进行研究，以计算机技术、通信技术及自动控制技术等先进技术为基础，目的在于系统地解决城市交通中的拥堵问题、安全问题、环境问题等一系列棘手问题。作为 ITS 的一个新兴发展领域，车路协同系统（Cooperative Vehicle Infrastructure System，简称 CVIS），也得到各个国家的重视，并投入研究和应用。

利用先进的无线通信技术和现代互联网等技术，车路协同系统（CVIS）可以使实时道路信息全方位在车与车、车与路之间达到动态交互，在车辆主动安全控制和道路协同管理中将全时空动态交通信息采集与融合技术融入其中，达到人、车、路的协同管理，提高交通系统运行的安全性以及道路网络通行效率，使得现代智能交通系统能够畅通、

安全、环保的有效运行[2]。全面开展车路协同系统的研究，重点解决车路协同的技术难题，不仅可以推动我国智能交通系统技术的快速发展，而且还能有助于我国的智能交通系统成为交通产业核心竞争力，也对我国城市交通的管理体系现代化发展，以及交通环境和设施的可持续战略具有深远意义[3]。在车路协同系统中，智能路侧系统作为车路协同的关键技术之一，它的研究进度，直接影响着车路协同技术的发展，所以该系统的研究进展，对于解决当前交通问题有着非常重要的意义。通过设计智能路侧系统，控制中心可以通过该系统全面监控整个监控范围内的所有交通状况以便统一管理，通过采集到的车辆情况反馈信息，提前干预发生状况的车辆。同时，智能路侧设备也可以将前面路段的拥堵信息及时传递给司机，通知司机绕路而行，减轻了交通拥堵的压力。当车辆经过危险路段，会收到来自路侧设备发出的警告信息，提醒司机注意，避免了交通事故的发生。总之，研究智能路侧技术可以使整个城市乃至全国的智能交通管理系统更高效的运行，同时将给交通产业带来非常高的经济效益和社会效益。

## 1.2 国内外发展及研究现状

车路协同作为智能交通系统的发展重点，一直受到各国的重视，目前已有多个国家对这一课题进行研究，并取得了一定的成果。

美国引领着全球的先进技术，对车路协同技术有着深入的研究，美国的 ITS 项目起始于 20 世纪 90 年代，自 1991 年美国通过了 ISTEA 法案之后，对交通的重视也与日俱增。随后，完成了加州 AHS 演示项目，又开始实施新的 IVI(Intelligent Vehicle Initiative)项目[4]。2005 年，美国提出了汽车基础设施的整合计划（VII），设想是在美国的每辆车上装备通讯设备以及 GPS 模块，以达到与广域道路网进行信息交换的目的。2007 年，IntelliDrive（智能驾驶）项目设立，IntelliDrive 计划在 VII 的已有成果上继续发展研究车路协同技术，该计划主张用人车路协同的方法来处理道路交通的某些短板。

日本作为一个能源极度缺乏，土地面积不足的国家，对于提高交通运输效率有着迫切的需求。其在车路协同方面的研究上起步也比较早，在 1995 年，日本便开始开发车辆信息和通信系统（Vehicle Information and Communication System，简称 VICS），该系统是在日本的首次尝试，此后 VICS 发展越来越成熟。2001 年，日本研发了电子不停车收费系统（ETC），随后提出了 ASV（先进安全车辆）项目计划。2007 年之后，日本将以上几个项目和通信技术有机整合起来，提出了 SmartWay 系统，目前该系统进入了技术全面推广阶段。之后在 2005-2010 年间，其又对若干关键技术继续展开研究，包括车路的协调系统技术等。总体来说，日本走在智能交通技术研究的前列。

西欧国家在车路协同上的研究也没有被落下，欧盟 ITS 组织 ERTICO 在第十届 ITS 世界大会上首先提出了 eSafety 的基本概念，并且得到了欧盟委员会的认可，列入了欧盟的计划当中，其最重要的内容是充分的利用通信领域中的核心技术，解决车辆通行安全问题，提高人们的出行安全[9]。eSafety 有几十项的研发项目把车路通信和协同控制作为研究的要点，其中重要的有 PreVENT 项目，利用较先进的通信、信息和定位等技术，开发了两种主动安全系统：自主式与协调式，来降低交通事故发生的几率和减小事故造成的损失；还有 Car2car 项目，使车车、车路通讯技术继续向前发展，也推动着其接口的标准化；除此之外，还有 CVIS 项目，车路多种方式混合通信解决方案等等。

相比国外，我国车路协同研究起步比较晚，从 2000 年开始，我国才逐渐关注车路协同领域，国家在“十二五”期间开始关注智能交通领域，很多城市随后才开始逐渐重视起城市的智能化交通的研究。2010 年，中国工信部批准了中铁通信集团公司申请的基于 TD-LTE 的高速铁路宽带通信的关键技术研究和应用验证课题的申报书。同年,“863”计划中也启动了智能交通领域的项目，如：车路协同和城市交通协同等。智能交通系统最重要的是智能车路协同系统，在智能交通系统中有至关重要的作用，依托国家“863”重点项目，全国诸多高校组成的科研团队也正在智能车路协同系统中的关键技术领域进行着研究。自 2011 年到今天，清华大学发起组织的科研团队在国家“863”计划的支撑下，紧密结合车路协同领域的核心技术进行了前瞻性的发展和创新。2014 年 2 月，清华大学组织的“十二五”“863”关键性项目“智能车路协同关键技术研究”通过科技部组织的验收。该项目的成果演示发布会与验收报告会在清华发展研究院与其试验场地举行，多部装备有车-车、车-路协同等仪器的“智能化车”，在“智能道路”上一起进行试验，先后展示了如盲区的预警等十余种智能车路协同系统中典型的应用场景；2014 年 10 月，在智能交通系统国际会议（ITSC-2014）上模拟了若干典型的应用场景，很大程度上实现了人、车、路协同的功能。

从国内外 ITS 系统发展的历程和现状来看，国外车路协同的研究比中国的更具有规模，但总体来说，车路协同的相关技术还是处于探索和测试阶段，并没有全面投入全国性的应用当中，而车路协同技术中的智能路侧系统技术同样也处于这一探索和完善的时期。

## 1.3　课题研究任务及指标

车路协同系统是各种技术、信息和功能的集合体，其中车辆与路侧、车辆与车辆之间的通信需求无所不在，接收数据和处理数据的方法也各不相同，所以车路协同系统需

要有一个完善的体系结构，这样才能为交通方面的设计、开发人员提供所需要的最核心的指导。根据架构，车路协同系统可分为智能车载系统（车辆）与智能路侧系统（基础设施）两部分[5]。本文主要针对智能路侧系统研究分析，设计了一个较为完善的智能路侧系统，这个系统需要通过 wifi 收集车载设备的信息，并与控制中心进行交互，将车辆信息发送给控制中心，控制中心接收到这些信息并进行处理，给路侧系统发送相应的控制信息，来控制智能路侧系统下的交通信号系统，来引导车辆的运行，提高了交通的运输效率。本系统经过测试，运行状况良好，无障碍物遮挡的情况下满足 wifi 通信距离不小于 50 米，通信协议符合国家标准，路侧的交通控制系统稳定可靠，视频采集显示的图像实时性好，延迟不超过 0.1s，图像的分辨率是 640×480，画质比较清晰，再经过后期完整评测，可以应用于实际交通中。

本文的结构安排如下：

第 1 章，介绍了智能交通系统的车路协同的研究背景和意义以及国内外的研究现状，在此基础上提出了本文的研究任务是围绕车路协同系统的智能路侧系统展开的，同时介绍了系统的性能指标，确定了文章的主要工作内容和结构安排。

第 2 章，介绍了系统的整体设计方案以及各设计模块划分，还介绍了系统的环境搭建，总体电路的连接，以及 Quartus II 中相关组件的添加与配置。

第 3 章，是智能路侧系统中交通控制系统的设计部分，首先介绍了交通系统的控制分类，然后介绍了交通灯、数码管、12864LCD 液晶屏的硬件电路的设计，最后介绍了软件实现对它们的控制与显示。

第 4 章，是智能路侧系统中网络通信模块的设计，介绍了一款网络芯片 ENC28J60 及其硬件电路设计，之后介绍了 TCP/IP 传输协议，完成了通信模块的软件设计，最后介绍了 Qt 界面编程的相关技术，完成了我们控制中心界面的设计。

第 5 章，是视频采集与图像信息公告模块的设计，首先我们对视频采集传感器进行了挑选，选择了一款性价比较高的数字摄像头 OV7670，然后介绍了 OV7670、SDRAM 和 VGA 接口的电路原理图，最后介绍了使用 Verilog HDL 语言模块化设计实现的过程以及相关技术。

第 6 章，介绍了系统调试方法、系统测试方法以及对调试和测试的结果进行了分析，验证系统的有效性。

最后结论，总结了本文的研究过程，对研究的后续工作进行了展望。

# 第 2 章　系统总体设计方案

## 2.1　系统总体结构介绍

车路协同智能路侧系统是车路协同系统的重要组成部分[12]，它是介于控制中心与车载系统的中间枢纽，有着举足轻重的地位，此系统的设计促进了整个交通系统运行效率的提高。其系统的场景示意图如图 2.1 所示。

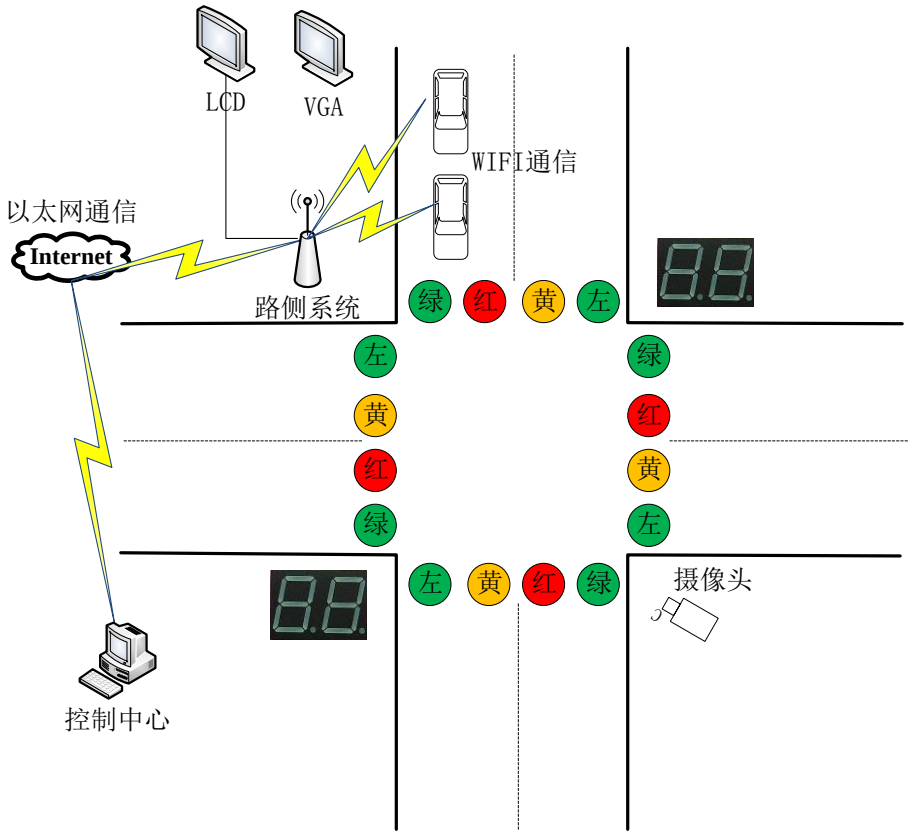



图 2.1　智能路侧系统场景示意图

图中，路侧系统通过与车辆的 wifi 通信，获取车辆的信息，再将车辆的信息存储起来经过以太网通信发送给控制中心，控制中心获取到车辆信息后进行信息的汇总处理，然后发送控制指令给路侧系统，路侧系统接收指令，控制交通灯、数码管和 LCD 液晶的工作状态。同时摄像头采集十字路口的图像信息，将其显示在 VGA 屏幕上，供驾驶员参考。这也是本文系统设计的大体流程。

本文中，将智能路侧系统的整体设计划分成了三个部分，如图 2.2 所示，三个部分协调工作，缺一不可。下面将着重讨论这三个部分的设计流程。

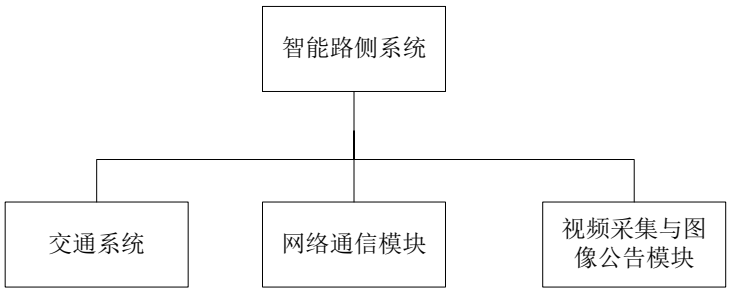



图 2.2　智能路侧系统的设计划分

### 2.1.1　智能路侧交通系统的设计流程

此部分的设计，主要是对交通信号灯和 LCD 路侧引导标识进行控制，其两者的控制流程都可以表示为如图 2.3 所示。

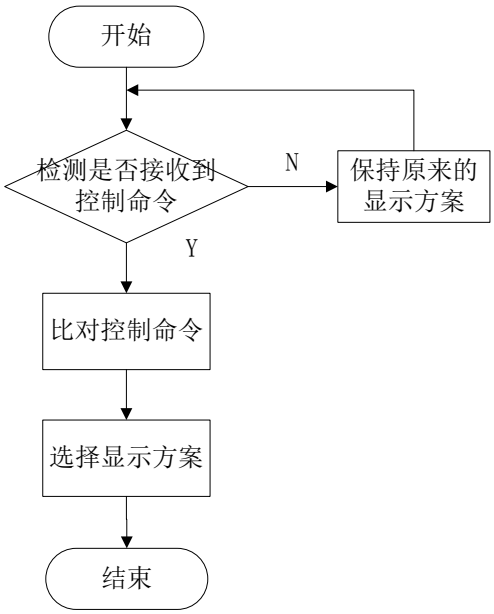



图 2.3　交通信号灯和 LCD 的控制流程

其中，交通信号灯的设计是使用纯 Verilog HDL 硬件语言编写，LCD 的显示是先在软核构建的时候加入 LCD 的端口，然后控制实现是在 Nios II IDE 下通过 C 语言编写实现。

### 2.1.2　智能路侧系统网络通信模块的设计流程

网络通信模块的设计需要分为路侧系统与车载设备的通信以及路侧系统与控制中心的通信，前者需要接入互联网进行远程通信，后者只需使用一个 wifi 无线路由器建立一个 wifi 热点，接入局域网即可，两者使用的传输协议都是 TCP/IP 协议。此部分的设计是先在软核上添加 SPI 总线控制接口，然后在 Nios II IDE 下通过 C 语言编写通信传输程序，两者的设计流程如图 2.4 所示。

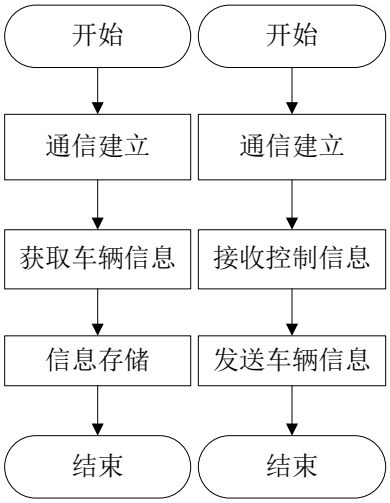



图 2.4　路侧与车载和控制中心的通信流程

左侧是路侧系统与车载系统的通信，右侧是路侧系统与控制中心的通信。车载系统我们使用了手持设备下载了 TCP 调试软件进行连接通信，控制中心我们使用 Qt 界面编程技术编程实现了控制中心的控制界面，使得信息交互变得清楚、直观。

### 2.1.3　视频采集与图像信息公告模块的设计流程

此部分的设计关键在于摄像头的采集、图像信息的存储和图像信息的显示三部分的实现，其中涉及到 I2C 总线技术、SDRAM 的乒乓操作以及 VGA 的显示，这些都会在下面的章节介绍。此模块的设计，是使用纯 Verilog HDL 硬件语言设计，流程图如图 2.5 所示。

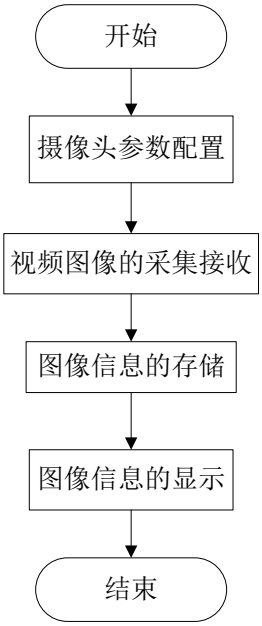



图 2.5　视频采集与图像公告模块的设计流程

## 2.2 系统环境搭建

### 2.2.1 主控芯片的选择

由于技术的发展，对芯片要求的提高，使得 FPGA 不断更新换代。如今的 FPGA 都可在一块芯片上进行软件和硬件的配合设计。随着其灵活性和集成度的进一步提高，它已经逐渐发展成为一个软件与硬件协同开发和自由定制的结合体，方便了用户同时使用软硬件开发[13]。基于以上原因，所以主控芯片我们选择 FPGA。现如今，世界上最大的 FPGA 供应商主要是 Altera 公司和 Xilinx 公司，考虑到成本问题，我们选择的 FPGA 芯片类型是 Altera 公司的 Cyclone IV 代芯片。此类芯片主要用于大批量的、需要考虑成本的应用中，使系统设计师在降低成本的同时，还能满足不断增长的带宽要求。几乎所有 Cyclone IV FPGA 供电只需要两路电源，简化了电源的网络，降低了 PCB 设计成本，减小了 PCB 的总体面积，缩减了设计周期。其含有两种型号——集成了 3.125 Gbps 收发器的 Cyclone IV GX FPGA 以及带有 1.0V 选择的 Cyclone IV E FPGA，本设计选用了 E 系列，表 2.1 是 Cyclone IV E 系列的部分相关芯片的参数特性。

表 2.1 Cyclone IV E 系列部分芯片的参数特性表

| 器件 | EP4CE6 | EP4CE10 | EP4CE15 | EP4CE22 | EP4CE30 | EP4CE40 |
|---|---|---|---|---|---|---|
| 逻辑单元 | 6,272 | 10,320 | 15,408 | 22,320 | 28,848 | 39,600 |
| M9K 存储器模块 | 30 | 46 | 56 | 66 | 66 | 126 |
| 存储器总容量(Kbits) | 270 | 414 | 504 | 594 | 594 | 1,134 |
| 18×18 乘法器 | 15 | 23 | 56 | 66 | 66 | 116 |
| PLL | 2 | 2 | 4 | 4 | 4 | 4 |
| 最大用户 I/Os | 179 | 179 | 343 | 153 | 532 | 532 |
| 最大差分通道 | 66 | 66 | 137 | 52 | 224 | 224 |

根据上表，选择 EP4CE15 足够完成本课题的设计，最终我们选择了 Altera 公司的 EP4CE15F17C8 芯片作为我们设计的主控芯片。由于采用的是 Altera 公司的产品，所以使用的硬件模块设计工具是 Altera 公司推出的适合 FPGA 开发的综合软件开发环境 Quartus II，我们通过此工具使用 Verilog HDL 硬件描述语言可以设计一个功能完善的硬件电路，同时可以使用此软件上的 SOPC Builder 构建一个 SOPC 系统，需要使用到 Altera 公司提供的另一款软件调试开发工具 Nios II IDE 对构建的系统进行软件编程实现功能。

### 2.2.2　外围电路的连接

系统的设计除了需要一块主控芯片以外，还需要其他的外围电路，其他的外围电路与 FPGA 的硬件连接图如图 2.7 所示。

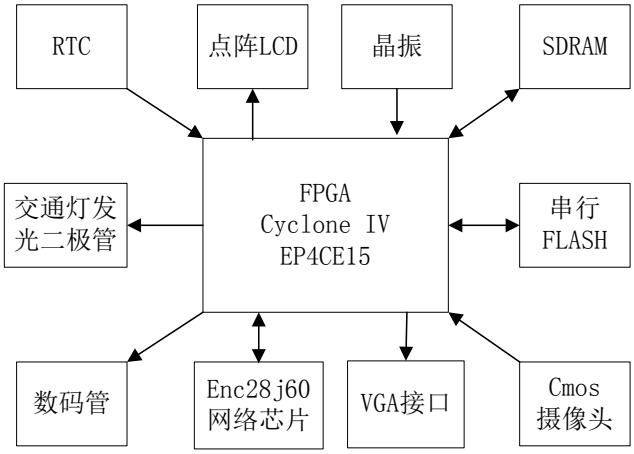



图 2.7　系统各部分外设连接图

其中各部分外设的型号和参数特性如表 2.2 所示：

表 2.2　系统主要外设型号参数表

| 外设 | 型号 | 参数 |
|---|---|---|
| SDRAM | H57V2562GTR | 2 块，内存大小：256Mbit = 16M * 16bit |
| 网络芯片 | ENC28J60 | 10M 网络芯片，可进行网络数据通信 |
| CMOS 摄像头 | OV7670 | 30 万像素 8 位数据接口的数字摄像头 |
| 实时时钟 | DS1302 | 支持年月日星期的实时时钟 |
| 点阵液晶屏 | ST7565P | 支持 128*64 分辨率的 COG 点阵液晶 |
| 串行 FLASH | M25P64 | 64Mbit |
| 晶振 | 有源晶振 | 50Mhz |
| 数码管 | | 6 位高亮数码管，动态扫描模式 |
| 交通灯发光二极管 | | 4 个红黄发光二极管，8 个绿色发光二极管 |
| VGA 接口 | | 支持 64K 种颜色显示 |

## 2.3　Quartus II 中相关组件的添加和配置

在系统设计当中，我们需要在 Quartus II 软件中添加一些组件，目的是为了实现软硬件协同开发，这些组件既有我们自己设计的模块，比如交通信号灯的控制模块、视频采集与图像信息公告模块，这两个都是使用 Verilog HDL 硬件语言实现的，我们需要将其例化成图元模块的形式，配合其他组件完成整个系统的设计；还有 Quartus II 自带的

一些宏模块，比如 PLL 锁相环等；还有使用 SOPC Builder 构建的软核处理器及其外设组件，以下对每个组件的添加和配置做一下简单的介绍。

### 2.3.1 自定义模块的组件添加

自定义模块的组件就是我们使用 Verilog HDL 硬件语言设计的模块，交通信号灯的控制模块和视频采集与图像信息公告模块。添加方式是选中我们所设计模块的顶层模块，单击 File→Creat/Update→Symbol Files For Current File 命令，生成了.bsf 格式的图元文件，然后添加到整个工程的顶层原理图上，其中交通信号灯模块生成的图元文件如图 2.8 所示，视频采集与图像信息公告模块生成的图元模块如图 2.9 所示[15]。

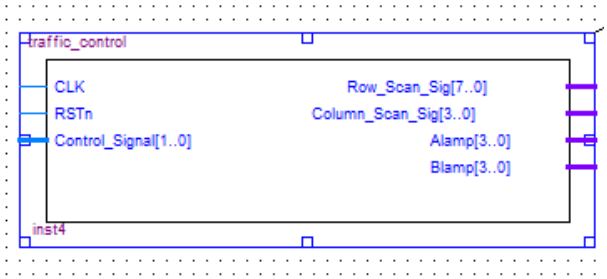



图 2.8 交通信号灯模块生成的图元模块

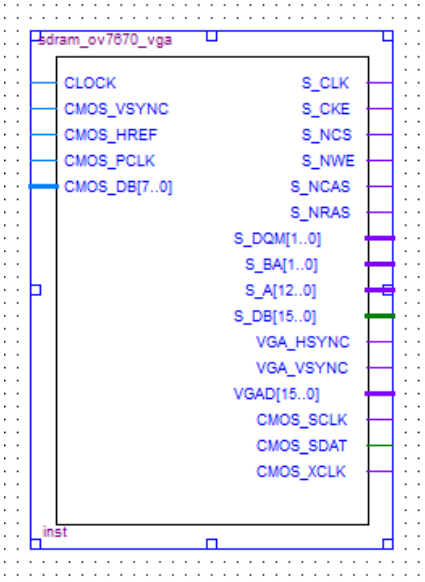



图 2.9 视频采集与图像信息公告模块生成的图元模块

### 2.3.2 Nios 软核及其相关外设的添加

软核的构建需要使用 Quartus II 软件中 SOPC Builder 工具，其中 Nios II Processor、System ID、SDRAM、EPCS、JTAG UART 是构建一个软核的几个基本外设，此处不再赘述[16]。在此，我们只介绍本课题所需其他几个关键外设的添加。

首先是网口 SPI 总线的添加，其配置如图 2.10 所示。这里，我们有 5 个地方需要注

意，方框 1 处是选择主模式还是从模式，我们 FPGA 上选择的是主模式（Master）；方框 2 处是从设备的个数，我们选择 1；方框 3 处是 SPI 时钟速率，我们填入 10M，方框 4 处是数据的位数，我们选择 8；方框 5 处是移位的方向，我们选择 MSB first。

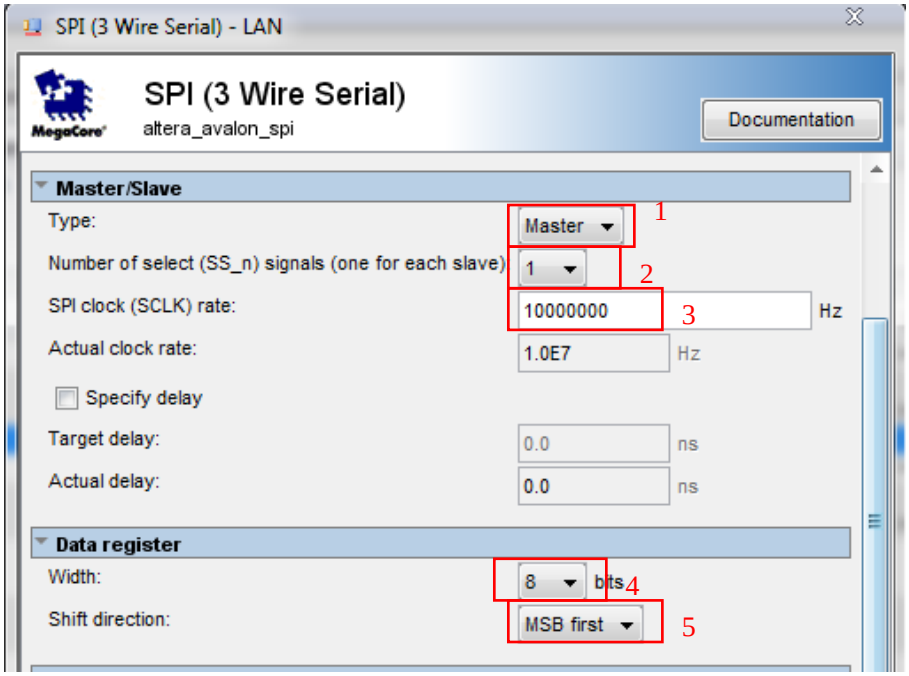



图 2.10　SPI 总线配置

其次是 LCD 组件的添加，针对 LCD 显示我们仅需配置 4 个 PIO 端口分别对应 LCD4 个引脚 LCD_CS，LCD_A0，LCD_SCL 和 LCD_SI，并将它们设为输出端口即可，其配置以 LCD_CS 为例，其他三个均与其相同，如图 2.11 所示。

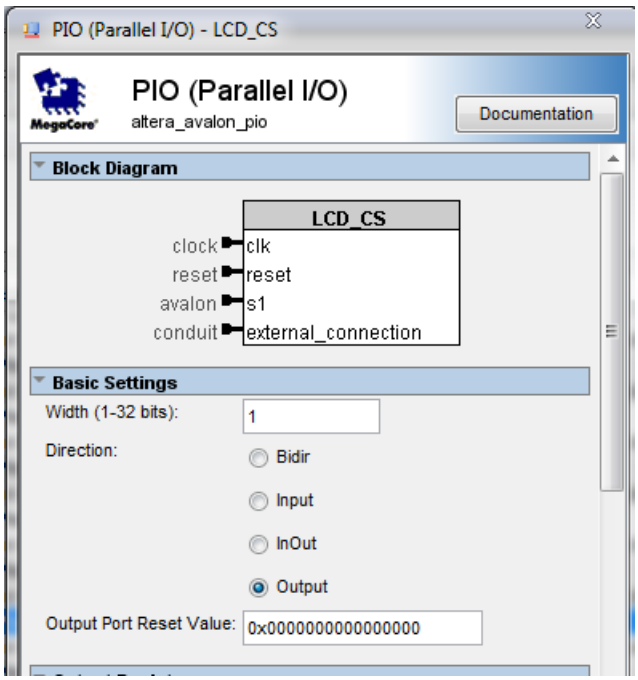



图 2.11　LCD 配置

下图是所构建软核的处理器和部分外设列表：

| Use | Conn... | Name | Description | Clock | Base | End | IRQ | Tags |
|---|---|---|---|---|---|---|---|---|
| ☑ | | cpu | Nios II Processor | [clk] | | | | |
| | | instruction_master | Avalon Memory Mapped Master | clk | | | | |
| | | data_master | Avalon Memory Mapped Master | [clk] | | | IRQ 0 | IRQ 31 |
| | | jtag_debug_module | Avalon Memory Mapped Slave | [clk] | 0x00001000 | 0x000017ff | | |
| ☑ | | sysid | System ID Peripheral | [clk] | | | | |
| | | control_slave | Avalon Memory Mapped Slave | clk | 0x000018c0 | 0x000018c7 | | |
| ☑ | | sdram | SDRAM Controller | [clk] | | | | |
| | | s1 | Avalon Memory Mapped Slave | clk | 0x04000000 | 0x05ffffff | | |
| ☑ | | epcs | EPCS Serial Flash Controller | [clk] | | | | |
| | | epcs_control_port | Avalon Memory Mapped Slave | clk | 0x00000000 | 0x000007ff | | 0 |
| ☑ | | jtag_uart | JTAG UART | [clk] | | | | |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | clk | 0x000018c8 | 0x000018cf | | 1 |
| ☑ | | LAN | SPI (3 Wire Serial) | [clk] | | | | |
| | | spi_control_port | Avalon Memory Mapped Slave | clk | 0x00001800 | 0x0000181f | | 2 |
| ☑ | | LAN_nINT | PIO (Parallel I/O) | [clk] | | | | |
| | | s1 | Avalon Memory Mapped Slave | clk | 0x00001820 | 0x0000182f | | 3 |
| ☑ | | LAN_CS | PIO (Parallel I/O) | [clk] | | | | |

图 2.12　软核处理器和相关部分外设

构建完的软核图元模块如图 2.13 所示。

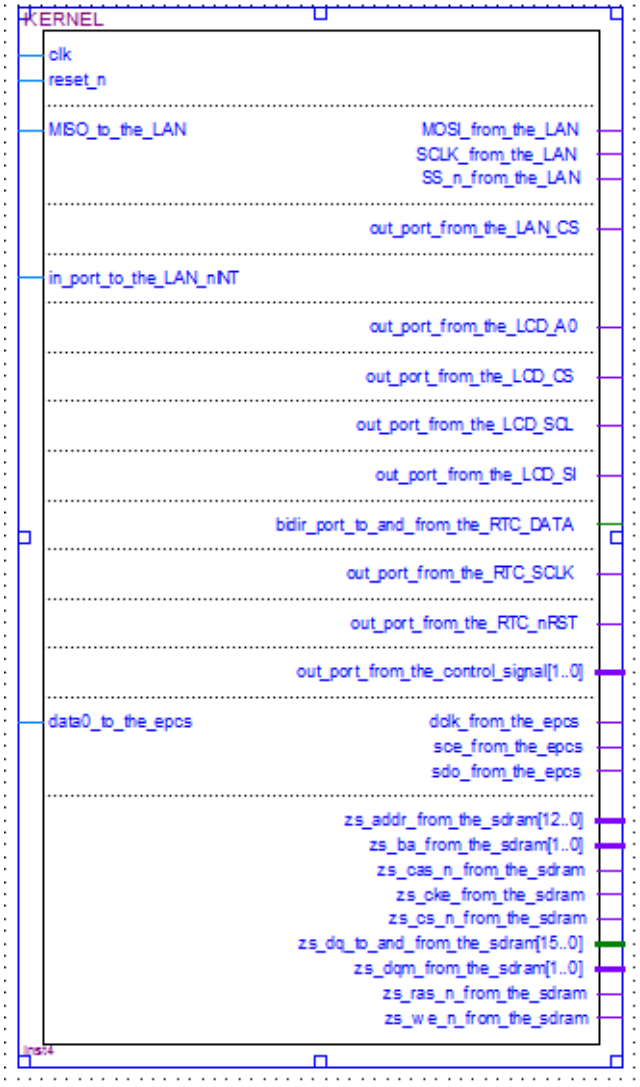



图 2.13　软核顶层图元模块

### 2.3.3　锁相环 PLL 的组件添加

FPGA 的设计离不开时钟，我们 FPGA 开发板上使用的是 50MHz 的有源晶振，但 CMOS 摄像头采集时钟和 VGA 工作时钟都是 25MHz，视频存储的 SDRAM 时钟是 100MHz，软核的时钟和其存储器 SDRAM 的时钟也是 100MHz，所以我们需要构建锁相环 PLL，满足不同的时钟需求。点击 Tools→MegaWizard Plug-In Manager，找到 ALTPLL 之后创建配置。其中，软核和其存储器 SDRAM 的时钟配置如图 2.14 和图 2.15 所示，由于电路板及芯片的信号延迟等原因，软核和 SDRAM 的时钟有相位差，如图中方框所示。生成的 PLL 图元模块如图 2.16 所示。

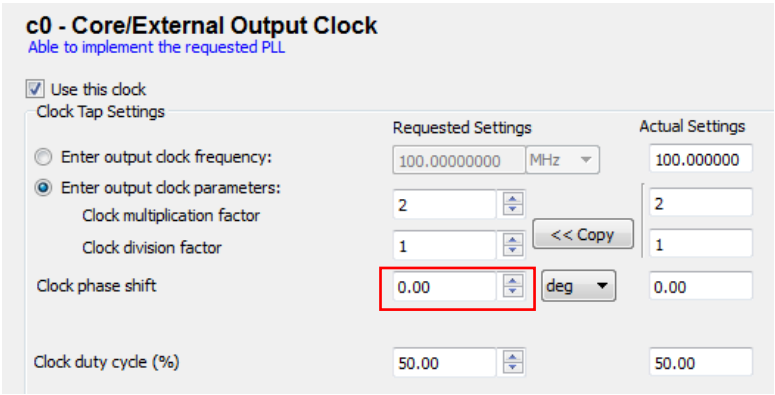



图 2.14　软核的时钟设置

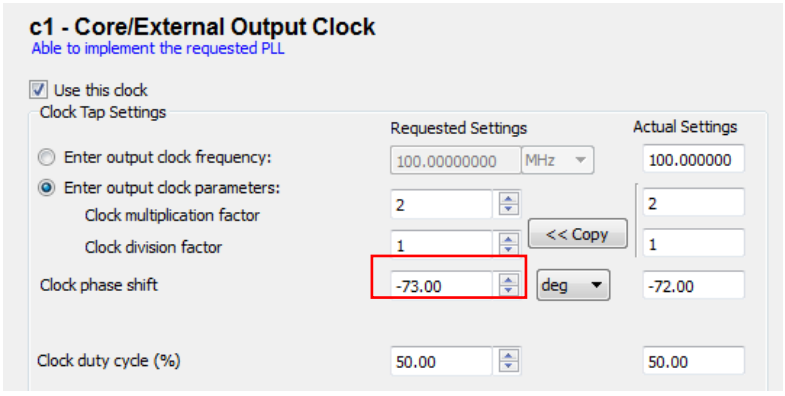



图 2.15　SDRAM 的时钟设置

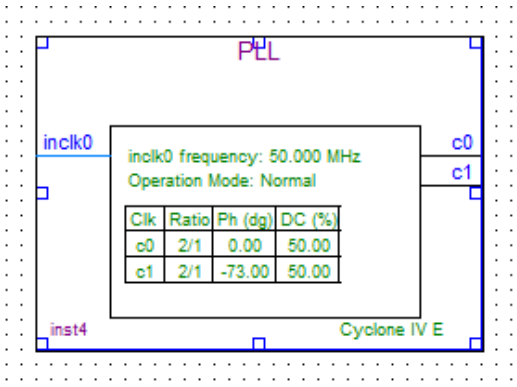



图 2.16　锁相环 PLL 图元模块

由于 CMOS 采集时钟、VGA 工作时钟和视频存储的 SDRAM 工作时钟的 PLL 配置是直接加入到了视频采集与图像信息公告模块中，不需要生成图形化模块，且与以上配置类似，所以在此不再赘述。

## 2.4 本章小结

本章首先介绍了智能路侧系统的整体设计方案，之后分三个部分分别阐述了每个功能模块的设计流程，然后简要介绍了系统的主控芯片和相关外设的型号参数，最后介绍了 Quartus 中使用到的相关组件的添加和配置，为后续的模块设计奠定了基础。

# 第 3 章　智能路侧交通控制系统设计

实际的交通控制系统主要包括交通信号灯模块，路侧引导标识模块等，所以本章主要围绕这两个方面进行分析设计。由于资源有限，实际的交通信号灯无法得到，所以采用红黄绿三色发光二极管替代，实际运用到交通信号灯上还需考虑到信号放大等问题；而路侧引导标识多是能动态显示来自控制中心的控制信息，同时考虑到成本问题，采用 12864 点阵 LCD 屏来显示。本模块的设计还需要来自控制中心发送过来的控制信息的控制，这部分内容会在下一章介绍。

## 3.1　交通控制系统的分类

对于交通控制系统我们从空间关系和控制方式两个方面进行分类[18]。

1.按空间关系划分

交通控制系统有很多种划分方法如果从空间关系上划分可划分为单交叉口控制（点控）、交通干线的协调控制（线控）、区域交叉口的网络控制（面控）[19]。

（1）单个交叉口的点控制

单个交叉口的点控制作为控制方式中的一种是最基础的。假如两个交叉口的距离比较远，车辆都聚集在这个路口，我们可以使用某个交通信号控制器来控制交通信号灯根据一定的规律而变化从而引导交通状况的运行，这种情况下两个交叉口之间不存在相对应的协调关系。现今点控制仍然是使用最多的控制方式，因为其设备简单、投资小、方便维护。点控制在技术层面上又分两个形式：离线点控制，就是按固定的配时方案控制信号灯；还有在线点控制，根据实际情况，智能设置信号灯的运行情况，保证交通畅通。

单个交叉口的有两个比较重要的控制参数：信号周期和信号的绿信比，这两个参数分别代表车辆通过交叉口等待时间和单位时间内交叉口通过车辆的最大值。最大的利用交通口通行能力及使得总延误时间最小是我们追求的最高目标。

（2）干线交通的协调控制

交通干线是城市路网中的主干线在城市路网中有着不可代替的作用，如果能利用干线交通的协调控制技术来控制主干线，就能改善一个区域甚至一个城市的交通状况，因此干线交通的协调控制技术非常重要。

道路交通中，经常遇到这种情况，相邻的两个路口相距太近，导致车辆在较短的时间内不能够完全通过。如果用单个交叉口的点控制，车辆经过每一个交叉口时遇到红灯的次数较多，严重影响了交通运行效率，所以我们需要协调这两个路口的信号，使得车

辆通行达到最优。协调信号方法的设计是在绿波的概念上发展而来的，两个相距较近的路口使用同一个周期信号，不同的主干线开启绿灯的时刻要错开，即需要错开一个相位，使得车辆能一路顺畅通行。使用此方法减少了能源的损耗，提高了交通运输效率。

我们把周期、绿信比、相位差等作为干线交通协调控制的控制参数，把车辆的延误的平均值和停车次数作为控制的目标。

（3）区域交通网络的协调控制

对于某个范围内交通控制的综合称之为区域交通控制，比如全市、全省范围内的交通控制。随着智能计算机技术不断创新、道路系统控制方法的不断优化、车辆检测和识别等技术不断发展，人们把城市中各个交叉口信号综合起来用自动控制的方法使得车辆在每个路口等待时间最少从而提高了道路的交通运输能力。这种控制方式是上位机接收交通信号机通过通信网传输的交通量的实时数据，依据不同时刻道路的不同情况，按预设的时间不断调整区域交通网络的协调控制方案。一个城市区域中的不同交叉路口由一个上位计算机同时控制，实现统一协调管理，提高了交通网络的通行效率。采用这种方式控制道路交通，能优化交通路网的管理与交通路网的统一调度。

区域交通网络的协调控制控制参数与干线交通的控制完全一致，但是控制的目标却非常多，包括全网络交通的运行效率等等。本质上来说，干线协调控制是网络协调控制的组成部分，若干个干线协调控制整合起来就称为区域交通的协调控制。

点控、线控、面控的划分示意图如图 3.1 所示。

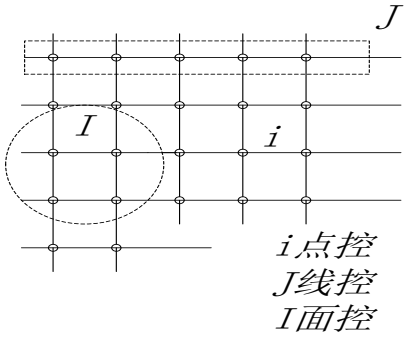



图 3.1　区域交通控制层次划分示意图

2.按控制方式划分

依据控制的不同方式道路交通控制可以按定时控制、感应控制、自适应控制以及智能控制进行简单的归类划分。

（1）定时控制

在原有的交通流数据基础上，定时控制推算出一周的不同时间段的交通流走势情况，通过人工或者计算机仿真等方法把一周的不同时间段内的配时方案预先准备好，然

后把上述方案保存于信号控制单元或者控制中心以备用。在运行期间即可按照合适的方式对预先保存的方案进行调用，一般可以在要求的时间表监测下通过时钟定时调用相应的配时方案，也可以根据检测器所检测的交通实际情况调用最佳的控制策略。定时控制属于开环控制的一种方式，不易进行实时交通流条件下的方案调整，更不易达到一定程度上的最优化控制。但是因为定时控制受到的限制较少，不需要对交通状况各方面进行实时监测，所以它仍然是现在交通控制广泛使用的一种方案。

（2）感应控制

感应控制是一种为了配合交通流的动态变化，依据检测到的交通流的数据来改变对应的交通灯状况持续长短以及时间次序的控制方式。上述方式与定时方式相比控制更加灵活。感应控制适合应用于车流量差别明显的交叉路口，在车流量变化杂乱无章、有较强随机性的情况下效果尤其显著。该控制原来是应用于单交叉口的控制，经过后续完善，在干线交通和区域交通网络中也得到了相应应用。

（3）自适应控制

自适应控制通过测量到的相应交通道路状况，并且根据预测模型预判到的未来一段时间内的交通量状况，从存储的信号配时方案中选取或者实时推算产生最优化策略来满足车辆对道路交通实现自动化的要求。在该控制方式中，配时方案的选取完善了对实时交通的需求，因此可以依据交通流量的动态变化来实时修改控制数据，既能达到很好的精确性，又能满足系统对实时性的要求。自适应控制是很有发展前景的道路交通控制策略，但是因为在实施过程中需要安装大量的传感器检测设备、铺设大量的通讯线路而抬高了系统的整体成本。

（4）智能控制

严格来说，智能控制的控制对象不但有交通信号还包括整个交通系统。它是将所有技术包括通信技术等等整合起来，建立一种覆盖所有范围的交通综合管理系统。它比自适应控制更高效，更完整，考虑的问题更全面，是未来交通的发展方向。

本文设计的交通信号控制系统，从空间关系上分属于区域交叉口的网络控制的一部分，整个区域的统一协调管理是由上位机控制中心统一管理的，从控制方式上还是划分为定时控制。

## 3.2　交通控制系统硬件电路的设计

根据实际交通控制系统的组成，本章的硬件电路设计可以分为两个部分，一个是交通信号灯，另一个是路侧引导标识。以下根据这两个部分分别阐述。

1. 交通信号灯

实际交通信号灯模块包括交通红绿灯和倒计时数码管两部分，其中交通红绿灯采用的是三色发光二极管模拟，倒计时数码管是共阳八段数码管，多位显示采取的是动态扫描方法。

交通红绿灯需要 8 个绿色发光二极管，4 个红色发光二极管和 4 个黄色发光二极管，一共 16 个发光二极管，每个二极管都需要串连一个 470 欧姆的限流电阻。发光二极管的正极与其相对的发光二极管的正极同时接入一个共同的用户自定义 I/O 口，发光二极管的负极经过限流电阻接到 FPGA 开发板的“地”端。其整体硬件电路结构如图 3.2 所示。

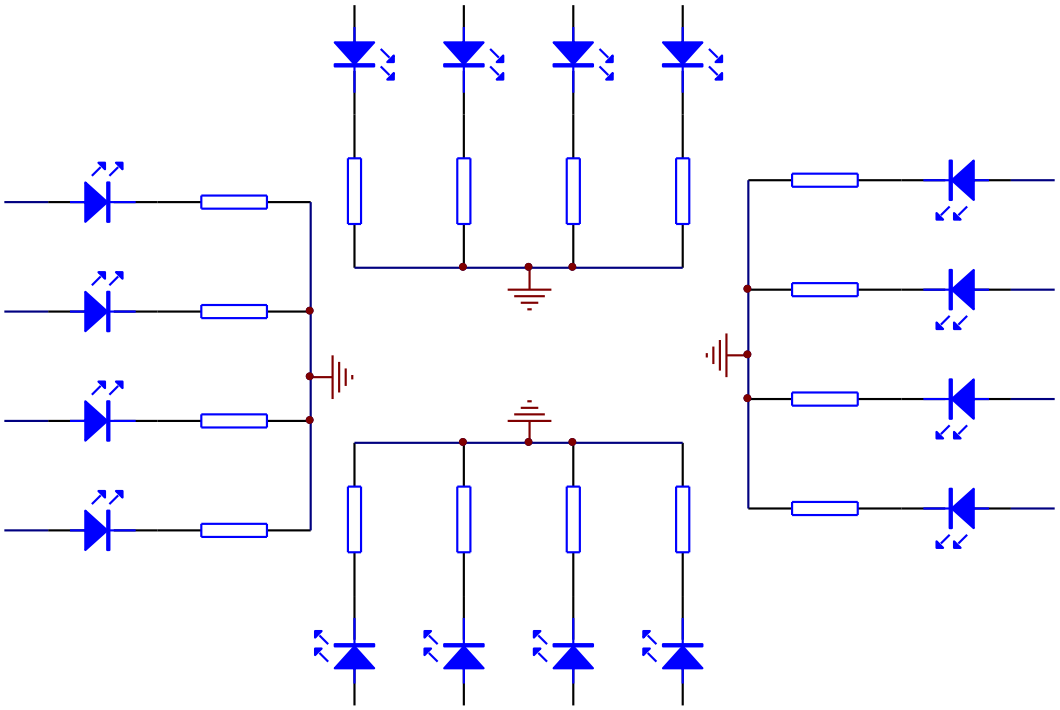


图 3.2　交通红绿灯电路原理图

倒计时数码管采用的是共阳极的，因此，数码中的每一段与 FPGA 的接口 DIG[0..7]都是低电平亮，高电平灭。例如“11000000”表示的就是 0 的数码，SEL[0..5]相当于每一段数码管的片选信号，本系统中相对路口的交通灯变化完全一致，所以可以共用一组数码管，所以一共只需要 2 对 4 个数码管，我们选择 SEL[0]，SEL[1]片选的数码管作为一对路口的倒计时显示，选择 SEL[4]，SEL[5]片选的数码管作为另一对路口的倒计时显示，电路原理图如图 3.3 所示。

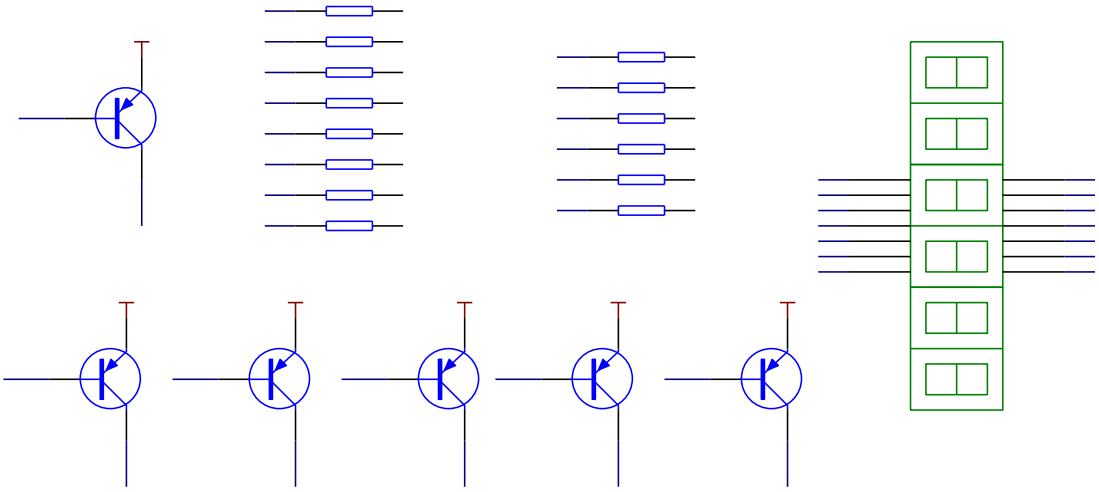


图 3.3　数码管电路原理图

2.路侧引导标识

路侧引导标识，我们使用的是点阵 LCD 液晶屏。通常所见到的 LCD 模块由三个部分组成，分别是 LCM（玻璃）、背光、PCB 板。其中 LCM 是不可或缺的。按照驱动控制器的集成方式划分，点阵的 LCD 模块可以分为两种：COB 和 COG，COB 是需要将驱动芯片焊接在 LCD 后面的 PCB 板上的，需要占用的空间较大，而 COG 是将芯片全部集成到了 LCM 里面，我们只需要在 PCB 板上加上一些电阻电容，所以使用 COG 集成方式就只需要一块 LCM 就能显示了。

本设计所使用的点阵 LCD 是 128*64 的 COG 液晶，所以省去了 PCB 底板，给我们节省了很大的空间。以下是 128*64 的 COG 液晶的部分参数，如表 3.1 所示，它的驱动芯片型号是 ST7565P，能够设置三种接口方式，我们选择的是串行时序方式，接口非常简单，而且也节省了很多管脚。此款 LCD 液晶屏的电路原理图如图 3.4 所示。

表 3.1　液晶 ST7565P 物理特性参数表

| 项目 | 内容 | 单位 |
|---|---|---|
| LCD 装配方式 | COG | |
| LCD 显示方式 | 反射式、全透式和半反射式 | |
| LCD 类型 | STN：黄绿、灰模、蓝模<br>FSTN | |
| 视角 | 6 点或 12 点 | |
| LCD 模块尺寸 | 58.0（宽）×39.0（高）×2.8（厚，最大值） | mm |
| LCD 视区尺寸 | 54.0（宽）×31.0（高） | mm |
| LCD 点阵方式 | 128×64 点阵 | |

| | | |
|---|---|---|
| 点尺寸 | 0.34（宽）×0.38（高） | mm |
| 点间距 | 0.37（宽）×0.41（高） | mm |
| LCD duty | 1/64 | |
| LCD 偏压 | 1/9 | |
| LCD 控制器 | ST7565P(COG) or EQUIVELENT | |
| LCM 工作温度（常温） | 0~+50 | ℃ |
| LCM 存储温度（常温） | -10~+60 | ℃ |
| LCM 工作温度（宽温） | -20~+70 | ℃ |
| LCM 存储温度（宽温） | -30~+80 | ℃ |
| 可选背光方式 | LED | |
| 自由设置接口方式 | 8080 时序方式<br>6800 时序方式<br>串行时序方式 | |
| 供电电源 | 2.8V-5.5V<br>内有 DC/DC 电路，对比度可以通过软件设置 | V |
| 预期寿命 | 50,000 | Hours |

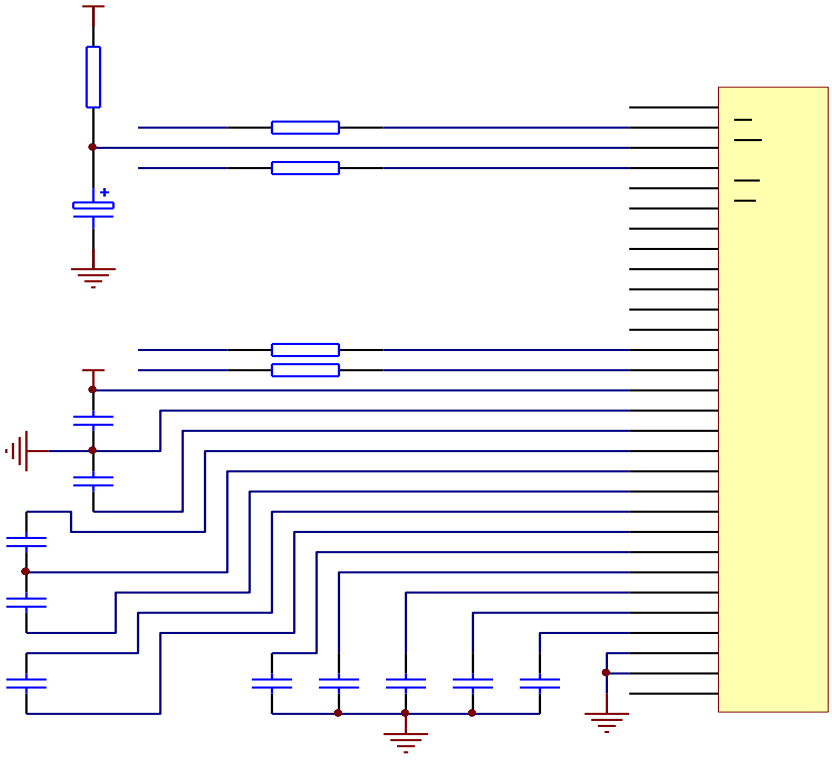


图 3.4　ST7565P 液晶电路原理图

## 3.3　交通控制系统软件设计

### 3.3.1　交通信号灯模块

整个交通信号灯的设计可以分为三大部分实现，包括分频计数、配时和显示，这三个部分都是使用 Verilog HDL 硬件语言设计实现的，三个部分协同工作缺一不可，其系统结构图如图 3.5 所示[20]。

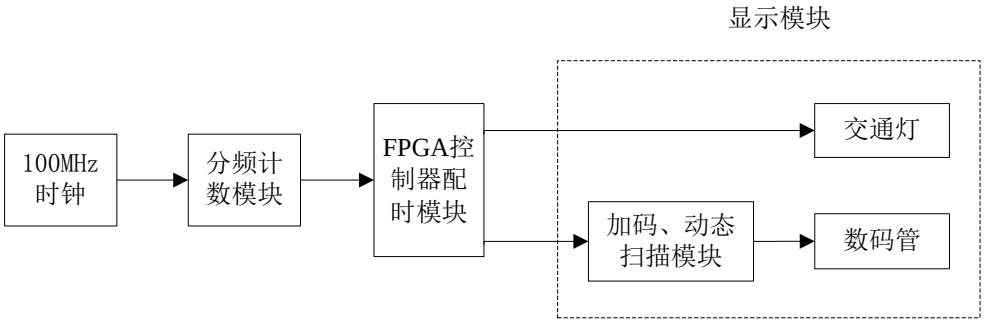



图 3.5　交通信号灯设计结构图

#### 3.3.1.1　分频计数模块设计

本设计采用的 FPGA 开发板使用 50MHz 频率的时钟，经过锁相环后与构建的软核时钟频率一致为 100MHz，本模块交通信号灯运转需要 FPGA 进行 1 秒钟为单位的计时，因此需要对输入源时钟进行分频。在当前的分频解决方案中，有许多芯片生产公司直接在 FPGA 芯片中集成锁相环，便于设计者快速得到自己需要的时钟频率，例如：ALTERA 系列的产品内置 PLL 对时钟进行分频以及倍频等操作。然而为了节省 FPGA 开发板的逻辑资源，本设计直接采用 Verilog HDL 硬件描述语言来设计分频模块。下面有必要先介绍一下硬件描述语言针对各分频系数分频常采用的方法。

第一，偶数倍分频：这种情况下，可以直接通过 Verilog HDL 程序计数来做到，例如，当需要偶数 N 倍分频时，可以利用源时钟（待分频）驱动计数器跳动，待计数器由 0 跳动到 N/2-1 时刻，此时将输出时钟的电平值取反，同时将计数器归零，当下一周期来临时，计数器重新由 0 跳动，以此循环下去，最终实现所需的偶数倍分频。

第二，奇数倍分频：这种情况下，也可以利用 Verilog HDL 程序计数来做到，例如需要对源时钟 3 倍分频，可以利用源时钟（待分频）驱动计数器跳动来进行 3 进制计数，这时需要待计数器跳动到某一邻近值时让输出时钟的电平值连续翻转 2 次，具体中，我们可以待计数器跳动到 1 时，输出时钟电平值取反，计数器跳动到 2 时，将输出时钟的电平再次取反，亦即：在计数器跳动到 1 和 2 时，输出时钟的电平值翻转了 2 次，由此实现了占空比为三分之一的 3 倍分频。当然，需要得到 50%占空比的 3 倍分频，这时就

要先使用源时钟（待分频）下降沿时刻来驱动计数器跳动，同时再用待分频源时钟上升沿时刻驱动计数器跳动进行 3 倍分频，最后将待分频源时钟上升沿驱动得到的 3 倍分频时钟和下降沿驱动得到的 3 倍分频时钟或运算处理，就能获取 50%占空比的 3 倍分频。利用这一思想，我们得到了占空比 50%的奇数 N 倍分频的一般方法：先使用源时钟（待分频）上升沿时刻来驱动计数器跳动进行 N 进制计数，待计数器跳动到某一个设定值 m（m<（N-1）/2）时将输出时钟的电平值取反，计数器再跳动（N-1）/2 次后，将输出时钟再次取反，此时实现了占空比不等于 50%的奇数 N 倍分频，再同时使用源时钟（待分频）下降沿时刻来驱动计数器跳动进行 N 进制计数，待计数器跳动到 m 值时将输出时钟的电平值取反，计数器再跳动（N-1）/2 次后，将输出时钟再次取反，此时又实现了一个占空比不等于 50%的奇数 N 倍分频，最后将上述两个奇数 N 倍分频输出时钟或运算处理，就能获取 50%占空比的 N 倍奇数分频。

本文是针对 100MHz 分频到 1Hz，分频系数是 100000000，为偶分频，无需考虑奇数分频，只需将 100MHz 的源输入时钟驱动计数器跳动进行计数，使计数器从 0 开始跳变到 49999999 状态时，将输出时钟翻转，并将计数器复位置零，之后计数器重新由 0 跳动，计数到 49999999，再次将输出时钟翻转，此为一个周期。以此循环下去，最终实现所需的 1Hz 时钟。其原理示意图如图 3.6 所示。

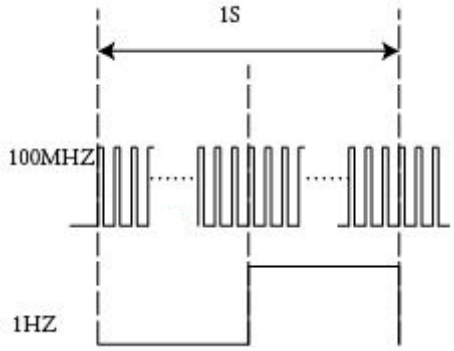



图 3.6　分频原理示意图

### 3.3.1.2　配时模块设计

配时模块设计是整个交通信号灯设计的重点和核心，是整个交通信号灯系统能正常、有效运行的保证。目前，在我国的城市交通体系中，城市的交通路口主要有 T 字路口、十字路口以及多路口交叉这几类，其中以十字路口最为普遍，所以本文以最基本的十字路口为研究对象，十字路口的交通分为主干道和副干道两个方向，其中每个方向有 3 种通行状态，分别是左行、直行和右行，由于右行和直行的通行状态可以合并，所以

可以不用考虑右行状态[48]。根据以上分析，主干道方向和副干道方向只需 4 盏灯即可保证交通系统的正常运行，它们分别是红灯(R)、黄灯(Y)、直行绿灯(G)和左转绿灯(L)。十字路口两个方向的灯必须按照一定的顺序亮灭，根据实际情况，此时交通信号灯的运行情况可以设置如下[20]：

主干道方向的交通信号灯运行流程为：绿灯→黄灯→左转绿灯→黄灯→红灯。

副干道方向的交通信号灯运行流程为：红灯→绿灯→黄灯→左转绿灯→黄灯。

在实际交通中，两个方向交通信号灯的状态有着密切的联系，一个方向的交通信号灯状态影响着另一个方向的交通信号灯状态，只有这样才能协调好两个方向的车流，如果两个方向的交通信号灯都各自独立变化，交通系统就会混乱，造成严重的后果，交通信号灯管理交通和协调交通的作用也就没有意义。

其一个周期的交通状态对应情况如表 3.2 所示，其中 1 表示灯亮，0 表示灯灭。在我们编程实现时，可以将主干道和副干道的状态分开进行，所以针对主干道 S4~S7 状态可以合并，针对副干道 S0~S3 状态可以合并。

表 3.2　交通状态对应表

| 状态 | 主干道方向 | | | | 副干道方向 | | | |
|---|---|---|---|---|---|---|---|---|
| | G | Y | L | R | G | Y | L | R |
| S0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| S1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| S2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| S3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| S4 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| S5 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| S6 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| S7 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

实际交通中，白天和夜间的车流量相差很大，同时白天期间的主副干道车流量也会出现不均衡的现象，这就要求准备多套配时方案，以适应不同的需求。针对白天正常情况下，主干道车流量大于副干道车流量时，配时时间设置如表 3.3 所示，视为方案一；针对白天主干道车流量小于副干道车流量时，配时时间如表 3.4 所示，视为方案二；针对夜间车流量较小的情况下，配时时间可以如表 3.5 所示设置，视为方案三；当发生交通事故或者有救护车、消防车等应急车辆需要优先通行时，所有路口都将红灯置亮，保

证应急车辆的通行。这 3 种配时方案的选择并非由智能路侧系统本身来控制，而是由控制中心综合各种信息之后，远程控制选择，选择完毕之后也并非立刻按照配时方案改变交通红绿灯的运行状态，而是等待一个周期结束之后即从状态 S0~S7 之后才开始改变配时方案，否则交通红绿灯显示时间突变会造成交通事故。

表 3.3　日间主干道车流量多于副干道各信号灯配时时间

| 时间(s) | Y | G | L | R |
|---|---|---|---|---|
| 主干道 | 5 | 40 | 15 | 55 |
| 副干道 | 5 | 30 | 15 | 65 |

表 3.4　日间主干道车流量少于副干道各信号灯配时时间

| 时间(s) | Y | G | L | R |
|---|---|---|---|---|
| 主干道 | 5 | 30 | 15 | 65 |
| 副干道 | 5 | 40 | 15 | 55 |

表 3.5　夜间各信号灯配时时间

| 时间(s) | Y | G | L | R |
|---|---|---|---|---|
| 主干道 | 3 | 20 | 10 | 36 |
| 副干道 | 3 | 20 | 10 | 36 |

本系统此模块以以上 3 种配时方案中的 1 种先按照状态 S0 对主干道绿灯和副干道红灯进行计数，计数采用减法计数器按照倒计时的方式当主干道绿灯计数到 1 时，然后进入状态 S1，此时副干道红灯继续递减计数，不受影响，而主干道绿灯熄灭，对黄灯进行计数，依次执行直到状态 S7，然后进入下一个周期的计数，往复循环。

#### 3.3.1.3　显示模块设计

本次设计的显示模块部分主要是交通红绿灯的显示和数码管的显示，其中交通红绿灯的显示比较简单，FPGA 以 4 位 Alamp 变量控制主干道上的四盏灯，其中 Alamp[0]控制左转绿灯，Alamp[1]控制黄灯，Alamp[2]控制红灯，Alamp[3]控制绿灯，其值为 1 则表示相应的灯亮，如 Alamp 为 4’b0001 表示左转绿灯亮；同时以 4 位 Blamp 变量控制副干道上的四盏灯，其设置与 Alamp 类似。每个灯的亮灭时间都是按照配时模块的部分设置的，此处就不赘述了。

数码管的显示相比于交通红绿灯的显示要复杂一点，因为需要加上加码、动态扫描等部分。整个交通信号灯系统实际需要 4 对共 8 个数码管，由于相对的路口倒计时完全

相同，所以为了节省逻辑资源，简化设计，可以让相对路口共用一对数码管，所以只需对 2 对共 4 个数码管的显示进行阐述。数码管是共阳的，而且 4 个数码管的段选信号都共用同样的引脚，全都是低电平有效。我们将主干道的倒计时时间设置成 Number_Data1，副干道的倒计时时间设置成 Number_Data2，数码管的显示设计思路如图 3.7 所示。

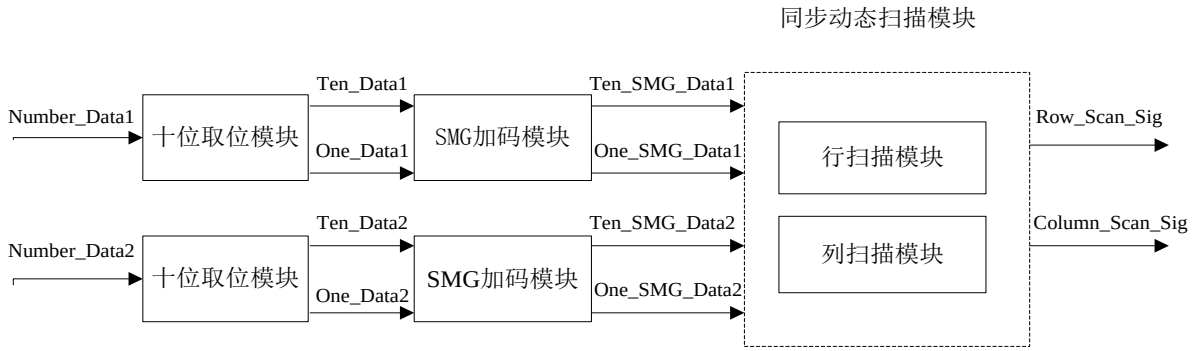



图 3.7　数码管显示设计模块图

由图中可以看出，通过“十位取位模块”分别对 Number_Data1 和 Number_Data2 进行取位的操作，将它们十位和个位划分，然后输出各自的信号线 Ten_Data 和 One_Data。之后通过“SMG 加码模块”，将每个路口的十位和个位数字转成 SMG 码，即数码管显示码。Ten_Data 和 One_Data 被转换后，再经 Ten_SMG_Data 和 One_SMG_Data 输出。最后由“同步动态扫描模块”驱动点亮数码管。现针对每一个模块进行分析。

1. 十位取位模块：就是利用数学运算符“%”和“/”分别取得主副干道计时时间的十位和个位，将十位和个位分开进行处理。

2. SMG 加码模块：就是分别将两个路口倒计时时间的十位与个位的数字转换成数码管可以显示的码型，每一个数字都是八段数码管 a,b,c,d,e,f,g 每段置亮置灭组合而成的，数码管显示示意图如图 3.8 所示。

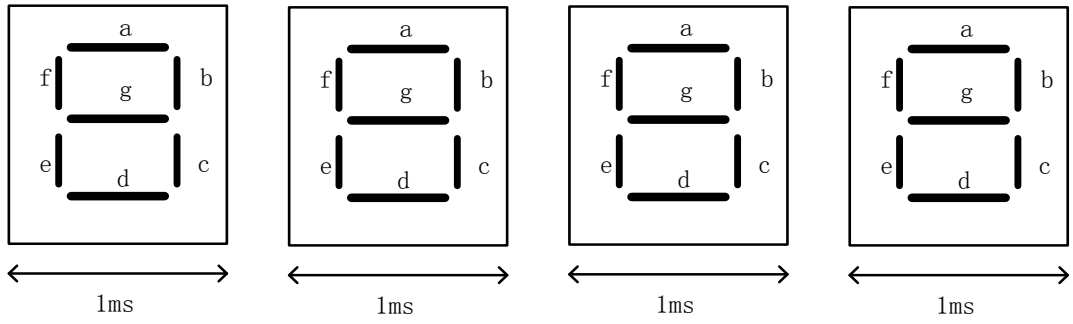



图 3.8　数码管显示示意图

3.同步动态扫描模块：此模块由两部分组成，分别是行扫描模块和列扫描模块，由于 FPGA 是可以并行操作的，所以这两个模块可以并行执行，达到真正的同步。同步动

态扫描模块利用的是时分原理和人类的视觉暂留效应，轮流驱动点亮 4 个数码管，由于每个数码管点亮时间为 1ms，人眼无法识别，所以看起来是 4 个数码管同时点亮。无论是列扫描模块还是行扫描模块扫描频率都需要设置成 1KHz，这样就对应了每个数码管的点亮时间 1ms，至于如何得到 1KHz 时钟信号，可以参考 3.3.1.1 节分频计数模块设计。同步动态扫描模块当中，列扫描模块相当于对数码管进行片选，每隔 1ms 就使能不同的数码管。而行扫描模块是输出不同的 SMG 码，根据列扫描对数码管的片选使能，输出当前数码管的 SMG 码。

### 3.3.2 路侧引导标识模块

路侧引导标识一般都架设在路边或者交通要道，是智能路侧系统不可或缺的一部分。它可以与交通信号灯有一定的距离，这样就能将前方交通路口的信息发布给每一个驾驶员，驾驶员根据引导标识提供的信息决定驾驶策略，是直行还是绕道。同时路侧引导标识还可以将控制中心发来的一些重要信息传达给每个驾驶员，提醒驾驶员一些注意事项等等。考虑到成本以及能动态显示信息的问题，所以本文使用了点阵液晶屏作为路侧引导标识，其具体实施方案流程如图 3.9。

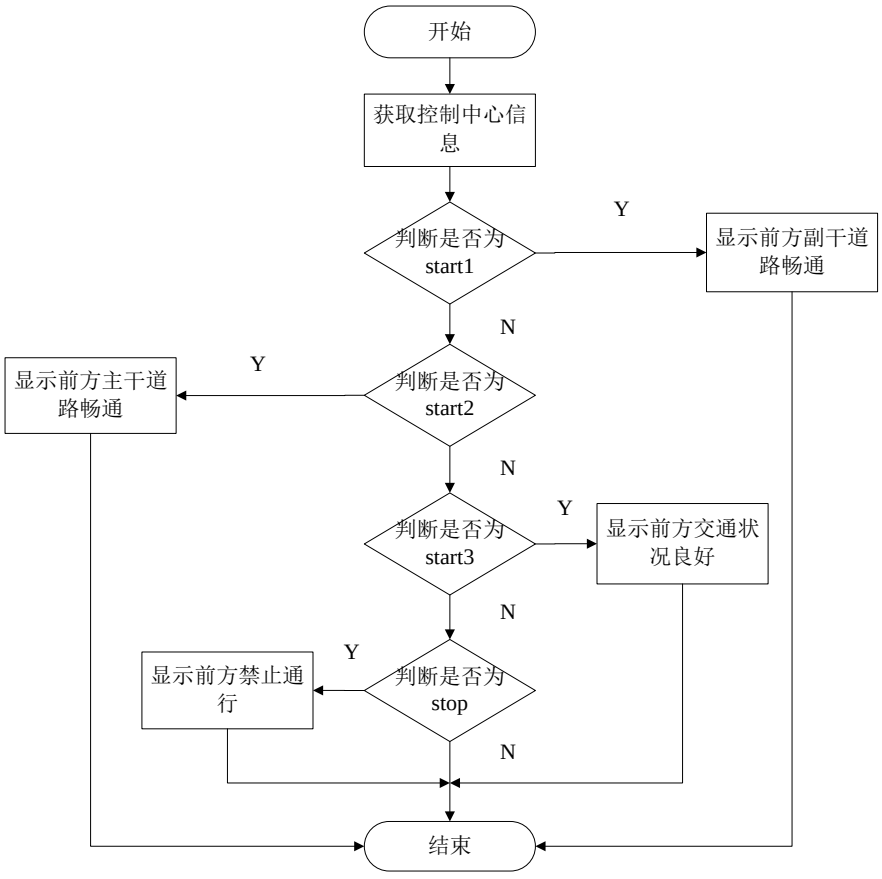



图 3.9 路侧引导标识设计流程图

图中 start1，start2，start3，stop 几个变量是我们自定义的控制变量，控制中心的信息发送过来之后要分别与这 4 个变量作比较，与其中一个变量相同，就显示与这个变量相应的显示信息，驾驶员会依据这些信息，选择行车路线。图中主干道、副干道通畅信息，在实际交通中会标明主干道、副干道的实际道路名称，这个需要确定路侧引导标识的实际设置位置，所以在此不作考虑，仅以主干道、副干道代替。LCD 屏的相关显示还加入了实时时钟，能够实时显示当前时间。以下是对 LCD 显示的相关技术进行阐述。

#### 3.3.2.1　LCD 液晶屏扫描实现

我们所使用的液晶屏是 64 高×128 宽的，它被分成 8 页 128 宽的 8 位数据。至于液晶的扫描次序就与 4 个命令有关系，分别是 0xA0，0xA1，0xC0，0xC8。当我们写入命令 0xA0，控制列填充是“从左向右”的，但如果命令是 0xA1，控制列填充是“从右向左”的。除了控制列扫描的命令，还有控制“页扫描顺序”的命令（每一页的高度是一个字节）。如果命令是 0xC0，页扫描就是“从下向上”的，但如果命令是 0xC8，页扫描就是“从上向下”的。其中，一页的高度是 8 位一个字节，每一页的列扫描，都是从低位到高位。在本设计中，列扫描次序是从左向右，页扫描次序是从上向下，程序中可设置如下：

```
write_command(0xa0);
write_command(0xc8);
```

假设填充的值都是 0x0f，那么列扫描 128 次以后，一页的扫描结果就是如下图 3.10 所示。


图 3.10　LCD 一页的扫描结果

由上图可知，整个 LCD 扫描分布就是 8 页*8 位*128 列，而事实上是不可能的，因为第 8 页和第 128~131 列是无法显示的，如图 3.11 所示，所以每完成列扫描 128 次之后，就要重新设置起始列地址和下一个页地址。

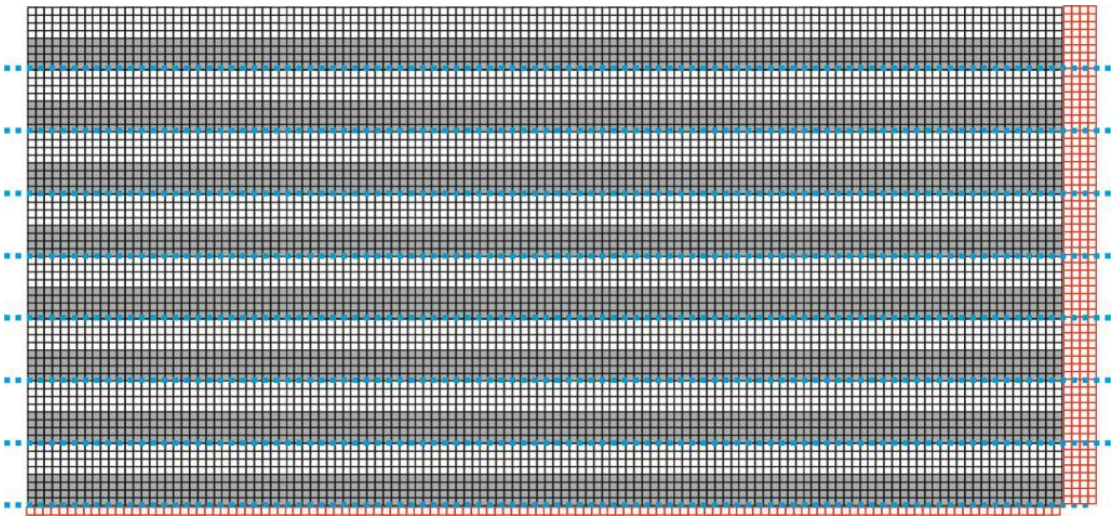


图 3.11　LCD 整体扫描结果

设置页地址的命令就是 0xb?，其中“?”就是页地址的设置值。假设输入 0xb0，也就是页地址 0。关于设置列地址的命令是 0x1?和 0x0?。命令 0x1?的“?”是列地址的“高四位”，而 0x0?的“?”是列地址的“低四位”。程序中，设置地址的方式如下，其中 set_x 函数是设置列地址，set_y 函数是设置页地址：

```
void set_x(unsigned char x)
{
    write_command(x>>4|0x10);
    write_command(x&0xf);
}
void set_y(unsigned char y)
{
    write_command(y|0xb0);
}
```

由于我们采用的传输方式是串行传输，所以需要考虑时序问题，针对本系统，FPGA 是主机，12864 液晶屏是从机，所以是从 FPGA 向液晶屏写数据，不存在双向传输的问题，在 SOPC 配置构建软核的时候一共仅需 4 根输出线即可，它们是 LCD_CS，LCD_A0，LCD_SCL 和 LCD_SI，后面简称 CS，A0，SCL，SI。A0 是用来判断读取的是命令还是数据，为 0 是数据，为 1 是命令。从机读取数据时，CS 片选信号会被拉低，并且发生在 SCL 信号的上升沿。SCL 信号是时钟信号，SI 是数据信号，SCL 信号在空闲的时候总是处于高电平。当主机开始向从机写入数据，主机会先拉低 CS 信号，再拉低 SCL 信号，然后“设置”数据，亦即主机更新 SI 的数据，最后再拉高 SCL 信号。同一时间，

从机会因为 SCL 的上升沿变化，从机“锁存”SI 上的数据，其时序图如图 3.12 所示。

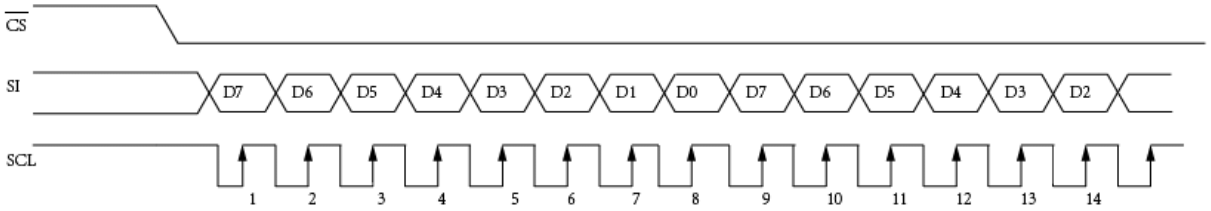



图 3.12　LCD 写入时序图

LCD 液晶屏扫描部分的实现是基于软核实现的，所以需要在 Nios II IDE 软件下实现以下一些比较重要的函数，现列举如下：

1. void data_send(unsigned char dat)

此函数是根据以上的时序分析，实现往 LCD 里面写入一个字节。

2. void write_command(unsigned char com)和 void write_data(unsigned char dat)

两个函数都调用了 data_send 函数，前者写入 8 位的命令，后者写入 8 位的数据。

3. void initialize_lcd(void)

此函数用来初始化 LCD 的相关参数，调用了 write_command 函数。

4. void set_x(unsigned char x)和 void set_y(unsigned char y)

这两个函数前面介绍过，用于设置列地址和页地址。

5. void clear(void)

此函数实现清屏，调用了 write_command 和 write_data 函数。

整个液晶屏的扫描流程如图 3.13 所示。

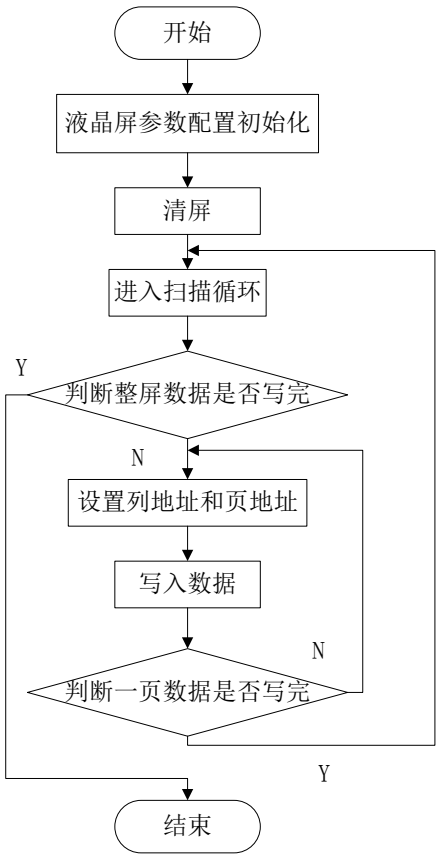



图 3.13　液晶屏的扫描流程

### 3.3.2.2　LCD 液晶屏字库添加

在 LCD 上进行字符的显示，无论中文还是英文，都离不开字库的支持。所以在进行字符显示时，字库的添加尤为重要。本课题选用的液晶屏是没有字库的，必须进行字库的添加。有些 LCD 模块，其液晶芯片中出厂时就已经烧写了字库文件，采用这种方式，其优点是使用方便，省去了添加字库的工作，但缺点是显示不够灵活，同时显示效果不能进行改变，并且价格也相对较高[21]。

1.英文字库

我们液晶屏上有数字显示，所以需要添加英文字库，我们添加的英文字库是 tahoma 字库，它是一个不定宽度的字库，其中每一个字符的偏移量如图 3.14 所示。

其字符宽度就是用后一个字符的偏移量减去自身的偏移量，然后再根据图 3.15 寻找这个字符的点阵码输出即可。

```
unsigned int  tahoma_font_offset[]={
          //    !   "   #   $   %   &   '    (
             0  ,3  ,7  ,11 ,19 ,25 ,36 ,43 ,45 ,49 ,
          // )   *   +   ,   -   .   /   0   1   2
             53 ,59 ,67 ,71 ,75 ,79 ,83 ,89 ,95 ,101,
          // 3   4   5   6   7   8   9   :   ;   <
             107,113,119,125,131,137,143,147,152,159,
          // =   >   ?   @   A   B   C   D   E   F
             168,175,180,190,197,203,210,217,223,229,
          // G   H   I   J   K   L   M   N   O   P
             236,243,247,252,258,263,271,278,286,292,
          // Q   R   S   T   U   V   W   X   Y   Z
             300,307,313,319,326,332,342,348,354,360,
          // [   \   ]   ^   _   `   a   b   c   d
             364,368,372,380,387,392,398,404,409,415,
          // e   f   g   h   i   j   k   l   m   n
             421,425,431,437,439,442,447,449,457,463,
          // o   p   q   r   s   t   u   v   w   x
             469,475,481,485,490,494,500,506,514,520,
          // y   z   {   |   }   ~
             526,531,536,540,545,553,};
```

图 3.14　tahoma 字库字符的偏移量

```
const char  font_tahoma_8[]={

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0B,0xF0,0x00,0x00,0x00,0x00,0x00,0x38,
0x00,0x00,0x00,0x38,0x00,0x00,0x02,0x00,0x0E,0x40,0x03,0xC0,0x0E,0x70,0x03,0xC0,
0x02,0x70,0x00,0x40,0x00,0x00,0x08,0xC0,0x09,0x20,0x3F,0xF8,0x09,0x20,0x06,0x20,
0x00,0x00,0x00,0x60,0x00,0x90,0x00,0x90,0x0C,0x60,0x03,0x00,0x00,0xC0,0x06,0x30,
0x09,0x00,0x09,0x00,0x06,0x00,0x00,0x00,0x07,0x60,0x08,0x90,0x08,0x90,0x09,0x60,
0x06,0x00,0x05,0x80,0x08,0x00,0x00,0x38,0x00,0x00,0x07,0xC0,0x18,0x30,0x20,0x08,
0x00,0x00,0x20,0x08,0x18,0x30,0x07,0xC0,0x00,0x00,0x02,0x80,0x01,0x00,0x07,0xC0,
0x01,0x00,0x02,0x80,0x00,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x0F,0xE0,0x01,0x00,
0x01,0x00,0x01,0x00,0x00,0x00,0x20,0x00,0x1C,0x00,0x00,0x00,0x00,0x00,0x01,0x00,
0x01,0x00,0x01,0x00,0x00,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x00,
0x07,0xC0,0x00,0x38,0x00,0x00,0x07,0xE0,0x08,0x10,0x08,0x10,0x08,0x10,0x07,0xE0,
0x00,0x00,0x00,0x00,0x08,0x20,0x0F,0xF0,0x08,0x00,0x00,0x00,0x00,0x00,0x0C,0x20,
0x0A,0x10,0x09,0x10,0x08,0x90,0x08,0x60,0x00,0x00,0x04,0x20,0x08,0x10,0x08,0x90,
0x08,0x90,0x07,0x60,0x00,0x00,0x01,0x80,0x01,0x40,0x01,0x20,0x0F,0xF0,0x01,0x00,
0x00,0x00,0x04,0xF0,0x08,0x90,0x08,0x90,0x08,0x90,0x07,0x10,0x00,0x00,0x07,0xC0,
0x08,0xA0,0x08,0x90,0x08,0x90,0x07,0x00,0x00,0x00,0x00,0x10,0x0C,0x10,0x03,0x10,
0x00,0xD0,0x00,0x30,0x00,0x00,0x07,0x60,0x08,0x90,0x08,0x90,0x08,0x90,0x07,0x60,
0x00,0x00,0x00,0xE0,0x09,0x10,0x09,0x10,0x05,0x10,0x03,0xE0,0x00,0x00,0x00,0x00,
0x0C,0xC0,0x00,0x00,0x00,0x00,0x20,0x00,0x1C,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,
```

图 3.14　tahoma 字库部分字符的点阵码

2.中文字库

本课题中需要进行中文字符的显示，中文字库的添加也就必不可少。国标中文字库一般将所有的国标汉字和符号分配在 94 行、94 列的方阵中，其中一行代表一个区,一列代表一个位，这就是区位码，每一个字符都有唯一的区位码与之对应。

为了避免与英文字库冲突，我们汉字字符区号和位号都加上了 A0H，这样就可以和英文字库区分开来，这就是机内码，但是再计算汉字字符的偏移量时，我们区号和位号都需要减去 A1H，多减了 1 是因为偏移量是从 0 开始计算的。汉字在汉字库中的具体位置可以使用以下计算公式：

94*（区号-0xA0-1）+（位号-0xA0-1）

我们使用的汉字字库是 HZK12，它是 12*12 的点阵字库，为了使用方便，我们将

其补成 16*16，多余的位用 0 补上，显示效果不受影响。其中一个汉字字模所占用的字节数是 32*32/8=128。如果想得到该汉字以字节为单位，在整个汉字库中的位置，只需在上面公式计算的结果乘以 128 即可，字库 HZK12 的部分点阵码如图 3.16 所示。

```
unsigned char achzk12[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x00, 0x60, 0x00,
  0x00, 0x00, 0x70, 0x00, 0x50, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x11, 0x00, 0x0A,
  0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x0A,
  0x00, 0x14, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x04, 0x00, 0x02, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
  0x00, 0x0A,
```

图 3.16　HZK12 字库的部分点阵码

## 3.4　本章小结

本章主要完成了智能路侧交通控制系统的设计部分，首先介绍了交通控制系统的一些分类，然后分别从硬件设计和软件设计两部分阐述了设计过程。硬件设计当中涉及到交通红绿灯的设计，数码管电路连接和 LCD12864 与 FPGA 的电路连接；软件设计当中涉及到分频计数、交通灯配时、数码管和 LCD 的显示实现等。

# 第 4 章　智能路侧系统网络通信模块设计

## 4.1　以太网通信介绍

传统的通信方式一般使用的是串口通信，即 RS232、RS485 标准等，但是其传输距离短，很难满足远程通信的距离和速度要求，而使用以太网则可以很好地解决这个问题，且可靠性高、实时性好[25]。

以太网是由 Xerox（施乐）公司创建，之后被 DEC、Intel 和 Xerox 三家公司合作开发成为了一个标准，它是目前使用最为广泛的一种基带总线局域网，它很大程度上取代了其他的局域网标准，其包括标准以太网(10Mbit/s)、快速以太网(100Mbit/s)和 10G(10Gbit/s)以太网，采用的是总线型拓扑和 CSMA/CD（载波监听多路访问及冲突检测）争用访问控制技术，它们都符合 IEEE802.3 标准[31]。

在网络飞速发展的今天，以太网与嵌入式系统和计算机系统的结合已是一种必然，原因除了能够满足远程通信和较高的通信速度外，其功能越来越丰富，比如局域网互连，以太网传送各种数据对硬件设备的要求并不高，只需要添加一个以太网控制器即可，而且简单实用，开放性好，协议、芯片的选择也可以非常灵活。本文选择以太网进行通信，就是因为其传输距离远，通信速度快，且能实现局域网互连。

现如今，嵌入式系统最常用的以太网解决方案就是以太网控制器+TCP/IP 协议，一个是硬件设计，另一个是软件编程实现，本文采用的也是这种方案[43]。

## 4.2　以太网控制器芯片选择与硬件电路设计

以太网控制器，也就是我们常说的网卡或者网络芯片，它是一种介于嵌入式设备和网络接口之间的芯片。网卡性能的好坏会对网络的通信速率、准确率和使用网络的应用软件产生影响，是网络通信好坏的硬件支持。

本文的设计中，数据量并不是很大，所以选用最高速度 10Mb/s 的 ENC28J60 网络芯片即可。ENC28J60 是一款兼容 IEEE802.3 的独立以太网控制器，能帮助用户实现嵌入式系统的以太网接入，并且任何拥有 SPI（行业标准串行化外设接口）的设备都能够通过该芯片进行以太网的接入。该芯片还支持全双工和半双工通信方式，适用不同的通信条件，本文采用全双工通信模式，该控制器体积小，占用供电电平一般选用 3.3V，拥有双色 LED 灯提示该控制器的工作状态[22]。此网络芯片的硬件电路原理图如图 4.1 所示。

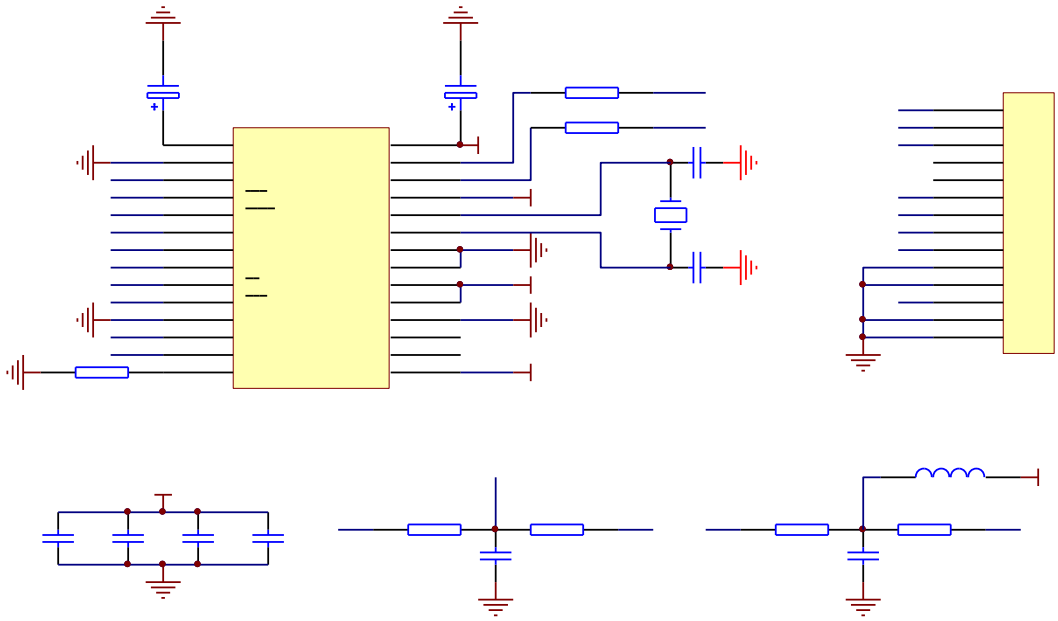


图 4.1　ENC28J60 电路连接原理图

图中，LAN_SCK，LAN_MOSI，LAN_MISO，LAN_nINT，LAN_nCS 端口需要与我们在 FPGA 上构建的 Nios 软核相连，前三者是与软核上 SPI 总线组件连接起来，后两者与软核的 PIO 口连接起来，LAN_nINT 作为中断信号需设置成输入端口，LAN_nCS 需设置成输出端口，其他的端口一部分是接入 RJ45 接口，另一部分无需设置。关于在 Nios 软核上添加 SPI 总线，在第 2.3.2 节已阐述，此处不再多做介绍。

## 4.3　网络通信的软件实现

### 4.3.1　TCP/IP 协议模型

TCP/IP 是目前全世界使用最广泛的一种网络通信协议标准。其实 TCP/IP 是一个协议系列，它至少包含了上百个各种功能各种内容的协议，用于将所有通信设备组成一个庞大的计算机网络。其中最根本、最主要的两个协议就是传输控制协议 TCP 和网络协议 IP，因此通常用 TCP/IP 来代表整个协议系列。TCP/IP 使所有的网络连接都有了标准化的网络协议系列，实现了不同协议间的整合[27]。

TCP/IP 协议不依赖于计算机的硬件配置和选用的操作系统平台，其协议标准是开放的，而且也不依赖于特殊的网络传输器件，用户几乎能够使用所有的网络硬件实现数据之间的交互[28]。并且，TCP/IP 通信协议中的设备的地址不会发生冲突，因为其地址的分配方案是统一的。

国际标准化组织定义的开放式系统互联(OSI)，将网络协议分为七个层次，分别是应用层、数据链路层、物理层、网络层、传输层、表示层和会话层[42]。而 TCP/IP 通信协

议相当于一个简单 OSI 模型，它总共有四层，分别是应用层、传输层、互联层和链路层。TCP/IP 结构对应的 OSI 模型如图 4.2 所示[29]。

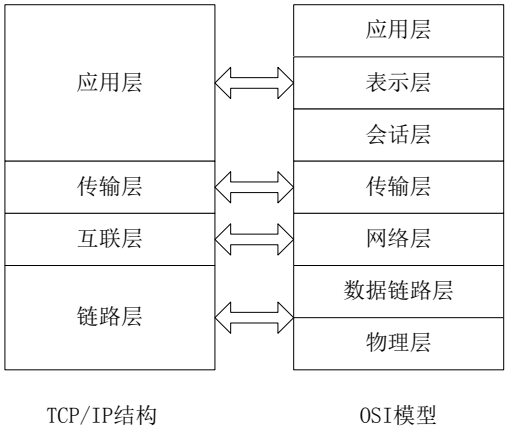



图 4.2　TCP/IP 协议与 OSI 模型对比图

各层之间只有相邻的层才可以进行信息传递，不可以跨层传输，而且每一层都有自己的协议栈。在 TCP/IP 分层模型中其中一些常用协议所处的位置如表 4.1 所示。

表 4.1　各个协议所在层级

| 层级 | 相关协议 |
|---|---|
| 应用层 | HTTP、FTP、Telnet、DNS 等 |
| 传输层 | TCP、UDP |
| 互联层 | IP、ICMP、ARP 等 |
| 链路层 | MAC 等 |

经过协议的分层，我们可以清楚了解到在 TCP/IP 下的数据包的流向，方便我们解析数据包。为了正确解析数据包，下面我们详细介绍 TCP 数据包的报文格式。TCP 报文段主要由 TCP 首部和 TCP 数据部分组成，他们共同构成了 IP 数据部分，如图 4.3。

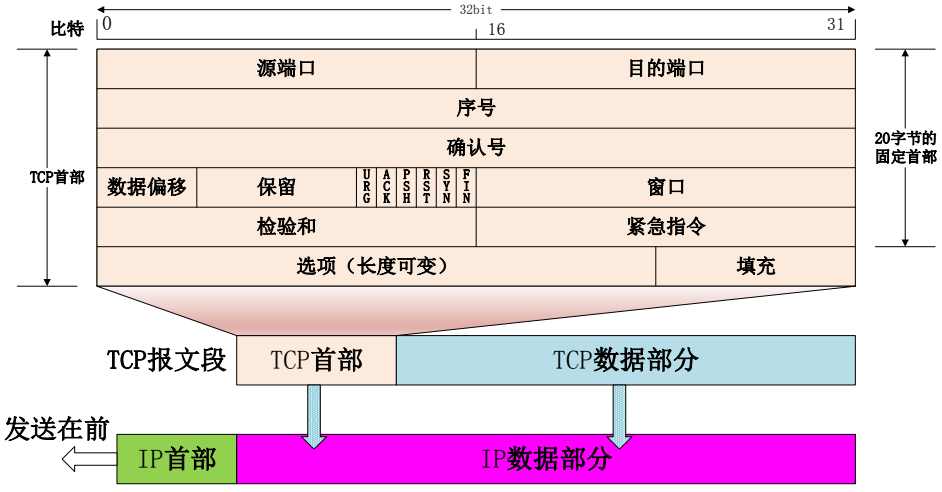



图 4.3　TCP 报文段格式

图中，TCP 首部报文段几个重要的部分介绍如下：

源端口和目的端口字段：各占 2 字节。传输层通过端口来实现复用和分用，是与应用层进行通信的服务接口。

序号字段：占 4 字节。此报文段数据的首个字节的序号，该序号由 TCP 传输时对数据的字节流统一自动标号所得。

确认号字段：占 4 字节。对方的下一个报文段的数据的序号字段。

数据偏移：占 4bit，即 4 位，单位是 32bit 字，它是实际有效数据与报文段第一位之间的距离。

确认比特 ACK：当 ACK=1 时，确认号字段有效，相当于应答信号。

同步比特 SYN：当 SYN=1 时，就表示这是一个连接请求。

终止比特 FIN：用来释放一个连接。当 FIN＝1 时，表明数据已发送完毕。

### 4.3.2 TCP/IP 传输实现

现今的 TCP/IP 协议较为复杂，内容很多，主要是适合 PC 机的通信，针对本设计，我们采用精简的 TCP/IP 协议，此协议没有嵌入操作系统，由于本文系统都是顺序执行单线程任务，FPGA 路侧端分别与控制中心和车载系统的通信只需将两者设为不同的通信端口即可放入一个线程当中，所以操作系统意义不大，然后配合 SOPC 技术构建的软核实现了路侧端与控制中心和车载的实时通信。

此协议栈，有几个函数比较重要，以下分别作简要概述。

1. void init_ip_arp_udp_tcp(unsigned char *mymac,unsigned char *myip,unsigned char wwwp)

此函数为初始化传输，主要用于填写 MAC、IP、端口参数。函数返回值为空。

2. unsigned char eth_type_is_arp_and_my_ip(unsigned char *buf,unsigned int len);

此函数为数据包解析，是否接收到指向本地 IP 的 ARP 请求，若是返回值为 1，若不是返回值为 0。

3. char eth_type_is_ip_and_my_ip(unsigned char *buf,unsigned int len);

此函数为数据包解析，是否接收到发送给本地 IP 的 IP 数据包，若是返回值为 1，若不是返回值为 0。

4. unsigned int get_tcp_data_pointer(void);

此函数为获取有效数据位置，若有数据返回值为有效数据位置，若无数据返回值为 0。

5. unsigned int fill_tcp_data_p(unsigned char *buf,unsigned int pos, const unsigned char *progmem_s);

此函数是将字符缓存到数组 buf 当中，用于发送给外部设备。返回值为在数据包中的位置信息。

6. void make_tcp_ack_from_any(unsigned char *buf);

此函数为向外部设备发送 ACK 响应包，返回值为空。

7. void make_tcp_ack_with_data(unsigned char *buf,unsigned int dlen);

此函数为将 IP 报文首部、TCP 首部以及有效数据进行封装，发送给外部设备，返回值为空。

FPGA 上数据传输通信流程图如图 4.4 所示，此精简的 TCP/IP 协议栈与 PC 控制中心和车载系统的网络传输对接良好。

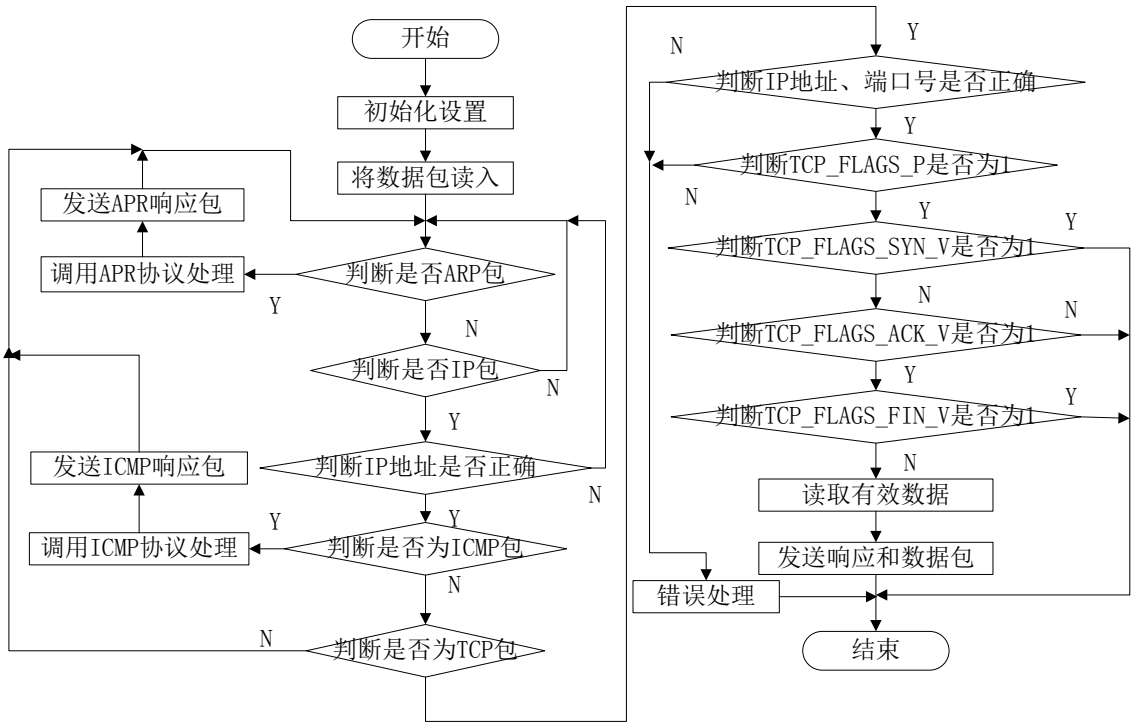



图 4.4　智能路侧系统端网络通信流程图

控制中心的网络传输是在 PC 机下实现，操作系统是 Windows 7，有完备的 TCP/IP 协议栈，而且还提供了名为套接字（Socket）的接口。应用程序和连接通过连接这个 Socket 进行网络通信。其网络通信框图如图 4.5 所示。具体实现是采用 Qt 界面的库函数来实现的。手持车载设备使用的是安卓系统，其通信流程与 PC 机类似。

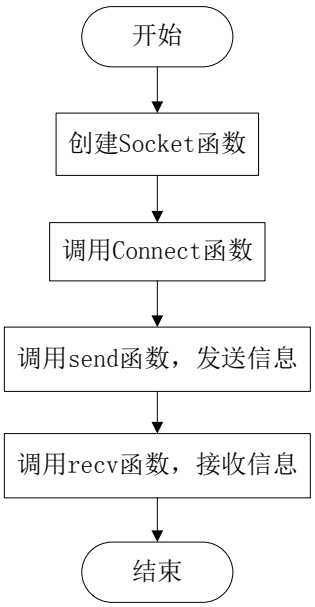



图 4.5　控制中心网络通信框图

## 4.4　控制中心界面设计实现

### 4.4.1　Qt 界面编程介绍

Qt 是一种跨平台的 C++用户界面应用程序框架（C++ GUI），能够为应用程序开发者提供艺术级图形用户界面所需的所有功能[32]。Qt 作为图形化界面框架基本上是同我们常用的 GTK+、MFC、Openwin 和 VCL 等相类似的产品。Qt 优良的跨平台性，使得 Qt 可以很好地支持大多数操作系统，不仅仅支持 Windows、HP-UX、IBMAIX、Linux 和 Unix 等 X11 平台，还支持嵌入式 Linux、Windows CE 和 Symbian 等嵌入式平台，而在 2014 年 4 月发布的跨平台集成开发环境 Qt Creator 3.1.0 还实现了对 IOS、Android 和 WP 的全面支持。所以，本文为了能够保证在各个平台下都可以使用本界面设计，而选择了跨平台的 Qt 图形化编程技术。

Qt 作为一款 C++用户界面应用程序框架具有很多优势。首先就是 Qt 的面向对象性，其非常高的模块化程度得益于良好封装机制，使得 Qt 具有较好的可重用性，为用户开发提供了极大的便利。Qt 的核心机制被称为 signals/slots 机制，在一些工具包中使用回调（callback）机制实现对象间通信，而在 Qt 中则使用信号和槽机制完成对象间通信。虽然该机制的回调速率比 callback 机制慢，但却提供了类型安全的、简单方便的优点。另外 Qt 还具有大量的开发文档以及非常丰富的 API，不仅包括超过 250 个的 C++类，还有基于模板的 date/time、directory management、file 和 I/O device 等类。Qt 的这些优势开发了很多我们熟知的产品，如 Google 地球、极品飞车、WPS Office、比特币等优秀

产品，同时也为我们的课题开发提供了极大地便利。

下面将要对本文涉及的 Qt 的主要编程知识进行简单介绍。

1.常用窗口部件

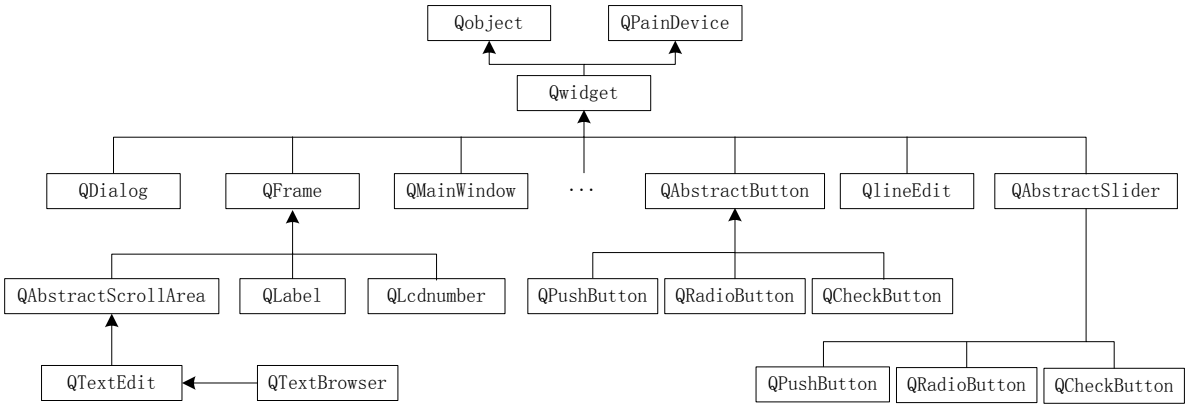



图 4.6　窗口部件类关系图

在我们创建窗体文件时，会有 QMainWindow、QWidget 和 QDialog 这三种基类进行选择。QMainWindow 类是带有一个有菜单栏、工具栏和一个状态栏的主应用程序窗口基类，QDialog 类是对话框窗口的基类，但它们则继承于 QWidget 基类，如图 4.6 所示，说明了本文中涉及的类继承关系。

如表 4.2 所示，这里简单介绍一些主要控件。

表 4.2　常用控件功能介绍

| 控件类 | 名称 | 功能 |
|---|---|---|
| QLabel | 标签 | 显示文本和图片 |
| QTextEdit | 文本编辑 | 一般和多信息文本编辑 |
| QTextBrowser | 文本浏览器 | 只读文本显示 |
| QPushButton | 按钮 | 常用按钮 |
| QRadioButton | 单选按钮 | 有两种状态 |
| Qtimer | 定时器 | 提供了定时器的信号和单触发的定时器 |

2.窗口布局管理

在进行界面设计的时候，不是随意的将所需控件直接拖拽到界面上，这样虽然对功能的实现没有多大影响，但对于一个完整的软件设计而言，窗口的布局管理是必不可少的。布局管理不仅仅只是窗口控件的整齐排列，还有窗体控件的自适应窗口等等。在 Qt 中主要使用 QLayout 类及其子类来进行窗口的布局管理，如图 4.7 所示。

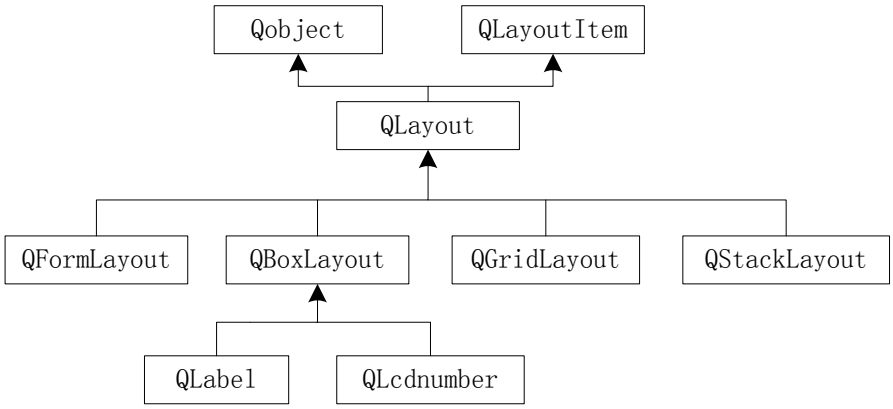



图 4.7　窗口布局类结构

布局管理设计时主要使用 QLayout 类的以下子类：QBoxLayout、QFormLayout、QGridLayout 和 QStackedLayout。本课题中窗口布局管理主要用到 QBoxLayout 类和 QGridLayout 及两者的交叉使用，如表 4.3 所示。

表 4.3　常用窗口布局类

| 布局管理类 | 名称 | 功能 |
|---|---|---|
| QHBoxLayout | 水平布局管理器 | 按水平方式进行窗口布局 |
| QVBoxLayout | 垂直布局管理器 | 按垂直方式进行窗口布局 |
| QGridLayout | 网格布局管理器 | 按格栅方式进行窗口布局 |

3.信号与槽机制

Qt 的核心特征是信号与槽机制，这也是 Qt 区别于其他 C++开发框架的最大特征，并通过该机制实现对象间的通信。如图 4.8 所示，这种信号与槽机制不仅仅实现一个信号对应一个槽，还可以实现一个信号关联多个槽以及多个信号关联单一槽，然而一个信号关联多个槽时，当信号被发射时，多个槽的执行顺序是逐一并随机执行并且无法指定先后顺序。信号与槽的通信机制，即是当一个特定事情发生时就可以发射一个信号，Qt 有很多预先定义好的信号，如 clicked，除此之外我们也可以通过继承的方式来添加自己的信号，而槽本身就是一个函数，它具有响应信号的能力，在信号被发射后槽就会被调用响应该信号。

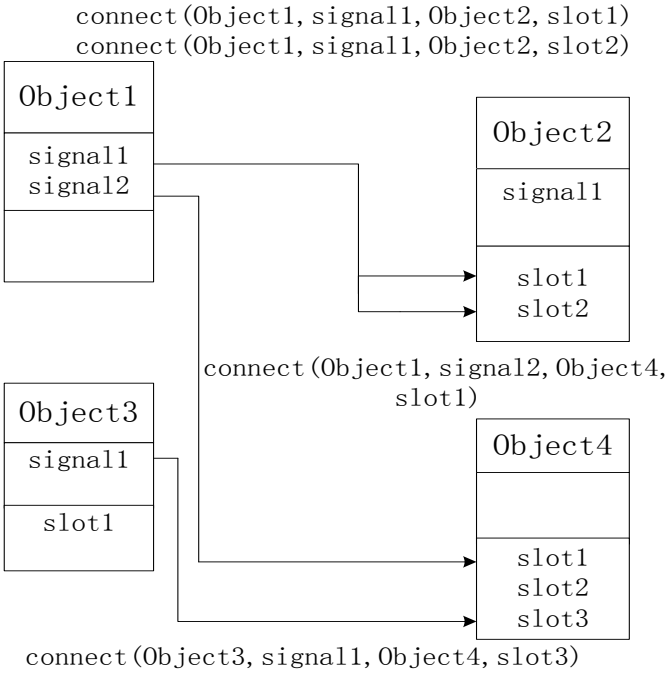



图 4.8　信号与槽的关联图

信号的声明需要用到关键字 signals，信号具有以下特点：

（1）信号的声明只能在 QObject 类及其子类中，并且信号前面不需要任何限定符，如 public、private 或 protected 等。

（2）信号只需声明不需要定义实现，且返回类型为 void 类型。

（3）信号的的发射需要用到 emit 关键字。

当一个信号被发射时，与其相关联的槽将被立刻执行，像正常的函数调用一样，只有当所有的槽返回以后发射函数（emit）才返回。

槽函数的声明需要用到 slots 关键字，槽函数具有如下特点：

（1）在声明槽函数时，需要用到限定符，也可以声明为虚函数。

（2）一个信号被触发后，多个关联槽函数的执行顺序是随机的，且不可指定。

（3）需要继承自 QObject 或其子类，槽函数同其他成员函数一样定义。

上面分别介绍了信号和槽函数的特点，信号和槽函数的关联需要用到 connect 关键字实现。函数的原型如下：

bool QObject::connect(const QObject* sender,const char* signal,const QObject * receiver,const char * member)

这条语句可以将信号和槽关联起来，发送者 sender 和接受者 receiver 都是指向 QObject 类型的指针。指定信号 signal 必须套入宏 SIGNAL（）中，指定槽函数必须套

入宏 SLOT（）中。

### 4.4.2 控制中心的 Qt 界面实现

控制中心的界面设计主要包括登陆界面的设计和网络传输界面的设计。下面将具体介绍各个界面设计的具体实现。

1.登陆界面的设计

此界面的设计需要设置用户名和密码的输入，将输入的用户名和密码与我们设置的进行对比，若完全一致则进入下面的网络传输界面，否则会提示登陆失败信息，清空用户名和密码栏，重新输入，当输入失败次数大于 3，退出登陆界面。

用户名和密码的比对具体实现如下：

```
if(ui->usrLineEdit->text()==tr("admin")&&ui->pwdLineEdit->text()==tr("123"))
    {
        accept();
    };
```

提示错误信息具体实现如下：

```
QMessageBox::warning(this,tr("警告对话框"),tr("用户名密码输入错误"),QMessageBox::Yes);
```

整个登陆界面的设计如图 4.9 所示。

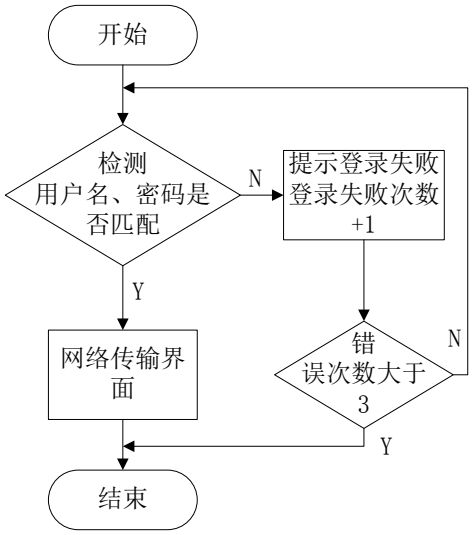



图 4.9　登陆界面设计流程图

2.网络传输界面的设计

在设计网络传输界面之前，需要在工程的.pro 文件下添加 QT += network，需要再添加<QTcpSocket>头文件，之后我们需要设置连接的 IP 地址和端口号，然后构建通信

机制的信号与槽关联函数，其设置如下：

- connect(&tcpSocket,SIGNAL(connected()),this,SLOT(slotConnected()));

在通信连接上之后，会执行 slotConnected()槽函数，进行连接之后的初始化工作。

- connect(&tcpSocket,SIGNAL(disconnected()),this,SLOT(slotDisconnected()));

在通信断开之后，会执行一次 slotDisconnected()槽函数，进行断开连接之后的结束工作。

- connect(&tcpSocket,SIGNAL(readyRead()),this,SLOT(dataReceived()));

当有可读数据时会发射 readyRead()信号，此时开始数据接收操作。

- connect(&tcpSocket,SIGNAL(error(QAbstractSocket::SocketError)),this,SLOT(displayError(QAbstractSocket::SocketError)));

当发生错误时会发射 error()信号，此时输出错误信息。

## 4.5　本章小结

本章首先对以太网通信进行简要介绍，然后介绍了本设计所选用的网络芯片及其与 FPGA 和 RJ45 接口的硬件电路的连接，之后分别针对网络传输实现和界面设计的关键技术进行阐述，最终完成了整个路侧系统网络通信模块部分的设计。

# 第 5 章　视频采集与图像信息公告模块设计

对于智能路侧系统，为了获取准确的交通参数信息，就必须加入传感器，比如电感线圈、红外线传感器、超声波传感器等等，但是这些传感器，有的成本太高，有的架设困难，有的检测可能并不准确。考虑到这些因素，本文使用一种视频采集传感器，此传感器价格不高，检测准确，架设起来也很方便。

## 5.1　视频采集传感器的选择

视频采集传感器，即摄像头主要可分为模拟摄像头和数字摄像头，即 CCD 摄像头和 CMOS 摄像头。CCD 称为电荷耦合器件，而 CMOS 称为互补型金属氧化物半导体，这是当今常用的两种视频采集传感器。CCD 摄像头总体上就三根 RGB 线，通过 AD 模数转换芯片转换为数字信号，接收端接收信号并解码，便能得到图像视频。CMOS 摄像头也是经过模拟采样，之后 AD 转换，再经过数字信号处理，最终得到 16 位的数据接口。COMS 摄像头相比于 CCD 摄像头，虽然 CCD 摄像头在图像效果和抗噪声性能方面优于 CMOS 摄像头，可是在价格上 CMOS 摄像头较为便宜，很大一部分电路都已经在内部完成了，电路设计上就会较为简单，而且功耗小。随着技术与工艺的发展，CMOS 摄像头在抗噪声性能上得到了一定的改善，分辨率也明显得到提高。综上所述，本设计使用的是 Ovmnivision 公司的 CMOS 摄像头模块 OV7670，其实物图如图 5.1 所示[35]。

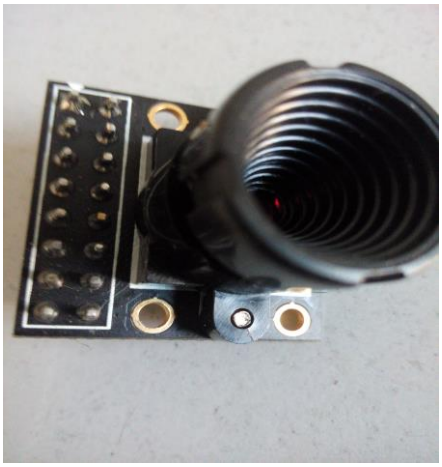


图 5.1　OV7670 摄像头实物图

## 5.2　视频采集与图像公告模块的硬件电路设计

此模块的硬件部分主要包含摄像头 OV7670，SDRAM 以及 VGA 接口，再加上 FPGA 对这 3 个外部模块的控制就能构成一个视频采集显示系统。以下对这 3 个模块的硬件结构和电路加以介绍。

1.摄像头 OV7670

OV7670 是一款高度集成的 CMOS 传感器，其具有高灵敏度，能够在低光照下使用，供电电压无需要求太高等优点，OV7670 摄像头模块的内部构成比较复杂，主要有 5 大部分组成，包括 1：图像模数转换；2：测试图案发生器；3：数据输出；4：656×488 的感光阵列（其有效像素是 640×480）；5：SCCB 通信接口等，如图 5.2 所示。

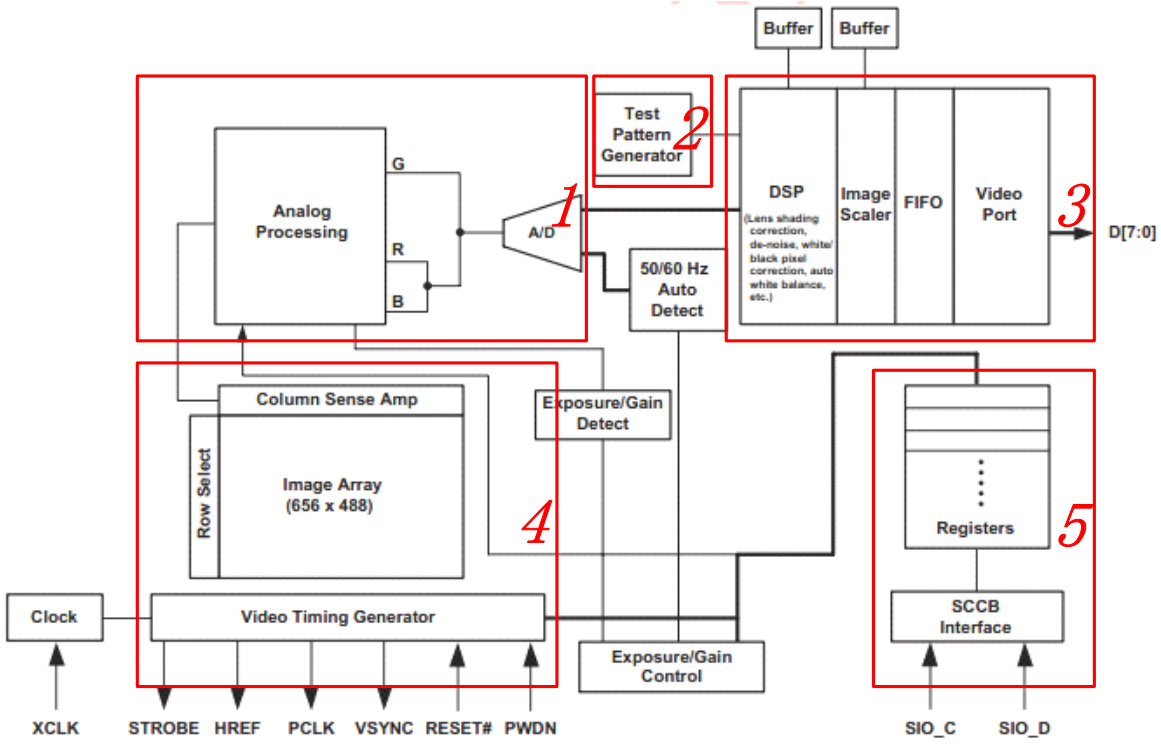



图 5.2　OV7670 内部结构图

OV7670 通过 SCCB 总线控制，支持多种数据输出格式，如 RawRGB，RGB565、YUV422 与 YCbCr 等。由于内部集成了模拟信号处理电路，所以还可以完成曝光、自动增益控制，色度调节等，并且具有降噪调节、坏点补偿等功能。OV7670 芯片虽然集成了很多电路，但它仍然保持了非常小的尺寸。

OV7670 的硬件电路原理图如图 5.3 所示。U1 是 OV7670 摄像头视频采集传感器，采用的是球状引脚栅格阵列封装技术（BGA）封装。P1 是外部接口，VCC 为 3.3V 电压。在电路上特别要注意的是，SCCB 的两根信号线时钟线和数据线都必须上拉一个 4.7k 的电阻，否则会因为电路内部结构本身的问题，使得数据线和时钟线一直输出低电平。

2.SDRAM

我们 SDRAM 芯片采用的是 Hynix 公司生产的 H57V2562GTR 芯片，其内存大小为 256Mbit（16M * 16bit），其电路连接原理图如图 5.4 所示。

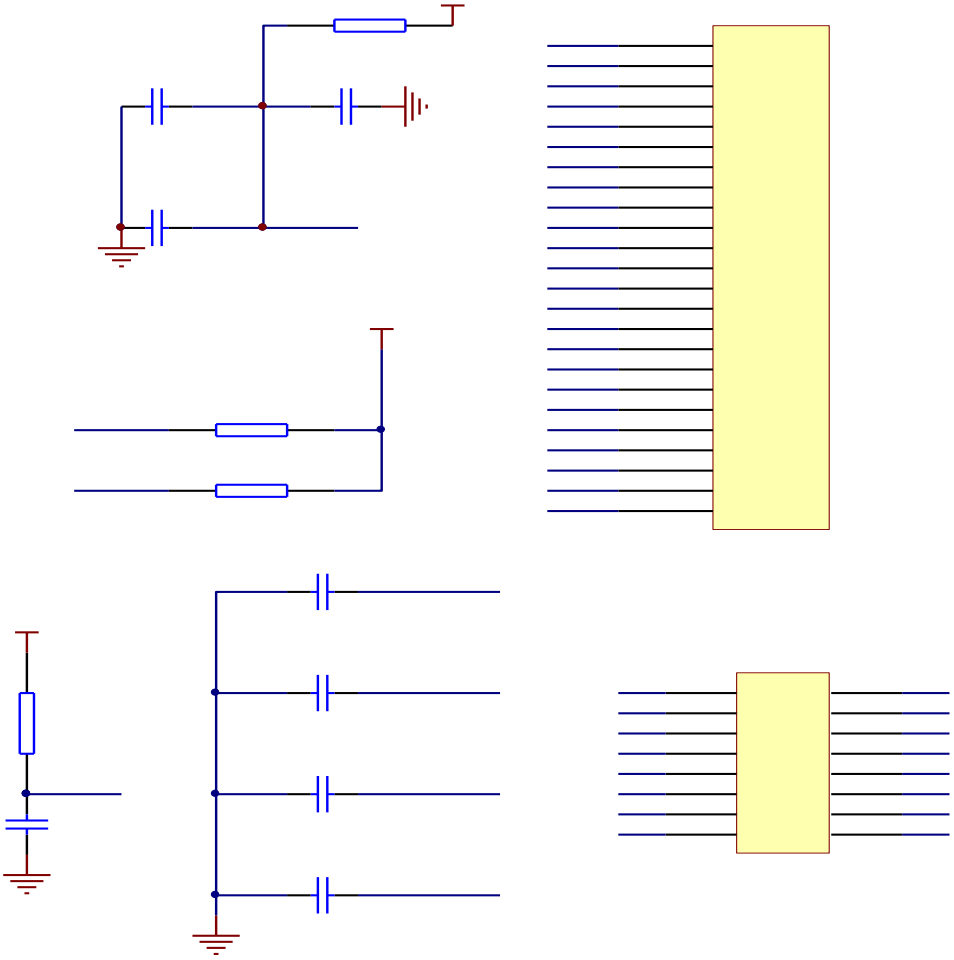


图 5.3　OV7670 电路原理图

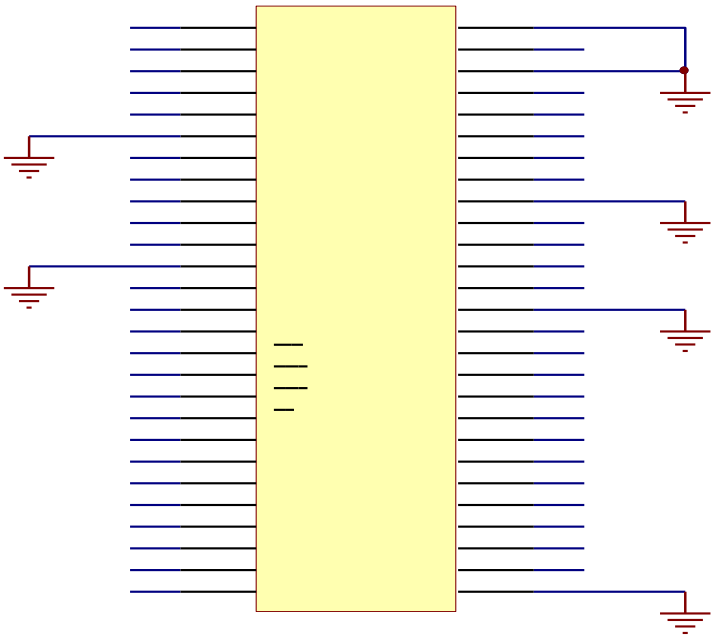


图 5.4　SDRAM 电路原理图

3.VGA 接口

VGA 是一种视频传输标准，其接口有 15 个接口针孔，每排 5 个孔，共分 3 排，其中最主要的是红基色、绿基色、蓝基色、行同步和场同步，通过编程控制这几个信号端口，就能使得 VGA 显示器正常显示。由于本系统采集的图像是 RGB565 格式的，一个像素是 16 位的数字信号，红色和蓝色分量都占 5 位，绿色分量占 6 位，为了转换成 VGA 可以接受的三基色模拟信号，所以需要加入 DAC 电路，实现数模转换，考虑到成本问题，本文采用的是搭建电阻网络来实现数模转换的，其具体电路设计如图 5.5 所示。

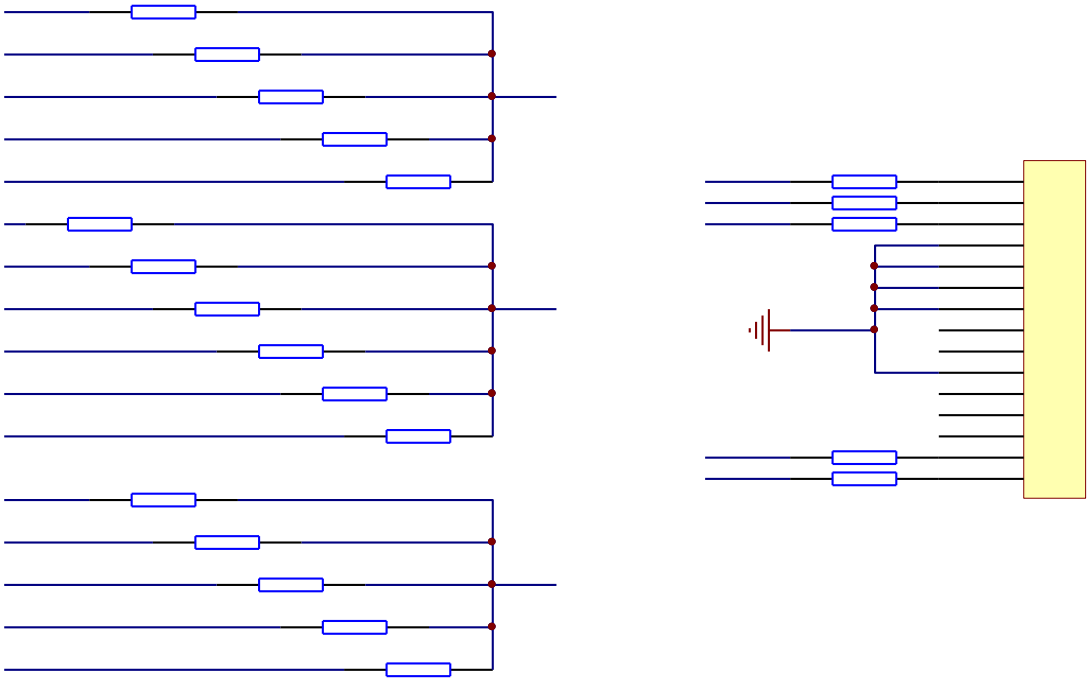


图 5.5　VGA 接口电路原理图

## 5.3　视频采集与图像公告模块的模块化设计

在设计中，OV7670 摄像头用来实时采集图像信息，其内部相关寄存器的配置信息被写入到 I2C 控制模块，通过 I2C 总线将 OV7670 配置为 VGA 分辨率，图像数据格式为 RGB565 等等。采集到的图像送入 SDRAM 控制器，经过处理，实现内部 2 个 bank 的乒乓操作，使得输入输出视频流读写加速且无冲突，之后 VGA 控制器从 SDRAM 控制器中读取图像信息，显示在 VGA 显示器上，整体设计框图如图 5.6 所示[36]。

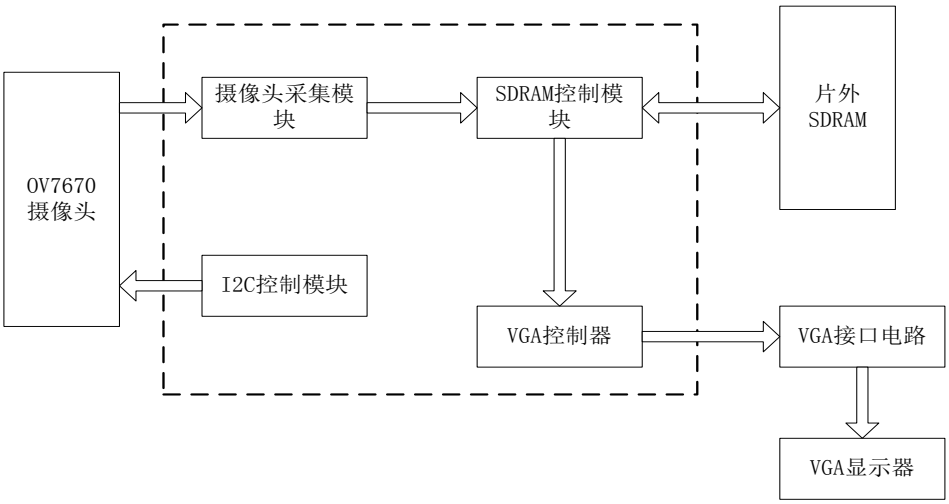



图 5.6　视频采集系统整体设计框图

图中虚线方框外部是独立的外设电路，在第 5.2 节已经对每一个外设详细介绍过，虚线方框内部是使用 Verilog HDL 设计的控制模块，此部分是使系统能正常工作的重点。

### 5.3.1　I2C 控制模块

FPGA 通过 SCCB 总线电路完成 OV7670 的初始化，SCCB 总线是一种串行摄像机控制总线，由 OmniVision 公司开发，是一种 3 线的总线，它由 SCCB_E、SIO_C 和 SIO_D 所构成，第一个是使能信号，后面两个分别是时钟信号和数据信号，这个总线可以实现一个主设备对多个从设备的控制，本系统只有一个摄像头，即一个从设备，所以可以不需要使能信号，只需要两根线时钟线和数据线，所以 SCCB 的时序就与 I2C 几乎完全相同，我们重新定义时钟线是 CMOS_SCLK，数据线是 CMOS_SDAT，其中数据线是双向的，既可以读传输，也可以写传输。I2C 的读写时序图如图 5.7，5.8 所示。

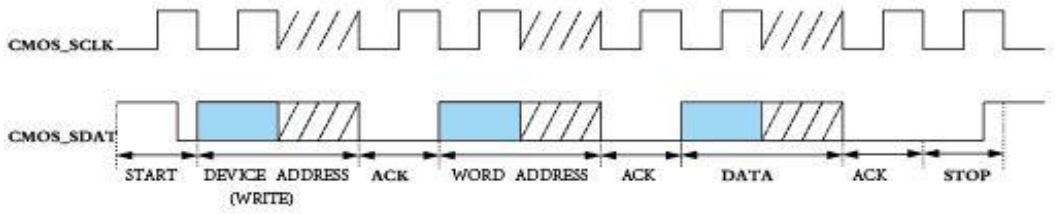



图 5.7　I2C 写时序

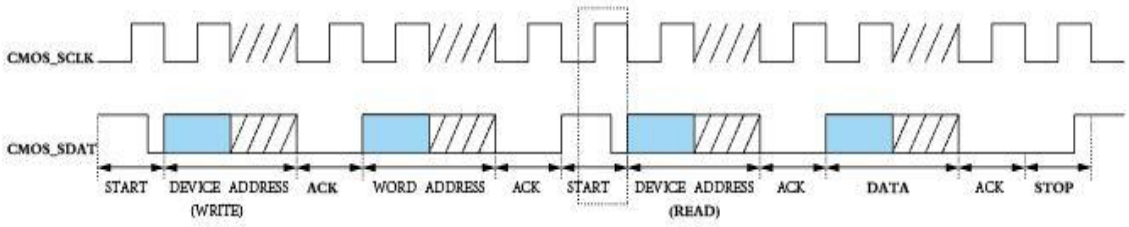



图 5.8　I2C 读时序

两图中前面的 START 是 I2C 的起始信号，后面的 STOP 是 I2C 的终止信号，中间的部分是 I2C 数据传输的部分。读时序和写时序的区别在于，写入的第一个设备地址的最后一位如果为 0 就是写，为 1 就是读，同时当发送完寄存器数据地址，接收到应答信号后，如果想要执行读命令，那还要再进行一次开始信号 START，然后再发送一次设备地址，当再收到应答信号后就可以从里面读数据了。

实际视频采集，我们是用 I2C 配置 OV7670，所以大部分用到的是 I2C 的写传输，只有厂商识别字节是需要读的。摄像头的写传输，根据时序图，需要先写设备地址，再写寄存器地址，最后写入寄存器的数据。OV7670 的设备地址是 0x42，需要配置的寄存器比较多，其中关键的寄存器地址和需要配置的值如表 5.1 所示，配置完成之后 OV7670 就可以采集图像了。

表 5.1　寄存器相关配置

| 地址 | 寄存器名称 | 配置数值 | 功能描述 |
|---|---|---|---|
| 00 | AGC | 00 | AGC 自动增益控制 |
| 01 | BLUE | 40 | 蓝色通道增益 |
| 02 | RE | 40 | 红色增益通道 |
| 03 | VREF | 0A | 帧竖直方向控制 |
| 11 | CLKRC | 80 | 内部时钟 |
| 12 | COM7 | 04 | 复位，选择图像输出模式 |
| 1C | MIDH | 7F | 厂商识别字节-高位（只读） |
| 1D | MIDL | A2 | 厂商识别字节-低位（只读） |
| 32 | HREF | B6 | HREF 控制 |
| 37 | ADC | 1D | ADC 控制 |
| 3A | TSLB | 04 | 行缓冲测试选项 |
| 40 | COM15 | D0 | 设置数据位 RGB565 |
| 6B | DBLV | 00 | 旁路 PLL 倍频 |

### 5.3.2　摄像头采集模块

在 I2C 控制模块中，已经将 OV7670 配置成 VGA 分辨率，RGB565 模式。在 VGA 分辨率模式下，VSYNC（垂直场帧信号）、HREF（水平场帧信号）、PCLK（像素时钟）等起着重要的作用。此 3 个信号的时序关系如图 5.9 所示，图中 PCLK 是连续的，为了能直观的看出图像接收的区域，做了一点的简化。

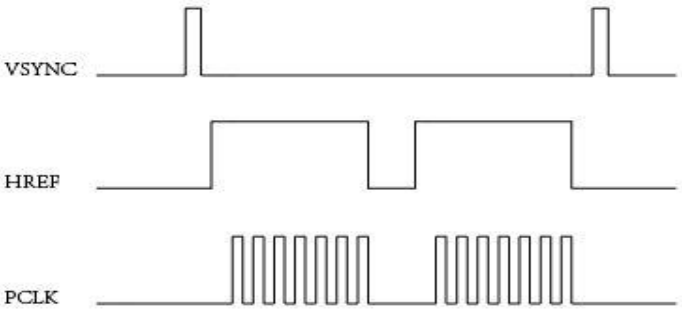



图 5.9　摄像头采集各信号时序关系

输出的每一帧图像数据里都包含有数以万计的像素，因此判断一帧图像的开始和结束就十分重要。在本系统中采用检测 VSYNC 信号上升沿的方法来判断一帧图像的开始和结束，即检测到一个 VSYNC 信号上升沿至下一个 VSYNC 上升沿为一帧图像数据的接收时刻[43]。OV7670 图像的时序和 VGA 十分相似，但 OV7670 摄像头采集时序是主动输出的，VGA 时序是需要 FPGA 来产生的。在 HREF 信号为高电平时图像数据才能有效接收，PCLK 才会发挥作用，PCLK 频率接近 25MHz，此时 FPGA 可以在 PCLK 上升沿读取一个 RGB565 格式 16 位像素的高 8 位，在下一个时钟周期再读低 8 位，数据拼接的具体实现如下：

```
if(~CMOS_VSYNC & CMOS_HREF)
    begin
        byte_state <= byte_state + 1'b1;
        case(byte_state)
        1'b0 :    Pre_CMOS_iDATA[7:0] <= CMOS_iDATA;
        1'b1 :    CMOS_oDATA[15:0]<={Pre_CMOS_iDATA[7:0],
    CMOS_iDATA[7:0]};
        endcase
    end
```

在 OV7670 配置完毕之后，采集到的图像的前 10 帧会很不稳定，所以建议舍弃。

### 5.3.3　SDRAM 控制模块

在本文中，SDRAM 控制模块主要分为 FIFO 缓冲模块和 SDRAM 读写控制器，其中为了实现图像数据的无缝缓冲与处理，SDRAM 读写控制器采用了乒乓操作。FIFO 与 SDRAM 读写控制器的连接关系如图 5.10 所示。


图 5.10　FIFO 与 SDRAM 读写控制器的连接

#### 5.3.3.1　FIFO 模块

FIFO 是先入先出队列，可以在具有相同的时钟域（同步 FIFO）和不同的时钟域（异步 FIFO）系统中做数据缓存器，解决了匹配速率等问题[44]。本文 SDRAM 与 VGA 和 OV7670 图像采集的时钟频率不相同，前者时钟频率 100MHz，后两者时钟频率都接近于 25MHz，所以需要引入异步 FIFO。FIFO 可以不需要外部的读写地址线，其主要参数有宽度、深度、读时钟、写时钟以及空标志和满标志。FIFO 一次能够读写的数据位数被称作 FIFO 的宽度，我们设为 N，FIFO 中能够存储的宽度为 N 的数据的个数被称为 FIFO 的深度。读写数据动作发生在每一个时钟沿，可以是上升沿，也可以是下降沿。

在 FIFO 中，空标志和满标志都是不可或缺的。我们假设 FIFO 的深度是 N，在 FIFO 计数器里面的值等于 N 而且也没有进行读操作，或者 FIFO 计数器里面的值已经等于 N-1 但是还在进行写操作的两种情况下，满标志置为 1，除此之外置 0。同样，在 FIFO 计数器里面的值已经全部读完等于 0 而且也没有进行写操作，或者 FIFO 计数器的值等于 1 时，但是还要进行读操作，则空标志置为 1，除此之外置 0。在实际运用中，为了有效避免读空和写溢出的情况，可能还需要设置更多的信号来提前预知 FIFO 的状态。图 5.11 给出了一种 FIFO 的原理图。

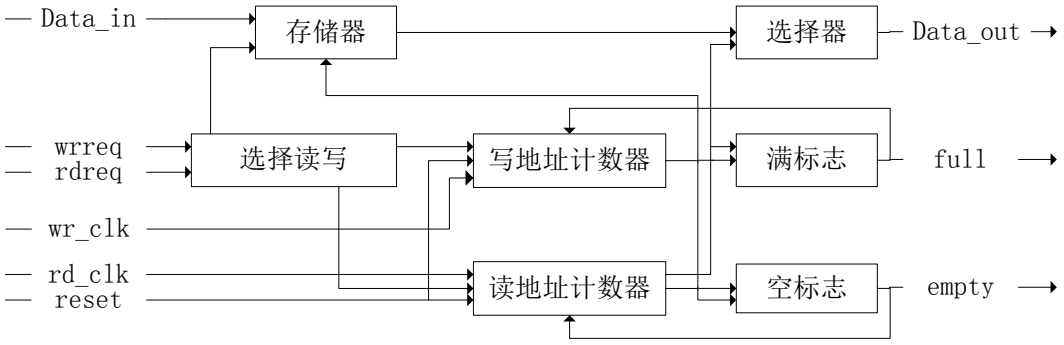



图 5.11　FIFO 原理图

本文通过 Quartus II 中自带的 FIFO 模块添加了宽度为 16 位，深度为 512 的 SDRAM 读 FIFO 和写 FIFO。在写 FIFO 模块中，模块当中使用 rdusedw 来标志 FIFO 的计数状态，表示当前 SDRAM 有多少字可以读取；在读 FIFO 模块中，模块当中使用 wrusedw 来标志 FIFO 的计数状态，表示当前 SDRAM 向 FIFO 已经写入了多少字。本文中 FIFO 的深度为 16，所以当 wrusedw 等于 15 时表示 FIFO 已经完全写满。FIFO 是一种环形结

构，因此此时若继续写入数据，在下一个有效时钟沿到来时，wrusedw 会等于 0，而不是保持 15 不变。同理，当 rdusedw 为 0 继续进行读操作，rdusedw 会在下一时刻变为 15。

#### 5.3.3.2 SDRAM 乒乓操作

为了提高 SDRAM 中数据的读取和写入速度，本系统中采用了基于 SDRAM 的乒乓操作来进行读写，所以摄像头采集的图片信息可以几乎无延时地显示在 VGA 显示器上，这是 FPGA 中面积换取速度的一种体现。所谓乒乓操作是一个主要用于数据流的处理技巧，其具体实现如图 5.12 所示[47]。

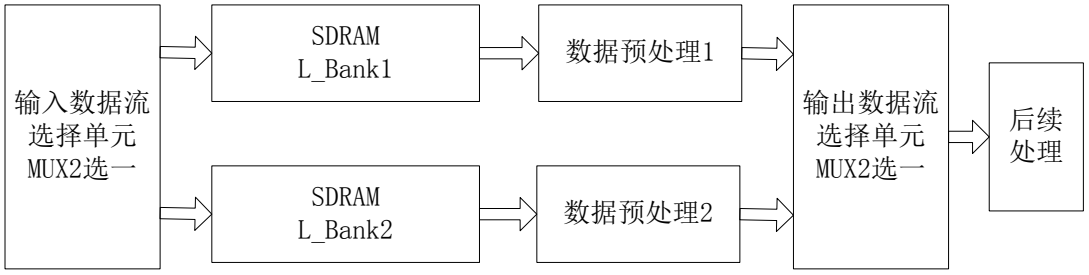



图 5.12 基于 SDRAM 乒乓操作

本文数据流也即图像流通过输入数据流选择单元模块送入 SDRAM 的 Bank1 和 Bank2 中，在第 1 个缓存时间周期内，将输入的图像流缓存到 SDRAM 的 Bank1。在第二个缓存时间周期内，输入数据流选择单元模块将输入的图像流缓存到 SDRAM 的 Bank2，同时，在第一个周期 SDRAM 的 Bank1 缓存的数据流经过预处理，再通过输出数据流选择单元模块送到后续处理中，进行后续的数据处理。在第 3 个缓存时间周期内，输入数据流又缓存到 SDRAM 的 Bank1，与此同时，SDRAM 的 Bank2 缓存的图像流经过预处理，通过输出数据流选择单元模块，送到后续处理，然后循环往复。

根据 SDRAM 控制模块中读确认信号和写确认信号有效的标志，当我们接收到读确认信号的时候置 frame_read_done 信号有效，接收到写确认信号的时候置 frame_write_done 信号有效。当 SDRAM 初始化完成并且进入写状态时，写乒乓操作控制模块首先将数据写入 Bank1 中并判断 frame_write_done 是否有效。若有效，则继续将数据写入 Bank2，若无效则继续保持 Bank1 的写状态。其后当 frame_write_done 再次有效时，再将数据写入到 Bank1，依次循环。

读乒乓操作和写乒乓操作过程非常相似，不过是先 Bank2 读取数据，然后判断 frame_read_done 信号是否有效，其有效时，转换至 Bank1 读取数据，否则继续保持 Bank2 的读状态。其后每当 frame_read_done 有效时，都转换一次读取的 Bank，读乒乓操作和写乒乓操作正好错开，互不影响。

### 5.3.4　VGA 显示模块

VGA 的显示扫描时序与 OV7670 的图像时序很类似，VGA 的扫描是从屏幕的左上方开始，从左到右，由上到下的，每扫描完一行对行进行同步，每扫描完一帧，对场进行同步，常用的几种 VGA 显示模式的水平时序和垂直时序分别如表 5.2 和表 5.3 所示，VGA 的正常显示只能在水平时序与垂直时序的有效时间中。我们使用的是分辨率为 640×480，刷新速率为 60Hz 的显示模式[54]。

表 5.2　VGA 显示的水平时序

| 分辨率 | 刷新速率(Hz) | 像素频率(MHz) | 同步脉冲 | 后沿 | 有效时间 | 前沿 | 帧长 |
|---|---|---|---|---|---|---|---|
| 640×480 | 60 | 25 | 96 | 48 | 640 | 16 | 800 |
| 640×480 | 75 | 31.5 | 64 | 120 | 640 | 16 | 840 |
| 800×600 | 60 | 40 | 128 | 88 | 800 | 40 | 1056 |
| 800×600 | 75 | 50 | 80 | 160 | 800 | 16 | 1056 |
| 1024×768 | 60 | 65 | 136 | 160 | 1024 | 24 | 1344 |
| 1024×768 | 75 | 78.8 | 176 | 176 | 1024 | 48 | 1312 |
| 1280×1024 | 60 | 108 | 112 | 248 | 1280 | 48 | 1688 |
| 1208×800 | 60 | 84 | 136 | 200 | 1280 | 64 | 1680 |
| 1440×900 | 60 | 106 | 152 | 232 | 1440 | 80 | 1904 |

表 5.3　VGA 显示的垂直时序

| 分辨率 | 刷新速率(Hz) | 像素频率(MHz) | 同步脉冲 | 后沿 | 有效时间 | 前沿 | 帧长 |
|---|---|---|---|---|---|---|---|
| 640×480 | 60 | 25 | 2 | 33 | 480 | 10 | 525 |
| 640×480 | 75 | 31.5 | 3 | 16 | 480 | 1 | 500 |
| 800×600 | 60 | 40 | 4 | 23 | 600 | 1 | 628 |
| 800×600 | 75 | 50 | 3 | 21 | 600 | 1 | 625 |
| 1024×768 | 60 | 65 | 6 | 29 | 768 | 3 | 806 |
| 1024×768 | 75 | 78.8 | 3 | 28 | 768 | 1 | 800 |
| 1280×1024 | 60 | 108 | 3 | 38 | 1024 | 1 | 1066 |
| 1208×800 | 60 | 84 | 3 | 24 | 800 | 1 | 828 |
| 1440×900 | 60 | 106 | 3 | 28 | 900 | 1 | 932 |

## 5.4 本章小结

本章首先选择了一款 CMOS 视频采集传感器 OV7670，然后介绍了摄像头 OV7670、SDRAM、VGA 接口的电路原理图及其设计中需要注意的地方，最后在 FPGA 上通过 Verilog HDL 硬件语言设计实现了 I2C 控制模块、SDRAM 控制模块和 VGA 显示模块三大模块，结合起来完成了 OV7670 从采集到显示的设计要求。

# 第 6 章　系统调试与结果分析

## 6.1　系统调试方案

系统的所有设计完成之后，我们需要不断调试各个模块的运行情况以及系统的整体运行情况。系统中很多模块的测试可以预先用 modelsim 工具仿真，预测结果，本文这部分没有阐述，直接用实际电路进行调试。测试环境是实验室，模拟实际交通环境，搭建了智能路侧调试系统。系统的整体调试图如图 6.1 所示。

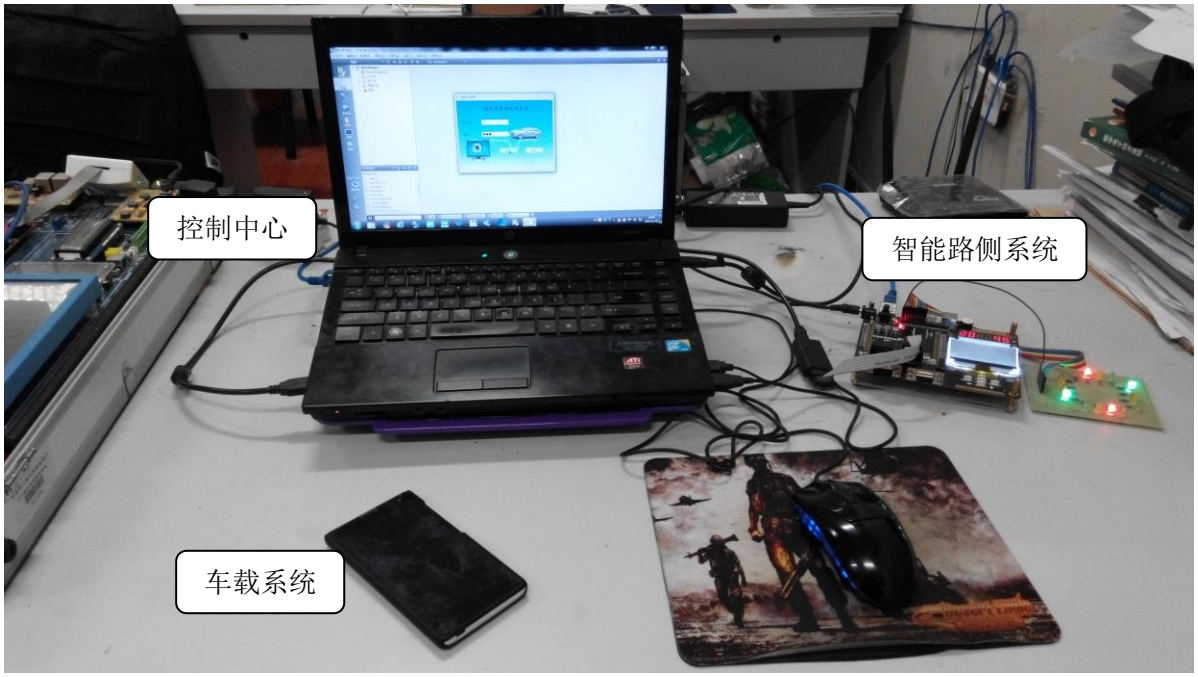



图 6.1　系统调试图

图中，FPGA 及其外设构成了智能路侧系统，安卓手持设备模拟的是车载系统，笔记本电脑作为控制中心，同时用来调试 FPGA。由于本系统调试比较繁琐，所以本章按照论文结构的形式分以下几个模块介绍。

## 6.2　智能路侧交通控制系统的调试

智能路侧交通系统主要包括交通信号灯和 LCD 路侧引导标识，交通信号灯一般都架设在交通十字路口上，控制车辆的通行，而 LCD 路侧引导标识为了能够提前引导车辆的行驶，所以需要架设在十字路口之前的路边。两者的运行情况都需要基于控制中心发来的控制信息，其中控制中心的控制界面如图 6.2 所示。

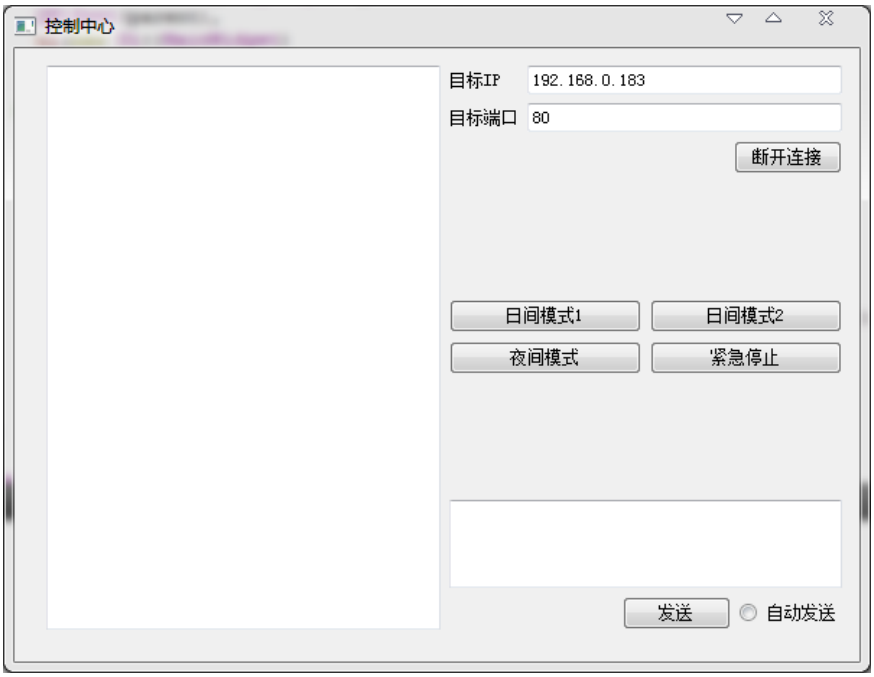
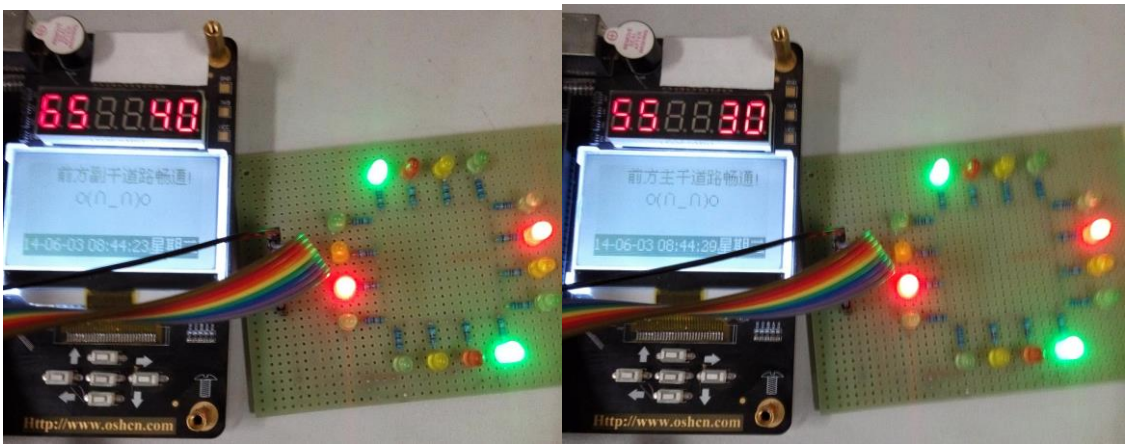


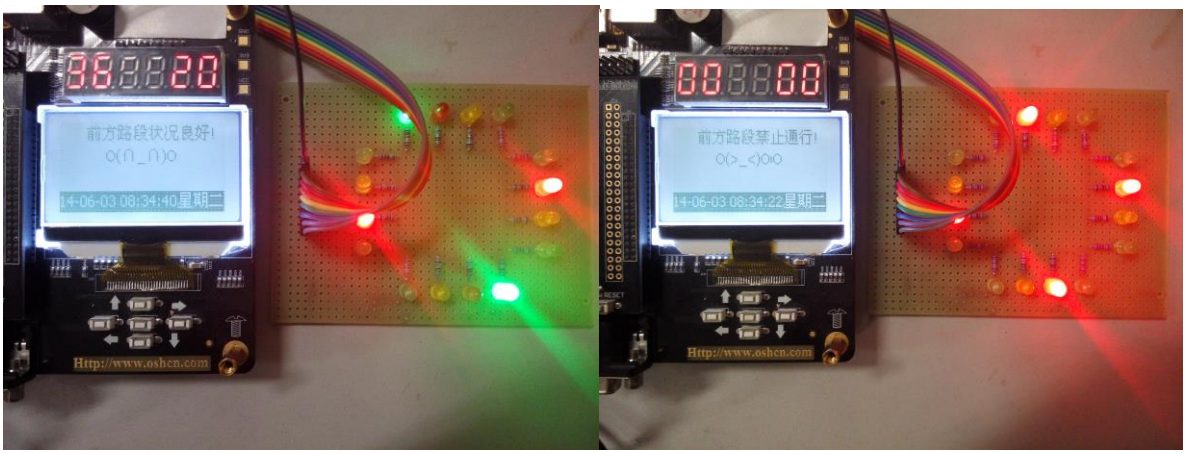

图 6.2　控制中心控制界面图

图中点击“日间模式 1”会触发 start1 信号，点击“日间模式 2”会触发 start2 信号，点击“夜间模式”会触发 start3 信号，点击“紧急停止”会触发 stop 信号。这 4 个信号是控制整个系统运行状况的重要信号。其中，接收到 start1 信号说明主干道车流量大于副干道车流量，接收到 start2 信号说明副干道车流量大于主干道车流量，接收到 start3 信号说明是夜间车流量较小的情况，当接收到 stop 信号说明发生交通事故或者急车强通的情况。

根据以上分析，比对第 3 章交通信号灯的配时方案进行调试，其运行结果正常且稳定，结果如图 6.3 所示。



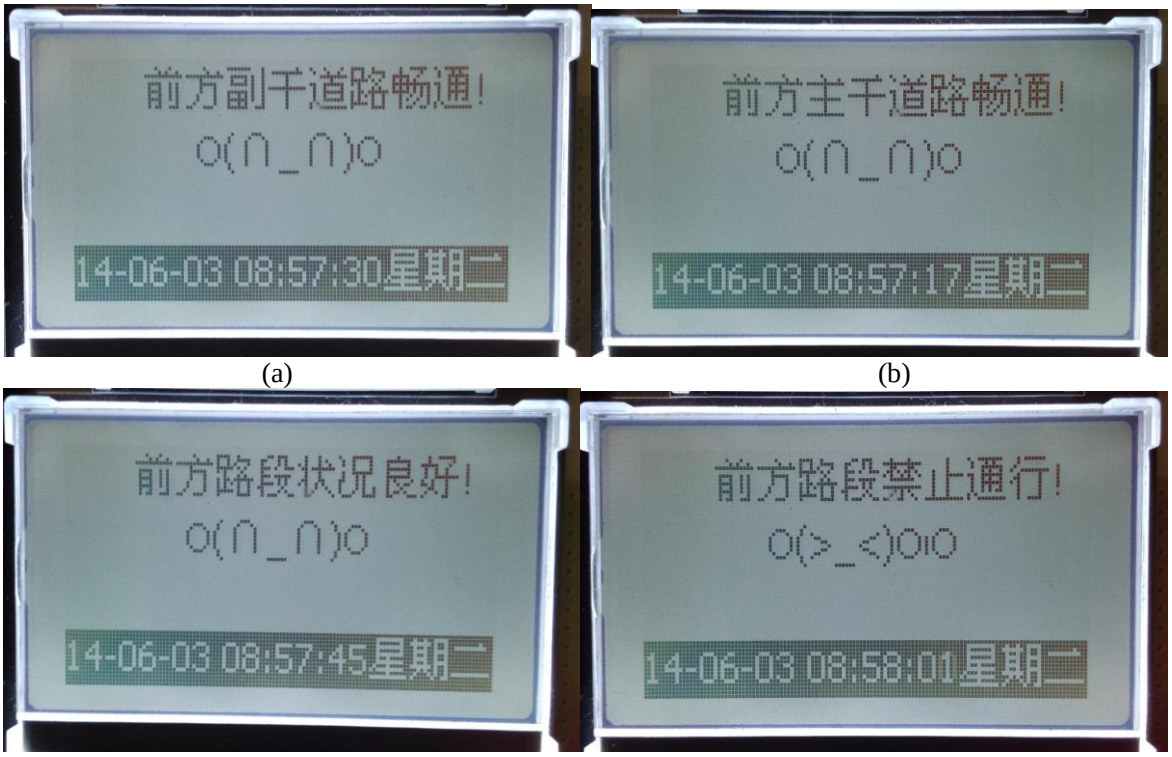
(a)　　(b)




(c)　　(d)

图 6.3　路侧交通控制系统的调试情况

图中，(a)，(b)，(c)，(d)的结果分别对应的是 start1，start2，start3 和 stop 信号的初始状态，LCD 路侧引导标识的调试与之类似，其结果如图 6.4 所示。




(a)　　(b)

(c)　　(d)

图 6.4　LCD 路侧引导标识的调试情况

LCD 屏的显示信息中，我们加入了一些表情字符符号，还使用了实时时钟显示当前时间，由于实时时钟中没有加入电池，不能记忆，所以时钟信号需要提前编程设置好，且不能断电。

## 6.3　智能路侧系统网络通信模块的调试

智能路侧系统网络通信模块是智能路侧系统非常重要的一部分，没有通信部分的设

计，智能路侧系统的控制也就无从谈起。此部分调试主要分为两部分进行，一部分是对智能路侧系统与车载系统的通信进行调试，另一部分是对智能路侧系统与控制中心的通信进行调试。

### 6.3.1 智能路侧系统与车载系统的通信调试

在进行调试之前，由于安卓手持设备并不是实际的车载系统，所以我们需要将车辆信息预先存储到手持设备上，然后再给手持设备下载一款 TCP 调试软件，此软件能够方便我们与智能路侧系统建立可靠的网络连接，同时还需要一款无线 wifi 设备，因为在车载系统上使用有线通信是不现实的，其他无线通信方式相比于 wifi 有所不足。本实验中无线 wifi 名称是“Dlink-150”，路由速度 100Mbps，FPGA 的接入 IP 地址设置为 192.168.0.183。首先将手持设备连接上此 wifi 热点，然后打开 TCP 调试软件，输入 FPGA 的 IP 地址和它们之间通信的端口号，其设置如图 6.5 所示。


图 6.5 手持设备的连接

设置完毕点击连接，这样就完成了 FPGA 与手持设备的连接。之后，就可以将车辆信息传输给 FPGA，FPGA 获取车辆信息并存储到数组中。

本系统使用 4 部手持设备模拟 4 个车载系统，分别命名为车辆 1，车辆 2，车辆 3 和车辆 4。其模拟终端界面相同如图 6.6 所示。


图 6.6 手持设备的终端界面

图中“车辆信息”按钮存储了车载系统的信息，包括车牌、车速、位置等，“呼救”按钮的设置是针对车辆遭遇交通事故或者此车辆是紧急车辆而预设的，当点击此按钮

时，就会将紧急信息发送给 FPGA，FPGA 会立刻将此信息发送给笔记本电脑模拟的控制中心，控制中心得到信息后会做出回应，FPGA 和控制中心的通信调试在下一节介绍。

### 6.3.2　智能路侧系统与控制中心的通信调试

在控制中心处，首先需要进入登陆界面，正确输入用户名、密码进行身份验证，如图 6.7 所示。


图 6.7　控制中心登陆界面

点击“登录”后，进入控制中心的工作环境，如图 6.8 所示。

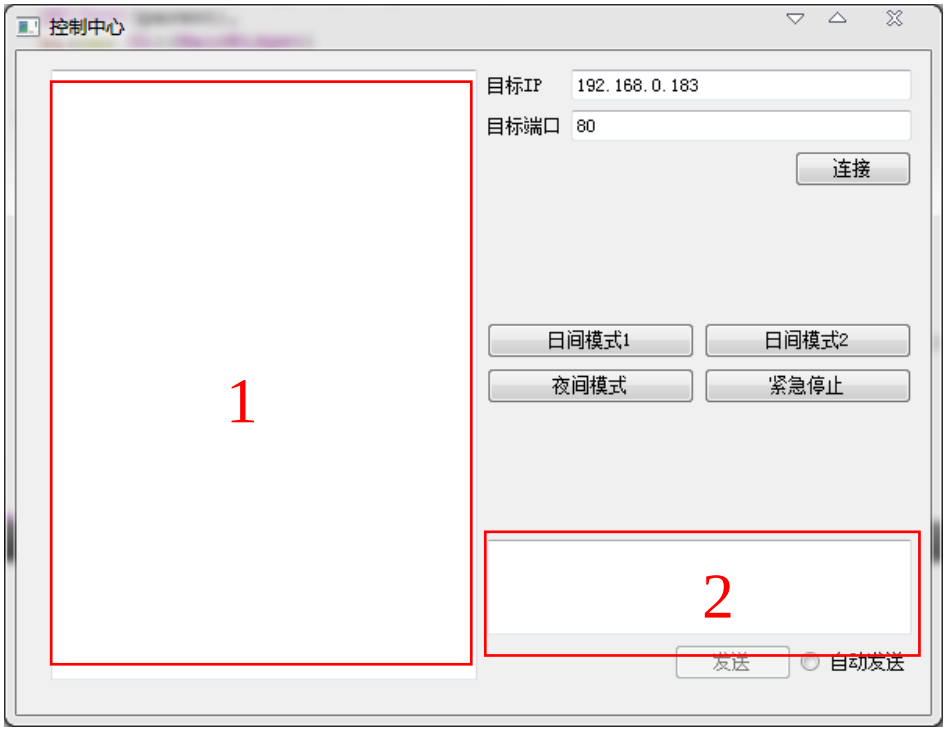



图 6.8　控制中心工作界面

输入 FPGA 的 IP 地址和它们之间的端口号，点击“连接”，就建立起了 FPGA 与控制中心的通信。这里，目标端口不能跟手持设备设置的一样。此时，点击“自动发送”就可以在方框 1 中接收到 FPGA 从 4 个手持设备上接收到的各个车辆信息，还有接收到的当前时间，如图 6.9 所示。其中还预留了一块信息输入框，为图中的方框 2，留作备用。

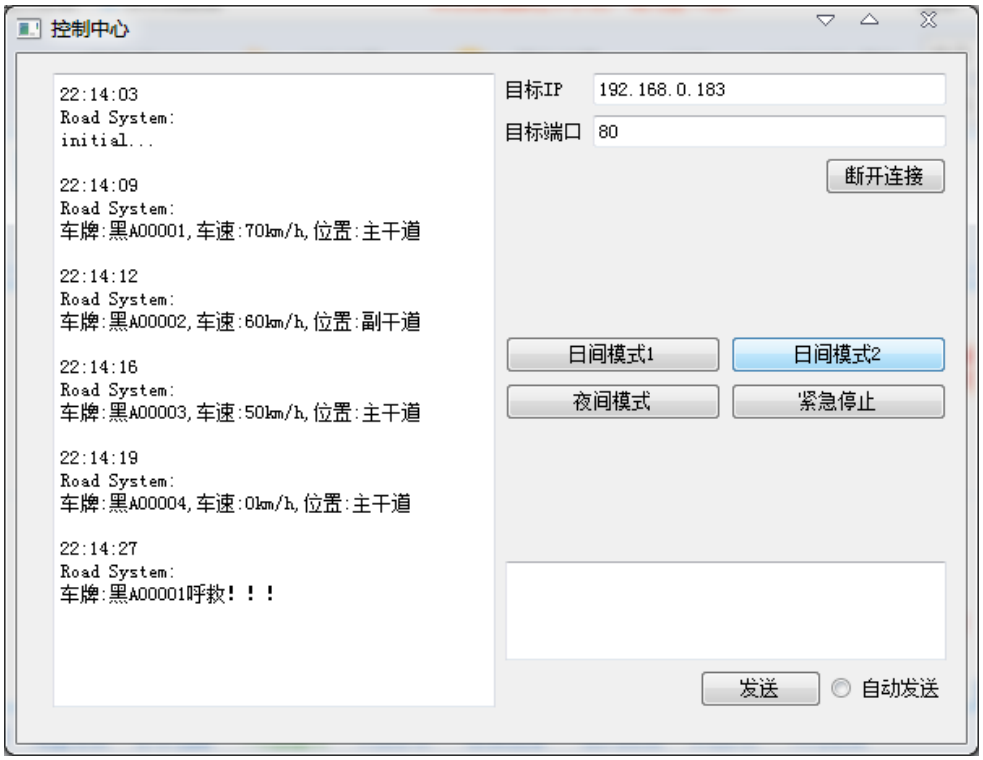



图 6.9　车辆信息的接收界面

根据上图，可以看出先前主干道车流量大于副干道车流量，所以我们点击“日间模式 1”，触发 start1 信号，其 FPGA 控制的模拟交通系统的运行结果如图第 6.2 节图 6.3(a)所示。之后，图中出现车辆 1 的呼救信息，所以我们需要点击“紧急停止”，此时模拟交通的运行结果如图第 6.2 节图 6.3(d)所示。

## 6.4　视频采集与图像信息公告模块的调试

实际中，摄像头可以架设在十字路口上，图像信息的显示模块可以与路侧引导标识架设在一起，提前告知驾驶员前方路口的交通状况。在实验中，我们通过 OV7670 采集视频，然后将采集到的视频通过 VGA 实时地显示出来，其中每采集一帧图像，都会将图像信息缓存到 SDRAM，我们可以将 SDRAM 中的图像再取出来做后续的处理。视频的 VGA 显示测试图如图 6.10 所示。

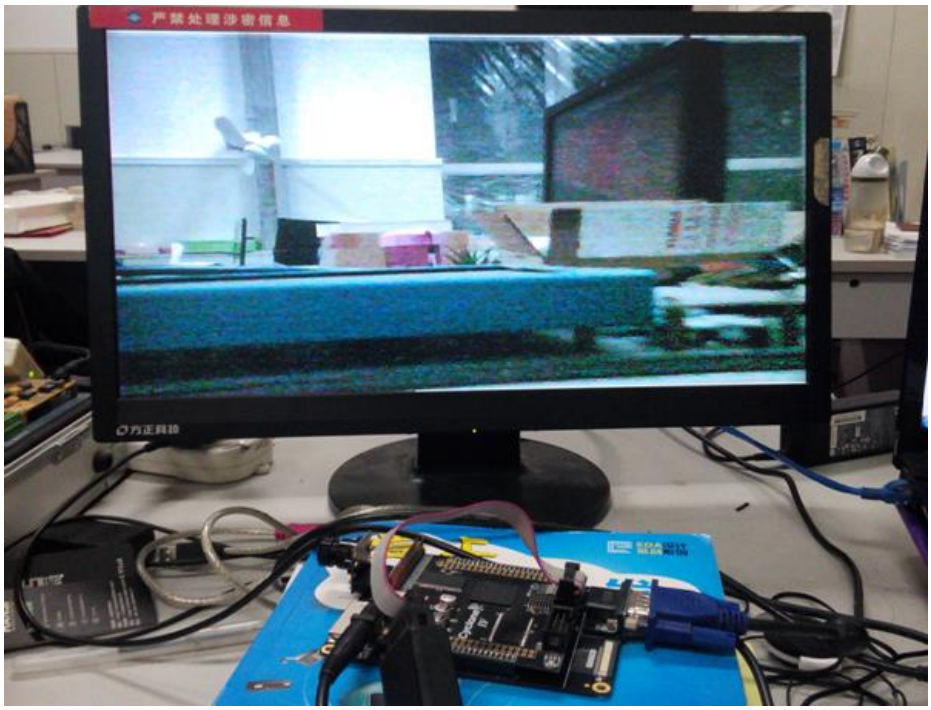



图 6.10　视频采集显示

图中，图像实时性很好，显示较为流畅，画面质量依据 VGA 显示的分辨率来看，效果良好，后续还可以继续对图像进行处理，比如使用 JPEG 压缩编码实现网络传输等等。

## 6.5　系统调试结果分析

经过上述调试过程，系统各部分模块运行正常，交通信号灯、LCD 显示屏能正常工作，网络通信状况良好，wifi 连接稳定，通信距离无遮挡的情况下大于 50 米，视频采集显示部分实时性好，图像流畅，延迟不超过 1s，图像分辨率是 640×480，刷新频率 60Hz。

## 6.6　本章小结

本章首先介绍了系统的整体调试方案，并在实验室搭建了模拟现场调试环境，然后详细阐述了系统调试的几个部分，分别是智能路侧交通系统，系统的网络通信模块和视频采集与图像公告模块，之后对系统的整体调试进行结果分析。测试结果说明本系统能够稳定运行，很好地满足了课题的设计要求。

# 结　论

车路协同系统在智能交通领域中占有重要地位，智能路侧系统作为实现车路协同的关键技术之一，对于解决当前交通问题有着重要的意义。本文以 FPGA 作为开发平台，采用 SOPC 技术，TCP/IP 传输技术，图形化界面编程技术，I2C 总线技术以及 SDRAM 的乒乓操作技术设计实现了基于网络远程控制的智能路侧系统，其中本文取得的主要研究成果如下：

（1）基于 FPGA 上的逻辑资源，使用 Verilog HDL 语言设计了控制交通信号灯的模块，同时利用 LCD 液晶屏作为路侧引导标识，动态地显示相关路口的路况信息，引导车辆最优化行驶。实验结果显示，交通信号灯运行正常，路侧引导标识工作情况正常，且都能根据控制中心发送的控制信息合理地选择不同的模式。

（2）采用 SOPC 技术在 FPGA 上构建了软核并添加了 SPI 总线组件，通过移植精简的嵌入式 TCP/IP 协议栈，实现了 FPGA 路侧端与控制中心和车载系统的信息传输交互。实验结果显示，FPGA 路侧端与控制中心和车载系统的网络信息传输对接良好，信息传输正确，没有丢失现象，传输速度比较快。

（3）应用了图形化界面编程技术，采用跨平台 Qt 界面编程实现了控制中心简单操作的人机交互界面。实验证明，此方法设计出的界面，美观且操作方便。

（4）采用 I2C 总线对视频采集传感器 OV7670 进行初始化，同时采用乒乓操作技术加快了图像信息的读写速度，提高了 VGA 显示的实时性。实验结果显示，视频图像信息显示比较清晰，并且实时性很好，驾驶员可以根据视频传感器采集到的实时路口图像信息提前选择行车路线。

本文设计达到了预期目标，完成了阶段性的任务，但同时也有一些不足，对本文进一步的工作展望主要有以下几点：

（1）为了达到控制中心远程接收路侧系统采集到的图像信息并对其分析，需要对图像进行压缩，降低对网络带宽的要求。

（2）在交通控制系统中可以引入自适应控制算法结合网络远程控制协调控制，达到对整个交通系统的精确控制，同时也减轻了控制中心部分的工作压力。

随着汽车保有量的不断增加，交通问题的越发严重，智能路侧系统的研究会越来越深入，更多的智能路侧系统的设计方案将会被提出。

# 参考文献

[1] 杜晓琳. 车路协同系统中通信管理系统设计及其功能仿真[D].北京交通大学,2013.

[2] 柴少丹. 车路协同系统功能测试与评价方法研究[D].武汉理工大学,2013.

[3] 王云鹏. 国内外 ITS 系统发展的历程和现状[J]. 汽车零部件,2012,06:36.

[4] 王国锋,宋鹏飞,张蕴灵. 智能交通系统发展与展望[J]. 公路,2012,05:217-222.

[5] 周户星. 车联网环境下交通信息采集与处理方法研究[D].吉林大学,2013.

[6] Michael Chapman. Using Vehicle Probe Data to Diagnose Road Weather Conditions-Results from the Detroit IntelliDriveSM [C]//Washington,D.C,State Transportation Departments,TRB 2010 An-nual Meeting.2010:1-16.

[7] Hyungjun Park,Brain L.Smith. Investigating Benefits of IntelliDriveSM in Freeway Operation-Lane Changing Advisory Case Study [R]. TRB 2010 Annual Meeting , Washington D.C : State Transportation Departments.2010.

[8] 范子健. 车路协同系统测试案例生成方法及其仿真实现[D].北京交通大学,2013.

[9] 陈超,吕植勇,付姗姗,彭琪. 国内外车路协同系统发展现状综述[J]. 交通信息与安全,2011,01:102-105.

[10] Ashwin Amanna.Overview of IntelliDrive/VehicleInfrastructure Integration (VII).[R]. 2009.

[11] G.A Giannopoulos. The application of information and communication technologies in transport[J]. European Journal of Operational Research . 2003 (2).

[12] 吴涛. 车路协同智能路侧系统关键技术研究[D].山东理工大学.2012,4

[13] 周春蕾. 基于 FPGA 技术交通灯智能控制系统的研究[D].河北科技大学,2010.

[14] 夏宇闻. Verilog 数字系统设计教程（第三版）[M].北京航空航天大学出版社, 2013.

[15] 陈欣波. Altera FPGA 工程师成长手册[M].清华大学出版社,2012.

[16] 吴厚航. 爱上 FPGA 开发——特权和你一起学 NIOS II[M]. 北京航天航空大学出版社,2011.

[17] 吴厚航. 深入浅出玩转 FPGA[M].北京航天航空大学出版社,2010.

[18] 杨立才. 城市道路交通智能控制策略的研究[D].山东大学,2005.

[19] 卫小伟. 城市智能交通控制系统研究与设计[J]. 现代电子技术,2010,17:189-192.

[20] 沈晟. 基于 FPGA 的交通灯控制系统的设计与实现[D].昆明理工大学.2008.

[21] 王海霞,武一. 基于 SOPC 的 LCD 显示模块的设计与实现[J]. 液晶与显示,2012,04:508-514.

[22] 祁树胜. SPI 接口以太网控制器 ENC28J60 及其应用[J]. 微计算机信息,2006,23:266-268.

[23] Shilpa S. Chavan (Walke), Dr. R. S. Deshpande, J. G. Rana2. DESIGN OF INTELLIGENT TRAFFIC LIGHT CONTROLLER USING EMBEDDED SYSTEM [J]. Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09.2009: 1086.

[24] M.F.M.Sabri, M.H.Husin, W.A.W.Z.Abidin, K.M.Tay, H.M.Basri. Design of FPGA-Based Traffic Light Controller System [J]. 2011：114-118.

[25] 王世隆. 基于 FPGA 的嵌入式交通信号控制机设计[D].哈尔滨工业大学.2013.

[26] Nath S, Pal C, Sau S, et al. Design of an FPGA based intelligence traffic light controller with VHDL [C].Radar, Communication and Computing (ICRCC), 2012 International Conference on. IEEE, 2012: 92-97.

[27] 谢喆. 基于 FPGA 的 TCP/IP 数据通信的设计与应用[D].武汉科技大学,2011.

[28] 张丁丁,孙志毅. TCP/IP 协议栈的实现方法[J]. 工业控制计算机,2013,09:35-36.

[29] 张凯. TCP/IP 网络通信协议的实现与探讨[J]. 网络与信息.2008(09):24-25.

[30] Forouzan B A. TCP/IP protocol suite[M]. McGraw-Hill, Inc., 2002.

[31] 整理：以太网 -haizhongyun 的专栏 - 博客频道 -CSDN.NET http://blog.csdn.net/haizhongyun/article/details/7414965

[32] 霍亚飞.Qt Creator 快速入门[M].北京航空航天大学出版社,2012.

[33] BehrouzA.Forouzan,SophiaChungFegan.谢希仁等译[M].TCP/IP 协议栈.清华大学出版社,2006.

[34] Omni Vision Serial Camera Control Bus (SCCB) Functional Specification. Version: 2.2[Z]. 2007.

[35] 李德明,韩剑.基于 OV7670 的图像采集及显示系统设计[J].仪器仪表学报.2010（8）：30-33.

[36] 杨光耀.基于 FPGA 的图像采集及处理系统研究[D].内蒙古大学.2014.

[37] 史运锋. 基于 Nios II 软核的嵌入式以太网设计[D].南京理工大学.2009.

[38] 杨威. 基于 FPGA 的以太网和串口数据传输系统设计与实现[D].哈尔滨工程大学.2013.

[39] Nath S, Pal C, Sau S, et al. Design of an FPGA based intelligence traffic light controller with VHDL[C].Radar, Communication and Computing (ICRCC), 2012 International Conference on. IEEE, 2012: 92-97.

[40] Ming Z, Ting L H. The design of the displaying system based on the SOPC embedded chips[C].Electric Information and Control Engineering (ICEICE), 2011 International Conference on. IEEE, 2011: 5477-5480.

[41] 张龙滨.一种基于 FPGA 的实时视频采集与远程传输系统[J]. 器件与应用.2011:45-47.

[42] 周帅. 基于无线通讯的 DCS 系统研究与实现[D].河北科技大学,2013.

[43] 刘成岩. 基于 FPGA 的图像采集处理系统设计[D].哈尔滨工程大学,2012.

[44] 肖静娴,戴亚文. 基于 FPGA 的异步 FIFO 缓存设计[J]. 电子测量技术,2009,11:92-94.

[45] Adam Dunkels. Design and Implementation of the LwIP TCP/IPStack[A].Swedish Institute of Computer Science，2001:45.

[46] Jiayi Zhu,Peilin Liu,Dajing Zhou. An SDRAM Controller Optimized for High Definition Video Coding Application [J].ISCAS. 2008:518-352.

[47] 李超纯,李亚兰,李小飞,杨特育,李志扬. 基于视频解码芯片与 CPLD 的实时图像采集系统[J]. 电子技术应用,2007,02:41-42.

[48] 高瑜. 基于 FPGA 的交通信号灯的智能控制系统[D].河北工业大学.2012.

[49] 王维松,章伟.基于 FPGA 的一种智能交通红绿灯设计[J].计算机应用与软件.2013(1):200-202.

[50] 彭坚,何渝. 基于 FPGA 的交通信息采集系统的设计与应用[J].计算机应用与软件.2011(1):82-84.

[51] Donald G. Bailey. Design for Embedded Image Processing on FPGAs[M]. Wiley-IEEE Press,2013.

[52] Jasmin Blanchette, Mark Summerfield. C++ GUI Programming with Qt 4 [M].Prentice Hall,2008.

[53] 姜旭,朱灿焰. 视频处理技术在智能交通系统的应用[J].通信技术,2010,43(1):99-101.

[54] 王龙飞.基于 FPGA 的视频采集系统设计[D].西安工业大学，2013

# 攻读硕士学位期间发表的论文和取得的科研成果

[1] 康维新，李定洋. 基于 FPGA 的交通系统远程网络控制设计[J]. 电子科技(已录用).

# 致　　谢

在整个硕士研究生学习阶段即将结束，在这篇论文即将完成的时候，我要向所有在此期间帮助和支持我的老师和朋友们表示我最真诚的感谢。

首先我由衷的感谢我的导师康维新教授，他谦和大度、学术严谨、学识渊博，在研究生期间，对于我的课题给予了很多指导和帮助，使我对课题的研究更加深入，受益良多。感谢康老师一直以来对我的谆谆教诲，使我提高了专业水平和能力、磨练了性格，同时也使我明白了很多做人的道理，这些都将使我受益终生。康老师的授业恩情，我将永远也不会忘记！

感谢实验室兄弟姐妹对我的关心和帮助，是你们的鼓励和帮助使我深深地感受到了实验室这个大家庭的温暖，感谢你们给实验室带来的欢乐。

感谢我的同学和朋友，在我遇到困难的时候是他们给予了我最真心的鼓励和支持，让我拥有了战胜困难的勇气和决心。

最后我要特别感谢我的父母和亲人，你们对我的关爱、理解和支持是我前进的最大动力，你们是我最坚实的后盾，正是有你们的支持，我才能坚定地一路走下去。

感谢评审阅读本文的各位专家学者以及参考文献的作者。

谨向所有给予我关怀、支持和帮助的人致以我最衷心的感谢，谢谢你们！