分类号_____ 学号 M201071892

学校代码 10487 密级_____

華中科技大學

硕士学位论文

基于软硬件协同处理的轮式机器人平台设计

学位申请人：周波

学 科 专 业：控制工程

指 导 教 师：何顶新 副教授

答 辩 日 期：2012.05.29

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree for the Master of Engineering

Design of a Wheeled Robot Platform Based on Hardware/Software Co-processing

Candidate : **Zhou Bo**

Major : **Control Engineering**

Supervisor : **Assoc. Prof. He Dingxin**

Huazhong University of Science & Technology

Wuhan 430074, P.R.China

May, 2012

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，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。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，　在_____年解密后适用本授权书。

　　不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月　日

摘　要

随着需求的不断增加，嵌入式系统已经发展得越来越复杂和多样化，如何对性能、成本、可重用等多种指标进行权衡和优化，是嵌入式系统设计领域的挑战性问题。而轮式机器人在交通、工业的应用日益增多，为了解决轮式机器人嵌入式系统开发的指标优化问题，基于软硬件协同处理架构，研究了轮式机器人的异构多处理器平台化，提高了系统性能与设计效率。

层次参考模型分为三层：高性能处理层、实时控制流层、数据平面层。系统利用 FPGA（现场可编程门阵列）处理密集数据流，利用 ARM 处理复杂控制流并扩展功能到高性能处理层。实时控制流层与数据平面层之间的拓扑关系分为预处理器、后处理器、协处理器三类，并采用松耦合方式通信，保证两者之间数据交换的高效和接口透明。实时控制流层通过有线网络、串口等多种方式与高性能处理层(PC、智能手机等)通信。基于机器视觉的自主导航、基于人工视觉的遥控导航是移动机器人领域两类热点应用技术。将上述软硬件协同处理架构应用到针对机器视觉的移动机器人领域，对控制系统进行适当的软硬件划分，FPGA 系统将分为数字图像处理预处理器、图形显示后处理器、运动控制协处理器。在该平台 FPGA 内部设计了通用的软硬件通信接口和 IP 扩展方式，使用同步 EMC 总线作为片内总线。数字图像预处理包含全局动态阈值分割、细化、压缩等；运动控制协处理器采用 A3 原则实现了双闭环直流调速系统；液晶后处理器则实现了软硬件图像动态回传。

所设计的软硬件协同处理架构是一个理想的嵌入式技术研究和验证平台。该系统不仅有较高的自主导航横向定位精度和优良的动力性能，还能利用无线智能终端实现远程操作和监控，取得了良好的运行结果。

关键词：软硬件协同处理，轮式机器人，动态阈值分割，现场可编程门阵列

Abstract

The optimization of metrics, such as performance, flexibility, versatility and cost, is a problem embedded system development has to face. On the other hand, as an important developing direction to solve traffic and industrial problems currently, intelligent mobile robots is attracting people's attention. In order to solve the problem of optimization of metrics in developing wheeled robot embedded system, a layered asymmetric multiprocessing platform is proposed in our design, and applied for a wheeled robot, with the Hadware/Software(HW/SW) co-processing to improve design efficiency.

The system is divided into 3 layers: high-performance process layer, real-time control flow layer and data plane layer. FPGA(Field Programmable Gate Array) in data plane layer is used for computationally intensive tasks, while ARM microcontroller equipped with RTOS is used for complex real-time control and application extension to the high performance layer. A lose couple channel are adopted to make the communication effective and flexible between data plane layer and real-time control layer. The role of FPGA in data plane layer can be a pre-processor, post-processor and co-processor of the ARM microcontroller. Different ways, such as wireless network and UART(Universal Asynchronous Receive and Transmission), are adopted to connect the real-time control layer and high-performance layer, which can be a PC or a smart phone. Autonomous navigation based on machine vision and remote control and monitor based on human vison are popular technology to realize an intelligent mobile robot. Applying the proposed HW/SW co-processing architecture to an autonomous navigation robot based on machine vision and partitioning its control system's HW/SW, the FPGA is divided into image pre-processor, DC motor control co-processor, LCM post-processor etc. User defined IP(Intelligent Property) in FPGA can be easily attached to local bus and communicate with the CPU through a universal on-chip mechanism. Dynamic threshold, image thinning and compress are implemented in the image

pre-processor. The A3 principle is adopted in DC motor double closed loop control. Effective display on the dot matrix LCM is implemnted.

The HW/SW co-processing architecture and co-design methodology tremendously boost the system performance and accelerate the design. The robot can not only autonomously drive along the line marker, but also be remotely controlled and transfer video through an Android smart phone. The mobile robot is just one application case of this platform which also can be applied in many fields of embedded systems and industrial control area.

Keywords: Hadware/Software co-processing　　Wheeled robot
Dynamic threshold segmentation　　Field programmable gate array

目　录

1 绪 论

1.1 研究背景和意义

1.1.1 嵌入式系统的软硬件协同设计

随着近年来需求的不断增加，嵌入式系统已经发展得越来越复杂和多样化，而研发成本和产品成本也随之急剧增长。如何对多种指标进行权衡和优化，是嵌入式系统设计领域的挑战性问题[1]。传统嵌入式系统设计过程中，系统在一开始就被划分为软件和硬件两大部分，软硬件之间的交互受到很大限制，设计修改困难、研发周期不能保障，系统研发成本较大[2]。为了解决这一尖锐矛盾，基于软硬件协同设计（HW/SW Co-design）的平台化方法应运而生[3]。基于软硬件协同设计平台化方法中，设计方法转向软硬件协同的系统级设计，这样就能较好地优化和权衡性能、成本等各种设计指标，并大大提高设计效率[4, 5]。

然而软硬件协同设计技术将分开的软硬件融合进系统级设计，共同的接口设计给设计人员带来了新的问题，标准化的表示方法和恰当的评估方法是需要解决的问题。为此，软件设计语言不断进化，而硬件抽象层次不断提高[6]，而基于可编程逻辑器件 FPGA 的 SOPC（System on Programmable Chip）技术作为一种系统级解决方案，能较好地实现软硬件协同设计。

从 SOPC 的设计和实践中，人们认识到设计方法的革命就是要完成一个转变，以功能设计为基础的传统流程转变为以功能组装为基础的全新流程[7]。基于软硬件协同设计平台化方法中，系统设计不可能一切从头开始，而是建立在较高的基础之上，利用已有的平台进行设计重用[8]。比如，FPGA 设计越来越倾向于以 IP 为中心，在 FPGA 中嵌入软核或硬核处理器、DSP 模块、存储器等。

基于 SOPC 技术设计一种高性能、灵活性、通用化、低成本的软硬件协同处理平台，并应用到工业控制领域，是本文的核心基础。

1.1.2 小型轮式机器人系统

轮式机器人是应用广泛的一种移动机器人，运动可靠稳定，执行机构简单，广泛应用于军事、工业等各个领域。而轮式滑动转向移动机器人对于较空间受限场合具有良好的灵活性[9]，其为全固定多轮驱动，靠车轮间的速度差动造成的车轮的滑动来实现车体转向[10]。

自主导航轮式机器人的概念是相对传统轮式机器人提出的，是指可以自动识别道路标志路径并进行自动驾驶的车辆。自主导航技术是智能轮式机器人系统研究的重点，是智能化的关键体现[11,12]。目前常见的小型轮式机器人自主导航定位方法可以分为以下三类[13,14]：

（1）相对定位：又称航位推进(Dead Reckoning)，利用内部传感器，依据运动学模型，通过测量机器人相对于初始状态的变化量来推测当前的航位。主要包含的传感器有：

a) 视觉：获取环境的视觉图像信息。静态信息丰富，但同时数据处理量大，系统实时性易受到限制；容易受光线等环境影响。因此需要做图像处理。

b) 超声波：采用目标对声波的反射来发现障碍物并测定其位置。在空气中传输损耗大，而功率太大会造成污染，所以主要获取近距离信息。多用于自动倒车、泊车系统。

c) 电磁：利用特定频率的甚低频交变磁场为引导，可以有效避开环境永磁场、工频、高频电磁场的干扰。但不适合于复杂环境，线路短。

（2）绝对定位：采用导航信标或全球定位系统进行定位。导航信标的建设和维护成本较高。GPS 的信号易被遮挡，易受定位精度和天气影响。适合室外开阔的环境，在楼房林立的城市或房屋内难以实现定位。

（3）组合定位：是针对单一定位方法的不足，采用的基于航位推测与绝对信息矫正相结合的定位方法。

而远程操控的轮式机器人则能利用人类对复杂环境的认知，通过实时回传的视频进行远程操作和监控，完成更为复杂的任务，适用在人类难以到达或较为恶劣的场合，如排爆、火星车探险、空气污染环境下的任务执行等。

本文采用了视觉定位的方式实现小型轮式机器人的自主导航和远程操控。

1.1.3 研究意义

近年来，疲劳驾驶、酒后驾驶等导致交通事故频繁发生；而一些特殊场合如化工现场、隧道矿井等迫切需要远程辅助驾驶的支持；日益繁重的货物运输让仓库码头等物流中心、公园公交等固定线路的人工费用居高不下，效率低下。传统车辆控制方式的局限性日益明显。而具有障碍物检测、智能避障、自主导航、远程遥控和视频回传等功能的智能移动机器人成为解决上述交通困境的一种方案。

基于机器视觉定位作为智能移动机器人的一种导航方式，在许多应用场合具有很大优势。而基于远程操作与视频监控的小型移动机器人也在军事、工业等场合应用广泛。为了充分适应各类环境并满足开发可行性，需要较丰富的嵌入式处理技术。因此，对轮式机器人的嵌入式系统设计研究不仅具有一定的学术意义，也同时具有重要的实际应用价值。

利用嵌入式 CPU 处理上层复杂控制流并扩展功能到应用层终端、利用 FPGA 处理密集数据流，是本平台最显著的特点。FPGA 相对 CPU 的特点在于高速并行计算，因此适合于处理底层密集数据流。CPU 相对硬件的特点是复杂函数计算，而实时嵌入式操作系统的移植可以丰富系统的应用及加强实时性，结合丰富的协议栈及应用支持，系统平台功能可以得到较大扩展，能满足诸多小型嵌入式系统应用需求。使用通用协议实现高性能处理层智能终端与实时控制流层通信，将大大提高系统应用扩展能力。

在移动机器人领域，重配置和重用化技术一直有大量的研究。而随着 SOPC 技术作为一种系统级解决方案的发展，使用基于 FPGA 的 SOPC 技术来实现系统功能成为了机器人平台研究的热点，通过 HDL 编程、数据流图和系统级建模来实现所需的逻辑功能，为机器人可重用、重配置提供了一种新的方法。FPGA 负责实现连接各类传感器和执行机构的逻辑电路功能和必要的信号处理，只需编写不同的硬件语言程序，即可实现所需的电路功能、处理相应的传感器和执行机构数据[15]。这类异构系统中，CPU 作为主要器件启动，而后对 FPGA 进行配置，完成系统初始化；在需要对机器人进行功能变更时，或者系统底层设计进行了较大修改，利用 CPU 对 FPGA 进行在线可重配置的优势就充分体现出来，系统研发成本、修改成本、产品上市周期将大大降低。

1.2 国内外研究现状

1.2.1 软硬件协同处理技术研究

使用软硬件协同异构多处理系统能有效降低软件处理负载。M.D. Edwards 等人使用软硬件协同设计技术加快了软件算法，通过 PCI 总线设计了两类 FPGA 与通用处理器的通信，加速比达到 1 ~3 倍[16]。Usman Ali 和 Mohammad Bilal Malik 设计了一种基于软硬件协同处理的实时视频跟踪系统的嵌入式解决方案，通过基于微控制器、协处理器的架构，极大地加速了软件算法[17]。

软硬件协同处理架构的关键是异构处理器间的通信方式。一些研究人员提出了有效的主处理器与 FPGA 通信框架。David Andrews 等对 CPU-FPGA 架构讨论了编程模型[18]。文献[19]中的作者为 CPU-FPGA 耦合架构的相互调用设计了一种间接（松耦合）接口和直接（紧耦合）接口，前者是通过中端级别的双口 RAM 和寄存器，后者是通过修改的软核处理器核指令级别的总线。后者能有效消除前者的时间消耗。M.D. Edwards 等为“硬件”调用/返回机制设计了两种类似的架构[16]。

国内外不少研究学者将软硬件协同处理技术应用到了嵌入式系统设计中。I. del Campo 等人通过 SOPC 软硬件技术实现了一个高效的自适应模糊-神经网络推理系统[20]，包括两种不同的方法：用于离线训练的高性能并行架构和适合在线参数调整的流水线架构，软硬件通信带宽达到 100MByte/s。文献[21]结合可重用方法与 SOPC 技术实现了一种嵌入式掌纹识别系统，可以支持二维条码扫描、RFID 卡读写、门禁控制等功能，主要包括 NIOS-II 软核处理器、图像采集、人机接口、通信接口等。但系统主要数据处理在片内软核中完成，没有充分发挥可编程逻辑的高速并行数据处理特性，逻辑部分的时钟性能也不高（58MHz）。

可以看出，基于 SOPC 技术的软硬件协同处理平台对可编程硬件部分开发要求较高，因为其直接涉及密集数据处理，对系统带宽、实时性影响较大，因此本文将在合理的软硬件划分基础上着重进行 FPGA 部分的性能提升的优化，以期达到较好的软硬件协同处理系统指标。

1.2.2 轮式机器人研究

1995 年，美国卡耐基梅隆大学进行了全长 5000 km 的高速公路无人驾驶试验，自主导航的 Navlab-5 系统完成了方向控制，但油门和刹车由人控制，没有超车实验[22]。

2005 年，“斯坦利”号自主跑完了美国内华达州沙漠中长达 213 公里的全部赛程，耗时 6 小时 52 分钟[23]。由斯坦福大学、美国大众汽车公司联合设计的“斯坦利”号，由一辆大众途锐 SUV 改装而成。赛车根据 GPS、6 自由度惯性测量和车轮速度传感器综合评估车辆姿态，使用四个激光测距仪、单雷达系统、一台立体摄像机和一套单眼视觉系统来感知周边环境。整个处理系统共使用了 7 台奔腾-M 计算机。

2011 年，国防科技大学自主研制的无人车红旗 HQ3，完成了从长沙到武汉高速公路全程 286 公里无人驾驶实验，历时 3 小时 22 分钟，途经雾、雨等复杂天气，达到世界先进水平[24]。实验中，无人车自主超车 67 次，人工干预里程占自主驾驶总里程不到 1%，低于国际上无人车领域 3%的无人驾驶标准。其通过车两侧的前视摄像头感知道路信息，通过计算机进行控制，能实时处理岔道、斑马线和虚线；对车体姿态变动，自然光照变化及树木、路桥阴影都具有较强的自适应力，最高时速达到 150km/h。

2011 年，美国 Brookstone 公司开发了一款利用苹果 iOS 设备[25]控制的轮式机器人 Rover Spy Tank[26]。前端配备 30 万像素摄像头，可实时向 iOS 设备传送视频；摄像头旁边装备红外夜视镜，微弱光线下能看清周围情况；iOS 设备可以设定路线，坦克会按照设定路线自行移动；使用 WiFi 通信，室内操控距离达 20~30 米。

2012 年，谷歌公司的无人驾驶汽车获得了在美国公共道路上的行驶牌照。系统的核心是车顶上的激光测距仪，能够提供精细的 3D 地图数据，行驶时将实时的数据和谷歌数据库中记录的数据进行比较，将行人和路旁的物体分辨开来[27]。在汽车的前后保险杠上有四个雷达，用于探测周边情况。后视镜的附近有一个摄像机，以检测交通灯情况。另外还包括一个 GPS、一个惯性测量单元、一个车轮编码器。

1.3 FPGA 的软硬件协同开发方法

1.3.1 开发流程

（1） 软硬件协同设计流程

在 SOPC 设计中，为了既可缩短开发周期，又能取得更好的设计效果，要求使用软硬件协同设计技术。它让软件和硬件作为一个整体并行设计，在设计过程中，二者设计过程通过划分的接口相互作用，体现出协同的特点[28, 29]。

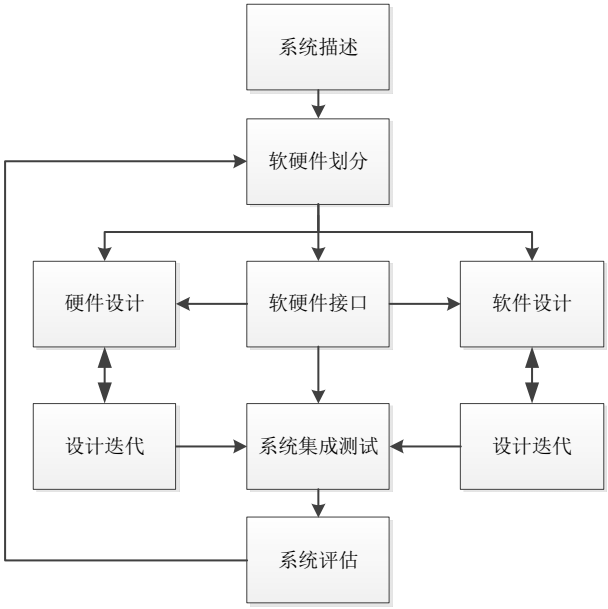


图 1.1 软硬件协同设计流程

（2） FPGA 开发工具对开发过程的影响

FPGA 的软硬件协同开发过程可以充分利用各类开发工具的优势，对算法进行各个层次的仿真测试，从而加快开发进度。Matlab 等高级建模和仿真工具使用的高级语言和流程图具有比硬件描述语言更好的抽象建模能力、数据流描述能力、仿真功能，而基于 Xilinx System Generator(XSG)和 Matlab Simulink 的软硬件协同设计、实现与仿真流程正成为数字信号处理领域一大热门设计方法。

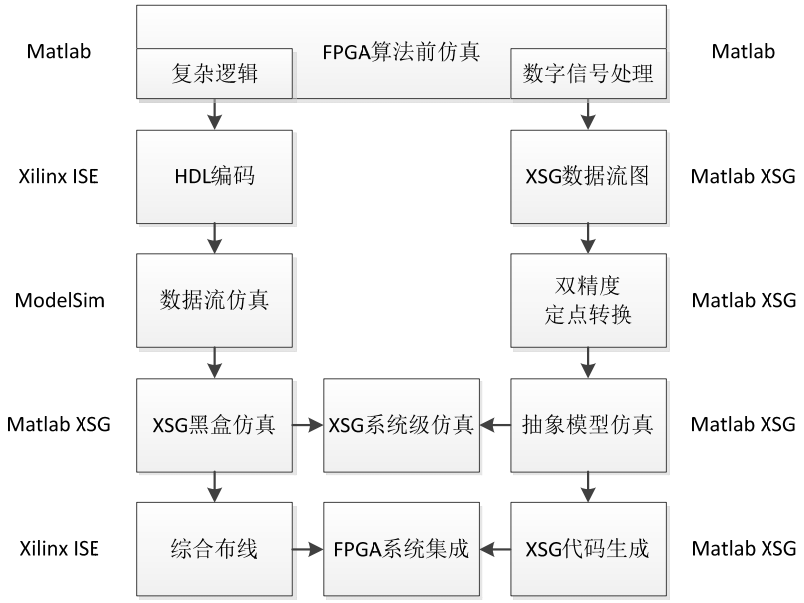


图 1.2　软硬件协同处理平台 FPGA 部分开发方法

对于逻辑复杂的软硬件协同开发仿真流程，FPGA 部分将分为前仿真、实现、后仿真等主要环节。前仿真在 C 语言或 Matlab Simulink 环境下进行基本算法验证; 实现环节可以在 ISE 环境下进行 HDL 编码，并进行数据流同步基本仿真；后仿真环节则利用 Matlab Simulink 建立外围仿真环境，导入原始数据并进行基本处理，然后利用 System Generator 的黑盒模型导入设计好的 HDL 文件，进行高级模型仿真，完成软件硬件协同设计与仿真过程。

对于数字信号处理的软硬件协同开发方法则更为灵活，实现环节是在 System Generator 图形界面环境下搭建数据流模型，可以充分利用可重用的 IP 和高效率的定点转换工具，并进行高级模型仿真，通过代码生成工具将直接得到中间网表，后仿真过程可以简化，而且开发与仿真环境在 Matlab 一个工具下即可完成，从而大大加快数字信号处理开发流程。

值得注意的是，在环境下可以结合上述两类仿真流程，搭建 HDL 黑盒和定点 XSG 数据流图的整体系统而进行系统级仿真，从而尽可能地实现系统仿真与实现结果的逼近。

1.3.2　设计原则

（1）　软硬件划分

软硬件协同设计的关键环节是软硬件划分。对于异构多处理器系统，为充分发挥处理与

计算平台层次结构的优势，软硬件划分的选择尤为重要。以 DSP 与 FPGA 为例，按照如下图的选择原则实现算法，能较好地解决软硬件划分问题[30]。对于高复杂度、低时序约束算法，适合用 DSP 实现，而对于低复杂度、高时序约束算法，适合于用 FPGA 实现。

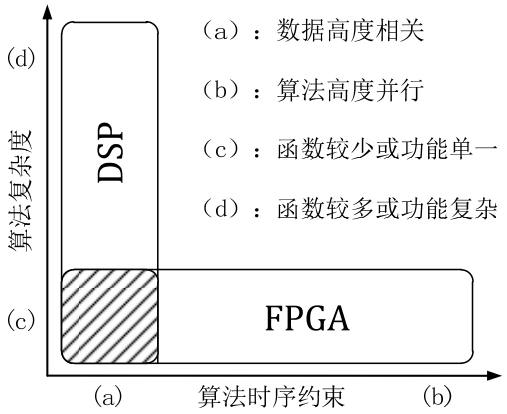


图 1.3　DSP 和 FPGA 的使用范围

（2）　A3 方法

对于比较成熟的算法在 FPGA 中实现，主要分为 HDL 状态机、XSG 数据流图两类，前者适合于复杂逻辑算法，而后者适合于数字信号处理。要想充分利用 FPGA 和开发工具的优势，必须在数字信号处理设计开发过程中遵循一定的设计方法。有三条主要原则：算法优化、模块化、实现的算法和选择的硬件最紧密的配合（Algorithm Architecture “Adequation” (A3)）[30]。A3 原则就是实现最优的执行时间和占用时间的最优配合，实现步骤如下：

第一步，建立数据流示意图（DFG）。

第二步，设计最优的 A3 结构。

1.4 论文安排

在分析国内外软硬件协同处理架构研究成果的基础上，对利用 FPGA、ARM、智能终端实现轮式机器人导航的相关技术进行了研究。设计了一种软硬件协同处理平台的层次参考模型、三种协同处理拓扑关系以及通用协同处理片内结构、软硬件通信方式，并在轮式机器人应用中进行具体软硬件划分和实现，完成了 ARM 与 FPGA 高效通信及重配置、数字图像预

处理器、双闭环运动控制协处理器、点阵液晶显示后处理器、控制流层自主导航算法、智能手机平台控制应用。

本文将涉及 FPGA 部分的设计，研究内容可以分为以下几个部分：

第一章首先介绍软硬件协同设计，引出本课题研究的软硬件协同处理平台，再介绍了移动机器人的研究背景，然后介绍了目前国内外发展情况。

第二章主要结合实现系统的相关要求，设计系统软硬件协同处理架构，介绍开发方法，最后将上述架构和方法运用到小型轮式机器人平台的设计中，完成软硬件划分和功能描述，包含两类控制系统。

第三章详细设计了数据平面层与实时控制流层的数据通信方法，将 FPGA 片内总线互联架构进行了详述。

第四章中对数字图像预处理器进行了详细设计，包括接口同步、迭代法动态阈值分割、图像细化、中值滤波、图像压缩、总线接口等。

第五章中对运动控制协处理器进行了详细设计，主要是利用 FPGA 和软硬件协同设计工具 XSG 实现直流电机双闭环调速系统，并进行了系统级仿真。

第六章中对液晶显示后处理器进行了详细设计，包含时序接口、电子电位器接口、清零、总线接口等。

第七章对基于软硬件协同处理的轮式机器人平台进行整合实验。对两类控制系统进行了测试比较，结果良好。并根据结果分析了软硬件协同处理系统的优势。

第八章对基于软硬件协同处理的轮式机器人平台的设计与实现进行了简要的总结，并提出了今后需要进一步改进的方向，以及该类技术和平台的广泛意义。

对于报告中的错误和不足之处，希望各位读者能及时批评和指证。

2 系统原理及概要设计

2.1 软硬件协同处理架构设计

本文提出并设计了一种通用的软硬件协同处理架构，具有良好的可扩展性和综合性能，能满足小型嵌入式系统的诸多需求。

2.1.1 软硬件协同处理架构层次参考模型

满足小型嵌入式系统的处理平台架构由低到高可概括为三个层次：数据平面层（包含传感器与驱动器）、实时控制层、高性能处理层。本文实现的结构如下：

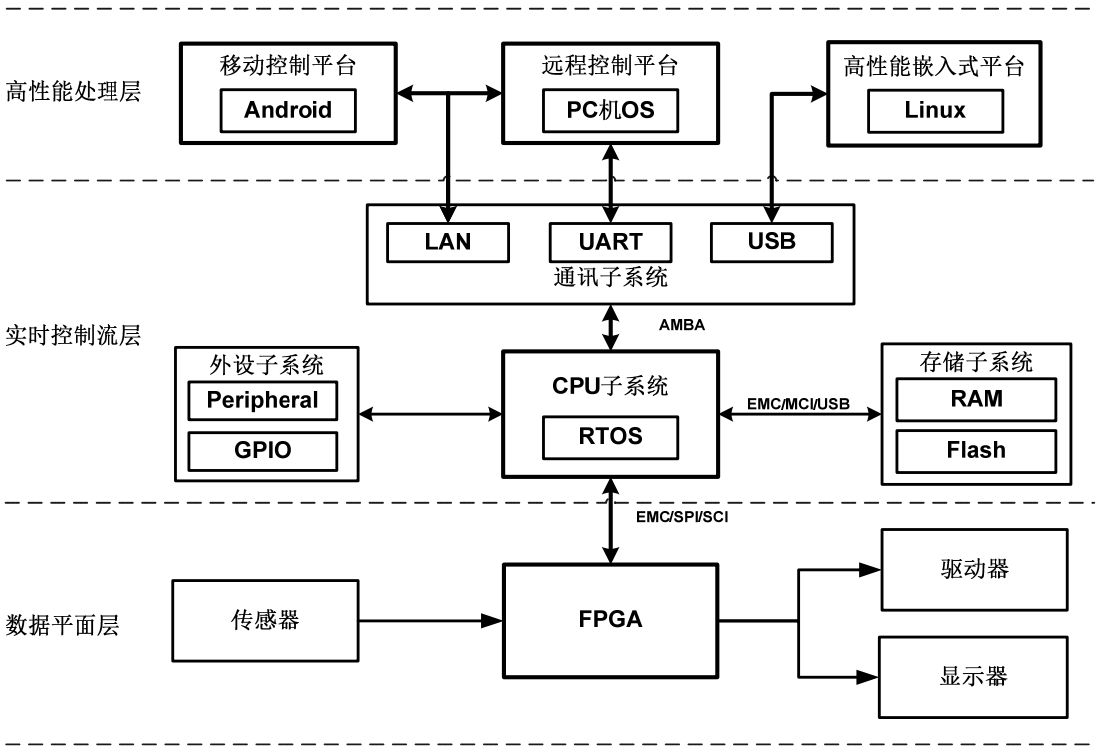


图 2.1 处理与计算平台三层模型架构

数据平面层要求搭载多种通用传感器和驱动机器人各方向运动的驱动器组。由于系统需要实时处理多种输入数据，并对输出数据进行实时控制，可使用软硬件结合的方法，扩

展一片中等性能的FPGA，将并行处理、大量数据运算等工作交由FPGA完成[31]。其中FPGA可以包含片上DSP模块[32]。

实时控制流层要求使用一块 50MHz 以上的中等性能嵌入式处理器作为该层核心处理器，满足控制任务的实时性、确定性，因此需要使用嵌入式实时操作系统，并且其能集成尽量多的通用外设，提供足够多的接口。

高性能处理层要求能够搭载大型嵌入式操作系统，如 Andriod 等；能扩展复杂应用；提供各种高级功能的通讯模块与传感部件；作为支撑系统的高性能处理，提供较为合适的人机界面。

2.1.2 实时控制流层与数据平面层拓扑关系

FPGA 与主处理器 CPU 的关系为预处理、协处理、后处理。

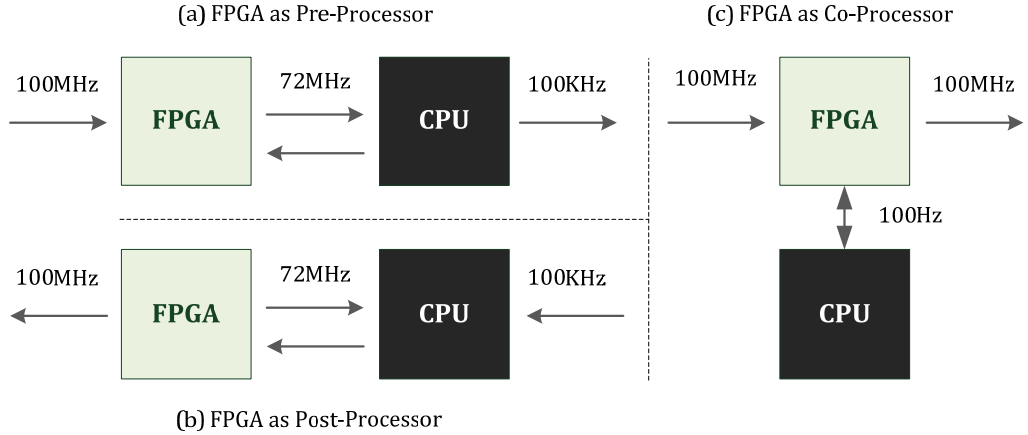


图 2.2 实时控制层与数据平面层处理器的三种拓扑关系

在预处理架构中，FPGA 位于数据平面前端，负责数字信号预处理，输出的信号可以高效又经济地移交给 CPU 处理器进行速率较低的后续处理[33]。

在协处理架构中，FPGA 与 CPU 呈现高低层次结构，FPGA 占据了整个数据平面，CPU 将数据处理算法交由 FPGA 处理，从而实现比单独采用 CPU 时达到更高的吞吐率。FPGA 的处理结果以较低数据量传回 CPU。

在后处理架构中，FPGA 位于数据平面后端，CPU 将预处理的结果交由给 FPGA 进一

步处理，完成更高带宽的运算和数据处理，从而减轻了 CPU 处理负担。

将上述软硬件协同处理架构应用到针对机器视觉的移动机器人领域，对控制系统环节进行适当的软硬件划分，系统结构将分为数字图像处理的预处理器、图形显示的后处理器、运动控制的协处理器。

2.1.3 通用软硬件协同处理平台：

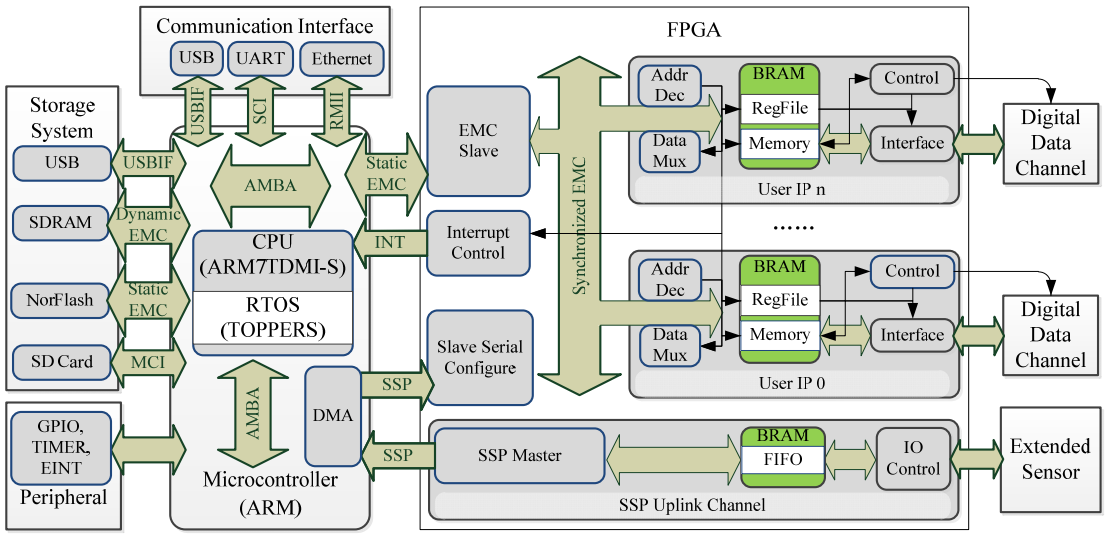


图 2.3 通用软硬件协同处理平台系统框图

具体设计时，高性能处理平台、传感器与驱动器则采用外扩方式，而将数据平面层与实时控制层紧密耦合在一块单板上，实现软件硬件协同处理平台。

按照接口，系统分为串行链路和并行链路两类。其中从串配置为下行，并行链路为双向总线。对于数据平面层，在 FPGA 内部设计了通用 IP 的扩展方式，将外部 EMC 总线同步，作为 FPGA 内部通用总线，每个 IP 需要设计自己的总线接口，包含地址译码、数据选择、片内寄存器和存储器选择等。而寄存器堆和双口 RAM 形式的存储器可以作为从 CPU 总线到 FPGA 外部物理数据接口的扩时钟域单元，三个时钟域相互独立，各自的时钟修改对设计影响大不。

对于实时控制流层，从开发控制系统的商业性、系统可升级性、算法复杂度以及工具链、生态链成熟度的角度考虑，使用基于 ARM 的 CPU 作为主控制器更易于实现。在 ARM

内部，采用了新兴开源实时嵌入式操作系统 TOPPERS[34]，这也是国内较早引入这套实时内核到原型开发平台的项目，经过充分验证，表明了其优良的性能和广阔的前景，同时通过一些实验验证了该软硬件协同处理平台在中小型嵌入式领域强大的综合性能。

这种软硬件协同设计平台有着较好的通用性和可扩展性，充分发挥了 ARM 与 FPGA 的各自特点，综合性能较好，可以作为小型嵌入式系统的原型开发平台。

2.1.4 实时控制流层与数据平面层的通信

（1） 通信架构分析

为了满足实时控制流层和数据平面层的高数据吞吐率，高效可靠的通信方式尤为关键。对小型嵌入式系统领域而言，系统通信方式可分为串行、并行两类。

使用普通串行接口的主要作用是 FPGA 的下行从串配置和专用数据链路传输。考虑到吞吐率、DMA 功能和丰富的协议兼容性，使用 ARM 的 SSP 接口作为 ARM 与 FPGA 的串行接口。SSP 可使用 SPI、SSI、Microwave 协议[35]。Xilinx FPGA 从串接口使用 SPI 协议，可以通过 ARM 微控制器配置，只用到了两线，即时钟和数据；配置电路其他握手协议信号有 PROG_B、INIT_B、DONE。

在小型嵌入式系统领域，并行通信是提高数据吞吐率较为合适的选择。由于 ARM 通常会外扩存储器总线，而 FPGA 内部包含丰富的寄存器资源、静态存储器资源，通过挂在 AHB 上的外部存储控制器 EMC（External Memory Controller）总线接口连接 ARM 和 FPGA，按照文献[19]的架构分类，就实现了一种间接（松耦合）的处理器间通信方式。基于这样的架构也是合理的：实时控制流层与数据平面层的吞吐率不需要太高，而数据平面层内部的 DSP、CPU 与 FPGA 逻辑之间则需要紧耦合。这样的架构把 FPGA 当做一块外部的存储器挂在 ARM 上，FPGA 响应 ARM 的总线操作，完成两者之间的通信。同时，将 EMC 作为 FPGA 的片内逻辑总线。这样，FPGA 和 ARM 的存储器一样，都是通过地址映射而接收 CPU 的访问，对 CPU 而言是透明的。正因为这种透明性，EMC 总线非常适合于做软硬件协同设计的接口。

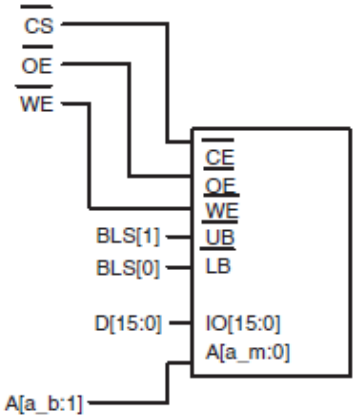


图 2.4　LPC24XX 的 16bit 异步静态 EMC 接口

（2）　串行、并行链路吞吐率总结

表 2.1 并行 EMC 总线与普通串行链路吞吐率比较（LPC24XX 系列 ARM7 微控制器）[36]

Interface	Width(bit)	Max Frequency(MHz)	Throughput(MB/s)
EMC	16	21.8	43.636
SPI	1	6.0	0.667
I2C	1	0.4	0.356

针对 LPC24XX 系列 ARM7 微控制器，通过上表可以看出，并行的 EMC 总线数据传输能力远高于普通串行接口，同时 EMC 通用的接口协议能兼容众多应用，因此将 EMC 作为实时控制流层与数据平面层高带宽的解决方案是较为通用和合适的选择。

2.1.5　实时控制流层与高性能处理层的通信：

（1）　通信架构分析

实时控制流层与高性能处理层需要完成两类基本通信：可靠的控制数据传输和高带宽的密集数据传输。对于可靠的控制数据传输，高性能处理层收发实时控制流层的关键控制数据，可靠性、容错性要求较高，需要差错控制，而带宽要求不高，需要满足一般控制系统的传输要求；对于高带宽的密集数据传输，高性能处理层收发实时控制流层的用户数据，对实时性、吞吐率要求较高，而对可靠性要求不高，允许误码、丢包。考虑到高性能处理层使用手机、平板电脑等移动智能终端的场合，以及实时控制流层硬件外设功能、软件协

议栈基础特性，将实时控制流层符合 IEEE 802.3 标准的以太网通过无线 AP 转换到高性能处理层符合 IEEE 802.11 标准的无线局域网，这种通用的通信方式较为合适，并能应用到工业控制领域的遥控机器人。

计算机网络对应上述两类基本通信方式的协议分别是 TCP (Transmission Control Protocol)和 UDP(User Datagram Protocol)协议，二者均属于 OSI/RM 参考模型中传输层[37]。其中 TCP 提供数据的可靠传输，通过面向连接、端到端实现可靠的数据包发送。而 UDP 对应的则是可靠性要求低、面向无连接传输快速的应用，包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。通过 TCP 支持的应用协议 Telnet[38]，可以实现虚拟的串口打印，从而实现高性能处理层的智能终端、PC 的兼容，对实时控制流层而言这两类平台是透明的，同时保证了控制系统关键数据的收发；而通过 UDP 实现密集数据流的收发，从而满足多媒体等高带宽要求应用，实现视频的移动智能终端显示等高级应用。

（2） TCP/UDP 协议吞吐率

对于视频实时回传应用，以 100*128 分辨率 256 灰度级别摄像头为例，如果需要每秒 30 帧的显示频率，则传输带宽要求至少为 375KB/s。而实时控制流层采用的 LPC24XX 系列 ARM7 微控制器提供的 TCP、UDP 数据吞吐率如下表：

表 2.2 以太网吞吐率 NETIO 测试结果（PC 机与 LPC24XX 系列 ARM7 微控制器连接）

Packet	**TCP (KB/s)**		**UDP (KB/s)**		**ICMP (ms)**
Size	**Rx**	**Tx**	**Rx**	**Tx**	**ping**
1 KB	1276	1027	N/A	2177	1.2
2 KB	1368	1014	N/A	2143	2.53
4 KB	1429	980	N/A	2159	3.98
8 KB	1418	1013	N/A	2136	7.3
16 KB	1429	1018	N/A	1838	N/A
32 KB	1403	993	N/A	1601	N/A

通过上表可以看出，当高性能处理层的 PC 机与实时控制流层的 LPC24XX 系列 ARM7

微控制器通过以太网通信时，ARM7 的 UDP 发送吞吐率高于 TCP，且二者发送吞吐率随着包容量的增加而减小、TCP 接收吞吐率随着包容量的增加而增加，但 UDP 最低 1601KB/s 的的吞吐率依然满足 30fps 帧速的视频实时回传 375KB/s 带宽要求。因此，只需选择合适带宽的无线局域网接入点（Access Point）即可满足高性能处理层移动智能终端与实时控制流层的通信需求。

2.2 轮式机器人控制系统软硬件划分

本文提出的软硬件协同处理平台架构具有良好的可扩展性和兼容性，能满足小型嵌入式系统应用的诸多需求。针对轮式移动机器人平台需求，结合上述处理与计算平台，进行系统设计。

2.2.1 控制对象：

机器人平台采用 4 轮驱动，差速转向。

（1）小车尺寸：长*宽*高为 360*355*165mm，小车净重：4.2kg；

（2）4 只 12V 空心杯行星齿轮减速电机，带 12 线编码器，空载转速：8100RPM，减速后转速：120RPM；空载电流：500mA；

（3）4 个 1：10 车轮，直径 130mm，最大行驶速度 80cm/s；

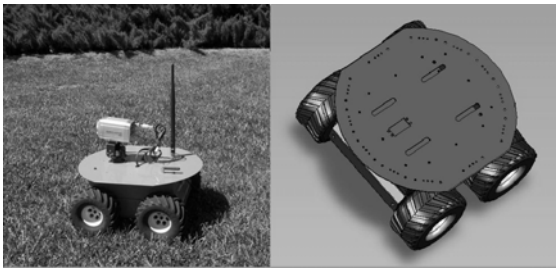


图 2.5 轮式机器人示意图

根据控制方法，控制系统可分为两类。下面分别进行控制系统的软硬件划分。

2.2.2 基于机器视觉的自主导航控制系统的软硬件划分

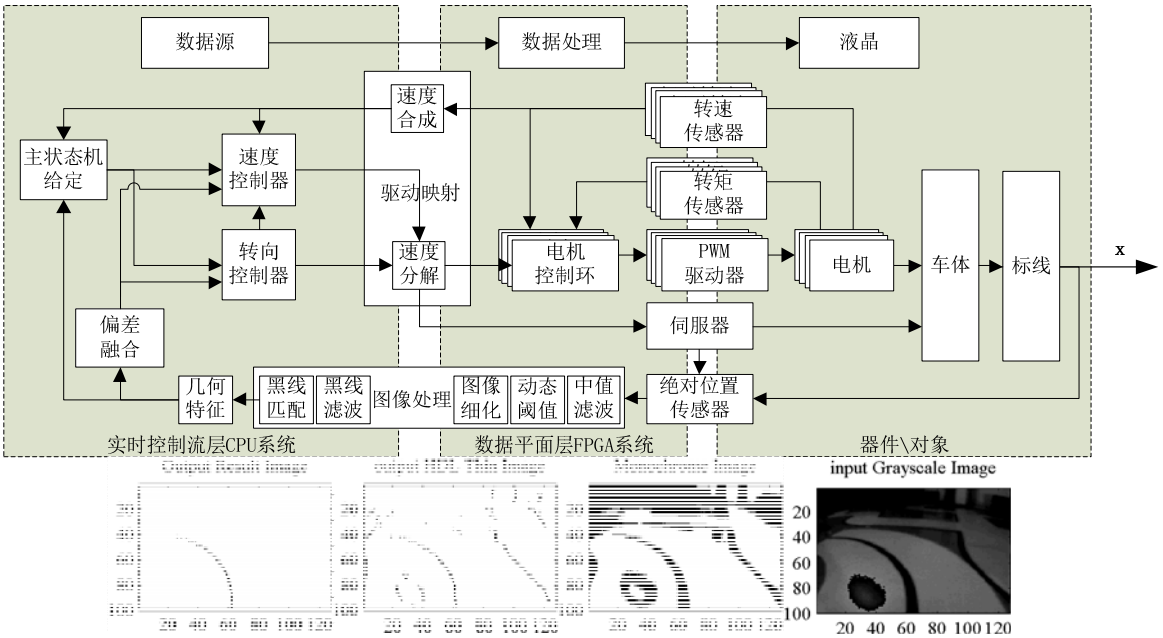


图 2.6 基于机器视觉的自主导航控制系统

对基于机器视觉的自主导航控制系统，该平台是针对机器而非人而进行处理，因此需要得到机器能有效时别的图像帧中的路径、标志等。轮式机器人根据检测的传感器信息进行自主导航，是一个自治智能控制系统，包含位置伺服、速度环、转矩换。轮式机器人采用 4 轮驱动，每个轮子均含一个电机及编码器，属于滑差控制。而位置传感器采用摄像头，航位推进算法以较小的同步采样周期对当前图像进行识别，并不对图像进行运动趋势检测，速度环由编码器检测。为了保证层次的兼容性，速度驱动映射层进行四轮编码器速度合成，位置环根据位置传感器信息和速度合成信息产生单一的速度和转向指令输出，经由驱动映射层进行速度分解，后续速度环、转矩换进行相应处理，从而完成整个系统闭环控制。下面针对各个子环节进行软硬件划分。

（1）数字图像处理与控制算法：

控制系统的数字图像处理部分主要包含接口同步、动态阈值分割、图像细化、标线滤波、标线识别。而控制部分包含几何特征计算、偏差计算、航位推进算法等。将以上两部

分程序在 ARM Cotex-M4 片内 128KB SRAM 以 100MHz 全速运行[39]，统计测试结果如下：

表 2.3 基于机器视觉的自主导航控制系统的软件时间消耗

	图像细化	标线识别优化	控制部分	总计
平均时间（us）	16652	3616	17	20285
百分比	82.09%	17.83%	0.08%	100%

可见，在固定阈值下，细化处理占用时间多达 80%，而后续图像处理算法、航位推进算法则较为复杂，因此将图像细化部分由数据平面层的 FPGA 处理，后续其他图像部分和控制算法由实时控制流层的 CPU 处理，形成典型的预处理器结构，节约 CPU 时间。

对实时系统而言，数据量大时通信接口负担较重，给系统实时性造成较大影响。以本设计中的图像处理为例，一帧 100*128 像素的图像通过 16 位 EMC 外部存储器接口从 FPGA 转移到 ARM7 的片外 SDRAM 需要的时间约为 1ms，仅仅一幅较小图像的传输就花费大量时间，对系统实时性能影响很大，因此为了减轻软硬件通信接口负担，FPGA 需要对数据进行必要的压缩再供软件处理。

（2）运动控制：

目前实现运动控制主要依靠具有硬件乘法器甚至浮点运算单元的高性能 DSP 完成。相对比较来说，使用 FPGA 开发所具有的优势主要包括以下几点。

- 定点转换：XSG 软件十分方便，操作在一个时钟周期内即可完成。
- 并行硬件（乘法器）资源：单个乘法器的价格对比明显。
- 内外流水线：保证在计算效率降低极少的情况下实现硬件资源的充分利用。
- 节拍加速比：算法确定、计算流程确定，没有无序响应等等。

将双闭环在数据平面层的 FPGA 中实现、位置环指令输出在实时控制流层的 CPU 中实现，是一种典型的协处理器结构。

（3）点阵液晶显示：

人机交互在自主控制系统中有一定的交互意义，第一类控制系统中采用点阵液晶进行机器视觉相关车载显示，呈现基本的人机界面图像处理结果和实时监控数据。LCM 采用背

光单色点阵液晶，8bit 并行数据总线，无字模，分辨率 256*64[40]。由于接口时需部分占用了大量时间，而数据源则因其随机、打印复杂性，更适合于 CPU 处理。因此，实时控制流层的 CPU 写虚拟显示器，数据平面层的 FPGA 实现显示器物理接口，形成后处理器结构。

2.2.3 基于人工视觉的远程导航控制系统的软硬件划分

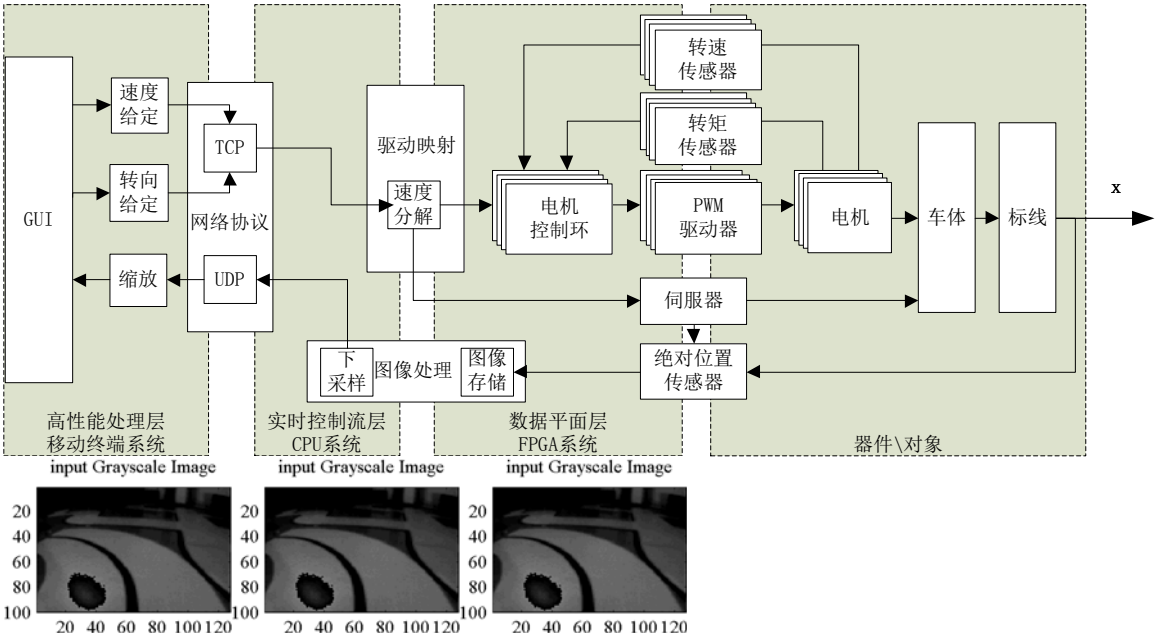


图 2.7 基于人工视觉的远程导航控制系统

对基于人工视觉的远程导航控制系统，其采用 Android 智能终端，利用 wifi 进行近距离无线手动控制与视频回传，是一种近年来伴随智能终端流行起来的移动机器人控制方法[41]。由于人工控制可以通过视频中标线运动趋势判断大致速度，不关心编码器反馈的速度，因此驱动映射层仅进行速度分解。位置传感器采用摄像头，经由无线网络 wifi 将原始摄像头视频实时回传至 Android 智能终端，可以协助使用者进行近距离监控的无线控制，在工业控制领域有较大应用前景。

类似第一类控制系统，数据平面层的 FPGA 协处理器完成运动控制部分，FPGA 预处理器完成数字图像处理的基本存储功能。不同之处在于，实时控制流层在第二类控制系统中仅完成协议转换和数据传输，位置环的速度输出、人工视觉在高性能处理层的 Android

智能终端由人机交互完成。

2.2.4 硬件平台

主 CPU 采用 ARM7 架构的 NXP LPC2478，FPGA 采用 Xilinx 的 Spartan6 系列 XC6SLX16-FTG256[42]，ARM 通过 FPGA 的接口完成运动控制、液晶显示，以及摄像头数字图像处理。

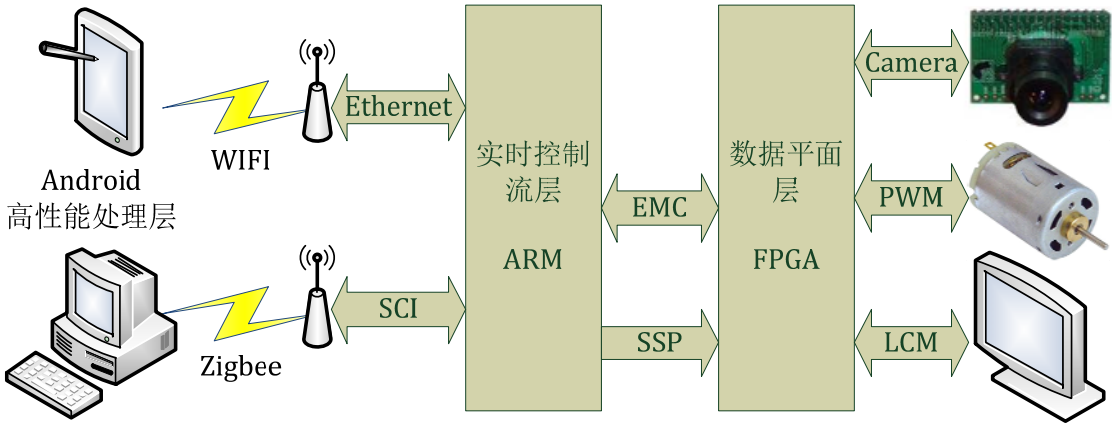


图 2.8 软硬件协同处理平台硬件示意图

数字摄像头 KAC-9630 为 256 灰度级别[43]，分辨率 128*100。因此一帧灰度图像容量为 12800 字节=12.5KB，XC6SLX16 有 64KB 内部 BRAM，能存储约 5 帧图像，可以满足基本应用。

3 实时控制流层与数据平面层的并行通信设计

3.1 基于总线互联的软硬件接口

为了完成实时控制流层 CPU 通过 EMC 总线对数据平面层 FPGA 的有效控制，需要设计通用的FPGA 片内总线互联方式。EMC 从机模块为FPGA 内部各模块提供同步过的EMC 总线信号，这样 ARM 能作为主机控制 FPGA 内部所设计的功能模块，各功能模块呈总线拓扑结构，其中只有一个主机，即 EMC 从机模块。各个子 IP 含有寄存器堆，进行 CPU 到 FPGA 的 IP 功能模块的控制；子 IP 还包含数据存储器，便于 CPU 与 FPGA 之间密集数据的处理。为了识别主机片选，可以采用地址译码器；为了将各个 IP 数据读总线输出到 EMC 从机模块统一总线上，需要使用数据选择器。

将 ARM 的 EMC 总线同部为 FPGA 的片内总线，连接到各个子 IP。通用片内 IP 总线互联结构如下图所示。

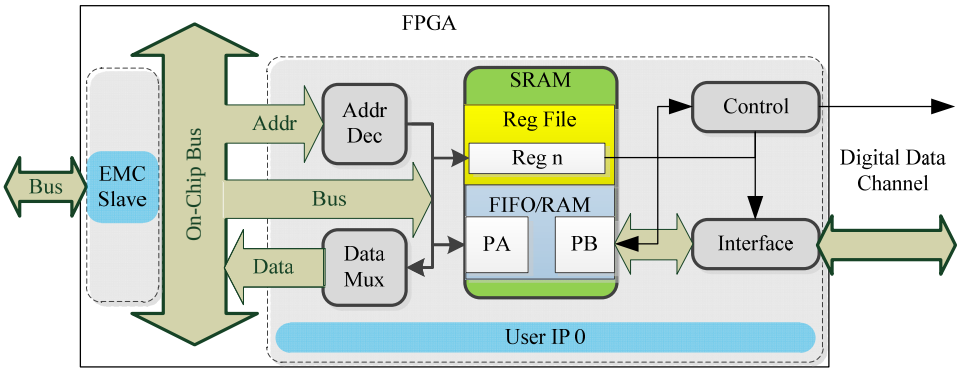


图 3.1 通用 FPGA 片内总线架构

可见该架构包含片内总线驱动器/同步器模块、子 IP 的总线接口模块、子 IP 的存储单元/同步单元、子 IP 的用户定义功能模块等。其中子 IP 的存储单元/同步单元包含寄存器堆和存储器，前者主要负责 IP 控制，后者主要负责数据存储，另外二者均可作为片内总线到用户功能模块的跨时钟域同步单元。

3.2 EMC 从机接口设计

3.2.1 功能

对 EMC 异步存储器总线做过采样同步，输出到片内各个功能模块作为片内存储器总线。由于在 ARM 端使用的是异步存储器总线，FPGA 端需要进行同步，使用比 EMC 控制脉宽更高频率的时候进行采样。总线转换结构如下图所示：

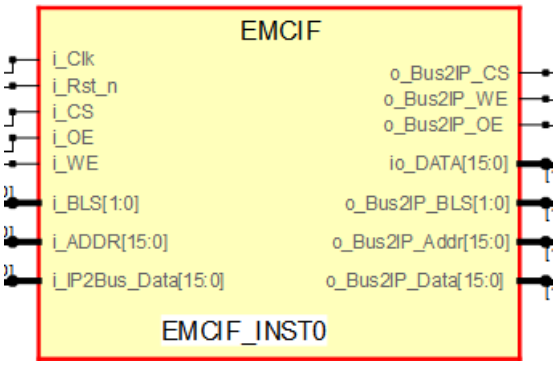


图 3.2 EMC 从机模块外部接口。左侧为输入，右侧为输出

3.2.2 同步设计

EMC 静态接口没有时钟同步，而其频率接近 FPGA 全局时钟，故需做跨时钟域设计。

（1） 总线仲裁

主要是为了得到正确的 read 和 write 信号。考虑到总线仲裁，即 EMC 总线读 read 和 BRAM 总线写 write，需要对 EMC 的 WE、OE 进行第三方判断，这里采用过采样，从而延后 WE、OE 一个过采样周期，保证保持时间。

（2） 时钟

WE 脉宽比 CS 窄，持续时间比 data、addr 短，因此只要能采样到 WE，CS、data、addr 信号一定是稳定的。ARM 的 EMC 读时序，地址一定先于 OE 有效，而且要求存储器在 Tam 时间内输出有效数据。Tam 为固定值，这个值限定了 FPGA 采样 OE 的频率[36]。读时序需

要的采样时钟为 Tcy(CCLK) × 1.3 / 2 = Tcy(CCLK) × 0.15，即 2*72MHz/1.3 = 110MHz，即至少需要 110MHz 的过采样时钟才能准确识别 WE、OE，从而保证总线占用时序正确。使用一路 DCM 生成的 110MHz 作为主时钟，因而 EMC 读写时序要求一致。

3.3 IP 总线接口设计

3.3.1 AddrDec：

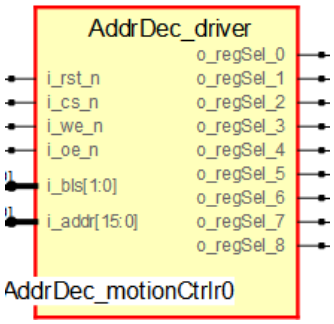


图 3.3 片内总线地址译码器模块外部接口。左侧为输入，右侧为输出

采用地址译码器识别主机片选。输入：EMCIF 同步后总线信号（除数据总线外）；输出：各个寄存器及存储器片选。

3.3.2 RdDataMux：

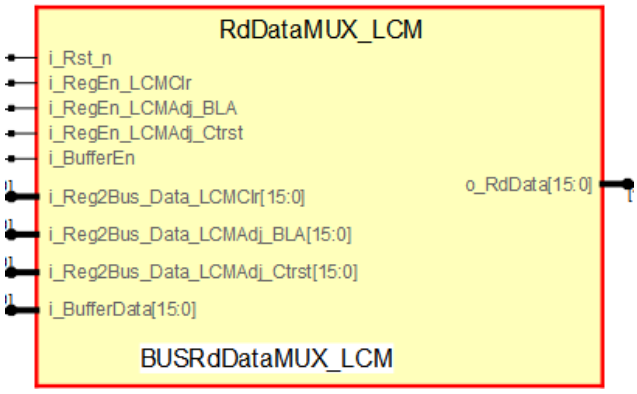


图 3.4 片内总线数据选择器模块外部接口。左侧为输入，右侧为输出

使用数据复用器将各个 IP 数据读总线输出到 EMCIF 统一总线上。

输入：各个寄存器及双口 RAM 片选，各个寄存器及双口 RAM 数据总线；

输出：EMCIF 数据总线。

3.4 寄存器堆设计

3.4.1 功能

ARM 由 EMCIF 总线读写控制数据的通用接口，完成后端 IP 控制或标示功能。同时作为 EMC 接口时钟域到 FPGA 片内 IP 与片外芯片接口时钟域的同步单元。

实现结构可采用专用存储器（BRAM、FIFO、分布式 RAM）或握手协议，简单考虑也可以采用电平同步器[44]。由于考虑到双端读写以及其在本设计中的细粒度性，使用专用存储器不合适，而 EMC 写寄存器控制 IP 属于 IP 长时间读取过程，采用握手协议较为复杂，不如简化为两拍电平同步器。

一般来说，EMC 总线同步单元时钟频率满足上文所述即可，即相对 72MHz 固定在 110MHz，而 IP 内部频率可以波动。若 IP 内部频率小于 110MHz，则为快时钟域到慢时钟域，但一般 EMC 写寄存器并不是脉冲型过程，而是长时间的电平过程，因此慢时钟域有足够的时间满足快时钟域的建立保持时间，同时，EMC 读寄存器则是快时钟域采样慢时钟域，若为窄脉冲则建议使用沿同步电路[45]，简化考虑用 ARM 端的中断捕捉外设；若 IP 内部频率大于 110MHz，则为快时钟域采样慢时钟域，同理，针对长时间电平采样可用电平同步器。

3.4.2 堆结构

以某一模块为例进行说明：

输入：EMC 主机异步总线；

输出：EMC 从机同步总线。

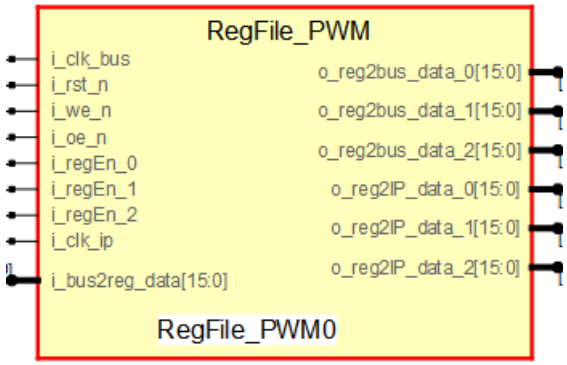


图 3.5　寄存器组模块外部接口。左侧为输入，右侧为输出

内部结构示意图如下，分为双端口读写、总线读写/IP 只读、总线只读/IP 只写：

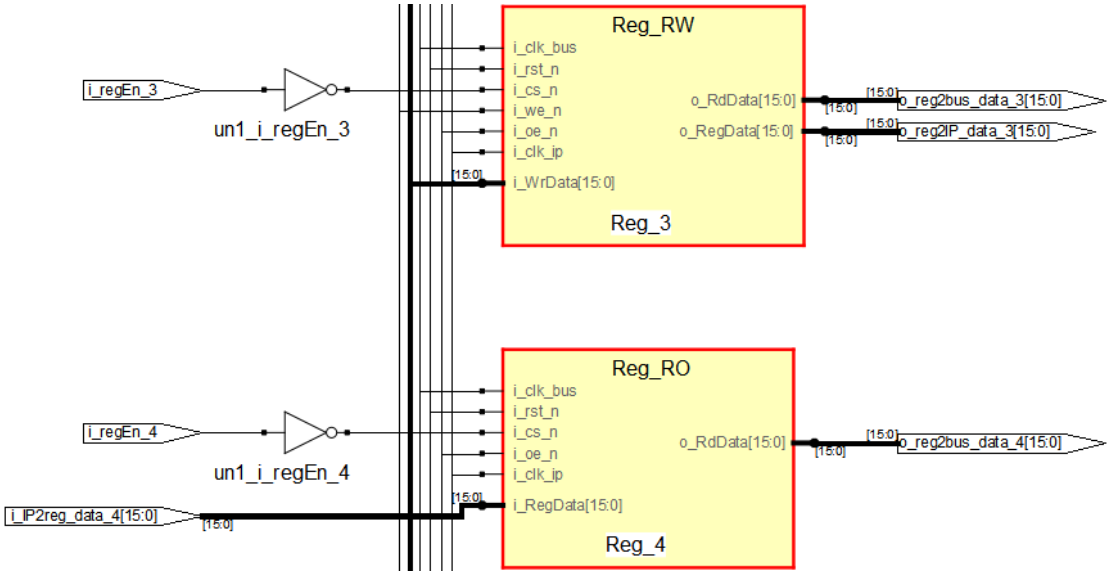


图 3.6　寄存器堆结构示意图

3.4.3　单元结构

双端读写单元每一个寄存器单元外部结构是一个基本位宽双端读写存储器。这里数据总线位宽为 16 位。

总线读写单元 IP 端结构是跨时钟域的双节拍电平同步器，而总线端是读写使能端口。总线端可读写，而 IP 端只读，总线端到 IP 端为双节拍电平同步器。

总线只读单元总线端只读，而 IP 端只写，为双节拍电平同步器。

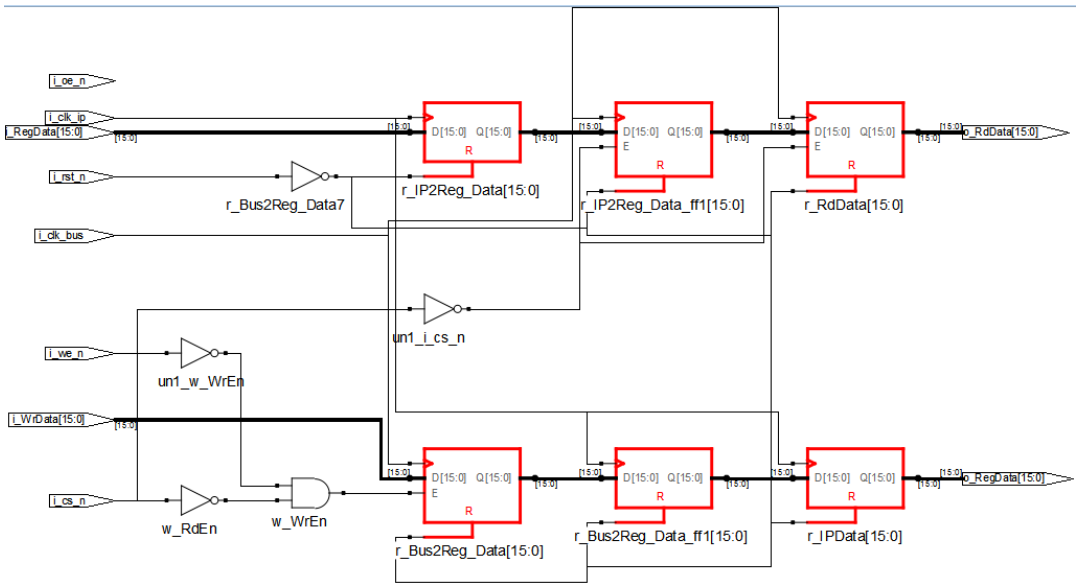


图 3.7　寄存器堆双端读写单元

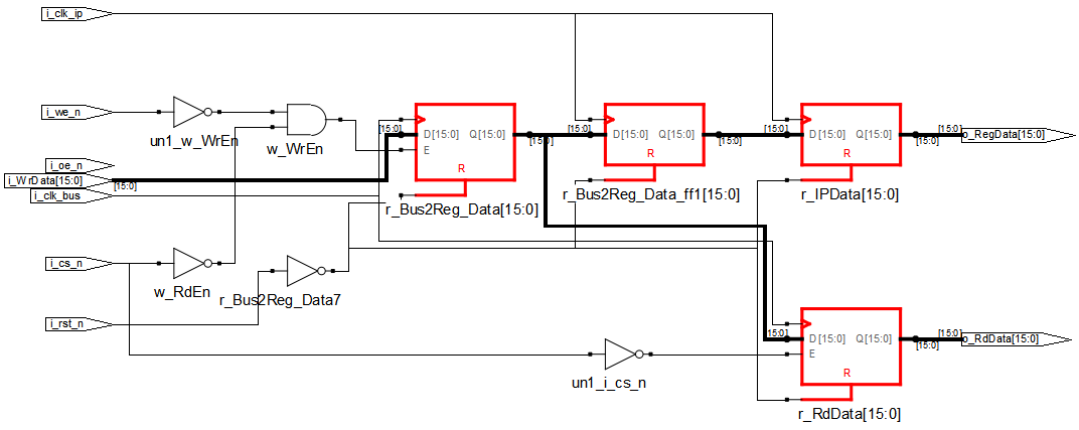


图 3.8　寄存器堆总线读写单元

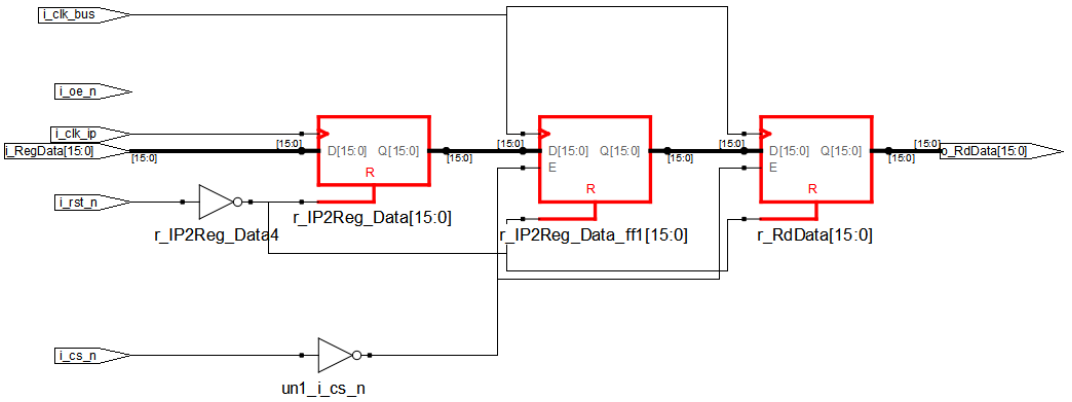


图 3.9　寄存器堆总线只读单元

4 摄像头数字图像预处理器设计

4.1 预处理系统框图

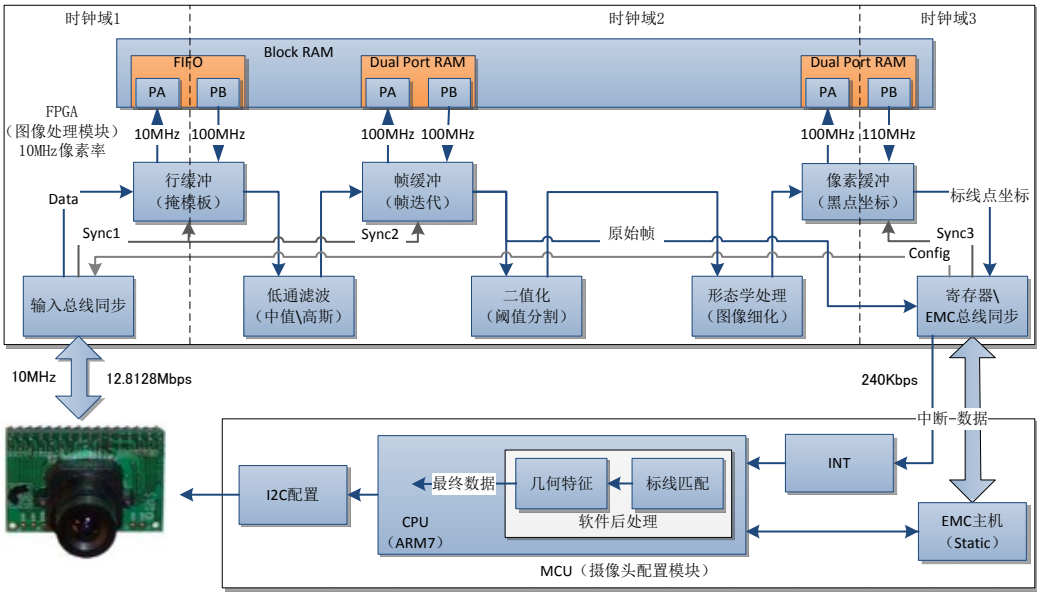


图 4.1 摄像头数字图像处理框图

按数据流划分，模块分为摄像头、FPGA 数字图像预处理器和 ARM 后端处理器。摄像头输出数据形式为 8bit 数据总线、1bit 时钟、1bit 行同步、1bit 帧同步。

摄像头接口同步将一定速率的接口信号从慢时钟域 1 同步进数字图像预处理器的时钟域 2（采用系统同步或源同步[46]），内部进行的处理包括掩模板低通滤波、灰度二值化处理、掩模板或数据流形式的图像细化。而后数据被总线接口同步至 EMC 主机，交由后端主处理器 ARM 处理，内容包括进一步滤波、引导线匹配、几何特征运算等。

由于 FPGA 将摄像头密集数据进行预处理后，带宽要求大为降低，实时控制层 CPU 仅进行低带宽的后处理操作，因此数字图像处理应用实际上是实时控制层与数据平面层的预处理器关系。

4.2 寄存器堆

表 4.1 数字图像处理器寄存器堆

双端读写	0	16'h0018	DIP	CTRL	THRESHOLD[7:0]		
					reserve	ENABLE	VSYNC
总线只读	1	16'h001a		BADDR	BLACK_ADDR[15:8]		
					BLACK_ADDR[7:0]		

（1） 编号 0：

寄存器 DIP_CTRL，可读写。

ENABLE 代表 ARM 写入的使能模块使能，VSYNC 代表模块产生的帧有效信号，其为一个脉冲宽度，通过 ARM 与 FPGA 之间的外部中断口进入 ARM 的外部中断 1（EINT1）。建议用上升沿或下降沿触发外部中断 1，因为此 VSYNC 中断信号脉宽很短。

高字节的 THRESHOLD[7:0]代表动态阈值模块对当前帧产生的阈值，软件对其进行读取可进行图像进一步的操作。

（2） 编号 1：

寄存器 DIP_BADDR，只读。

当 VSYNC 有效时代表本帧压缩完的黑点数据在另一个存储器中的地址长度，ARM 收到中断后将从存储器中按 0 到高位地址的顺序读出压缩后的数据。

4.3 引导线压缩坐标存储器与转换

4.3.1 数据转换模块

这部分主要实现将图像细化部分输出的压缩后的黑点坐标数据与同步信号转化为双口 RAM 的写入信号，并写入到双口 RAM 中，写完一帧后，若上游功能 IP 产生握手信号 o_ack（详见二值化小节）则产生中断信号，通知 ARM 读取双口 RAM 中的黑点坐标值和宽度。

4.3.2 ARM 存储空间地址映射

表 4.2 引导线中坐标存储器映射

方向	注	addr	reg		7:MSB	bit	0:LSB
总线只读	2KB	16'h2000	DIP	DPRAM[0]	ADDRX[7:0]		
					WIDTH[7:0]		
		……		……	……		
					……		
		16'h27fe		DPRAM[1023]	ADDRX[7:0]		
					WIDTH[7:0]		

对于 100*128 分辨率 256 灰度摄像头，其数据容量为 12.5KB，经二值化、黑线细化后中点骨架数据量大大减小，经测试，对于赛道环境，一般一帧数据为几百字节，使用 2KB 容量的 8 位输入、16 位输出简单双口 RAM 足够。ARM 通过 EMC 读取时每两字节拼接成 16 位，低字节为可能的黑线宽度，高字节为可能的黑线中点坐标。

4.4 图像动态二值化实现

4.4.1 结构与算法

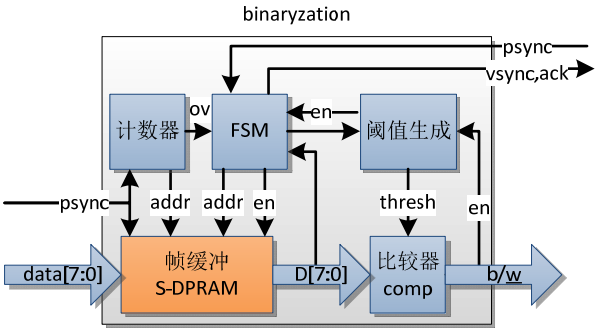


图 4.3 动态阈值分割模块框图

采用上图的通用结构，阈值生成模块可以采用不同的算法。对于嵌入式实时预处理系统，若光照环境较为均匀而且分割图像直方图波形有一定规律，采用迭代次数较少的迭代

法较为合适。

迭代法又称统计门限法[47]，本质上属于基于点的全局阈值动态分割算法，基于二分逼近的思想，用统计的方法寻找最佳门限，使分割出错的概率最小。适合于图像中物体和背景有较明显的灰度分布差异，阈值选择在两个波峰谷底较为合适。而对于基于机器视觉的自主导航控制系统，轮式机器人需要识别黑线白底道路，灰度直方图呈现两个波峰，因此选择迭代法较为合适。

为了在 FPGA 中实现并满足软硬件协同处理系统的实时要求，需要对常规迭代算法进行修改。具体实现步骤如下[48,49]：

（1） 确定初始阈值T_n，最好选在灰度直方图密度较大处，而不是采用最大最小值二分法;

（2） 根据当前阈值T_n，将图象进行阈值分割，得到阈值两端的灰度图像部分，分别求出两者的平均灰度值$G_{1,n}=\frac{\sum_{g(i,j)<T_n} g(i,j)}{\sum_{g(i,j)<T_n} n(i,j)}$，$G_{2,n}=\frac{\sum_{g(i,j)>T_n} g(i,j)}{\sum_{g(i,j)>T_n} n(i,j)}$；

（3） 求出新阈值$T_{n+1}=\frac{G_{1,n}+G_{2,n}}{2}$；

（4） 若$|T_n - T_{n+1}| \le \delta$，则前一次所得阈值$T_n$产生的二值化输出即为确认数据流，同时输出确认信号；否则转步骤（2），迭代计算。

其中$G_{1,n}$、$G_{2,n}$是第 n 次迭代后分割的两组数据的平均灰度，收敛因子δ根据实际处理对象和环境不同，这里选为 3。之所以选择$|T_n - T_{n+1}| \le \delta$的非严格收敛条件而不是$T_n = T_{n+1}$，是考虑到了特定引导线环境下的不精确性和实时性，机器视觉处理对象直方图呈现一定波峰波谷，没必要为了较小的阈值误差而进行多次迭代。而初始阈值的选择则是为了排除椒盐噪声对不收敛、迭代过多产生的影响。

按帧处理，100MHz 时每 128us 读取一帧，若迭代次数少于 10 次，则动态阈值分割可

在 1ms 内完成。

4.4.2 利用 A3 原则进行阈值生成模块设计

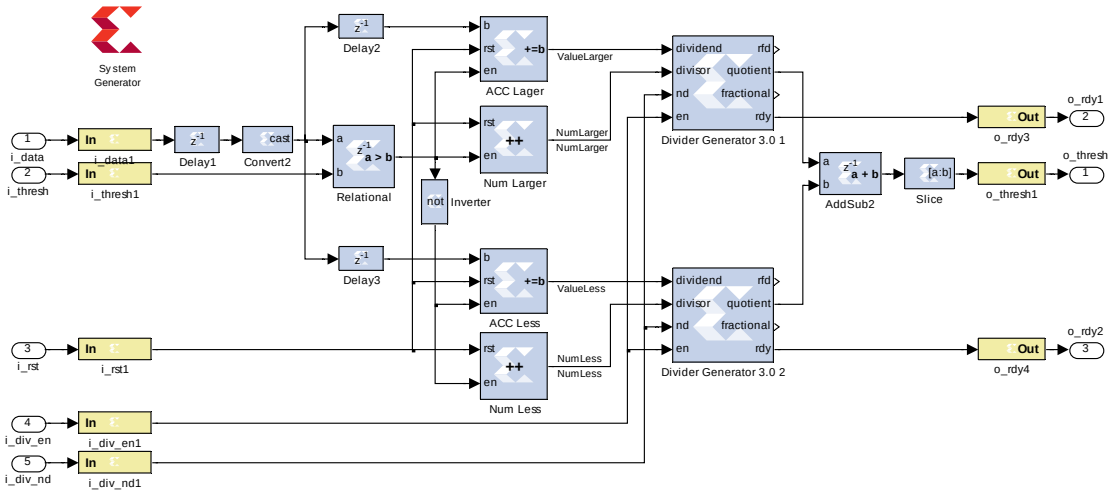


图 4.4 迭代法阈值产生模块的 XSG 建模

由于阈值生成模块是基本的数据流运算，而迭代是条件判断，阈值生成模块可以采用 XSG 数据流图建模实现[50]，而迭代条件采用 HDL 状态机实现。

关于除法环节的实现，有二进制除法器和高进制除法器两种，前者采用纯逻辑资源实现，后者采用逻辑资源和 DSP 核实现[51]。二者对比如下表。

表 4.3 二进制除法器、高进制除法器的 DFG 实现对比（Spartan-6 系列 FPGA）

除法器	位宽	Fmax（MHz）	延迟（拍）	FF	LUT	RAM	DSP
二进制	32	166	1	3,200	2,157	0	0
高进制	36	228	<30	1,121	744	1	13

可见高进制除法器通过利用 DSP 核、RAM 核来减少对逻辑资源的占用，考虑到 Spartan-6 系列 XC6SLX16 有 32 个 DSP 核和 64KB 块 RAM，而且除法运算只在每一帧结束后迭代时进行，对流水线节拍要求不高，因此阈值生成模块的除法器采用高进制除法器 IP 核实现，需要额外的握手信号，由外部状态机完成。

4.4.3 状态机设计

采用迭代法，外部状态机主要进行数据迭代推进，完成双口 RAM 的数据存储与读取、阈值生成模块的握手信号与使能、计数器地址计算。状态转换图如下图。这是实现的第一类状态机，状态本身输出或者状态转换条件输出。阈值生成模块包含三部分计算：基流水线的累加模块、基于握手信号的除法器模块、基于流水线的加法、减法模块。

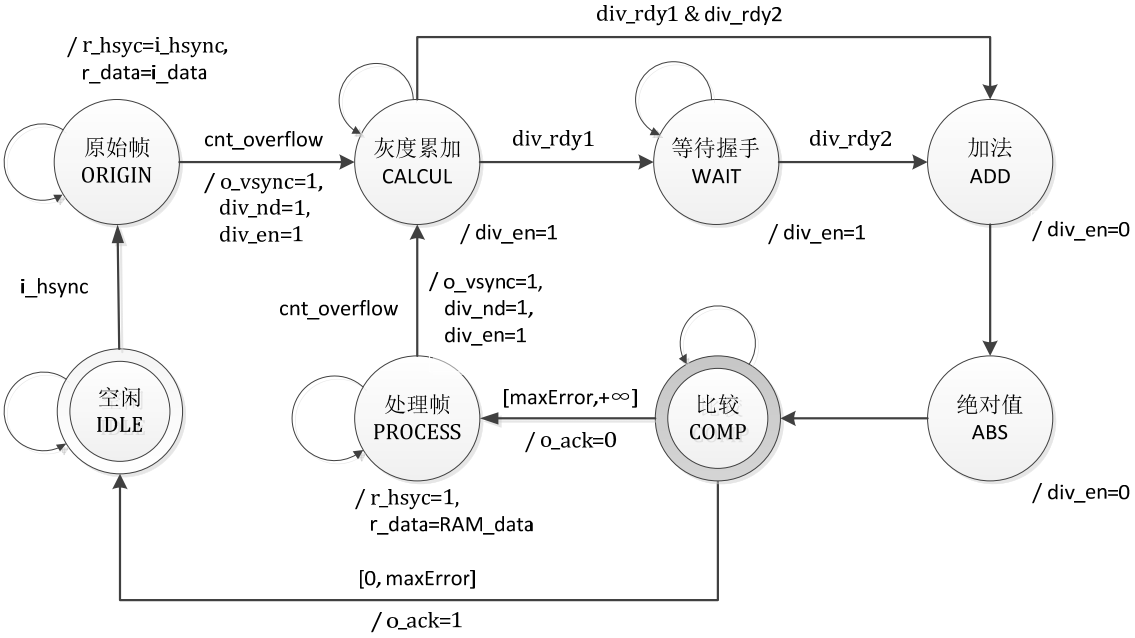


图 4.5 迭代法阈值模块外部状态机状态转换图，/代表输出

空闲状态 IDLE 检测到输入行同步信号 i_hsync 后进入原始图像输出状态 ORIGIN，输出的行同步信号和数据有两处目的地：阈值生成模块输入与下一级模块输入。当计数器溢出时表示当前帧已计数完毕，此时输出帧同步信号与阈值生成模块的除法器使能信号，并进入阈值计算状态。若检测到阈值生成模块输出的两个除法器的数据有效标志 rdy，则进入加法状态，并将除法器使能关闭。下两个周期分别进入绝对值计算状态和阈值比较状态，靠时钟触发，仅仅为了与流水线同步。绝对值计算状态对当前帧计算出的阈值与上一帧的阈值进行减法，并取绝对值得到阈值误差。阈值比较状态中，若阈值误差小于最小误差 maxError，则进入空闲状态等待下一帧图像输入，并输出握手信号到下一级表示确认当前

帧已经输出的数据有效；若大于 maxError 则进入处理图像输出状态，输出行同步信号与时钟使能同步，即维持 1，而输出数据为原始图像输出状态写入到双口 RAM 中的数据。同理，当计数器溢出，则进入阈值计算状态。

注意高进制除法器采用迭代法，会不断检测输入被除数与除数，需要保持输入不变。该状态机触发条件结合了流水线与握手信号，需要做好同步处理。

4.4.4 XSG 仿真平台设计

将实现的 HDL 及网表导入 XSG 环境下的黑盒模型：

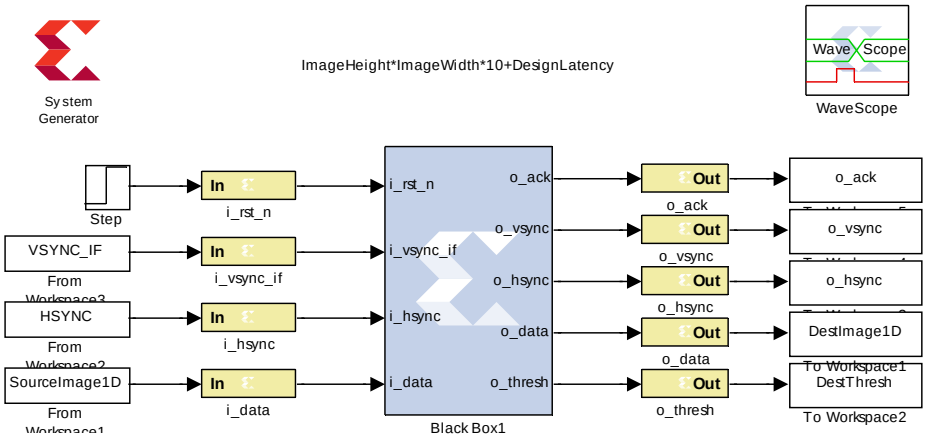


图 4.6 迭代法动态阈值分割模块的 XSG 黑盒仿真模型

结果如下：

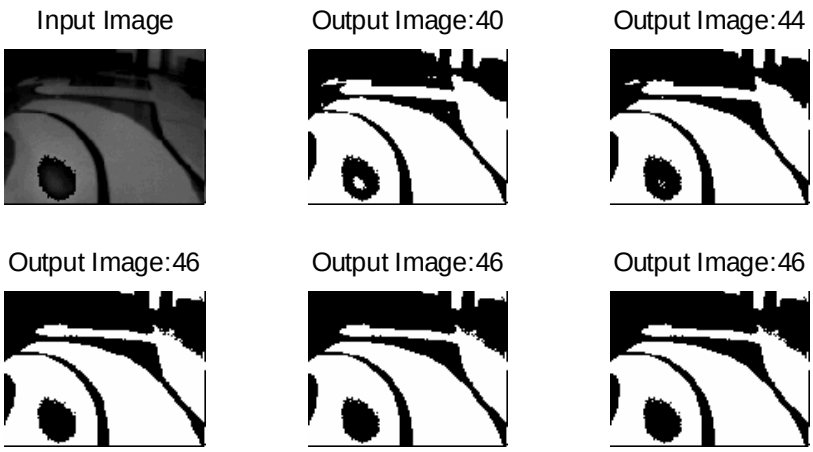


图 4.7 迭代法动态阈值分割模块 XSG 黑盒仿真结果

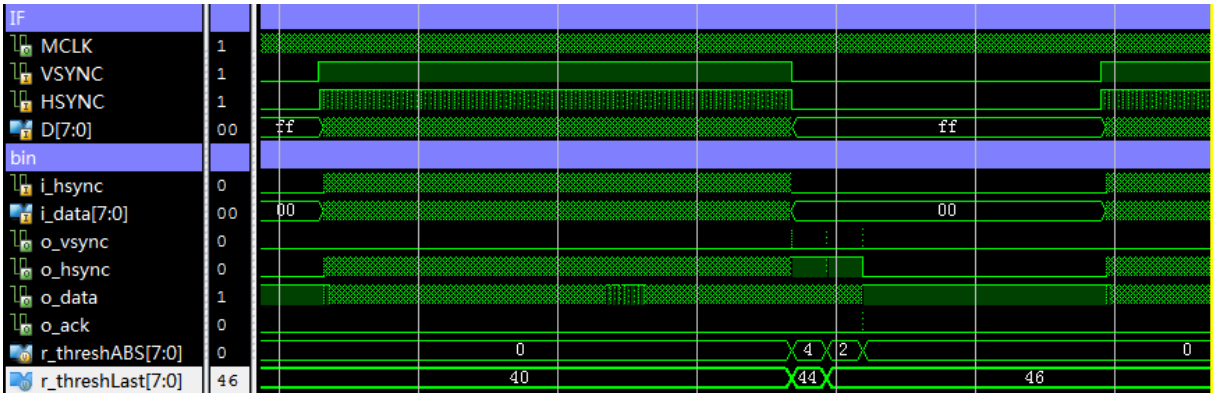


图 4.8　迭代法动态阈值分割模块 ISE 数据流仿真结果

可见对初始阈值 40，只需两次迭代即可达到收敛阈值 46，其花费的时间仅为两帧片内图像时间 12800*2/100MHz=256us，性能显著优于一般软件平台。

4.4.5　可能存在的问题及解决方案

经试验，发现对于直方图双峰明显，谷底较深的图像，迭代方法可以较快地获得满意结果。但基于点的全局动态阈值分割算法在特殊情况下存在一些问题。

对于二分法取平均值，如果其中一个商是局部椒盐噪声的极值，那么会对本次迭代结果产生很大影响；如果初始阈值选取不当、未低通滤波，则迭代次数相对较多。如下图：

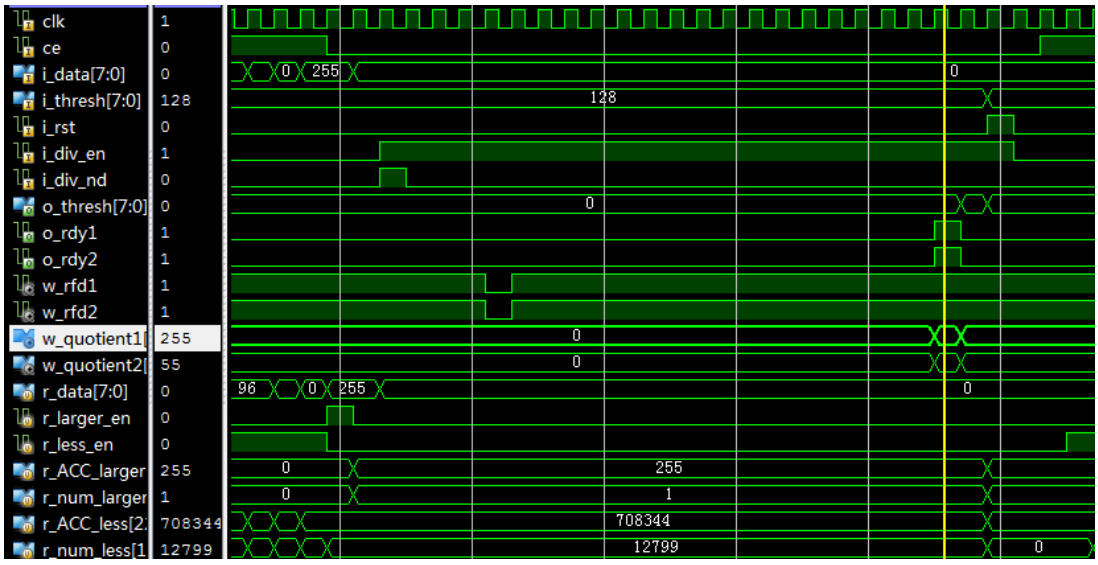


图 4.9　迭代法问题 1：大于 128 只有一个椒盐噪声点 255，但二分法产生一个输入 255

本帧灰度直方图的几个峰值孤立较远时，初始阈值若选取到直方图空白区域，与本帧数据恰好产生上述椒盐阈值，那么生成阈值结果将不收敛。如下图：

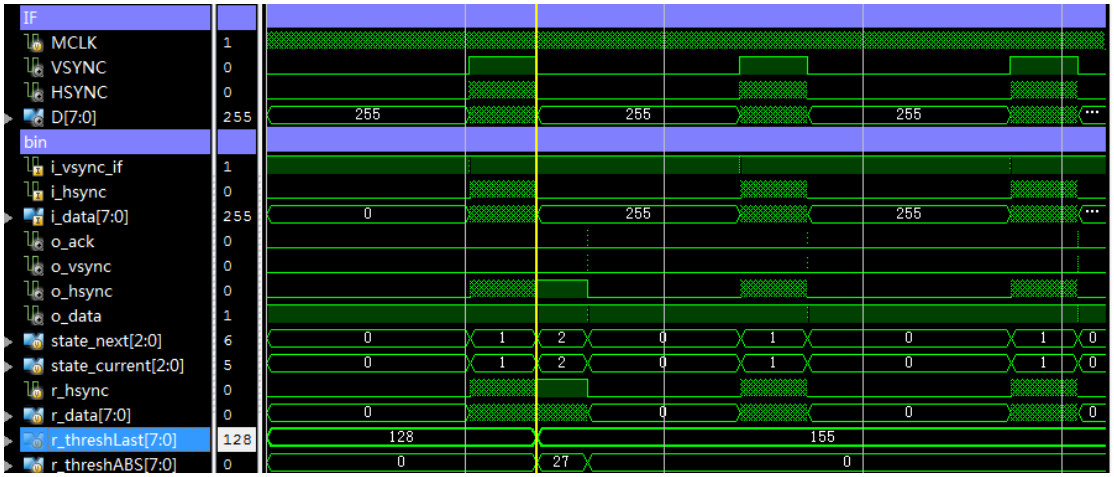


图 4.10　迭代法问题 2：初始阈值选在数据与椒盐噪声之间的空白区域，引起不收敛

因此，最好在二值化前一级使用中值滤波进行数据预处理；初始阈值可以采用最大最小平均值法，但需要低通滤波；也可以采用统计平均值法，避免孤立灰度的影响。将原始图像进行中值滤波[52,53]，去除影响迭代结果的椒盐噪声[52]，得到迭代阈值分割结果如下图：

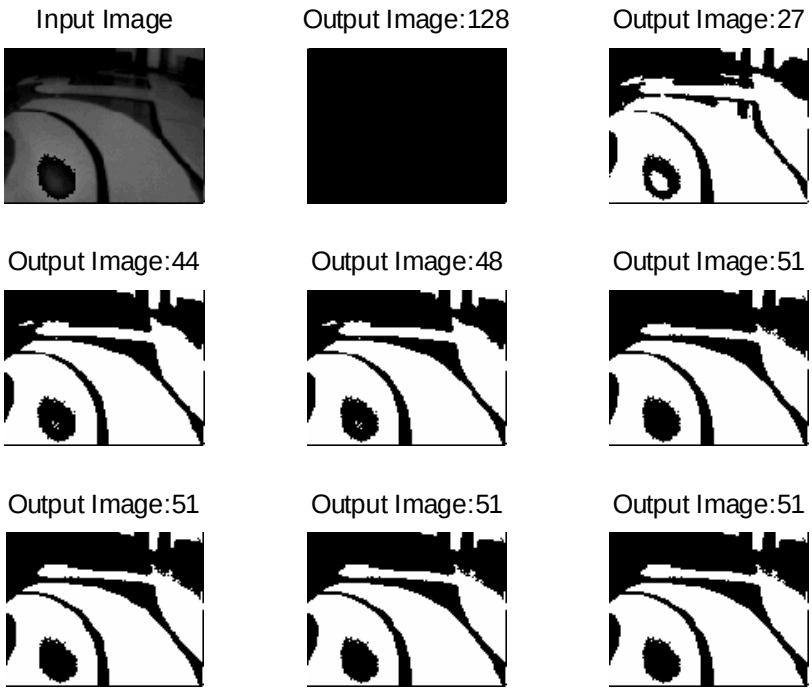


图 4.11　中值滤波后迭代法处理结果

可见初始阈值 128 的选取不恰当，但是由于滤除了椒盐噪声，解决了二分阈值引起的单点对全局影响很大的问题，同时也解决了初始阈值不恰当导致的生成阈值发散问题，4 帧迭代就实现了阈值收敛，达到了较好的系统特性。

4.5 针对前向导引线的图像细化实现

4.5.1 状态机设计

不同于图像形态学处理中的细化，项目根据轮式机器人黑线导航的机器视觉特性，设计了方向异性的黑线压缩算法。对此模块，输入的图像为黑白数据，黑点为 1，白点为 0。输出的有效数据为黑点在 128*100 像素矩阵中的列坐标。为了充分利用 EMC 总线的 16 位数据，同时为了记录压缩前的有效信息，算法采用了地址 1 固定黑点和记录连续有效黑点宽度的方式，并将列坐标与宽度拼接为 16 位输出到下一级。

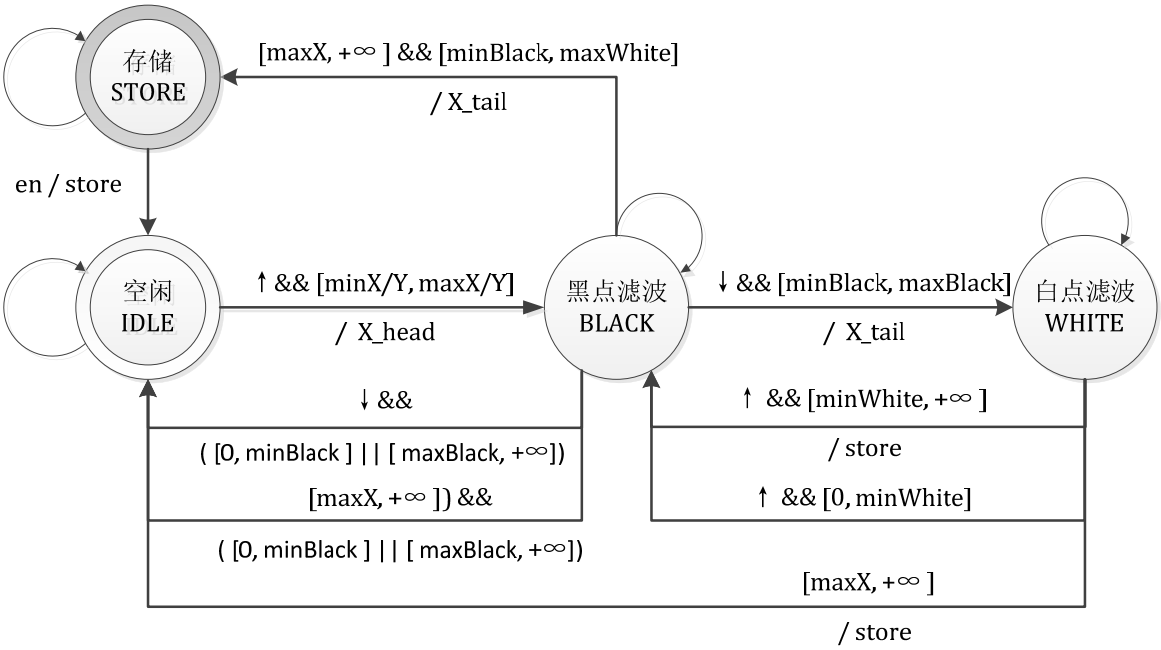


图 4.12 针对前向引导黑线图像细化算法，/代表输出

由于原算法采用高级语言实现，当用移植到 FPGA 上时，采用不同的方式其开发周期也不尽相同。当算法复杂度越高时，高级语言建模能力越强，图形化输入工具如原理图输

入、XSG 的 DFG 等较难描述，采用 HDL 则需要设计有效的状态机。

这是实现的第二类状态机，依靠状态转换条件输出。跳转瞬间依赖输入信号，而状态与跳转信号作为输出信号跳转条件，不同于常规状态机的规律性。比如，状态机状态数不多，但跳转条件很多，而且输出信号从同一状态跳转到下一同一状态，不同跳转条件输出不同但状态机跳转相同。

复位后进入空闲态，如果检测到黑点则进入黑点状态；

黑点状态如果检测到换行则进入存储态；检测到白点则进入白点状态；

白点状态若检测到换行则进入空闲态；检测到黑点则进入黑点状态；

存储状态若检测到行同步为，则回到空闲状态。

以上过程还需使用摄像头定位校准对应的每行连续黑点宽度数组。

4.5.2 XSG 仿真平台设计

针对数字图像处理的特点，可以采用 Mathworks 与 Xilinx 开发的 System Generator 平台进行高级仿真，将编写好的 HDL 代码导入到黑盒中进行测试。图像输入和基本处理由 XSG 完成，而内部处理则由黑盒中的 HDL 代码完成，输出结果包含使能信号和数据，导出到 Matlab 环境中通过 M 文件进步处理，提取出已经细化压缩过的黑点，还原出原始数据。

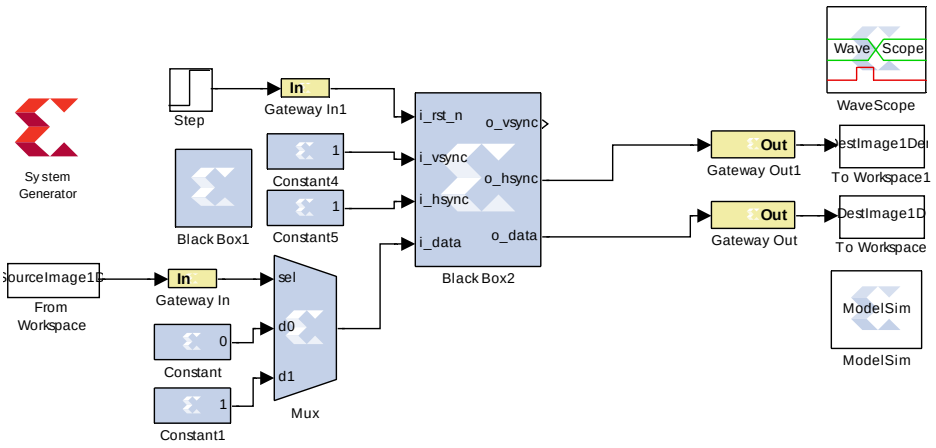


图 4.13 图像细化的 XSG 黑盒测试模型

原始图像、二值化图像、FPGA 处理后的图像、软件处理图像分别如下图所示，其中处理后的图像采用二值化、细化、坐标转换、压缩存储，可见每行仅有一个点，提取了赛道黑线的中点，同时排除干扰，较好的复现赛道特征，压缩后的黑点数据较原图大大减小，为程序的进一步处理节省很多时间，而软硬件测试结果基本一致。这样的软硬件划分，能大大提高数据处理流与控制算法流各自的处理效率，充分利用 FPGA 的数据并行流水线处理和 CPU 的任务调度函数运算能力。测试结果如下图：

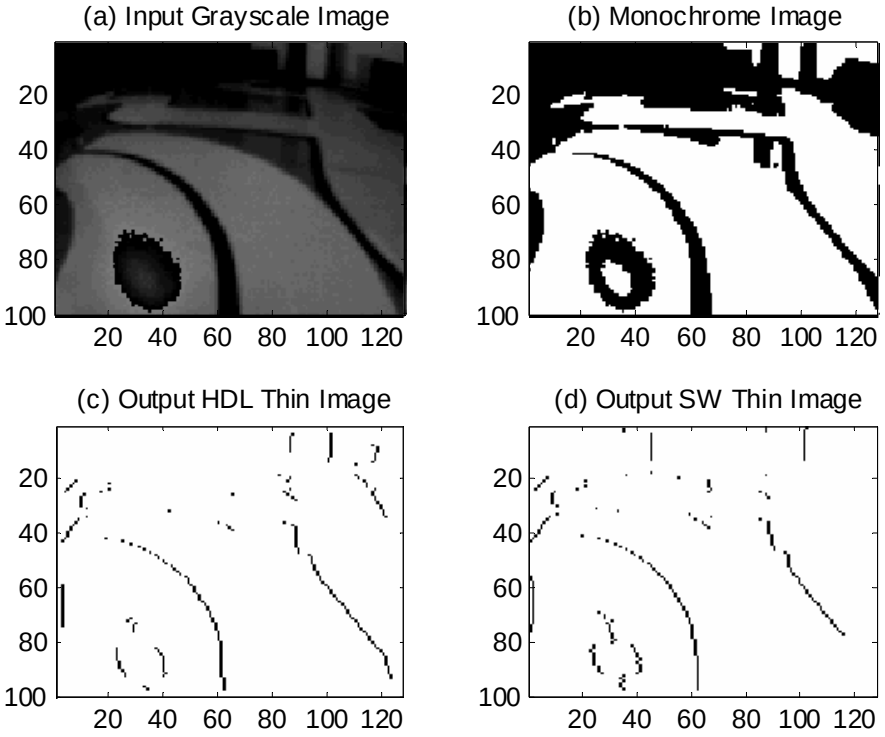


图 4.14 软硬件细化图像的对比

5 双闭环运动控制协处理器设计

5.1 直流电机双闭环控制系统原理

为了实现直流电机转速和转矩的精确控制，需要在运动控制系统中引入速度负反馈控制环和电流负反馈控制环，如下图：

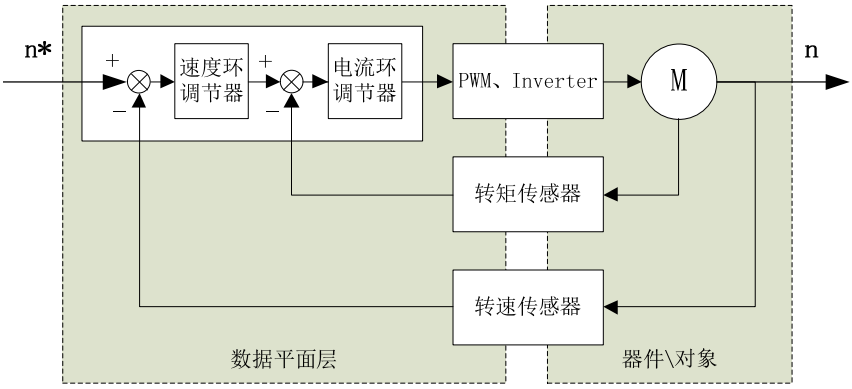


图 5.1 电机速度环、电流环双闭环控制框图

双闭环实现的理论控制波形如下图所示[54]：

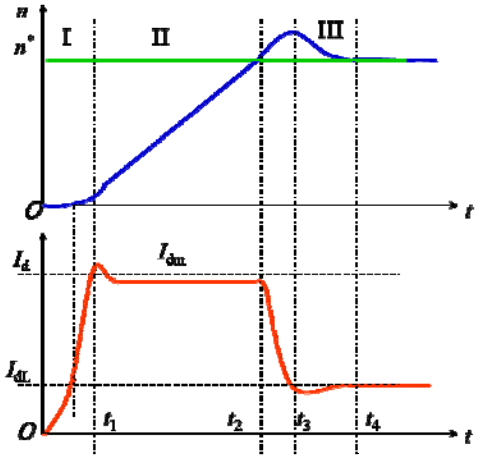


图 5.2 双闭环调速系统起动过程的转速和电流波形

可见电机启动时电流环起主要作用，电机速度稳定时速度环起主要作用。电流环的主要功能为启动过程的加速和抗电压扰动，转速环的主要功能为实现转速无静差和抗负载扰

动。速度环和电流环调节器实现方法较多，根据各环节要求，选取了 PI、滞环、PDFF、Mamdani 模糊、T-S 模糊、自适应模糊等算法进行比较，测试结果如以下两表：

表 5.1 各种控制器的综合性能、优缺点比较

控制器	优点	缺点
常规 PI	快速跟踪，抗负载扰动	超调，有静差，参数适应性差
Mamdani 模糊	综合性能最好	稍微慢
T-S 模糊	容易实现	超调，参数适应性一般
自适应模糊	参数适应性最好	比较慢

表 5.2 双闭环实现各种调节器的综合性能、优缺点比较

控制器		优点	缺点
电流环	滞环	简单，快速跟踪，抗负载扰动	超调，有静差、纹波，频繁开关
	PI	综合性能良好，简单通用	动态静态性能不能兼顾
速度环	PI	简单通用	比较慢
	PDFF	综合性能最好	无明显缺点

可见常规 PI 控制器基本可以满足双闭环要求，考虑在 FPGA 实现复杂度，速度环、电流环均使用 PI 调节器实现，D 参数寄存器设置为 0 即可。

5.2 协处理系统框图

使用 FPGA 实现运动控制系统有着相当大的优势，因为使用 PI 或者 PID 调节器实现对电机的速度闭环或者速度、电流双闭环计算有着相对固定的计算格式，并且逻辑判断相对较少，同时，运动控制系统使用速度、电流双闭环控制时对计算速度要求高。

由于运动控制模块的内环输入输出带宽远大于外部指令输入，且处理周期远小于外部指令输入，运动控制模块相当于运行在一个数据平面中，实时控制流层 CPU 只是周期性唤醒该模块进行低比特率通信，因此数据平面层的运动控制模块相当于实时控制流层 CPU 的

协处理器。按照功能，系处理器分为通用总线接口模块、寄存器堆/同步单元模块、双闭环运动控制功能模块。其中双闭环运动控制功能模块分为：速度闭环定点 PID、电流闭环定点 PID、PWM 模块、编码器正交解码、AD 芯片 SPI 接口等。

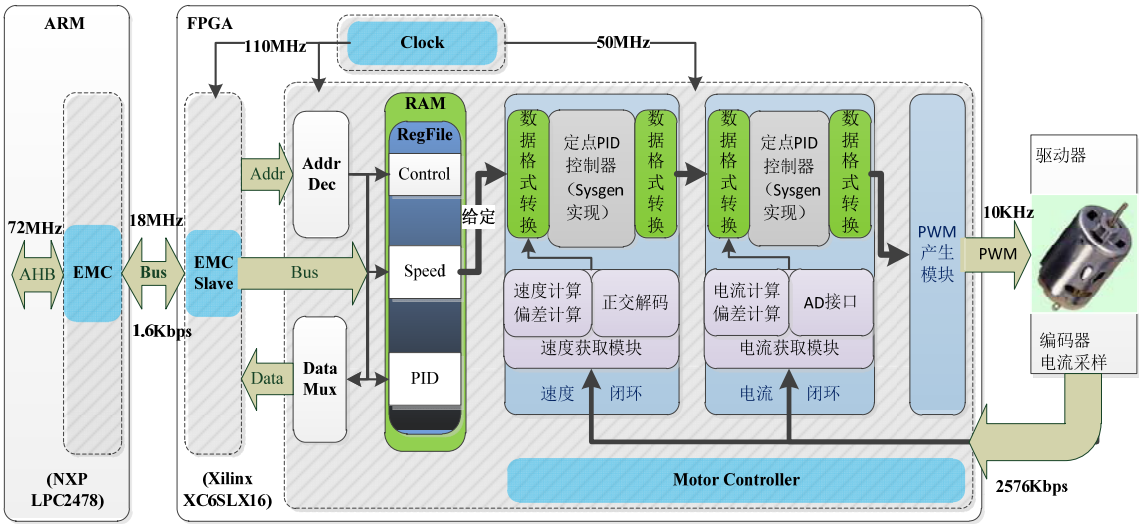


图 5.3　双闭环运动控制协处理器框图

5.3 寄存器堆

5.3.1　ARM 存储地址空间映射

表 5.3 以其中一路为例

方向	注	addr	reg		7:MSB	bit		0:LSB
总线读写	0	16'h0020	PWM	CTRL	reserve			
					reserve	POL	CAE	ENABLE
总线读写	1	16'h0022		PREDIV	reserve			
					PREDIV[7:0]			
总线读写	2	16'h0024		PERIOD	PERIOD[15:8]			
					PERIOD[7:0]			
总线读写	3	16'h0028	speed	REF	data[15:0]			
					data[7:0]			
总线只读	4	16'h002a		CUR	data[15:0]			
					data[7:0]			

总线只读	5	16'h002c		ERR	data[15:0]
					data[7:0]
总线读写	6	16'h0030	PID_speed	P	data[15:0]
					data[7:0]
总线读写	7	16'h0032		I	data[15:0]
					data[7:0]
总线读写	8	16'h0034		D	data[15:0]
					data[7:0]

5.3.2 功能说明

（1） 编号 0：

控制寄存器，可读写。

ENABLE——使能运动控制模块，高有效。

（2） 编号 1：

PWM 模块预分频系数。可读写。

（3） 编号 2：

PWM 周期。可读写。写入后执行延迟一周期，通过 PWM 内部状态机实现，避免功率器件受损。

（4） 编号 3、4、5：

速度环参考给定值，可读写；当前检测值，只读；当前误差值；只读。

（5） 编号 6~8：

速度环 PID 系数 P、I、D。可读写。

5.4 PID 控制器实现

5.4.1 PID 离散化表达式

PID 调节器是一种线性调节器，这种调节器是将设定值 r 和实际输出值 y 进行比较，

构成控制偏差 e=r-y，并将其比例、积分和微分（偏差的现在、过去、将来）通过线性组合构成控制量[55]。PID 控制是一种负反馈控制。本课题 PID 控制器设计需求包括能够实现定点参数输入、积分饱和限制、便于参数整定等。

考虑离散化 PID 计算公式[56]：

公式 1：位置 PID

$$u(k) = K_P\left[e(k) + \frac{T}{T_I}\sum_{i=0}^{k} e(i) + T_D\frac{e(k)-e(k-1)}{T}\right] \tag{5-1}$$

公式 2：增量式 PID

$$\Delta u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{5-2}$$

其中：

$$q_0 = K_P(1+\frac{T}{T_I}+\frac{T_D}{T}),\ q_1 = -K_P(1+\frac{2T_D}{T}),\ q_2 = K_P\frac{T_D}{T} \tag{5-3}$$

可见增量式 PID 控制算法提供执行机构的增量△u(k) ，而采用第二种算法时，从程序上处理只需要保持 3 个时刻的偏差值即可。

5.4.2 输出限幅功能

在位置式算法中，由于积分环节与历史数据相关，初始误差将会造成错误累加，容易出现积分饱和，将叠加到最终输出结果上。如果该值超过了执行元件所允许的最大限度，那么实际上实现的控制量将是受到限制的值，容易出现较大误差。因此，有必要对位置式 PID 进行积分限幅。

在增量式算法中，特别在给定值发生阶跃时，有算法的比例部分和微分部分计算出的控制量增量可能比较大（一般情况下 T/T_I 的值要小得多；积分部分的值相对比较小）[57]。如果该值超过了执行元件所允许的最大限度，那么实际上实现的控制增量将是受到限制的值，计算值的多余信息没有执行就遗失了。因此，有必要对增量式 PID 进行输出限幅。

5.4.3 A3 原则设计 PID 调节器的 FPGA 实现

位置式与增量式 PID 公式的数据流图分别如下所示：

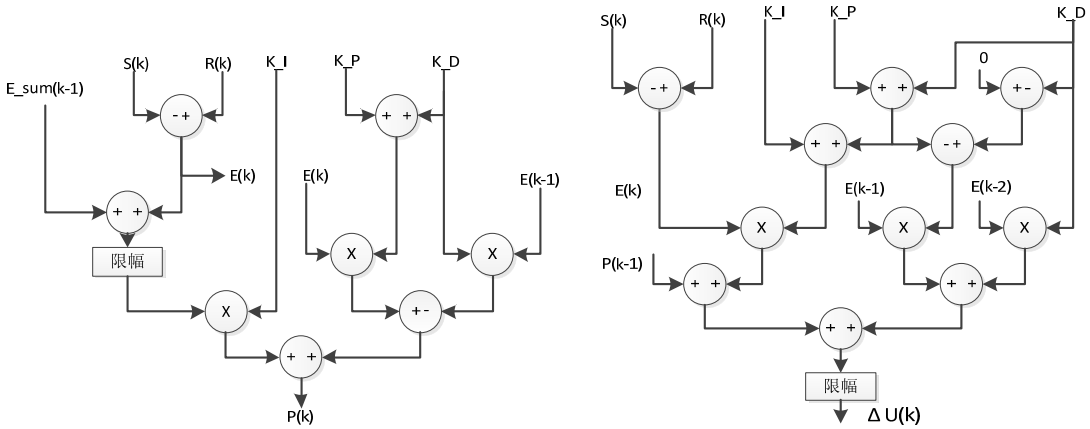


图 5.4 位置式 PID 调节器与增量式 PID 调节器数据流图

表 5.4 位置式 PID、增量式 PID 的 DFG 实现对比

方法	计算时间（拍）	资源消耗
位置式 PID	4	3 个乘法器，5 个加法器
增量式 PID	5	3 个乘法器，8 个加法器

左边为位置式 PID 调节器 DFG，右边为增量式 PID 调节器 DFG。对比如下表，可见选用位置式 PID 可以减少资源消耗和计算时间，所以选用位置式 PID。

使用 XSG 工具，根据数据流图搭建好 PID 控制器计算模块如下图所示。由于模块中的积分限幅单元使用 sysgen 实现效率较低，并且难以实现，这里考虑使用 sysgen 的 BlackBox 调用 HDL 语言实现限幅单元。然后使用 sysgen 生成网表文件，顶层文件加入整体工程，便于整体工程使用。

考虑精度等问题，要求 PID 控制器的比例（Kp）、积分（Ki）、微分（Kd）参数使用定点书实现。定点格式为 Fix_16_8。

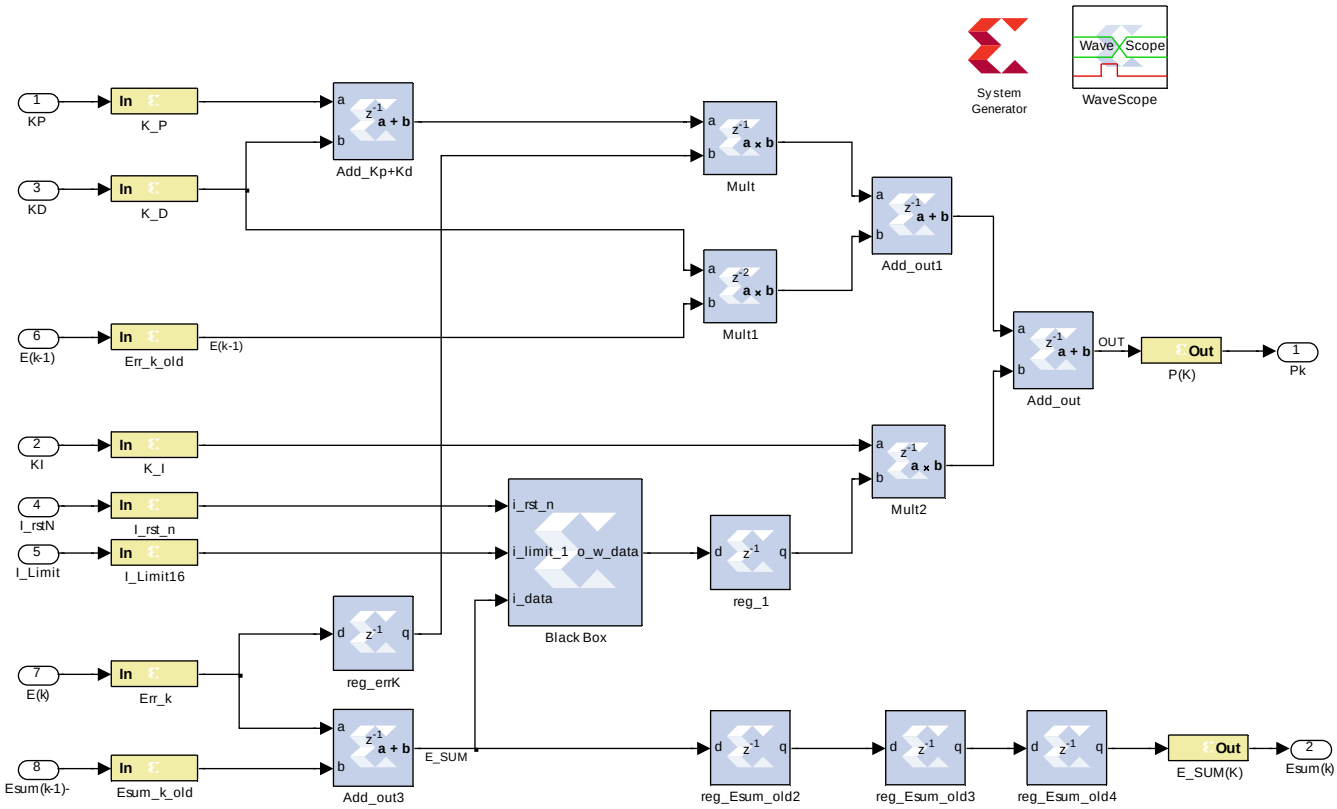


图 5.5　PID 调节器的 XSG 建模

表 5.5　PID 模块接口格式说明

Name	Direction	HDL Type	XSG Type	作用
k_d	in	[15:0]	Fix_16_8	PID 常量参数
k_i	in	[15:0]	Fix_16_8	
k_p	in	[15:0]	Fix_16_8	
e_sum_k	out	[15:0]	Fix_16_0	闭环输入输出
err_k	in	[15:0]	Fix_16_0	K 次误差输入
err_k_old	in	[15:0]	Fix_16_0	K-1次误差输入
esum_k_old	in	[15:0]	Fix_16_0	K-1次误差积分输入
i_limit16	in	[15:0]	Fix_16_0	限幅输入
i_rst_n	in	[0:0]	UFix_1_0	复位输入
p_k	out	[34:0]	Fix_35_8	K 次 PID 调节输出

5.5 双闭环直流调速模块实现

5.5.1 顶层状态机设计

速度获取模块使用正交解码获取当前编码器转速和方向，与给定速度求差后速度环PID调节器，积分限幅输出后经过低通滤波器与采集电流值求差，输出到电流环PID调节器，然后输出到PWM模块，产生PWM波控制驱动器，AD芯片SPI接口反馈采样的电流量化值到电环偏差模块，最终完成双闭环控制。

需要注意的是，由于XSG输入输出模块以外的matlab环境中数据都是双精度浮点类型，因此PID环节在FPGA中实现前需要经过XSG的双精度浮点到定点转换模块，定点格式见上节表。

本设计方案使用状态机控制PID运算时间，设计状态机为顺序状态机，在状态机的每一个状态进行相关计算，并可以通过状态机的切换时间控制相关状态下的计算时间消耗。状态机状态转移图如下图所示。

5.5.2 系统级仿真

XSG的数据流模型搭建好外围浮点接口以后可以直接集成在Matlab Simulink环境中，并能结合HDL黑盒模块完成系统级仿真。构建好电机模型，电机参数R表示电机内阻、T_l表示电机机电时间常数、T_m表示电机轴上转动惯量，Ce表示电机在额定磁通下的电动势系数[58]。

根据轮式机器人平台参数以及电机参数，可以计算出一套仿真使用参数为R = 0.33Ω，T_l = 0.000619，T_m = 0.85，Ce = 0.00022048。

速度环PID参数如下：Kp=3.8，Ki=0.31，Kd=0；

电流环PID参数如下：Kp=20，Ki=8，Kd=0；

实现的XSG 与 Simulink模型如下：

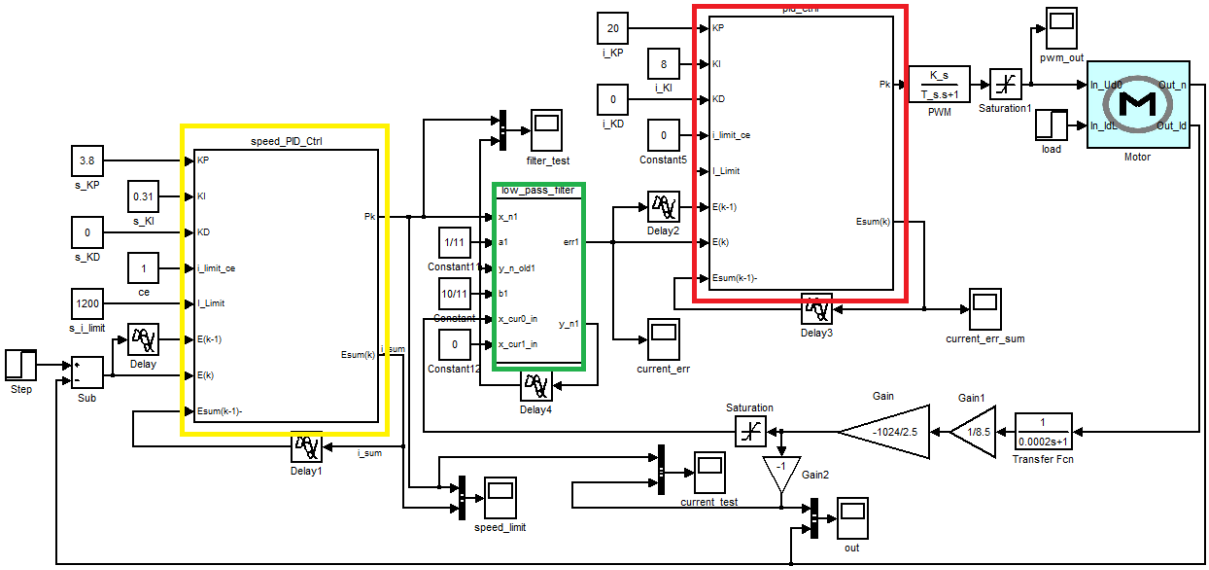


图 5.6　FPGA 实现双闭环直流调速的 Simulink 系统级仿真建模

两个较大的子系统模块为 PID 调节器模块，采用 XSG 的数据流建模实现；中间小子系统模块为低通滤波器和电流偏差计算模块，采用 HDL 语言实现并导入到 XSG 黑盒。设定 XSG 仿真周期为 10uS，保证在一次电流控制中可以完成一次 PID 计算。仿真结果如下图所示：

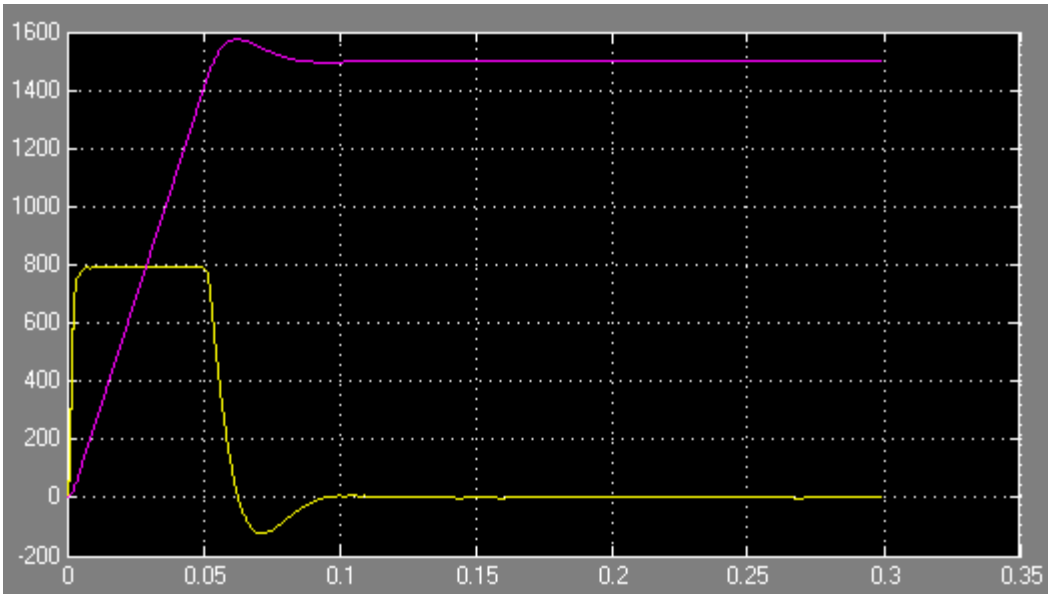


图 5.7　FPGA 实现双闭环直流调速的系统级仿真波形。显示每大格单位：速度环 200r/min/div；电流环 2.15A/div；时间轴 50ms/div

6 点阵液晶显示后处理器设计

6.1 FPGA 需要控制的物理接口

6.1.1 LCM 接口：

输入地址总线：CSC， CSB， CSA；

输入控制总线：复位 RST#， 触发信号 E， 指令/数据信号 RS， 读/写信号 R/W;

双向数据总线：DB0-DB7。

6.1.2 背光和对比度接口

LCM 的背光和对比度通过 X9313 系列电子电位器[59]调节，程序控制 INC 引脚脉冲、UD 引脚方向来调节 RW 端位置，5bit 计数器值可控制 32 级。

6.2 后处理系统框图

对软件而言，处理点阵液晶大量数据所需时间较长。引入映射缓冲区这一思想，可以把 LCM 接口的物理接口部分和数据部分分开。实时控制流层 CPU 只读写映射区的数据，利用 C 语言逻辑描述优势实现复杂的画图功能；而数据平面层 FPGA 负责液晶后续的具体时序要求，实现一个 CPU 的后处理器，利用 HDL 并行优势在数据间插入复杂的时序操作，系统通用性、实时性大大增强。同时，引入双口 RAM，则可实现 ARM 的快速数据通道 EMC 接口与 LCM 的慢数据通道接口的跨时钟域设计，有效提高系统的同步性、模块化和高效性。

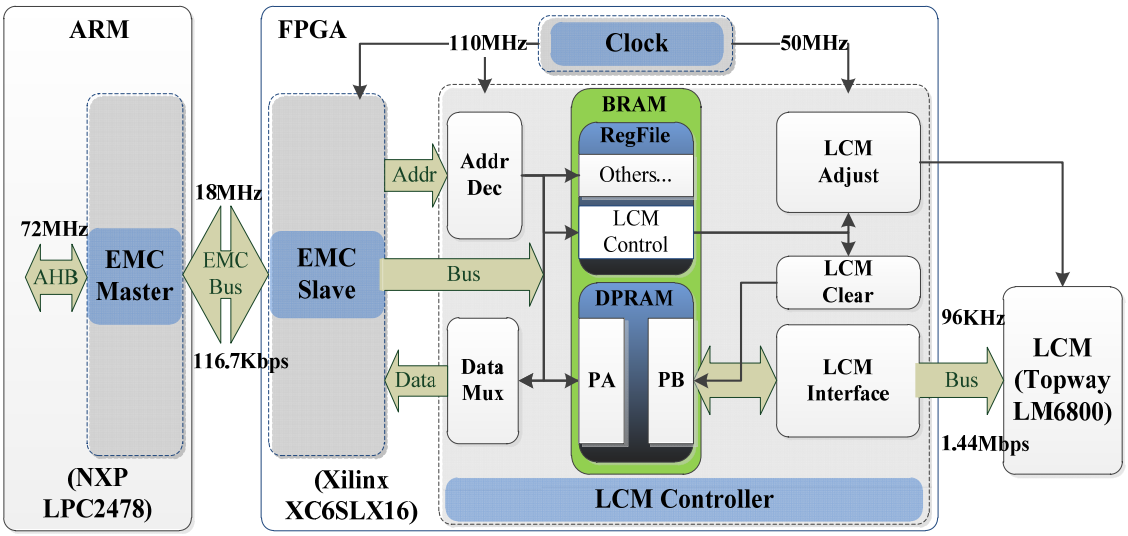


图 6.1　点阵液晶显示后处理器系统各环节吞吐量

6.3 寄存器堆 RegFile

6.3.1　ARM 存储空间地址映射

表 6.1 液晶寄存器堆地址映射

方向	注	addr	reg		7:MSB	bit		0:LSB
双端读写	0	16'h0010	LCM	CTRL	reserve			
					reserve	INIT	CLRRAM	RSTB
双端读写	1	16'h0012		ADJ_BLA	reserve	MODE	MINUS	PLUS
					data[7:0]			
双端读写	2	16'h0014		ADJ_CTRST	reserve	MODE	MINUS	PLUS
					data[7:0]			

6.3.2　功能说明

（1） 地址 0xh0010：

功能：控制寄存器 Reg_LCMCtrl，两种清屏方式和液晶初始化控制。

RSTB——双口 RAM 的 B 端口输出寄存器硬件清零；

CLRRAM——双口 RAM 的 B 端口写使能，写数据总线拉低。

INIT——从低到高的边沿处开始进行液晶初始化。

（2） 地址 0xh0012：

功能：背光电子电位器寄存器 Reg_LCMAdj_BLA，两种调节方式：增量加减和绝对值计数，通过 mode 选择。

表 6.2 电子电位器模块模式寄存器配置

mode=0	Plus=1	Minus=1
	电位器计数值+1	电位器计数值-1
mode=1	Data[7:0]	
	电位器计数值=data[4:0]	

（3） 地址 0xh0014：

功能：对比度电子电位器寄存器 Reg_LCMAdj_Ctrst，两种调节方式，同上。

6.4 液晶时序物理接口 LCMIF 设计

6.4.1 时序要求

（1）E 时钟的建立保持时间：

数据表时序图显示，读写对 E 锁存信号与其他数据总线、地址总线的时序要求相同：

写：E 上升沿之前需要地址总线有相应建立时间，E 下降沿之后地址总线、数据总线有相应保持时间。

读：E 上升沿之前需要地址总线有相应建立时间，E 下降沿之后地址总线、数据总线有相应保持时间。

（2）忙时间

可见尽管 E 脉宽要求很低（700ns），但对 E 脉冲之后的等待时间要求很高，大大降低了系统效率，速度远慢于前级 ARM 和 FPGA 对映射区操作频率。因此需要进一步降低 E

频率，并减小 E 占空比。数据或指令写入周期为

$$T_{write} = T_E + T_{Busy} = 0.75us + [T_{clk}, 3T_{clk}] \quad (6\text{-}1)$$

经过实际测试发现，$T_{write} = 10.4us$ 是可以满足的最低时间。

6.4.2 状态机设计

（1）状态设计

映射区组织结构采用逐行扫描方式，需要改变 CSC、CSB、CSA 时序。写满屏幕时发送到 LCM 的完整流程数据序列各阶段如下：

a) RST：

LCM 复位时，CSC 由高变低，LCM 四个显示块被依次选中，状态机结束 RST 状态，转向 INIT 状态，

b) INIT0：

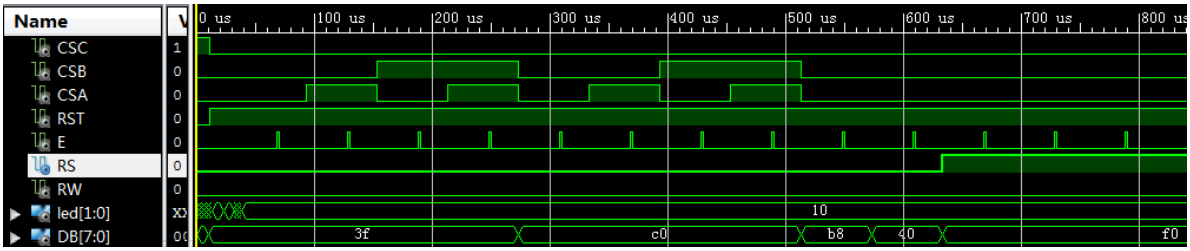


图 6.2 液晶外部接口初始化时序图

开始每块写命令字 DispOn=0x3f，写完后再开始每块依次写 Z 起始行命令字 0xc0。每个 CS 周期为模 4 计数器，CSC 拉低，CSB 四分频，CSA 二分频。

c) INIT1：

写入 Z 起始行命令字 ZeroAddr=0xc0。

d) DISP_INSTR0（CS++,X++,Y0=0,DB=？）：

写入 Y 坐标清零命令字=0xb8。

每次写 64 字节数据前会写 X、Y 坐标定位命令字。Y 固定为 0x40，相当于给 LCM 的

Y 计数器复位清零；X 则计数 8 次后清零，重载值为 0xb8。

e) DISP_INSTR1（CS++,X++,Y0=0,DB=？）：

写入 Y 坐标清零命令字=0x40。

f) DISP_RAM（CS++,X++,Y0=0,DB=？）：

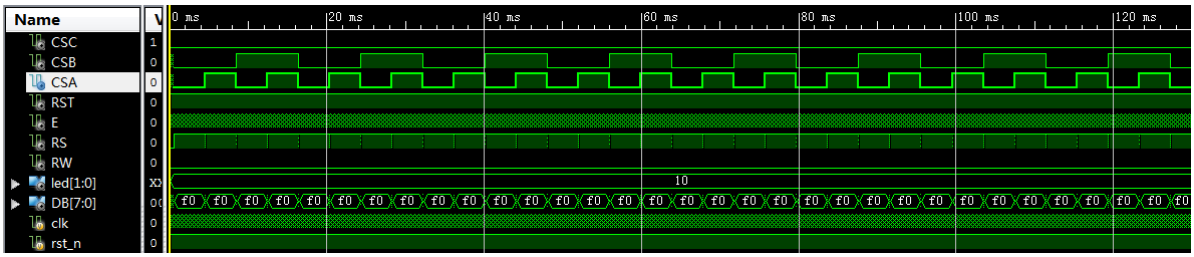


图 6.3 液晶外部接口单帧数据时序图

由于双口 RAM 中初始写入的顺序是逐行扫描，则每一行需要换块 4 次，{CSB,CSA} 的二进制计数满为一个行周期，64/8=8 行需要 8 个行周期，因此。可以看到上图中 CSB 计数 8 次。64 字节数据与 2 字节命令由 RS 引脚信号区分，RS=1 时读写数据，RS=0 时读写命令。

（2）状态转换图

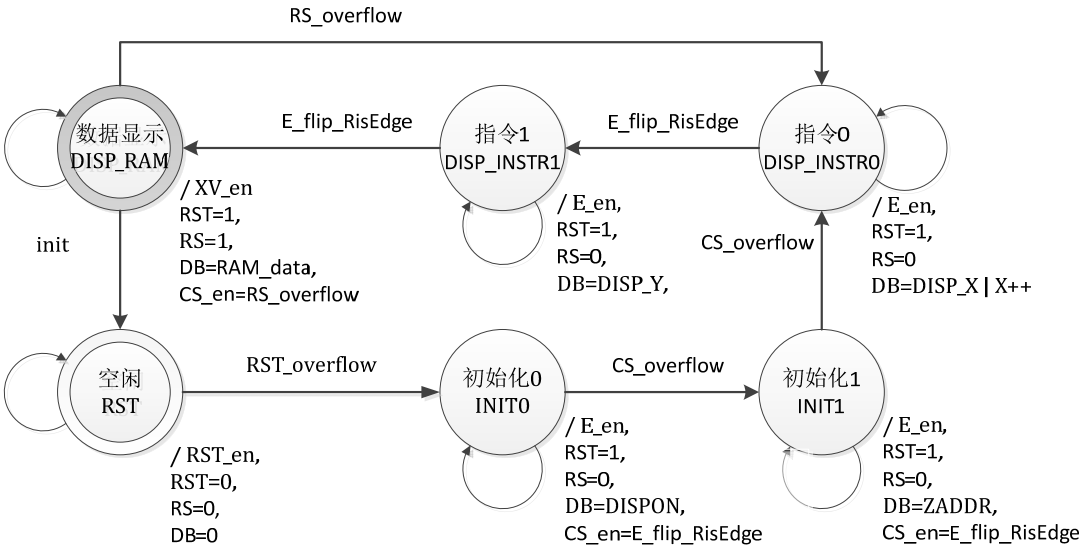


图 6.4 液晶接口模块状态转换图，/代表输出

这是实现的第三类状态机，状态本身作为输出。可见分为复位、初始化 0、初始化 1、

x 坐标指令、y 坐标指令、数据显示 5 个状态。其中 x、y 以及数据显示 3 个状态循环跳转。

6.4.3 程序设计

（1）锁存时钟 E

E 需要建立保持时间而非与其他信号对齐。为此，可以生成沿对齐的时钟 E_clk 驱动其他信号，然后单独生成与 E_clk 同频率并满足建立保持时间的 E。

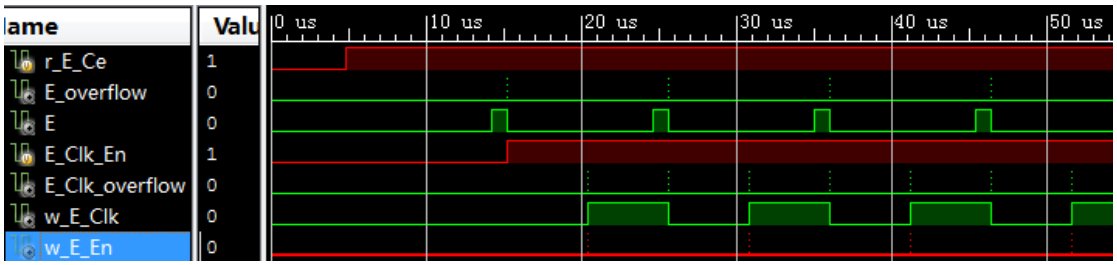


图 6.5 液晶接口模块引脚测试时序图

E 是为 LCM 单独的输出脉冲锁存信号，由计数器生成，在 RST 结束后使能，输出占空比约 10%，周期约 10.4us。而 E_Clk 为 LCM 其他信号做基准使能，由于需要满足建立保持时间和写入等待时间，E_Clk 滞后 E 相位 90°。

（2）片选{CSC,CSB,CSA}

RST 状态：{CSC,CSB,CSA}=CS_RST={1, 0, 0}，相当于片选无效；

INIT 状态：{CSC,CSB,CSA}=CS_INIT，为 8 个 E 周期的 2 进制加计数，溢出值为 0x3；

DISP 状态：{CSC,CSB,CSA}=CS_DISP，为 4*66 个 E 周期的 2 进制加计数，溢出值为 0x3。

（3）X 及 Y 命令字

在 DISP 状态有效，每一屏数据需要在每块、每行 RS=0 时插入 X、Y 坐标命令，按照逐行扫描的方法，每行插入 4 个（X+Y），每屏插入 4*8 个（X+Y）。则计数规律为：

X：每屏遍历 32 块加计数 8 次，然后重载；

Y：每次换块都清零，因此用常量即可。

7 实验结果及分析

7.1 性能与资源利用结果

布线后无时序错误，实现最大时钟频率与资源利用如下表：

表 7.1 XC6SLX16 实现系统的资源使用与性能

Device Utilization Summary					
Device Part Number	XC6SLX16-2FTG256C		Fmax(MHz)	166	
Slice Logic Utilization		Used	Available	Utilization	Note(s)
Number of Slice Registers		4,705	18,224	25%	
Number of Slice LUTs		4,074	9112	44%	
Number of occupied Slices		1,640	2,278	71%	
Number of RAMB16BWERs		4	32	12%	
Number of RAMB8BWERs		28	64	43%	
Number of DSP48A1s		21	32	65%	
Average Fanout of Non-Clock Nets		3.2			

7.2 利用逻辑分析仪 ChipScope 测试 EMC 总线从机时序

（1）EMC 写数据到双口 RAM：

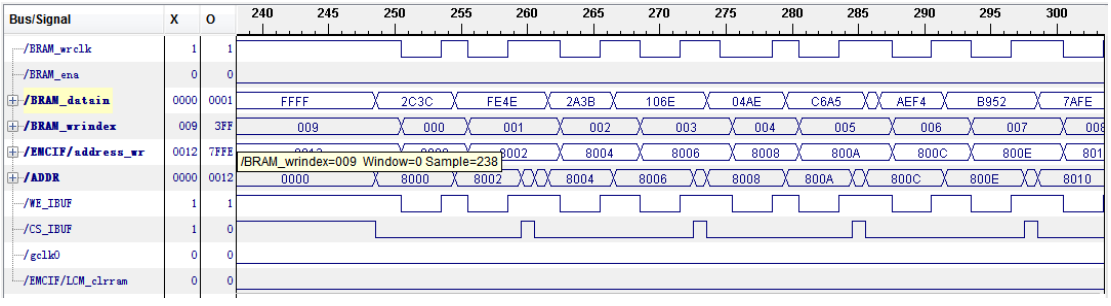


图 7.1 Chipscope 测试 EMC 总线写数据时序图，32bit 分为两次 16bit 发送

可见 EMC 采用 32bit 数据总线写数据时，对于 16bit 的 FPGA 从双口 RAM，一个 CS 周期有两个 WE 脉冲，而 WE 脉冲在数据总线 DATA 和地址总线 ADDR 的有效范围内，建立保持时间由 EMC 主机保证。

（2）EMC 从双口 RAM 读数据，多周期读：

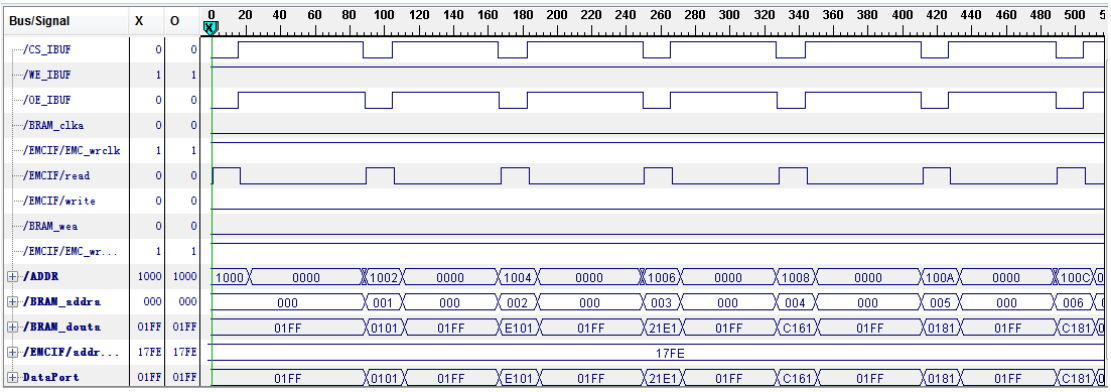


图 7.2 Chipscope 测试 EMC 总线读数据时序图，16bit 接收

可见 EMC 采用 16bit 数据总线读数据时，对于 16bit 的 FPGA 从双口 RAM，一个 CS 周期有一个 OE 脉冲，而 OE 的下降沿比 FPGA 数据出的数据总线 DATA 有效读数据早，上升沿在数据总线 DATA 和地址总线 ADDR 的有效范围内，建立保持时间由 FPGA 保证。

7.3 整体运行结果与分析

7.3.1 基于机器视觉的自主导航控制系统

（1） 数字图像处理完整流程与结果

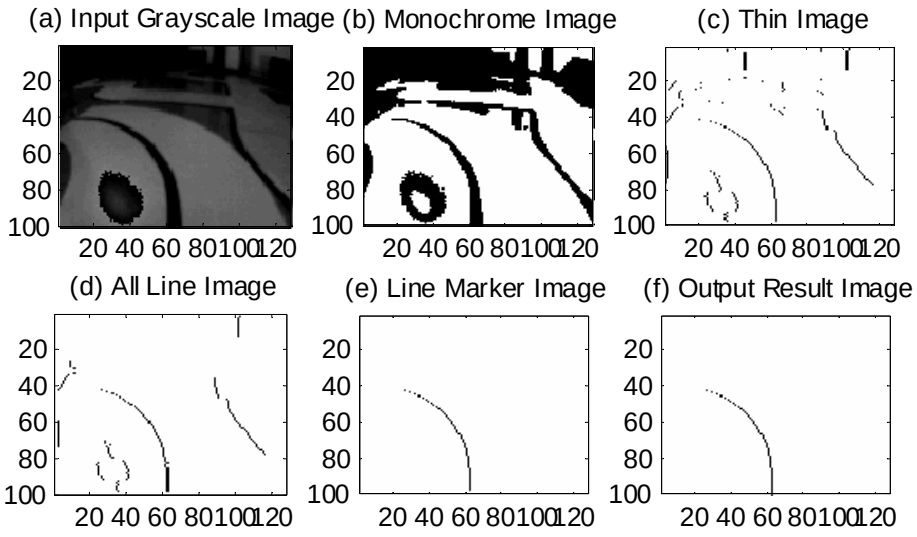


图 7.3 数字图像处理流程结果：(a)输入灰度图(b)动态阈值分割(c)图像细化(d)引导线滤波(e)引导线

匹配(f)输出引导线结果。

由图中可见，通过数据平面层 FPGA 数字图像预处理器和实时控制流层 CPU 软件的后处理，道路中央引导线被完整识别出来，而且 CPU 处理数据量大大降低，完成了机器视觉的识别过程。

（2） 数字图像处理与液晶显示

将轮式机器人平台放置于下图复杂赛道环境进行验证。

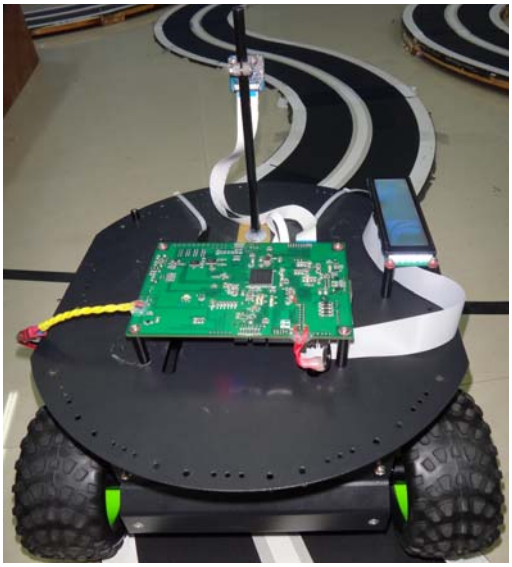


图 7.4 基于软硬件协同处理的轮式机器人平台测试环境

图 7.5 液晶显数字图像处理图片及监控参数，动态阈值、细化处理、图像压缩。

动态阈值分割图像、细化图像、参数如图所示，其中处理后的图像采用提取、二值化、细化、坐标转换、压缩存储。可见当前光照下，动态阈值模块产生的阈值 Tr 为 38，而细

化结果为每行仅存储连续亮线的一个点，提取出了赛道标线的中点并率除了大多数干扰，软件进行后续的标线匹配算法。这样的处理流程和软硬件划分，能大大提高数据处理流与控制算法流各自的处理效率，充分利用 FPGA 的数据并行流水线处理和 CPU 的任务调度函数运算能力。

点阵液晶作为车载平台，可以实时显示数字图像处理部分的动态阈值二值化图像、细化图像和参数。密集数据由 FPGA 完成数字图像预处理，压缩后由 ARM 完成显示源处理，再由 FPGA 完成显示后处理，完成了软硬件协同处理过程。

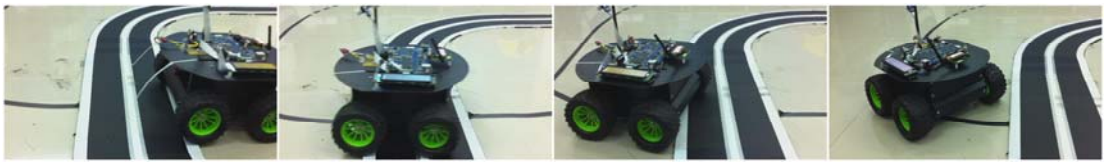
（3） 自主巡线导航测试

实时运行照片如下：

(a) 虚线干扰后进入蛇形弯

(b)蛇形弯后排除十字交叉线干扰

(c)复杂环境排除干扰，并跨越障碍



(d)复杂环境排除干扰，跨越障碍后进入直角弯

图 7.6 轮式机器人复杂环境自主导航测试

模型车实际运行的部分特性曲线如下图所示。在导引线最小曲率半径为 50cm 情况下，设置参考速度 20cm/s，复杂引导线平均导航速度达一致。根据偏差融合计算出的车体位置偏移 x 与车体偏转角，可知前者最大也仅为 5cm，而后者非常小，车体始终实时紧密跟踪导引线，速度、稳定性及精度都达到良好的效果。

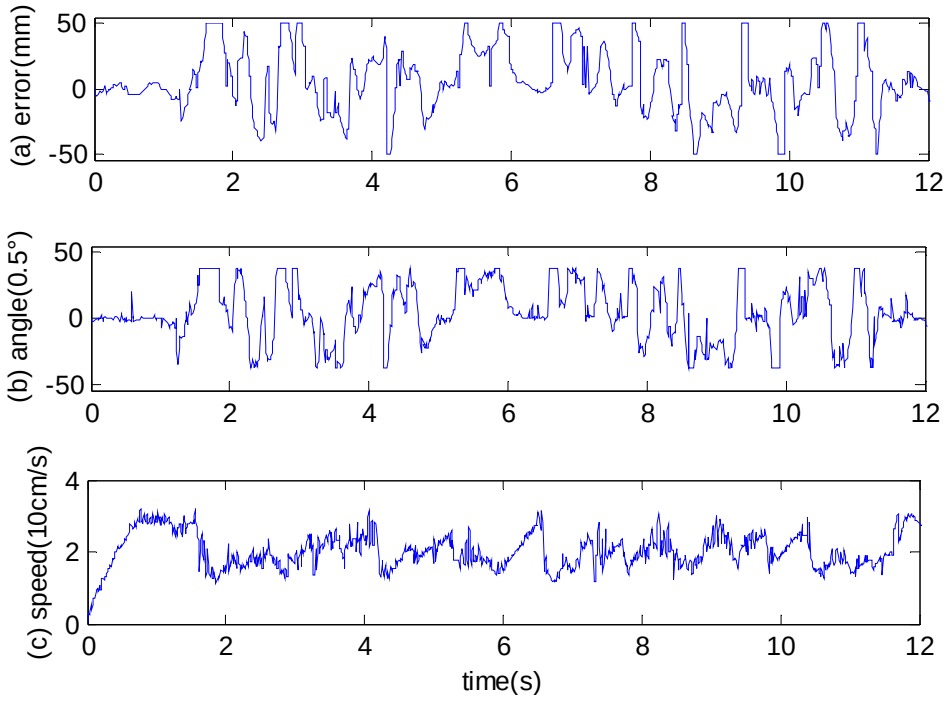


图 7.7　轮式机器人运行检测值：(a)横向偏移(b)横向角度(c)质心速度（设定 20cm/s）。

7.3.2　基于人工视觉的远程导航控制系统的智能终端控制与视频回传

（1）　控制指令传输协议(TCP 传输)

控制指令全部由主机发起，在一个 TCP 包内传输完毕。

格式：控制指令编码(2Bytes)+数据长度(2Bytes)+数据(长度不定)；

（2）　视频回传协议(UDP 传输)

一帧 100*128 实时 256 灰度图像数据分几个 UDP 包传递。

格式：控制指令编码(2Bytes)+数据长度(2Bytes)+[帧编号(4Bytes)+数据(长度不定)]

其中帧编号格式为最低 4bits 为一帧图像数据被拆分为 16 个小帧的编号，高 28bits 为

图像数据的编号，便于将图像数据拼接起来。

（3） 视频回传流程(考虑 android 客户端已经连接上机器人端点)：

传输视频：android 客户端给主端发送控制指令请求回传摄像头当前图像，从端收到消息后以 UDP 的方式按照视频回传的格式进行封包传输一帧图像回来，android 端此时启动一个超时定时器(超时时间为 1000/24=41.67ms)，并开始接收数据，根据封包的格式将一帧图像组合起来，不会请求重传，如果小帧图像丢失现象不严重就将图像显示出来，缺少的小帧图像用全白代替。

（4） 视频实时回传实现效果：

点击按钮后直接发送命令到 UDP 的主机端，主机端接收到数据后，将数据重新组装成帧，将一副图像数据分成 16 个小帧，传输出去，然后主机接到数据后封装起来，通过 UDP 传输，最后显示在图片控件上，如下图所示：

图 7.8 轮式机器人 Android 智能终端应用程序截图

左图中下方屏幕即为回传的视频截图，上部分为操作界面，达到了预期目标。

7.3.3 运动控制:

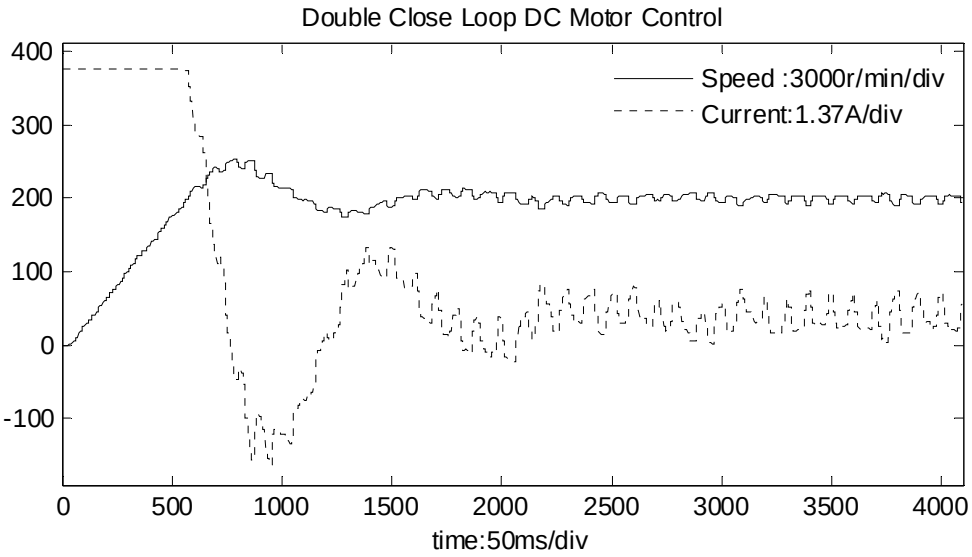


图 7.9 FPGA 实现的双闭环直流调速系统测试，时间单位 100us

构建好实际的 FPGA 程序，实现程序状态机，在 xc6lx16 实验板上实际测试电机运转性能，使用 xilinx 自带逻辑分析工具 ChipScope IP 核测试 FPGA 内部数据，在一个控制周期内生成一帧数据，使用 FPGA 内部的 BRAM 模块保存这些数据，在数据捕获完毕后，发回所有数据，得到双闭环调速系统电机启动时速度及电流响应波形如上图所示。

可见速度 PID 环节能有效达到指定速度，超调量较小，上升时间可以满足基本要求，没有稳态误差；而电流环在速度环上升阶段输出满幅，速度环达到指定速度后电流环输出下降，直到达到稳定负载转矩。利用 FPGA 实现的双闭环直流调速系统启动调速性能与动态性能良好。以上测试波形与 XSG 的系统级仿真波形基本一致，但速度环、电流环纹波较大，需要进步一步改进。

7.4 处理性能分析

7.4.1 数字图像预处理器

由于 FPGA 数字图像处理模块内部采用全流水线，布线后能达到的周期最小为 6ns，

即能达到 160MHz 左右。摄像头曝光时间配置为 10ms 一帧，由于行同步与帧同步时间不变，一帧有效时间为 1.7ms 左右。

（1） 图像细化应用的软硬件协同处理性能

项目设计了一种针对轮式移动机器人应用的前向图像细化算法，并在 FPGA 中通过全流水线的状态机实现，测试结果显示，将纯软件方式与软硬件协同设计方式结果对比，发现虽然前者将 100MHz 频率 CPU 程序存储于 32 位片内 SRAM 中，而后者将 72MHz 频率 CPU 程序存储在片外 16 位 SRAM 中，但是同样的算法，软硬件协同设计的方式下 CPU 占用时间却减少了 1000 多倍，充分显示出软硬件协同设计在复杂计算与数据处理方面的综合性能。

表 7.2 图像细化应用的时间消耗，软硬件协同设计对比纯软件

方法	CPU 主频	软件主存储器	软件响应时间(us)	加速比
纯软件	100MHz	32bit 片内 SRAM	16652（细化处理）	-
软硬件	72MHz	16bit 片外 SDRAM	14（EMC 传输）	1189.4

（2） 图像二值化应用的硬件处理性能

项目基于迭代法在 FPGA 中实现了高效的动态阈值分割，国内外文献比较少见，而测试结果显示，与同类算法的不同平台实现相比，其实现性能非常好。本项目以初始阈值 40 迭代两帧为例，频率 100MHz，像素同步频率 10MHz，第一帧原始数据花费 1.7ms，第二帧迭代数据花费 128us；而对比设计在 PC 机软件中实现[47]，初始阈值为 145，结果如下：

表 7.3 图像阈值分割应用的时间消耗，软硬件协同设计对比纯软件

方法	分辨率	帧速(fps)	吞吐率(Kbps)	处理时间(ms)	加速比
PC 软件	320*240	30	691.2(30%压缩)	21541	-
FPGA	128*100	100	1280.0	1.965	11012

可见，虽然对比设计数据吞吐率仅为本设计的一半，但软件方法实现迭代法的时间消耗却为 FPGA 实现的 104 数量级，足以可见用 FPGA 实现大运算量算法的优势，也体现出本设计创新的高效性，具有一定意义。

（3） 利用预处理器减少软件的处理数据量

原始图像为2bit控制信号和8bit数据信号，12800像素数，每秒100帧，需要12812.8Kbps的数据吞吐率，这对实时控制流层而言负担较重。而经过中值滤波、动态阈值分割、图像细化、黑点压缩以后传输的数据量为16bit黑点坐标数据，数据吞吐量减少到240Kbps，压缩比达到53.4，极大地减轻了实时控制流层软件负担，增强系统实时性和确定性。

表 7.4 图像预处理器输入输出带宽对比

	接口时间(/s)	吞吐率(Kbps)	吞吐率比
实时控制流层	1.4ms	240.0	-
数据平面层	170.0ms	12812.8	53.4

7.4.2 运动控制协处理器

（1） 对同样的数据流图PID控制器计算时间消耗。

纯软件测试环境：STM32系列ARM Cortex-M3控制器。

软硬件测试环境：NXP LPC2478 系列 ARM7 微控制器与 Xilinx Spartan-6 系列XC6SLX16 FPGA。

对同样的数据流图（限幅位置式 PID），运行在 72MHz 带 1 个硬件乘法器的 ARM Cortex-M3 微控制器完成一次运算需要 3.2us，而利用硬件并行性以及 50MHz 全流水线的FPGA 完成 4 个 PID 运算只需要 80ns，4 个电机的时间加速比达到 40 倍。

表 7.5 运动控制应用的时间消耗对比

方法	乘法器个数	频率（MHz）	处理时间（us）	时间加速比
Cotex-M3	1	72	3.20	-
FPGA	3	50	0.08	40

（2） 定点乘法运算。

纯软件测试环境：Freescale K60 系列 ARM Cortex-M4 微控制器，使用 CMSIS 库（Cortex

Microcontroller Software Interface Standard）实现。

软硬件测试环境：NXP LPC2478 系列 ARM7 微控制器与 Xilinx Spartan-6 系列 XC6SLX16 FPGA。

XSG 有高效的定点转换工具，而一般微控制器实现定点转换需要 CMSIS 库。运行在 100MHz 带 1 个硬件乘法器的 ARM Cortex-M3 微控制器完成一次定点转换需要 1us，而利用硬件并行性以及 50MHz 全流水线的 FPGA 完成一次 XSG 定点转换只需要 1 个周期，时间加速比达到 50 倍。

表 7.6 运动控制应用的时间消耗对比

方法	乘法器个数	频率（MHz）	处理时间（us）	时间加速比
Cotex-M3	1	100	1.00	-
FPGA	1	50	0.02	50

（3） 利用协处理器降低对软件吞吐率的要求。

协处理器能充分利用数据平面的特性，输入输出带宽相等且远大于中央处理器的给指令周期。对于双闭环直流调速系统的软硬件架构实现，速度环周期 1ms，电流环周期 0.1ms；电流环每 8 次采样取平均值以消除白噪声，每次采样 14*2bit；转速环每次采样 16bit；实时控制流层中央处理器每 10ms 给 16bit 参考速度。因此二者的吞吐率分别为 2576Kbps 和 1.6Kbps，运动控制协处理器吞吐率比达到 10^3 数量级。

表 7.7 运动控制协理器输入输出带宽对比

	控制周期（ms）	吞吐率(Kbps)	吞吐率比
实时控制流层	10.0	1.6	-
数据平面层	0.1	2256.0	1410

以上对比说明，协处理器主要特点是卸载了中央处理器的大部分数据处理工作，虽然周期加速比并不高，但是因为数据平面内部输入输出数据量很大，所以数据吞吐率非常高，极大提高了系统的数据处理能力。

7.4.3 液晶显示后处理器：

液晶控制应用从软件接口到液晶接口的数据流相应的频率和位宽如下：ARM 的 AHB 总线工作在 72MHz/32bit，EMC 工作在 18MHz/16bit，而液晶接口工作在 96KHz/8bit。

（1） 利用后处理器对软件进行加速

将纯软件方式与软硬件协同设计方式结果对比，发现虽然软硬件协同设计方式将 CPU 程序存储在片外 16 位 SRAM 中，而纯软件方式将 CPU 程序存储于 32 位片内 SRAM 中，但是软硬件协同设计的方式下 CPU 占用时间却减少了约 137 倍，充分显示出软硬件协同设计在复杂计算与数据处理方面的综合性能。

表 7.8 液晶控制应用的时间消耗，软硬件协同设计对比纯软件

方法	软件主存储器	帧周期(ms)	软件消耗时间(us)	加速比
纯软件	32bit 片内 SRAM	21.2600	21260	-
软硬件	16bit 片外 SDRAM	21.9648	155	137.2

（2） 利用后处理器增加软件输出的数据信息量

将软硬件协同设计方式结果对比，测试环境同上，软硬件协同设计的方式下吞吐率比减少了约 12 倍。

表 7.9 液晶后处理器输入输出带宽对比

	接口时间(/s)	吞吐率(Kbps)	吞吐率比
实时控制流层	7.1ms	116.7	-
数据平面层	1000.0ms	1440.0	12.3

7.4.4 三种拓扑关系的协同处理器的特点对比

从以上三种图拓扑关系的协同处理器性能分析结果来看，CPU 串行链路上的预处理器与后处理器的加速比很高，而数据吞吐率比不高；CPU 并行链路上的协处理器加速比不高，但是数据吞吐率比却非常高。

8 全文总结与展望

8.1 总结

本文在综合国内外软硬件协同设计技术基础上，设计了一种通用的软硬件协同处理架构，并应用到轮式机器人平台中。

所设计的软硬件协同处理架构能结合各自的优势，包括四个方面：层次参考模型、低两层拓扑关系、低两层的通用片内架构、软硬件接口。

（1）三个层次依次为数据平面层、实时控制流层、高性能处理层。高性能智能终端完成复杂的交互、应用和通信，ARM 作为专注实时控制任务，FPGA 负责数据密集型任务。

（2）数据平面层与实时控制流层的拓扑关系包含预处理器、协处理器、后处理器。预处理器减少了软件处理数据量，后处理器增加了软件输出信息量，而协处理器减轻软件数据吞吐负担。

（3）处理器间通信机制为 CPU 以松耦合方式通过 EMC 总线与 FPGA 通信，以地址空间映射方式实现了一种比较好的软硬件接口。

（4）FPGA 通用片内架构为总线互联形式的 IP，其结构包含总线接口模块、同步单元/存储器、数据通道功能模块。

同时，针对轮式机器人的具体需求，对两类控制系统进行了软硬件划分和 FPGA 设计。

（1）基于机器视觉的自主导航系统中，数字图像预处理主要实现了迭代法动态阈值分割和针对自主巡线应用的引导线图像细化，运动控制协处理器实现了双闭环直流调速系统，液晶后处理器实现了映射缓冲的几种 IP。以上过程采用 XSG 与 HDL 结合的开发方式，效率大大提升。

（2）基于人工视觉的远程操控系统中，数字图像预处理主要实现了原始图像存储，运动控制协处理器实现了双闭环直流调速系统。

通过仿真和测试结果可以看出，软硬件协同处理架构大大提升了系统性能，所设计的轮式机器人平台有良好的自主导航性能和应用扩展能力，非常适合关注性能但对成本敏感的嵌入式应用。平台的软硬件协同处理架构是一种层次结构、通用化、高性能的设计方案，在工业控制领域具有一定广泛意义和推广价值。

8.2 展望

软硬件协同设计作为一种新兴的嵌入式系统开发方法，在小型嵌入式控制系统中将会取得较好的效果。本文提出的层次参考模型、三种拓扑关系以及通用协处理片内结构，在轮式机器人应用中得到了较好的验证，同时结合软硬件协同设计、仿真的开发方法，使得该平台在工业控制领域具有一定广泛意义和推广价值。

但需要看到，系统还存在诸多问题。所设计的基于软硬件协同处理的轮式机器人平台系统比较简单，需要丰富的内容还有很多。运动控制部分还有很多工作要做。比如软硬件性能对比，双闭环直流调速系统相对其他两个协同处理的加速比很小，这类控制系统使用DSP 可以实现相当的性能。而随着控制算法的深化，FPGA 的真正优势将在复杂系统中体现出来，比如 PMSM、BLDC 等控制应用中，再比如模糊控制系统中。可以预计，复杂算法带来的加速比是很显著的，例如本文利用 FPGA 实现的两个图像处理算法。另外，速度环、电流环纹波比较大，需要改进，可能是电流环采样频率与实际系统相比不够，或者 HDL 实现的简单低通滤波器带宽选择不当。关于低通滤波器，考虑使用 Matlab XSG 的 FDA Tool（数字滤波器工具箱）进行更加专业的统级设软硬件系计。

还需要看到，FPGA 与 CPU 混合处理还存在一些需要解决的问题，如软硬件划分方法的标准、开发工具的系统集成、流程的简化等。针对这些问题，不少厂商正在努力，比如 Xilinx 提出了全新的设计框架 Vivado，以 IP 为中心实现可重用的软硬件协同设计流程；而 Altera 推出了 QSys 框架，以期实现片上网络系统，利用解决网络复杂拓扑的方案解决越来越拥挤的片上系统。相信在嵌入式技术的不断前进下，这些问题终将解决。

致谢

经过两年的学习与研究，在学校、老师、师兄、同学等多方的支持与协作下，我完成了硕士研究生的课题，撰写了硕士学位论文。作为一个多学科交叉与技术融合的系统，基于软硬件协同处理的轮式机器人平台的设计纷繁复杂，仅靠一个人的力量难以完成如此多的工作。在此向整个项目研究团队的开发成员以及在我研究生阶段给予各种帮助、建议与意见的人表示衷心的感谢。

感谢我的父母对我默默的支持、多年来的抚养和教育，是你们让我感受到了家庭港湾的温暖。

感谢我的导师何顶新副教授的大力支持和帮助。在两年的研究生学习与生活中，何老师始终对我耐心指导，细心培养，使我在学术与技术上取得了不小的进步。您不仅从学术方向对我进行指导，同时也从思想上给予我点拨，告诉我做人做事的道理，竖立正确的人生观与价值观，保持一颗低调做人，高调做事的心态。

感谢郑南雁老师在硬件系统设计上的指导和帮助；感谢在研究生阶段授予我课程的赵金老师、李叶松老师、沈安文老师、王永骥老师、周纯杰老师；感谢秦肖臻和陈虹老师在我作为华中科技大学智能车团队队长期间给予的倾力帮助。

感谢瑞萨实验室基于软硬件协同处理的轮式机器人平台项目的所有成员，是你们的让这个复杂系统在半年内完成；感谢华中科技大学智能车团队的队员们在开发过程中的支持与建议、以及多年来的技术积累，是大家的齐心协力与无私奉献我们系统的站在一个比较高的起点上进行快速开发；感谢任慰博士在嵌入式系统理论、实时操作系统、计算机体系结构以及学科前沿研究上对我的指导，以及在项目实施、论文撰写过程中给予的关心与帮助；感谢胡航笛、汤国旺、叶存奎、顾强等师兄等对我的指导与帮助；感谢方华启、牛盼情、胡灿、孙佳将、胡春旭在开发与研究过程中的软硬件协同合作、团结拼搏；感谢同门汪卓作为朋友的指点与建议，是你让我看到了人生的另一番天地。

最后，感谢所有关心、支持、帮助过我的朋友们，好人一生平安。

参考文献

[1] A. Sangiovanni-Vincentelli, G. Martin, Platform-based design and software design methodology for embedded systems, Design Test of Computers, IEEE 18 (6) (2001) :23-33.

[2] A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, M. Sgroi, Benefits and challenges for platform-based design, in: Proceedings of the 41st annual Design Automation Conference, DAC'04, ACM, New York, NY, USA, 2004:409-414.

[3] A. Sangiovanni-Vincentelli, Defining platform-based design, EEDesign of EETimes.

[4] D. Andrews, D. Niehaus, R. Jidin, M. Finley, W. Peck, M. Frisbie, J. Ortiz, E. Komp, P. Ashenden, Programming models for hybrid fpga-cpu computational components: a missing link, Micro, IEEE 24 (4) (2004): 42-53.

[5] W. Wolf, A decade of hardware/software codesign, Computer 36 (4) (2003) :38-43.

[6] Vahid, F., "The Softening of Hardware", IEEE Computer, April 2003:27–34.

[7] 詹瑾瑜.SOC软/硬件协同设计方法研究[博士学位论文].成都:电子科技大学,2006.

[8] 郭鹏飞.SOC设计中的软硬件协同设计[J].今日电子,2004,(6):73-75.

[9] 熊光明,龚建伟,徐正飞等.轮式移动机器人滑动转向研究综述[J].机床与液压,2003,6:9-12.

[10] 王鸿鹏.复杂环境下轮式自主移动机器人定位与运动控制研究[博士学位论文].天津:南开大学,2009.

[11] Chatila R. Deliberation and reactivity in autonomous mobile robots. Robot and Autonomous System,1995,16:197-211.

[12] 王志文,郭戈. 移动机器人导航技术现状与展望. 机器人,2003,25(5): 470-474.

[13] 王卫华. 移动机器人定位技术研究[博士学位论文]. 武汉：华中科技大学,2005.

[14] Borenstein J,Everett R,Feng L．Mobile robot positioning - sensors and techniques．Journal of Robotic System,1997,14(4):231-249．

[15] 郑凯. 小型移动机器人系统平台的模块化设计与实现[硕士学位论文].武汉：华中科技大学,2011

[16] M. Edwards, J. Forrest, A. Whelan, Acceleration of software algorithms using hardware/software co-design techniques, Journal of Systems architecture 42 (9) (1997): 697–707.

[17] U. Ali, M. Malik, Hardware/software co-design of a real-time kernel based tracking system, Journal of Systems Architecture 56 (8) (2010): 317-326.

[18] K. M. Goh,W. P. Moh, P. Kusolpalin, A. J. R. Aendenroomer, K. V. Ling, D. Ma, Hybrid-processors real-time embedded linux platform, in: Proc. TENCON 2009 - 2009 IEEE Region 10 Conf, 2009: 1-5.

[19] G. Nguyen Thi Huong, Y. Na, S. Kim, Applying frame layout to hardware design in fpga for seamless support of cross calls in cpu-fpga coupling architecture, Microprocessors and Microsystems 35 (5) (2011): 462-472.

[20] I. del Campo, J. Echanobe, G. Bosque, J. Tarela, Effcient hardware/software implementation of an adaptive neuro-fuzzy system, Fuzzy Systems, IEEE Transactions on 16 (3) (2008): 761-778.

[21] 高文韬．基于SOPC技术的嵌入式掌纹识别系统设计与实现[硕士学位论文]．北京:北京交通大学,2006．

[22] Pomerleau, D. and Jochem, T., "Image Processor Drives Across America," Photonics Spectra, 1996: 80-85.

[23] 乔维高,徐学进.无人驾驶汽车的发展现状及方向[J].技术导向,2007,07:40-43.

[24] 无 人 驾 驶 车 从 长 沙 开 到 武 汉 ． [Online]. Available: http://news.sciencenet.cn/htmlnews/2011/7/250048-1.shtm

[25] iOS. [Online]. Available: http://www.apple.com/ios/

[26] Rover App-Controlled Spy Tank . [Online]. Available: http://www.brookstone.com/rover-remote-control-spy-tank-for-ipad

[27] Sebastian Thrun, How Google's Self-Driving Car Works, IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep. 26-29, 2011, San Francisco.

[28] Massimilano Chiodo , Paolo Giusto , Attila Jurecska , Harry C. Hsieh , Alberto Sangiovanni-Vincentelli , Luciano Lavagno, Hardware-Software Codesign of Embedded Systems, IEEE Micro, August 1994,14(4):26-36

[29] 王少平,王京谦,钱玮.嵌入式系统的软硬件协同设计[J].现代电子技术,2005,28(2):83-84.

[30] Eric Monmasson and Marcian N. Cirstea, 2007. " FPGA Design Methodology for Industrial Control Systems-A Review. IEEE transactions on industrial electronics, 54(4): 1824-1842

[31] G. Steiner, D. Isaacs, D. Pellerin, Control plane/data plane video processing with an FPGA, Xilinx Xcell 71 (2010): 24-28.

[32] 鲁欣,赵亦工,徐秀红等.数字电路设计方案的比较与选择[J].电子技术应用,2002,28(1):6-8.

[33] T. Hill, The benefits of FPGA coprocessing, Xcell Journal, 58 (2006):29-31

[34] H. Takada, Introduction to the TOPPERS Project - open source RTOS for embedded systems, in: Proc. Sixth IEEE Int Object-Oriented Real-Time Distributed Computing Symp, 2003:44-45.

[35] NXP Semiconductors, LPC24XX User manual, 2010/08/26

[36] NXP Semiconductors, LPC2478 Datasheet, 2010/9/29

[37] 瞿坦．计算机网络及应用．北京，化学工业出版社,2002:74~93

[38] 高雪娟,卓力,王素玉等.基于H.264标准的无线局域网视频传输系统实现方案[J].测控

技术,2008,27(5):15-18.

[39] Freescale Semiconductor. K60 Sub-Family Data Sheet, 2011/6/9

[40] TOPWAY Technology Co., Ltd. LM6800 LCD Module User Manual, 2004/03/17

[41] 基于远程WIFI控制的四轮驱动机器人. [Online]. Available: http://forum.eet-cn.com/BLOG_ARTICLE_6172.HTM

[42] Xilinx, Spartan-6 Family Overview, 2011/03/21

[43] Kodak, DEVICE PERFORMANCE SPECIFICATION, 2011/03/21

[44] Mike Stein.跨越鸿沟:同步世界中的异步信号[J].电子设计技术,2004,11(7):76-86.

[45] 魏堃. 跨时钟域信号同步技术研究[硕士学位论文]. 西安：西安电子科技大学,2009.

[46] Kannan Srinivasagam, David Mahashin. 针对高速接口的源同步时钟实现方案的研究[期刊论文].电子设计应用, 2005(04):93-94

[47] 赵瑜. 基于计算机视觉的移动机器人路径识别与跟踪[硕士学位论文]. 西安：西北大学,2008.

[48] RAFAEL C.GONZALEZ、RICHARD E.WOODS 著,阮秋琦、阮宇智等译.数字图像处理(第二版)[M].北京:电子工业出版社,2007.

[49] 赵瑜,种兰祥,张万绪. 基于计算机视觉的移动机器人导航[J].现代电子技术,2008(08) :165-173.

[50] 云创工作室. 多媒体处理FPGA实现——System Generator篇[M].北京:电子工业出版社,2010.

[51] Xilinx, LogiCORE IP Divider Generator v3.0 Datasheet, 2011/05

[52] 黄艳军. 基于FPGA的数字图像预处理算法研究[硕士学位论文].南京：南京理工大学,2009

[53] 陈昊,翁显耀. 基于FPGA的智能车路径图像识别的预处理设计[J].自动化技术与应用,2010(01) :107-110

[54] 陈伯时. 电力拖动自动控制系统——运动控制系统,第3版. 北京,机械工业出版

社,2003:14-15

[55] 孙德宝,王永骥,王金城．自动控制原理．北京,化学工业出版社,2002:58~93

[56] 谢剑英,贾青．微型计算机控制技术,第3版．国防工业出版社,2006

[57] 肖金栋. ABB Industrial IT系统在水泥生产中的应用[J].水泥,2005(6):49-50 .

[58] 李俊源 秦忆 周永鹏．电力拖动基础,修订版．武汉,华中科技大学出版社,1999:16~17

[59] Intersil, X9313 Digitally Controlled Potentiometer data Sheet ,2008：1~12

附录 1 攻读学位期间发表论文目录

[1] Wei Ren, Bo Zhou, Jin Zhao. Design and Implementation of an ARM-FPGA Hybrid Embedded System Based On TOPPERS Software Components. Journal of Systems Architecture.(Under Review)