

电子信息与电气学科规划教材·电子信息科学与工程类专业

复杂电子系统设计与实践

刘克刚 陈 曦 王卫兵 张 兰 编著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书以复杂电子系统设计为目标,其内容围绕电子系统的设计与实现方法来安排。全书共 19 章,第 1 章至第 8 章详细介绍微机应用系统的设计与实践,第 9 章至第 15 章主要阐述 EDA 的典型应用——FPGA/CPLD 电路设计与实践,第 16 章至第 18 章重点分析若干复杂电子应用系统的设计思想和设计方法,第 19 章简要讨论电子系统设计中所涉及的工程实现方面的有关问题。为方便教学,本书配有免费电子教学课件。

本书取材广泛,内容上既有深度又有广度,叙述由浅入深,理论、分析与设计相结合,前后连贯,系统性较强。为了体现本书的实践性,书中对每一种典型电子系统都提供了设计方案和设计方法,同时在进行各种电子系统设计时尽可能采用能反映近代电子技术发展的新器件、新技术,注重内容的新颖性和实用性。

本书可作为高等院校电子科学与技术及信息与通信类专业高年级本科生和研究生的教材及参考书,也可作为全国大学生电子设计竞赛赛前训练和大学生从事电子技术方面的课外科技创新等实践环节的教材,还可作为工程设计人员的参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

复杂电子系统设计与实践/刘克刚等编著. —北京:电子工业出版社, 2010.6

电子信息与电气学科规划教材. 电子信息科学与工程类专业

ISBN 978-7-121-11032-0

I. ①复… II. ①刘… III. ①电子系统—系统设计—高等学校—教材 IV. ①TN02

中国版本图书馆 CIP 数据核字(2010)第 104132 号

策划编辑: 段丹辉

责任编辑: 段丹辉 特约编辑: 王纲

印 刷:

装 订:

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1092 1/16 印张: 23.5 字数: 695 千字

印 次: 2010 年 6 月第 1 次印刷

印 数: 4000 册 定价: 42.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话: (010)88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010)88258888。

前 言

综合性、设计性实践训练已成为高等工程教育体系中不可或缺的课程之一,它与传统模式下的实验教学最大的不同之处在于:传统模式下的实验教学一般给出实验原理(或电原理)、实验器材、实验步骤及方法,以验证性实验为主,缺乏对学生综合素质能力的训练和培养。近年来由于电子技术的发展,电子产品设计理念、方法及环境都有了重大的变化。尤其是设计环境的变化、微电子技术的迅猛发展及 EDA 技术的引入,使现代电子产品的复杂性大大上升。当代电子系统的设计方法与环境所发生的巨大变化,是对高等院校理工科电子科学与技术及信息与通信类专业大学生的一个强有力的挑战。但目前的现状是,电子科学与技术及信息与通信类专业大学生在电子系统设计及应用上获得的训练远远少于传统模式下的验证性实验的训练和指导,满足不了现代社会在工业应用及科研实践活动中对复合型应用人才的需求。基于对这种状况的反思,武汉大学电子信息学院从 1996 年开始设置了“电子系统设计与实践”的实验课程,并编写了相应的讲义。经过多年的教学实践取得了较好的效果,2001 年获湖北省高等学校省级教学成果三等奖,2005 年获湖北省高等学校省级教学成果二等奖。2007 年武汉大学国家电工电子实验教学示范中心的建立为实验教学的改革注入了新的动力,为了进一步深化实验教学改革,满足高校电子科学与技术及信息与通信类专业培养具有综合电子系统设计能力与创新素质人才的需要,在总结经验与吸收各方面意见的基础上,我们组织编写了《复杂电子系统设计与实践》这本书。

本书是依据高等院校电子技术实践教学大纲的基本要求,并结合作者多年教学与科研的经验而编写的。本书以普通高等院校电子科学与技术及信息与通信类本科三、四年级学生为主要对象,以具有较大难度的基于微处理器和大规模可编程器件的测控和通信系统及具有严格技术指标的集成化功能产品为设计目标,旨在通过提供系统的综合性、设计性的电子系统设计的训练,来提高学生的综合素质能力。本书在正式出版前,曾作为校内选修课和电子设计竞赛赛前培训的内部教材,使用过程中得到了学生的肯定和好评。

本书共 19 章,主要内容大体分为三个部分:第一部分(第 1 章~第 8 章)为微机应用系统的设计与实践,主要介绍基于 MCS-51/52 单片机应用系统的设计,内容涉及数字信号源、数字频率计、数据采集与回放、数据传输、控制策略与算法研究及简单测试仪器的设计与实现等;第二部分(第 9 章~第 15 章)为 EDA 应用——FPGA/CPLD 电路设计与实践,在简单介绍 FPGA/CPLD 的设计流程的基础上,通过若干设计实例讨论了基于 FPGA 的外设控制电路、协议转换电路、数字信号处理电路及其他复杂系统的设计;第三部分(第 16 章~第 19 章)为复杂电子系统设计,内容涉及基于 NiosII 处理器、DSP 处理器和 ARM 处理器的应用电路系统的设计思想、设计方法及其他细节。对于电子系统设计中所涉及的工程实现方面的有关问题,如抗干扰及电磁兼容、热特性、可测性、可靠性及信号完整性等,也进行了讨论。

本书具有如下主要特点:

(1) 以综合性、设计性的实验作为训练手段,以素质教育为目标,具体内容围绕基于微型计算机平台的复杂电子系统的设计与应用而展开。本书强调了对构成电子系统的三种基本子系统——模拟系统、数字系统及微机子系统的设计问题及电子系统工程实现中的难点的训练。对 DSP(数字信号处理)子系统的设计与应用也有专门章节讨论。

(2) 以新颖、实用的综合性、设计性课题为牵引点,带动学生的学习积极性,强调学生的动手能

力的培养和提高。采用引导、开放式的教学形式,由浅入深,逐步引导,注重学生理性逻辑思维能力及综合能力的提高。

(3) 本书的着眼点是让学生从理论学习的轨道逐步走向实际应用与工程设计实现,把理论上的定性分析与定量计算、实验调整等手段结合起来,掌握电子系统工程设计的步骤和方法,了解和熟悉科学实验的程序和实施方法,为以后的工程技术及科研实践工作奠定基础。

(4) 采用与计算机技术、微电子技术及电路系统理论平行发展起来的 EDA 技术完成现代电子系统的设计是本书内容的另一重点。本书强调了在选择和使用 EDA 工具及从设计方法学的观点出发掌握 EDA 工具设计系统流程的重要性。

(5) 注重拓宽学科基础、扩展知识面,使强电与弱电结合、硬件和软件结合,注重新技术、新知识、新方法、新器件的学习和应用。

(6) 选题均为综合性、设计性课题,涵盖内容广泛,包括了低频、高频、模拟电路、数字电路、通信原理、微机原理、单片机技术、可编程器件、数字信号处理器件、嵌入式系统 ARM 等课程的知识,培养学生对相关理论知识的综合应用能力。

(7) 注意选题与设计的合理性及难度的综合分析与把握。选题的多样化及不同难度的搭配,方便教师在组织教学过程中采取分层次的教学方法,使不同水平的同学都能得到符合自身能力特点的训练。

(8) 强调和突出自我能力的培养,给出了设计范例及方案论证,但要求学生独立进行系统设计。由于题目及内容的多样化、实用性,有助于培养学生独立获取资料及信息的能力,以获得好的设计思想。

本书由刘克刚教授负责统稿与定稿。第 1~8, 19 章由刘克刚、张兰编写,第 9~11, 14, 15, 18 章由陈曦编写,第 12, 13, 16, 17 章由王卫兵编写。孙世磊、徐江丰、徐心毅、王春林也参与了本书部分章节的编写。

为了方便教师教学,本书配有免费电子教学课件,可以登录华信教育资源网(<http://www.hxedu.com.cn>)注册下载或发送电子邮件至 duandh@phei.com.cn 索取。

本书的编写既是对多年教学成果的总结和升华,也是一种尝试,虽然付了很大的努力,但由于编者水平有限,书中难免有错误和不妥之处,恳请读者批评指正。如果读者在阅读本书的过程中发现了错误或有改进的建议,请通过电子邮件 LKG@eis.whu.edu.cn 与编者联系。

编 者
于武汉大学

目 录

第 1 章 电子系统设计导论	(1)	3.2.1 频率或脉冲速率的数字测量方法	(39)
1.1 电子系统概述	(1)	3.2.2 时间参数的数字测量方法	(39)
1.1.1 相关概念	(1)	3.2.3 电子计数器的测量误差	(39)
1.1.2 电子系统的构成	(2)	3.3 数字频率计系统设计	(44)
1.2 电子系统的设计	(2)	3.3.1 设计任务的分析及方案论证	(45)
1.2.1 电子系统设计的一般方法	(2)	3.3.2 等精度测量的技术实现难点分析	(47)
1.2.2 电子系统设计的一般步骤	(3)	3.3.3 周期脉冲信号占空比测量原理	(50)
1.2.3 传统手工设计步骤	(5)	3.3.4 系统具体的设计和实现	(50)
1.2.4 电子系统设计的 EDA 方法	(5)	3.3.5 基于 CPLD 的数字频率计的设计	(53)
1.3 各种电子系统设计步骤综述	(6)	3.4 相位测量	(55)
1.3.1 数字系统的设计步骤	(7)	3.4.1 相位测量方案分析与论证	(55)
1.3.2 模拟系统的设计步骤	(7)	3.4.2 系统设计与实现	(57)
1.3.3 以微机(单片机)为核心的电子 系统的设计步骤	(7)	3.5 本章小结	(58)
第 2 章 简单系统的设计与实践	(8)	第 4 章 数字信号源的设计与实现	(59)
2.1 引言	(8)	4.1 引言	(59)
2.2 通用 MCS-51/52 单片机最小系统	(9)	4.2 频率合成技术及常用方法介绍	(59)
2.2.1 通用 MCS-51/52 单片机最小系统的 主要组成部分介绍	(10)	4.2.1 直接频率合成(DFS)	(60)
2.2.2 通用 MCS-51/52 单片机最小系统的 应用实验举例	(14)	4.2.2 采用锁相环(PLL)电路的频率 合成	(60)
2.3 基于单片机系统的简易电子琴的设计 与实现	(19)	4.2.3 直接数字频率合成(DDS)	(61)
2.3.1 设计任务的分析及设计模型的 建立	(19)	4.3 基于 PLL 的数控信源的设计	(63)
2.3.2 系统具体的设计和实现	(22)	4.3.1 系统设计方案分析	(63)
2.3.3 提高部分难点提示	(25)	4.3.2 系统模块分析和设计	(64)
2.4 LED 显示屏系统的设计与实现	(26)	4.3.3 系统软件设计	(68)
2.4.1 设计任务的分析	(27)	4.3.4 系统调试	(68)
2.4.2 以单片机为控制核心的 LED 显示 屏的设计	(27)	4.4 基于 DDS 的数控信源的设计	(69)
2.4.3 系统的设计思路与实现方式	(30)	4.4.1 DDS 的相关分析	(69)
2.4.4 基于 PC 平台的 LED 显示屏系统 分析	(34)	4.4.2 系统设计分析与理论计算	(71)
2.5 本章小结	(36)	4.4.3 系统整体设计	(74)
第 3 章 时间(频率)的数字化测量	(37)	4.4.4 系统硬件设计	(74)
3.1 引言	(37)	4.4.5 系统软件设计	(76)
3.2 频率测量原理及误差分析	(37)	4.4.6 噪声分析和降噪措施	(77)
		4.5 任意波形发生器的设计与实现	(80)
		4.5.1 波形发生器系统设计方案	(80)
		4.5.2 系统设计关键点分析	(81)
		4.5.3 系统软件设计	(84)
		4.6 数字移相器的设计	(85)
		4.6.1 系统设计原理分析	(86)

4.6.2	系统设计方案和难点分析	(88)	7.2.5	变结构控制	(135)
4.6.3	系统软件设计	(90)	7.2.6	智能控制	(135)
4.7	本章小结	(91)	7.3	恒温控制系统的设计与实现	(136)
第 5 章	数据采集与回放系统的设计与实现	(92)	7.3.1	设计任务分析	(136)
5.1	引言	(92)	7.3.2	系统硬件电路方案设计及分析	(138)
5.2	数据采集与回放系统设计方法概述	(92)	7.3.3	控制策略及算法实现与比较	(146)
5.2.1	采集系统分类	(92)	7.3.4	系统设计	(152)
5.2.2	数据采集过程	(93)	7.3.5	系统调整与性能测试	(154)
5.2.3	A/D 转换器与 D/A 转换器的选取	(93)	7.4	本章小结	(155)
5.3	高精度数据采集与回放系统的设计	(99)	第 8 章	简单测试仪器的设计与实现	(156)
5.3.1	系统分析与设计	(100)	8.1	引言	(156)
5.3.2	系统各模块的设计	(100)	8.2	数字电容测量仪	(157)
5.3.3	工作模式分析与设计	(105)	8.2.1	测量原理分析与论证	(157)
5.3.4	数据采集系统的误差调整	(106)	8.2.2	系统参数的计算	(160)
5.3.5	系统软件设计	(109)	8.2.3	系统硬件电路设计	(161)
5.3.6	系统抗干扰措施	(109)	8.2.4	软件设计	(162)
5.4	语音存储与回放系统的设计	(110)	8.3	数字工频多用表	(162)
5.4.1	系统分析与参数计算	(110)	8.3.1	系统设计方案分析和理论计算	(163)
5.4.2	数据编码与存储	(111)	8.3.2	各模块电路设计与分析	(165)
5.4.3	系统整体设计框图	(114)	8.3.3	系统软件设计	(168)
5.4.4	系统各模块电路的设计方案	(114)	8.4	简易数字存储示波器	(169)
5.4.5	三种模式软件设计流程图	(117)	8.4.1	简易数字存储示波器的系统设计 方案	(170)
5.4.6	噪声分析与降噪措施	(117)	8.4.2	主要技术指标与设计参数计算	(170)
5.5	本章小结	(118)	8.4.3	系统各模块电路设计	(173)
第 6 章	数据传输系统的设计	(119)	8.4.4	系统软件设计	(177)
6.1	引言	(119)	8.5	简易逻辑分析仪设计	(178)
6.2	数据采集与传输系统的设计	(119)	8.5.1	任务分析与方案论证	(178)
6.2.1	系统设计方案分析	(120)	8.5.2	理论分析和参数计算	(179)
6.2.2	调制方案的确定与相应数学模型 的建立	(120)	8.5.3	系统各个模块的设计与实现	(180)
6.2.3	噪声模拟发生器的设计和模型 分析	(124)	8.5.4	系统软件设计	(183)
6.2.4	系统各模块设计分析	(126)	8.6	本章小结	(185)
6.2.5	系统软件设计	(129)	第 9 章	基于 FPGA/CPLD 的电路设计流程 简介	(186)
6.3	本章小结	(130)	9.1	引言	(186)
第 7 章	控制策略与算法的研究	(131)	9.2	FPGA 设计软件	(186)
7.1	引言	(131)	9.2.1	HDL 设计输入	(188)
7.2	基于非线性、纯时滞被控对象的控制 策略及算法的分析	(132)	9.2.2	原理图设计输入	(194)
7.2.1	Smith 预估控制和大林算法	(132)	9.3	FPGA 器件的使用	(199)
7.2.2	自适应控制	(134)	9.3.1	FPGA 与 CPLD 器件的比较	(199)
7.2.3	预测控制	(134)	9.3.2	FPGA 器件的使用	(200)
7.2.4	鲁棒控制	(134)	9.3.3	FPGA 实验开发平台的介绍	(201)
			9.4	本章小结	(202)

第 10 章	基于 FPGA 的外设控制电路的设计	(203)			关讨论	(244)
10.1	引言	(203)	12.3	基于 FPGA 的 FFT 算法的实现	(245)	
10.2	LED 数码管的控制与显示	(203)	12.3.1	FFT 基础知识	(245)	
10.2.1	LED 数码管的显示原理	(203)	12.3.2	基-r Cooley-Tukey FFT 算法	(246)	
10.2.2	FPGA 实现 LED 显示控制	(204)	12.3.3	基-2 Cooley-Tukey FFT 算法的 FPGA 实现	(247)	
10.2.3	模块功能实现	(204)	12.3.4	离散傅里叶逆变换 (IDFT) 的快速 计算方法	(249)	
10.3	A/D 转换电路的控制与实现	(206)	12.3.5	改进的 DFT 实现方法	(249)	
10.3.1	ADC0809 模数转换芯片的工作 时序	(206)	12.4	Goertzel 算法及其在 FPGA 上的 实现	(250)	
10.3.2	模块功能实现	(207)	12.4.1	基本 Goertzel 算法	(250)	
10.3.3	系统整体设计	(209)	12.4.2	Goertzel 优化算法	(250)	
10.4	D/A 转换电路的控制与实现	(210)	12.4.3	Goertzel 算法在 FPGA 上的 实现	(253)	
10.4.1	D/A 芯片的时序和控制方式分析	(210)	12.5	本章小结	(253)	
10.4.2	模块功能实现	(211)	第 13 章	基于 FPGA 的其他复杂电路设计与 实现	(254)	
10.4.3	系统整体设计	(214)	13.1	引言	(254)	
10.5	LED 点阵的控制与显示	(215)	13.2	基于 FPGA 的电子密码锁的设计与 实现	(254)	
10.5.1	LED 点阵的显示原理	(215)	13.2.1	电子密码锁原理	(255)	
10.5.2	模块功能实现	(216)	13.2.2	系统设计描述	(256)	
10.6	步进电机的转速 / 方向控制	(217)	13.2.3	密码锁各模块的设计	(257)	
10.6.1	步进电机的工作原理	(217)	13.3	基于全数字锁相环的频率合成器的 设计	(260)	
10.6.2	模块功能实现	(218)	13.3.1	全数字锁相环的性能分析	(260)	
10.7	本章小结	(222)	13.3.2	系统整体设计	(261)	
第 11 章	基于 FPGA 的协议转换电路设计	(223)	13.3.3	全数字锁相环设计	(262)	
11.1	引言	(223)	13.3.4	自适应频合器的设计	(270)	
11.2	简易 UART 接收模块的设计与实现	(223)	13.3.5	频合器系统仿真分析测试	(270)	
11.2.1	UART 的基本通信原理	(224)	13.4	本章小结	(272)	
11.2.2	系统总体分析	(225)	第 14 章	基于 FPGA 的 JPEG 图像压缩	(273)	
11.2.3	关键模块设计	(226)	14.1	引言	(273)	
11.3	基于 USB2.0 数据收发模块的设计 与实现	(228)	14.2	JPEG 图像压缩的原理与实现	(273)	
11.3.1	USB2.0 简介及传输类型	(228)	14.2.1	JPEG 图像压缩算法分析	(274)	
11.3.2	USB 控制器 CY7C68013	(229)	14.2.2	图像分割	(275)	
11.3.3	基于 USB2.0 从设备数据收发 模块的实现	(231)	14.2.3	离散余弦变换	(275)	
11.4	本章小结	(236)	14.2.4	量化与游程编码	(277)	
第 12 章	基于 FPGA 的 FFT 算法实现	(237)	14.2.5	熵编码	(279)	
12.1	引言	(237)	14.3	本章小结	(286)	
12.2	设计任务的提出与傅里叶变换的理论 分析	(237)	第 15 章	基于 FPGA 的神经网络对数-S 形 函数的设计与实现	(287)	
12.2.1	离散傅里叶变换	(238)				
12.2.2	傅里叶变换的相关讨论	(240)				
12.2.3	基于 FPGA 的 FFT 算法设计的相					

15.1	引言	(287)	17.4.1	$\mu\text{C}/\text{OS}-\text{II}$ 的基本组成	(328)
15.2	设计任务的提出与神经网络的基础知识	(287)	17.4.2	$\mu\text{C}/\text{OS}-\text{II}$ 的移植	(333)
15.2.1	人工神经网络	(287)	17.5	本章小结	(336)
15.2.2	神经元模型	(288)	第 18 章	数码相机伴侣系统的设计与实现	(337)
15.2.3	神经网络结构	(290)	18.1	引言	(337)
15.3	坐标旋转数字计算机(CORDIC)		18.2	数码相机伴侣系统的设计	(337)
	算法	(290)	18.2.1	系统分析	(338)
15.3.1	CORDIC 算法	(290)	18.2.2	硬件系统设计	(338)
15.3.2	混合 CORDIC 算法	(293)	18.2.3	软件系统设计	(339)
15.4	基于 FPGA 的对数-S 形函数模块的硬件设计与实现	(294)	18.3	本章小结	(350)
15.5	本章小结	(297)	第 19 章	电子系统工程实现中的问题	(351)
第 16 章	基于 TMS320C55XX 系列 DSP 的系统硬件和软件设计	(298)	19.1	概述	(351)
16.1	引言	(298)	19.2	电子系统的抗干扰设计	(351)
16.2	TMS320C55XX 系列 DSP 简介	(298)	19.2.1	电磁干扰与电磁兼容问题	(351)
16.2.1	DSP 芯片的特点	(298)	19.2.2	干扰的类型	(352)
16.2.2	TI 公司 DSP	(299)	19.2.3	干扰传播的途径	(353)
16.2.3	DSP 芯片选型	(300)	19.2.4	抗干扰设计方法	(353)
16.3	TI DSP 开发集成环境 CCS 简介	(301)	19.3	电子设备热设计	(355)
16.3.1	CCS 的简介	(301)	19.3.1	功率器件的散热	(355)
16.3.2	CCS 的安装与使用	(302)	19.3.2	整机的散热	(356)
16.4	基于 DSP 的水声通信终端设计	(305)	19.4	可靠性设计	(356)
16.4.1	水声通信系统介绍	(305)	19.5	数字电路的可测试性设计	(357)
16.4.2	前端信号处理模块	(305)	19.6	印制电路板(PCB)的设计与装配	(359)
16.4.3	A/D 数据采集接口	(307)	19.6.1	PCB 的设计	(359)
16.4.4	DSP 信号处理模块	(308)	19.6.2	PCB 的装配与焊接	(361)
16.5	卷积码编解码实现	(310)	19.7	电子系统的调试	(361)
16.5.1	卷积码编码	(310)	19.7.1	通电调试之前的检查	(361)
16.5.2	卷积码解码	(311)	19.7.2	调试的一般顺序与步骤	(362)
16.5.3	交织器及维特比译码算法的部分程序代码	(312)	19.7.3	做好调试记录	(362)
16.6	本章小结	(316)	19.7.4	模拟电路的调试	(362)
第 17 章	嵌入式操作系统	(317)	19.7.5	数字电路系统的调试	(363)
17.1	引言	(317)	19.7.6	带微处理器系统的软件调试	(364)
17.2	嵌入式实时操作系统的基本概念	(318)	19.8	本章小结	(364)
17.2.1	嵌入式实时操作系统的特点	(318)	参考文献		(365)
17.2.2	嵌入式实时操作系统的相关概念	(318)			
17.3	常见嵌入式实时操作系统介绍	(322)			
17.4	嵌入式实时操作系统 $\mu\text{C}/\text{OS}-\text{II}$ 及其移植	(328)			

第 1 章 电子系统设计导论

1.1 电子系统概述

1.1.1 相关概念

1. 电子系统的概念

系统是由两个以上各不相同且相互联系、相互制约的单元组成的、在给环境下能够完成一定功能的综合体。系统的基本特征是：在功能与结构上具有综合性、层次性和复杂性。这些特征决定了系统的设计与分析方法将不同于简单的对象。电子系统通常是指由电子元器件或部件组成的、能够产生、传输或处理电信号及信息的客观实体，典型的电子系统包括通信系统、计算机系统、电子测量系统、自动控制系统等。

通信系统的种类很多，现以移动电话为例来看其组成。图 1.1 是一个简化的 GSM900 蜂窝移动电话的子系统组成方框图。该系统是一个包括了发射机、接收机、微型计算机和音频及数字信号处理器(DSP)、SIM 卡等子系统的复杂系统。下面通过该移动电话系统来看下一个复杂系统在结构上的层次性。框图中的每一个子系统又可分解为由若干部件组成的系统，例如，其中的微型计算机子系统就是由微处理器、存储器、键盘及显示器等几个部件组成的。而组成子系统的每个部件又可分解为由许多元件组成的电路。类似地，发射机、接收机也可由顶层(子系统级)向下，一层一层地分解，直到元件级(底层)。一般情况下，稍微复杂一点的电子系统均具有如图1.2 所示的层次式结构。

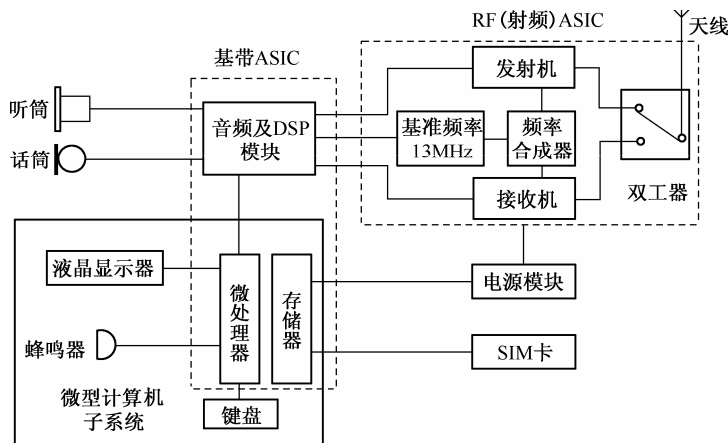


图 1.1 简化的 GSM900 蜂窝移动电话子系统组成方框图

2. 电子系统、网络与集成

组成电子系统的主要部件中包括了大量的、多种类型的电子元器件和电路。电路也称为电网络或网络。当研究一般的抽象规律时多用网络一词；反之，讨论一些指定的具体问题时则称之为电路。一般来说，系统比网络更复杂、规模更大的组合体。然而，实际中常常将一些简单的网络或电路也称为系统。同一个事物当作为系统问题研究时应注意其全局，而作为网络问题研究时则关心其局部。

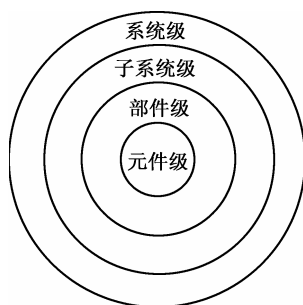


图 1.2 电子系统结构的层次性

而在电子系统中的集成大多是指集成电路或集成系统，集成电路最简单的例子是利用电路实现时序功能时，可以把不同的触发器和门电路组合在一个芯片上构成锁存器或寄存器。也可以通过将一串触发器连接在一起构成计数器。集成系统一般是指 PCB 上集成系统，它是由若干功能不同但又相互联系的子系统组成的，在给定环境下能够完成一定功能的综合体。由于技术的进步，已有越来越多的 PCB 上集成系统被片上集成系统(SoC)所替代，单片机取代单板机就是这样的例子。电子系统的构成就是由若干电路(包括集成电路和分立电路)及由不同电路组成的不同子系统的集成。

1.1.2 电子系统的构成

如前所述，电子系统是由若干不同的相互连接、相互作用的子系统构成的，尽管子系统的类型很多，但是归纳起来不外乎 5 种基本类型：模拟子系统、数字子系统、微机子系统、模拟与数字混合子系统和 DSP(数字信号处理)子系统。从设计的基本方法上来讲，掌握了模拟子系统、数字子系统和微机子系统这 3 种最基本的子系统的设计方法，就可以设计上述 5 种子系统。DSP 子系统的设计可在掌握 DSP 的理论和算法的前提下，借助数字子系统的设计方法、微机程序设计方法和硬件配置方法去完成；模拟与数字混合子系统的设计可通过将模拟子系统与数字子系统的设计方法结合起来去完成。读者一般已经接受过上述 3 种最基本子系统的设计与分析的学习和实践训练，在此基础上，通过本书的学习可以掌握更为复杂的电子系统的设计和分析方法。

1.2 电子系统的设计

1.2.1 电子系统设计的一般方法

因为电子系统的复杂性，必须用有效的方法去管理其复杂性才能使系统设计得到成功。基于系统的功能与结构上的层次性，演化出了如下 3 种设计方法。

1. 自顶向下法(Top-Down)

自顶向下法是一种概念驱动的设计方法。该方法要求在整个设计过程中尽量运用概念(即抽象)去描述和分析设计对象，而不要过早地考虑实现该设计的具体电路、元器件和工艺，以便抓住主要矛盾，避免纠缠在具体细节上，这样才能控制好设计的复杂性。整个设计从顶层到底层，应当逐步由概括到具体，由粗略到精细。只有当整个设计在概念上得到验证与优化后，才能考虑“采用什么电路、元器件和工艺去实现该设计”这类具体问题。该设计方法首先从系统级设计开始。系统级的设计任务是：根据原始设计指标或用户的需求，将系统的功能(或行为)全面、准确地描述出来，即将系统的输入-输出关系描述出来，然后进行子系统级设计。进行一项大型、复杂系统设计的过程，实际上是在自顶向下的过程中包括了由底层返回到上层进行修改的多次反复的过程，如图 1.3 所示。

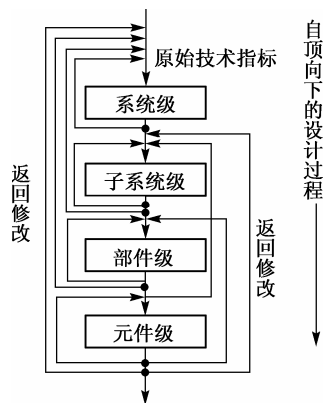


图 1.3 实际的自顶向下的设计过程

2. 自底向上法(Bottom-Up)

该方法是根据要实现的系统的各个功能的要求,从现有可用的元件中选出适用的,设计成一个一个的部件,当一个部件不能直接实现系统的某个功能时,就需要设计由多个部件组成的子系统去实现该功能,上述过程一直进行到系统所要求的全部功能都实现为止。该方法的优点是可以继承使用经过验证的、成熟的部件与子系统,从而实现设计重用,以减少设计的重复劳动和提高设计时效性。其缺点是设计过程中设计人员的思想受限于现有可用的元件,故不容易实现系统化的、清晰易懂的、可靠性高及可维护性好的设计。然而自底向上法,在系统的组装和测试过程中确是行之有效的,因此该方法常用于这些场合。此外,对于以IP核为基础的VLSI片上系统的设计,自底向上法也得到了重视和采用。

3. 以自顶向下法为主导,并结合使用自底向上法(TD & BU Combined)的方法

在近代的系统设计中,为了实现设计重用及对系统进行模块化测试,通常采用以自顶向下法为主导,并结合使用自底向上法的方法。这种方法既能保证实现系统化的、清晰易懂的、可靠性高及可维护性好的设计,又能减少设计的重复劳动,提高设计生产率。这对于以IP核为基础的VLSI片上系统的设计特别重要,因而得到了普遍采用。

上面所述的电子系统的一般设计方法,从方法学上来说与大型软件的设计方法是完全一致的。如果读者在软件设计方面已经具有一定的实践经验,在学习硬件设计的方法和原则时不妨将其与软件设计中的方法和原则做一个对照,从而加深理解。

1.2.2 电子系统设计的一般步骤

为了介绍电子系统设计的一般步骤,首先引入描述所要设计的电子系统属性的Y图。在图1.2的基础上,从圆心出发画出3个坐标轴(呈倒立的大写英文字母Y状),分别代表系统属性的行为域、结构域和物理域,如图1.4所示。可以认为该图是描述系统特性的三维坐标系。它全面地表示出了系统设计在各个层次和域上所涉及的信息及其内在联系。一个完整的电子系统设计过程,均是由顶层(系统级)从行为域出发,沿Y图以一种向心式螺旋线行迹逆时针旋转,每转一圈下降一层,直至底层(元件级)的设计全部完成为止。不论在哪一级(层)上逆时针转一圈,均要经历如下3个设计步骤:

- (1) 行为描述与设计;
- (2) 结构描述与设计;
- (3) 物理描述与设计。

下面分别对每一个设计步骤予以具体说明。

1. 行为描述与设计

行为域的设计一般按照自顶向下的设计方法,首先从系统级开始。设计人员首先要对用户需求与市场状况做深入细致的调查研究,然后对收集来的原始信息进行需求分析,最后用工程语言将所要设计的系统的各项功能和技术指标、与外部世界的接口方式和协议等描述或定义出来。例如,移动电话的双工通话功能、短信息功能、来电显示功能、存储功能、时钟/闹钟功能及与传真机/计算机接口的功能,接收/发射频率、调制方式、待机时间、连续通话时间、供电电池电压、尺寸和质量等。而子系统级、部件级和元件级上的行为则由各个层次上所用单元的功能——即输入-输出关系来描述。它们是由设计人员从系统级逐层向下进行功能划分并逐步推演和定义出来的。显然,不同的设计人员会得出不同的结果,这是一种一对多的映射关系。

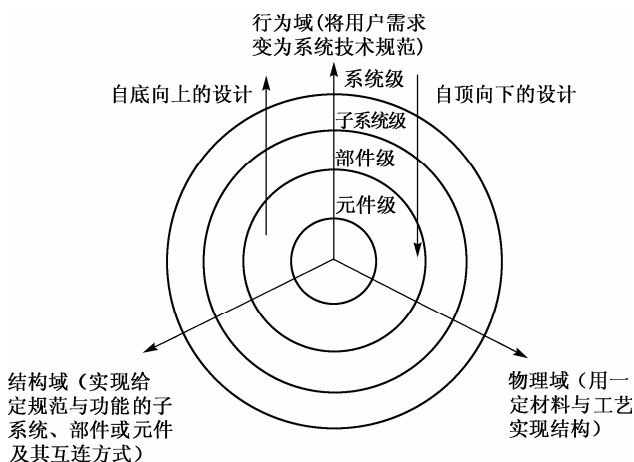


图 1.4 电子系统属性的 Y 图

2. 结构描述与设计

(1) 行为域到结构域的映射

完成了行为域的描述与设计后，下一步就要将行为映射为结构。即以行为域的设计结果作为原始输入信息，选用或设计一定的单元并按一定方式(含规则)互连起来，实现给定层次上的行为(功能)。系统从行为域到结构域的映射又称为综合。

(2) 设计表达

系统级上结构设计的任务就是确定系统与外部世界(包括使用者、其他系统或部件等)的相互作用、互连方式与协议。子系统级、部件级和元件级上的由行为域到结构域的映射是大家所熟悉的，这些级上的结构设计结果通常用方框图、电路图来表达。例如，移动电话系统级的结构设计就是确定用户操作界面和该移动电话与 GSM 网之间信息交换的方式与协议，确定与传真机、计算机之间的连接方式与协议等关系。在子系统级上，移动电话的结构可用图1.1所示的方框图来描述。一张详细的移动电话的电路图就是对该系统元件级或者元件级、部件级、子系统级混合结构的描述。当然，实际的结构设计文档除了图之外，还应有相应的文字说明。

3. 物理描述与设计

(1) 物理描述与设计的一般性过程

结构域的描述与设计完成后，最后一步就是进行从结构域到物理域的映射。即用结构域的设计结果作为原始输入信息，选用一定的材料、技术与工艺去实现给定的结构。仍以移动电话为例，其系统级上的物理设计包括机壳、主机板、操作按键、显示屏窗口、与外部互连的接插件等的外形、尺寸、材料及工艺的确定，还要决定是否采用 VLSI 专用芯片(ASIC)来实现整个系统(即片上系统)；子系统级与部件级上的物理设计的内容基本相同，包括每个子系统或部件的尺寸、安放位置、互连线的材料与布局的确定，还要决定是否采用 ASIC 来实现子系统或部件、是否需要屏蔽与散热等；元件级上的物理设计包括每个元件的型号与尺寸、主机印制底板布线的设计、是否需要屏蔽与散热等。如果系统、子系统或部件决定采用 ASIC，那么物理设计还需要进行集成电路版图的设计。根据物理设计提供的完备而正确的设计文档，就可以送交工厂制造出系统的样机。

(2) 正确性验证

必须强调指出的是，设计过程中必须适时地进行正确性验证。传统的做法是靠人工检查及搭试电路和制作样机来实现。随着系统的规模与复杂度的增加，单靠人工方法去设计与验证不但无能为

力,也是根本不可能的。必须采用下面所要介绍的 EDA(电子设计自动化)方法。设计验证所用的基本方法是分析,利用电子设计自动化(Electronics Design Automation, EDA)工具进行模拟就是以分析为基础的验证过程。一个完整的系统分析过程其步骤正好与设计过程相反,即物理域→结构域→行为域(在图1.4所示 Y 图上是按顺时针方向旋转的)。

(3) 设计的层次问题

最后还要说明的一个问题就是设计的层次问题。按照自顶向下的设计方法,当考虑到设计重用时,某个层次的设计中所用的单元可以是不限于该层次上的。这是因为引用已有的成熟单元时,是不用关心其内部结构的,只需知道其外部特性与技术指标就够了。这样一来,当设计由顶层向底层过渡时,被引用的单元就不必再细化(即不必重新设计)而保持原样。结果,在部件级上出现的单元除了部件外,还会夹有在上一级设计中被引用的子系统;同理,在元件级上会出现元件、部件和子系统相混合的情况。这种将不同层次上的单元混合使用的设计方式,称为混合层次设计方法。现代各种先进的 EDA 工具均支持这种混合层次设计方法。实际中还会遇到一些较简单的系统,不宜按层次再做任何分解,否则就难以理解该系统的功能。这种仅在一个层次上完成系统所有功能元件的详细设计的方法,称为平坦式设计方法。

1.2.3 传统手工设计步骤

传统的手工设计包括审题、方案选择与可行性论证、单元电路的设计与参数计算,以及元器件选择、组装与调试、编写设计文档与总结报告等步骤,具体过程为:

(1) 通过审题对给定任务或设计课题进行具体分析,以明确所要设计的系统的功能、性能、技术指标及要求。

(2) 把系统所要实现的功能分配给若干单元电路,并画出一个能表示各单元功能的整机原理框图,提出几种不同的方案,对它们的可行性进行论证,即从完成的功能的齐全程度、性能和技术指标的高低程度、经济性、技术的先进性及完成的进度等方面进行比较,最后选择一个较好的方案。

(3) 对各个单元电路可能的组成形式进行分析与比较,单元电路的形式一旦确定,就可以开始选择元器件,然后根据某种原则或依据先确定好单元电路中部分元件的参数,再去计算其余的元件参数和电路参数(如放大倍数、振荡频率等)。

(4) 将设计的系统在面包板或印制电路板上进行组装,并用仪器进行测试,发现问题时随时修改,直到所要求的功能和性能指标全部符合要求为止。

(5) 从设计的第一步开始就要编写文档。文档的组织应当符合系统化、层次化和结构化的要求。总结报告是在组装与调试结束之后开始撰写的,是整个设计工作的总结。

1.2.4 电子系统设计的EDA方法

随着电子技术的不断发展,电子系统设计方法得到了不断的改进和创新,基于 EDA 的方法成为电子系统设计的主流。

1. 用EDA工具设计电子系统的流程

要用 EDA 工具设计电子系统,除了需要坚实的电路与系统的理论知识外,还必须具备两个条件:一是要会选择和使用 EDA 工具;二是要清楚地知道用 EDA 工具设计电子系统的流程。虽然不同公司的 EDA 软件有不同的使用方法,但用这些 EDA 工具设计电子系统的基本流程却是一样的,具有普遍意义。如图1.5所示,这是一个采用自顶向下设计方法的流程图。为了控制设计复杂性和规范设计文档,通常采用硬件描述语言来描述系统的行为与结构,并且在部件级或子系统级以上同时伴以

方框图来描述系统结构。当系统级的模拟验证通过后，就可进行子系统级以下的设计了。这时根据子系统、部件的类型需要选择不同的设计工具，常用的工具有数字模拟软件工具、模拟电路模拟软件工具、DSP 模拟软件工具及软件设计工具，如图 1.5 所示。经过模拟验证后的各个子系统电路，在进入物理设计与实现之前，首先按其实现的物理类型将电路的组成模块做一个划分，每个模块选择一种最合适的实现方式。选择何种实现方法，是由系统设计目标决定的，涉及性能、价格等多方面的因素，属于一种多目标优化的工程问题。一个完整的电子系统设计，除了电路和软件的设计外，还要做电磁兼容性(EMC)、热学、机械等方面的设计，并由专业人员使用相应的工具去完成这些工作。

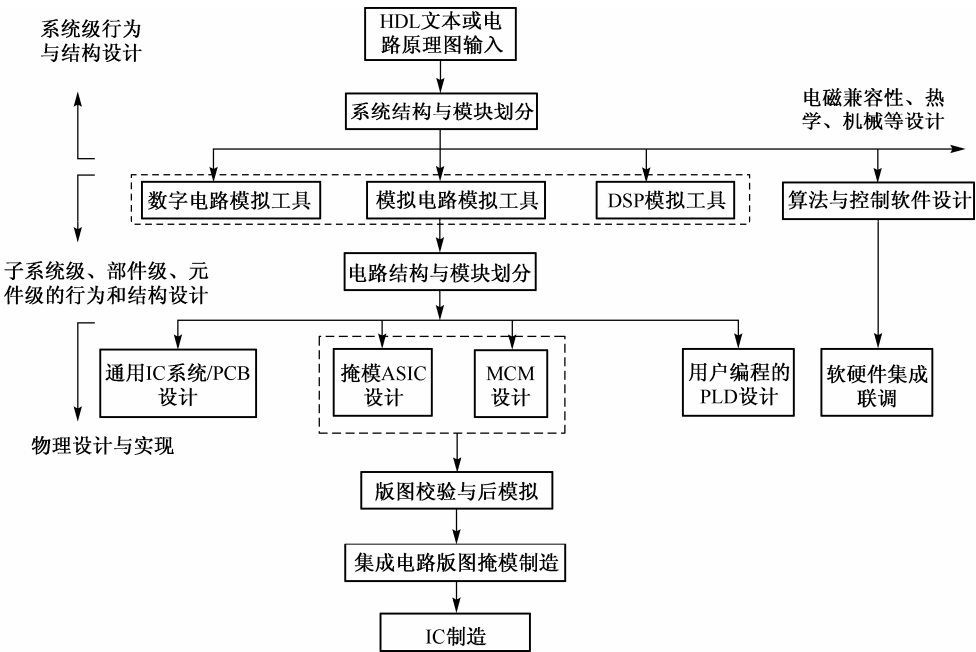


图 1.5 用 EDA 工具设计电子系统的简略流程图

2. 本课程用到的部分 EDA 工具

表 1.1 中的 EDA 软件均是 PC 版的，可安装在 PC 上，用起来很方便。读者可根据自己的设计任务与设计流程选择其中的一种或几种。关于这些软件的使用方法可查阅有关书籍、手册及软件自带的帮助文件。

表 1.1 部分 EDA 软件

EDA 工具名称	功 能
Protel	画电路图、印制电路版图
EWB (Electronics Workbench)	模拟电路、数字电路、混合电路模拟分析；电路图输入
Maxplus II, Quartus II	CPLD, FPGA 设计、仿真、下载；电路图及文本输入 (VHDL, Verilog HDL, ABEL)
MATLAB	矩阵运算、高级绘图；控制和信息处理等系统问题的分析与设计

1.3 各种电子系统设计步骤综述

前面介绍过电子系统的一般设计步骤，它与下面所列出的三类系统的设计步骤之间的关系是一般与具体、共性与个性及原则与实施的关系。由此决定了前者对后者将起着导向、规范与统筹的作

用,从而保证后者遵循正确的理念与方法。虽然下面所列的这些设计步骤,最初是面向采用通用集成电路和印制电路板去实现电子系统的的方法的,但只要将使用新器件、新工艺来实现电子系统的新方法绑定到这些设计步骤上去,它们也适用于诸如以用户可编程的 PLD 或者 ASIC 芯片等新器件来进行电子系统的设计。

1.3.1 数字系统的设计步骤

数字系统的设计步骤包括:明确设计要求(完成系统功能示意框图);确定系统方案[完成系统总体方框图——将控制器与受控器分开,拟定系统的详细算法状态机(Algorithm State Machine, ASM)流程图];设计受控器;设计控制器和实现与调试工程。

1.3.2 模拟系统的设计步骤

模拟系统的设计步骤包括:任务分析、方案比较、确定总体方案;将系统划分为若干相对独立的功能块,画出系统的总体组成框图;以实现各功能块的集成电路为中心,通过选择和计算完成各个功能单元外接电路与元件的配置;通过单元之间耦合的核算及电路的整体配合与调整,得到一个比较切合实际的系统整体电路原理图;根据结果,重新核算系统的主要指标,检查是否满足要求且留有一定裕量;画出系统元器件布置图和印制电路板的布线图,并考虑其测试方案,设置相关的测试点。

1.3.3 以微机(单片机)为核心的电子系统的设计步骤

1. 确定任务,完成总体设计

- (1) 确定系统功能指标,编写设计任务书;
- (2) 确定系统实现的硬件、软件子系统划分,分别画出硬件子系统与软件子系统的方框图。

2. 硬件、软件设计与调试

- (1) 按模块进行硬件设计,力求标准化、模块化,要有高的可靠性和抗干扰能力;
- (2) 按模块进行软件设计,力求结构化、模块化,要有高的可靠性和抗干扰能力;
- (3) 选择合适的单片机开发系统和测试仪器,进行硬、软件的调试。

3. 系统总调、性能测定

将调试好的硬、软件装配到系统样机中,进行整机总体联调。排除硬、软件故障后,进行系统的性能指标测试。

第2章 简单系统的设计与实践

2.1 引言

虽然在第1章中已经阐述了多种电子系统的设计方法和设计步骤,但初次接触电子系统设计的学习者仍然会有很多困惑。由于电子系统的设计具有很强的实践性,从理论学习逐步走向实际应用与工程设计实现,需要学习者亲身经历工程设计的每一个阶段,仔细体会其中的每一个细节,发现和解决在系统设计中所面临的每一个问题,它是一个完整的过程。然而,大多数学习者并没有这样的经历,因此自然会产生很多的困惑。诸如,什么是设计?什么是工程设计?设计很难吗?在没有电子系统设计经历的情况下,如何在较短的时间内提高自己的系统设计和管理能力,参与到复杂电子系统设计的实践之中?解开这些困惑,对于后面的学习和训练是非常必要的。

什么是设计?由于设计的范畴很大,所以不可能用很短的文字加以说明。仅就设计本身而言,设计是处理人与物、人与泛义的机器之间的问题的最重要的活动。也就是说,所谓设计,是指把一种设想、规划、问题解决的方法,通过恰当的方式(通常为视觉方式)传达出来的活动过程,它的核心内容包括以下三个方面:

- (1) 计划、构思的形成;
- (2) 视觉传达方式;
- (3) 计划通过传达之后的具体应用。

一般来说,现代设计的计划、构思是以社会需求为目标的。而传达这种计划和构思的方式,可以从简单的、传统的手工绘图、模型到复杂的计算机设计预想表现,因具体的设计要求而不同。最后的设计应用,则与具体设计所涉及的生产方式的技术条件密切相关。

多数初学者认为,设计十分玄妙,设计很难,它是少数人做的事情,其实并非如此。现代设计所涉及的面非常广泛,从复杂的宇宙飞船、飞机、汽车、手机、电视到简单的包装、服装服饰及商业广告,我们生活的空间中几乎没有一样东西是没有经过设计处理的。正因为现代社会中设计活动的广泛性,可以说设计活动充斥我们工作生活的每一个侧面,或许我们已经参与了和生活相关的设计活动,只是有自觉和不自觉的区别而已。比如,孩提时在地下画的游戏格子,自制的一份贺卡或是自己组装的一台单管收音机等,其本身就是一个设计的过程。由此看来,设计本身并不玄妙,设计就在我们身边,它是一个大家都可以参与的活动。

所谓工程设计,是指解决人造物(如机械、设备、交通工具、建筑等)中物与物之间关系的问题。显然,我们所关注的复杂电子系统设计属于工程设计的范畴。比如说,对于本章的训练中简易电子琴的设计,从工程设计的视角来看,主要解决的问题是如何产生乐音,其中微处理器和扬声器之间的关系显然就是工程设计的问题。同样,在LED显示屏的设计中,微处理器、接口电路和LED阵列之间的关系也是工程设计的问题。工程设计具有鲜明的特点,这就是设计具有的可操作性及高度的应用性,设计表达应能实现预想的功能,而功能的实现又必须满足一定的技术指标。

如前所述,设计与工程设计是有区别的,设计的范围非常繁杂,所以不可能有一个统一的设计界定范围,而只能根据具体不同的设计情况来决定它的属性。比如,平面设计、包装设计、广告设计等与工程技术的关系比较疏远,与美感的敏感表现关系比较密切,因此自然就多偏向美术设计。工业生

产及其产品因为与工程技术关系密切,因此自然应该是倾向于工程设计。对于电子信息、通信类专业本科生而言,设计教育强调的是以电子系统设计为主导的工程设计的学习和训练,其目的在于:让学生从理论学习的轨道逐步走向实际应用与工程设计实现,把理论上的定性分析、定量计算与实验调整等手段结合起来,掌握电子系统工程设计和方法和步骤,了解和熟悉科学实验的程序和实施方法,为今后的工程技术工作及科研实践工作奠定基础。

工程设计训练的重要性已毋庸置疑,但对于初学者来说从何做起,如何上手,确实是一个棘手的问题。一般来说,当今的电子产品大多都是集多种电路技术的复杂系统,更由于微电子技术的迅猛发展及 EDA 技术的引入,使现代电子产品的复杂性大大上升。这无疑增加了训练难度,这需要一个循序渐进的过程,先易后难,逐步掌握复杂电子系统的设计思想和方法。而相对于复杂电子系统而言,电路技术比较成熟且涉及的问题规模较小。即电子学基本上是一种简单的技术,它是那些基本定律、经验准则与大量电路技巧的结合。因此,对于初次接触复杂电子系统设计的学习者应从一些较为简单的电子系统入手,尝试在一些相对简单的系统上亲身参与设计实践,去完成一个完整的设计过程,从而建立自信心。基于这种考虑,在本章中安排了“简易电子琴的设计”和“LED 显示屏的设计”这两个相对较为简单的电子系统设计题目,作为复杂电子系统设计初步。

需要说明的是,现代的电子产品大多都是智能的,在系统中嵌入微处理器以提高产品的使用性能。因此,微机子系统的设计是不可缺失的,2.2 节专门讨论了基于 MCS-51/52 系统的单片机通用系统的设计。为什么选用 MCS-51/52 系统?这是因为尽管单片机不断向纵深发展,但目前乃至今后若干年,8 位单片机仍然有广泛的应用市场,MCS-51/52 系列是目前 8 位单片机的主流机型,大多数高校的微型计算机教学仍然采用 MCS-51/52 系列单片机作为主要内容,显然要方便一些。其实采用什么样的单片机并不重要,主要原因有两个:一是虽然有非常多的新型单片机出现,但仍然有相当数量的单片机是基于标准 8051 单片机或与其兼容的,即便是与 8051 不兼容的单片机,在掌握好 8051 单片机之后,也可以迅速地掌握它;二是在复杂电子系统设计中微机子系统虽然是整个系统的核心,但系统设计的焦点并不在微机子系统上,而是在微机子系统和前向、后向通道的接口电路及应用电路和算法实现上。相对于接口电路和应用电路而言,单片机最小系统具有固定的电路模型。因此,可以设想能否设计一个单片机通用系统作为一个工具平台。当面对多任务的时候,不再每次都要重新设计单片机最小系统来满足设计要求,而是直接采用一个经优化设计的单片机通用系统,以提高设计效率和资源利用率。基于这种考虑,2.2 节给出了一个经优化设计的单片机通用系统的完整电路。这一单片机通用系统于 2001 年获国家专利并经多届学生使用,其正确性已得到证实。本章的两个训练“简易电子琴的设计”和“LED 显示屏的设计”比较简单,可以不采用单片机通用系统。第 3 章及以后的训练题目由于复杂性提高,可采用单片机通用系统来完成系统设计。尽管复杂电子系统设计的重点或难点不在微机子系统上,但并不是说掌握单片机的基本原理和设计方法不重要。读者应通过单片机最小系统及接口电路的深入学习,掌握微机应用系统的工作原理和分析设计方法,为电子系统工程的实现打下必备的基础。

2.2 通用MCS-51/52 单片机最小系统

从本节开始,进入第一部分“微机应用系统设计与实践”的讨论。我们将会系统地介绍基于 MCS-51/52 单片机最小系统的应用系统设计。在此之前,读者应该对通用单片机最小系统有充分的了解。在大多数含单片机的系统中,都包含有外部 RAM、ROM、键盘、显示、A/D、D/A、I/O 扩展、中断扩展、串行通信、总线驱动、电源监控、看门狗等一些最基本的模块,我们把它们统称为“单片机最小系统”,它们是大多数控制系统所必不可少的关键部分。在实际单片机系统的应用环境中,采用一个功能齐全、性能稳定、价格低廉的通用最小系统作为核心部件,是系统正常工作的前提保证,而

且可以避免在单片机系统开发中重复设计造成的人力和物力的浪费。因此,设计并实现一个通用单片机最小系统显得非常重要。本节将以现有的通用 MCS-51/52 单片机最小系统(该设计已获得国家专利)为例,对通用单片机最小系统的主要组成部分和应用实验做详细介绍。

2.2.1 通用MCS-51/52 单片机最小系统的主要组成部分介绍

通用 MCS-51/52 单片机最小系统包括 CPU、外部存储器扩展部分、外部 I/O 口的扩展部分、键盘显示部分、模数转换和数模转换部分、总线驱动部分、硬件看门狗部分、电源监控部分、复位电路部分、系统抗干扰部分等,能满足广泛的应用要求。下面将对其中一些典型的模块进行介绍和分析。

1. CPU

所配 CPU 可以为 8031/8032/8051/8052/89C51/89C52/90C31/90C32 等多种系列,所配晶振频率为 6 MHz, 12 MHz, 24 MHz 等,均可自由选择。它是系统的控制核心。

2. 外部存储器扩展部分

在对单片机进行开发时,首先遇到的问题就是存储器的扩展,单片机内部虽然设置了一定容量的存储器,但是这种存储器的容量一般较小,往往不能满足实际需求,因此需要从外部进行扩展,配置外部存储器,包括程序存储器和数据存储器。

MCS-51 的程序存储器空间与数据存储器空间是相互独立的。当单片机不带片内 ROM 或者片内 ROM 不够用时,需扩展程序存储器。在系统 ROM 扩展时,一般使用 P0 口作为地址低 8 位(与数据口分时复用),而使用 P2 口作为地址高 8 位,它共有 16 根地址总线,寻址空间为 64 KB,用做程序存储器的器件包括 EPROM, E²PROM 等。在系统 RAM 扩展时,常用的有静态 RAM 和动态 RAM 等。给出参考的外部存储器扩展设计原理图如图 2.1 所示。

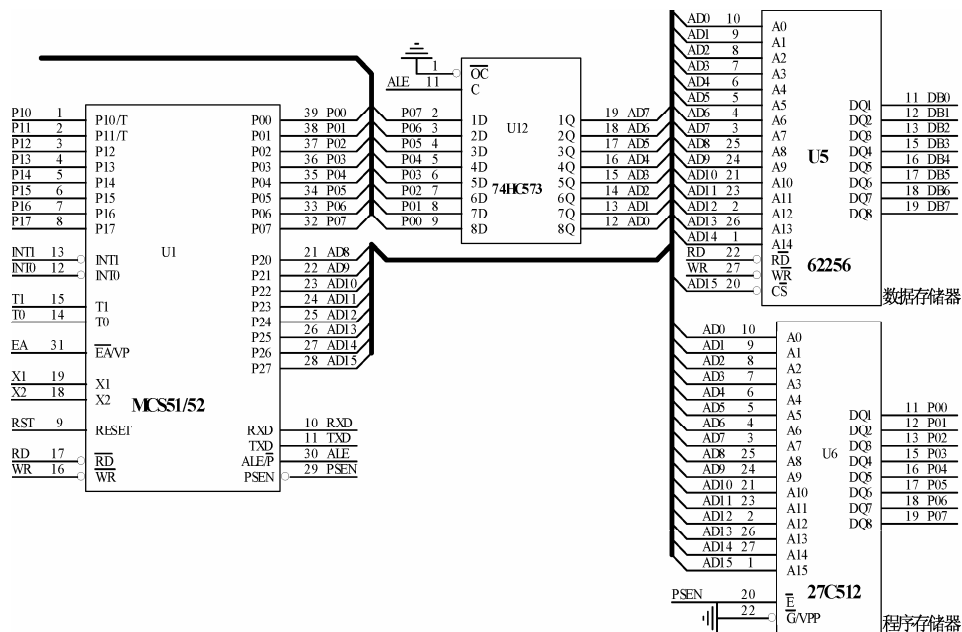


图 2.1 外部存储器扩展参考设计原理图

(1) ROM: 采用 27C512, 存储容量为 64 KB, 全地址空间为 0000 H~FFFF H。以 PSEN 信号作为

扩展程序存储器的读选通信号,在读外部 ROM 时 $\overline{\text{PSEN}}$ 是低电平有效,以实现对外部 ROM 的读操作。以 EA 信号作为内、外程序存储器的选择控制信号,当其为低电平时,对 ROM 的读操作限定在外部程序存储器;当其为高电平时,对 ROM 的读操作是从内部存储器开始的,并可延至外部程序存储器。

(2) RAM: 采用 62256, 存储容量为 32 KB, 地址范围为 0000 H~7FFF H。由 $\overline{\text{RD}}$ 和 $\overline{\text{WR}}$ 信号分别作为扩展数据存储器和 I/O 口的读选通、写选通信号。

3. 外部 I/O 口的扩展

在 MCS-51 应用系统中,单片机本身提供给用户使用的输入、输出口线并不多,因此其应用系统设计中都不可避免地要进行 I/O 口的扩展。

(1) 并行 I/O 口资源扩展

MCS-51 系列单片机有 4 个 8 位并行口,但真正可供用户使用的并行口往往只有 P1 口,因此并行 I/O 口扩展对于通用 MCS-51 单片机最小系统是必需的。常用的并口接口芯片有 8255, 8155 等。8255 具有 3 个 8 位的并行 I/O 口,具有三种工作方式,可通过程序改变其功能,因而使用灵活,通用性强,可作为单片机与多种外围设备连接时的中间接口电路。

(2) 中断系统扩展

MCS-51 为用户提供两个外部中断申请输入端: $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 。在实际的应用系统中,外部中断请求源往往比较多,因此中断系统扩展对于通用 MCS-51 单片机最小系统是必需的。

中断扩展的方式有很多,这里介绍采用中断控制器 82C59 的扩展方式,接口电路如图 2.2 所示。在解决了 82C59 和单片机工作时序不兼容的问题前提下,82C59 有 5 个中断口留给用户使用,大大方便了实时性要求很高的系统。当外部中断到来时,由 INT0 向 CPU 申请中断, CPU 再通过 82C59 得到中断向量地址值,并转入相应的中断子程序。

4. 键盘 / 显示

键盘 / 显示是通用 MCS-51/52 单片机最小系统的人机接口部分。键盘 / 显示的设计方式有多种。这里介绍采用可编程的键盘 / 显示专用控制芯片 82C79 统一管理的方式。82C79 的时钟由 CPU 的 ALE 信号经过内部分频后得到,它能完成键盘输入和显示控制两种功能,其参考设计电路如图 2.3 所示。键盘由 20 个普通键和 1 个 Shift 键组成,可以完成 40 个按键的功能,当有键按下时,由 IRQ 向 CPU 申请中断。显示部分由 8 位七段共阴 LED 管组成,当暂时不用显示时,可以消隐,节省电源开销。

需要注意的是,单片机的 ALE 信号是经过晶体分频得到的标准时钟,但在读、写外部地址时由于脉冲的丢失,大大影响了 ALE 的精确度,因此需要经过 ALE 修复电路,把丢失的脉冲找回,这样 ALE 才能作为标准频标来使用。

5. 模数转换和数模转换部分

在单片机应用系统设计中,经常会涉及模数(A/D)转换和数模(D/A)转换,因此在通用 MCS-51/52 单片机最小系统中内置 A/D 和 D/A 模块电路通常会给应用系统设计带来便利。A/D 和 D/A 可供选择的芯片种类很多,设计中要根据具体的需求做出合理的选择,而在最小系统设计时通常选用能够满足

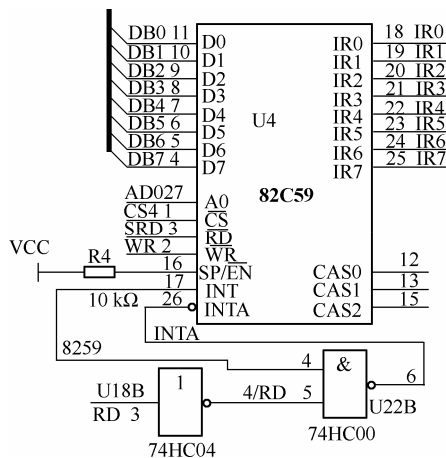
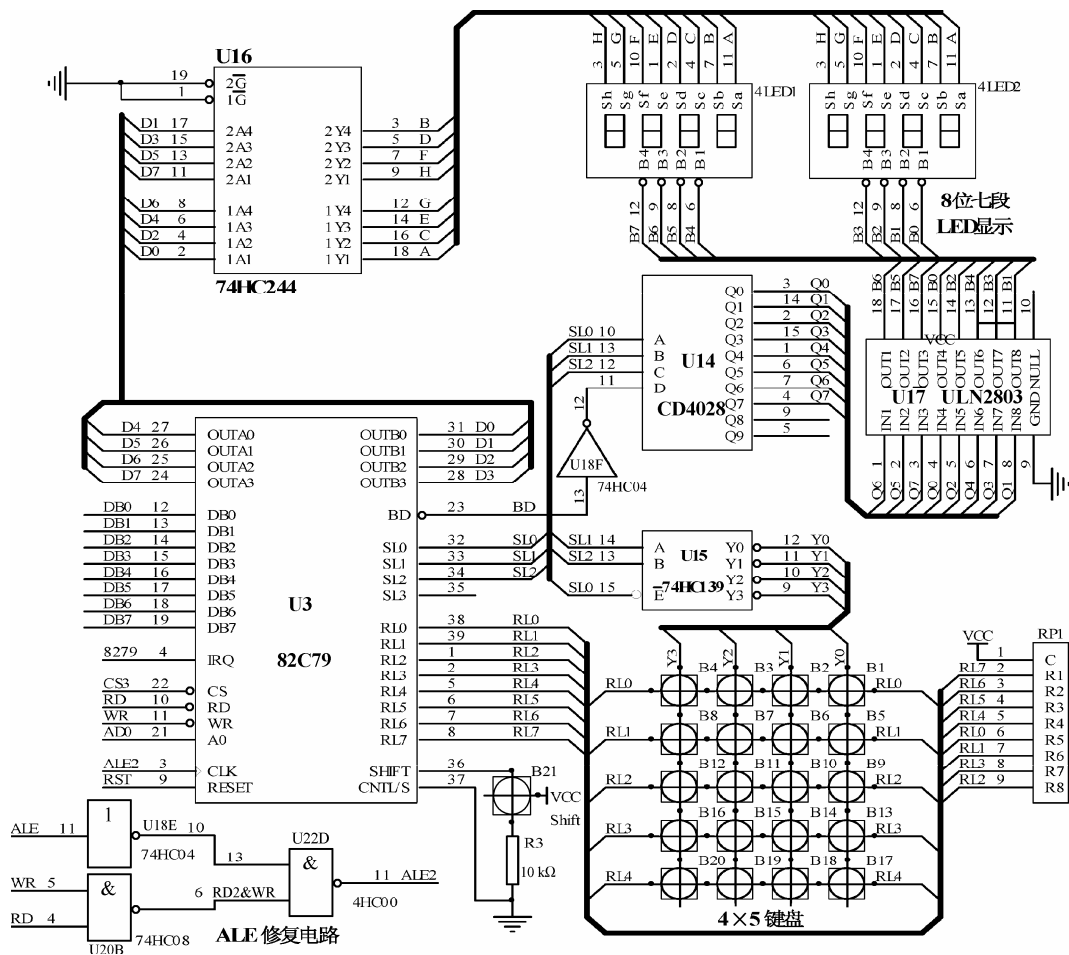


图 2.2 采用 82C59 扩展中断原理图

一般系统需求的芯片即可。这里给出参考的 A/D 和 D/A 模块电路设计方案如图 2.4 所示。



系统内置 8 位的 D/A 芯片 DAC0832, 电流稳定时间为 $1\ \mu\text{s}$; 单缓冲输入; 模拟输出电压范围为 $-V_{\text{REF}}\sim 0\ \text{V}$ 。它可以直接用来输出各种波形信号和音频信号, 也可作为精密系统的控制电平。

6. RS232 串行接口部分

MCS-51 单片机内部有一个全双工串口, 该串口有 4 种工作方式, 以供不同场合使用。为了方便单片机系统与微机、打印机、外设等的接口, 可在通用 MCS-51/52 单片机最小系统中添加一个通用 RS232 接口, 这样能很方便地与微机和外设等进行串口通信。其参考电路如图 2.5 所示。

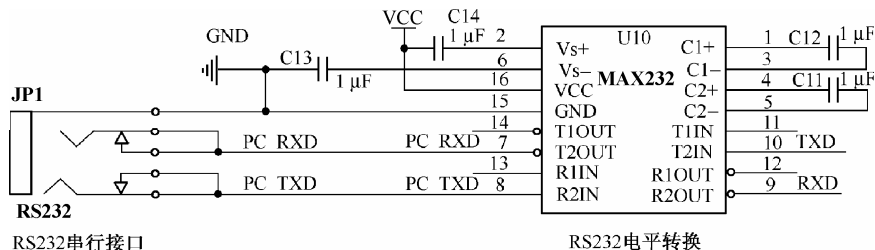


图 2.5 RS232 串口模块电路

单片机的串口电平是 0 V, 5 V, MAX232 将它转化为标准的 RS232 通信电平 +10 V, -10 V。

7. 硬件看门狗电路和电源监控电路

由于软件缺陷等原因引起的程序跑飞是令人头痛和常见的问题之一。通过在通用 **MCS-51** 单片机最小系统中添加硬件看门狗电路,由跳线来设置,自行监测系统运行,可以有效地提高系统的可靠性与抗干扰性。硬件看门狗利用了一个定时器来监控主程序的运行,也就是说,在主程序的运行过程中,要在定时时间到来之时对计数器进行复位。如果由于干扰或者程序缺陷造成程序跑飞,而没有对计数器及时清零,那么看门狗计数器将会溢出,从而让系统复位,保证了系统安全。

在电源不稳定或有强大的干扰源时，系统会出现异常情况，给系统的开发和实际应用带来极大不便。通用 MCS-51/52 单片机最小系统中采用专用的电源监控芯片，当系统电压下降到一定的程度，它会发出复位信号，保证系统正常工作。此外，它还提供了一个监视外设电源电压的引线，当外设电压下降到一定程度时，可直接利用中断向 CPU 提出电源故障申请，以便系统及时地进行处理。由 MAX813 构成的看门狗电路和电源监控电路的参考原理图如图 2.6 所示，它主要实现系统正常复位、电源电压干扰引起的系统复位、电源以外的干扰引起的系统复位及监视其他电源电压的功能。

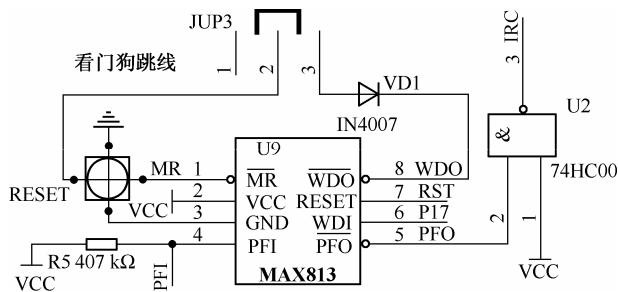


图 2.6 看门狗电路和电源监控电路的参考原理图

8. 其他功能

通用 MCS-51/52 单片机最小系统可对外提供数据总线、地址总线和控制总线等三大总线，以备扩展使用。

2.2.2 通用MCS-51/52 单片机最小系统的应用实验举例

整个通用 MCS-51/52 单片机最小系统的电路设计原理图如图2.7所示(见插页)，它是后面实现单片机应用系统的基础。下面将对基于此最小系统的相关应用实验和程序设计做一介绍。

1. 键盘显示模块实验

此实验基于最小系统编写程序，实现按键控制并显示相应键值 0~F 等(不含 Shift 键的 20 个键)。由图2.7可知，82C79 数据口地址为 B000H，82C79 命令口地址为 B001H，参考的程序示例如下。

```

org 0000h
    sjmp start
org 0013h
    sjmp int1
start:  mov dptr, #con_8279      ;82C79 控制口地址
        mov a, #34h            ;设定 82C79 的工作频率为 2 MHz/20 = 100 kHz
        movx @dptr, a
        mov a, #0              ;设定 82C79 显示及键盘工作方式
        movx @dptr, a
        setb ex1               ;开 INT1 中断
        clr it1               ;设为电平触发方式
        setb ea               ;开中断
        sjmp $
int1:   clr ea                 ;关中断
        mov dptr, #data_8279   ;82C79 数据口地址
        movx a, @dptr          ;读入键码
        mov 40h, a             ;键值存入 40H 单元
        mov dptr, #keytable    ;键值表地址
        mov r0, #0            ;设键值初值
re1:    mov a, r0
        movc a, @a+dptr        ;查询与读入键码相对应的键值
        cjne a, 40h, next
        sjmp out               ;查到与读入键码对应的键值则结束
next:   inc r0
        simp rel               ;未查到与读入键码相对应的键值则继续
out:    mov a, r0
        mov dptr, #disptable   ;显示段码表地址
        movc a, @a+dptr        ;查对应显示段码
        mov dptr, #data_8279   ;82C79 数据口地址
        movx @dptr, a          ;送段码到 LED 显示
        setb ea
        reti
keytable:
        db 30h, 20h, 10h, 00h, 31h, 21h, 11h, 01h, 32h, 22h, 12h, 02h, 33h,
        23h, 13h, 03h
        db 34h, 24h, 14h, 04h
disptable:
        db 3fh, 06h, 5bh, 4fh, 66h, 6dh, 7dh, 07h, 7fh, 6fh
        db 77h, 7ch, 39h, 5eh, 79h, 71h, 73h, 76h, 38h, 37h
end

```

2. A/D转换模块实验

此实验基于最小系统编写程序，利用 8259 对中断进行扩展，改变 A/D 模拟输入，ADC0804 采样 16 次，送入以 30H 为起始地址的存储单元。ADC0804 的控制字地址为 A000H，参考的程序示例如下。

```

org 0000h
    sjmp start
org 0003h
    sjmp int
start: mov dptr, #even_8259      ;8259 偶地址
      mov a, #76h              ;设定 8259 工作方式
      movx @dptr, a
      mov dptr, #odd_8259      ;8259 奇地址
      mov a, #12h              ;送入中断向量高位入口地址
      movx @dptr, a
      mov dptr, #odd_8259      ;8259 奇地址
      mov a, #11111101b        ;允许 IR1 中断
      movx @dptr, a
      mov dptr, #con_0804
      movx @dptr, a            ;启动 A/D 转换
      mov r0, #30h            ;存储单元起始地址
      setb ex0                 ;开 INT0 中断
      setb it0                 ;令 INT0 为边沿触发
      setb ea                 ;开所有中断
      simp $
int: mov dptr, #even_8259      ;8259 偶地址
     movx a, @dptr            ;读 CALL 操作码, 丢弃
     movx a, @dptr            ;读低 8 位中断矢量
     mov dpl, a               ;存入 DPTR 低 8 位
     movx a, @dptr            ;读高 8 位中断矢量
     mov dph, a               ;存入 DPTR 高 8 位
     clr a                    ;清累加器 A
     jmp @a+dptr              ;转入 IR1 上中断的服务程序
org 1264h
IR1:  sjmp cont
cont:  mov dptr, #con_0804      ;0804 控制口地址
     movx a, @dptr            ;读入数据
     mov @r0, a               ;送入存储单元
     inc r0
     cjne r0, #40h, next       ;存储单元地址是否为 40H, 即是否完成 16 个数据
     clr ea                    ;若采完, 关中断
     sjmp out                  ;跳出中断
next:  mov dptr, #con_0804      ;0804 控制口地址
     movx @dptr, a            ;启动下次 A/D 转换
     mov dptr, #even_8259      ;8259 偶地址
     mov a, #20h
     movx @dptr, a            ;非指定 EOI 命令送 8259
out:  reti
end

```

3. D/A转换模块实验

此实验基于最小系统编写程序，使 D/A 输出矩形波和正三角波。其中矩形波的占空比为 2/3，频率为 1 kHz。DAC0832 控制字地址为 9000 H，参考的程序示例如下。

(1) 矩形波发生程序

```

org 0000h
    sjmp start
start:    clr f0                ;高、低电平标志位初始化
    mov dptr, #con_0832
nextDA:  jb f0, highbit
    mov a, #0h                ;低电平的电平值
    mov r7, #107              ;低电平延时时间参数
    sjmp outDA
highbit: mov a, #0ffh          ;高电平的电平值
    mov r7, #214              ;高电平延时时间参数
outDA:   cpl f0                ;高、低电平切换
    movx @dptr, a             ;启动 D/A 转换
    acall delay               ;延时
    sjmp nextDA
delay:   nop                  ;延时子程序(R7 是延时的参数)
    djnz r7, delay
    ret
end

```

(2) 正三角波发生程序

```

org 0000h
    sjmp start
start:   mov b, #0             ;电平值初始化
    clr f0                     ;电平增减标志位
    mov dptr, #con_0832
nextDA:  mov a, b
    jnz nochange               ;是否改变增减标志判断
    cpl f0
nochange:jb f0, down
    dec b                      ;电平减 1
    sjmp outDA
down:    inc b                 ;电平增 1
outDA:   movx @dptr, a         ;启动 D/A 转换
    sjmp nextDA
end

```

4. 串口通信模块实验

此实验基于最小系统编写程序，实现 PC 和单片机的通信。当 PC 响应键值时，向单片机发送键值；单片机接收到 PC 发送的键值，在 LED 数码显示器上显示所按键的 ASCII 码值，并返还给 PC；PC 接收到单片机发送来的键值，在 PC 屏幕上显示键值。参考的程序示例如下。

(1) 单片机接收程序

```

org 0000h
    sjmp start
org 0023h
    ljmp serve
start:  mov tmod, #20h           ;定时器 T1 初始化
        mov th1, #0f3h
        mov tl1, #0f3h
        mov scom, #50h         ;串行口初始化
        mov pcon, #80h         ;SMOD = 1
        setb tr1               ;启动定时器 T1
        setb ea                ;开中断
        clr ti                 ;清 TI
        setb es                ;允许串行中断
        sjmp $
serve:   clr ea                 ;关中断
        clr ri                 ;清接收中断标志
        mov a, sbuf            ;接收 PC 发过来的数据
        cjne a, #1bh, senddata ;若收到通信结束命令, 则退出中断
        sjmp return
senddata:mov sbuf, a           ;将数据送回给 PC
wait:    jnb ti, wait          ;发送器不空则等待
        clr ti
return:   setb ea              ;开中断
        reti
end

```

(2) PC 收发程序, 按键并发送, 接收字符, 并在 PC 屏幕上回显程序

```

data segment
    send    db'    Send to MCS-51/52: "'
    sendnum db 1 dup(?)
    endsend db' "', 0dh, 0ah, '$'
    rece    db' Receive from MCS-51/52: " '
    recenum db 1 dup(?)
    endrece db' "', 0dh, 0ah, 'ESC to quit program.', 0dh, 0ah, 0dh, 0ah, '$'
data ends
code segment para public 'code'
start pro far
    assume cs:code, ds:data
    cli
main:  push dx
        mov ax, 0
        push ax
        mov ax, data
        mov ds, ax
        mov dx, 2fbh
        mov al, 80h           ;通信控制寄存器第 7 位置 1, 以便设置波特率

```

```

out dx, al
mov dx, 2f8h           ;设置除数锁存器低位
mov al, 18h
out dx, al
mov dx, 2f9h           ;设置除数锁存器高位
mov al, 0
out dx, al
mov dx, 2fbh           ;设定数据格式, 8 个数据位, 1 个停止位, 无校验
mov al, 03h
out dx, al
mov dx, 2fch           ;设置 MODEM 控制信号
mov al, 03h
out dx, al
mov dx, 2f9h           ;禁止所有 8250 中断(4 种类型)
mov al, 0
out dx, al
mov dx, 2fdh
in al, dx
test al, 01h
jz forever
mov dx, 2f8h           ;复位 PC 的接收寄存器
in al, dx
forever: mov ah, 1       ;检查键盘缓冲区, 无字符则循环等待
int 16h
jz forever
mov ah, 0               ;若有, 取键盘字符
int 16h
sendkey: mov dx, 2f8h    ;发送键入的字符
out dx, al
cmp al, 1bh             ;判断是否是通信结束键
jz outhere
mov sendnum, al
lea dx, send
mov ah, 9
int 21h                 ;显示发送提示及发送的字符
receive: mov dx, 2fdh
in al, dx
test al, 01h            ;检查接收数据是否准备好, 未准备好继续查询
jz receive
mov dx, 2f8h            ;从接收寄存器中读取数据
in al, dx
and al, 7fh             ;去掉无效位, 得到数据
mov recenum, al
lea dx, rece
mov ah, 9
int 21h                 ;显示接收提示及接收的字符
jmp forever

```

```
outhere: mov ah, 4ch
          int 21h
start     endp
code      ends
end start
```

5. 其他模块实验

除了前面的模块实验,在充分掌握系统设计原理和编程方法的基础上,利用该通用 MCS-51/52 单片机最小系统,还可以实现打印模块实验、各种综合实验和各种外设接口实验等。

2.3 基于单片机系统的简易电子琴的设计与实践

1. 设计任务

设计并制作一个基于单片机系统的简易电子琴。

2. 设计基本要求

- (1) 手动演奏。能手动演奏 4 个八度音的乐音,音色纯正,且音长由按键按下持续时间控制。
- (2) 自动演奏。能存储并播放 6 首以上的不同曲目,每首曲长度(即音符个数)不少于 50 个,音调、音色优美;可选曲重复演奏。
- (3) 功能开关(手动,自动切换)可由硬件拨动开关构成,也可由键盘完成手动、自动切换。
- (4) 功率 $\leq 500\text{ mW}$,音质悦耳,无失真。
- (5) 写出内容翔实的实验报告并附程序清单。

3. 设计提高部分要求

- (1) 键盘音域扩展(变调)。
- (2) 语音提示功能(可采用语音合成)。
- (3) 频谱合成以产生不同音色(如钢琴、小号等)及和弦。
- (4) 由程控增益放大器产生“强弱强”或“弱弱强”等节奏。

2.3.1 设计任务的分析及设计模型的建立

本训练的设计任务为设计并制作一个以单片机为控制核心的简易电子琴,要求通过按键选择来实现乐曲的手动演奏和自动演奏。乐曲是由不同频率和节拍的音符组成的,因此如何通过单片机来控制乐音的产生是本系统设计的关键所在。

1. 乐音产生的原理分析

音的物理属性有高低、长短、强弱和音色 4 个方面。若不考虑音色,音有音调、音长和音强这三种基本特性。音调、音强和音长分别与声波的频率、振幅和持续时间这三个物理量相对应。

- **音调** 音调又称为音高,即听觉所分辨声音高低的属性。音的高低是由物体在一定时间内振动次数(频率)的多少决定的,频率越高,则音调越高。
- **音强** 当发音体的振动频率一定时,音强决定于其振幅,振幅越大,音越强。
- **音程** 两个音的音调之间的距离称为音程。两个音的频率比为 2 : 1 时的音程称为倍频程,在音乐学中称为八度。

- **音阶** 音阶由一系列向上或向下逐级进行的音构成,各音之间具备一定的音程关系。
- **十二平均律** 若将一个倍频程(八度)分为 12 个等分的音阶,则每一个音阶就称为半音,每隔一个音阶为一个全音。

如上所述,频率决定音乐的音高(音调)。乐音有八十四音阶,八十四音阶中最低音频率为 21.827 Hz,每高八度则频率加倍。一个八度音音程内分为 12 个半音,每相邻半音比值相等而构成十二平均律,其中任何两相邻半音频率的比值为 $K=\sqrt[12]{2}=1.0595$,据此可得所有八十四音的频率。

2. 简易电子琴设计模型的分析

声是一种机械扰动在气态、液态、固态物质中传播的现象。所谓扰动,是指在上述物质中的密度、压力或是速度上的某种微小变化,这个变化在弹性介质中会传播出去,传递的能量就是声。或者说,物质的机械扰动即可产生声波。

在本训练中要求由电信号产生声音,通常可使用扬声器来实现。根据声学知识得知,只要给扬声器馈送一个按规定频率变化的电流,扬声器就会发生与该电流相同频率的声音,所以问题归结为如何使计算机具备发声的环境和产生具有一定频率值的振荡信号。

乐曲是由不同音调的节拍组成的,因此控制馈送至扬声器的驱动脉冲的频率和持续时间是产生音乐的关键所在,也是本系统设计的切入点。

位触发方式(位控法)产生音乐的基本原理是:位触发方式是根据音调频率算出周期从而确定初值的,使单片机的定时器每 1/2 周期中断一次,并在中断服务程序中将某一指定的 I/O 口取反,即可在该端口输出该频率的方波,将其接到扬声器即可产生对应于该音调的乐音。实际设计中因方波含有高次谐波,若直接驱动扬声器,音质较差,应采用平滑滤波处理。

因此,简易电子琴系统的设计思路可利用位控法,根据按键选择,通过单片机的定时器来产生给定的音调和音长的方波频率信号,经过功放电路后驱动扬声器,从而得到对应的乐音。基于单片机的简易电子琴系统的设计框图如图 2.8 所示。

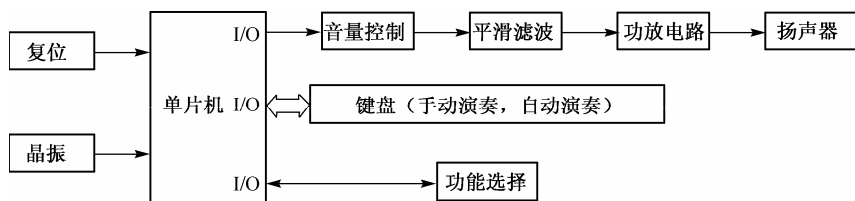


图 2.8 基于单片机的简易电子琴系统的设计框图

3. 简易电子琴设计的难点分析

(1) 音调的控制

音调控制的关键就是控制输出信号频率。采用位触发方式,利用延时程序控制“高”、“低”电平持续时间,从而改变输出频率,改变音调。具体实现方式可采用 MCS-51/52 系统的定时/计数器来完成。编程时,只需将音符的频率转化为控制脉冲宽度的计数值,根据不同的按键或音乐的音调,在程序中给定时器送入不同的初值,调节定时器的溢出时间以实现音调的控制。位触发方式产生的声波信号示意图如图 2.9 所示。

设音频为 f_M , 则脉冲周期为 $1/f_M$, 一个半波(脉宽)所需时间为 $1/(2 \times f_M)$ s, 用这个时间来控制端口输出脉冲的延时时间,也就控制了脉冲频率,从而驱动扬声器产生给定频率的乐音。因此,在设计中应计算出音频、音长及定时器初值之间的对应关系。

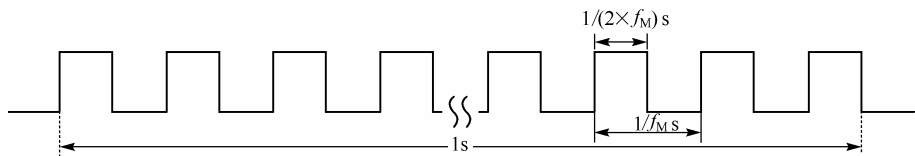


图 2.9 位触发方式产生的声波信号示意图

定时时间常数(初值)算法如下：置定时器 T_1 为工作方式 1，即 TH_1 和 TL_1 构成 16 位定时器，根据 $T_M = 1/f_M$ ，计算出脉冲信号的周期；将 T_M 除以 2，得 t ；由公式 $t(\mu s) = (2^{16} - N) \times \frac{12}{f_{osc}(\text{MHz})}$ ，即可求出定时器初值。

按照每个音阶对应的频率，分别计算出对应的定时器初值，在片内 ROM 中形成一个 4×7 的矩阵，这就是音阶表。为简化设计难度，表 2.1 给出了计算得到的音阶表，其中一个音阶的定时初值占用两字节。

表 2.1 4×7 音阶表

音区 \ 音符		1	2	3	4	5	6	7
	频率 / Hz	262	294	330	349	392	440	494
低音区	初值(H) Hex	F88B	F95B	FA14	FA67	FB03	FB8F	FC06
	频率 / Hz	524	588	660	698	784	880	988
中音区	初值(H) Hex	FC45	FCA1	FD09	FD33	FD81	FDC7	FE05
	频率 / Hz	1048	1176	1320	1396	1568	1760	1976
高音区	初值(H) Hex	FE22	FE56	FE89	FE99	FEC0	FEE3	FF02
	频率 / Hz	2096	2352	2640	2792	3136	3520	3952
高高音区	初值(H) Hex	FF11	FF22	FF43	FF4B	FF61	FF72	FF82

(2) 音长的控制

音长(节拍)决定了音频的持续时间，音的长短和强弱决定了节奏，音乐中规定 94 拍 / 分，可算出节拍时间，如表 2.2 所示。

表 2.2 节拍时间表

音符/节拍	1/8 拍	1/4 拍	2/4 拍	3/4 拍	1 拍	6/4 拍	2 拍	3 拍	4 拍
节拍时间/s	0.08	0.16	0.32	0.48	0.64	0.96	1.28	1.92	2.56
时间常数	08H	10H	20H	30H	40H	60H	80H	C0H	FFH

音长控制可以采用延时程序定时或者定时器中断的方式控制节拍来实现。

采用延时程序定时的方法，给定延时初值控制循环次数来实现各种节拍的 control。此方案避免了中断嵌套使程序结构简单，但缺点是使用延时程序定时效率低下。

采用定时器中断的方式控制节拍，根据音符节拍所对应的时间常数，向定时器 T_0 送入一个固定的初值，控制中断循环的次数，从而得到成倍数关系的时间间隔，即为各节拍数。此方案程序运行效率高，并且在中断服务程序中加入控制子程序便于控制其他功能；但缺点是使用中断嵌套，增加了程序的复杂程度。

以采用定时器中断的方式控制节拍方案来讨论，如表 2.3 所示，按一拍 0.64 s 计算，取 1/4 拍为最小定时间隔，即 0.04 s，因此设定 T_0 的初值为 63C0H。同样，在片内 ROM 中形成一个 1×4 的表格，

其数值分别为 2, 4, 8, 16。通过控制中断循环次数(也就是引用次数)来控制输出对应于音调的方波的持续时间,从而实现对音长的控制。

表 2.3 常用音乐节拍及时间常数表

音乐节拍	1/8	1/4	1/2	1
节拍时间/s	0.08	0.16	0.32	0.64
时间常数	2×0.04	4×0.04	8×0.04	16×0.04
循环次数	2	4	8	16

(3) 手动演奏

在手动演奏过程中,音阶由按键决定,音长由弹奏者按键时间的长短决定。程序不断查询键盘状态,一旦有键按下,立刻算出键值,查频率表取出初值,给定时器 T₁送初值,启动计数,开始发声;然后继续监视键盘。一旦键盘释放,立即停止计数,返回查询,等待下一次按键。在手动演奏中,音高的选择可由硬件开关或者键盘定义,其控制方式在原理上是相同的。手动演奏和自动演奏由一位开关控制切换,其控制逻辑由设计者确定。

(4) 自动演奏

在自动演奏中,音长由音长代码决定,对一首歌曲的乐谱编码后形成一段连续的数据,而乐谱中的每一个音符具有音阶、音高和音长三种属性,所以编程中将乐谱每一个音符的音阶、音高和音长定义为三个数据表。程序根据乐谱表中每个音符的音阶及音高来确定音符对应的频率,同时根据音长取出相应的节拍计数初值以控制这一音调的长短。

(5) 乐谱代码

在自动演奏中,通过查询乐谱代码来控制音乐。那么如何有效地实现乐谱代码的编码呢?每一个乐谱代码包括音长、音高和音阶三部分,如果用三个字节的空间来存储,则需将音长码、音高码和音阶码分别存储于一个字节内,这样读代码时方便,不需要进行代码的翻译,但需要占用较大的程序空间。在满足存储 6 首以上不同曲目的题目要求下,需占用到较大的存储空间,因此这种方法是不可取的。考虑到常用音乐节拍 4 种、音高 4 种和音阶 7 个,因此可以把每个乐音代码存在同一个字节内,如表 2.4 所示,将音阶存低三位,音高存四五位,音长存六七位,最高位为结束符。这样虽然译码相对麻烦,但乐音代码占用空间较小,每个乐谱代码比前一方案可节省 2/3 的空间。

表 2.4 可达 4 个八度音的乐谱代码

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
结束符	音长		音高		音阶		

代码由一个字节组成。音长码存储的是表 2.3 中循环次数,00, 01, 10 和 11 分别对应着 2, 4, 8 和 16;音高码存储的是表 2.1 的行数,00, 01, 10 和 11 分别对应低音区、中音区、高音区和高高音区;音阶码存储的是表 2.1 的列数,000~111 分别对应着 1~7 和休止符;代码定义字节的最高位为 1 的字符为结束符,当程序读到最高位为 1 时,停止播放;休止符表示停顿,即不发音,它只有时间长短的属性,休止符定义为代码的低三位,即 111B。

2.3.2 系统具体的设计和实现

1. 系统硬件设计

(1) 系统复位电路的设计

任何含有计算机的系统,在启动运行时都需要复位,以便 CPU 和系统中的其他部件都处于某一个

确定的初始状态,并从这个状态开始执行工作。同样,单片机在外界的干扰下出现程序跑飞或者进入死循环的状况时,需要人为地进行复位操作,恢复正常状态。因此,手动复位是微机子系统的一个基本功能要求。

系统复位电路可采用的电路形式多样,可参考的系统复位电路如图2.10所示。

(2) 功放电路的设计

功放电路的设计应能满足设计任务的指标要求,在这一前提下,可采用分立器件或者集成功放电路等多种电路形式。相比而言,采用集成功放电路设计难度较小,较为简捷的设计可使用专用音频集成功放 LM386。

LM386 是 8 引脚 DIP 封装,消耗的静态电流约为 4 mA,是应用电池供电的理想器件。该集成功率放大器同时还提供电压增益放大,其电压增益随着外部连接的变化可在 20~200 倍范围内调节。其供电电源电压范围为 4~15 V,内部没有过载保护电路。功率放大器的输入阻抗为 50 k Ω ,频带宽度为 300 kHz。

可参考的功放电路如图2.11所示。

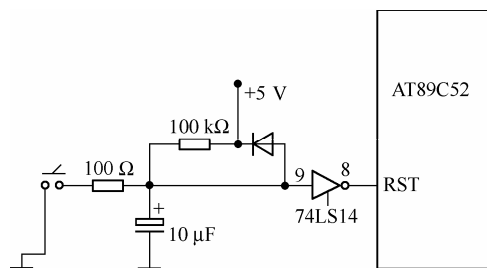


图 2.10 系统复位电路

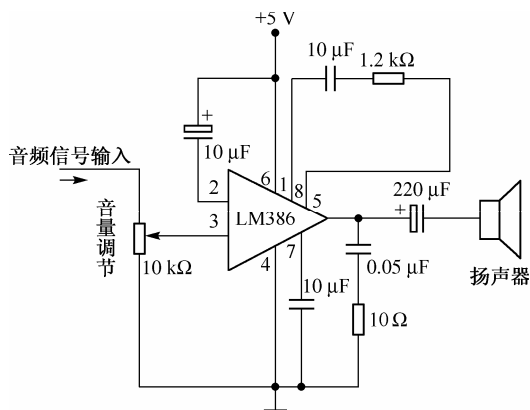


图 2.11 功放电路

(3) 键盘的设计

键盘的设计是本训练课题的重点之一。一般来说,微机子系统的键盘接口有两种形式。第一种是独立式按键接口。它的特点是每个按键独立地占用一个 I/O 口线,故在按键较少(不超过 8 个)时使用。第二种是矩阵式键盘接口。它的特点是将多个按键排列成一个 M 行 N 列的按键矩阵。这种方式只需使用 $M+N$ 根 I/O 口线来控制 $M \times N$ 个按键,而使用独立按键接口则需要使用 $M \times N$ 根 I/O 口线,其中的差别是显而易见的。然而,设计本身就是一个选择的过程,并非仅是由于资源占用及管理难度上的差异使得某一方案不被选择。设计者应综合考虑设计要求及可实现性、效率等方面的因素,做出选择。下面将对这两种方案分别进行讨论。

① 矩阵键盘设计方案

手动演奏需要用到低音、中音、高音和高高音分别对应的 7 个音阶键及功能切换键;自动演奏需要用到曲目选择键、暂停键、曲目切换键和功能切换键。考虑到音阶键和曲目选择键可以复用,系统总共需要至少 29 个按键。键盘设计可以采用 8×4 的矩阵键盘方案,这样按键一目了然,软件控制比较简单。

② 独立式按键设计方案

采用 7 个单独键及 2 个控制键的 4 个状态实现 28 个音阶,音阶键和曲目选择键复用,1 个单独键作为功能切换键(手动、自动切换),这样可以节省系统硬件资源,同时也便于系统的扩展,但软件设计相对复杂。

参考的独立式按键设计方案和所有按键功能如下。

P2.7 口所对应的拨动开关为功能切换键，P2.7 口为高电平时实现手动演奏，P2.7 口为低电平时实现自动演奏。

P2.0 口和 P2.1 口所对应的按键构成了手动演奏的音高，具体为：00H 为低音，01H 为中音，10H 为高音，11H 为高高音。

P2.0 口所对应的按键复用为自动演奏的暂停键。

P2.1 口所对应的按键复用为自动演奏的曲目切换键，即当一首歌正在播放时，按此键后再按曲目选择键可任意切换到其他曲目。

P0 口所对应的按键复用，实现音阶和曲目选择。按键对应的键值和功能如表 2.5 所示。

表 2.5 按键键值表

按键 功能		Key1 (P0.0)	Key2 (P0.1)	Key3 (P0.2)	Key4 (P0.3)	Key5 (P0.4)	Key6 (P0.5)	Key7 (P0.6)	开关 1 (P2.0)	开关 2 (P2.1)
自动演奏		第一首	第二首	第三首	第四首	第五首	第六首	第七首	暂停	曲目 切换
手 动 演 奏	低音	do	re	mi	fa	so	la	si	0	0
	中音	do	re	mi	fa	so	la	si	0	1
	高音	do	re	mi	fa	so	la	si	1	0
	高高音	do	re	mi	fa	so	la	si	1	1

(4) 系统硬件电路的具体实现

根据系统键盘的实现方案、系统复位电路、功放电路和晶振的常规接法，给出参考的系统硬件设计电路如图2.12所示。

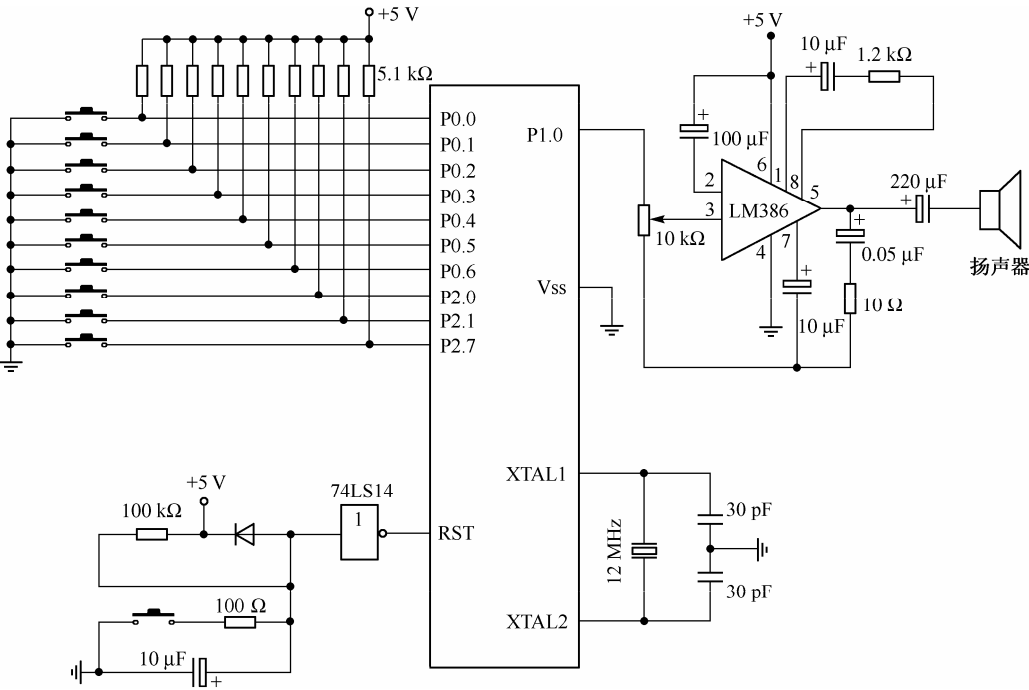


图 2.12 系统硬件设计电路图

2. 系统软件设计

参考的系统软件设计流程如图2.13所示。

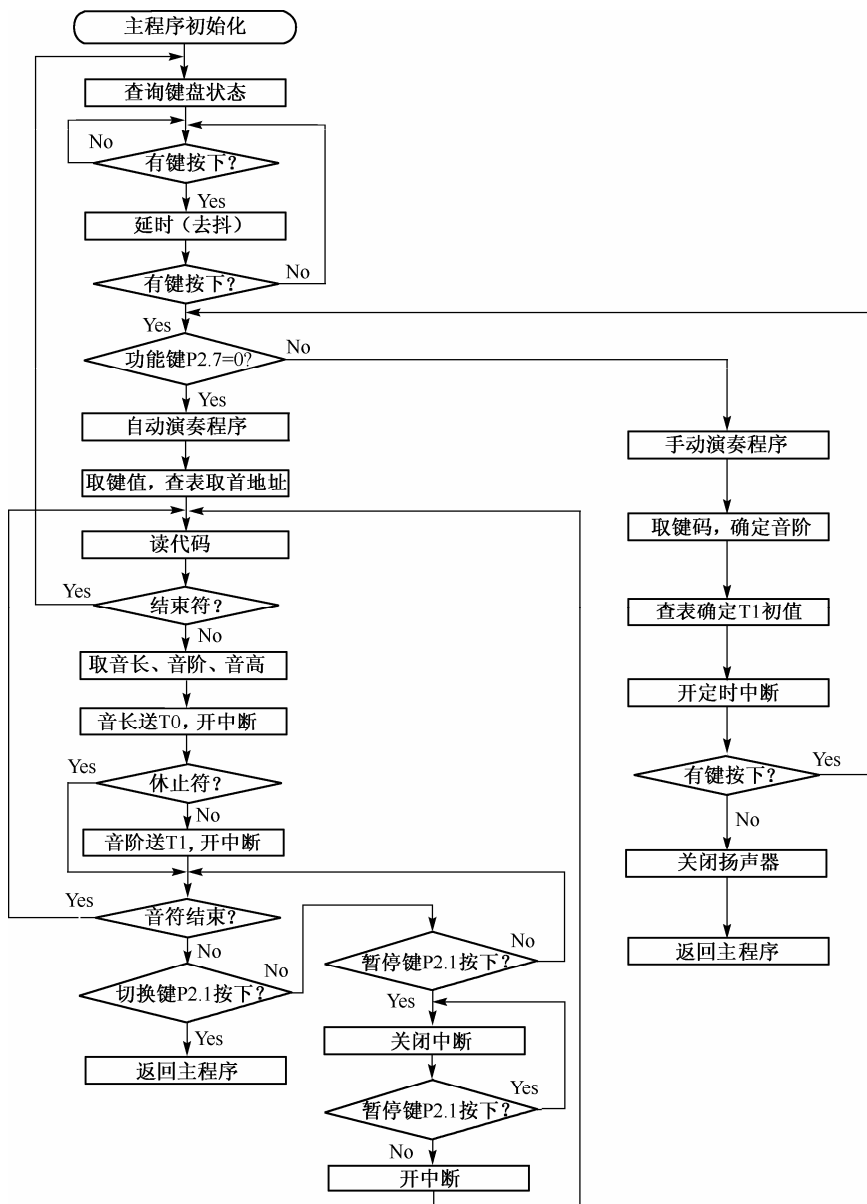


图 2.13 系统软件设计流程图

2.3.3 提高部分难点提示

1. 语音提示功能

为简易电子琴系统增加语音提示功能，比如在自动演奏中，在乐曲的演奏前，添加类似于“第*首曲：*****”这样的语音提示。语音提示功能可以通过语音合成芯片实现，通过单片机控制语音芯片将文本转化成音频输出。

2. 由程控增益放大器产生“强弱强”或“弱弱强”等节奏

原设计系统中可以通过调节 LM386 的 3 引脚和地之间的滑动变阻器来实现音量可调,但是对于同一组乐音是没有实现节奏“强弱”变化的。要产生强弱变化的节奏,在节奏变化时,通过单片机控制多路选择开关切换电阻实现程控放大,以达到音量强弱变化的效果。

2.4 LED显示屏系统的设计与实现

1. 设计任务

设计并制作一个 LED 点阵显示系统。

2. 设计基本要求

基于单片机系统的单色 LED 点阵显示屏设计的要求如下:

- (1) 显示图形、中西文字、三条以上广告用语及图形的播放。
- (2) 模块化设计,若干模块连接可组成给定大小的显示屏。
- (3) 可选择广告用语。
- (4) 显示方式的变化功能(如上移、下移、左移、右移、旋转),显示速度的键盘控制(加速、减速)。
- (5) 完成单色 LED (16×128) 显示屏条屏的设计与实现。

单色 LED 显示屏的主要技术指标如下:

- 像素结构 LED 发光二极管(红);
- 像素直径 5 mm;
- 像素点距 10.25 mm;
- 解析度($w \times h$) 16×16 ;
- 画面转换速度 60 帧/秒。

3. 设计提高部分要求

三色 LED 显示屏的设计要求在两个系统平台上分别给予实现。第一,在单片机系统上予以实现;第二,在 PC 上实现,要求能够在 LED 屏上实时显示监视器上的内容。

- (1) 红、绿、黄、黑、彩色图形、文字显示,画面清晰无闪烁。
- (2) 模块化单元驱动电路设计,若干模块连接可组成给定大小的显示屏。
- (3) 12 种颜色配置循环显示,集成 ASCII 码字库及三条以上中文广告语。
- (4) 通过按键键盘控制显示广告语的选择功能。
- (5) ASCII 码的输入及输入 ASCII 码的循环显示。
- (6) 32 种动画效果。

(7) 显示方式的变化功能(如上移、下移、左移、右移),显示速度的键盘控制(加速、减速)及中英文显示。

- (8) 256 灰度级控制(本要求实现难度较高,可酌情选做)。

三色 LED 显示屏的主要技术指标如下:

- 像素结构 LED 红(2), LED 绿(2);
- 像素直径 10 mm;
- 像素点距 14.52 mm;

- 解析度($w \times h$) 32×16;
- 画面转换速度 70 帧/秒。

2.4.1 设计任务的分析

LED 显示屏是随着计算机及相关的微电子、光电子技术的迅猛发展而形成的一种电子广告媒体,其应用日益广泛。LED 电子显示屏的像素采用 LED 发光二极管,将多个发光二极管以点阵的方式排列起来便构成了 LED 阵列。LED 显示屏的迅速发展首先应归功于 LED 的卓越性能。

- 响应时间短。迄今的多种 LED 响应时间均在 100 ns 以下,因此为快速动态显示提供了可能。
- 高亮度。现代技术生产的 LED 管芯有高亮、超高亮、特高亮等种类,发光强度高达 1000 mcd(以单管 $\phi 5$ 测试),高亮度使 LED 屏幕远距离观看和户外显示成为可能。
- 发光光谱窄,光色纯正。更由于蓝色发光二极管的生产使全彩 LED 显示屏的实现成为可能。
- 发光寿命长,一般可达 10 亿小时以上。

基于 LED 在显示上的优越性,LED 显示屏的设计与实现具有较高的实用价值。

本训练可以分为两个不同的模型,基本部分是一个基于单片机的单色 LED 显示屏系统的设计,通过键盘选择调用预存的广告用语或图形画面,由单片机来控制 LED 屏动态显示。而提高部分作了更高要求,基于 PC 平台,要求在三色 LED 显示屏上实时显示监视器屏幕上的内容,这就需要通过特定的接口将监视器屏幕上显示的数据信息传输到上位机,上位机经过实时处理、转换并存储后,再来控制 LED 屏实时显示。

归纳起来,LED 显示屏系统主要由驱动模块、接口电路和控制电路三部分组成。

对于基于单片机的 LED 显示屏系统来说,单片机作为控制核心,从数据存储器读出字模数据,通过接口电路传输到驱动模块,然后由驱动模块驱动 LED 阵列,即可实现显示。接口电路的设计取决于单片机需要给驱动模块传送的信号,而驱动模块是包括 LED 阵列的驱动显示电路,它的设计主要是根据 LED 屏的大小及对于显示的要求。LED 显示屏的主要技术指标(刷新速率、显示亮度、显示色彩等)均由驱动模块直接实现。只有确定驱动模块的工作方式才能确定接口电路和控制电路需提供的信号。因此,系统实现的难点和切入点应归结于驱动模块的设计。

2.4.2 以单片机为控制核心的LED显示屏的设计

1. 原理分析与方案比较

单片机对 LED 显示屏的显示控制,一般是先从数据存储器读取数据,然后将数据传送给驱动模块进行显示。此系统的关键部分在于如何控制 LED 驱动和如何快速实现数据的有效传输,因此设计方案的比较主要体现在显示方式的选择及输出数据的不同形式。

(1) 显示方式选择及输出数据形式的方案论证

① 显示方式的方案比较论证

LED 显示屏是以发光二极管为像素,由 LED 点阵显示单元拼装而成的。根据驱动方式的不同,LED 屏幕显示方式可分为静态显示和动态扫描两种方式。

a) 静态显示

所谓静态显示,是对 LED 电子显示屏中每一像素点都通过硬件单独控制,整个显示屏的显示实际上是所有 LED 的同时显示,一幅画面输入以后要保持到下一幅画面的输入。

b) 动态扫描显示

所谓动态扫描显示，是将整屏画面分为若干部分分别进行刷新，利用人眼的视觉暂留特性而实现的一种显示方法。动态扫描采用分时选通技术，通过行驱动管的分时工作，使得每行 LED 的点亮时间占一帧内总时间的 $1/n$ ，只要扫描频率大于一定值，利用人眼的视觉惰性，人们就可以看到一幅完整的画面。

以一共阴极 4×4 LED 显示阵列为例(如图2.14所示)，采用动态扫描显示方式，行扫描列控制，各行轮流选通。当行上有一负脉冲选通信号时，列端四位数据中为“1”的发光二极管导通点亮。图2.14 中所示 16 个 LED 若要显示一方框，则

- (a) 行 1~4 上加循环选通脉冲；
- (b) 对应行选通时，在列上(四位数据端 A, B, C, D) 分别送数据，如表2.6所示。

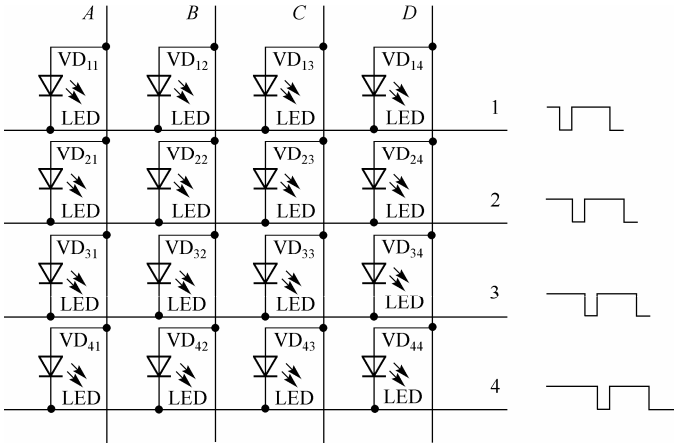


图 2.14 4×4 共阴极 LED 阵列

表 2.6 动态显示方框列输出数据

行\选通	A	B	C	D
1	1	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	1

当刷新频率足够高时，由于人眼的视觉暂留特性，可观察到稳定的方框。

静态显示的优点在于编程简单且硬件保证无闪烁；缺点是硬件利用率低，造成硬件成本较高。每一个像素需要一套驱动电路，如果显示屏有 N 行、 M 列，则需要 $N\times M$ 套驱动电路。例如一个 16×16 的 LED 显示屏，如用 8 位锁存器，需 32 个，此外还有 32 个锁存器口地址所需的译码电路及 LED 驱动电路。显然，这种硬件开销是不能接受的。

动态扫描显示方式可以避免以上不足，但是其缺点是容易造成显示亮度低、屏幕闪烁等问题。这些问题通过采用一定的技术措施是可以解决的，只要帧之间有较高的刷新速率，或者说只要同一行的两次扫描时间间隔不超过人眼的视觉暂留时间，即可保证显示画面稳定无闪烁。

② 输出数据方式论证

显示数据通常以字节的形式顺序存放在控制系统的存储器中。在行扫描列控制显示时，把显示数据从存储器中取出并传送到选通行对应的列驱动器上，这就存在列数据传输方式的问题。

a) 并行输出数据方式

数据并行传输的速度比较快,每次可同时传送数据 8 bit,占用 I/O 口线(数据线)8 根。例如对于一个 16×16 的显示屏,控制电路采用分时的方法,并行送入第一行 16 bit 的数据,然后选通第一行的 LED,延时一段时间,使该行 LED 对应于该行的数据显示时间;然后并行送入第二行的数据,选通第二行,再延时一段时间,依次循环直到所有行都被扫描。

b) 串行输出数据方式

数据串行传输的速度比较慢,但它可以大大简化传输线路。采用串行传输的方法,控制电路可以只用一根数据线,在串行时钟信号的控制下,将列数据一位一位地传往列驱动器,与此同时,列驱动器中每一列都把当前数据传向下一列,并从前一列接收新数据,一直到整行的各列数据全部传输到位后,才能并行地进行显示。

比较而言,并行输出数据方式和串行输出数据方式在显示效果上并无大的差异,不同之处在于:并行输出数据方式的译码电路较复杂,相对硬件开销大一些;串行输出数据方式电路构成简单,译码电路简捷,控制方便,调试容易,且成本较低。基于成本和设计的便利性等因素的考虑,采用串行输出数据方式应该是一种较好的选择。当然,串行输出数据方式也有不足,在后面的驱动模块的设计中有详细的说明。

(2) LED 显示屏驱动模块分析

LED 显示屏驱动电路的设计须与所用控制系统相配合。为保证在全屏刷新时有足够的亮度,驱动模块的大小通常设计为 16×16, 16×32, 32×32 的独立模块组,全屏由若干功能完全一致的独立单元模块按要求尺寸拼装而成。

① 驱动模块工作原理分析

由方案论证可知,当采用动态扫描显示方式时,驱动模块的功能可通过以下 4 个方面实现。

- a) 给出各行轮流导通的脉冲信号。
- b) 显示亮一行后更换输出显示数据。
- c) 大电流驱动电路,以满足实际工程应用。
- d) 控制信号和数据信号的传递,输出至下一驱动模块。

② 驱动模块中的功率驱动电路

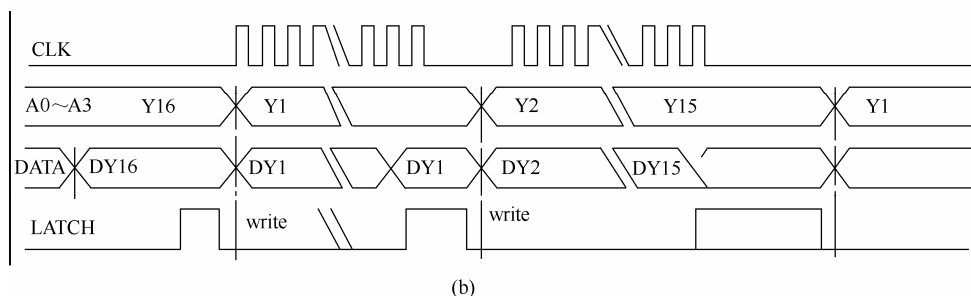
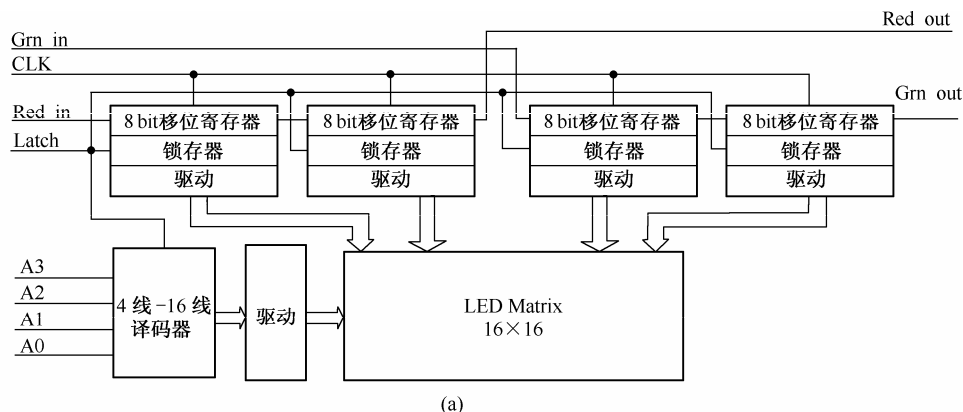
驱动电路作为驱动模块中的基本电路,不论数据传输及信号选通采用何种方式,其基本电路都是大同小异的。该部分包括输出数据的驱动和行选驱动(即行、列驱动),其设计要求应满足给定的 LED 阵列的显示亮度。设计时应以驱动模块为单元(例如 16×16),分别算出行选驱动和数据驱动的脉冲电流,据此设计选择电路及器件,完成功率驱动电路的设计(给定亮度应表示为在日光照射下,白天也可清楚观看显示屏上的信息内容)。

③ 驱动模块的设计

采用动态扫描法和串行数据传输的方案时,对于串行传输来说,数据要经过并行到串行和串行到并行两次变换,因此列数据的准备时间可能相当长,在行扫描周期确定的情况下,留给行显示的时间就少一些,以至于影响到 LED 的亮度。解决串行传输中列数据准备时间较长和列数据显示时间较短的矛盾,可以采用重叠处理的方法,即在显示本行各列数据的同时,准备下一行的列数据,这就需要列数据的显示具有锁存功能。

经过上述分析,可看出列驱动器电路应具备以下功能:对于列准备数据来说,它应能实现串入并出的移位功能;对于列显示数据来说,应具有并行锁存的功能。这样本行已准备好的数据输入并行锁存器进行显示时,串并移位寄存器就可以同时准备下一行的数据,而不会影响本行的显示。

考虑到典型性,以三色 LED 为例,给出动态扫描法串行数据输出的驱动模块设计及时序图(如图 2.15 所示),方案采用移位寄存器完成实时数据流的控制。



CLK—数据移位脉冲(上升沿)；A0~A3—行选通地址信号；Red, Grn—红、绿数据信号(高有效)；Latch—数据锁存信号

图 2.15 三色 LED 显示屏驱动模块电路框图及时序图

分析图2.15可知，在每一个 CLK 的上升沿，数据右移一位，直至所有数据已移完。Latch 信号变成高电平将此时各移位寄存器输出端信号锁入锁存器中，同时 Latch 信号控制地址译码器，使所有行都不被选中以避免数据锁存时在屏幕上出现混乱。此后，锁存信号变低，锁存器中数据输出，同时某行被选中，从而显示数据。与此同时，下一行的数据在移位寄存器中出现，当前行显示完毕之后，又将新的数据装入锁存器中，新的周期开始，点亮行的选通信号由 4 根地址线译码产生。采用此显示方案，电路简单、控制方便，易于调试，且成本较低，不足之处在于：

- (1) 接口需提供比较复杂的时序，使接口成本较高；
- (2) 数据通过移位寄存器传输，任何一片移位寄存器损坏将影响其后的全部显示。

需要指出的是，图2.15给出的是三色 LED 显示屏驱动模块的框图。如若设计全彩 LED 显示屏，其原理是一样的，只需加入蓝色数据驱动模块电路即可，其电路结构形式和红色、绿色数据驱动电路相同。单色 LED 显示屏驱动模块只需减去一路数据驱动电路。这一设计的最大优势在于，可以由若干块单元模块拼装成可能的任意大小的电子显示屏，并保证整体亮度的一致。

2.4.3 系统的设计思路与实现方式

1. 系统硬件部分设计

前面已讨论了三色 LED 显示屏的驱动模块的设计，单色 LED 显示屏少了一路数据信号通道，显然要简单一些。以 16×16 单色 LED 显示屏为例，根据对系统的设计分析可得到系统的硬件设计框图如图2.16所示。因为采用模块化设计，所以对于 16×128 的条屏设计，只需要在这个基础上对驱动模块进

行拼装扩展, 比较容易实现。

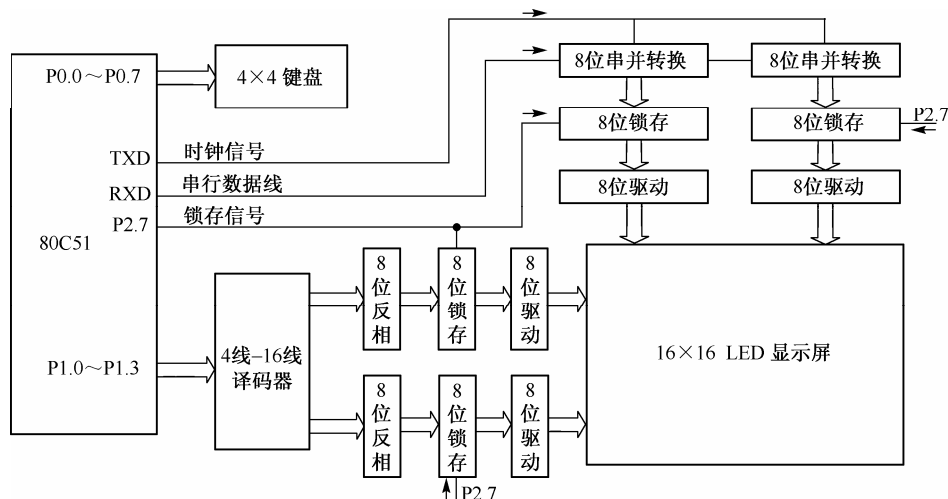


图 2.16 16×16 单色 LED 显示屏系统的硬件设计图

2. 系统软件部分设计

软件的设计是该设计的关键部分之一, 对文字的输出控制比较复杂, 其中要采用多重循环, 同时还要考虑显示屏的刷新速率、代码执行效率等问题, 故软件算法的合理设计极为重要。这些都是该系统设计需着重解决的问题。

(1) 单色显示程序算法分析

① 静态画面显示

以 16×16 单色 LED 显示屏为例来分析, 显示前, 需要对待显示的文字或画面进行编码和预存储。一个 16×16 像素的文字显示, 每行的 8 个像素为 1 Byte, 每行 2 Byte, 共需要 32 Byte 的字模数据。

静态画面显示是通过反复帧扫描实现的, 而每一帧画面的显示是通过顺序点亮 0~15 行来实现的。为了保证画面稳定无闪烁, 行与行之间的点灯时间间隔不能超过人眼的视觉暂留时间 0.1 s。刷新速度是由每一帧的显示时间决定的, 调整这个参数可以改变相应的指标。16×16 显示屏需要 32 个点灯数据, 故静态显示的算法如下:

(a) 取第一个字的头两个数据。

(b) P1.0, P1.1, P1.2, P1.3 口输出第 0 行地址。

(c) 由 RXD 串行发送数据, 然后由 TXD 产生时钟信号, 依次使 16 位数据移位到寄存器中。

(d) 锁存信号有效, 16 位数据被锁存并输出, 与此同时, 译码器译码使相应行选通, 之后适当延时, 这样就点亮一行; 然后更换显示数据, 行地址增量依次显示下一行。

(e) 判断 P1.0, P1.1, P1.2, P1.3 口输出的行地址是否为第 15 行地址, 若是, 跳至 (b), 否则跳至 (c)。

② 动态画面显示

动态画面显示是通过连续显示不同的静态画面实现的, 静态画面显示是其基础。动态画面的移动速度与静态画面的转换速度是一一对应的, 对一幅画面(一帧)的扫描次数越多, 则转换速度越慢, 即移动速度越慢。所以, 可以通过改变扫描的次数来控制移动速度。

动态平行左移或右移是显示子程序的主要部分, 现以左移为例来进行算法的分析。所谓左移, 即由 M 个字节一起左移 N 位 ($0 \leq N \leq 7$)。左移可以通过将上一帧静态画面数据左移得到, 即由移位指令

(RLC A)加入小循环 M 字节及大循环 N 次来实现,但这种方法在 N 为 0 和 N 为 7 时,时间差别较大,容易造成显示速度的不均匀。

显示子程序可利用乘法指令 $MUL\ AB$ 及循环 M 次来实现左移。因为左移 N 位即为乘以 2 的 N 次方,这样 N 为 0 和 N 为 7 时,时间几乎一样,可以避免显示速度不均匀的问题。

考虑 $MUL\ AB$ 前后 ACC 及 BCC 的内容如图 2.17 所示,则 M 个字符一起左移 N ($0 \leq N \leq 7$) 位时 (以 $M = 2$ 为例),前后的结果如图 2.18 所示。

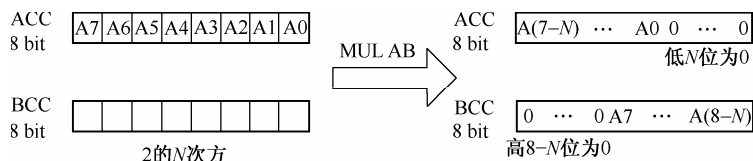


图 2.17 乘法指令 $MUL\ AB$ 示意图

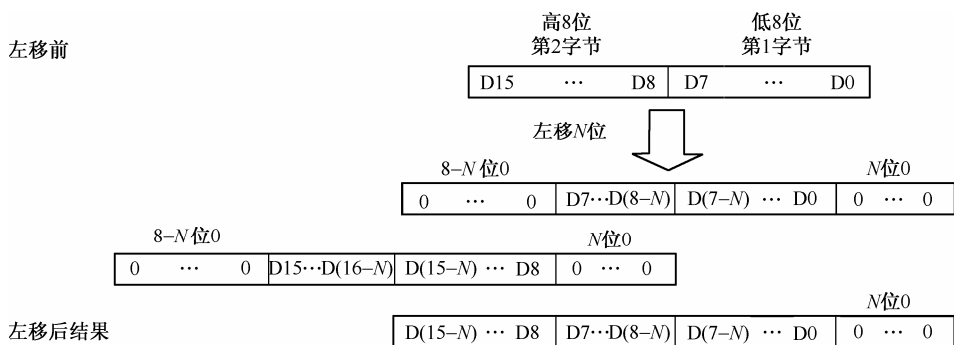


图 2.18 左移算法示意图

由图 2.18 可知,要获得 2 字节数据在左移 N ($0 \leq N \leq 7$) 位后的结果,只要第 2 字节左移 N 位后所得的数据低 8 位加上第 1 字节左移 N 位后的高 8 位即可。故要获得第 M 字节在左移 N ($0 \leq N \leq 7$) 位后的结果,只要第 M 字节左移 N 位后的数据低 8 位加上第 $M-1$ 字节左移 N 位后的高 8 位即可。

以广告语:“武汉大学,欢迎您!”为例,对于一屏显示一个字的 16×16 单色 LED 显示屏来说,平行左移显示的算法如下:

(a) 先取“武”字第一行的两个数据存 30H, 31H 和“汉”字的第一行的第一个数据存 32H。显示 30H, 31H 的数据。

(b) 将 30H 和 7FH 相与得到低 7 位,再将 31H 和 80H 相与得到高 1 位,通过乘法指令和加法指令,将其拼凑得到一个 8 位数据,同理取 31H 的低 7 位和 32H 的高 1 位拼凑组成一个 8 位数据。显示 16 位数据。

(c) 延时 1 ms, 1~15 行重复步骤(b), 显示后“武”字整体左移一位。

(d) 再将 30H 和 3FH 相与得到低 6 位,再将 31H 和 0C0H 相与得到高 2 位,通过乘法指令和加法指令,将其拼凑得到一个 8 位数据,同理取 31H 的低 6 位和 32H 的高 2 位拼凑组成一个 8 位数据。显示 16 位数据。

(e) 延时 1 ms, 1~15 行重复步骤(d), 显示后“武”字整体左移两位。

(f) 根据此算法,可以将字体依次左移。

取相反的方式,就可以得到平行右移显示的算法。

M 字节在左移 N ($0 \leq N \leq 7$) 位后的结果的参考程序片段如下:


```

SHIFT:  ACALL READCHAR      ;读入第 M 字节到 ACC
        MOV B, BITOFF2      ;BITOFF2 即为 2 的 N 次方运算后的结果
        MUL AB               ;
        MOV SAVELSB, A      ;第 M 字节的左移 N 位后的低 8 位
        MOV R6, #NUM        ;NUM = M
WORK3:  ACALL READCHAR      ;读入第 K 字节到 ACC
        MOV B, BITOFF2      ;BITOFF2 即为 2 的 N 次方计算后的结果
        MUL AB               ;
        XCH A, B             ;
        ADD A, SAVELSB      ;第 K 字节左移 N 位后的高 8 位加上第 K+1 字节的左移 N 位
                                后的低 8 位, 即为第 K 字节左移 N 位后的第 K 字节(ACC)
        MOV SAVELSB, B      ;第 K 字节左移 N 位后的低 8 位送入 BCC 中
        ACALL SEND
WORK4:  DJNZ R6, WORK3

```

● 动态垂直移动子程序

对于一屏显示一个字的 16×16 单色 LED 显示屏来说, 垂直上移显示的算法如下:

(a) 第一帧画面正常显示, 接下来每帧画面只需将要显示的数据地址依次加 2, 直至取完 16 行的数据。

(b) 一个字移完后, 取下一个字的数据, 仍然按照步骤(a)直至所有字模数据取完。

取相反的方式, 即可得垂直下移的显示方式。

● 字体显示移动速度的控制

显然, 每一帧画面扫描次数取不同数值时, 字体显示移动速度也就不同。把扫描次数分为 2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57, 62, 代表 13 种显示移动速度, 其中正常速度 1 级(扫描次数 32), 加快和减慢各分为 6 级。这样在不影响发光亮度的前提下实现了速度快慢各分 6 级的设计要求。

(2) 三色 LED 显示程序算法分析

相比而言, 三色 LED 显示程序的算法较为复杂一些, 若同样以单片机为控制核心实现三色 LED 显示屏的设计, 难度主要体现在算法上。

① 伪彩色数据输出

由三色 LED 显示屏的技术指标得知其像素结构为 LED 红(2)、LED 绿(2), 即为在同一个像素内集成了 2 只红色 LED 发光管和 2 只绿色 LED 发光管。由三基色原理可显示三种颜色: 红、绿、黄。显示屏的主要功能用于显示广告语句, 其信息主要是汉字点阵信息。为了充分发挥三色 LED 显示屏的颜色种类多的特点, 必须把二值逻辑(有/无)的点阵信息转化为多色的伪彩色数据, 如红底绿字、黑底绿字、黄底绿字、绿底黄字等, 共有 12 种组合。当显示为红底绿字时, 绿色通道的数据即为点阵数据(有该点时绿色 LED 点亮), 而红色通道的数据则刚好和点阵数据相反(有该点时红色 LED 熄灭, 没有该点时红色 LED 点亮, 以构成红色的背景)。当显示为黑底绿字时, 绿色通道的数据即为点阵数据, 而红色通道的数据则应始终为 0(即红色 LED 应始终熄灭)。当显示为黄底绿字时, 绿色通道的数据应为点阵数据, 而红色通道的数据则应和点阵数据相反, 为实现黄底绿字显示, 绿色通道应先送入点阵数据, 再根据点阵数据得到红色通道数据, 然后绿色通道再送全 1 数据(即绿色 LED 全部点亮)。综合上述几种情况可知, 红、绿通道输出的数据是全 0、全 1、点阵数据或点阵数据的反码。在实现中可对点阵数据进行处理, 用 8 位点阵数据先同一个数(0 或 255)相与, 再同一个数(0 或 255)相或, 最后再同一个数(0 或 255)相异或, 用以产生红绿通道所需要的数据。同 0 相与结果为全 0, 同 255 相与则数据不变; 同 255 相或则 8 位全为 1, 同 0 相或则数据不变; 而同 255 异或可生成反码, 同 0 异或则数

据不变。因此,通过选择合适的与、或及异或的逻辑操作,即可生成任意的字符(前景)和底色(背景)的颜色组合。前景和背景都各有红、黄、绿和黑 4 种颜色可选择,共计 16 种颜色,去掉 4 种前景和背景一样的情况,共计实现了 12 种颜色的组合。

② 动画刷新的实现方法

考虑讨论问题的方便,仍以 4×4 的 LED 阵列说明 LED 屏的逻辑结构,如图 2.19 所示,则此系统的每一个 LED 要使其点亮,必须同时对其行寄存器的对应位置 0,列寄存器的对应位置 1。此时只要给每个行、列寄存器一个掩码,每个送到列寄存器的数均和该掩码相与,则列寄存器掩码相应位为 0 的点的列寄存器的输出始终为低电平,LED 始终不会亮。同理,输出到行寄存器的数均与其相或,则行寄存器掩码相应位为 1 的点的行寄存器的输出始终为高电平,LED 始终不会亮。因此,此时显示器上可显示的区域一定是由列寄存器掩码相应位为 1 的点及行寄存器掩码相应位为 0 的点所组成的,即实现了局部显示。例如,在 4×4 的 LED 屏中取列寄存器的掩码为 0110,行寄存器的掩码为 1001,则此时 4×4 的 LED 屏中只能有正中间的 2×2 的 4 个 LED 被点亮。只要给行、列寄存器一个动态的掩码(每个送到行寄存器的数据均和其相或,每个送到列寄存器的数均和其相与),则此时显示屏上的可视区域就是一个动态变化的区域。所谓的 32 种动画效果的设计要求,转化为根据 32 种不同显示图形的变化来设计 32 种掩码变化序列,用以产生动画效果。

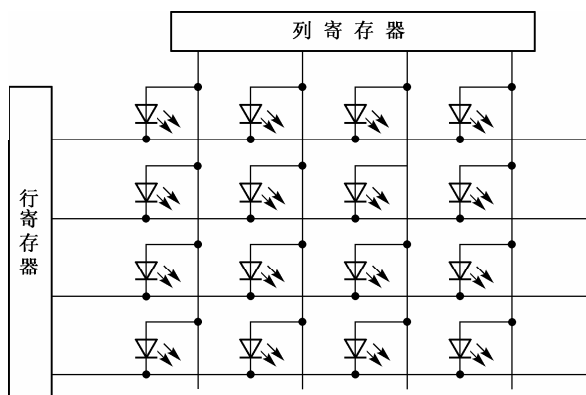


图 2.19 LED 显示屏逻辑结构图

2.4.4 基于PC平台的LED显示屏系统分析

1. 接口电路的设计难点分析

基于 PC 平台的 LED 显示屏系统,需要将所需的图像和数据同步显示在 LED 屏上而不能有所遗漏。如何实现实时处理和实时显示是此类系统设计的关键。

为实现实时显示,必须借助于计算机作为信息和控制平台,而计算机的各种显示模式帧频变化在 60~120 Hz 之间,电视场频为 50 Hz,相应的点频都在 25 MHz 之上,通常的图形 12 模式下,时钟为 25.17 MHz。显然,数据不经过转换是不能用驱动模块直接显示的,而驱动显示单元的最高工作频率必须在 10 MHz 以下,考虑到发光二极管的响应时间,矛盾是显而易见的。为了不遗漏任何信息细节,必须扩大信道容量,使低频系统能实时配合高频系统工作。

(1) 并行多路方式存储和输出

采用并行多路工作方式,屏幕数据输出时,根据屏幕大小将其分区,在该帧时间内将数据分别存入不同的存储单元中,每一单元里的数据对应屏幕上某一区的显示数据,在第二帧期间,各存储单元数据的数据被同时读出,送往驱动模块显示。由于传输容量扩大 N 倍,所以传送速度至少可降低至原

来的 $1/N$ 。

(2) 两组 RAM 交替读写

为保证显示数据与 LED 屏驱动电路传输的数据同步,需要采用两组 RAM 来存储数据,两组 RAM 交替处于读写操作,设某一帧时 RAM1 处于写状态,将要显示信息的数据存入,则 RAM2 处于读状态,将前一帧写入的数据按一定的时序读出,用于驱动 LED 模块显示。一帧过后,两套单元的位置互换,即通过两组 RAM 的“乒乓”读写既保证了实时又降低了进入驱动模块的信号频率。

2. 三色LED屏的设计分析

无论单色屏、三色屏、全彩屏或灰度屏,若要求实时显示,上述工作原理是一样的,而屏的类型的区别主要在驱动单元上。当然,单色屏的设计相对要简单一些,讨论三色屏的设计更有典型意义。

(1) 输入信号

为能实时显示 CRT 屏幕上的内容,必须获得视频的即时信号。计算机 TVGA 显卡提供了标称为 Feature connector 的端口,输出有 8 位数据信号、行场同步、复合消隐及工作时钟信号(均为 TTL 电平)。TVGA 卡支持 256 色模式显示,因此有 8 位数据信号输出。由于三色屏只需显示 4 种颜色(红、绿、黄、黑),因此取二位即可,工作时钟 CLK 决定了扫描频率的快慢,也就是说,数据信号的宽度就是 CLK 的周期。

帧同步信号是实时工作中的重要控制信号,在不同显示模式下帧频是变化的,可能是正脉冲,也可能是负脉冲。复合消隐(BLANK)中包含有行频,而且行同步信号出现在复合消隐期中,所以同步信号结束时屏幕扫描线尚在回扫期间,无数据输出,若用行同步信号控制地址产生会引起困难,而消隐信号克服了这个缺陷,因此可选用复合消隐信号。帧同步、行同步及复合消隐信号时序如图 2.20 所示。

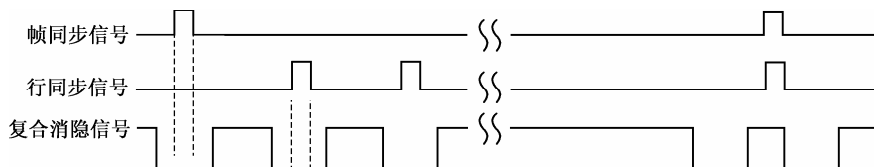


图 2.20 帧同步信号、行同步信号及复合消隐信号时序图

(2) 工作原理

由 TVGA 显卡送来的数据信号经由移位寄存器后,在不同的周期,数据通过数据缓冲区分别存储在 RAMA 和 RAM B 中。

地址发生器单元的工作状态及各缓冲器的开关状态均由帧同步信号控制。由于帧同步信号脉冲在不同显示模式下是变化的,因此对输入到不同单元电路的帧同步信号脉冲(控制信号)应做相应的变换及整形处理。复合消隐和帧信号一起控制监视器屏幕数据点的采集的起始位置,通过拨动开关和电路参数的调节,使起始点的位置可在显示屏的任意位置上。

CLK 信号在帧信号及复合消隐信号的控制下产生 RAM 读/写信号所需的全部地址信号,同时有两路输出:一路信号提供给片选信号产生单元,产生 RAM 工作时所需的选通信号;另一路产生驱动模块的控制信号。

数据信号从 RAM 中读出, RAMA 和 RAM B 的输出数据经缓冲器后在总线上汇合,然后通过移位寄存器转变成串行数据后送往驱动模块。

输出的控制信号包括移位时钟 SCLK、数据锁存信号 Latch 及行选通地址信号,这 4 根地址信号

线就是在 RAM 读数据过程中的高 4 位地址,这样保证了对应行的数据显示在 LED 屏幕的正确位置上。系统工作原理图如图 2.21 所示。

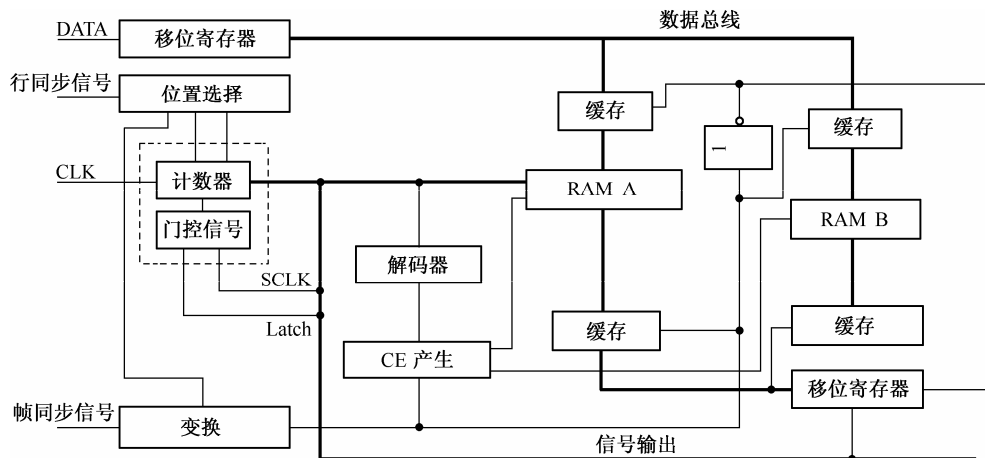


图 2.21 系统工作原理图

2.5 本章小结

本章以通用 MCS-51/52 单片机最小系统、简易电子琴的设计和 LED 显示屏的设计三个训练题目为例,详细讨论了相对简单的电子系统的设计方法和设计要点。

首先以 MCS-51/52 单片机最小系统为例,对通用单片机最小系统的组成部分和一些编程示例做了详细介绍,旨在引导读者较深入地理解单片机最小系统的构成和设计方法。通用的单片机最小系统的设计对于后续的单片机应用系统开发是非常有益的。芯片的选择和设计方式是多种多样的,在设计时应尽可能选择典型、标准化的电路,系统扩展与外围的配置水平应充分满足应用系统的功能要求,并留有适当的余地,以便进行二次开发。此外,还应考虑系统的可靠性、抗干扰性等方面的要求。

基于单片机系统的简易电子琴是一个典型的电子系统设计,在训练中应注意如何通过物理模型抽象出系统设计模型。音调、音强和音长是乐音的三种基本特性,它们分别和声波的频率、振幅和持续时间等物理量相对应。因此,系统设计的关键所在是如何通过单片机来控制馈送至扬声器的驱动脉冲的频率、振幅和持续时间,从而产生乐音,这也是该系统设计的切入点。该训练旨在通过该系统的设计与实现,使读者建立系统设计的基本概念和分析思路,掌握基本的设计方法。

LED 显示屏系统设计和实现具有较高的使用价值,基于不同的功能及技术指标,系统设计的复杂程度有很大差别。训练先通过一个基于单片机的单色 16×16 LED 显示屏的设计,使读者掌握基本的设计方法。由于采用了模块化设计,对于 16×128 的条屏或者更大的屏幕,设计均可以继承使用,实现有效的拼接和扩展,这也是自底向上的设计方法的一种很好体验。此外,该训练所做的设计分析是在分立器件的基础上实现的,也可以采用可编程逻辑器件来实现,尤其是对于大型 LED 显示屏的显示系统,采用可编程逻辑器件完成设计实现可能会更方便有效。但要说明的是,没有对基于常规器件的电子系统的设计思想的形成和设计方法的掌握,把一个毫无实践根据的想法盲目地移植到可编程逻辑器件中将是无意义的。所以,在复杂电子系统设计训练的初期阶段,基于常规器件的系统设计方案的提出和设计理论的建立是必不可少的。

第3章 时间(频率)的数字化测量

3.1 引言

时间(频率)是电子测量中的重要基本参量,同时,也是许多非电量物理量测量的基础,因为许多物理量都可变换为时间(频率)这一基本电参量。正因为如此,时间被确定为国际单位制中7个基本物理量之一。时间和频率的测量在航天、航海、工业、交通、通信及国民经济各个领域有着十分广泛的应用。时频测量的重要性是不言而喻的,在工业控制、信息传输和处理、现代数字技术和计算机应用等领域都离不开时频处理和时频测量。

时频测量的基本方法为比较法,在测量中时间(频率)基准的建立和获得至关重要。由于频率和周期互为倒数,它们共用一个基准,所以知道了一个信号的频率,也就知道了它的周期。从广义的角度去理解,周期也可以看成是一种时间间隔,时间的测量可以转化为频率的测量。由于技术的进步,时间(频率)的测量大多采用了数字化测量的方法,因此,时间(频率)的数字化测量技术是本章讨论的重点。

3.2节介绍了频率测量的基本原理并对测量误差进行了分析,为后续的频率计设计奠定了理论基础。3.3节给出了一个数字频率计设计任务,并详细讨论了基于等精度频率测量原理的实现方案,随后对系统设计中的抗干扰措施进行了介绍。3.4节讨论了相位测量系统的原理和实现方案。3.5节对本章进行了总结。

本章通过频率和相位测量原理及实现方法的介绍,一方面希望读者能够通过这些典型的设计实例掌握如何将理论分析映射到实际的电路系统设计中,如何根据理论分析改进系统的设计方案,另一方面也希望读者学会运用已有的知识分析系统的极限性能,从而得到有关系统的性能指标。时间(频率)测量是一个基本物理量的测量,可以应用在很多其他的测量系统中,比如温度、电压等信号的测量都可以转换为时间(频率)测量,因此读者可以将本章的训练结果作为一个独立的模块应用到以后的系统设计中。

3.2 频率测量原理及误差分析

实现时间参数的数字化测量仪器是电子计数器。电子计数器测量时间(频率)的实质是通过计数器记录待测信号周期变化的次数,然后通过频率的定义计算出待测信号的频率。已知频率可表达为 $f = N/T$ 。由测量原理及频率的数学表达式不难看出,计数法测量时间(频率)必须具备以下三个条件:

- (1) 测量是一个比较的过程,被测信号要和基准做比较,必须获得一个标准的单位时间。
- (2) 为实现在单位时间内对被测信号振动次数的记录,必须有一个控制电路。
- (3) 被测信号采样后的量化由电子计数器完成,以获得量化值 N 。

图3.1所示为电子计数法测频的原理框图及各点波形。

对应于电子计数法测量时间(频率)的必备条件,电子计数法测频、测周时的电路主要由下列三部分组成。

- (1) 时间基准 T 产生电路

时间基准产生电路的作用是用来产生计数器所使用的标准频率或时间间隔。它一般由石英晶体振荡器、分频整形电路与门控(双稳)电路组成。对时间或频率的基准有两点要求:一是标准性。时基是量化的标准,若其值不准,将直接影响转换精度。所以作为时间或频率的基准源应当是一个具有高稳定度($10^{-6} \sim 10^{-10}$ 量级)的信号源,因为没有稳定性就谈不上标准性。通常采用高稳定石英晶体振荡器充当标准信号源。二是多值性。为了便于各种输入值的量化比较,要求电子计数器中备有多种量化单位值。晶体振荡器只能产生单一的固定频率信号(如图3.1所示的晶振频率 f_c),采用多级分频或倍频的方法,可获得多种标准的量化单位值。常用的标准单位时间(时标信号) T_0 有1 ms, 0.1 ms, 10 μ s, 1 μ s, 0.1 μ s, 10 ns等几种;常用的标准单位频率(频标信号) f_0 有1 kHz, 100 Hz, 10 Hz, 1 Hz, 0.1 Hz等几种。

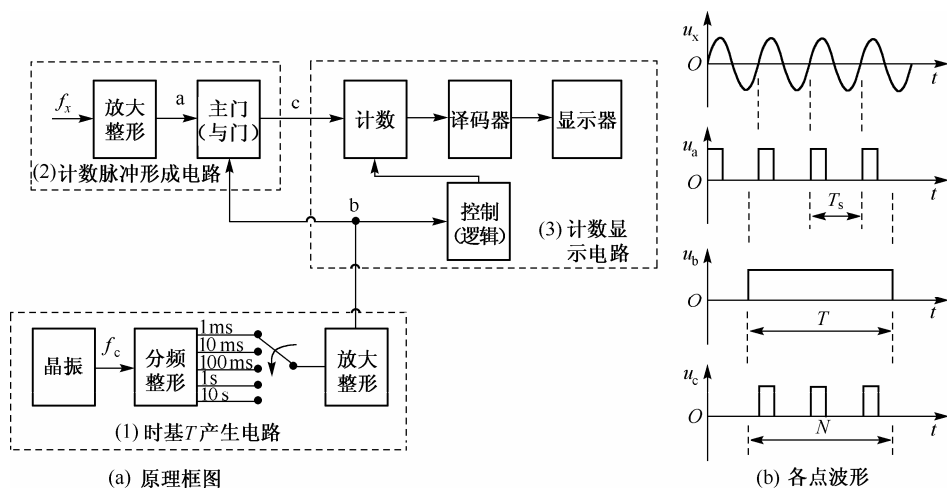


图 3.1 电子计数法测频的原理框图及各点波形

(2) 计数脉冲形成电路

计数脉冲形成电路的作用是将被测的周期信号转换为可计数的窄脉冲。它由放大整形电路和主门(与门)电路组成,输入的被测周期信号(频率为 f_x ,周期为 T_x)经放大整形得到周期为 T_x 的窄脉冲,送至主门的一个输入端。主门的另一个输入端为控制端口,控制端输入的是时间基准产生电路产生的闸门脉冲。在闸门脉冲开启主门期间,周期为 T_x 的窄脉冲才能经过主门,在主门的输出端产生输出。在闸门脉冲关闭主门期间,周期为 T_x 的窄脉冲不能在主门的输出端产生输出。在闸门脉冲控制下的主门输出脉冲,被送入计数器计数,因此将主门输出的脉冲称为计数脉冲,相应的这部分电路称为计数脉冲形成电路。

(3) 计数显示电路

计数显示电路的作用是对主门输出的脉冲进行计数,其结果用数字显示出来。它一般由计数电路、逻辑控制电路、译码器和显示器组成。在逻辑控制电路的控制下,计数器对主门输出的计数脉冲实施二进制计数,其输出经译码器转换为十进制计数值,输出到数码管或显示器件以显示计数结果。因为时间基准 T 是10的整次幂倍秒,所以显示出的十进制数就是被测信号的频率或时间。测量时间间隔时,时间基准产生电路给出标准单位时间(时标信号) T_0 ,其单位可能是s, ms, μ s, ns。测量频率时时间基准产生电路给出标准单位频率(频标信号) f_0 ,其单位可能是Hz, kHz, MHz。逻辑控制电路用来控制计数器的工作程序(准备—计数—显示—复零—下一次测量)。

由此可看出,电子计数器测量时间(频率)的原理实质上是以比较法为基础的。时间测量和频率测量共用一个基准,测频时将被测信号频率 f_x 和已知的标准单位时间(时标信号) T_0 相比,测量时间时将测量信号周期 T_x 和已知的标准单位频率(频标信号) f_0 相比,再将相比的结果以数字的形式显示 出

来。

3.2.1 频率或脉冲速率的数字测量方法

频率是在时间轴上无限延伸的,因此对频率的测量需要确定一个取样时间。对于频率 f_x 或脉冲速率 n_x 的待测模拟信号,首先用比较器将它们转换为矩形脉冲序列。为了使这些矩形脉冲在预先给定的测量时间段 T 内进入计数器,如前所述,在计数器电路前设置一个门电路(主门),控制电路在规定的时间内打开主门,允许信号进入计数器作累加计数。在已知的标准时间内累计未知的被测输入信号的脉冲个数,这就实现了频率或脉冲速率的测量。在测量频率或脉冲速率时,测量时间 T 是已知的常数,计数器读数是频率 f_x 或脉冲速率 n_x 的度量:

$$f_x = N/T, \quad n_x = N/T \quad (3.1)$$

3.2.2 时间参数的数字测量方法

在电子技术应用中,时间的测量包括周期、脉冲上升(下降)时间、时间间隔及长时间测量等参数。时间测量和频率测量非常类似,但也有它特殊的地方。由于周期和频率互为倒数,因此在测频的原理电路中对换一下被测信号和时标信号的输入通道就能完成周期的测量。此时,闸门时间 T_x 是由被测信号产生的,计数脉冲是由晶振信号经整形、窄脉冲形成而产生的。计数显示电路与测频时相同。

在未知的待测时间间隔内累计已知的标准时间脉冲个数,就实现了周期或时间间隔的测量。测量时用一个频率为 f_0 的脉冲序列在时间间隔 T_x 内进行计数,得到计数值

$$N_x = f_0 \cdot T_x \quad (3.2)$$

在测量时间时,频率 f_0 是已知的常数,计数器读数是所求时间 T_x 的度量。以上对频率、时间的测量方法称为门控计数法。

3.2.3 电子计数器的测量误差

数字频率计的设计是一个典型的电子测量系统的设计。在测量中,误差分析计算是必不可少的。理论上讲,对于任何物理量的测量,不管采用什么样的测量方法,只要进行测量就有误差存在。误差分析的目的就是要找出引起测量误差的原因,有针对性地采取有效措施,以减小测量误差,提高测量的准确度,从而满足系统设计的指标要求。由于频率测量有计数器直接测频和间接测周两种方法,所以对频率测量和周期测量的误差分析分别给出讨论。

1. 频率测量误差分析

(1) 频率测量的误差表达式

计数器直接测频的误差主要由两项组成,即 ± 1 量化误差和标准频率误差。一般地,总误差可采用分项误差绝对值合成,即

$$\frac{\Delta f_x}{f_x} = \pm \left(\frac{1}{f_x T} + \left| \frac{\Delta f_c}{f_c} \right| \right) \quad (3.3)$$

式中,等号右边括号内第一项为 ± 1 量化误差,这是数字化仪器所特有的误差;括号内第二项为标准频率误差。需要说明的是,实际输出的频标信号为 f_0 , f_0 为晶振信号 f_c 分频后所得到的标准量化单位值。分频系数不同, f_0 的取值也就不同(时间基准的多值性)。考虑到分频误差较小及讨论问题的方便,对于标准频率误差的分析均以晶振频率 f_c 的相对不确定度来讨论。

(2) 量化误差

用电子计数器测量频率,实质上是一个量化过程,量化的最小单位是数码的一个字或者一个脉冲。在测频时,由于主门的开启时刻与计数脉冲之间的时间关系是不相关的,它们在时间轴上的相对位置是随机的,因此被测信号与门控信号之间没有同步锁定的关系,也就是说,闸门开启时刻和第一个被测计数脉冲到来的时刻之间的关系是随机的。如图 3.2 所示,某个门控时间 T 等于被测信号的 5.7 个周期,由于测量结果的表达是量化的,其值用整数来计算。对于 0.7 的尾数,要么舍弃,要么凑整为 1 计入测量结果之中。因而,对于同一个被测量结果会有差别。

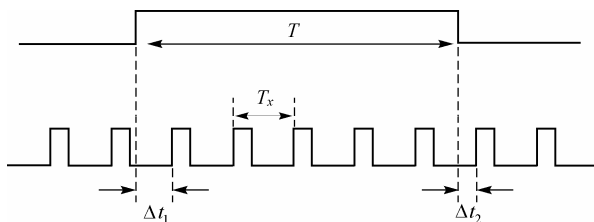


图 3.2 脉冲误差示意图

在图 3.2 中, T 为计数器的主门开启时间, T_x 为被测信号周期, Δt_1 为主门开启时刻至第一个计数脉冲前沿的时间(假设计数脉冲前沿使计数器翻转计数), Δt_2 为闸门关闭时刻至下一个计数脉冲前沿的时间。设计数值为 N , 由图 3.2 可得

$$T = NT_x + \Delta t_1 - \Delta t_2 = \left[N + \frac{\Delta t_1 - \Delta t_2}{T_x} \right] T_x = (N + \Delta N) T_x \quad (3.4)$$

式中,

$$\Delta N = \frac{\Delta t_1 - \Delta t_2}{T_x}$$

由于 Δt_1 和 Δt_2 在 $0 \sim T_x$ 之间任意取值, 则可能有下列情况:

- 当 $\Delta t_1 = \Delta t_2$ 时, $\Delta N = 0$ 。
- 当 $\Delta t_1 = 0$, $\Delta t_2 = T_x$ 时, $\Delta N = -1$ 。
- 当 $\Delta t_1 = T_x$, $\Delta t_2 = 0$ 时, $\Delta N = +1$ 。

由此可知, 最大计数误差为 ± 1 个数, $\Delta N = \pm 1$ 。所以, 计数器计数的最大相对误差为

$$\frac{\Delta N}{N} = \pm \frac{1}{N} = \pm \frac{1}{f_x T} \quad (3.5)$$

式中, f_x 为被测频率; T 为闸门时间。由式 (3.5) 可以看出, 脉冲计数相对误差和被测信号频率与闸门时间的乘积成反比。也就是说, 被测信号频率越高, 闸门时间越宽, 相对误差越小。

(3) 标准频率误差

标准频率误差又称为闸门时间误差, 在频率测量中, 闸门时间 T 是由晶振信号分频得到的。晶振输出频率的不稳定引起闸门时间的不稳定, 造成测频误差。设晶振频率为 f_c (周期为 T_c), 分频系数为 k , 则有

$$T = kT_c = k / f_c \quad (3.6)$$

由误差合成原理可知

$$\frac{\Delta T}{T} = - \frac{\Delta f_c}{f_c} \quad (3.7)$$

式 (3.7) 表明, 闸门时间相对误差在数值上等于晶振频率的相对误差。因此, 在实际应用中, 要求标准

频率的相对不确定度比测量要求的相对不确定度高一个数量级。

(4) 减小测频误差方法的分析

由计数器直接测频的误差表达式(3.3)中测频误差与量化误差和标准频率误差的关系,可画出如图3.3所示的误差曲线。从图中可以看出,当 f_x 一定时,增加闸门时间 T 可以提高测频分辨率和准确度。当闸门时间一定时,输入信号频率 f_x 越高,则测量准确度越高。但是,随着 ± 1 量化误差影响的减小到 $|\Delta f_c/f_c|$ 以下,标准频率误差 $|\Delta f_c/f_c|$ 对测量结果产生影响,并以 $|\Delta f_c/f_c|$ (图中以 5×10^{-9} 为例)为极限,即测量精度不可能优于 5×10^{-9} 。

(5) 信号触发误差

在计数器直接测频时,除了量化误差、标准频率误差还应考虑信号的触发误差对测量不确定度的影响。所谓信号触发误差,是指采用比较器将模拟信号转换为矩形脉冲序列的过程中,由于脉冲序列的上升或下降沿是计数的对象(计数脉冲前沿或后沿使计数器翻转计数),当模拟信号为调幅信号且受干扰或包含高次谐波时,比较器门限如果设置不当,就可能产生信号触发误差,导致计数错误的产生,如图3.4所示。需要注意的是,图3.4(a)中比较器的比较电压 U_r 的设计值应选择正好能测量到基波的周期。显然,信号触发误差对系统测量不确定度的影响是不可忽略的。

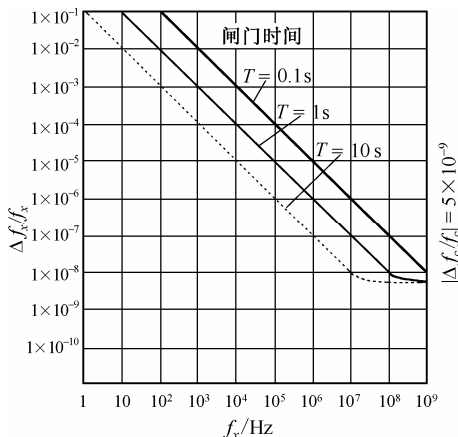


图 3.3 计数器测频时的误差曲线

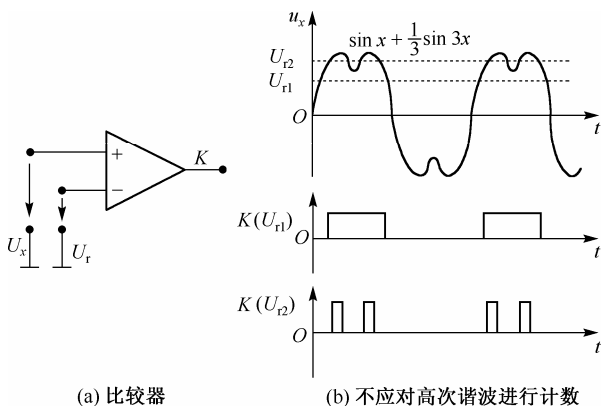


图 3.4 信号触发误差

2. 周期测量误差分析

(1) 误差表达式

由周期测量的数学表达式

$$T_x = NT_c = N/f_c \quad (3.8)$$

和误差合成原理可得周期测量的误差表达式为

$$\frac{\Delta T_x}{T_x} = \frac{\Delta N}{N} + \frac{\Delta T_c}{T_c} \quad (3.9)$$

同样,电子计数器测量周期的误差也由两项构成:第一项为计数器计数误差(量化误差),第二项为标准频率误差。对计数器计数误差的分析与计数器测频的分析相同,由于在极限情况下,量化误差 $\Delta N = \pm 1$,所以

$$\frac{\Delta N}{N} = \pm \frac{1}{N} = \pm \frac{T_c}{NT_c} = \pm \frac{T_c}{T_x} = \pm \frac{1}{f_c T_x} \quad (3.10)$$

由式(3.10)可以看出, T_x 越大(即被测频率越低), 时基频率 f_0 越高(即晶振频率 f_c 越大, 分频系数越小), 则 ± 1 量化误差对测量误差的影响越小。

对于第二项标准频率误差, 由于晶振频率误差的符号可能为正, 也可能为负, 考虑可能的最大误差, 在计算测量周期误差时取绝对值相加, 所以有

$$\frac{\Delta T_x}{T_x} = \pm \left(\frac{1}{N} + \left| \frac{\Delta f_c}{f_c} \right| \right) = \pm \left(\frac{T_c}{T_x} + \left| \frac{\Delta f_c}{f_c} \right| \right) \quad (3.11)$$

(2) 减小测量周期误差的方法

由式(3.11)可画出周期测量的误差曲线如图 3.5 所示。从图中可以看出, 计数器测量周期时, 其误差主要取决于量化误差, 被测周期越大(f_x 越小)时误差越小, 被测周期越小(f_x 越大)时误差越大。为了减小测量误差, 可以通过对更高频率的时间基准信号进行计数来减小量化误差的影响(增大计数基数), 即减小 T_c (增大 f_c)。另一种方法是把 T_x 扩大 k 倍, 形成的闸门时间宽度为 kT_x , 以它控制主门开启, 实施计数。计数器的计数结果为

$$N = kT_x / T_c \quad (3.12)$$

由于 $\Delta N = \pm 1$, 所以

$$\frac{\Delta N}{N} = \pm \frac{T_c}{kT_x} \quad (3.13)$$

将式(3.13)代入式(3.11), 可得

$$\frac{\Delta T_x}{T_x} = \pm \left(\frac{T_c}{kT_x} + \left| \frac{\Delta f_c}{f_c} \right| \right) = \pm \left(\frac{1}{kT_x f_c} + \left| \frac{\Delta f_c}{f_c} \right| \right) \quad (3.14)$$

式(3.14)表明量化误差降低了 k 倍。

当被测信号频率足够低时, 标准频率误差不能忽略不计, 它可以被看成是周期测量所能达到的最高精度。

(3) 形成闸门信号时的触发误差

式(3.14)所示的测量周期误差表达式是在被测波形为方波脉冲和无噪声干扰的条件下获得的。若被测波形为非矩形方波或在测量时存在噪声干扰, 则应考虑触发误差的影响。在测量周期时, 被测信号经放大整形后通过触发器转换为时间闸门的控制信号(简称门控信号), 因此其触发电平的波动及噪声等均会影响门控信号(T_x)的准确性, 造成所谓的触发误差, 如图 3.6 所示。

若被测正弦信号为理想的情况, 在过零时刻触发, 则开门时间为 T_x 。若存在噪声, 则有可能使触发时间提前 ΔT_1 , 也有可能使触发时间延迟 ΔT_2 (图 3.6 仅画出了触发时间提前的情况, 触发时间延迟的情况类

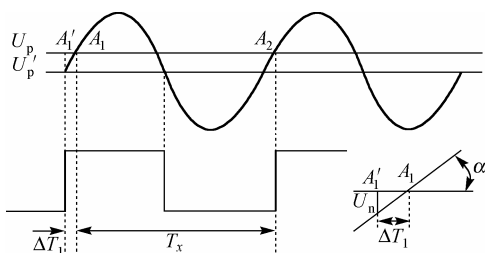


图 3.6 触发误差的产生和计算

同)。对上面情况进行粗略分析, 设正弦波过零点的斜率为 $\tan \alpha$, α 角如图 3.6 所示, 则得

$$\Delta T_1 = \frac{U_n}{\tan \alpha} \quad (3.15)$$

$$\Delta T_2 = \frac{U_n}{\tan \alpha} \quad (3.16)$$

式中, U_n 为被测信号上叠加的噪声“振幅值”。当被测信号为正弦波, 即 $u_x = U_m \sin(\omega_x t)$, 门控电路触发电平为 U_p , 则

$$\begin{aligned} \tan \alpha &= \left. \frac{du_x}{dt} \right|_{u_x=U_p, t=t_p} \\ &= 2\pi f_x U_m \cos \omega_x t_p \\ &= \frac{2\pi}{T_x} U_m \sqrt{1 - \sin^2 \omega_x t_p} \\ &= \frac{2\pi}{T_x} U_m \sqrt{1 - \left(\frac{U_p}{U_m} \right)^2} \end{aligned} \quad (3.17)$$

将式 (3.17) 代入式 (3.15) 和式 (3.16), 可得

$$\Delta T_1 = \Delta T_2 = \frac{U_n T_x}{2\pi U_m \sqrt{1 - \left(\frac{U_p}{U_m} \right)^2}} \quad (3.18)$$

因为一般门电路采用过零触发, 即 $U_p = 0$, 因此

$$\Delta T_1 = \Delta T_2 = \frac{T_x}{2\pi} \cdot \frac{U_n}{U_m} \quad (3.19)$$

在极限情况下, 开门的起点将提前 ΔT_1 , 关门的终点将延迟 ΔT_2 , 或者相反。根据随机误差的合成定律, 可得总的触发误差为

$$\begin{aligned} \Delta T_n &= \pm \sqrt{(\Delta T_1)^2 + (\Delta T_2)^2} \\ &= \sqrt{2} \frac{T_x}{2\pi} \cdot \frac{U_n}{U_m} = \frac{T_x U_n}{\sqrt{2} \pi U_m} \end{aligned} \quad (3.20)$$

如前类似分析, 若门控信号周期扩大 k 倍, 则由随机噪声引起的触发相对误差可降低为

$$\frac{\Delta T_n}{T_x} = \pm \frac{1}{k\sqrt{2}\pi} \cdot \frac{U_n}{U_m} \quad (3.21)$$

式 (3.21) 表明, 测量周期时的触发误差与信噪比成反比。例如, $U_m/U_n = 10$ 时, $\Delta T_n/T_x = \pm 2.3 \times 10^{-2}$; $U_m/U_n = 100$ 时, $\Delta T_n/T_x = \pm 2.3 \times 10^{-3}$ 。由此可以直观地看出, 信噪比越大, 其触发误差就越小。若对引起触发误差主要因素分别单独考虑, 由式 (3.15) ~ 式 (3.18) 稍做推理分析即可看出, 信号过零点斜率 ($\tan \alpha$) 值大, 则在相同噪声幅度 U_n 条件下, 引起的 ΔT_1 和 ΔT_2 小, 从而使触发误差小; 信号过零点斜率一定, 则噪声幅度大时, 引起的触发误差大。信号幅度 U_m 对触发误差的影响已隐含在信号过零点斜率因素当中。信号频率一定, 当信号幅度值大时, 其过零点的斜率也大, 由此推知,

信号幅度 U_m 大时引起的触发误差小。触发误差还应与触发器的触发灵敏度有关, 若触发器的触发灵敏度高, 一个小的噪声扰动就可使触发器翻转, 所以在其他条件相同的情况下, 触发器触发灵敏度高, 则引起的触发误差大。

分析至此, 若考虑噪声引起的触发误差, 那么用电子计数器测量信号周期的误差共有三项, 即量化误差 (± 1 量化误差)、标准频率误差和触发误差。按最坏的可能情况考虑, 在求其总误差时, 可进行绝对值相加, 即

$$\frac{\Delta T_x}{T_x} = \pm \left(\frac{1}{k T_x f_c} + \left| \frac{\Delta f_c}{f_c} \right| + \frac{1}{\sqrt{2} k \pi} \frac{U_n}{U_m} \right) \quad (3.22)$$

式中, k 为“周期倍乘”数。

本章接下来将详细介绍采用相关计数法等精度测频及相位测量的设计方案, 并给出相应的设计实例, 为后续的复杂电子系统的设计打下基础。

3.3 数字频率计系统设计

1. 设计任务

设计并实现基于单片机最小系统平台的数字频率计系统。

2. 设计基本要求

(1) 频率测量

- ① 测量范围 信号: 方波、正弦波;
幅度: 0.5~5 V;
频率: 0.1 Hz~10 MHz。

- ② 测试误差 $\leq 0.01\%$ (最大闸门时间 ≤ 10 s)。

(2) 周期测量 (技术指标及要求同频率测量)

(3) 周期脉冲信号占空比测量

- ① 测量范围 频率: 1 Hz~15 kHz;
幅度: 0.5~5 V;
占空比变化范围: 10%~90%。

- ② 测试误差 $\leq 1\%$ 。

(4) 小信号放大和整形电路

系统设计方案应考虑使用小信号放大、整形通道电路来提高系统的测量精度和灵敏度, 满足小信号的频率测量。

(5) 要求

设计前应在信息资讯上下功夫, 写出方案论证、理论分析与计算及电路实现的技术方案报告。

3. 设计提高部分要求

(1) 测量符号显示。测量时应在 LED 最高位显示测量状态, 频率测量时显示 F, 周期测量时显示 T, 占空比测量时显示自定义标志。

(2) 在完成用常规器件实现的基础上, 设计基于 CPLD 器件的数字频率计系统。同时, 写出内容翔实的技术报告, 并对测量数据进行分析, 给出误差分析的数学模型。

3.3.1 设计任务的分析及方案论证

本设计是一个基于单片机系统平台的时间参数测量系统。时间的含义有两个：一个是指“时刻”，即某个事件何时发生；另一个是指“时间间隔”，即某个事件相对于一开始时刻持续了多久。所谓频率，是指周期信号在单位时间内变化的次数。如果在一定时间间隔 T 内周期信号重复变化了 N 次，则其频率可表示为 $f_x = N/T$ 。

测量频率的方法有多种，其中电子计数器测量频率具有精度高、使用方便、测量迅速，以及便于实现测量过程自动化等优点，是频率测量的重要手段之一。由于本系统要求测频范围宽 ($0.1 \text{ Hz} \sim 10 \text{ MHz}$)、精确度高 (测试误差 $\leq 0.01\%$)，因此精确控制闸门的开启和关闭，追求计数器较高的计数频率和较大的计数容量，保持系统在整个测量频段内的测量精度不变及实现频标信号的高稳定性和高精度成为评价系统设计优劣的关键。

1. 频率测量原理与方案论证

(1) 测频方案论证

由于对电子计数器的测量误差进行了较为详细的讨论，使得测频方案的论证有了理论依据。下面给出不同方案的比较。

① 直接测频法(闸门时间计数法)

所谓直接测频法，是指在确定的闸门时间内，通过计数器记录待测信号周期变化的次数，并根据频率的定义计算待测信号的频率。

由图3.7可知，测频法是在一单位时间 T 内测出通过控制闸门的脉冲数 N ，信号频率 $f_x = N/T$ 。由于测量的起始时刻与结束时刻相对于信号都是随机的，将会有有一个脉冲周期的量化误差，也就是说，可能在同一闸门时间 T 内对同一个被测量的测量结果会有差别，或者说在不同的闸门时间 T 内产生相同的计数值 N 。如图3.7中闸门1不等于闸门2，但计数值相等。

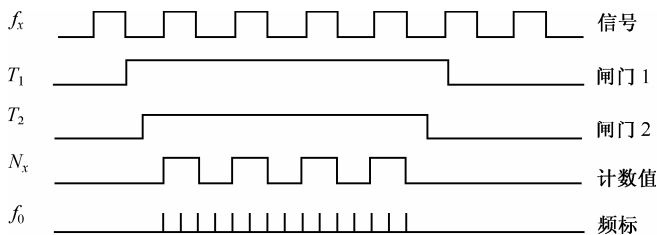


图 3.7 直接测频法示意图

若待测信号的脉冲周期为 T_x ，频率为 f_x ，则量化误差 $\gamma = T_x$ ，当测量时间 $T = 1 \text{ s}$ 时，测量准确度 $\delta = T_x/T = 1/f_x$ 。可见，测量准确度与信号频率有关 $\begin{cases} f_x \uparrow \rightarrow \text{准确度高} \\ f_x \downarrow \rightarrow \text{准确度低} \end{cases}$ ，所以测频法适合于测量频率较高的信号。

② 间接测周法

所谓间接测周法，是指在一个信号周期内记录下基准定时脉冲的个数，如图3.8所示。

信号频率为 $f_x = f_0/N_0$ 。测量的量化误差 γ 为一个基准定时脉冲周期 T_0 ，即 $\gamma = T_0$ ，测量准确度

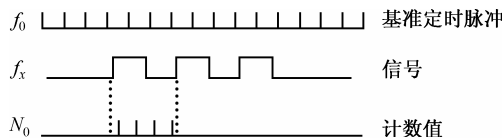


图 3.8 间接测周法示意图

$\delta = T_0/T_x$ 。因为基准定时脉冲周期 T_0 是恒定的,信号周期 T_x 越大,测量精度就越高,所以间接测周法适合于测量频率较低的信号。

③ 分段法

所谓分段法,是指采用直接测频法和间接测周法相结合的方法,将所测信号分成两个频段,高频信号的频段采用直接测频法,低频信号的频段采用间接测周法。其目的在于兼顾从低频到高频整个频段的测量精度,使之基本相等。如何划分所谓的高频和低频呢?这就涉及到中界频率的问题,即当直接测频和间接测周的量化误差相等时,就确定了一个测频和测周的分界点,这个分界点频率称为中界频率。中界频率的确定可由测频和测周的误差表达式并根据系统设计的测量精度要求求出,也可以在同一坐标图上同时作出直接测频和间接测周时的误差曲线,两曲线的交点即为中界频率点。由系统设计的精度要求(10^{-6}),在大于 100 kHz 的频段内采用直接测频法,在小于 10 Hz 的频段内采用间接测周法。

④ 相关计数测频法(等精度测频法)

所谓相关计数测频法,是指采用了多周期同步测量原理,测量输入信号的多个(整数个)周期值,再进行倒数运算而求得频率的一种测量方法。由于使用了同步闸门技术,被测信号与门控信号之间采用了同步锁定的方式,使得主门的开启时刻与计数脉冲之间的时间关系是相关的,这样便可以在整个测频范围内获得同样高的测量精度和分辨率。相关计数法测频原理的时序图如图 3.9 所示。

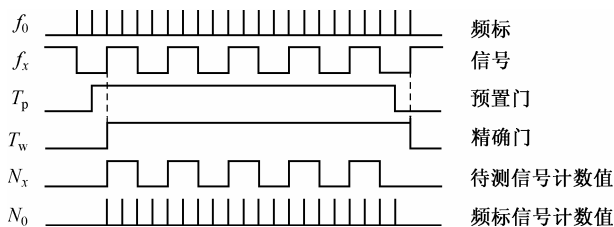


图 3.9 相关计数法测频原理的时序图

运用相关计数原理测频时,先预置闸门时间为 T_p ,预置门开启后,精确门 T_w 并不马上打开,而是由待测信号的下一个上升沿到来时开启,精确门开启后,计数器 A 和 B 分别对待测信号 f_x 和基准信号 f_0 (频标)同时计数,预置门关闭后,精确门并不马上关闭,而是在待测信号的下一个上升沿到来时才关闭。这时两个计数器停止计数,得到计数值 N_x 和 N_0 ,则被测频率为

$$f_x = N_x/T_w = N_x f_0/N_0 \quad (3.23)$$

在测量中,由于计数是在相关同步门控时间 T_w 内完成的,即由待测信号同步控制,因而 N_x 不存在误差,但是对于基准信号 f_0 来说,闸门的开启和闭合仍然是随机的,因而 N_0 存在 ± 1 量化误差。而 N_0 最多可能有 ± 1 量化误差,由于频标 f_0 很高, N_0 的计数基数很大,所以量化误差很小。

例如,频标 f_0 为 1 MHz,闸门 T_w 为 1 s 时, N_0 的精度为 10^{-6} ,测量精度可达很高的量级,而且测量精度仅与频标 f_0 和闸门时间 T_w 有关,而与被测频率无关。优化选择闸门时间和频标,可以保证在设计指标要求的整个测量频段内精度不变,即可实现等精度测量。

需要指出的是,等精度测量并非严格意义上的等精度,而是如前面所说的假定在测量中,闸门的开启和闭合受控于被测信号的上升沿或下降沿,即闸门时间 $T_w = N_x/f_x$ 。测量系统工作于同步方式,而且系统属于比较式测量仪表,其测量精度有赖于基准源(频标)的稳定度和精度。若系统要求测量精度为 10^{-6} ,基准源应有较高的开机稳定度和温度稳定度,其综合性能应优于 10^{-7} 。

(2) 测频方案选择

直接测频法和间接测周法的原理简单,电路实现较容易,但是直接测频法只适合测量频率较高的

信号,间接测周法只适合测量频率较低的信号,而且都不能满足在整个测量频段内的测量精度保持不变的要求。分段法在理论上可保证在整个频段上的等精度测量,中界频率的确定也比较容易。但是,当系统要求有较高的测量精度要求时,可能会出现测量盲区,同时如何兼顾测量精度和测量响应时间之间的矛盾也是难题。相关计数测频法要同时对未知待测信号和基准信号两路信号进行计数,而且对闸门的控制和频标的稳定度有很高的要求,但却可以满足在整个测量频段内的测量精度保持不变的要求。下面将详细讨论相关计数测频法的有关设计。

3.3.2 等精度测量的技术实现难点分析

1. 预置门与精确门

根据等精度测量原理,相关计数器计数闸门的启闭时间的控制成为系统设计的首要难点。由于等精度测量采用了多周期同步测频的原理,两个计数器在同一闸门(同步闸门)时间内分别对 f_x 和 f_0 进行计数。如何使闸门时间 T_w 准确地等于 f_x 的整周期倍数是设计的主要问题,即必须满足 $T_w = N_x / f_x$, 因而同步电路的设计是关键所在。

相关计数法测频原理框图如图3.10所示,同步闸门是由预置开门脉冲 P_0 经 f_x 同步后得到的,因而闸门时间 T_w 准确地等于 f_x 的整周期倍数,所以无 ± 1 量化误差,但由于同步闸门与 f_0 并不同步,因此 N_0 存在 ± 1 量化误差。由 $f_x = \frac{N_x}{T_w} = \frac{N_x}{N_0} \cdot f_0$, 可得同步测频的最大相对误差为

$$\frac{\Delta f_x}{f_x} = \pm \left(\frac{\Delta N_0}{N_0} + \left| \frac{\Delta f_0}{f_0} \right| \right) \quad (3.24)$$

式中, $\Delta f_0 / f_0$ 为 f_0 的频率准确度。由于晶体振荡器有较高的稳定度,误差很小 ($< \pm 10^{-7}$), 一般可忽略,因而最大相对误差 $\Delta f_x / f_x$ 主要取决于 $\Delta N_0 / N_0$, 由 3.2 节中误差分析的讨论可知,最大相对误差 $|\sigma| \leq 1/N_0$ 。假定 f_0 有较高频率,如 $f_0 = 1 \text{ MHz}$,则在 $T_w = 1 \text{ s}$ 的同步闸门时间内,可使 N_0 达 10^6 量级,则 $\Delta f_x / f_x$ 可达 10^{-6} 量级,能满足技术指标要求。

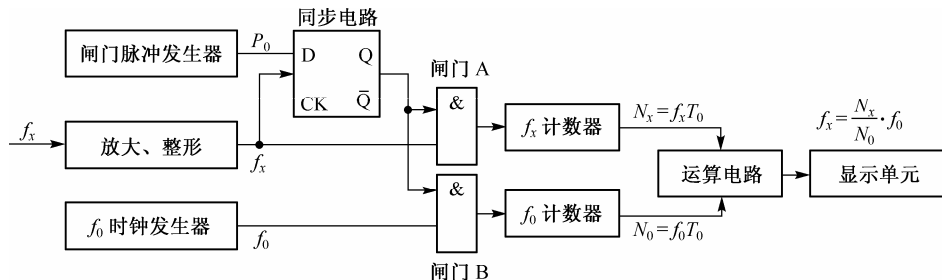


图 3.10 相关计数法测频原理框图

若要进一步提高测量精度,可对时钟计数器采用内插扩展法。由于 N_0 的 ± 1 量化误差是均匀分布的随机误差,因此也可采用滑动平均滤波技术,既能提高测量精度又不降低测量速度。

2. 测频范围(仅讨论测频法)

基于 MCS-51/52 单片机系统的定时/计数器接口,在给定的单片机晶振频率 f_c 下,可输入信号的频率上限是 $f_x \leq f_c / 24$,若晶振 $f_c = 12 \text{ MHz}$,则 $f_x \leq 500 \text{ kHz}$ 。定时/计数器的计数时间间隔可由软件延时完成,理论上定时时间可达到无穷大,即可认为没有下限频率。

测频范围是重要的技术指标,然而,单片机以 12 MHz 的晶振工作,且定时/计数器以计数方式工作时,可输入计数脉冲的最高频率为 500 kHz,限制了系统的测频范围,可采用如下解决方案:

- (1) 高频被测信号分频;
- (2) 外接级联计数器,组成宽位计数器;
- (3) 内插扩展法扩展计数容量。

在实际应用中应考虑到“分频误差”,高频信号 f_x 进行 n 分频后计数采样, n 分频将导致 $n-1$ 个待测频率周期的分频误差。其误差量级与“ ± 1 量化误差”相当,甚至更大。可以采用硬件同步分频的方法来消除可能产生的“分频误差”,并同时采用上述 (1)、(2) 和 (3) 三种方法,扩展测频范围,保证系统有较高分辨率,但这一方法的硬件开销较大。

3. 测频时间(仅讨论测频法)

如前所述,同步测频的最大相对误差为

$$\frac{\Delta f_x}{f_x} = \pm \left(\frac{\Delta N_0}{N_0} + \left| \frac{\Delta f_0}{f_0} \right| \right) \quad (3.25)$$

误差主要取决于 $\Delta N_0/N_0$,若要减少 $\Delta N_0/N_0$,必须增大 N_0 。在被测信号频率较低时,则要求闸门开启时间很长,假定被测频率为 10 Hz,精度要求为 $\pm 0.01\%$,则 $N_0 = \Delta N_0 f_x / \Delta f_x = 1/0.0001 = 10\,000$,最短闸门开启时间为 $T_0 = N_0 / f_x = 1000\text{ s}$ 。显然,这种情况是不允许出现的。

在连续测频系统中,每次测量的测量时间是一个非常重要的指标。一次测量的测量时间取决于计数器的计数周期,当频标 f_0 确定后,对不同的闸门时间有不同的测量精度,如何兼顾测量时间和测量精度也是难点之一。在实际应用的电子测量系统中,测量范围、测量精度和测量响应时间这三者相互牵制。在较小的测量动态范围内实现较高的测量精度是比较容易的;当测量动态范围较大时,若要兼顾待测信号从低频到高频的高精度测量在技术实现上是有难度的,而较高的测量精度又要求有较长的测量响应时间,理论分析已证明:多周期重复测量是实现高精度测量的重要方法。显然,对于设计者而言,应考虑如何在设计中实现既有较高的测量精度和较大的测量动态范围,又有较短的测量响应时间。

4. 计数单元电路的设计

计数单元作为数字频率计的核心,其作用是将串行的脉冲序列量化转换为二进制、十进制数据,并对转换后的数据进行锁存,以供计算机完成数据的采集与处理。如前所述,由于受单片机内部定时/计数器结构的限制、待测信号频率 f_x 不能大于单片机时钟频率的 $1/24$,显然不能满足本训练题目的技术指标要求,在单片机外设计计数单元硬件电路成为必然的选择。

对于计数单元电路的设计,最重要的技术指标参数是最高计数频率和计数器容量。最高计数频率对应于待测信号的最高频率,计数器容量对应于测量的计数精度及可能的测量范围。在 3.2 节的误差分析中已说明,对测量结果应考虑其量化误差,计数值 N_x 和 N_0 越大(对应较大的计数容量),量化误差所占的比重就越小。

为满足设计要求,必须仔细考虑计数器单元电路的技术指标要求。而在常见可编程定时/计数器电路中,8254-2 定时/计数器的计数频率为 10 MHz,是速度较高的计数器。8254-2 定时/计数器芯片结构和引脚如图 3.11 所示。

其内部有三个相互独立的 16 位可预先置数的递减计数器,且 8254-2 还有一个特别的功能,即 8254-2 可用读回命令将三个 16 位计数器同时锁存,以供“读回”计数值。根据这一特点,可以方便地将 8254-2 两个或三个计数器串联起来构成大容量计数器。用两片或多片 8254-2 分别完成 f_x 与 f_0 的

计数，显然这是一种可取的方案。

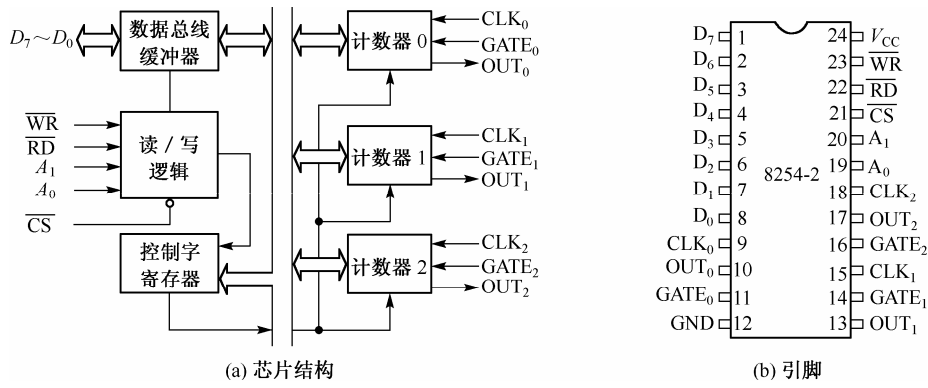


图 3.11 8254-2 定时 / 计数器芯片结构和引脚

需注意的是，读回命令对 8254-2 锁存时，必须将两片或多片 8254-2 的计数值同时锁存，而计算机在读取时又必须分时读取，应有控制电路满足这一要求。

另一选择是采取复杂可编程逻辑器件 (CPLD) 或者现场可编程门阵列 (FPGA) 来设计实现宽位计数器功能。CPLD 具有集成度高、运算速度快、开发周期短、设计灵活等特点，运用 CPLD 来直接设计符合要求的计数器，非常方便，而且极大地减小和抑制了高频信号的干扰，系统具有常规器件不可能达到的高稳定性。

5. 输入通道的设计

因输入信号幅度、频率不可预测，故在数据处理前应有合适的放大整形电路。在电子测量的系统设计中，输入通道的设计是保证仪器或系统可靠性和测量精度的关键，输入通道的基本功能包括：

- ① 波形变换、(正弦波→矩形波)整形。
- ② 放大 / 衰减。
- ③ 电平变换(产生 TTL 电平标准信号)。
- ④ 失真信号的预处理。

6. 小信号放大和限幅整形电路

理想的小信号放大器应具有高输入阻抗、较强的抗干扰能力、很宽的通频带，因此输入通道的设计通常都有较大的难度。本训练的测量范围较大，在 0.1 Hz~10 MHz 这样一个宽的频带内如不分段处理，很难找到满足要求的集成放大器。在被测信号频率的低端要求采用直流放大器，并且有高输入阻抗、高灵敏度。由于采用直流耦合，放大器本身特性受结电容影响，在高频时增益下降，须做高频补偿。在被测信号频率的高端，其设计要求更高，对放大器的稳定性和增益带宽比有较高要求，如何选择和设计电路成为难点之一。

通常在设计时以场效应管、绝缘栅 CMOS 管作为前级放大电路。由于该电路具有高输入阻抗，可降低对信号源内阻的要求。后级采用集成宽带放大电路组成小信号输入电路。

整形电路一般采用高速 CMOS 施密特触发器(74HCT132)电路构成，输入通道电路应考虑到输入信号的保护，应将小信号放大、阻抗变换及限幅整形电路作为整体来考虑。

7. 抗干扰措施

由于被测信号有较宽的带宽，在测量高频信号和小信号时极易受到干扰，必须考虑抗干扰措施，

采用的方法有：

- ① 大面积接地，引线采用屏蔽线。
- ② 电源及相关电路采用电感、电容滤波，消除相互干扰。
- ③ 采用施密特触发器，提高比较门限。
- ④ 系统自校正。
- ⑤ 软件容错技术及均值滤波等。

3.3.3 周期脉冲信号占空比测量原理

根据占空比的定义，信号的占空比等于信号的周期和信号的正脉冲宽度之比，占空比 D 可表示为 $D = \tau/T$ ，式中 τ 为脉宽， T 为脉冲周期。测量信号的占空比时，只需分别测出脉宽和周期时间，就可以计算出占空比。这一方法需要单独设计测量正脉冲宽度所需要的门控电路，并且占空比的求解须作求倒数的运算。另一简便的方法是：将被测信号与频标信号相与后的信号作为被测信号 D_x ，采用相关

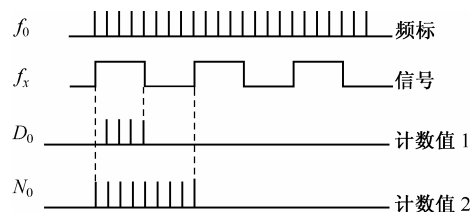


图 3.12 占空比测量原理

计数测频法在信号的周期时间段内对 D_x 和 f_0 分别进行计数，获得计数值 D_0 和 N_0 ，则被测信号的占空比为 D_0/N_0 ，如图 3.12 所示。

这一方案无论是在硬件还是软件实现上都借鉴了测频部分的设计，实现方法简单，但是会在相与操作中引入 ± 1 个脉冲的误差，在对 f_0 的计数中会有 1 个脉冲的误差，但是整体相对误差较小。

3.3.4 系统具体的设计和实现

1. 系统整体设计

测频模块设计框图如图 3.13 所示。

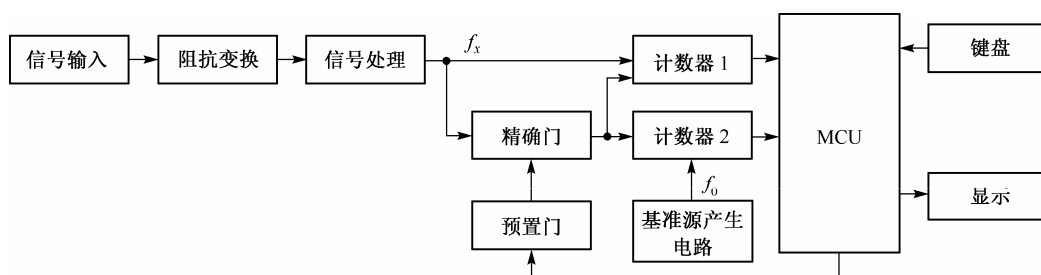


图 3.13 测频模块设计框图

2. 系统硬件设计

本训练可使用单片机最小系统平台，计数器的设计采用可编程计数器/定时器接口芯片 8254-2 实现。单片机最小系统包括键盘显示人机交互电路、82C59 中断扩展电路、外设端口地址扩展电路及数据与并行口电路等。

(1) 输入通道的设计

由于输入信号的幅度和频率不可预测，故在数据处理前应对信号进行处理，保证系统的可靠性和测量的精度。本系统对信号的处理主要包括：

- ① 波形变换：测频时，为了满足输入信号作为 8254-2 时钟信号的要求，将输入信号转换为矩

形波。

② 对信号幅度的处理和电平变换: 对小信号的放大及大信号的限幅, 要对前置放大器的增益和带宽指标进行估计。为了与计数器的输入时钟匹配, 必须将输入信号电平转换为 TTL 电平。

③ 失真信号处理: 为了避免失真信号引起误差的产生, 对输入信号利用一个高速比较器 LM361 对输入信号进行处理, 将波形变换和电平变换合而为一处理。

(2) 闸门开启和关闭的控制电路

根据等精度测频法的原理, 对相关计数闸门开启和闭合时间的精确控制是系统设计的首要难点。为了保证对被测信号的计数值 N_x 不出现量化误差, 要求闸门时间准确地等于待测信号 f_x 周期的整数倍。相关门控实现电路的原理框图如图 3.14 所示。

图中 D 触发器 A 起门控同步作用, P12 是两触发器的复位信号。P10 是预置门启闭信号, 其中 P12 和 P10 分别为单片机的 P1.2 和 P1.0 引脚。D 触发器 A 的 Q 端输出的逻辑电平为精确门信号, 配合两个与门分别控制待测信号和频标信号的通过。如前面相关测频原理所述, 精确门的开启和关闭由预置门和待测信号的上升沿决定。D 触发器 B 产生计数完成中断信号, 即在门控时间 T 之后, 计数完成则向单片机系统 82C59 发出一个中断信号 IR3, 请求单片机读出计数值, 进行计算。此时触发器 A 的精确门控信号 Q 关闭, Q 产生一个下跳沿, $\bar{Q}$ 则产生一个上跳沿, 送给 D 触发器 B, 它的 Q 端产生一个正脉冲中断信号送给 82C59。

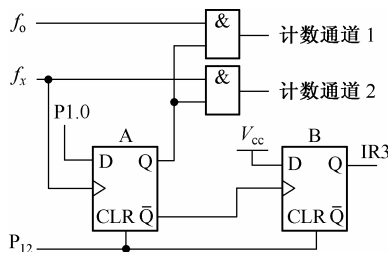


图 3.14 相关门控实现电路原理图

(3) 8254-2 的级联设计

8254-2 的计数频率为 10 MHz, 可以满足设计要求, 但是如果采用单通道计数(16 位二进制计数器), 在测量高频段信号时计数容量是不够的, 因此必须对 8254-2 进行级联, 扩大其计数容量(需说明的是, 采用溢出计数的方法也可以使计数容量较小的计数器实现高精度的较大动态范围的频率测量, 但每次高位的溢出结果必须保存, 设计较为复杂)。

为提高测频范围, 需将 8254-2 内的两个 16 位计数器进行级联, 以构成一个 32 位的宽位计数器。如何完成 8254-2 内两个计数通道的级联, 也是本设计中的一个重点。为此, 有必要对 8254-2 的计数方式和特点进行简略说明。8254-2 有 6 种工作方式, 每种方式在芯片手册上都会有详细说明, 这里只对一些容易被忽略的细节进行阐述。8254-2 正常工作前, CPU 需送一组控制字给 8254-2, 将其设置成预定的工作方式。在预置之前, 8254-2 中所有计数器的工作方式、计数初值和输出都是不确定的。8254-2 的一个重要特点是初始化时只赋值给计数器初值寄存器。而装入减 1 计数器要靠外部信号 CLK 来触发, 也就是说, 计数初值的装入严格依赖于一个完整的正脉冲, 即 CPU 送完控制字和计数初值后, 控制字装入了控制寄存器, 但计数初值并没有真正装入, 而是必须等 CLK 端出现一个完整的正脉冲后才真正装入。由于触发时序与 CLK 无关, 无论是软触发还是硬触发对于 CLK 信号都是异步的, 所以在 8254-2 的各通道内部, 都要采用各自的 CLK 信号对触发进行同步, 只有在其后 CLK 信号完成一次上升和下降后, 计数初值才得以装入, 减 1 计数器准备就绪。

显然, 单独一个事件是无法让 8254-2 计数的, 单个事件在 8254-2 的计数常数没被装入之前仅能起到装初值的作用, 而不能被认为是第一个计数值。这在级联时对于低阶计数器(低 16 位)将会会有一个计数值的误差, 而在高阶计数器(高 16 位)会引起极大的计数误差。所以, 在初始化 8254-2 时必须把高端计数初值写进去。

为了在初始化阶段就把高位计数器的初值装入, 不能将低位计数器的 OUT 端与高位计数器的 CLK 端直接相连, 必须对硬件进行改进。仔细研究 8254-2 定时 / 计数器芯片特性可知, 在往控制寄存器写

控制字时，指定计数器的输出会发生预期的变化，这一变化不受计数器时钟输入的限制，如写计数器 0 为方式 4 时，计数器 0 的输出 OUT0 立即变高，如写计数器 0 为方式 0 时，OUT0 立即变低。利用这一特性，可以在低位计数器的 OUT 端与高位计数器的 CLK 端之间加反相器，通过写低位计数器工作方式的形式来人为地控制低位计数器输出的电平状态，从而模拟出一个正脉冲，使得高位计数常数得以装入。8254-2 内部计数器的级联图如图 3.15 所示。

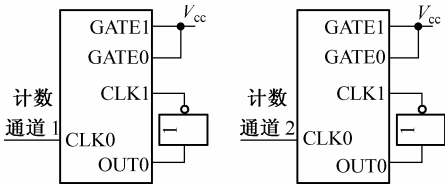


图 3.15 8254-2 内部计数器的级联图

(4) 高稳定度的基准源产生电路

基准源信号直接关系到系统测量的精确度和稳定度，在设计时应力求保证输出的频标信号具有很高的精确度和稳定度，可采用高精度的晶振作为振荡源并配合适当的外围电路完成电路设计，参考电路如图 3.16 所示。

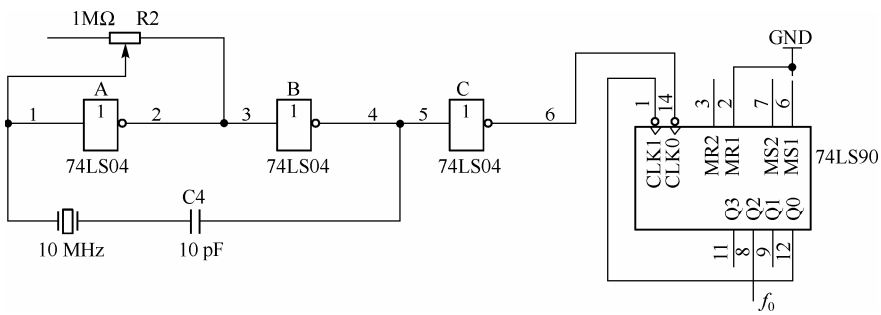


图 3.16 高稳定度的基准源产生电路图

3. 系统软件设计

(1) 系统软件设计流程图

系统主程序的参考设计流程图如图 3.17 所示。

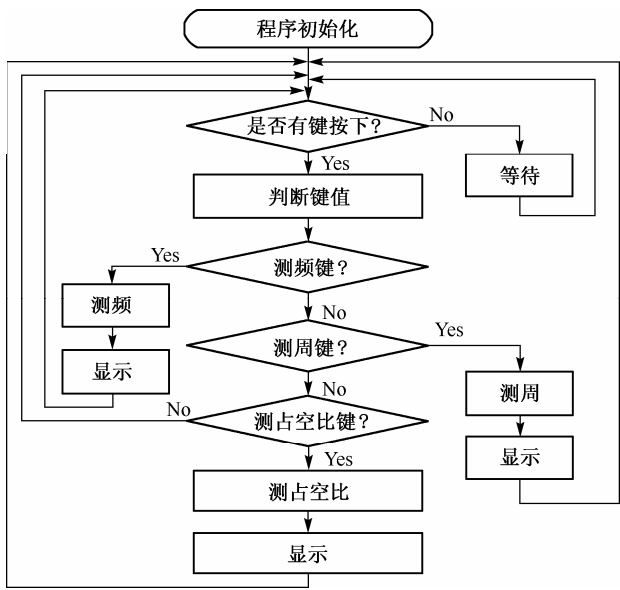


图 3.17 系统主程序的参考设计流程图

(2) 软件设计中的几个重要环节

① 对闸门启闭的控制程序设计

考虑到 8254-2 的最高计数频率为 10 MHz, 为了避免引入额外的误差, 没有对被测信号进行分频。从 3.3.2 节的讨论中可以看出: 一方面, 预置门控时间 T_P 影响到精确门控时间 T_w (由 $T_w = N_x / f_x$ 可知, 精确门和预置门仅有 ± 1 计数脉冲周期的差别), 继而会影响计数器的计数容量; 另一方面, 预置门控时间 T_P 也会影响到测量中实时跟踪频率变化的速度, 较长的测量响应时间使系统设计指标劣化。综合考虑, 单片机编程时设置预置门控时间为 1 s, f_0 为 1 MHz。要实现整个频段内的等精度测量, 则在 1 Hz 以上定时 1 s, 在 1 Hz 以下定时 10 s。在软件中定时 1 s 后, 开启中断 IR3, 再延时 1 s 后, 期间若检测到中断, 则去执行数据的读取及数据处理; 若没有检测到中断, 则再延时 8 s 去等待中断, 即可实现 1 Hz 以下信号的频率测量。

② 8254-2 的级联程序设计

鉴于 8254-2 的工作特点, 在用反相器进行计数器级联后, 构成 32 位的宽字节计数器。采用软件给低阶计数器写控制字使低阶 OUT 端输出不同高、低电平的方式, 在高阶计数器的 CLK 端引入一个正脉冲, 从而使计数初值装入高阶计数器的计数初值寄存器。

具体指令操作如下:

- 高阶计数器控制字和计数初值的写入;
- 低阶计数器设置为方式 4;
- 低阶计数器设置为方式 0;
- 低阶计数器设置为方式 4;
- 低阶计数器控制字和计数初值的写入。

在级联计数器的低端, 则采用软件加 1 的方法来弥补低阶计数误差。

③ 计数器读回程序设计

利用 8254 读回功能读取计数值时, 32 位的计数值需分 4 次读入, 为保证计数值的正确读入, 计数器读回程序设计应能做到同时对所有的计数通道进行锁存, 然后分别读出。

3.3.5 基于CPLD的数字频率计的设计

由以上基于 8254 和单片机系统的实现方案可知: 8254 本身的缺陷导致了测量误差; 8254 的计数速度限制了测频范围的进一步提高; 硬件电路也不简捷, 调试不方便。鉴于此, 在相关计数原理和以上方案已实现的基础上, 本训练要求使用 CPLD 芯片 EPM7128 重新实现数字频率计的设计, 用以提高读者的硬件设计能力。系统的设计指标要求将测频范围扩展为 0.1 Hz ~ 16 MHz, 其余指标不变。EPM7128 是 ALTERA 公司 MAX7000s 系列 CPLD 器件, 可用门数为 2500 门, 全局时钟一般可用 50 MHz 有源晶振。芯片采用 E²PROM 或 Fast Flash 技术, 断电后, 编程数据可以保持, 由于芯片下载和相应开发软件 MAX+PLUS II 使用简单方便, 当前在公司及高校实验教学中得到了广泛的运用。该方案电路框图如图 3.18 所示。

设计的关键是用 Verilog HDL 或 VHDL 语言描述实现图 3.18 中两个计数器的功能。由于 CPLD 生成的计数器不再出现 8254 计数的问题, 自然就消除了丢失计数脉冲所带来的测量误差。考虑预置门控时间对计数容量的影响和 EPM7128 内部资源, 两个计数器均为 24 位。在 MAX+PLUS II 中的顶层模块电路图(gdf 文档)如图 3.19 所示。由图 3.19 可以看出, 其测量原理及设计思想和基于分立器件 8254 的方案是完全相同的, 这里不再赘述。计数器源代码附于图 3.19 后。用 CPLD 实现的前端控制和

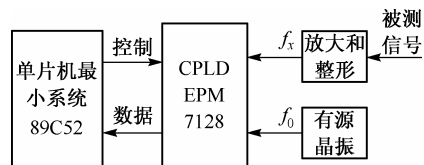


图 3.18 基于 CPLD 的测频系统电路框图

计数电路,不再存在丢失计数脉冲的问题。而且从图 3.19 可以看出,计数器由于不需要写入控制字和计数初值 0,它与单片机系统接口也就不再需要读写信号,可由片选信号 CS 和地址信号 A0, A1 配合来读出计数值。由此也可以看出,在 CPLD 应用中,可以通过硬件描述语言的编程来自行设定所需的单元电路的工作方式,以便和单片机系统相连。这种方法由于可移植性强,可以在拥有更多内部资源的 CPLD 或 FPGA 芯片中使用,以提高计数容量和频率计的测量范围,应用非常灵活和方便。

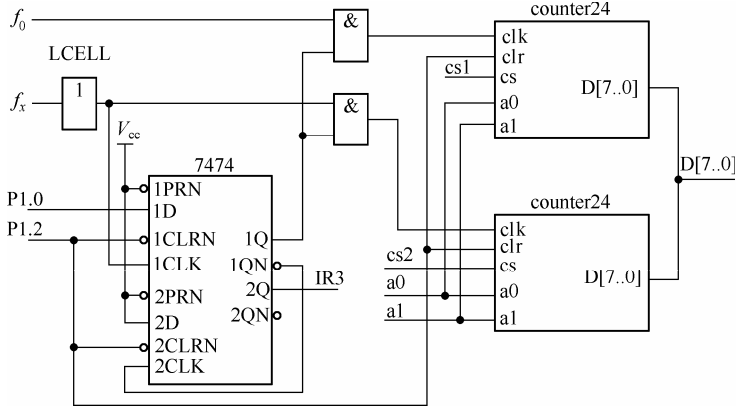


图 3.19 CPLD 测频模块顶层电路图

24 位计数器 Verilog HDL 源代码如下。

```
module counter(clk, clr, cs, a1, a0, q);
    output [7:0] q;
    input clk, clr, cs, a1, a0;
    reg [7:0] q;
    reg [1:0] sel;
    reg [23:0] cout;
    always@(posedge clk)
    begin
        if(clr)
            cout=0;
        else
            cout=cout+1;
    end
    always@(cs)
    begin
        sel={a1, a0};
        if(cs==0)
            begin
                case(sel)
                    2'b00:q=cout[7:0];
                    2'b01:q=cout[15:8];
                    2'b10:q=cout[23:16];
                    default:q=8'b00000000;
                endcase
            end
    end
end
```

```
q=8'bzzzzzzzz;  
end  
endmodule
```

本设计是典型的时间参数测量问题。该数字频率计具有测频、测周、测占空比等功能。本设计的重点是在满足精度的情况下如何实现测频和测时(包括测周和测占空比)。难点有两个:一是在给定的测量范围、测量精度的情况下,如何选择合适的设计方法和设计参数来实现有效的测量;二是如何对小信号进行处理,以满足测量的需要。

本设计主要讨论的是如何通过常规器件来实现测量功能,但从整个设计的过程来看,基于常规器件的数字频率计的设计与实现是基于 CPLD 的数字频率计得以实现的前提;可以说,没有对常规器件的设计思想的形成和设计方法的掌握,把一个毫无实践根据的想法盲目的移植到 CPLD 器件中将是无意义的。通过系统设计,使读者从整体上和细节上对于系统有了一个全面深刻的认识。在此基础上,可以通过硬件描述语言,将其移植到 CPLD 来实现时间参数测量系统的模块化和集成化设计。基于 CPLD 的数字频率计无论从设计所达到的精度,还是从设计的简易程度来看,都明显优于基于常规器件的设计。

3.4 相位测量

1. 设计任务

设计并制作一个低频相位测量仪。

2. 设计基本要求

- (1) 频率范围: 20 Hz~20 kHz。
- (2) 相位测量仪的输入阻抗 $\geq 100\text{ k}\Omega$ 。
- (3) 允许两路输入正弦信号峰-峰值可分别在 1~5 V 范围内变化。
- (4) 相位测量绝对误差 $\leq 2^\circ$ 。
- (5) 具有频率测量及数字显示功能。
- (6) 相位差数字显示: 相位读数为 $0^\circ \sim 359.9^\circ$, 分辨率为 0.1° 。

3. 设计提高部分要求

在保持相位测量仪测量误差和频率范围不变的条件下,扩展相位测量仪输入正弦电压峰-峰值至 0.3~5 V 范围。

3.4.1 相位测量方案分析与论证

本设计要求制作一个测量任意两同频正弦信号之间的相位差的相位测量仪,并将测量结果以数字形式显示出来。信号在传播过程中,其相位是以某一规律周期变化的,检测信号的绝对相位是没有意义的。相位差测量的实质是提取相对于某一参考点的相位差信息,通常测量两个同频信号之间的相位差有相位-幅度转换测量法和相位差-时间转换测量法。

1. 相位-幅度转换测量法

该测量方法利用信号的干涉原理,将信号 v_1 , v_2 馈入一个合路器(耦合器),产生合信号:

$$V = v_1 + v_2 = A\cos(\omega t) + A\cos(\omega t - \varphi) \quad (3.26)$$

再将合信号经过检波器后,测出信号的直流分量,得到的直流分量的幅度与相位差在区间 $\varphi \in (0, \pi)$ 上建立一一对应关系。即测得直流分量的幅度值,从而获得相位差 φ 。

这一方法不同于传统的相位提取方法,因电路要求利用互补的对数和指数放大器实现幅度值的归一化,从而建立不受输入信号幅度和温漂的影响、仅与输入信号相位差呈余弦函数的一一对应关系的幅相特性,完成相位差提取。

2. 相位-时间转换测量法

利用高精度比较器实现的相位测量,将两个同频正弦信号 v_1 和 v_2 送到两个过零比较器,将 v_1 和 v_2 分别转换成方波 f_{x1} 和 f_{x2} ,然后测量两个方波过零点的时间差 τ ,即可得到信号的相位差值,其时间差 τ 与相位差 φ 的关系为

$$\varphi = \frac{\tau}{T} \times 360^\circ \quad (3.27)$$

相位-时间转换测量法测相位差,电路结构较简单,但当输入信号有失真时,则不可避免要产生测量误差,可采用测量两个方波脉冲中心时间差的方法来代替测量两个方波过零点时间差的方法来解决这一问题。

3. 方案的选择和理论计算

由相位-时间转换测量法原理可知,时间差的测量实质上是填充计数的过程。运用相关计数测频原理,可以实现对双路同频信号相位差的测量。设信号 v_1 和 v_2 经过零比较后得到待测同频信号为 f_{x1} 和 f_{x2} ,相位差为 φ ,则测量原理时序图如图3.20所示。

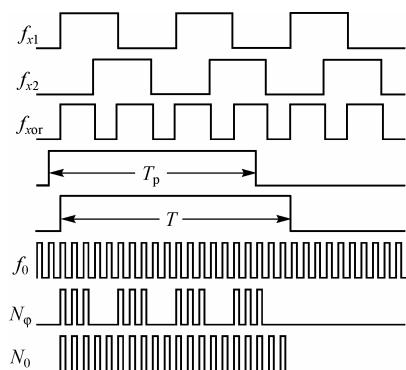


图 3.20 相关计数测量相位差原理时序图

图中 T_p 为预置门控信号的门控时间, T 为精确门控信号的门控时间, T 由 T_p 和待测信号 f_{x1} (或 f_{x2}) 共同决定。 f_{xor} 为 f_{x1} 和 f_{x2} 异或后所得信号。 N_ϕ 是精确门控信号 T 、 f_{xor} 和频标信号 f_0 相与后所得信号的计数值; N_0 是精确门控信号 T 和频标信号 f_0 相与后所得信号的计数值。与测频原理类似,分别测得 N_ϕ 和 N_0 , 便可计算出 f_{x1} 和 f_{x2} 的相位差。

那么如何根据误差指标来确定参数设置呢? 设计数脉冲频率(频标信号 f_0) 在一个周期 T 内的计数值为 N , v_1 和 v_2 经过零比较后的两个相邻前沿时间差计数值为 n , 得相位差为

$$\varphi = \frac{n}{N} \times 360^\circ \quad (3.28)$$

由此得出相位分辨率为 $360^\circ/N$ 。当被测信号的频率一定时,计数脉冲的频率决定相位差测量的精度。假定在一个信号周期内所得计数脉冲个数为 360 个,则可测相位差的最小值为

$$360^\circ/360 = 1^\circ$$

因此要满足相位测量误差的绝对误差不大于 2° 的要求,频标信号 f_0 和待测信号 f_x 应满足 $f_0 > (360/2) \times f_x$ 。当 $f_{\max}$ 为 20 kHz 时, f_0 需满足 $f_0 > (360/2) \times 20 \text{ kHz} = 3.6 \text{ MHz}$ 。

显然如同测频一样,采用基于 MCS-51/52 单片机系统的定时/计数器接口电路实现相位差测量时,待测信号的频率范围是很小的,不能满足技术指标的要求,因此要以高精度测量较高频率信号的

相位差,必须外接高速宽位计数器单元电路,应在系统设计中将频率测量和相位差测量统筹考虑,将计数器单元电路作为共用电路来设计。实现上述过程的相位差测量原理框图如图3.21所示。设两个计数器的计数值分别为 N_φ 和 N_0 ,则由图3.21可知两路信号的相位差为

$$\varphi = (N_\varphi / N_0) \times 180^\circ$$

若要明确两路信号 f_{x1} 和 f_{x2} 的相位先后关系,可以用一个上升沿触发的 D 触发器来判断 f_{x1} 超前还是滞后于 f_{x2} ,参考电路如图3.22所示。图3.22中两路信号分别接至 D 触发器的数据端 D 和时钟端 CLK,由 D 触发器的性质可知:当 f_{x1} 正跳变时,若 f_{x2} 为 1,则表明 f_{x1} 滞后于 f_{x2} ;反之,若 f_{x2} 为 0,则表明 f_{x1} 超前于 f_{x2} 。于是 Q 端的电平高低变化就表明了两路信号的超前/滞后关系,即当 f_{x1} 超前 f_{x2} 时, Q 端始终指示高电平;当 f_{x1} 滞后 f_{x2} 时, Q 端始终指示低电平。由测得的 φ 值和超前/滞后的相位关系,就可以计算得到 $0^\circ \sim 360^\circ$ 的相位差。

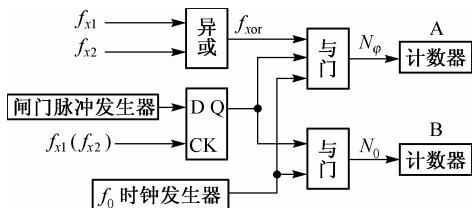


图 3.21 相位差测量原理框图

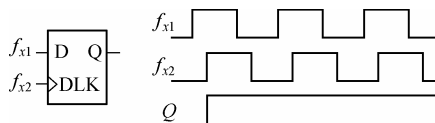


图 3.22 指示相位关系电路原理图

3.4.2 系统设计与实现

考虑到系统设计的便利性,相位测量仪的设计选用相位-时间转换测量法。首先将同频信号 v_1 和 v_2 经运算放大器放大后,输入到过零比较器中转化为矩形脉冲序列,再对其进行异或操作,并通过计数器 A 对异或后的脉冲信号进行计数,同时在精确门控制信号 T 内启动计数器 B 对频标信号 f_0 计数,通过运算即得到信号 v_1 和 v_2 的相位差。而 v_1 和 v_2 的频率测量则可利用相关计数法实现,最后由单片机控制显示单元显示频率和相位差。低频相位测量仪的设计框图如图3.23所示。

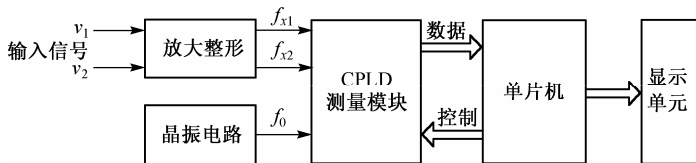


图 3.23 低频相位测量仪设计框图

系统设计以单片机最小系统及 CPLD 为核心,单片机最小系统包括键盘显示人机交互界面、中断扩展电路、外设端口地址扩展电路及数据与并行接口电路等。单片机最小系统向 CPLD 发出控制信号,待一次测量结束后,接收并处理来自 CPLD 的测量数据,实时显示处理结果。

系统设计中小信号处理部分是设计的难点之一。小信号处理电路包括阻抗变换、放大限幅、电平转换和波形整形等部分。设计任务要求输入阻抗不小于 $100\text{ k}\Omega$,可以采用跟随器作为前向通道的输入电路,再在输入端并接一个 $100\text{ k}\Omega$ 的电阻,这样就能满足输入阻抗的要求。电路中运算放大器的选择也要根据设计要求,选择带宽和性能合适的运算放大器。

CPLD 测频/测相顶层电路如图3.24所示。MUX21 模块为 2 选 1 数据选择器, COUNTER24 模块为 24 位计数器,单片机 P1.0 产生预置门控信号, P1.2 用于相位差、频率测量选择, P1.1 为两个 D 触发器的复位信号, P1.3 指示两路信号相位的先后关系。第一个 D 触发器的输出端 1Q 产生精确门控信

第4章 数字信号源的设计与实现

4.1 引言

能产生不同频率、不同幅度的规则或不规则波形的信号发生器称为信号源。信号源在电子系统的研制、生产、测试、校准及维护中有着广泛的应用。例如在电子系统设计的过程中需要对系统进行测试,而系统电参数的数值或特征(如输入阻抗、电压增益和频率特性等)必须在一定的电信号作用下才能表现出来。这时可以借助于信号源,将其产生的信号作为输入激励信号,观察系统的响应以获得测试结果。信号源主要有三个方面的用途:(1)作为激励源,无论是模拟系统,还是数字系统,都必须用信号源来激励;(2)作为标准信号源,用于为电子设备的测量校准提供标准信号;(3)用于信号仿真,当研究电子设备在实际环境下所受到的影响时,可能无法到实际环境中去测量,但可以利用信号源给其施加与实际环境相同特性的信号来测量,如噪声信号、高频干扰信号等,这时信号源就要仿真实际的特征信号。

对信号源的基本要求主要有以下几个方面。

(1) 输出信号的波形参数是已知的,即能输出指定的波形。对于正弦信号而言,波形参数是指幅度、谐波含量、调幅系数、频偏和扫频范围等。对于脉冲信号而言,波形参数是指上升时间、下降时间和平顶下降等。

(2) 输出信号的重复频率已知,能在一定的范围内调节。由测量的特征和要求来决定频率范围及其准确度、稳定度等指标。

(3) 输出信号的幅度已知,且能在一定的范围内调节,调节可以是步进的,也可以是连续的,并且具有一定的稳定度。

(4) 输出阻抗已知。通常高频信号源的输出阻抗为 $50\ \Omega$,低频信号源的输出阻抗为 $600\ \Omega$ 。

本章介绍了常用的数字频率合成方法,着重讨论基于单片机最小系统的 PLL 数控信源设计和 DDS 数控信源设计,分别讨论了它们的实现原理和实现方法,分析了它们的误差并给出了如何改进的措施。本章在读者已经熟练掌握数控信源设计的基础上,还进一步讨论了任意波形发生器和数字移相器的设计原理和实现方案。

本章的设计实例和第3章的数字频率计设计都是具有典型意义的现代电子系统训练实例,通过对这些典型设计实例的介绍,希望能够培养读者根据设计的要求进行系统分析、方案论证、数学计算、模块设计、系统调试和结果分析的能力,让读者通过这些训练后不但能“知其然”,更能做到“知其所以然”,理解设计过程,领悟“设计”才是这些训练的目的。

4.2 频率合成技术及常用方法介绍

数字信号源的设计与实现,其核心问题是频率合成器生成原理与实现方法的掌握和应用。所谓频率合成,是指从一个高度稳定和准确的参考频率经过各种技术处理,生成所需的离散的频率输出的技术。现代频率合成的技术处理方法主要有直接频率合成(Direct Frequency Synthesis, DFS)、锁相环(Phase Locked Loop, PLL)式频率合成和直接数字频率合成(Direct Digital Synthesis, DDS)。从设计的角

度来说,根据目标任务的不同,频率合成的技术处理方法可以采用传统的硬件方式,也可采用锁相技术和各种数字技术、计算机技术来实现频率的加、减、乘、除基本运算。频率合成的参考频率由高稳定度的参考振荡器(一般为晶体振荡器)产生,所生成的一系列离散频率与参考频率有严格的比例关系。运用频率合成技术获得不同频率信号的组件及仪器称为频率合成器。

4.2.1 直接频率合成(DFS)

直接频率合成是传统的频率合成方法。直接频率合成器由混频器、倍频器、分频器和带通滤波器组成,由一个或多个参考频率通过倍频、分频和混频实现对频率的加、减、乘、除四则运算来合成所需的频率,并用窄带滤波器选出。直接频率合成主要有强制法、谐波法、双混频法、三混频法和双混频-分频法等。直接频率合成器设计中混频比的选择应仔细考虑,双混频法的混频比为 $r = f_1/f_2$,其中 f_1 和 f_2 是加到混频器的两输入频率。如果混频比选择不合适,则可能造成混频器某一次 $\pm mf_1 \pm nf_2$ 的互调频率分量与所需频率靠得很近,用再窄的滤波器也无法滤除的情况。在实际应用中,通带内的无用互调分量是不能滤除的,所以混频比的选择应使得通带内互调分量的电平在允许的范围之内。

双混频-分频法是直接频率合成中常用的一种方法,其原理框图如图4.1所示。模块输入为固定频率 f_i 和离散频率 f_n ,输出即为所需频率 $f_i + f_n/10$ 。图4.1中 f_1 和 f_2 为辅助频率,其作用是使混频器输出的和频与差频的间隔增大,以便带通滤波器将不需要的差频成分滤除。

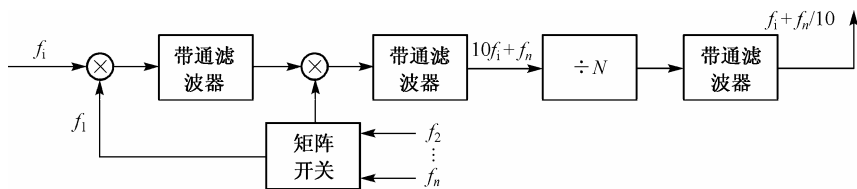


图4.1 双混频-分频法原理框图

f_1 和 f_2 的选取应满足 $f_i + f_1 + f_2 = 10f_i$ 。例如可选取 $f_i=1\text{ MHz}$, $f_1=3\text{ MHz}$, $f_2=6\text{ MHz}$ 。只要 f_n 为具有一定增量的10个频率中的一个,那么用多个这样的模块串接,就可以很容易构成所需频率分辨率的合成器。

直接频率合成可以产生很低和很高的频率,具有频率转换时间短、相位噪声低、频率分辨率几乎任意高和在频率转换时不易发生错频等优点。但是由于直接频率合成需要大量的硬设备(如振荡器、混频器、滤波器等),使得频率合成器体积庞大、成本高、结构复杂,且容易产生难于抑制的寄生频率(即杂波),频带越宽,可能出现的寄生分量就越多。这些缺点大大抵消了直接频率合成在功能、速度等方面的优势,因此在大多数的应用场合,直接频率合成已经被基于锁相技术的合成方法所取代。

4.2.2 采用锁相环(PLL)电路的频率合成

锁相环频率合成利用锁相环把压控振荡器(VCO)的频率锁定在某一谐波或组合频率上,由压控振荡器产生所需要的频率。因为合成过程中要完成从频率—电压—频率的变换,所以又常称为间接频率合成。基本的锁相环频率合成器的原理框图如图4.2所示,它由晶振、鉴频/鉴相器(FD/PD)、环路滤波器(LPF)、可变分频器($\div N$)和VCO组成。在环路锁定时,鉴相器两输入端的频率是相同的,且相位差恒定,即

$$\begin{cases} f_r = f_d \\ \theta_r - \theta_d = C \end{cases} \quad (4.1)$$

f_d 是由压控振荡器的输出频率 f_0 经过 N 次分频得到的,因此

$$f_d = f_0 / N \quad (4.2)$$

所以输出频率是参考频率的整数倍, 即 $f_0 = N \times f_r$, 再对 f_0 进行 f_r 分频, 得到 $N \text{ Hz}$ 的脉冲信号。如果 N 是由外部数字量控制的, 就实现了输出脉冲波频率的任意设置。

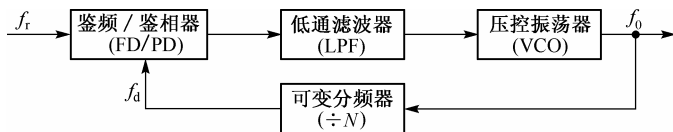


图 4.2 基本的锁相环频率合成器的原理框图

这样, 环中带有可变分频器的 PLL 就提供了一种从单个参考频率获得大量频率的方法。环中的可变分频器 ($\div N$) 用可编程分频器来实现就可以按增量 f_r 来改变输出频率, 这是组成锁相频率合成器的一种最简单的方法。锁相频率合成的优点是由于锁相环 (PLL) 相当于一窄带跟踪滤波器, 因此能很好地选择所需频率的信号, 抑制杂散分量, 输出频谱纯度高、稳定性好, 且避免了大量使用滤波器, 有利于集成化和小型化, 同时能得到很高的信号输出频率。

然而, 这种简单的锁相频率合成器也存在一些问题, 以致难以满足合成器多方面的性能要求。第一, 可编程分频器的最高工作频率往往要比合成器所需的频率低许多, 如图 4.2 所示, 将压控振荡器的输出直接加到可变分频器是不行的。解决这个问题可以用前置分频器、多模分频器及下变频等多种方法; 第二, 输出频率以增量 f_r 变化, 即分辨率等于 f_r 。为了提高合成器的频率分辨率就必须减小 f_r , 但是这与较短转换时间的要求是矛盾的。转换时间工程上常用的经验公式为 $t_s = 25 / f_r$, 即转换时间大约需要 25 个参考频率周期。所以分辨率与转换速度成反比, 显然简单的 PLL 合成器是不能同时满足这两个要求的, 可以采取多环频率合成或小数分频等方法加以改善, 但这是以增加电路的复杂度为代价的。

锁相理论发展成熟, 其电路复杂度小于直接频率合成, 输出频谱纯度高、稳定性好、杂散电平低。目前, 锁相式频率合成技术依然在频率合成领域占有重要的地位, 并得到了广泛的应用。

4.2.3 直接数字频率合成 (DDS)

DDS 所依据的基本原理是: 定频信号的相位变化与时间呈线性关系, 并且相位变化率为一个常数, 不同频率信号的相位变化率也不同。因此, DDS 的工作原理可以描述为: 一个给定频率的数字化波形能被以更高频率的累加相位方式合成。相位累加器用来实现这种过程, 不同的频率对应于不同的相位累加步进量。在参考时钟的推动下, 相位累加器通过对存储器中波形查找表的寻址, 得到输出频率的离散化振幅值; 经过 D/A 变换后得到连续的量化振幅值; 再经过低通滤波器得到所需频率的模拟信号。基于相位累加器及正弦查找表算法原理的 DDS 的原理框图如图 4.3 所示。由图 4.3 可看出, DDS 主要由相位累加器、只读存储器 (ROM)、数模转换器 (DAC) 及低通滤波器 (LPF) 构成。

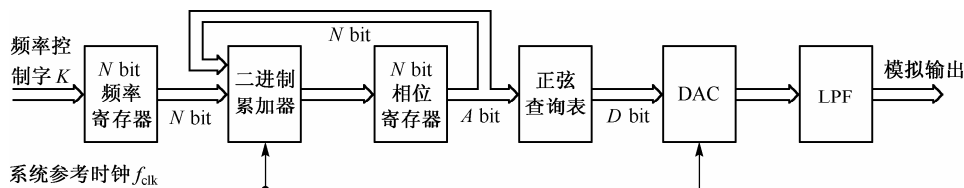


图 4.3 基于相位累加器及正弦查找表算法原理的 DDS 的原理框图

DDS 有两种基本的实现方法: 一种是根据正弦函数关系式, 按照一定的时间间隔利用计算机求解

数字递推关系式,求解瞬时正弦函数幅值并实时地送入数模转换器,从而合成出所要求频率的正弦信号。这种方式具有电路简单、成本低的特点,但受计算机运算速度的限制,合成信号频率较低。另一种是利用硬件电路取代计算机的软件运算过程,查寻高速存储器中存储的数字形式的正弦波幅值。大多实际应用的 DDS 合成器都是使用查表法,如图4.3所示。查表法的基本思想是在存储器里存入正弦波的 N 个均匀采样值,以均匀速率把这些样本输出到数模转换器变换成模拟信号。由一个周期输出的样本个数决定输出频率的大小。

图4.4(a)显示了一个产生正弦波的相位累加器,其频率为 $1/8$ 时钟频率,圆周显示相位累加器的每时钟周期步进为 $\pi/4$ (rad)。圆周上的点表示给定时间的相位值,正弦波上的点表示相应的幅度,这一相位幅度转换发生在正弦查找表中。每时钟周期的相位累加增量是 $\pi/4$ (rad),即 2π 的 $1/8$ 。

由图4.3和图4.4可看出,一个 N 位字长的二进制加法器的一端和一个以固定时钟脉冲取样的 N 位相位寄存器相连,另一个输入端连接的是频率寄存器,频率寄存器受外部输入的频率控制字 K 的控制。在每一个时钟到来时,前一次相位寄存器中的值和当前的 K 值相加,并作为当前相位寄存器的输出。加法器不断的对相位增量进行线性累加,每产生一次溢出,即完成一个正弦波周期的取样。所得出的相位值 ($0 \sim 2\pi$) 被输出至正弦查找表,查找表将相位信息转化为相应的正弦幅度值,再经数模变换器得到相应的阶梯波,最后经低通滤波器对阶梯波进行平滑,即得由频率控制字 K 决定的连续变化的输出信号波形。

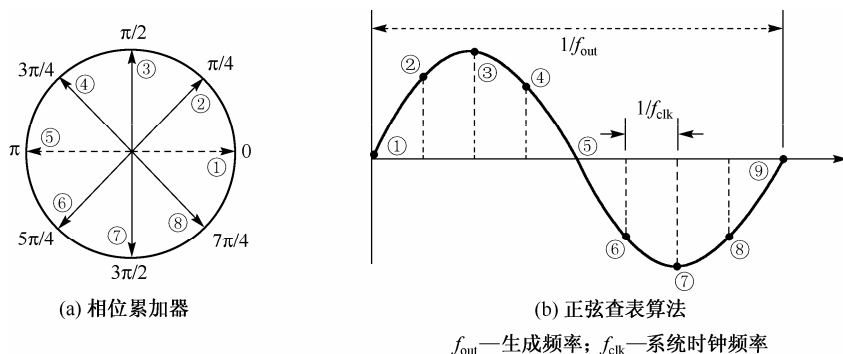


图 4.4 相位累加器及正弦查表算法原理

根据上述原理,为合成所需频率信号,需要完成以下步骤。

① 控制每次采样的相位增量(即频率控制字 K)并对其累加,输出模为 2π 的累积相位(用相位累加器完成)。

② 将模 2π 的累积相位变换成相应的正弦波的幅值,一般用 ROM 来存放正弦函数的相位-幅度对应表。

③ 用 DAC 将幅度代码转换成模拟电压信号。

④ DAC 输出的电压信号是阶梯波形,需经 LPF 平滑后才能得到所需的模拟电压信号输出。

因为 DDS 信号的产生是在数字域中实现的,所以可被很精确地操作,其函数特性很容易用软件来构造。DDS 对频率分辨率的控制能够接近 m Hz 数量级,具有超高速的频率转换能力(ns 到 μ s 之间),所有换频在连续相位方式下自动完成。此外,DDS 具有集成度高、体积小、可输出宽带正交信号,很容易实现调频(FSK)或调相(PSK)及灵活的数字调制等特点。正因为 DDS 的优越性,其应用日益广泛。

在接下来的几节中将分别讨论数控信源、任意波形发生器和数字移相器的设计和实现,其基本思想是:通过分类细化训练内容,使读者掌握 PLL 和 DDS 的频率合成的数学模型和物理模型,尤其是对 DDS 的原理、数学模型及噪声模型应有较为全面的理解,从而建立科学、合理的电子系统设计思想。

4.3 基于PLL的数控信源的设计

1. 设计任务

在给定 $\pm 15\text{ V}$ 电源电压条件下,设计并制作一个基于PLL的正弦波和脉冲波信号源。

2. 设计基本要求

(1) 正弦波信号源

① 信号频率: $20\text{ Hz}\sim 20\text{ kHz}$ 步进调整,步长为 5 Hz 。

② 频率稳定度: 优于 10^{-4} 。

③ 非线性失真系数: $\leq 3\%$ 。

(2) 脉冲波信号源

① 信号频率: $20\text{ Hz}\sim 20\text{ kHz}$ 步进调整,步长为 5 Hz 。

② 上升时间和下降时间: $\leq 1\text{ }\mu\text{s}$ 。

③ 平顶斜降: $\leq 5\%$ 。

④ 脉冲占空比: $2\%\sim 98\%$ 步进可调,步长为 2% 。

(3) 上述两个信号源的公共要求

① 频率可预置。

② 在负载为 $600\text{ }\Omega$ 时,输出幅度为 3 V 。

③ 完成 5 位输出频率的数字显示。

3. 设计提高部分要求

(1) 正弦波和脉冲波的频率步长都改为 1 Hz 。

(2) 正弦波和脉冲波的幅度可步进调整,调整范围为 $100\text{ mV}\sim 3\text{ V}$,步长为 100 mV 。

(3) 正弦波和脉冲波的频率可自动步进,步长为 1 Hz 。

(4) 降低正弦波非线性失真系数。

4.3.1 系统设计方案分析

数控信源的设计与实现,其核心问题是频率合成器生成方法的选择。本训练要求基于PLL电路设计一个频率可设置、步进可调的正弦波和脉冲波信号源,其整体设计框图如图4.5所示。系统设计采用锁相合成技术,利用锁相环将VCO的输出频率锁定在所需频率上,可以很好地选择所需频率信号,抑制杂散分量,并可避免使用大量的滤波器。锁相频率合成的输出频率是 N 分频设置的分频系数,从而实现由单个参考频率获得大量精度与参考频率精度相当的频率。系统设计的原理是根据单片机键盘输入,通过锁相环合成频率可任意设置的脉冲波信号(所谓的可任意设置,是指在技术指标要求的 $20\text{ Hz}\sim 20\text{ kHz}$ 范围内,可以以 1 Hz 的频率分辨率任意设置不同频率的脉冲波信号),通过脉冲波占空比设置电路和幅度调整电路实现输出频率在 $20\text{ Hz}\sim 20\text{ kHz}$ 范围内的脉冲波的占空比、幅度的步进可调,通过脉冲波到正弦波变换电路产生 $20\text{ Hz}\sim 20\text{ kHz}$ 的正弦波,在输出波形的同时将参数显示在显示单元上。

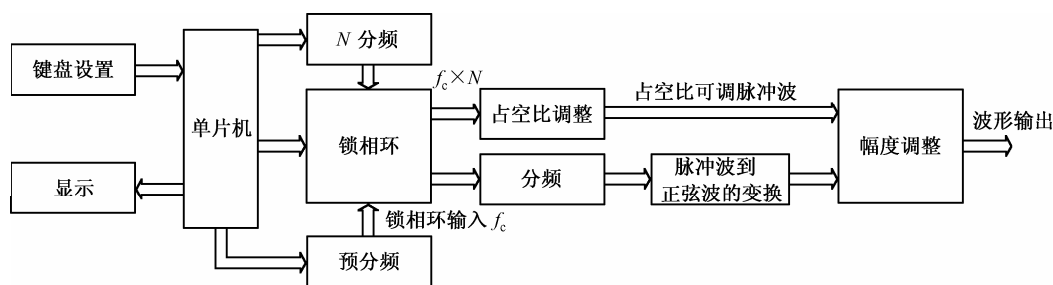


图 4.5 基于 PLL 的数控信源系统的整体设计框图

该设计的关键组成部分包括：

- 频率可任意设置的脉冲波产生电路；
- 实现脉冲波占空比可任意设置的电路；
- 脉冲波到正弦波的变换电路；
- 输出波形幅度可调电路。

4.3.2 系统模块分析和设计

1. 频率可任意设置的脉冲波的产生

(1) 参考频率的确定。锁相环电路采用专用芯片 74HC4046，它的内部主要由相位比较器、压控振荡器、源跟随器、线形放大器和整形电路等组成，74HC4046 的最高振荡频率 $f_{\max}$ 为 12 MHz ($V_{CC}=4.5\text{ V}$)。由 74HC4046 的技术指标可计算出参考频率 f_r 为 $f_{\max}/f_0$ 。其中， f_0 为设置的最高输出频率(本设计要求为 20 kHz)，可求得 $f_r = 600\text{ Hz}$ 。对于 f_r 的取值，应综合考虑设计指标中对占空比控制范围的要求及低通滤波器的设计要求， f_r 取 600 Hz 不利于占空比控制电路的设计。综合考虑后确定 f_r 为 100 Hz，此时占空比控制电路的输入信号频率 $f_c = f_r \times N = 100 \times N\text{ Hz}$ ，由 $f_c = f_r \times N$ 作为占空比控制电路中的 100 分频计数的时钟信号，则占空比控制电路的输出脉冲信号频率为 $N\text{ Hz}$ ，即可实现 1%~99% 的占空比调节。参考频率 f_r 取 100 Hz 对于低通滤波器的设计也带来了便利性，在后面有详细的讨论。

(2) 基准频率的生成。对于基准频率的生成，考虑到精度要求，一般要求采用晶体振荡电路。可将 1 MHz 的晶体振荡电路通过整形电路整形和分频，获得频率为 100 Hz 的参考源。另一选择可由单片机 ALE 信号分频得到 100 Hz 的参考源，这一方案的前提是 ALE 信号已经修复，如 2.1 节所述。

(3) 分频器的设计。假设采用单片机的 ALE 信号分频得到 100 Hz 的参考源，分频比为 $N_{\text{ALE}} = f_{\text{ALE}}/100 = (f_{\text{osc}}/6)/100 = 2 \times 10^4$ 。其中， $f_{\text{osc}} = 12\text{ MHz}$ 为单片机的时钟振荡频率， f_{ALE} 为 ALE 信号频率。利用可编程定时/计数器 8254 的三个 16 位定时器，使其工作在方式三，用以完成分频器的设计。其中，定时器 0 分频比设为 2×10^4 ，产生 100 Hz 的方波信号作为 74HC4046 的基准时钟。定时器 2 作为锁相环的 N 分频器，实现基准时钟频率的 20~20 000 倍频，即得到频率为 $100 \times N\text{ Hz}$ 的方波。定时器 1 分频比为 100，可实现对倍频后的信号 100 分频，即可得到输出频率为 20~20 kHz、步长为 1 Hz 的方波信号。此部分的参考电路如图 4.6 所示。

2. 脉冲波占空比可任意设置的电路的实现

根据题目要求，脉冲占空比为 2%~98% 步进可调，步长为 2%，占空比控制电路的输入信号频率为 $f_c = 100 \times N\text{ Hz}$ ，如采用计数法，占空比从 2% 开始变化为 98%，对应的脉宽内所计的脉冲个数为 2~98 个。

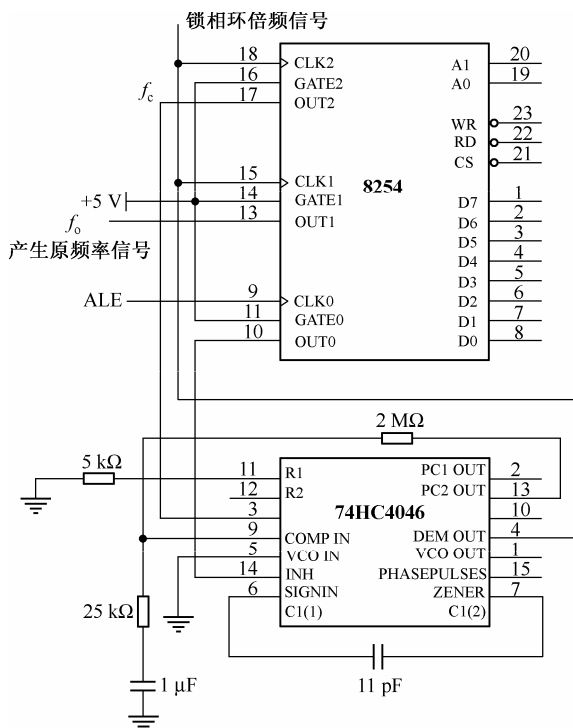


图 4.6 任意频率脉冲波产生电路原理图

如果利用单片机对输入信号计数的方案来控制脉冲波占空比，并设置比较数值，当计数值小于比较数值时，输出 1，否则输出 0，这样电路简单，而且充分利用了单片机资源，但是受单片机工作时钟频率的限制，对信号频率的高端（最大计数频率为 $20\text{ kHz} \times 100 = 2\text{ MHz}$ ）的计数是十分困难的。

另一选择是，利用计数器和数值比较器构成数控占空比调节电路，基于计数器和数值比较器芯片的高速特性，既可以使占空比准确调节，又可以不受输入信号频率的限制，电路结构也不复杂。计数器可以利用 8254 或者采用两个 74LS90 构成计数周期为 100 (0~99) 的计数器来实现。

以两个 74LS90 构成计数周期为 100 (0~99) 的计数器为例，两个计数器的计数周期均为 10，再级联即可。计数器输出为 4 位十进制 BCD 码，数值比较器采用两片 74LS85，单片机通过扩展的 8255 并行接口的 PA 口向两个比较器送比较数值，其中 PA0~PA3 送低 4 位比较器；PA4~PA7 送高 4 位比较器。当计数值小于单片机所送数值时，输出为 1，否则为 0，这样 BCD 码的设置范围为 1~99，可以实现 1%~99% 的占空比调节，具体电路如图 4.7 所示。

3. 脉冲波到正弦波变换

波形变换可采用传统的方波→三角波→正弦波的函数波形变换电路，其组成框图如图 4.8 所示。此方案的不足之处在于：大量使用分立器件，电路复杂，且输出信号失真度较大；受积分电路阻容元件的限制，频率覆盖范围低，不能达到指标要求；需要级联调节，电路的稳定度和精确度较差。

另一选择是，采用集成有源滤波器芯片实现波形变换，使用时截止频率设定为脉冲波基波频率，滤除方波中的谐波分量，得到同频正弦波。考虑到滤波器截止频率必须跟踪输入信号频率，选用截止频率可变的集成有源低通滤波芯片 MAX292，这样，对滤波器截止频率的控制变成简单的对输入时钟信号的控制，操作方便，滤波效果好。MAX292 为 8 阶贝塞尔型低通滤波器。截止频率 f_c 受输入时钟信号 CLK 控制，满足 $f_1 = f_{\text{clk}}/100$ ， f_1 的范围为 0.1 Hz~25 kHz，满足设计要求。脉冲波到正弦波的

变换电路如图4.9所示，当采用有源集成低通滤波器 MAX292 时，外部输入时钟信号 CLK 由锁相环倍频信号 $f_c = 100 \times N$ 给出。这样，低通截止频率为

$$f_1 = f_{\text{clk}} / 100 = N \text{ Hz}$$

与输入的信号频率相同，可以保证较好的滤波效果，实现脉冲波到正弦波的变换。

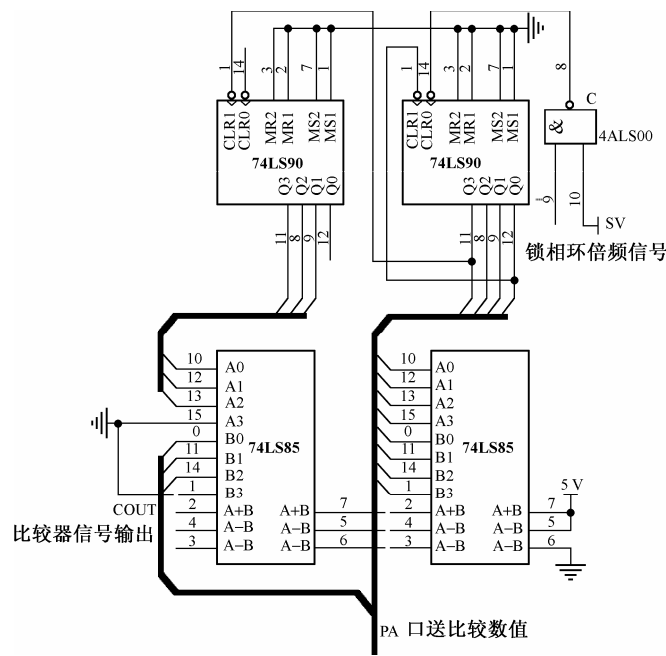


图 4.7 占空比可调电路原理图

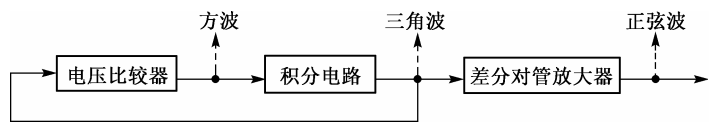


图 4.8 函数波形变换电路组成框图

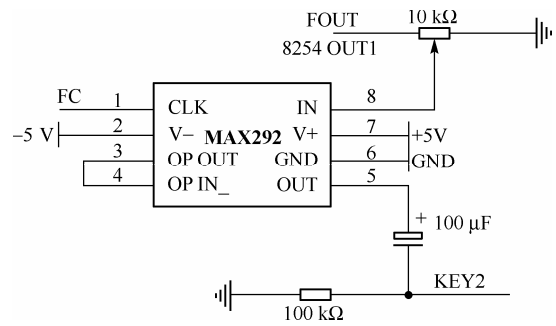


图 4.9 脉冲波到正弦波的变换电路图

4. 输出波形幅度可调电路的实现

输出波形幅度可调电路可以使用程控增益放大器，实现对输入波形幅度的调节。程控增益放大器可以直接利用输入数字量来控制放大器的信号增益。但一般的程控增益放大器的可变增益挡位有限，无法实现多挡位增益变换，而且芯片较昂贵，性价比较低。

另一选择是,利用 D/A 芯片 DAC0832 实现程控电压衰减。信号由 DAC0832 的参考电压端 V_{REF} 输入,单片机通过改变送入 DAC0832 的数据值控制输出信号的幅度,即 $V_o = (D \times V_{in})/255$, 再在其后设置增益固定的放大器,实现对信号先数控衰减再固定放大。该方案可实现多挡电压衰减,满足指标对幅度控制的要求。这一方案要求数模转换器有良好的线性特性,即对应给定的数字量输入与输出电压之间成正比关系;而且要求数模转换器有较快的转换时间,即能跟得上输入波形的变化。因此,对于低频信号的幅度控制较准确,但对较高频率的信号(≥ 24 kHz),输出波形可能产生失真。

该电路的原理框图如图4.10所示, DAC0832 工作在单缓冲方式。先调节可调衰减器,使 DAC0832 在输入数字量 D 为 200(十进制数)时,输出信号幅度 V_{out} 为 2 V,经过计算,此时输入 DAC 的信号幅度 V_{in} 为 2.55 V。程控衰减数字量与输出信号幅度的对应关系为

$$V_{out} = 10 \times D \text{ mV} \quad 0 \leq D \leq 200 \quad (4.3)$$

再经过 2 倍的固定增益放大器,得到

$$V_o = 2 \times V_{out} = 20 \times D \text{ mV} \quad (4.4)$$

对应幅度调节最小步进为

$$\Delta V_o = 20 \times 1 \text{ mV} = 0.02 \text{ V}$$

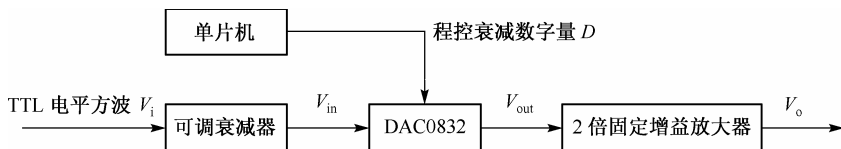


图 4.10 输出波形幅度可调电路的原理框图

DAC0832 为 8 位 D/A 转换器,电流建立时间为 $1 \mu\text{s}$ 。满量程为 5 V 的量化误差为 $\pm(1/2) \times (1/2^8) \times 5 \text{ V} = \pm 10 \text{ mV}$,按满度归一化的相对误差为 $\pm(1/2) \times (1/2^8) = \pm 0.2\%$ 。由以上分析可看出,设计满足信号垂直幅度分辨率的要求。输出波形幅度可调电路原理图如图4.11所示。

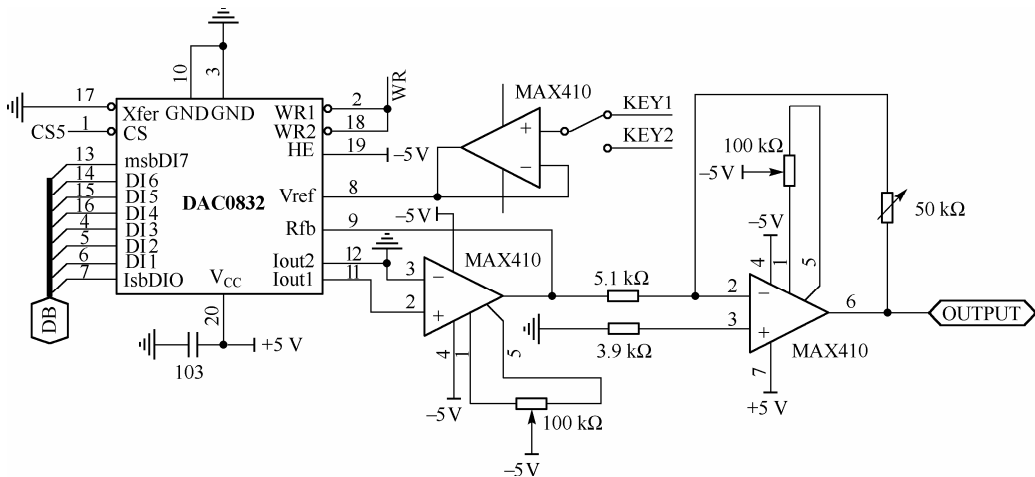


图 4.11 输出波形幅度可调电路原理图

5. 键盘参考设计

键盘设计应包括数字键和功能选择键，以完成波形选择、频率控制、幅度控制和占空比控制，表 4.1 所示为参考键盘定义与按键功能。

表 4.1 参考键盘定义与按键功能

键 盘 定 义	按 键 功 能	键 盘 定 义	按 键 功 能
0	数字 0	空键 1	设定并显示频率
1	数字 1	空键 2	设定并显示幅度
2	数字 2	空键 3	设定并显示占空比
3	数字 3	Shift	正弦波/脉冲波选择
4	数字 4	Enter	设定后确认
5	数字 5	Up	数值增，波形切换
6	数字 6	Down	数值减，波形切换
7	数字 7	Clear	清屏
8	数字 8	—	—
9	数字 9	—	—

系统自举以后，输出信号频率预置为 1 kHz，占空比预置为 50%，电压幅度预置为 1 V。按 Shift 键为正弦波和脉冲波输出选择，按空键 1, 2, 3 可以设定并显示波形频率、幅度、占空比，数字键 0~9 对应输出信号频率及幅度设置，按 Enter 键确认。Up 和 Down 键实现输出信号频率的步进增减和波形切换功能，按下 Clear 键清屏。

4.3.3 系统软件设计

软件设计应采用模块化的设计方法，主要分为初始化、波形选择、频率控制、幅度控制、占空比控制、输出显示等模块，初始化模块的设计细节在键盘设计中已有说明。系统以单片机作为控制中心，完成初始化、键盘管理、显示控制、对 8254 编程以实现频率控制、占空比预置及实现幅度控制等功能。系统主程序的参考流程图如图 4.12 所示。

4.3.4 系统调试

调试过程是发现错误、改正错误的过程，能够越早地发现错误，纠正它所付出的代价就越小。错误可以分为系统方案错误、系统设计错误和系统实现错误等。系统方案错误是致命的，为避免方案错误，在设计前要进行周密的方案论证。系统设计错误通常意味着要对设计方案做较大的改动。例如，器件选择错误，在高速应用中使用了低速器件；元件参数计算错误以致达不到预定的指标，这些都是设计上的错误。例如本设计中用到低通滤波器，要验证滤波器的特性，可以先使用计算机对电路进行模拟，模拟的结果为调试提供参考依据。系统实现错误主要是器件接线错误或工作点设置错误。查找实现错误时可以根据模拟的结果进行对照调试，或由电路的因果关系确定故障的位置。

根据方案设计的要求，调试过程包括硬件调试、软件调试和软硬件联调。电路按模块调试，各模块(包括锁相环频率合成模块、脉冲波转换为正弦波模块等)逐个调试通过后再联调。软件调试采取自下而上的调试方法，即单独调试好每一个模块，然后再连接成一个完整的系统调试。

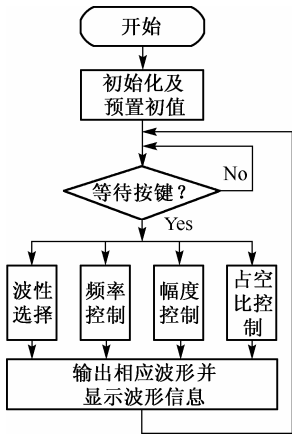


图 4.12 系统主程序参考流程图

4.4 基于DDS的数控信源的设计

1. 设计任务

在给定 $\pm 15\text{ V}$ 电源电压的条件下,设计并制作一个基于DDS的正弦波和脉冲波信号源。

2. 设计基本要求

(1) 正弦波信号源

① 信号频率: $20\text{ Hz}\sim 20\text{ kHz}$ 步进调整,步长为 5 Hz 。

② 频率稳定度: 优于 10^{-4} 。

③ 非线性失真系数: $\leq 3\%$ 。

(2) 脉冲波信号源

① 信号频率: $20\text{ Hz}\sim 20\text{ kHz}$ 步进调整,步长为 5 Hz 。

② 上升时间和下降时间: $\leq 1\text{ }\mu\text{s}$ 。

③ 平顶斜降: $\leq 5\%$ 。

④ 脉冲占空比: $2\%\sim 98\%$ 步进可调,步长为 2% 。

(3) 上述两个信号源的公共要求

① 频率可预置。

② 在负载为 $600\text{ }\Omega$ 时,输出幅度为 3 V 。

③ 完成 5 位输出频率的数字显示。

3. 设计提高部分要求

(1) 正弦波和脉冲波的频率分辨率为 1 Hz , 正弦波信号频率为 $20\text{ Hz}\sim 200\text{ kHz}$ 。

(2) 正弦波和脉冲波的幅度可步进调整,调整范围为 $100\text{ mV}\sim 3\text{ V}$,步长为 100 mV 。

(3) 正弦波和脉冲波的频率可自动步进,步长为 1 Hz 。

(4) 降低正弦波非线性失真系数。

4.4.1 DDS的相关分析

DDS 是一种应用数字技术产生信号波形的方法。DDS 技术建立在采样定理的基础上,首先对需要产生的信号波形进行采样和量化,然后存入存储器作为待产生信号波形的数据表。输出信号波形时,电路在一个高稳定时钟信号的控制下从数据表中依次读出信号波形的数据,产生数字化的信号,这个信号再通过 DAC 转换成所需的模拟信号波形,经过低通滤波器滤除不需要的采样频率分量,即得到频谱纯净的输出信号。

1. 输出信号的数字化描述

DDS 为了输出特定频率信号,相应的相位增量必须输入到频率寄存器中,输出频率 f_{out} 与相位增量 $\Delta\phi$ 关系如下:

$$f_{\text{out}} = \frac{f_{\text{clk}} \Delta\phi}{2^n} \quad (n \text{ 为相位累加器位数}) \quad (4.6)$$

$\Delta\phi$ 的预置值用 K 表示,则上式变为

$$f_{\text{out}} = \frac{f_{\text{clk}} K}{2^n} \quad (4.7)$$

也就是说, DDS 的输出频率由频率控制字 K 来决定, 使用这个公式, 所生成信号可达到非常微小的频率步进量 ($10^{-6} \sim 10^{-3} \text{ Hz}$ 量级)。可以看出, 在相位分辨率范围内, 通过对相位增量的编程可产生任何频率, 即输出信号可用数字化方式来描述。

2. 相位频率带宽

式 (4.7) 为 DDS 的输出频率关系式, 当 $K=1$ 时, DDS 的最低输出频率为

$$f_{\text{min}} = f_{\text{clk}} / M \quad M = 2^n \quad (4.8)$$

式中, n 为相位累加器的字长。当 n 很大时, 最低输出频率达 Hz, mHz 直至 μHz 量级, 可以认为 DDS 的最低合成频率接近于零频。

根据采样定理, DDS 的最高输出频率应小于 $f_{\text{clk}}/2$, 即

$$f_{\text{max}} \leq \frac{1}{2} f_{\text{clk}} \quad (4.9)$$

在实际应用中, 考虑到输出滤波器的特性不可能是理想的, 所以一般限制最大输出频率为

$$f_{\text{max}} / f_{\text{min}} = M \times 40\% = 2^n \times 40\% \quad (4.10)$$

若取 $n=32$, 则 $f_{\text{max}} / f_{\text{min}} = 2^{32} \times 40\% = 1.7 \times 10^9$, 如此之大的相位频率带宽是传统的频率合成技术所无法相比的。

3. 频率分辨率

DDS 的最小频率步进量就是它的最低输出频率, 即

$$\Delta f = f_{\text{min}} = f_{\text{clk}} / 2^n \quad (4.11)$$

可见, 只要相位累加器有足够的字长, 就可以实现非常精细的频率分辨率, 输出频率已十分接近连续变化。

4. 频率转换时间

DDS 的相位序列在时间上是离散的, 当频率控制字改变后, 只经过一个时钟周期之后就能按新的相位增量累加。所以频率转换的时间主要取决于外部频率控制字的传送时间, 即接口速率。一般集成 DDS 产品的频率转换时间可达 10 ns 量级。

5. 相位连续性

在实际应用中, 对频率合成器的相位连续性的要求根据应用场合的不同可能会有很大不同, 如在通信中一般要求载波信号的相位不能发生跃变。从 DDS 的原理可知, 在改变输出频率时, 实际上改变的是每次的相位增量, 即改变相位函数的增长速率。其相位函数曲线是连续的, 只是在改变频率的瞬间其斜率发生了突变, 从而保持了输出信号相位的连续性。直接频率合成 (DFS) 的相位是不连续的, 锁相合成的相位虽然是连续的, 但因为 V_{∞} 的惰性, 频率转换时间较长。

6. 任意波形输出

以上讨论都是建立在输出正弦波波形的基础之上的, 而这个正弦波的波形又完全是由 ROM 中存储的函数表所决定的。因此, 如果在 ROM 中存入其他形式的函数波形表, 在一个周期内给出按

一定的电压幅度变化规律组成的波形, DDS 即可输出相应的周期波形。对于任何周期性波形, 只要能满足采样定理, 都可以方便地通过更新 ROM 中的数据用 DDS 来实现, 这种方法又称为波形合成法。

4.4.2 系统设计分析与理论计算

基于 DDS 的数控信源设计, 其系统主要由以下单元电路组成: (1) 频率可任意设置的正弦、脉冲波形的产生电路。(2) 脉冲波占空比可任意设置的电路。(3) 输出波形幅度可调电路。(4) DAC 及后向平滑滤波器电路。其中, 如何根据 DDS 原理和系统设计要求建立数学模型并选择合适的参数是本设计的关键。

1. 系统时钟频率选择与相位累加器的设计

已知 DDS 的输出频率 f_{out} 与相位增量 $\Delta\phi$ 关系为 $f_{\text{out}} = f_{\text{clk}} \times \Delta\phi / 2^n$ (n 为相位累加器位数), $\Delta\phi$ 的预置值用 K 表示, 则上述关系式可变为

$$f_{\text{out}} = f_{\text{clk}} \times K / 2^n, \quad \Delta f_{\text{out}} = f_{\text{clk}} / 2^n \quad (4.12)$$

式中, $K = f_{\text{out}} \times 2^n / f_{\text{clk}}$, n 为累加器的字长, Δf_{out} 为 DDS 的输出信号频率分辨率, 也就是最小频率步进量。DDS 的输出频率由频率控制字 K 来决定, K 的取值对应于输出频率的范围, 频率分辨率由系统时钟频率和累加器的字长决定。

理论上, 为了追求较高的指标, f_{clk} 可以取很高的频率, 相位累加器的位数也可以取很多位, 但考虑到设计指标的要求及常用的 DAC 的转换速率一般不高于 10 MHz, 故 f_{clk} 不能取得太高。如何选取时钟频率、累加器字长及频率控制字的取值是系统设计的关键。数字频率合成可以看成是一个采样过程, 应首先确定采样频率的取值。4.4.1 节在相位频率带宽的分析中已说明, 根据采样定理, DDS 的最高输出频率应不大于 $f_{\text{clk}}/2$, 即

$$f_{\text{max}} \leq \frac{1}{2} f_{\text{clk}} \quad (4.13)$$

在实际应用中, 考虑到现实世界中被采样的信号不可能是理想的带限信号, 而且输出滤波器的特性也不可能是理想的, 所以一般限制最大输出频率为

$$f_{\text{max}} \leq f_{\text{clk}} \times 40\% \quad (4.14)$$

以上的讨论是以采样定理为准则而展开的, 而在工程中常采用 5~10 倍的待采信号最高频率分量作为采样频率, 以满足要求。考虑到为了保证输出正弦信号为 20 Hz~200 kHz, 使整个频段内信号失真度满足题目要求, 每个正弦周期内取样点数应不少于 32 点。关键在于最高输出频率信号应满足要求, 即输出 200 kHz 时保证有 32 个取样点, 那么时钟信号频率必须大于 $200 \text{ kHz} \times 32 = 6.4 \text{ MHz}$ 。综合考虑, 本系统时钟频率 f_{clk} 选取 8.388 608 MHz, 对应相位累加器的位数 n 取 23 位。这样 $\Delta f_{\text{out}} = 1 \text{ Hz}$, $f_{\text{out}} = K$ 。频率控制字 K 的位数为 18 位 ($2^{18} = 262 \ 144 \text{ Hz}$), 此时在满足每周期取样点数不少于 32 点的前提下, 输出频率可以达到 262 kHz, 确保输出频率范围覆盖 1 Hz~200 kHz。

2. 脉冲波占空比可任意设置的电路设计

结合 DDS 相位累加原理, 对一个周期内的相位累加次数进行计数(高 A 位每变化一次计数值加 1), 利用此计数值和预先设定数值进行比较, 得到占空比可调脉冲波。具体方案示意图如图 4.13 所示。

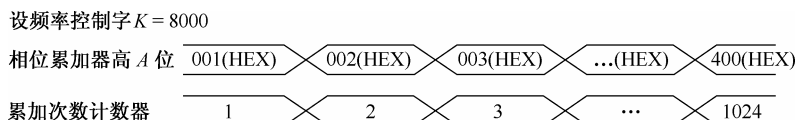


图 4.13 累加计数实现占空比可调示意图

如图4.13所示,一个周期内相位累加器的高 A 位每改变一次,累加次数计数器就加 1,累加次数计数器(以下简称累加计数器)的最大值 $\leq 2^A$,假定脉冲信号范围是 $1\text{ Hz} \sim 20\text{ kHz}$,设 $A=10$,那么累加计数器在一个周期内对方波信号最多计数 1024 次,分析频率控制字 K ,设定不同的比较数值,就可以实现全频段范围内脉冲波的占空比调节,计算方法如下:

f_{clk} 选取 $8.388\,608\text{ MHz}$,相位累加器长度 n 为 23 位, $f_{\text{out}} = f_{\text{clk}} \times K / 2^n = K$,全频段范围内,累加计数器在一个周期内的计数值 M 为

$$M = \begin{cases} 1024 & (K \leq 2^{n-A}) \\ 2^n / K & (K > 2^{n-A}) \end{cases} \quad (4.15)$$

由上式可知,脉冲波频率在 $1\text{ Hz} \sim 8.192\text{ kHz}$ 范围内时,累加计数器在一个周期内的计数值 M 为 1024;脉冲波频率在 $8.192 \sim 20\text{ kHz}$ 范围内时,累加计数器在一个周期内的计数值 M 随着 K 的增大而减少;脉冲波频率为 20 kHz 时, K 为 20 000,累加计数器在一个周期内的计数值 M 等于 420,脉冲波占空比步进调节为 $1/420$,远远满足不大于 1% 的指标要求。

此方案的优点是:

- 抓住了脉冲波的特点,充分利用相位累加器提供的累加信息,无须额外设置硬件电路,工作可靠;
- 占空比调节全部数字化,只需要经过简单的运算就可以给出控制量。

此方案的不足之处是:

- 不是模 50 或者模 100 计数,因此占空比调节有误差,而且误差随着输入信号频率的增大而增大,因此不是等精度调节,但整个频段内误差可控且满足不大于 1% 的指标要求;
- 占空比调节与系统时钟频率有关,只有提高系统时钟频率才能得到更精细的占空比调节。

3. E²PROM 存储波形数据

设计中为了实现逼真的输出波形,对应于正弦信号的一个周期,等相位间隔取 1024 个采样点依次存储于 E²PROM 中,也就是说,把正弦输出波形的一个完整周期采样幅值按相位步进顺序分别量化并存储于 E²PROM 中。

4. D/A 的设计

(1) 数据位宽度及匹配

由于采样点数为 1024 点,因此 D/A 的位数最佳是 10 位,即相位量化和 D/A 的幅度量化能一一对应,完成对等的 1024 个点的幅度量化。问题是 ROM 为字节寻址,即 ROM 输出的数据是 8 位的,如何用 8 位数据输出的 ROM 来控制 10 位输入的 D/A? 解决的方法为:在 ROM 中偶地址存入样本数据的低八位(对应于 10 位 D/A 的低八位),奇地址存入样本数据的高两位(对应于 D/A 的高两位)。用相位累加器的高 11 位地址控制 ROM 的寻址和输出,同时用相位累加器的高十位控制 D/A 的输出。那么相当于高十位地址发生变化时,ROM 已将其对应的低八位和高两位样本数据分时写入了 D/A 的 10 位输入锁存器,并输出高 11 位地址对应的幅值,10 bit 数据采用复用模式(8+2 位)。

采用 10 位 D/A 的设计可以保证输出信号有较高质量,但在硬件设计上过于复杂;而且相位量化和 D/A 的幅度量化是两个不同的过程,并非一定要一一对应。一种较为便利的方案是采用数据位宽度匹配的 8 位 D/A,在设计中采用在幅度上仅分为 256 个点,而在相位上仍然分为 1024 个点。这时 D/A 的控制比较简单,可以工作在直通方式。

(2) 输出波形幅度可调电路的设计

设计要求正弦波和脉冲波幅度可步进调整,要求输出波形调整范围为 100 mV~3 V,步长为 100 mV。

为实现幅度可调,可以利用常用芯片 DAC0832 实现程控电压衰减,波形信号由 DAC0832 的参考电压端输入,单片机通过改变送入 DAC0832 的数据值控制输出信号的幅度。实现方法已在 4.3.2 节做出说明,这里不再赘述。

另一选择是,采用双 D/A 设计来实现幅度可调,第一级 DAC 输出作为 DDS 系统中第二级 D/A 转换的参考电压,以此控制信号发生器的输出电压。因此考虑使用 DAC0832(第一级 DAC)作为幅度控制器,由单片机控制对 DAC0832 置数,得到一个在 0~5 V 内可以程控变化的直流电平,将此直流电平作为 DDS 系统中实现 D/A 转换的 DAC0800(第二级 DAC)的参考电平,由此实现幅度可调的功能。这一方法和单片 DAC 实现幅度控制的方法在原理上是相同的,都是利用改变波形生成子系统中 D/A 转换器基准电压 V_{REF} 的方法去改变波形的输出幅度。信号电压峰-峰值步进幅度为 $5/255 = 0.02$ V,足以满足指标要求。该方案电路较简单,控制方便,可以满足题目对于幅度步进调节和分辨率的要求,具体的实现电路在后续的系统整体设计一节给出。

5. DAC后向平滑滤波器的设计

DDS 输出的数字化的正弦波可用 DAC 转换为连续波形,DAC 输出波形中的主要成分是所需频率的正弦波,但也包括高频镜像及谐波分量。DAC 的非线性特性对 DDS 输出数据的影响表现为产生输出频率的谐波分量及这些谐波分量的镜像分量,即含有频率为 $f_m = If_c \pm nf_n$ ($I = 0, \pm 1, \pm 2, L$; $n = 2, 3, L$) 的分量,如图 4.14 所示。

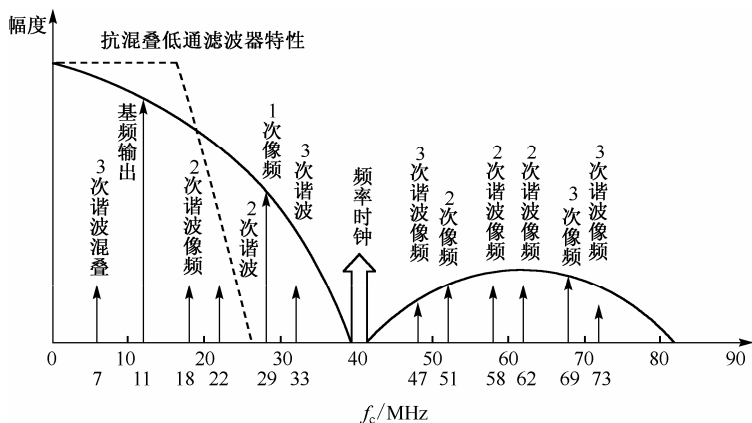


图 4.14 DAC 非线性对 DDS 输出的影响

DAC 后向滤波电路的作用是平滑信号,对于输出正弦波,需要滤除高频噪声。设计要求滤波器能尽可能抑制谐波和高频噪声,由于信号发生器的输出波形中的主要噪声为 D/A 转换产生的高频分量,与设计的频率范围相差很远,所以相对来说,滤波器在频带内的平坦程度比其衰减陡峭度更为重要。常见的模拟滤波器有巴特沃斯(Butterworth)、切比雪夫(Chebyshev)及椭圆滤波器。巴特沃斯低通滤波

器的特点是通带内部有最大平坦的振幅特性,而且随着频率的升高而单调地下降,在截止频率处振幅的衰减为 3 dB,阶数越高,过渡带越陡,通带和阻带与理想低通的近似性越好。综上考虑,可采用集成有源滤波器设计截止频率约为 250 kHz 的二阶巴特沃斯有源低通滤波器。

4.4.3 系统整体设计

本系统的实现可以通过选用 DDS 集成芯片,通过控制模块完成对它的频率控制字和控制信号的置入,输出模块完成滤波和放大功能,使输出符合信号源的设计要求,但 DDS 集成芯片成本较高,而且控制复杂。

基于训练的要求,系统设计采用 CPLD 实现 DDS 技术中累加器及占空比调节等功能,单片机构成系统的控制中心,实现控制功能选择和结果显示, CPLD 在单片机的控制下实现 DDS 的功能。系统初始化后对按键进行判断,从而根据设定选择相应的功能并执行。在更改输出波形频率时,根据输入产生相应的频率控制字送到 CPLD 中,控制输出频率值。CPLD 的输出作为 ROM 中波形表的读取地址,选取波形数据经 DAC0832 输出模拟波形。同时,单片机控制第一片 DAC0832 输出直流电压作为输出波形的 DAC0800 的基准源,控制输出波形的幅值。系统整体设计框图如图 4.15 所示。

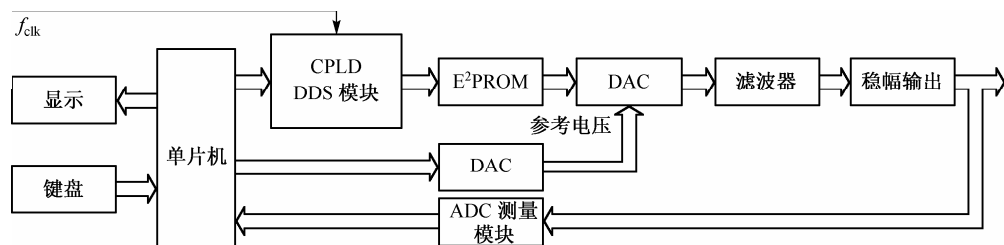


图 4.15 系统整体设计框图

4.4.4 系统硬件设计

1. 系统时钟产生电路模块

为了提高 DDS 的频谱纯度,必须使用高稳定度的时钟信号。可以直接选取晶振频率为 8.388 608 MHz 的晶振,或者采用锁相环(PLL)倍频电路,使用 4.194 304 MHz 晶振构造起振电路,通过锁相环电路对晶振信号 2 倍频,得到高精度的 8.388 608 MHz 时钟信号,具体电路如图 4.16 所示。

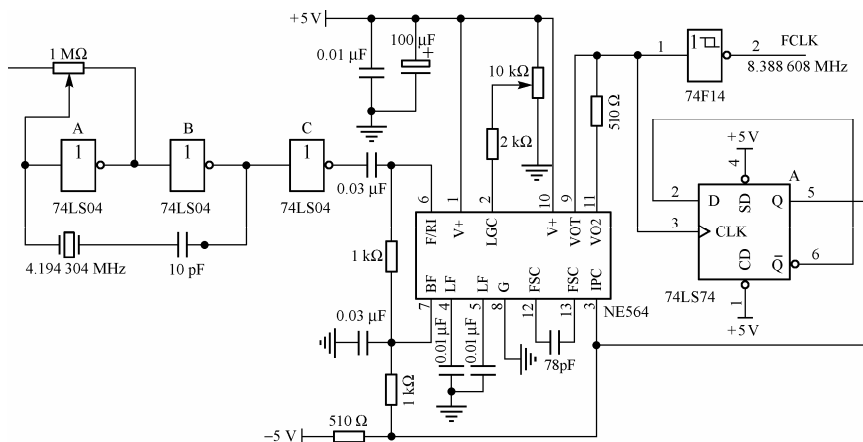


图 4.16 系统时钟产生电路

2. CPLD控制电路模块

由 CPLD 实现 DDS 发生器、占空比调节等功能。

DDS 发生器由频率控制字寄存器和相位累加寄存器组成,通过单片机向频率控制字寄存器送控制字 K , 相位累加器在系统时钟的控制下,对控制字 K 和相位寄存的值进行累加。

占空比调节模块由累加次数计数器、比较数值寄存器和三态数据输出寄存器组成。累加次数计数器对相位累加器高 10 位累加次数进行计数；比较数值寄存器存储单片机送来的比较数值，用于决定占空比的大小；三态数据输出口在输出占空比可调脉冲波时与 DAC 数据口相连，传送信号，否则处于高阻态。CPLD 控制电路如图 4.17 所示。

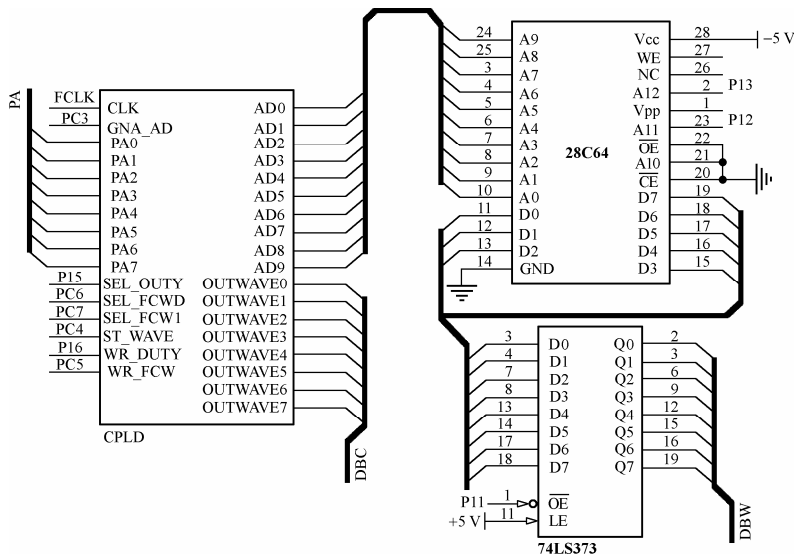


图 4.17 CPLD 控制电路

3. 只读存储器模块

设计使用 8K×8 E²PROM(28C64) 存储波形数据, 一共存储 4 幅波形图, 分别是方波、正弦波、三角波和锯齿波。每幅波形图均匀存储 1024 个样值。存储方式如图 4.18 所示。

只读存储器工作在读方式下, 存储器的地址线的高三位选择波形, 低十位地址线接相位累加器的高十位, 存储器的数据口通过三态门接到 DAC 的数据线上。在读方式下, 只要相位累加器的高十位给出地址, 数据线上就呈现寻址单元的存储数据。

脉冲波存储单元	地址高三位为 000
正弦波存储单元	地址高三位为 010
三角波存储单元	地址高三位为 100
锯齿波存储单元	地址高三位为 110

图 4.18 E²PROM (28C64) 内部存储情况

4. 双D/A式幅度调节电路模块

双 D/A 式幅度调节电路原理如前所述。当选择幅度控制时, 利用模拟开关将 DAC0800 的 V_{REF} 端接至 DAC0832 构成的幅度控制电路的输出可调信号端即可 (0~5 V 可编程变化的直流电平), 具体电路如图 4.19 所示。

5. D/A转换及后级滤波电路模块

选择高速数模转换器 DAC0800 作为 D/A 转换芯片, 它的转换时间为 100 ns, 满足指标最大转换速率的要求。后级滤波采用截止频率为 250 kHz 的二阶 Butterworth 低通滤波器。并设计模拟开关: 只

对正弦波进行滤波,当输出信号为方波、脉冲波时,跳过滤波器。由于可采用的有源开关电容滤波器的种类较多,后级滤波电路的具体电路不再给出,D/A 转换电路如图4.20所示。

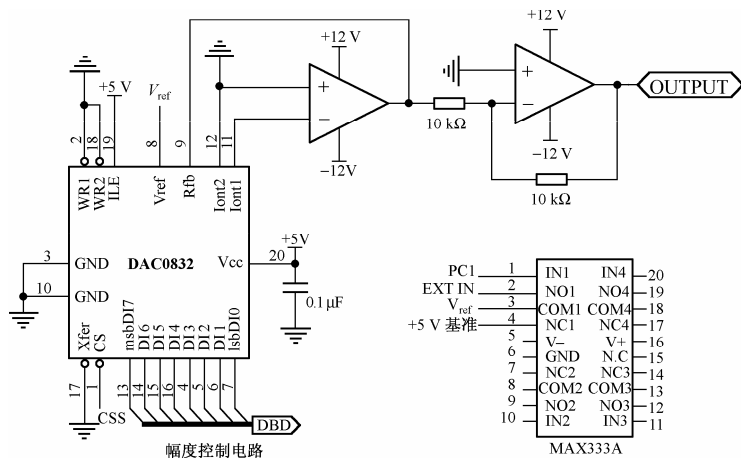


图 4.19 双 D/A 式幅度调节电路

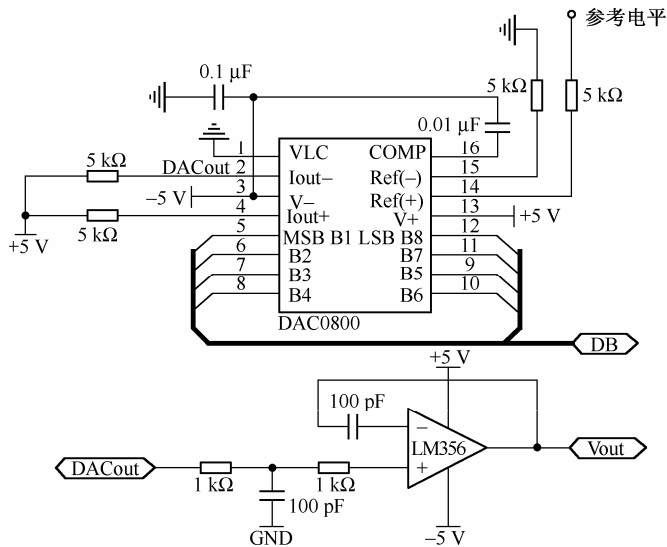


图 4.20 D/A 转换电路

6. A/D测量信号峰-峰值电路模块

为了保证幅度控制的准确性,系统设计可以考虑增设一个 A/D 测量模块,用于对输出的信号峰-峰值进行实时测量,测量结果送单片机分析,以决定送幅度控制器的数字量。具体方案是:使用 A/D 转换器 MAX197,读回 D/A 转换输出波形的峰-峰值,量化数据反馈至单片机分析,ADC、DAC 和单片机一起构成闭环测量控制电路,不间断地调整幅度控制量,保证输出幅度控制准确。具体电路如图 4.21 所示。

4.4.5 系统软件设计

基于 DDS 的数控信源的设计和 4.3 节(基于 PLL 的数控信源的设计)在最终实现功能上基本相同,因此系统主程序流程也是相同的。但由于频率合成器采用的方法不同,控制对象不同,所以在系统软件设计的具体细节上是不同的。

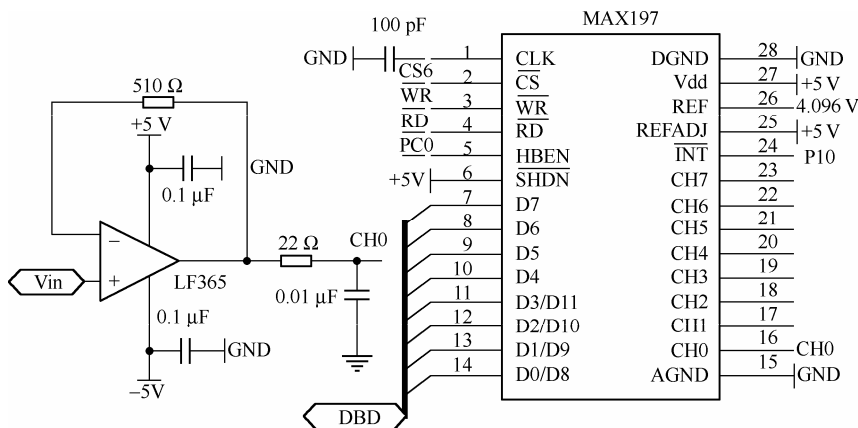


图 4.21 A/D 测量信号峰-峰值电路

在系统软件设计中，单片机是系统的控制核心，它主要实现以下功能：一方面控制 LCD 显示输入控制信息，控制按键识别和功能选择；另一方面与 CPLD 通信，实现直接数字频率合成。

单片机在等待按键过程中输出信号频率预置为 1 kHz，占空比预置为 50%，电压幅度预置在 1 V。按下按键后根据不同的按键实现对应的功能。软件设计 4 种工作状态，分别是波形选择、频率控制、幅度控制和占空比控制。根据设置的值对 CPLD 内部实现控制和功能选择，并配合 CPLD 对 E²PROM 存储数据的读取，另外单片机还和两片 D/A 相连，实现对输出信号幅度的控制。

4.4.6 噪声分析和降噪措施

1. DDS的杂散模型

DDS 具有许多优点，其应用十分广泛。但同基于 PLL 的频率合成相比，DDS 的输出带宽和杂散是限制 DDS 技术应用的两个主要因素。输出频率带宽受限于时钟信号的频率和 DAC 的带宽，而杂散是由全数字结构的 DDS 的自身特点所决定的，如图 4.22 所示。

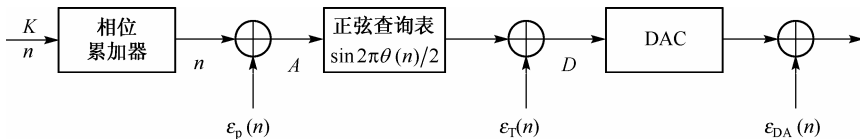


图 4.22 DDS 的杂散模型

图中， K 为频率控制字； n 为相位累加器的位数； A 为经过相位截断后的位数； D 为 D/A 转换器的位数； $\varepsilon_p(n)$ 为相位截断噪声； $\varepsilon_T(n)$ 为幅度量化噪声； $\varepsilon_{DA}(n)$ 为 DAC 采样及非线性产生的噪声。

由 DDS 的杂散模型可以看出，DDS 的杂散有 5 个来源：(1) 相位截断效应；(2) 周期性相位累积效应；(3) 幅度量化噪声；(4) DAC 的非理想特性；(5) 其他噪声和干扰。

(1) 相位截断效应

当输出频率不是参考时钟的整约数时，存在这种误差。这是由于为了保证分辨率，相位累加器的位数 n 取得较大，但受体积和成本限制 ROM 的容量较小，远小于 2^n 。相位累加器输出的低位被舍弃，从而引入误差。

一般来说，当相位累加器的位数 n 与可查找的正弦相一幅表的地址数相等，同时满足采样定理时，理论上不存在相位误差，因此可以保证输出频谱的纯度。但实际应用中为了减小存储容量，正弦相一幅表的尺寸不可能做得太大，而为了保证 D/A 转换恢复信号的质量，累加器的相位分辨率又不能太低。

通常 ROM 的容量不是 $2^n \times b$ ，而是 $2^m \times b$ ($m < n$)，其中 b 为数据位。实际相位量化单位为 $\Delta\phi = 2\pi/2^m > 2\pi/2^n$ ，这就是所谓的截短 ROM 地址。

实际的 DDS 系统，一般只取累加器相位序列的高 A 位来寻址 ROM，而舍弃 B 位 ($B + A = n$)，因而产生误差，引起输出信噪比有所下降。

有关分析如下： n 位相位字长的 DDS 在零初始相位条件下，其相位累加器的输出量化相位序列为

$$\Phi(m) = mK \bmod 2^n \quad m = 0, 1, 2, L \quad (4.16)$$

式中， m 表示相位累加器的第 m 次累加； K 表示频率控制字； $\bmod$ 表示模除运算。对任意两个整数 p 和 q ， $\bmod$ 运算定义为

$$p \bmod q = p - \text{int}(p/q)q \quad (4.17)$$

式中， $\text{int}(\cdot)$ 代表取整运算。

假定只取相位累加器的高 A 位而将低 B 位舍弃，那么就得到一个相位截断的 DDS，截断部分有 B 位，可以视为一个 B 位的累加器，截断后的量化相位序列为

$$\Phi_t(m) = \Phi(m) - [\Phi(m) \bmod 2^B] = mK \bmod 2^n - mK \bmod 2^B \quad (4.18)$$

量化相位误差序列为

$$\varepsilon(m) = \Phi(m) - \Phi_t(m) = mK \bmod 2^B = mU \bmod 2^B \quad (4.19)$$

式中， $U = K \bmod 2^B$ 是 K 的低 B 位。

当 $U = 0$ 时， $\varepsilon(m) = 0$ ，此时 $f_0 = K \frac{f_c}{2^A}$ ($K = 1, 2, 3, L$)。这些频率称为 DDS 的输出主频率，其输出频谱与在相同时钟驱动下输出相同频率的无相位截断的 DDS 的输出频谱特性相一致。

由于相位截断，DDS 可以产生相邻主频率之间的那些频率，但同时引入杂散，也就是 $U \neq 0$ 的情形。可以导出 $\varepsilon(m)$ 是一周期序列，其周期为

$$T_p = \frac{2^B}{\text{GCD}(2^B, K)} \quad (4.20)$$

式中， $\text{GCD}(x, y)$ 表示取 x 和 y 的最大公约数。相位截断产生的杂散表现为对输出频率的相位调制 (PM)，对于不同频率的输出信号，调制信号是不同的周期锯齿波，峰-峰值为 $2\pi/2P$ ， P 为有效的相位累加器位数 (一般高于 20 位)。

(2) 周期性相位累积效应

由以上分析可看出，DDS 相位累加器的输出数字相位有严格的周期性。由于波形存储器从相位到幅度是一一映射的，因而 DDS 的输出幅度也具有与数字相位相同的周期性。输出波形的重复周期 μ 为

$$\mu = \frac{2^n}{\text{GCD}(2^n, K)} \quad (4.21)$$

该周期性的累积效应也导致了相位截断及幅度量化误差的周期性。

(3) 幅度量化噪声

由于 DDS 的 ROM 中存储的波形样点值由有限位二进制数表示，因此引入了幅度量化误差，设 D 是样点值的二进制位数 (一般也是 D/A 变换器的位数)，则量化后的信噪比为 $(1.8+6D)$ dB，也就是 ROM

的字长每增加 1 bit, 量化信噪比增加 6 dB; 幅度量化误差产生的杂散表现为对输出频率的幅度调制 (AM)。

(4) DAC 的非理想特性

前面已作过相关讨论, DDS 输出的数字化正弦波可用 DAC 转换为连续波形。DAC 输出波形中的主要成分是所需频率的正弦波, 但也包括高频镜像及谐波分量。

在高频、超高频 DDS 系统中, DAC 的非线性成为 DDS 输出杂散的主要来源之一。DAC 的非线性特性包括差分、积分非线性、DAC 转换过程中的尖峰电流、转换速率受限等特性, 各杂散信号的频率和幅度依赖于生成的频率和时钟频率的比值, 以及输出波形与采样时钟的相位关系, 同时又依赖于 DAC 和 LPF 的设计及印制电路板的特性和时钟频率的相位噪声。

DAC 的非线性特性对 DDS 输出数据的影响表现为产生输出频率的谐波分量及这些谐波分量的镜像分量, 即含有频率为 $f_m = If_c \pm nf_n$ ($I = 0, \pm 1, \pm 2, L$; $n = 2, 3, L$) 的分量。

可采用低通滤波器去除镜像分量, 以使波形达到所需要的指标。根据采样定理, LPF 的带宽 $\leq 1/2$ 采样频率。DDS 能产生的输出频率是 $0 \sim 1/2$ 时钟频率, 然而 LPF 的截止斜坡决定了实际输出频率上限约为时钟频率的 40%。同时为了消除镜像混叠, 应认真选择时钟频率, 才能有效抑制镜像混叠, 例如, 若选择 $f_{\text{clk}} > (n+1)f_{\text{max}}$, 可抑制 n 阶镜像混叠, 其中 n 为镜像混叠阶数, f_{max} 为通带的上限频率。

(5) 其他噪声和干扰

除杂散外, DDS 存在其他噪声和干扰。由于时钟的精度通过 DDS 系统传递, DDS 系统的输入时钟是系统的主要相位噪声来源。除此之外, 噪声还和其他因素 (如数字电路的触发噪声) 有关, 各控制端和电源引入的噪声和干扰都会使 DDS 输出信号的相位噪声恶化, 所以实际系统的指标和理论值是有差距的。因此, 电路结构、设计工艺、印刷板的规划设计, 对限制频率合成器的噪声特别重要。

2. 降噪措施

为了克服各控制端及电源引入的噪声与干扰, 提高 DDS 的频谱纯度, 可采取如下措施:

(1) 使用高稳定度的时钟信号。采用石英晶体振荡器作为主振源, 通过放大、限幅来得到 TTL 电平的时钟信号, 或者通过高性能的高速比较器来得到 TTL 电平的时钟信号。更好的设计方案是采用锁相环 (PLL) 电路获得高稳定度、高精度的时钟信号, 因为由分频器的相位噪声性能可知: 分频输出的相位噪声将比其输入信号低 20lgM dB, 使相位噪声有较大改善。

(2) 对电源良好的退耦以保持低噪声。电源及地的电平抖动限制在低纹波范围内对使用 DDS 来获得较好的结果是十分关键的, 特别是当 DDS 工作在最大极限频率时。电源线及地线应尽可能地短粗 (因本训练不可能制作印制电路板, 实际应用中印制电路板应独立定制电源层与地线层)。应注意电源电路的退耦, 电源开关噪声耦合入部分异步输入端, 并引起异步寄存器数据写错误或加载命令错误是有可能的, 这些错误将引发输出数据端的不稳定, 数据总线输入也将因此受到影响。

(3) 防止时钟信号的泄漏及辐射。例如, 可用同轴电缆线馈入时钟信号。

(4) 选用低相噪器件。

(5) 在不能采用 PCB 的情况下, 合理电路布局, 减小电路分布参数, 防止模拟信号与数字信号之间的串扰。

(6) 可参考文献资料, 采用插入伪随机数的办法将相位杂散分散, 降低单一杂散谱线输出幅度及使用高品质的滤波器, 滤除杂散高频分量。

4.5 任意波形发生器的设计与实现

1. 设计任务

设计一个波形发生器,要求该波形发生器能产生正弦波、方波、三角波和由用户编辑的特定形状波形。

2. 设计基本要求

- (1) 具有产生正弦波、方波、三角波三种周期性波形的功能。
- (2) 用键盘输入编辑生成上述三种波形(同周期)的线性组合波形,以及由基波及其谐波(5次以下)线性组合的波形。
- (3) 具有波形存储功能。
- (4) 输出波形的频率范围为 100 Hz~20 kHz(非正弦波频率按 10 次谐波计算);重复频率可调,频率步进间隔 ≤ 100 Hz。
- (5) 输出波形幅度范围 0~5 V(峰-峰值),可按步进 0.1 V(峰-峰值)调整。
- (6) 具有显示输出波形的类型、重复频率(周期)和幅度的功能。

3. 设计提高部分要求

- (1) 输出波形频率范围扩展至 100 Hz~200 kHz。
- (2) 用键盘或其他输入装置产生任意波形。
- (3) 增加稳幅输出功能,当负载变化时,输出电压幅度变化不大于 $\pm 3\%$ (负载电阻变化范围:100 Ω ~ ∞)。
- (4) 具有掉电存储功能,可存储掉电前用户编辑的波形和设置。
- (5) 可产生单次或多次(1000 次以下)特定波形(如产生 1 个半周期三角波输出)。
- (6) 其他(如增加频谱分析、失真度分析、频率扩展 >200 kHz、扫频输出等功能)。

4.5.1 波形发生器系统设计方案

本节要设计一个能产生正弦波、方波、三角波和由用户编辑的特定形状的波形发生器,和前面的设计相比,主要增加了对波形线性组合的输出要求,以及增加了由输入装置产生任意波形输出的功能。本系统主要由单片机最小系统、频率合成模块、任意波形输入模块、幅度控制模块、滤波和稳幅输出模块等组成,系统总体设计框图如图4.23所示。

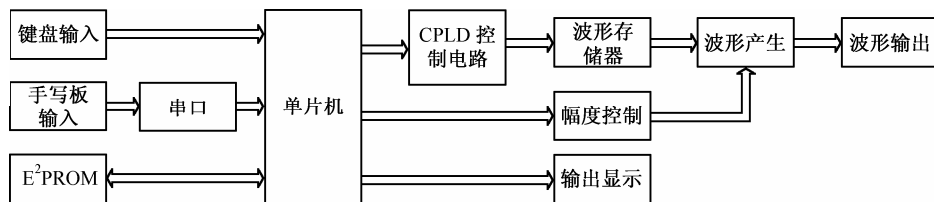


图 4.23 系统总体设计框图

单片机是系统的控制核心,它主要实现以下的功能:一方面控制显示单元显示输入控制信息(包括实时显示输出波形的频率、幅度信息及修改的数值信息),控制按键识别和功能选择;另一方面与 CPLD 结合,实现直接数字频率合成。单片机提供 CPLD 的控制端口,对 CPLD 内部进行控制和选择,

配合 CPLD 对双口 RAM 内存储数据的读取。另外,单片机还和两片 D/A 相连,实现对输出波形的幅度控制。本设计的重点包括波形生成方式、频率变化范围和步进、输出电压幅度范围和步进及稳幅控制等,而设计的难点在于任意波形及各种组合波形的生成,以及发挥部分的频率扩展、频谱分析、失真度分析和扫频输出等。

4.5.2 系统设计关键点分析

1. 频率合成模块

频率合成模块前面已做了详细的讨论,频率合成模块可采用 DDS 技术由 CPLD 电路来实现。采用 DDS 技术不仅可以生成正弦波、方波、三角波及上述波形的基波和谐波的线性组合波,还可以生成任意波形。方波、三角波生成原理和正弦波生成原理一样,需要先在 ROM 中预存波形表。

由于任意波形发生器设计的核心组成部分是波形生成(频率合成),所要求产生的所有波形均由该核心子系统产生,所以,必须首先确定频率合成模块的技术参数。根据设计要求,波形频率范围为 100 Hz~200 kHz,步进不大于 100 Hz,为保证 200 kHz 时取样点不小于 32 点,选取相位累加器时钟频率为 8.388 608 MHz,相位累加器位数为 23 位,则频率步进为 1 Hz,频率控制字 K 的位数为 18 位 ($2^{18} = 262\ 144$ Hz),最高输出频率为 262 kHz,最低输出频率为 1 Hz。

2. 波形线性组合

(1) 正弦波、方波和三角波线性组合波生成原理

从键盘上可编辑生成正弦波、方波和三角波三种波形(同周期)的线性组合波形,其函数形式表示为

$$V_{\Sigma}(t) = AV_{\sin}(t) + BV_{\text{sqr}}(t) + CV_{\text{tri}}(t) \quad (4.22)$$

式中, $V_{\Sigma}(t)$ 为组合的波形函数; $V_{\sin}(t)$, $V_{\text{sqr}}(t)$ 和 $V_{\text{tri}}(t)$ 分别为标准的正弦波、方波和三角波函数; A , B 和 C 分别为 3 种波形在组合波形中的加权系数。只要通过键盘输入 3 种波形的加权系数,经计算就可以得到给定的组合波形。

波形的线性组合具体实现方法为:采用等间隔采样,每个周期取 1024 个点,把对应的采样点加权相加后的值作为合成波形的一个对应点的值,最后再对这些值进行归一化,即把它们都量化在 0 到 255 之间,得到新的函数表存入 ROM 或 RAM 中,等待调用。计算公式如下式所示:

$$A_n = \sum b_i \times a_{in}, \quad a_n = (A_n - A_{\min}) \times 256 / (A_{\max} - A_{\min}) \quad (4.23)$$

式中, A_n 为第 n 点的加权和; b_i 为第 i 个待叠加波形的权值; a_{in} 为第 i 个待叠加波形的第 n 点的幅值; a_n 为第 n 点叠加波形的量化值; $A_{\min}$ 为加权和最小值; $A_{\max}$ 为加权和最大值。

(2) 编辑生成基波及谐波(5次以下)线性组合波的原理

对于基波及谐波(5次以下)的线性组合波形,处理方法与前面相同,只是此时的正弦波、方波和三角波是由各自的基波和 5 次以下的谐波所构成的。

对正弦波、方波和三角波进行傅里叶级数分解,分别可分解成:

$$\text{正弦波} \quad f_1 = \sin \omega t \quad (4.24)$$

$$\text{方波} \quad f_2 = \frac{4}{\pi} \left(\sin \omega t + \frac{1}{3} \sin 3\omega t + \frac{1}{5} \sin 5\omega t \right) \quad (4.25)$$

$$\text{三角波} \quad f_3 = \frac{4}{\pi^2} \left(\cos \omega t + \frac{1}{9} \cos 3\omega t + \frac{1}{25} \cos 5\omega t \right) \quad (4.26)$$

由此可见，只要利用一个正弦函数表，通过改变频率和振幅的办法分别求出基波、各次谐波(5次以下)对应点的值，然后求和，就可以得到新的函数表，存放在 ROM 或 RAM 中，等待调用。

3. 任意波形输入模块

任意波形输入模块可由用户选择预设波形、调整波形和输入自定义波形，可以采用键盘输入，也可以采用手写板输入。采用键盘输入，输入值精确，实现方便，但是用户自定义输入时无法自由输入想要的特殊波形。采用手写板输入，对于用户输入想要的特殊波形非常方便，但是输入确定值则比较困难。因此，可以由键盘和手写板配合起来实现任意波形输入，在需要输入数据时，可以使用键盘；在需要自定义波形时，可以使用手写板。这样结合了上述两种方案的优点，实现了极为方便的人机交互方式。

采用手写板输入时，以手写板作为前向通道，采集用户在手写板上绘制的波形，并将其存储和显示。手写板和单片机之间通过串口进行数据传输。当手写板被触及时，它便将触及点的坐标值进行适当的编码，并打包传给单片机，单片机接收到数据后计算触及点的坐标并进行相应处理，然后存储起来，并通过频率合成模块，输出所需的波形。

手写板可分为三类：电阻式压力板、电磁式感应板和电容式触控板。用手写板做输入，需要破译串口通信的命令码。下面以“联想 1+1 手写连笔王”为例，对手写板的工作原理给以说明。

(1) 手写板的编码方式

当手写板感应到有输入波形信号时，发送四帧数据到单片机的串口上，这四帧数据有固定的格式(如表 4.2 所示)。

表 4.2 手写板的四帧数据格式

第一帧数据			第二帧数据			第三帧数据			第四帧数据		
含义	总位数	有效位数	含义	总位数	有效位数	含义	总位数	有效位数	含义	总位数	有效位数
X 轴方向的高位数据	8	低 3 位	Y 轴方向的高位数据	8	低 3 位	X 轴方向的低位数据	8	低 7 位	Y 轴方向的低位数据	8	低 7 位

(2) 手写板的解码方式

手写板串口通信数据包格式为 7 个数据位、1 个停止位，波特率通常为 1200 b/s，而单片机采用的通信数据格式为 8 位数数据位，若要两者能够互相正确地通信，则要相同的波特率、数据位、停止位等。可采用查询 RXD 的方法，单片机若每隔 1/1200 s，约 800 μs 查询一次 RDX，若 RXD 为下降沿，则开始读一位数据，循环 7 次即可。

若从双口 RAM 中读出的波形有不规则的错值尖峰，可采用下述方法进行解码的修正。

$$\text{Data} = \begin{cases} \text{Data} - 20 & 18 < a < 23 \\ \text{Data} + 20 & -23 < a < -18 \\ \text{Data} - 40 & 39 < a < 43 \\ \text{Data} + 40 & -43 < a < -39 \end{cases} \tag{4.27}$$

式中，Data 为当前从 RAM 中读出的数值；a 为从 RAM 中读出的前、后两个数值的差值。

通过这种修正方式使得 RAM 的输出波形具有很好的平滑度和连续性，较好地实现了手写板的解码和任意自定义波形的产生。

4. 存储器的选择

由于需要产生任意波形，因此需要编辑波形，可以采用 RAM 或 E²PROM 存储待编辑的波形数据，通过 CPU 改变存储器数据，并由相位累加器完成读取数据的功能，以实现波形编辑的功能。

此外,也可以采用特殊存储器——双口 RAM。双口 RAM 有左右两套完全相同的 I/O 口,即两套数据总线、两套地址总线、两套控制总线,并有一套竞争仲裁电路,双口 RAM 的两套数据总线、地址线、片选线和读写控制线相互隔离,可以通过左右两边的任一组 I/O 口进行全异步的存储器读写操作,硬件电路简单可靠,性价比高。双口 RAM 可以选择 CMOS 工艺的芯片 IDT7130,其存储容量为 1 KB 而且片内带有总线仲裁电路,可提供左右两端各自的控制端口和地址线,具有 $\overline{\text{BUSY}}$ 和 $\overline{\text{INT}}$ 两种总线仲裁方式。要产生任何波形,单片机只需向双口 RAM 存储一个周期的波形数据,然后通过增量地址信号控制双口 RAM 输出该波形,经过 D/A 转换后,得到所需波形。

5. 波形存储功能的实现

设计要求具有掉电存储功能,可存储掉电前用户编辑的波形和设置,常用的掉电保护方法一般有以下几种。

(1) 对系统板加备用电源

当电压监测电路监测到电源电压低于某电压值时,自动启动备用电源对系统板进行供电。电源电压恢复后,再关闭备用电源。

(2) 采用非易失性存储器对关键数据进行存储

常用的非易失性存储器有 E^2PROM 、 NVRAM 、 FLASH 等,把系统运行的关键数据存储到非易失性存储器中,当系统掉电后,其中的数据仍然存在,不会丢失,系统加电后,从其中读出数据,系统就可以继续运行。此方案操作简单,易于实现与单片机的连接。

例如可采用 $\text{E}^2\text{PROM} 2817(2\text{KB} \times 8)$ 作为存储器,在写操作时,单片机对其 $\text{RDY}(\text{R}/\overline{\text{B}})$ 信号进行查询,有效则继续写入,无效则等待,每次输出波形之前,先对波形进行存储。

6. 稳幅输出

稳幅输出常用的方法是功率驱动加闭环反馈,采用小功率运算放大器外加两只晶体管做推挽功率输出,从输出端到输入端实现闭环负反馈。这一方法需要仔细计算运算放大器的增益带宽积及外部晶体管引起的相移对负反馈的影响。另一简便的方法是根据稳幅输出的指标要求,当负载变化时,输出电压幅度变化不大于 $\pm 3\%$ (负载电阻变化范围: $100\Omega \sim \infty$),可以在输出级采用几个小功率运算放大器的跟随器并联起来构成增益为 1 的功率电路,使驱动能力提高 3~4 倍,而频率特性与单只运算放大器接近。这样相当于增加了电压串联负反馈,对输出电压有稳定作用,且输出电阻小,带负载能力强。

7. 单次或多次(1000 次以下)特定波形的产生

单次或多次特定波形的产生方案有多种,可以通过计数法来实现,即通过计数累加器进位信号来得到输出波形个数,当达到要求输出个数时就禁止输出。另外,也可以通过单片机中断请求法来实现,即将进位信号反相后输入至单片机中断口,每输出一个完整波形就向单片机发送中断。单片机对中断次数计数,到达预定值即控制 CPLD 停止输出。

8. 扫频输出

采用 DDS 技术实现扫频输出比较方便。设扫频一次时间为 T ,将 T 均分为 N 个时间等份,每一个时间间隔改变频率一次,并设置换频次数计数器,每次换频时改变频率控制字 K ,同时计数器加 1, K 进行线性累加,频率就会由低到高或由高到低变化。当计时器的计数值等于 N 时自动返回并重复,即可实现不间断的扫频输出。

4.5.3 系统软件设计

系统的软件设计，除了要完成基本的功能外，还应有良好的界面，同时还要考虑较好的适应性、通用性，以最大限度地方便调试阶段的修改与调整。故在程序的编制中除了注重结构化设计外，还要注重层次化设计。

系统软件的实现上，底层主要分为三个大的相对独立模块及一个公共模块：第一个是键盘接口模块，第二是显示接口模块，第三是与 CPLD 的控制接口模块；公共模块是不属于上面三部分程序中的公共使用的重要子程序集合。

1. 键盘设计方式

系统键盘设计应包括直接操作键和功能菜单选择键。直接操作键应包括掉电恢复，频率加、减，幅度加、减，波形存、取，波形类型选择，幅度显示，频率显示，波形切换等。菜单键应包括上、下、左、右及确认键，功能选择应有相应的显示提示。

2. 菜单结构设计

菜单设计可以采用多级菜单模式，应包括预设波形、自定义波形和波形叠加等部分。预设波形包括输出正弦波、三角波和方波；自定义波形包括输入波形、保存波形和输出波形。系统初始化时及选择输出波形后，显示界面转入输出参数画面，通过按键设置频率和幅度，选择单次输出、扫频输出等模式。这样菜单设计结构清晰，层次分明。

3. 系统软件设计流程图

软件设计中可采用键盘中断方式，对不同功能键的输出有不同的响应，系统参考的软件流程如图 4.24 所示。

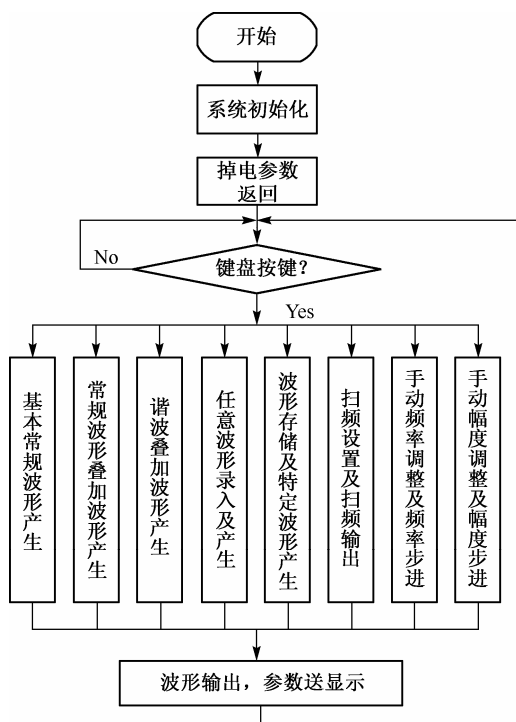


图 4.24 系统主程序流程图

子程序流程图主要包括叠加波形数据产生(参见图4.25)、手写板数据输入与存储、波形幅度控制、波形存储与回放等。

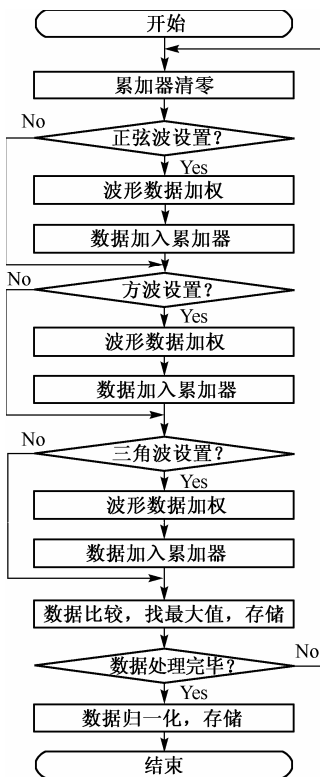


图 4.25 叠加波形数据产生子程序流程图

4.6 数字移相器的设计

1. 设计任务

设计并制作一个具有两路交流信号输出的可移相信号源，两路交流信号的频率、幅值及它们之间的相位皆可以键控调节。其结构示意图如图4.26所示。

2. 设计基本要求

- ① 频率范围：20 Hz~20 kHz，输出频率可预置。
- ② 频率分辨率： ≤ 1 Hz。
- ③ A、B 输出的正弦信号峰-峰值可分别在 0.3~5 V 范围内变化，并可按 0.1 V 步进值增减。
- ④ 相位差范围为 $0\sim 359^\circ$ ，相位步进为 1° ，相位差可预置。
- ⑤ 非线性失真系数 $\leq 3\%$ 。
- ⑥ 检测并显示两路电压的输出有效值，精度优于 1%。
- ⑦ 数字显示预置的频率、相位差值及输出电压。

3. 设计提高部分要求

- ① 频率范围：扩展至 10 Hz~55 kHz。
- ② 相位差步进 0.1° 。

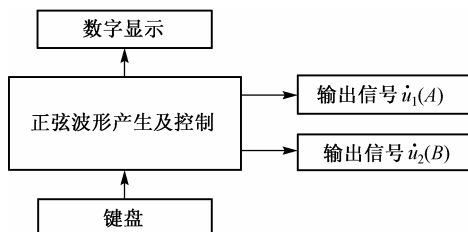


图 4.26 数字移相器结构示意图

③ 将 B 路电压输出改制为一个功率放大器, 额定输出功率值为 10 W, 负载电阻为 $4\ \Omega$, 失真度小于 1%。

④ 功率放大器输出具有过流、短路保护电路。

4.6.1 系统设计原理分析

1. 设计原理

所谓移相, 是指两个同频信号, 以其中的一路作为参考, 另一路相对于该参考作超前或滞后的额定移动, 即称为相位的移动。两路信号的相位不同, 便存在相位差, 简称相差。若将一个信号周期看做为 360° , 则相差的范围为 $0\sim 359^\circ$ 。数字移相器的原理图如图 4.27 所示。由图 4.27 可看出: 数字移相器的设计需构造两路完全相同的 DDS 发生器, 即两个相位累加器的溢出门限值 sum1, sum2 相等, 对应的波形表 1 和波形表 2 也完全相同。第一路产生波形作为基准信号, 只要在起始时刻通过单片机在 sum2 中预置一个相位初值 $K\phi$, 则在同一时钟信号 f_{clk} 作用下, 即可实现具有相位差 ϕ 的两路输出波形。输出波形频率及两路波形的相位差分别为

$$f_{\text{out}} = \frac{f_{\text{clk}}}{\text{sum}} \cdot K, \quad \varphi = \frac{K\phi}{\text{sum}} \cdot 360^\circ \quad (4.28)$$

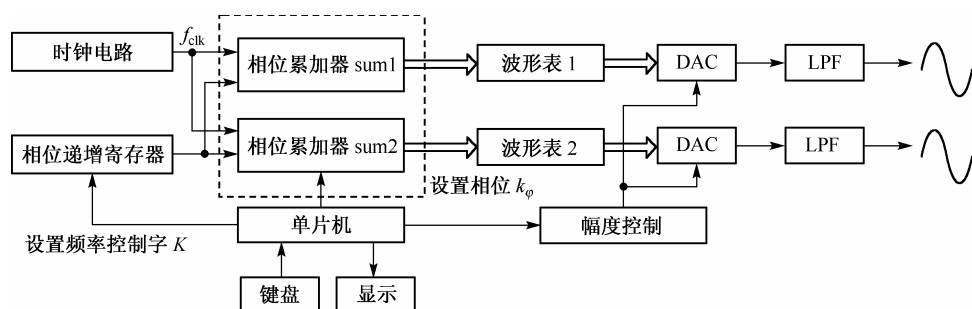


图 4.27 数字移相器的原理图

2. 理论分析与计算

(1) 系统时钟与相位累加器及频率控制字的设计

如前所述, 如何选取时钟频率、累加器字长及相位增量的值是系统设计的关键。数字频率合成可以看成是一个采样过程, 应首先确定采样频率的取值。根据采样定理, DDS 的最高输出频率应小于 $f_{\text{clk}}/2$, 另一个需考虑的因素是 DAC 的转换率, 目前转换速率达到几十 MHz 的 D/A 转换器是很多的, 例如 Maxim 公司的 MAX5186 及 MAX5189, 是一种转换速率可达 40 MHz 的 8 位 D/A 转换器。MAX5186 为电流输出型, 而 MAX5189 为电压输出型。选择上述型号的 D/A 转换器显然有利于系统指标的实现。但考虑到系统设计的成本等因素, 在实际应用中可获得的 DAC 转换率一般不高于 10 MHz, 故 f_{clk} 不能取太高。考虑指标中输出正弦波频率为 20 Hz~20 kHz, 在上限输出频率 20 kHz 时必须满足 1° 的步进, 取样点数不能少于 360 点。因此, 时钟频率 f_{clk} 必须大于 $20\text{ kHz} \times 360 = 7.2\text{ MHz}$, 这仅仅是一个下限。在设计中考虑到 1 Hz 的频率分辨率的要求及累加器工作时做模 2^n 的循环累加等特点, 应尽可能使 f_{clk} 的取值满足 2 的 n 次方的关系, 而可选的晶振频率为 32 768 Hz 或 4.194 304 MHz。

所以有

$$f_{\text{clk}} = 32\ 768\text{ Hz} \times 2^8 = 2^{23}\text{ Hz} = 8\ 388\ 608\text{ Hz}$$

$$f_{\text{clk}} = 4\ 194\ 304\text{ Hz} \times 2^1 = 2^{23}\text{ Hz} = 8\ 388\ 608\text{ Hz} \quad (4.29)$$

当 f_{clk} 的取值选定后, 对应的累加器的位数也就确定了, $n = 23$ (累加器的位数取 23 位)。频率控制字为

$$K = (f_{\text{out}} \times 2^n) / f_{\text{clk}}, \quad \Delta f_{\text{out}} = 1 \text{ Hz}, \quad f_{\text{out}} = K \quad (4.30)$$

经计算后取频率控制字的位数为 15 位 ($2^{15} \text{ Hz} = 32\,768 \text{ Hz}$), 频率分辨率为 $\Delta f_{\text{out}} = f_{\text{clk}} / 2^{23} = 1 \text{ Hz}$ 。

(2) 相位分辨率

由图 4.27 可以看出, 为了实现精密相位调节, sum2 中通过单片机预置了相位累加初值 $K\phi$, $K\phi$ 加在相位累加器的高 A 位 (高 A 位对应于波形表的样本个数), 所以两路输出信号的相位差与累加器初值 $K\phi$ 的关系是

$$\varphi = (K\phi / 2^A) \times 360^\circ, \quad \Delta\varphi = 360^\circ / 2^A \text{ (相位分辨率)} \quad (4.31)$$

若要求产生相位差为 φ 的双路同频信号, 可求得预置的相位累加器初值为

$$K\phi = (\varphi \times 2^A) / 360^\circ \quad (4.32)$$

由相位分辨率公式 $\Delta\varphi = 360^\circ / 2^A$ 可知, 当 A 足够大时, 可以产生任意小的相位差, 从而实现输出信号相位的细微调节。

(3) ROM 的设计 (波形表及寻址方式)

考虑到设计要求在一个周期内实现 $0^\circ \sim 359^\circ$ 的相位调节, 相位步进为 1° , 则在一个周期内的采样点数至少为 360 点。设计中为了实现准确的相位调节和逼真的输出波形, 对一个正弦周期, 等相位间隔取 1024 个采样点依次存储于 $E^2\text{PROM}$ 中, 也就是说, 把正弦输出波形的一个完整周期采样幅值按相位步进顺序分别量化存储于两个 $E^2\text{PROM}$ 中, 因此从累加器内输出的 $E^2\text{PROM}$ 地址为 10 位, 即 $A = 10$, ROM 内的正弦查找表大小为 1 K, 可选定一个 $4\text{K} \times 8$ 的 $E^2\text{PROM}$ 作为正弦查找表。

(4) D/A 的设计

① 数据位宽度及匹配

采用 10 位 D/A 的设计可以保证输出信号有较高质量, 但在硬件设计上过于复杂。采用数据宽度匹配的 8 位 D/A, 在设计中采用在幅度上仅分 256 个点, 而在相位上仍然分为 1024 个点, 所以可以保证 1° 的相位分辨率。当然这种方法的缺点也是明显的, 输出波形的失真比采用 10 位 D/A 的设计方案要大一些。

② D/A 的建立时间

在设计中, 应注意采样点的间隔时间应大于 D/A 的建立时间, 或者说 D/A 应有较快的建立时间, 以保证在第二个采样点到来之前建立完毕。这可以分成两种情况进行讨论。当频率控制字 $K < 2^{23-10} = 2^{13}$, 即输出频率小于 2^{13} Hz , 输出信号的每个周期包含 1024 个样点, 此时要求 D/A 的建立时间 (Setup Time) $= 1 / (K \times 1024)$; 但是当输出信号频率大于 2^{13} Hz 时, 输出信号每个周期中包含的点数小于 1024 个点, 此时要求 D/A 的建立时间为 (Setup Time) $= \frac{1}{K} \left\lceil \frac{2^{23}}{K} \right\rceil > 1 / (K \times 1024)$, $[\cdot]$ 表示取整。综上所述, 可得设计要求 D/A 的建立时间满足关系式 (Setup Time) $< T_{\min} / 1024$ 。

对于前面所分析的 10 位 D/A 来说, 由于采用了 ROM 分时寻址输出到 D/A 的方法, 为使 D/A 工作正常, 公式应修改为 (Setup Time) $< T_{\min} / (1024 \times 2)$ 。式中 $T_{\min}$ 对应于输出最高频率信号的周期, 指标要求为 $f_{\max} = 20 \text{ kHz}$, 所以 $T_{\min} = 5 \times 10^{-5} \text{ s}$, 计算得到 (Setup Time) $< 2.44 \times 10^{-8} \text{ s}$ 。显然, 寻找这样高速的 D/A 是十分困难的。

由上述分析可知, 采用 8 位 D/A 时, 幅度量化仅取 256 点, 所以有 (Setup Time) $< T_{\min} / 256$ 。又 $f_{\max} = 20 \text{ kHz}$, 所以 $T_{\min} = 5 \times 10^{-5} \text{ s}$, 计算可得 (Setup Time) $< 2 \times 10^{-7} \text{ s}$ 。

DAC0800 的建立时间为 100 ns, 可以满足设计要求。

4.6.2 系统设计方案和难点分析

1. 系统设计方案

数字移相信号发生器系统设计框图如图 4.28 所示。由图 4.28 可以看出, 在设计方案中, 预先将输出波形的一个完整周期采样幅度值按相位步进顺序分别量化存储于两个 E²PROM 中。单片机最小系统根据用户输入的频率 f_{out} 和相差值 φ , 由下式

$$K = (f_{\text{out}} \times 2^n) / f_{\text{clk}}, \quad K\varphi = (\varphi \times 2^A) / 360^\circ \quad (4.33)$$

计算出相应的频率控制字 K 和相位累加器初值 $K\varphi$, 在 FPGA/CPLD 中设置两个 n 位相位累加器, 在系统时钟 f_{clk} 的驱动下, 分别以 0 和 $K\varphi$ 为累加器的初值, 均以 K 为累加量做模 2^n 的累加运算。同时将两个 n 位相位累加器的高 A 位作为地址, 对存储在 E²PROM 中的波形幅度值进行查表, 查到的波形量化幅度值经高速 DAC 转换、LPF 平滑滤波后, 即可得到两路频率为 f_{out} 相差为 φ 的输出信号(由前面分析计算可知: $A=10$, 输出信号的相位分辨率 $\Delta\varphi = 360^\circ / 2^{10} = 0.35^\circ$, 满足设计要求, 其中 K 取 15 位, $f_{\text{clk}} = 8.388\ 608\ \text{MHz}$, 相位累加器位数 n 取 23 位, $\Delta f_{\text{out}} = 1\ \text{Hz}$, $f_{\text{out}} = K$)。

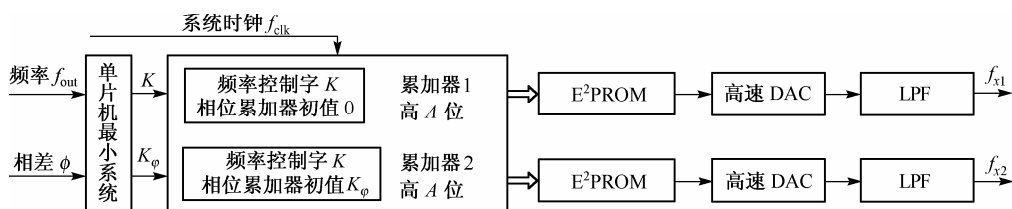


图 4.28 数字移相信号发生器系统设计框图

2. 难点提示

显然, 系统设计中不同的技术指标和功能的要求对应于不同的设计难度。本课题由于在发挥部分提高了合成频率的范围和相位分辨率精度的技术指标, 增大了设计难度, 下面分别加以讨论。

(1) 时钟频率 f_{clk} 的设计和确定

由前面分析, 在指标变更后其上限频率 $f_{\text{max}} = 55\ \text{kHz}$, 若要满足 0.1° 的相位步进(分辨率), 取样点数不能少于 3600 点, f_{clk} 应满足下式要求:

$$f_{\text{clk}} > f_{\text{max}} \times 3600 \quad (4.34)$$

即 $f_{\text{clk}} > 55\ \text{kHz} \times 3600 = 198\ \text{MHz}$, 显然无法找到能配合这样高速时钟工作的高速 DAC, 从而完成信号的重建。

(2) 累加器位数和波形表地址对应的问题

由于在 DDS 的设计中并不是取累加器全部位作为地址线, 一般都是取相位累加器的高 A 位(A 对应于波形表的数据位数)。可能产生相位累加器的溢出值 sum 在其数值不同的情况下, 其地址线的输出值却可能是相同的, 使得波形表输出值相同。现举例如下: 设 sum 取 4 位, 则其最大值为 15, 若取其高两位作为地址线输出, 当 sum 为 12(1100)时, 地址线输出是二进制 11。当 sum 为 15(1111)时, 地址线输出依然是二进制 11。也就是说, sum 可能的 4 个不同的累加值, 对应的地址线输出都是一样的。造成在相位累加器的溢出门限值一定的情况下, 不同的累加值却对应于同一输出地址, 由此造成某些频率及相位的预置不能实现, 系统出现盲点。而这一现象在较高频率段设置两个相邻相位(极小值)

时表现得尤为突出。

例如在频率为 50 kHz, 相位在 359.9° 和 359.8° 时的误差产生波形如图 4.29 所示。

3. 改进设计的方案

(1) 改进方案的初步设计

改进设计使 sum1 也可以预置相位, 不同于前一方案 sum1 的相位初始值恒为 0, 两路波形生成单元均可预置相位。对于一个给定的频点, 两个相位累加器分别预置不同的相位值(相对于 0° 的起始点)实现 0.1° 相位差的步进调整, 如图 4.30 所示。

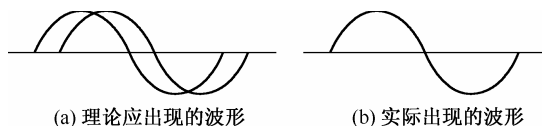


图 4.29 误差产生的波形示意图

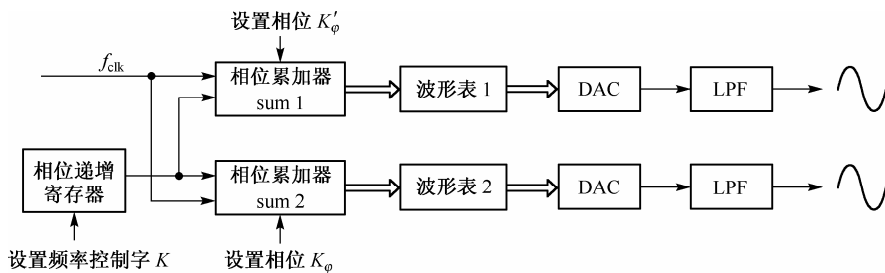


图 4.30 改进的移相器设计框图

(2) 如何解决地址重叠的问题

需要注意的是, 上面介绍的基本移相器的实现方法是基于前面介绍的 DDS 信源设计的方法所做的设计, 只能保证输出信号的相位分辨率小于 1°, 并不能保证输出信号的相位分辨率恰为 1° 的整数倍。下面以发挥部分的设计为例, 介绍一种改进的移相器设计方法。

由于发挥部分的指标要求相位分辨率为 0.1°, 较基本要求提高了一个精度量级, 对应于正弦波形表的取样点应等于或大于 3600 点, 即每一个点代表 0.1° 的相位差, 由于 ROM 为字节寻址, 则需要 12 条地址线来指定波形表的输出。12 条地址线对应 4096 个样本, 所以并不需要使用 12 条地址线所代表的全部地址, 只用到其中前 3600 个 (0~3599) 地址即可满足需求, 即二进制数 0000 0000 0000B~1110 0000 1111B。

假设在初始时刻, sum1 的值为 0, sum2 的值应能保证使输出两路同频信号的相位差超前或滞后 0.1°。问题是, 既然是同频信号, sum1 和 sum2 的累加值(累加器溢出门限值)应相同。如何实现在预置了任意两个不同的相位值时, 不会出现所有时刻地址线输出都相同的情况。显然, 解决办法只有在 sum 的高 A 位 (A=12) 上想办法, 使得 sum1 值为 0, sum2 的高 A 位地址线的最低位置 1, 如图 4.31 所示。

由图 4.31 可看出, 由于 sum2 的输出地址值总比 sum1 的输出地址值多 1, 于是两路输出波形实现了 0.1° 的相位差。同样, 0.2°~359.9° 的相差输出依照同一方法也就不难实现。

但仔细分析后发现问题, 问题是 sum 中的高 A 位作为波形表的寻址地址, 其后还有 B 位 (若 $n=23$, 则 $A=12$ 时其后还有 11 位, $n=A+B$), 则有

$$3600 \times 2^B \leq \text{sum} \leq 3600 \times 2^B + (2^B - 1) \quad (4.35)$$

其中, 3600×2^B 是低 B 位全为 0 的情况, 若低 B 位全为 1, 则有 $3600 \times 2^B + (2^B - 1)$ 。显而易见, 为了确保 sum1 和 sum2 同时溢出或分别溢出时, 仍然保持相对恒定的地址差值, 应使 B 位全为 0, 即此时应当取累加器的溢出门限为 $\text{sum} = 3600 \times 2^B$ 。

此时产生的疑虑是，若使 B 位全为 0，相位累加器的溢出门限值减小，输出信号的频率分辨率能否达到指标要求的 1 Hz？分析和计算如下：

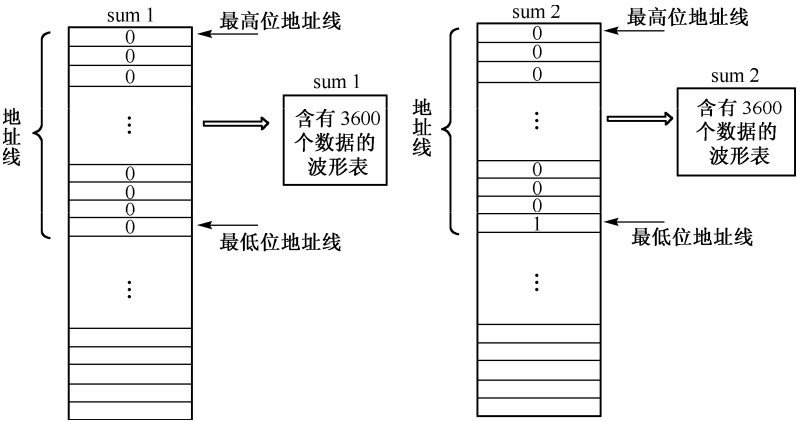


图 4.31 实现 0.1° 相位差的寻址方式

假设采用 32.768 kHz 的晶振，将其 225 倍频： $32\,768\text{ Hz} \times 225 = 7\,372\,800\text{ Hz}$ ，取 $f_{\text{clk}} = 7.3728\text{ MHz}$ ， $B=11$ ，则 $\text{sum} = 3600 \times 2^{11} = 7\,372\,800$ 。此时频率分辨率为 $\Delta f_{\text{out}} = f_{\text{clk}} / \text{sum} = 1\text{ Hz}$ ，相位分辨率为 0.1° 。由相位预置初值公式 $\varphi = \frac{K\varphi}{\text{sum}} \cdot 360^\circ$ ，可计算出相位差值并由单片机预置。当输出信号频率为 $f_{\text{max}} = 55\text{ kHz}$ 时，每个信号周期中含有 134 个采样点，满足 Nyquist 采样要求，能够很好地恢复波形。表 4.3 给出了快速预置的方法，用以简化单片机的运算。请读者思考：为什么发挥部分要求的相位分辨率提高了，而要求的外部输入晶振频率(时钟信号频率)却降低了？

表 4.3 相位值预置表

预置的相位值	地址线对应的预置值
0.1°	1 (0.1×10)
0.2°	2 (0.2×10)
0.3°	3 (0.3×10)
N	N
1°	10 (1×10)
N	N
359.9°	3599

- 以上方法的主要思考在于：
- (1) 设计中应保证预置的相位值有唯一对应的地址，采用方法为把预置的相位值直接预置在确定的波形表输出的地址线上。
 - (2) 采用查找表方式简化了计算，并确保两个相位累加器在同时溢出或先后分别溢出的情况下，仍然保持恒定的地址差值。
 - (3) 增加波形表长度可使相位调节精度更高。利用信号周期的对称性和算术关系，采用优化的软件算法可减少系统硬件资源的开销。

4.6.3 系统软件设计

数字式移相信号发生器程序的参考软件流程图如图4.32所示。首先通过按键设置两路信号的幅度、频率和相位差，然后通过控制波形发生子系统电路输出相应的波形，并将参数显示在显示单元上。

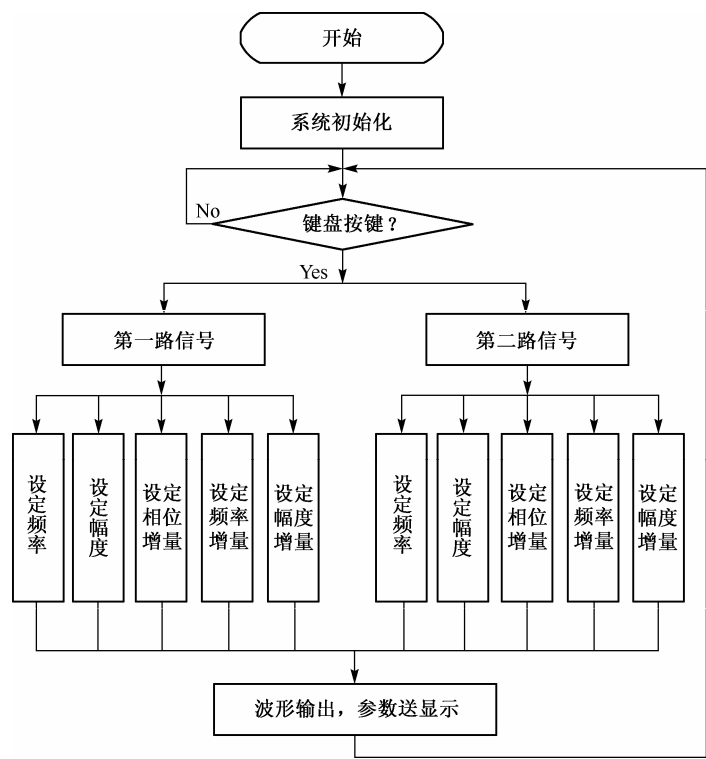


图 4.32 数字式移相信号发生器程序参考软件流程图

4.7 本章小结

本章的主要内容是在基于单片机最小系统的基础上完成数字信号源的设计，通过分类细化训练内容，让读者熟悉并掌握锁相频率合成与直接数字频率合成的数学和物理模型，并进行有效的设计。在对这些模型有较为全面理解的基础上，能够系统地完成任意波形发生器、数字移相等的设计。

数字信号源的类型是多种多样的，在基于 DDS 和 PLL 的设计模型基础上，可以选用的设计方案和实现方式也是多样的，要根据具体的设计要求进行选择。本章通过几个典型设计实例的介绍，旨在培养读者根据设计的要求进行系统分析、方案论证、数学计算、模块设计、系统调试和结果分析的能力，从而建立科学、合理的电子系统设计思想。

第 5 章 数据采集与回放系统的设计与实现

5.1 引言

数据采集与处理技术是信息科学的一个重要分支，它与传感器技术、信号处理技术和计算机技术一起构成了现代检测技术的基础。数据采集实质上是对信号的采集，是对外界信息的一种获取过程，它是将外界各种类型的信息通过某种形式的转换，变为可度量的信号以进行捕获。事实上，多种形式的信息采集，贯穿于人类的发展过程中，从古代的日晷、地动仪到近代的温度计、电压表，无不是人类为了准确获取外界信息的创造。在当代，随着计算机的广泛应用，信号采集往往指将外界的各类自然信号利用各种传感器转换为可编码的电信号形式，然后输入计算机进行信息处理的技术。与此同时，将计算处理后得到的数据进行显示或打印，以便实现对某些物理量的监视，其中一部分数据还将被生产过程中的计算机控制系统用来控制某些物理量。简而言之，数据采集是将温度、压力、流量、位移等模拟量采集转换成数字量后，再由计算机进行存储、处理、显示或打印的过程。

随着科学技术的发展和数据采集与处理系统的广泛应用，人们对数据采集系统的主要技术指标，如速率、分辨率、精度、输入电压范围及抗干扰能力等方面，都提出了越来越高的要求。数据采集系统性能的好坏，主要取决于它的精度和速度。在保证精度的条件下，应有尽可能高的采样速度，以满足实时采集、实时处理和实时控制的要求。

本章首先对数据采集与回放系统的原理和设计方法做出较为详细的分析，然后以高精度数据采集与回放系统和语音存储与回放系统为例，详细介绍不同系统性能指标下的数据采集与回放系统的设计方案，并给出相应的设计实例，其基本思想是通过训练，使读者能获得数据采集与回放系统的设计思路，即在已知输入信号带宽(BW)的条件下，从总体上把握设计要素，综合考虑系统在各个环节上的匹配和计算结果，从而实现合理的设计。

5.2 数据采集与回放系统设计方法概述

5.2.1 采集系统分类

从系统结构来划分，数据采集系统有集中型和分布型两种，分别如图5.1和图5.2所示。从处理方式来划分，可分为实时处理和事后处理两类。实时处理一般只能对采集数据进行简单基本的处理，而事后处理则可以对数据进行复杂的处理。

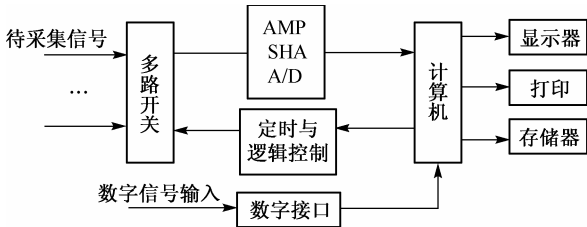


图 5.1 分布型数据采集系统

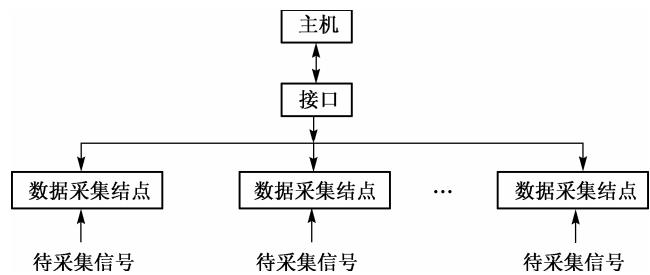


图 5.2 集中型数据采集系统

5.2.2 数据采集过程

模拟信号的采集是一个多步过程，如图5.3所示。

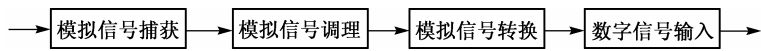


图 5.3 模拟信号采集的主要过程

模拟信号捕获的主要任务是获取特定的外界信息。就近代技术而言，信号捕获主要是由特定的传感器实现的，其主要作用是获取外界的多种信息，如温度、湿度或复杂环境的变化信息等。它的另一作用是将获取的各种形式的物理信号，如声音、气压等转换为可度量的电信号，这是它在采集系统中的基本作用。

模拟信号调理是把输入的信号进行归一化、滤波等处理，以优化待量化的信号。

模拟信号转换即通过 A/D 转换器将模拟信号转换为数字信号，它是采集过程的核心，在这一过程中，待采集的模拟信号被转换为一定精度的数字信号，这一过程决定了采集系统的基本指标。

数字信号输入是指将经过 A/D 转换器转换的数字信号输入到计算机中去处理。

当然，一个完整的系统还包括信号的预处理与数据存储等部分。它们的意义在于改善采集信号的质量和保存数据。另外，某种形式的人机交互，以方便在必要时对系统进行干预的功能，也是必不可少的。

5.2.3 A/D转换器与D/A转换器的选取

1. A/D转换器的选取和技术条件

对于 A/D 转换器的选取，应从以下几个方面考虑。

- 模拟信号输入范围、信噪比、系统总的精度要求，并根据这些要求确定分辨率。
- 线性误差、相对精度及稳定性。
- 转换时间。
- 系统电源稳定性要求及由于电源引起的误差变化范围。
- 输入信号特征，是否为采样信号，滤波环路的影响及信号频率的大小。
- 环境温度变化的误差限。
- 和 MCU 的连接方式。

2. A/D转换器的类型及选择

A/D 转换器的品种繁多，其分类方法主要有以下几种。

- 按速度分类。

- 按精度分类。
- 按位数分类。
- 按转换方式分类。
- 按组成结构分类。

就其组成结构分类,有计数比较型、逐次逼近型、积分型、 $\Sigma\Delta$ 调制型、并行转换型等。若按速度分类,一般把 10 MHz 以上的 A/D 器件归于高速器件类型。若按精度分类,一般把 16 位以上的 A/D 器件归于高精度器件类型。根据测量学的原理,A/D 转换实质上是数字化测量的一种方法。从“测量是一种比较过程”这一概念出发进行 A/D 分类,归根结底只有两类:直接比较型和间接比较型。直接比较型中典型的是逐次比较型,间接比较型中典型的是积分型。若要提高 A/D 转换器的分辨率,逐次比较型 A/D 转换器是通过增加基准电压的十进制权重分辨率及电压比较次数来提高分辨率。积分型 A/D 转换器则通过将输入电压幅度变换为对应的积分时间,再把时间单位细分来提高其分辨率。随着技术的进步,将直接比较型与间接比较型 A/D 转换器的工作原理结合起来,构成了复合型 A/D 转换器,其性能优良,但价格较贵。

3. A/D转换器的字长

A/D 转换器字长的确定直接关系到数据采集系统的精度、动态特性等,应首先予以确定。A/D 转换器是量化噪声的一个源,这是因为从理论上讲,取样不会引进误差,量化是对时间连续的模拟信号进行幅度离散,这一过程是取整的过程,因此会引入量化误差——量化噪声。为使量化误差限制在一定范围内,A/D 转换器应有足够的字长,而决定有效字长的因素是输入信号的动态范围、分辨率要求、对量化噪声的要求及系统的允许误差。

1) 由输入模拟信号的动态范围确定字长

假定输入模拟量 $|x(t)|$ 的最大值为 $x_{\max}$, 最小值为 $x_{\min}$, 而对于舍入量化应有 $x_{\min} \geq q/2$, 其中 q 为最小量化单位,数值等于最低有效位的值, $x_{\max} = VF_S$, 于是有

$$\frac{x_{\max}}{x_{\min}} \leq 2^{c+1} \quad \text{或} \quad c \geq 10 \lg \frac{x_{\max}}{x_{\min}} - 1 \quad (5.1)$$

对于截尾量化应有 $x_{\min} \geq q$, $x_{\max} = VF_S$, 于是有

$$\frac{x_{\max}}{x_{\min}} \leq 2^c \quad \text{或} \quad c \geq 10 \lg \frac{x_{\max}}{x_{\min}} \quad (5.2)$$

要满足系统对静态精度的要求,应按式(5.2)确定字长 c 。但应注意的是,在实际应用中,动态范围 $x_{\max}/x_{\min}$ 的取值不仅取决于 A/D 转换器,还取决于测量系统内其他仪表的相对精度,在动态情况下,应考虑采集系统前向通道的综合精度,即使得系统的总误差尽量不是由 A/D 转换器引入的。

2) 由信噪比 S/N 确定字长

考虑到一位符号位后,A/D 转换器的字长为 $c_1 = c + 1$ (c 为有效字长或数值位),所以按信噪比确定 A/D 转换器的字长 c_1 , 有

$$c_1 = c + 1 = \frac{S/N}{6} + 0.8 + 1 \quad (5.3)$$

因此,A/D 转换器的字长为

$$c_1 \geq \max \left\{ \left\lceil 1 + \lg \frac{x_{\max}}{x_{\min}} \right\rceil, \left\lceil \frac{S/N}{6} + 0.8 + 1 \right\rceil \right\} \quad (5.4)$$

3) 由系统对误差(或精度)的要求确定字长

相对于前两种方法而言,从系统误差角度考虑是一种更为合理的方法。其原则是:A/D转换器的位数由系统的总精度确定,取 c_1 高出计算位数的两位以上,以保证系统的总精度要求。由于从系统的角度选定A/D转换器的字长,所以需分析和计算各个环节的误差,工作量较大,因对系统而言,精度主要由传感器和测量装置精度、幅度分辨率误差等因素决定。而幅度分辨率误差又与信号带宽和采样频率有关,相互关联,有一定难度。

需要强调说明的是,并非A/D转换器的字长越长、分辨率越高就越适合进行测量,事实是精度、灵敏度与分辨率是同等重要的。分辨率和精度并不是一回事。分辨率是指转换器所能分辨的模拟信号的最小变化值,是理想的无误差转换器的一种特性。精度是指转换后所得结果相对于实际值的不确定度。二者显然是不同的。实际的转换器均有其精度限制,这是由转换器的性能误差决定的,如线性误差、偏置误差及环境参数引起的这些误差的漂移等。

4. 采样周期的选择

采样周期的选择关系到系统的精度和动态特性,是数据采集系统设计的关键和难点。

1) 传统准则

所谓传统准则,即奈奎斯特(或香农)准则。该准则规定:采样频率必须大于或等于连续信号中所含的最高频率的两倍,即

$$\omega_s \geq 2\omega_{\max} \quad (5.5)$$

式中, ω_s 为采样频率,其中 $T_s = 2\pi/\omega_s$ 为采样周期; $\omega_{\max}$ 为连续信号所含的最高频率。从理论上,该准则可以保证复现原来的连续信号,并给出了采样周期的计算方法,但这一准则并不十分有效,因为

(1) 现实世界中被采样的信号不可能是理想的带限信号,即实际信号不可能是精确规定的已知信号,必须考虑带宽和测量噪声。而对一个阶跃输入信号,由于其包含无限的频率分量,因此无法确定 $\omega_{\max}$ 。

(2) 准则没有表明采样周期与系统性能的关系。虽然在工程中常采用5~10倍的 $\omega_{\max}$ 作为采样频率,以满足要求,但缺乏科学计算及理论支持。

事实上,系统的构成和面对任务的不同及输入模拟信号的多样性,使对采样周期的要求是不同的,甚至是相互矛盾的,并非是采样周期越小越好,必须综合考虑,做出折中的选择。

通常要把A/D分辨率提高到最大限度,就得选择有最好噪声抑制作用的慢转换速率(线路循环积分)。因为采样频率越高,量化噪声放大越大,量化后两个顺序数之间的差别越小,只有在最低位才发生变化,过高的采样频率产生的噪声放大会在输出端显示出来,信噪比下降。由此可看出,精度与速度就是一对矛盾。例如ADS1210在采样频率为10 Hz时可实现24位精度;当采样频率提高到1000 Hz时,它的分辨率只有20位,采样频率再提高,分辨率还要降低。

但从另一方面看,采样频率越高,灵敏度也随之提高。需指出的是,灵敏度和精度、分辨率是不同的参数,灵敏度是绝对量,分辨率是相对量。灵敏度是指在测量时测量装置能检测出的最小绝对变化量。应注意,这里指明的是“测量装置”,A/D转换器包含于测量装置之中。所以确定灵敏度的简便的方法是考察测量装置在最低量程的性能,当然,该量程的噪声指标会在很大程度上决定装置的灵敏度。问题在于:某一测量过程对灵敏度有较高要求,同时又对分辨率有较高的要求,这时采样频率的选取成为矛盾。

从一般意义上看,采样频率的提高是以牺牲分辨率为代价的。采样频率越高,要求计算机的字长越长,否则系统的性能变坏,精度降低。实际上,精度、灵敏度及分辨率是同等重要的,或者说前者

更重要。

2) 根据系统带宽选择采样周期

如前所述,系统的频率响应是各个分量的合成,最高频率分量的确定应以该分量的幅值不低于信号总幅值的 10%左右为标准,若低于此,则可忽略不计。

若系统为闭环系统,其本身具有一定的低通滤波特性,则整个闭环系统响应中各分量的频率由闭环频带决定。因此,采样频率可由闭环系统的带宽(5~10 倍)确定。

若系统为开环系统,采样频率要选得高于系统开环带宽的 10 倍以上。

3) 根据采样保持器的影响选择采样周期

在数据采集系统中,为了减小孔径误差对转换分辨率的影响,大多系统在设计时采用采样保持器。采样保持器会使信号产生延迟,从而产生重构误差。重构误差与采样频率有关,因此在信号的数字处理系统中,采样频率一般应由重构误差的要求确定。对于开环系统,应根据零阶保持器所产生的最大相对误差来计算信号周期内的采样点数。而对于闭环系统,采样保持器带来的相位滞后比误差的影响更大,应根据系统允许的相位滞后来确定采样频率。

4) 实时控制系统的采样周期的选择

对于实时控制系统,要求采样周期与系统的时间常数相比充分小,采样周期越小,数字模拟越精确,控制效果越好。但必须要考虑计算机的字长、速度、工作量及计算时间,特别当系统为多回路控制时,应保证每一回路的控制算法有足够的时间,所以在满足技术指标的要求下,尽可能把采样周期选得大一点。

5) 实验法选择采样周期

综上所述,多种因素的影响使得对采样周期的要求是不同的,甚至是相互矛盾的,必须综合考虑,根据系统设计的具体情况和主要要求做出折中选择。事实上,尽管有多种方法选择采样周期,但最终要满足给定的技术指标要求,仍要用实验方法检验系统在不同的采样周期时间下输出响应的性能。在实际系统中检验和验证,从而最终确定采样周期。需说明的是,尽管用实验的方法选择采样周期是非常有效的,但绝不应因此而低估采样周期选择理论的价值。

5. D/A转换器的选取

在 A/D 转换器确定以后,如何选取一个合适的 D/A 转换器是保证波形回放精度的关键。首先要考虑的是性能参数,D/A 转换器的性能不能低于 A/D 转换器的性能。其次要考虑的是接口和操作方便,方便的接口和操作使得系统容易实现。

6. A/D转换器与缓冲器的匹配问题

A/D 转换器与缓冲放大器的匹配实际上是噪声、失真、阻抗匹配、带宽、建立时间及 ADC 的输入结构等诸多因素对系统精度影响的综合考虑。由于数据采集的系统精度是一个综合精度,即

$$E_{\text{RSS}} = \sqrt{E_{\text{MUX}}^2 + E_{\text{AMP}}^2 + E_{\text{SH}}^2 + E_{\text{ADC}}^2} \quad (5.6)$$

可见,一个数据采集系统前向通道上的误差是模拟开关、缓冲放大器、采样保持电路、ADC 等误差的综合累积。所以必定有如下结论:前向通道上的总误差应小于或至少应该相当于 A/D 转换器的量化误差。或者说,数据采集系统的设计应以 ADC 的信噪比和量化误差为基准来确定系统中其他电路的选取和设计。假定已选 ADC 内含 MUX 和 SHA 电路,此时 A/D 转换器的精度就完全取决于 ADC 与缓冲放大器的匹配。也就是说,为了避免额外的噪声引入系统,输入缓冲放大器(对 ADC 而言是输入信号源)的信号噪声加失真比最低也应该优于 ADC 的理论上限值。

1) 阻抗匹配

阻抗匹配是负载为了获得一个稳定的信号而对信号输出阻抗和负载输入阻抗所施加的要求。数据采集系统中输入通道的匹配实际包含了以下两层含义。

(1) 信号源与缓冲放大器的阻抗匹配。

(2) 缓冲放大器与 A/D 转换器的阻抗匹配。

对(1)而言,其作用在于使信号源能够提供最大功率和稳定信号,并降低对信源的要求;对(2)而言,其作用在于使 A/D 转换器获得最大稳定的输入信号。也就是说,缓冲放大器除了信号放大功能外,在信号源和 A/D 转换器之间还起着一个关键作用即阻抗变换,以使信源和 A/D 转换器之间完成阻抗匹配。

2) 带宽和建立时间

对于缓冲放大器的速度要求,应使其建立时间与 ADC 的采样时间相对应,以保证转换结果的精度。所谓建立时间,是指输入一个阶跃电压,当输出信号达到并且在中心位于最终稳态输出电平附近的一个给定误差带内摆动时的时间间隔,或者说,在多宽的频带范围内的信号能被完整地重建。

另外,放大器要在整个给定的输入信号频率范围内保持低输出阻抗(高输出阻抗的运算放大器不能迅速响应 ADC 输入电容的改变,也不能处理 ADC 产生的瞬态电流)。要获得低输出阻抗,就应该具有高环路增益,而在高频下,宽带运放具有更高的环路增益,因此也就具有低的输出阻抗。所以,对低阻抗的要求变成了对更高带宽的要求。

综上所述,一个实际满足要求的 ADC 组合应仔细考虑缓冲放大器的设计,应把缓冲放大器和 A/D 转换器作为统一体来设计。在噪声、失真、阻抗匹配、建立时间及 A/D 转换器的输入结构等因素之间进行折中考虑。

7. 抗混叠/平滑滤波器的设计

在数据采集集中为避免 ADC 环节对模拟信号采样时产生频谱混叠,在 ADC 环节前必须设置低通滤波器(LPF),称为抗混叠滤波器。抗混叠滤波器的设计效果直接影响到系统的精度。滤波器的设计应遵从三个特性指标:①幅度平坦响应特性;②相位响应特性;③群延时特性。设计时应视信号和任务的不同选择逼近方式。抗混叠滤波器应用的主要目的在于:在采集过程中,原信号的频谱 $x(t)$ 将会在频域周期延拓,其重复周期即是采样速率 f_s ,因此在欠采样的情况下,周期延拓后的频谱必然会发生交叠,从而产生失真。这种失真称为混叠失真。为减小混叠失真,一般应在采样之前对原信号进行低通滤波,其作用可归纳如下:

- 滤除连续信号中高于折叠频率($f_0 = \omega_s/2$)的频谱分量,使采样信号基本频谱的低频段尽可能不混有连续信号的高频分量,以保证采样信号的基本频谱和连续信号的频谱尽可能接近。
- 滤除高频干扰。这是因为信号一般用模拟传感器测得,在检测和传输中不可避免地受到各种噪声的污染。
- 压缩传输信道频带宽度,以利于信号处理和节省存储空间。

在后向通道 DAC 环节,数字量经 DAC 所得到的模拟量其波形实际上是阶梯波,含有高频噪声,所以必须在 DAC 环节后设置低通滤波器,去除高频噪声,这种滤波器称为平滑滤波器。

1) 滤波器的类型选择

在滤波器设计中,首先要考虑选择滤波器的类型,主要有以下两种分类。

从通带性质来划分,主要有 4 种基本类型:低通滤波器(LPF)、高通滤波器(HPF)、带通滤波器(BPF)和带阻滤波器(BEF)。

按照不同的频域或时域要求来划分,可分为巴特沃斯(Butterworth)型、切比雪夫(Chebyshev)型、贝塞尔(Bessel)型和椭圆型滤波器。选择原则是:从系统精度和目标任务的要求考虑选择适合的滤波器类型。

2) 滤波器的设计方法

滤波器的设计是系统设计的难点之一,其设计方法可分为传统设计和近代设计方法两种。传统设计方法着重用归一化数表设计滤波器,利用图表,结合基本公式、定律及准则,可设计出符合要求的滤波器,通过简单的变换能将表中列出的归一化低通滤波器的数据变为高通、带通和带阻滤波器的数据;该方法的缺点在于:由于器件的分布参数影响,需进行较大工作量的实验修正。近代设计方法采用专用的开关电容滤波器来完成滤波器的设计,将滤波器的设计任务简化到仅仅是对时钟频率的选择,其快捷性是传统方法所不能比拟的。

8. 基准源的设计与电压变化抑制比

1) 基准源

“测量是一种比较过程”。因此,基准源的设计实质上是高精度数据采集系统工程设计的最基本保证。如果没有高精度的电压基准源,采用一个高精度的 A/D 转换器变得毫无意义,一个高精度的数据采集系统也是不可实现的。

对一个基准电压源,除了考虑它的精度外,输出噪声、线性度、稳定性和温度漂移也是必须考虑的因素,否则会严重影响系统的精度。输出噪声会增大系统的总噪声误差,从而影响精度。而稳定性、线性度和温度漂移都会引起基准电压源输出电压的变化。一个 12 位 A/D 转换器的分辨率折算成电压为

$$5\text{V}/4096 = 1.22 \text{ mV}$$

当+5 V 的基准电压源输出变化 0.125%时产生的误差为

$$6.25 \text{ mV}/1.22 \text{ mV} \approx 5.12 \text{ LSB}$$

也就是说,12 位的 A/D 转换器实际精度只有 $12 - \lg 5.12/\lg 2 \approx 9$ 位。

由此可见,电压基准源的选取特别重要。实际应用中参考电压源的选取,应根据系统要求对精度、线性度、稳定性和温度漂移等特性参数进行权衡考虑。

2) 电压变化抑制比

A/D 转换器对电源电压的抑制比(PSRR)用改变电源电压使数据产生 $\pm 1 \text{ LSB}$ 变化时所对应的电源变化范围来表示,也可称为电源灵敏度。当电源电压发生变化时,A/D 转换器的输出发生变化,变化的实际作用相当于 A/D 转换器输入量的变化。例如,电源灵敏度为 0.05%VS 时,其含义是电源电压的变化为电源电压 VS 的 1%时,相当于引入了 0.05%的模拟输入量的变化。因此,实际电路中 A/D 转换器和电压基准源的电源都必须进行低阻驱动的稳压处理。

9. 采集数据的预处理

从信号产生的可能性来看,使数据预处理成为必要的原因可分为系统性因素和随机性因素。前者是由理论设计、系统结构等因素造成的;而后者是由不确定性因素如噪声等产生的。

1) 系统因素

(1) 系统标度

由于被测信号是未知信号,多种物理量的单位和量纲不同,但在数据采集时均被统一到 ADC 相关的数字量上,然而被测对象的各种数据的量纲与 A/D 转换器的输入值是不一样的。例如压力的单位为 Pa(帕斯卡)、温度的单位为 K(开尔文)等。这些参数经传感器和 A/D 转换器后得到一系列数码,而这些数据值并不一定等于原来带有量纲的参数值,它仅仅对应于参数值的大小,必须转换成带量纲的

数据才能运算、显示和输出打印等,这种变换称为标度变换。

系统标度从另一方面看,由于量程的不同需进行标度变换,比如对于 8 位 ADC 的输入信号的最大值被固定为 255,中间值则取 0~255 之间的值,然而输入信号的真实值可能在 0~1 V 之间,显然需要转换,以使二次处理程序能被使用。

(2) 采集方法造成的误差

理想采样是抽取连续信号的瞬时函数值,采样中测量的分辨率主要由附加的噪声和干扰信号来决定。从理论上讲,一般可认为采样不会带来误差,但对于大动态信号,若采用动态范围内分段采样的方法,应考虑在信号恢复时采用处理措施,以减少由于采集方法造成的误差。

2) 不确定因素

不确定因素主要是指干扰和噪声,包括系统噪声、环境噪声等,它们使信号产生高频噪声、奇异点和趋势项等。特别是当噪声很大时,会使小信号淹没其中,使有效信息丢失。

- **信号噪声** 主要指高频噪声,除器件噪声外。系统设计不良和强磁环境等均会产生噪声。此时合理设计和适当的屏蔽可大大减小噪声,对采集数据中的噪声可用数字滤波算法如均值滤波来减小。
- **奇异项** 奇异项也是由于干扰造成的。例如采集过程中,外界串入的脉冲电磁波干扰信号。奇异项数据明显地与真实数据有较大的差别,如果采用均值滤波等方法,奇异项可能会被引入处理后的数据中,使处理后的信号与真实信号有较大的差别,因此奇异项应被去除。
- **趋势项** 需处理的信号往往是时变信号,信号中很可能存在小于感兴趣信号成分频率的低频成分(如 50 Hz 工频噪声)。它会影响处理后的结果,需采用一定的方法去除这种趋势项。
- **信号噪声去除** 采集信号的噪声处理有几种简单而有效的滤波算法,根据处理数据对象的不同,可分为两类。

① 第一类是对多个采集序列的对应点进行处理,即对某一个采样点采样 n 次,用这 n 个值进行处理,典型的有:中值滤波和加权平均。

② 第二类是对一个采集序列的相邻的数个点进行平滑处理,是一般意义上的数字滤波。它的任务是消除干扰噪声,同时保持原数据的变化特性。

3) 参数量化误差

所谓参数量化误差,是指在数字系统中有以下几个数值误差源。

- ① A/D 的量化误差。
- ② 系数量化误差。
- ③ 数值运算过程中的有限字长效应。

A/D 的量化误差是系统误差。一般意义上可认为系统误差为固有误差,所能做的工作是如何使系统总的不确定度尽可能地接近固有误差。系数量化误差是由于存储单元的字长有限,引起的截尾或舍入误差。A/D 的量化误差仅取决于传递函数。由运算产生的误差可看成是一种输出附加噪声,其影响因实现方式不同而异。系数误差等效于参数取值的不确定性,必然影响系统总的不确定度。

5.3 高精度数据采集与回放系统的设计

1. 设计任务

以 16 bit A/D 转换器为核心并配置 16 bit D/A 转换器,设计并制作高精度数据采集系统,要求有效字长不低于 14 位。要求系统通过 RS232 串行通信口与 PC 通信,开始采样及采样速率的设定等人机交

互功能均可由 PC 键盘控制。

2. 设计基本要求

- (1) 熟练掌握 A/D 转换器的转换函数及误差调整的方法，并测定精度及有效字长。
- (2) 直通方式下的单、双极性输入信号的响应测试。
- (3) 存储器回放方式下的单、双极性输入信号的响应测试。
- (4) 不同采样频率下采集数据的误差比较。

5.3.1 系统分析与设计

精度是指相对于相关绝对标准而言存在于测量中的总的不确定度。所以在高精度数据采集的设计中，应综合考虑采集系统的精度要求，其中包括：传感器的精度、信号调理电路精度、参考电压源(基准源)的精度、D/A 转换器的转换精度等。从系统设计的角度考虑，高精度数据采集系统的设计是在已知输入信号带宽的条件下，从总体上把握设计各要素，综合考虑信号调理电路、D/A 转换器和 A/D 转换器等的精度，以及系统在各个环节上的匹配和精确度，从而实现合理的设计。

训练任务对采样精度等各方面的要求已明确，系统的整体设计框图如图5.4所示。由理论分析及系统设计框图可看出，系统设计中需考虑的问题有：

- (1) 信号源与 A/D 输入通道中前置放大器(缓冲器)的匹配。
- (2) A/D 芯片的选择，前置放大器(缓冲器)与高精度(16 bit) A/D 转换器的匹配。
- (3) A/D 转换器与微处理器的接口方式。
- (4) A/D 转换器的硬件及软件零点偏移校正、增益误差校正、线性补偿、采样数据的处理。
- (5) 微处理器与高精度(16 bit) D/A 转换器的接口方式。
- (6) D/A 芯片的选择，D/A 转换器与输出放大器的匹配。
- (7) 抗混叠/平滑滤波器的设计。
- (8) 参考电压源(基准源)及工作环境的选取。

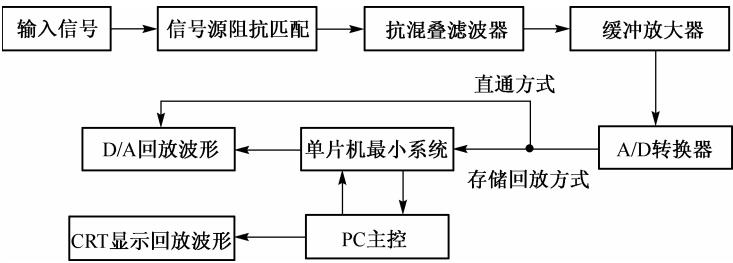


图 5.4 高精度数据采集与回放系统的整体设计框图

5.3.2 系统各模块的设计

1. 信号源阻抗匹配电路设计

信号源匹配电路设计的目的是使信号源的输出阻抗足够小，能够向系统提供最大功率的稳定信号，从而降低系统对信源的要求。利用低噪声精密集成运算放大器 OP37 构成简单的电压跟随电路，可实现较好的匹配效果。

2. 抗混叠滤波器设计

如前所述，在数据采集系统中，抗混叠滤波器的设计效果将直接影响到系统的精度，因此前置通

道的抗混叠滤波器设计就显得十分重要。混叠现象的产生源于以下原因：(1) 连续信号 $x(t)$ 的频谱并非有限带宽。(2) 采样周期 T_s 选得过大，采样频率不满足奈奎斯特采样定理的要求。因此，克服混叠的方法通常有两种：一是提高采样频率，使采样频率不低于连续信号 $x(t)$ 最高频率分量的两倍；二是采用具有陡峭下沿频谱特性的滤波器构成抗混叠滤波器。由于理想采样(采样脉冲宽度 $\tau = 0$)是不可实现的，也就是说，采样频率并不是可以无限提高的，因此抗混叠滤波器的设计是必须的。

如选用八阶开关电容椭圆滤波器集成芯片 MAX293，使对低通截止频率的控制转化为对芯片输入时钟频率的控制，其主要性能参数如下：

- 八阶低通椭圆滤波器。
- 频率可调范围 0.1 Hz~20 kHz。
- 无须外连电阻或电容。
- 可使用内部或外部时钟。
- 输入时钟与低通截止频率比值为 100:1。
- $\pm 5\text{ V}$ 供电，可对正、负极性输入信号滤波。

抗混叠滤波器参考电路如图5.5所示。

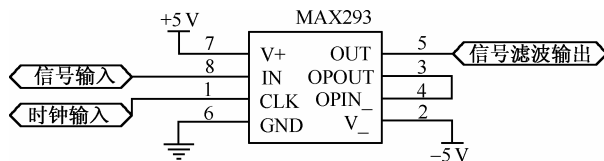


图 5.5 抗混叠滤波器参考电路

3. A/D转换器的选取

A/D 转换器的选择应综合分辨率、精度、灵敏度、输入信号特征等因素，采集速度和高分辨率的统一考虑和兼顾是设计选择的关键，这在5.2.3节中已详细介绍过。根据任务的要求，可供选择的 A/D 芯片有多种，现以 16 位高性能的串行 A/D 转换器 MAX195 为例做出设计分析。

MAX195 是 Maxim 公司的 16 位逐次逼近式 A/D 转换器，具有转换速度快、精度高、功耗低等特点。其主要性能如下：

- (1) $0 \sim V_{REF}$ 单极性输入和 $-V_{REF}$ 、 $+V_{REF}$ 双极性输入；
- (2) 内置采样 / 保持电路；
- (3) 线性误差小：0.003%FSR；
- (4) 转换时间：9.4 μs ；
- (5) 信噪比：90 dB；
- (6) 三态串行数据输出，单极性时输出二进制码，双极性时输出偏移二进制码；
- (7) 关闭模式下最大耗电仅为 10 μA 。

图5.6所示为 MAX195 的引脚图。MAX195 的部分引脚的作用如下：

- BP/UP/ $\overline{\text{SHDN}}$ ：双极/单极/关闭三态输入选择端。接地时为关闭模式；接+5 V 时为单极性输入；悬空时为双极性输入。
- CLK：转换时钟(125 Hz~1.7 MHz，占空比为 25%~75%)输入端。
- SCLK：串行时钟输入端，用在两次模数转换之间输出数据，最大可达 5 MHz。
- EOC：转换结束信号输出端，低电平有效。

- CS: 片选端，低电平有效，高电平时，输出 DOUT 为高阻态。
- CONV: 转换开始信号输入端，低电平有效，在 CONV 下降沿出现后模数转换开始。
- RESET: 复位信号输入端，低电平有效。当 RESET 变低时，MAX195 内部逻辑电路复位并停止工作，当 RESET 的上升沿出现后，MAX195 恢复正常工作。
- REF: 参考电压输入端，0~5 V。
- AIN: 模拟信号输入端，0~ V_{REF} 或 $\pm V_{REF}$ 。

MAX195 内有一个逐次逼近寄存器(SAR)，用以将输入的模拟信号转变为 16 位二进制数码，然后以串行方式输出，它的内部结构如图 5.7 所示。MAX195 在双极性模式下的编码方式如表 5.1 所示。

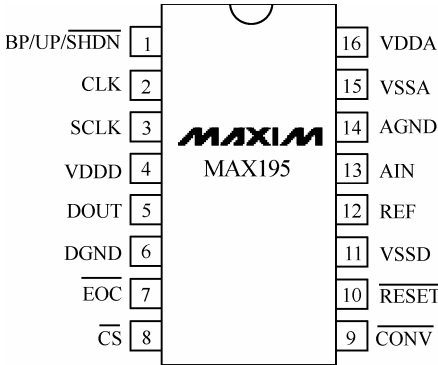


图 5.6 MAX195 的引脚图

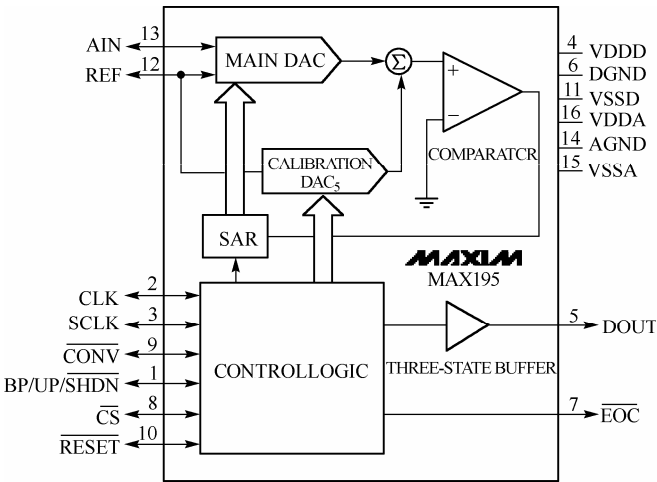


图 5.7 MAX195 的内部结构

表 5.1 MAX195 在双极性模式下的编码方式

输出数字量	对应输入电压值
1111111111111111	$+V_{REF}$
1111111111111110	$+V_{REF} - 1 \text{ LSB}$
...	...
1000000000000000	0 V
0111111111111111	-1 LSB
...	...
0000000000000001	$-V_{REF} - 1 \text{ LSB}$
0000000000000000	$-V_{REF}$

数据可以在转换期间以 CLK 时钟频率(最大 1.7 MHz)输出，也可在模数转换结束后以 SCLK 时钟频率输出，此时最大输出速率可达 Mb/s，输出时高位(MSB)在前。使用时应确保模数转换开始信号 CONV 在转换时钟 CLK 为低时出现，并且至少保持 40 ns 的时间。当 CONV 变低时，开始模数转换。转换过程必须由转换时钟 CLK 同步，两次转换之间至少应有三个转换时钟 CLK 的等待时间。当

- 低功耗：1.5 mW。

MAX542 在双极性模式下的编码方式与 MAX195 兼容(如表 5.2 所示),硬件接口和软件处理特别方便,配合 MAX195 可以实现直通模式的数据采集与回放,这一点在设计及系统检测时十分重要。利用 MAX542 实现模数转换的典型接法如图 5.9 所示。

表 5.2 MAX542 在双极性模式下的编码方式

输入数字量	对应输出电压值
1111111111111111	+1 LSB×32767
1111111111111110	+1 LSB×32766
...	...
1000000000000000	0 V
0111111111111111	-1 LSB
...	...
0000000000000001	-1 LSB×32767
0000000000000000	-1 LSB×32768

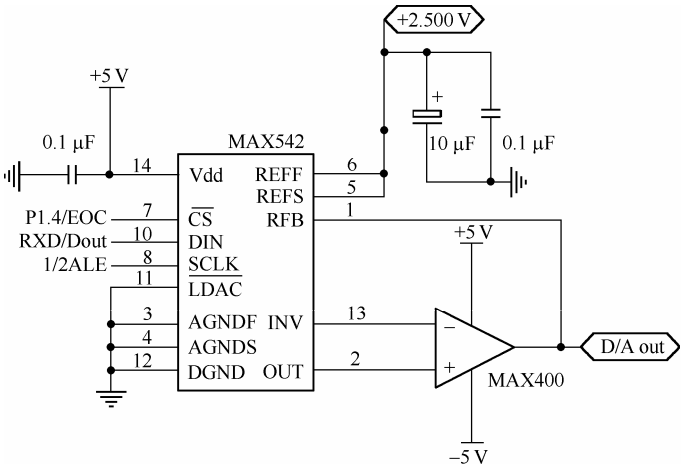


图 5.9 MAX542 实现模数转换的典型接法

6. D/A转换器的后向平滑滤波电路设计

D/A 转换器的后向平滑滤波可以去除 DAC 输出波形的高频噪声,保证输出波形平滑、稳定。这里对滤波器的阶数设计要求不高,普通的二阶低通滤波即可实现很好的滤波效果。

7. 基准源的设计

5.2.3 节已对基准电压源的设计原则及重要性做出了说明。

因此,电路中 A/D 转换器和电压基准源的电源都必须进行低阻驱动的稳压处理。基于上述考虑,可以选择稳压芯片 MAX874 为 MAX195 提供稳定的 4.096 V 参考电压(如图5.11 所示),MAX873 为 MAX542 提供稳定的 2.500 V 参考电压(如图5.12 所示),而 MAX874 和 MAX873 的 5 V 电源电压由 MAX883 提供(如图5.10 所示),这样可以保证基准源向 A/D 转换器提供足够精度的基准参考电压。

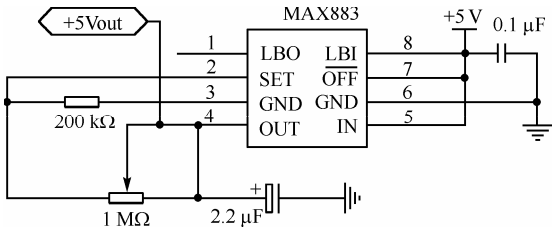


图 5.10 MAX883 产生 5.000 V 电压基准源

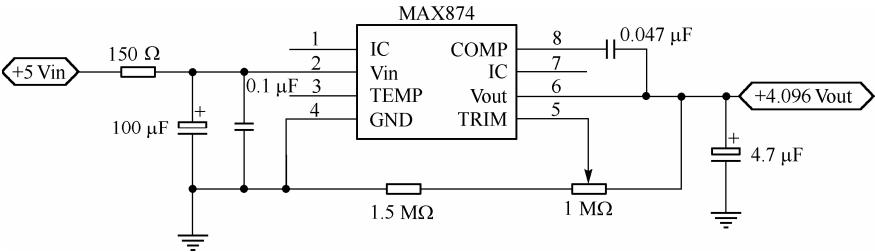


图 5.11 MAX874 产生稳定的 4.096 V 参考电压

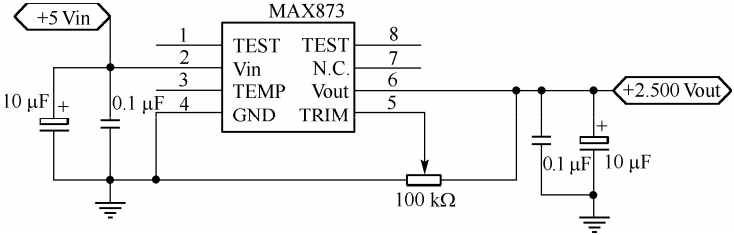


图 5.12 MAX873 产生稳定的 2.500 V 参考电压

8. 串并转换电路设计

由于选用的 MAX195 为 16 位串行输出 A/D 转换器,在双极性模式下,与 16 位 D/A 转换器 MAX542 编码兼容。直通方式可以不经单片机传送数据,直接通过少量的控制口线实现。但在存储回放方式下,数据必须经单片机存入 RAM,串并转换是必需的。

74LS595 是串行输入并行输出移位寄存器,输出带锁存功能,避免了额外的锁存器开销。将两片 74LS595 串联,即可实现一个 16 位的串行输入并行输出移位寄存器,参考电路如图 5.13 所示。

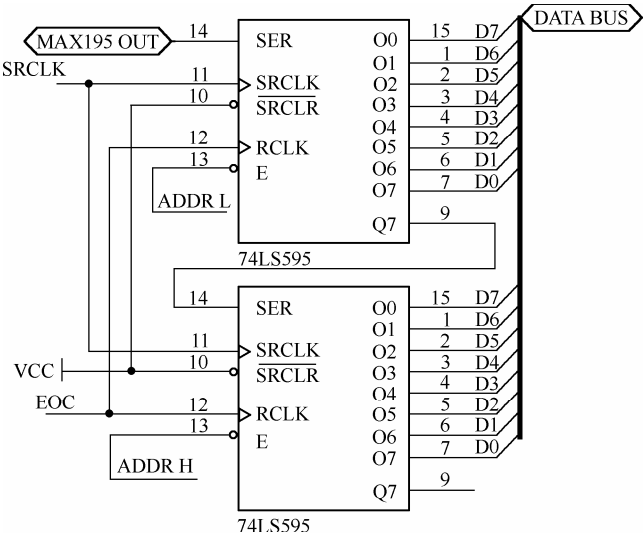


图 5.13 由 74LS595 构成串行输入并行输出移位寄存器的参考电路

5.3.3 工作模式分析与设计

系统设计任务中包括直通方式和存储回放方式。在直通模式下,A/D 输出的采样数据直接送到 D/A 进

行回放；而在存储回放模式时，先将 A/D 输出的采样数据存储，等按下回放键后，再逐点将数据送到 D/A 进行回放。

转换的作用是将连续模拟量 V_i 转换成相应的离散的数字量 D_i ，以便后续的数字系统进行分析 and 处理。在这个过程中，转换的数字量与模拟输入量之间存在对应的函数关系，即

$$D_i = f(V_i) \tag{5.7}$$

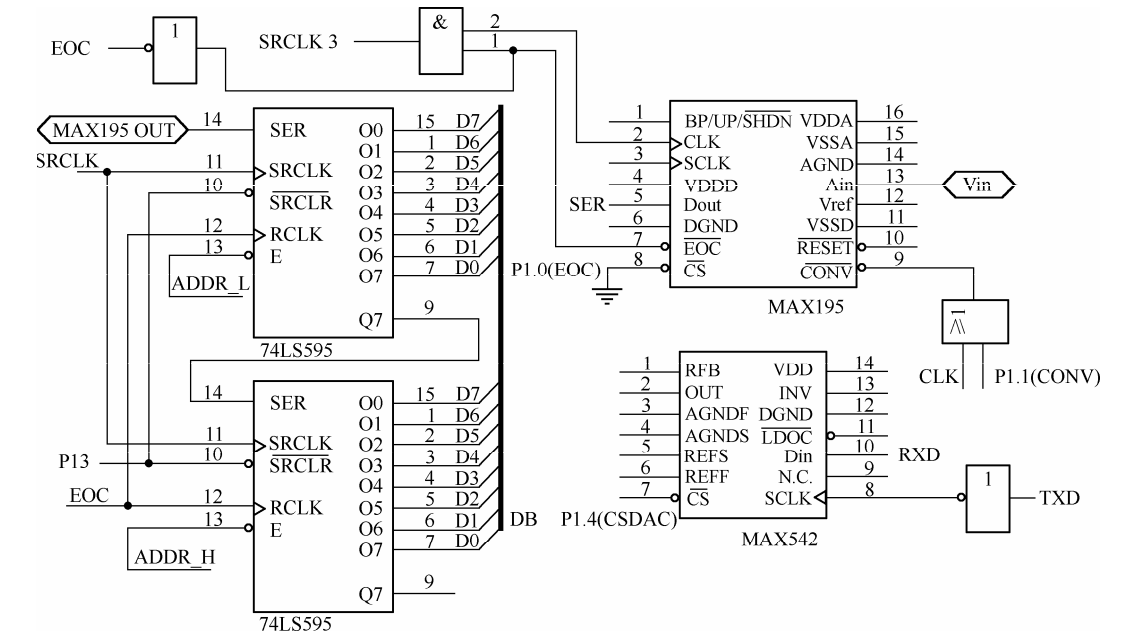


图 5.15 存储回放方式下 ADC、DAC、单片机控制信号连接图

线性转换关系的 A/D 转换器采用的量化电平是等间隔的，转换中将模拟量以量化电平的大小进行数字编码，其数字量的大小与输入模拟量成线性正比关系，如图 5.16 所示。在理想的转换关系中，仍然存在 $\pm 1/2$ LSB 的量化误差，表明处于这个 $\pm 1/2$ LSB 的误差范围内的模拟量(如图 5.16 中的 A~B 之间的模拟量)均产生相同的数字量。其原因在于 ADC 的有限字长，因此固有的量化误差是不可避免的。

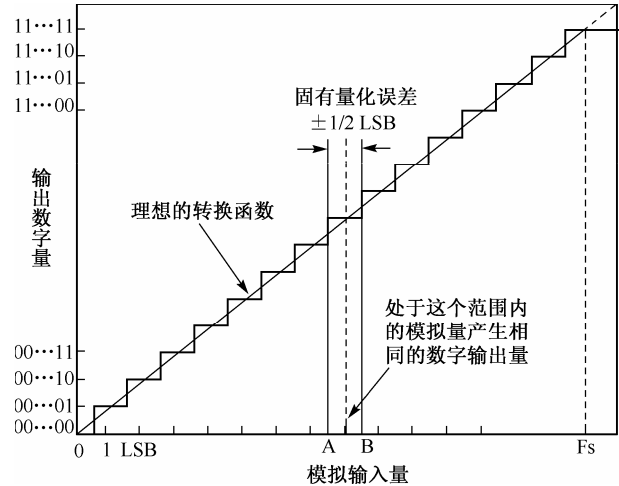


图 5.16 A/D 转换器理想的线性转换关系

在实际应用中，A/D 转换器一般都存在零点误差(又称为失调误差)、增益误差及线性误差，它们称为 A/D 转换器的三个基本误差，如图 5.17 所示。具体分析如下：

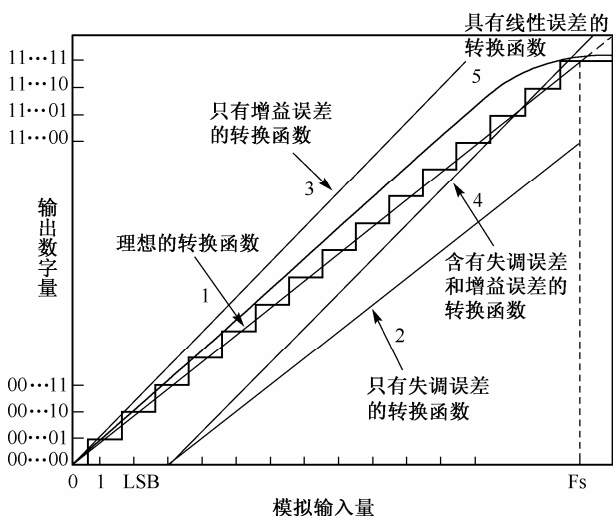


图 5.17 A/D 转换器的失调误差、增益误差和线性误差

(1) 零点误差。其转换函数是一条与理想函数平行但起点不在坐标原点的直线，这表明当输入模拟信号为零值时，由于前置通道内部的各个电路，如输入放大器的失调误差、滤波器环路的电平迁移、电源电压等因素的影响，使得转换的数字量不对应于零值，如图 5.17 中的曲线 2 所示。

(2) 增益误差。其转换函数则是一条通过坐标原点但不通过满刻度点的直线，这表明由于某种原因(如参考电压、增益电阻值等)使得 A/D 转换器实际的转换函数按比例偏离了理想的关系曲线，如图 5.17 的曲线 3 所示。

(3) 线性误差。当 A/D 转换器的零点误差和增益误差均已经调整为零时，转换器实际的传输函数与理想的传输函数之间的最大偏差称为线性误差。线性误差可以出现在转换特性曲线范围内的任何位置，如图 5.17 中的曲线 5 所示。线性误差又分为积分非线性 and 微分非线性。积分非线性是指在动态情况下 A/D 转换器实际转换特性曲线与理想特性曲线之间的偏差，即实际转换码与理想转换码之间的偏差。微分非线性是指在动态情况下 A/D 转换器实际转换特性的码宽(1 LSB)的相对偏差，即某个码对应的模拟输入电压宽度与理想量化电平的偏差。

一般而言，零点误差(失调误差)及增益误差和 A/D 转换器的偏置参数有关，如系统参考电压、与 A/D 一体化设计的缓冲放大器、增益电阻值及滤波器等的设计或设置不当，使实际的转换函数偏离了理想的关系曲线。线性误差是由 A/D 转换器本身的品质因素所决定的。由此可看出，尽管实际的 A/D 转换器具有这么多的误差，却又几乎都能通过外部电路进行调节。虽然线性误差是不可调节的，但可以对线性误差进行优化，以达到实际应用中尽可能小的误差。

2. 误差的调整方法

对于 A/D 转换器而言， $1/2$ LSB 的量化误差是理想误差。对于零点误差的调整可采用以下方法。

(1) 在系统输入端馈入一个 $+1/2$ LSB 的稳定直流电平信号，使输入信号处于量化电平的边界上，一般可以输入一个直流零电平。

(2) 启动 A/D 转换，开采样、量化进行测量，重复 50 次，由于输入信号处于量化电平间隔的边界上，噪声(高斯噪声)使得 A/D 转换器的真实输入信号在此边界上上下下摆动，而导致了输出数字量在 0 和 1 之间跳动。若转换过程是平稳的随机过程，即输出数字量的高位相同，仅最低位在 0 和 1 之间取值的概率大致相当，此时可认为 A/D 转换器处于理想状态。

(3) 由测量误差理论可知，就测量而言，有限次测量下的测量性能可采用算术平均值的标准差来

度量,虽然算术平均值的标准差随测量次数 n 的增大而减小,但减小速度要比 n 的增长慢得多,因此,实际测量中 n 的取值并不大,一般在 10 到 50 之间。基于此调整的依据如下:

$$\bar{D} = \frac{1}{n} \sum_{i=1}^n D_i \quad (5.8)$$

$$\Delta D = \sqrt{\frac{1}{n} \sum_{i=1}^n (D_i - \bar{D})^2} \quad (5.9)$$

式中, D_i 为第 i 次的转换结果; $\bar{D}$ 为多次转换的平均值(算术平均值); n 为转换次数; ΔD 为标准偏差值。如果在调整中 $\bar{D} \approx 0.5$, 并且 $\Delta D \approx 0.5$, 此时可以认为已经把转换函数的起点调节到了坐标原点。

增益误差的调整方法及原理同上,仅将模拟输入量改为 $FS-1/2 \text{ LSB}$ 即可。其判定方法与零点误差的判定方法也相同。一般而言,误差调整时应先调整零点误差,然后调整增益误差,调整后得到转换特性曲线两端被校正的转换函数,而线性误差完全被保留了。如前所述,线性误差是不可调整的,但可以优化,优化的方法主要有两种:一是根据实际模拟信号的变化范围,有选择地调节零点误差和失调误差以改变转换函数曲线的位置,使关心的某一输入范围内具有更小的线性误差;二是在已知线性误差范围及大小的情况下,采用软件插值校正、线性拟合的方法来减小误差的影响。

在实际应用中,对于有效字长的测定十分重要。所设计的系统必须具有可测性,而有效字长的测定需逐点测量,其方法和上述的误差调整的方法是相同的。需要说明的是,所有的测量调整都必须在器件热稳定之后进行,显然工作量是不小的。误差调整及有效字长的测定其意义在于:系统中总的不确定度应是已知的,且是可控制的,在给定条件下,误差应可能是最小误差,以最大限度地发挥 ADC 的性能。

5.3.5 系统软件设计

系统由上位机 PC 实现主控,下位机 89C51 实现对数据的采集、存储与波形回放。PC 和 89C51 之间的通信是通过 RS232 串行口来实现的。在上位机 PC 的控制下,下位机 89C51 完成采样速率、采样点数的初始化,并在 PC 的命令下开始采样。采样结束后,在上位机的控制下将采样数据送回 PC 存储和显示。

直通方式下,先向 CONV 控制口写一个负脉冲,启动 A/D 转换,再查询 EOC 端口,一旦检测到 A/D 转换结束,再向 CONV 控制口写负脉冲。重复上述过程,实现对输入信号的实时采集回放。

存储回放方式下,要将 ADC 转换的数据依次存储在 RAM 区连续的单元内,再依次读出并送 DAC 回放。

5.3.6 系统抗干扰措施

数据采集系统对电路中存在的干扰特别敏感,抑制系统干扰应从软件和硬件两个方面考虑。

1. 硬件抗干扰

在硬件抗干扰中,最重要的设计是阻止噪声耦合到输入端。因此,合理的系统电路布局是基本要求,也是最重要的要求。具体方法:数字信号不能与模拟输入信号通路平行,而应当尽可能地远离输入电路。设置电源去耦电路以减少电源干扰。电路应有良好的接地,数字地与模拟地相连以减小接地线上的噪声电平。为优化接地和电源结构,推荐使用多层印刷版(PCB)。采用平板接地和平板电源的好处是:(1)减小了信号回路的环绕;(2)减小了接地通路和电源通路的阻抗;(3)减小了电源平面、PCB 介质和地平面之间形成的分布电容。因此系统的电磁耦合(EMI)减小,可明显改善系统的整体性

能。作为训练应考虑成本等因素，不使用 PCB 也可以实现设计要求。在设计中，A/D 转换启动信号可采用 74F04 反相器驱动，以防止不良脉冲前沿对转换结果产生影响。输入缓冲放大器尽可能地靠近 MAX195，它们之间的连线尽可能地短而粗，基准源也尽可能地靠近 MAX195，并考虑温度升高的影响。系统设计时采用集成度高的芯片，以减小单元电路之间的噪声耦合。

2. 软件抗干扰

为了减少干扰信号对采样值的影响，提高采样数据的可靠性，对采样值需要进行判断，即所谓的数字滤波处理，对明显偏大的采样值可以通过程序去除，提高数据采集的精度。软件抗干扰的方法很多，读者应根据系统设计的目标任务及实际应用环境做出选择。

5.4 语音存储与回放系统的设计

1. 设计任务

设计并制作一个数字化语音存储与回放系统，其示意图如图 5.18 所示。

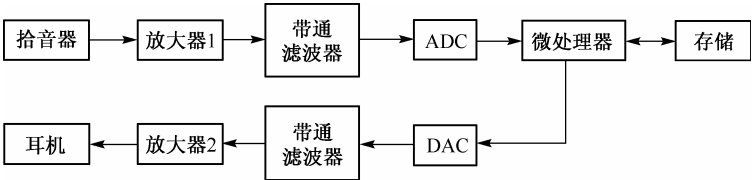


图 5.18 数字化语音存储与回放系统示意图

2. 设计基本要求

- ① 放大器 1 的增益为 46 dB，放大器 2 的增益为 40 dB，增益均可调；
- ② 带通滤波器：通带为 300 Hz~3.4 kHz；
- ③ ADC：采样频率 $f_s = 8\text{ kHz}$ ，字长位 = 8 位；
- ④ 语音存储时间 $\geq 10\text{ s}$ ；
- ⑤ DAC：变换频率 $f_c = 8\text{ kHz}$ ，字长位 = 8 位；
- ⑥ 回放语音质量良好。

3. 设计提高部分要求

在保证语音质量的前提下，

- ① 减少系统噪声电平，增加自动音量控制功能；
- ② 语音存储时间增加至 20 s 以上；
- ③ 提高存储器的利用率(在原有存储容量不变的前提下，提高语音存储时间)；其他(例如校正

$\frac{\pi f / f_s}{\sin(\pi f / f_s)}$ 等)。

说明：不能使用单片语音专用芯片实现本系统。

5.4.1 系统分析与参数计算

本节的设计任务是设计一个数字化语音信息的采集与回放系统，其主要技术指标为：采样频率 $f_s = 8\text{ kHz}$ ，编码字长为 8 bit，回放频率(变换频率) $f_c = 8\text{ kHz}$ ，语音存储时间 $\geq 10\text{ s}$ ，提高部分要求语音存

储时间 ≥ 20 s 以上。

从题目的技术指标要求看,采样频率、回放速率及编码字长、采样量化精度均不构成设计难点,而且题目已给出了该系统包括子系统级的总体结构图,硬件设计的难度相对也不大。只有语音存储时间的指标要求有可能成为主要难点。具体分析如下:

(1) 语音信号的物理特性及特点:①清晰度;②可懂度;③语音信号的频谱;④相关性;⑤统计特性。

(2) 语音信号采样合成的参数指标

$$f_s \times N = BR \quad (5.10)$$

式中, f_s 为采样频率; N 为采样值的编码位数、字长; BR 为比特率(也称为数码率),是指在单位时间(1 s)采集合成语音信号时所对应的存储空间的大小,或者说,比特率是存储 1 s 语音信号所需的位数(bit)。

由原始设计指标: $f_s = 8$ kHz, 编码字长为 8 bit, 有

$$f_s \times N = 8 \times 10^3 \times 8 = 64\,000 \text{ b/s}$$

即为 64 kb/s, 而存储容量 C 为

$$C = T \times BR \quad (5.11)$$

式中, T 为存储时间。根据以上的计算可知: 1 s 的合成语音时间所对应的存储容量为 8 KB, 问题由此产生。MCS-51/52 系统(单片机)的最大寻址空间为 64 KB, 而 10 s、20 s 的语音信号存储时所对应的存储空间使常规的 MCS-51/52 系统显然不能满足, 也就是说, 本设计任务要求系统必须有 64 kb/s 的数据传输通道, 这一数码率使存储空间的扩展和有效利用成为了本系统设计的主要难点。可以采用的解决方案有

(1) 多级译码电路进行存储空间扩展。

(2) 信源编码压缩以提高存储空间的利用率。

问题是: 存储空间的扩展在技术实现上并无大的问题, 实现手段很多, 可采用的存储器种类也很多(如 SRAM 和 FLASH 等), 但仅限于存储器扩展不能体现设计水平和价值, 尤其是对于数字化系统而言, 采用新技术、新知识应成为设计理念, 因此信源的压缩编码应成为选择方案。

从广义上看, 编码分为两类:

(1) 信源编码—提高信号传输和存储效率, 压缩数字信号的比特率, 又称为压缩编码。

(2) 信道编码—提高传输可靠性, 又称为可靠性编码。

从通信的角度看, 编码是对信号进行处理、使其变换成适合于信道传输的形式, 在语音信号的数字化处理中, 信源编码的目的是:

(1) 使得相同的信道容量能传输更多路的语音信号。

(2) 提高存储空间的利用率, 即在给定存储容量的条件下实现较大的数字化语音信号的合成记录时间。

5.4.2 数据编码与存储

由系统设计的难点分析得出: 在系统级设计中应首先进行有关数据编码的方案讨论, 考虑到系统方案的对比检测需要, 验证编码效率, 在系统级设计时应考虑三种以上的语音存储回放模式, 以方便对比检测。下面就三种模式分别进行讨论。

1. 直通PCM模式

脉冲编码调制(PCM)是指输入模拟信号在时间上进行采样,每个采样再进行量化,并以数字信号进行存储及传输。所谓直通模式,即为不经处理的即采即放模式,单片机将 ADC 采样量化值经简单处理后直接送 DAC 回放。其特点是:实时性强,采样速率高,回放失真小(可用于检测系统是否能正常响应,前、后向通道的硬件设计是否正确)。

2. 存储回放PCM模式

存储回放 PCM 模式将 8 位 ADC 采样量化值不经压缩编码处理,直接存储在数据存储器中,回放时送 8 位 DAC 回放。其特点是:采样速率较高,回放失真小,但由于未对采样数据进行压缩处理,存储时间短(可用于对比检测编码效率)。

3. 差分脉冲编码调制(DPCM)模式

在介绍 DPCM 编码之前,应对增量调制(DM)模式进行简述。

(1) 增量调制(DM)

增量调制原理框图如图5.19所示。从图5.19可以看出,增量调制可看成是差分脉冲编码器,即当差分脉冲编码调制系统的量化字长为 1 时,可得到 DM 系统。设原始语音信号(取样)为 $x(n)$,因语音信号有很大的冗余度,可在量化和编码之前去除冗余信息,最简单的方法是取相邻两取样值之差,得到差值信号 $d(n)=x(n)-x(n-1)$, $d(n)$ 是去除 $x(n)$ 中冗余信息后得到的结果,然后对差值信号 $d(n)$ 进行量化编码。在增量调制中,被量化的不是输入语音信号,而是预测信号 $d(n)$ 。

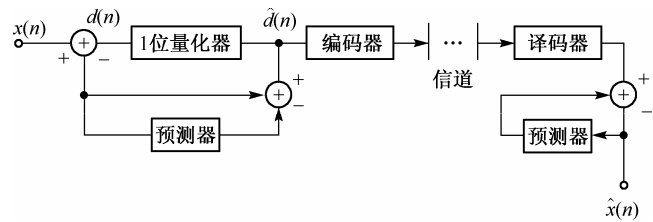
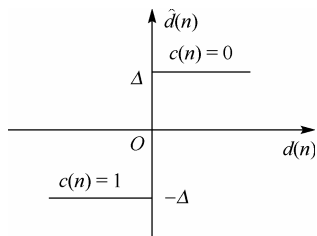


图 5.19 增量调制原理框图

由于 $d(n)=x(n)-x(n-1)$, 令 $x(n-1)=\Delta$, 即设起始信号值先前的预测信号值为 Δ , 此时调制非常简单,如图5.20所示。



$$d(n)=\begin{cases} \Delta & \text{当 } d(n)>0 \text{ 时} \\ -\Delta & \text{当 } d(n)<0 \text{ 时} \end{cases}$$

(5.12)

只有两个量化电平(+ Δ 和- Δ)这是增量调制的最大优点。因此,量化器输出信号可以用一个二进制符号表示,例如,用“0”表示 Δ ,用“1”表示- Δ ,则预测误差信号为

图 5.20 线性增量量化特性

$$d(n)=x(n)-\hat{x}(n)=x(n)-a_1\hat{x}(n-1)$$

(5.13)

式中, a_1 为一阶线性预测器参数。由此,增量调制的结果为由 0 和 1 组成的序列,恢复语音信号时采用图5.19所示的译码器,可以得到一个由上升和下降阶梯波构成的阶梯波信号,如图 5.21 所示。

从图5.21中 a, b 点可以看出:当语音信号大幅度变化时,阶梯波的升或降可能跟不上信号的变化而产生滞后,造成斜率失真,失真期间的码字为一串 1 或一串 0。从图5.21中 c 点可以看出:信号变

化缓慢时, Δ 和 $-\Delta$ 发生概率相等、交替出现, 恢复信息等幅振荡, 造成颗粒噪声。由图 5.21 可以看出, 增量调制的噪声主要有以下两种: ① 斜率过载失真(惰性失真); ② 颗粒噪声。

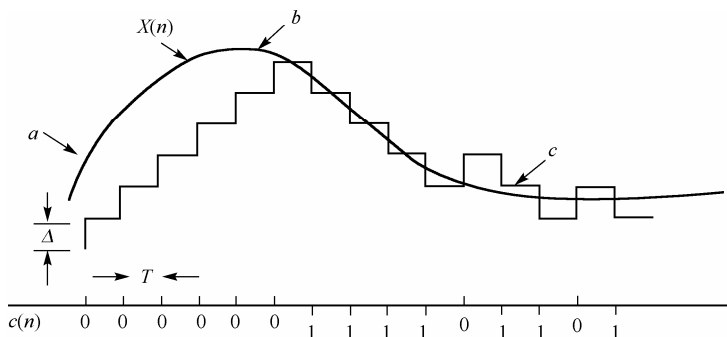


图 5.21 增量调制得到的恢复信息(均匀量化)

解决方法是: 对于斜率过载失真, 其原因显然是量化电平 Δ 过小, 使得阶梯波的上升和下降斜率小于语音信号的最大变化斜率, 为使噪声减小, 应有

$$\frac{\Delta}{T} \geq \max \left| \frac{dx(t)}{dt} \right|$$

式中, T 为取样时间; $x(t)$ 为原始模拟语音信号。由于采样速率已定, T 为常数, 所以只有加大量化电平 Δ 才能使斜率过载失真减小。但问题是: 对颗粒噪声而言, 仅当量化电平 Δ 减小才能使颗粒噪声减小。矛盾的原因是 DM 方式是一位编码, 对不变化的状态不进行符号分配, 即残差 e_n 近似为零, 这时取 $+\Delta$ 还是 $-\Delta$ 难以确定, 由于是均匀量化, 量化电平固定, 不能根据语音信号的幅值变化信息相应地改变 Δ 幅值, 跟随特性差。

为了解决问题, 应抓主要矛盾。因语音信号是模拟信号, 其连续性决定相邻两个采样值之间的 Δ 值不可能太大。假设 Δ 较大, 则易懂度下降, 而且人的听觉器官不易察觉斜率过载失真, 而颗粒噪声产生影响较大, 所以一般要折中考虑。

(2) DPCM 模式

DPCM 是增量调制的一种改进, 它采用将差值量化应用于脉冲调制。它适当地降低了增量调制的数据压缩比(DM 方式可达 8 倍的压缩比, DPCM 为 2 倍), 是一种比较成熟的压缩编码技术。由于 DPCM 方式是对两个采样值之间的差分信号进行编码, 而相邻两个采样信号由于语音信号的相关性较强其采样值不会突变, 所以 DPCM 方式编码的精度要比 DM 方式高。

现给出具体的编码方法: 利用 M 位($M=4$)的变量 COMP 记录前、后语音信号采样值 $s(n-1)$ 和 $s(n)$ 的比较结果(差分值), 其中 COMP 的最高位为符号位。记录 $s(n-1)$ 和 $s(n)$ 的大小, 其余位记录 $s(n-1)$ 与 $s(n)$ 的差距, 此时 COMP 的 4 位分别为 F1ag, D_2 , D_1 , D_0 。

$$\text{COMP} = \begin{cases} 1111 & 6\Delta < s(n) - s(n-1) \leq 7\Delta \\ \text{N} & \text{N} \\ 1001 & \Delta < s(n) - s(n-1) \leq 2\Delta \\ 1000 & 0 < s(n) - s(n-1) \leq \Delta \\ 0000 & -\Delta < s(n) - s(n-1) \leq 0 \\ 0001 & -2\Delta < s(n) - s(n-1) \leq -\Delta \\ \text{N} & \text{N} \end{cases} \quad (5.14)$$

0111 $-8\Delta < s(n) - s(n-1) \leq -7\Delta$

存储时，将第一次采样值 Data(1) 和以后每次比较的结果 COMP 存储下来；回放时，前后两次回放数据的关系为

$$\text{Data}(n) = \begin{cases} \text{Data}(n-1) + (2^2 \times D_2 + 2 \times D_1 + D_0) \Delta & \text{Flag} = 1 \\ \text{Data}(n-1) - (2^2 \times D_2 + 2 \times D_1 + D_0) \Delta & \text{Flag} = 0 \end{cases}$$

在设计中，应根据信号的幅度和采样速率确定 Δ 的大小，即由量化阶距来确定 D_2, D_1, D_0 分别为 8 位 A/D 采样值的第几位。由于已将增量值分为 Q 个等级，用 K 位二进制代码表示差分值，并考虑到还有一位符号位，所以有 $Q = 2^{K+1}$ 。如图 5.22 所示， $\Delta = 2^\circ \text{LSB} = \text{LSB} = 20 \text{ mV}$ (8 位 A/D，最小量化阶距)，相邻两次采样量化值之间的最大差值(最大量化阶距)为 $7\Delta = 7 \times 20 \text{ mV} = 140 \text{ mV}$ ，显然量化电平 Δ 偏小，可能产生隋性失真。

若如图 5.23 所示，则 $\Delta = 2^2 \text{LSB} = 4 \times 20 \text{ mV} = 80 \text{ mV}$ (8 位 A/D)，最大差值为 $7\Delta = 7 \times 80 \text{ mV} = 560 \text{ mV}$ ，量化电平适中。

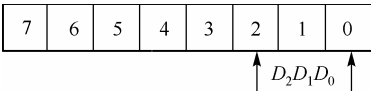


图 5.22 DPCM 编码位数的选取(低 3 位)

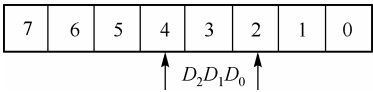


图 5.23 DPCM 编码位数的选取(2~4 位)

脉冲编码调制的优点是：具有较高的数据压缩比，实现比较简单，信噪比损失较小。采用上述方法可实现的 DPCM 压缩编码示意图如图 5.24 所示。

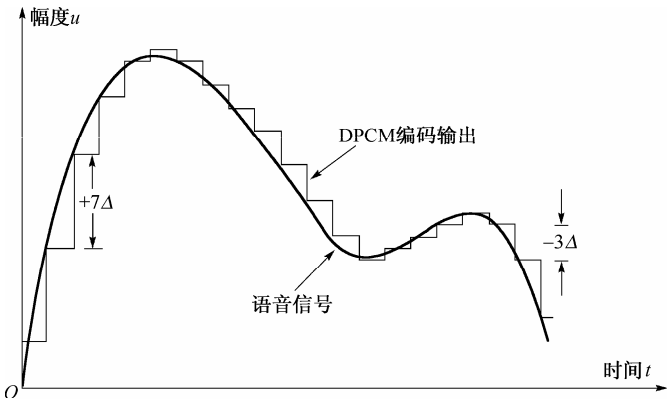


图 5.24 DPCM 压缩编码示意图

5.4.3 系统整体设计框图

系统整体设计框图如图 5.25 所示。

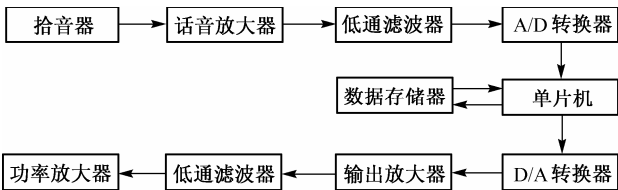


图 5.25 系统整体设计框图

5.4.4 系统各模块电路的设计方案

1. 前向通道设计

前向通道由话筒、语音信号阻抗匹配、语音放大器、低通滤波器和 A/D 转换器等组成。

(1) 话筒

声电信号转换通过磁波式话筒实现。它具有全方位、灵敏度高、噪声小等诸多优点。输出电信号幅值为毫伏级。设计电路时必须要给话筒输入端加载直流偏置电平，才可以正常工作。

(2) 语音信号阻抗匹配

本单元设计的目的是使信号源的输出阻抗足够小，从而能够向系统提供最大功率和稳定的信号，降低系统对话筒输出的要求。设计中，采用低噪声精密集成运算放大器 OP37 构成简单的跟随电路即可实现很好的匹配效果，具体电路如图 5.26 所示。

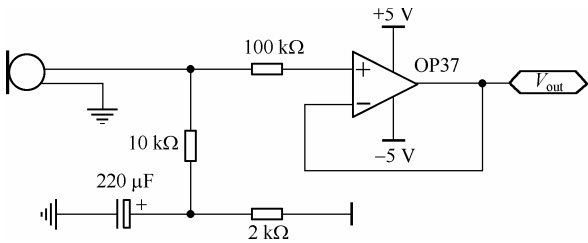


图 5.26 语音信号阻抗匹配器电路

(3) 语音放大器

话筒输出电压为毫伏级，为保证采样精度，必须经过上百倍的放大才能送 A/D 采样。设计两级放大，第一级为自举交流电压放大，对输入交流信号固定放大 16 倍，第二级为同相放大，放大增益由 50 kΩ 的电位器调节。此外，在第二级同相放大器上加直流电平调节电路，抬升幅度由 50 kΩ 的电位器调节，可以方便地调节输入信号的直流分量。两级放大电路可以将输入语音信号放大 150~200 倍。语音放大器电路如图 5.27 所示。

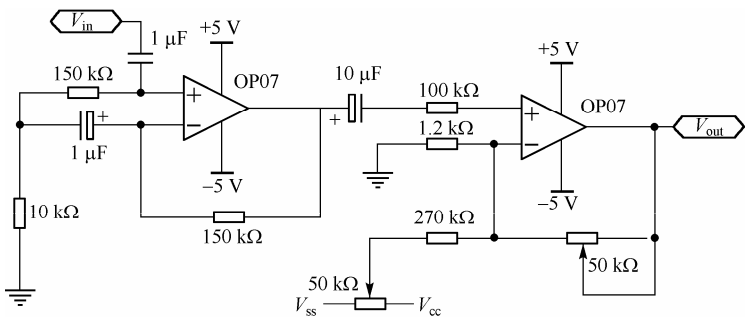


图 5.27 语音放大器电路

(4) 低通滤波器

为防止频谱混叠失真和提高信噪比，必须对输入语音信号进行低通滤波处理。对语音信号的低通滤波截止频率为 3.4 kHz。设计中可使用 TLC04 芯片设计四阶低通滤波器，其特点是截止频率可调，电路简单，

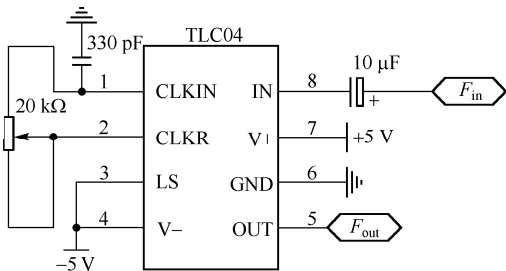


图 5.28 低通滤波器电路

使用方便。低通滤波器电路如图5.28 所示。

(5) A/D 转换器

设计中考虑语音信号的最高频率 $f_{\max} = 3.4\text{ kHz}$ ，根据奈奎斯特采样定理，采样频率 $f_s \geq 2f_{\max}$ 。如果取采样频率为 8 kHz ，则理论上采样周期 $T_s = 125\text{ }\mu\text{s}$ ，即可无失真地恢复原语音信号。可以选用的 A/D 转换器的型号很多，设计可采用 8 位逐次比较型 A/D 转换器 ADC0804，由于 ADC0804 完成一次模数转换需要 $100\text{ }\mu\text{s}$ 的时间，因此在 ADC0804 之前还必须加一级取样 / 保持 (S/H) 电路，可选用 LF398 集成 S/H 芯片。在实际的 S/H 电路中，使其能正常工作的主要因素还取决于保持电容 C_H (S/H 电路的外部元件) 的选择。 C_H 取值小，跟随性好，但保持性差。 C_H 取值大，保持性好，电压漂移小，但跟随性差。而且 C_H 的绝缘系数也会影响采集工作，应选择低泄漏的电容器。 C_H 的选择应综合考虑保持步长、捕获时间及下垂度等因素。由 LF398 的技术参数可知，选取保持电容 $C_H = 1000\text{ pF}$ 时，捕获时间 $t_{\text{acq}} = 4\text{ }\mu\text{s}$ ，孔径不确定时间 $t_{\text{ap}} = 20\text{ ns}$ 。相比于被处理的 $300\text{ Hz} \sim 3.4\text{ kHz}$ 的语音信号而言，误差可以忽略不计。信号重构所带来的频率和相位误差也可忽略不计。相比而言，本节题目所要求的语音信号的采集对幅度分辨率的要求不高，因此，A/D 转换器及缓冲放大器的设计难度较小。但应注意的是，实际的输入信号是过零的双极性信号，在设计中如选用了输入信号为单极性的 A/D 转换器时，则必须考虑在 A/D 转换电路前设置相应的直流电平变换电路，使过零的双极性信号整体迁移，变换为适合 A/D 转换的单极性信号。如某一 A/D 转换器只能转换单极性信号，输入信号范围为 $0 \sim +5\text{ V}$ ，则直流电平变换电路应使前置信号通路的直流电平相对于基准 0 电平向上迁移 2.5 V 。

(6) $(\pi f / f_s) / \sin(\pi f / f_s)$ 校正

由于平顶采样效应的影响，造成被采集语音信号高频分量的丢失。可采用一阶 RC 网络对高频分量作局部提升，进行近似校正。根据公式 $(\pi f / f_s) / \sin(\pi f / f_s)$ 计算可得，当采样频率 f_s 为 8 kHz 时，在 $f = 300\text{ Hz}$ 处衰减 0.02 dB ，在 $f = 4.3\text{ kHz}$ 处衰减 -4.61 dB ，因此采用一简单的阻容网络实现在 4.3 kHz 处提升约 4.59 dB 即可，如图 5.29 所示。

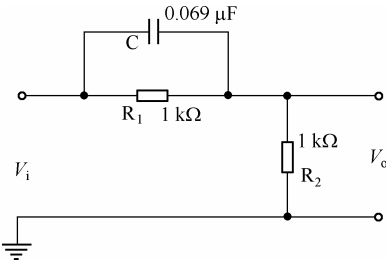


图 5.29 校正曲线的一阶 RC 网络

2. 后向通道设计

后向通道由 D/A 转换器、输出放大器、低通滤波器、功率放大器和扬声器组成。

1) D/A 转换器

同样，可选用的 D/A 转换器的型号也很多，设计可采用 8 位 D/A 转换器 AD7528 或 TLC7528 和与之匹配的放大器实现数据的回放，以上两种 D/A 转换器的指标参数基本相同，建立时间小于 $1\text{ }\mu\text{s}$ 。

2) 输出放大器

为保证回放的音量合适，在后向通道中还需加入放大器，对 D/A 转换后的语音信号放大。设计时可采用简单的同相放大电路。输出放大器电路如图5.30所示。

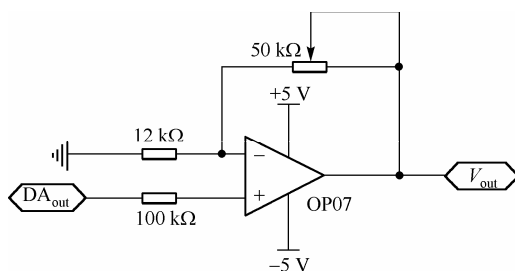


图 5.30 输出放大器电路

3) 低通滤波器

D/A 转换器的后向平滑滤波可以去除 DAC 输出波形的高频噪声, 保证输出语音信号平滑、稳定。这里对滤波器的阶数设计要求不高, 普通的四阶低通滤波即可实现很好的滤波效果, 故仍然使用上面提及的 TLC04 完成。

4) 功率放大器

功率放大器采用 LM386, 单端输入, 增益为 20 倍。功率放大器电路如图 5.31 所示。

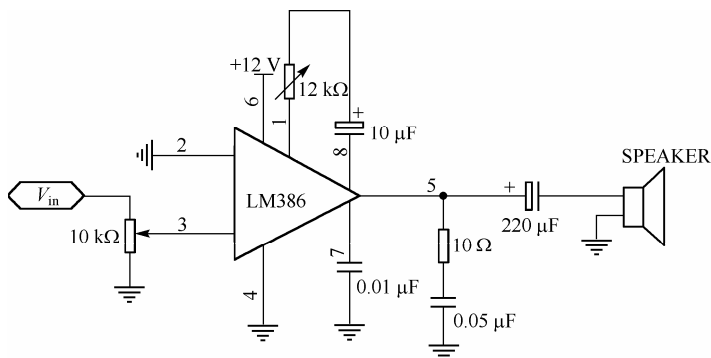


图 5.31 功率放大器电路

5.4.5 三种模式软件设计流程图

直通模式软件设计流程图如图 5.32 所示, 存储回放 PCM 模式软件设计流程图如图 5.33 所示, DPCM 模式软件设计流程图如图 5.34 所示。

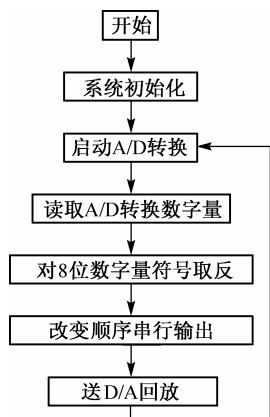


图 5.32 直通模式软件设计流程图

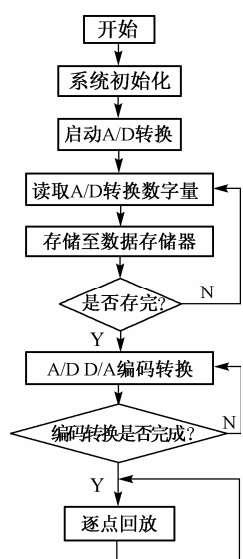


图 5.33 存储回放 PCM 模式软件设计流程图

5.4.6 噪声分析与降噪措施

初步完成的系统在放音过程中会夹杂着明显的噪声，这些噪声源于数字系统的干扰和外界窜入的工频信号干扰。可采取以下降噪措施：

- 模拟地和数字地分开，在电源处一点接地；
- 正、负电源与地之间均采用 10 μF 和 0.01 μF 电容并联用做退耦；
- 数字集成电路芯片的电源接入端接 0.01 μF 电容；
- 话筒和后级之间用射随器隔离；
- 使用低噪声运算放大器。

以上措施可以提高回放语音的清晰度和可懂度，使回放噪声较小。

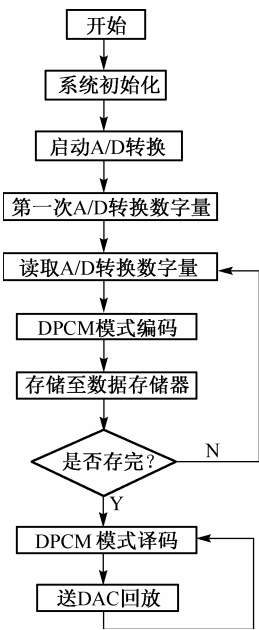


图 5.34 DPCM 模式软件设计流程图

5.5 本章小结

本章首先对数据采集与回放系统的工作原理和设计方法进行了详细的分析，然后以高精度数据采集与回放系统和语音存储与回放系统为例，详细介绍了不同系统性能指标下的数据采集与回放系统的设计方案。在数据采集与回放系统中，A/D 转换器、D/A 转换器的选取和相应电路的设计是重要部分，转换器速率的要求应根据信号对象的变化及转换精度的要求而确定，采集速率和高分辨率的统一考虑和兼顾是设计选择的关键。

数据采集与回放系统的应用非常广泛，而且使用环境也是错综复杂的。在掌握数据采集与回放系统基本原理和设计方法的基础上，实际应用中应根据数据采集系统应用环境和目标任务的不同，按照不同的幅度分辨要求和不同的采集速率要求等因素，以不同目标任务的侧重点、难点为切入口，设计符合实际需求的数据采集与回放系统，并以此掌握复杂电子系统的设计方法，加强逻辑思维训练，提高分析和解决问题的能力。

第6章 数据传输系统的设计

6.1 引言

通信按照传统的理解就是信息的传输和交换,因此,数据采集与传输系统的设计应看成是一个通信系统的设计。在通信系统中,信息(或称为消息)的传递是通过它的物理载体——电信号来实现的,即把信息寄托在电信号的某一参量上,如连续波的幅度、频率或相位,脉冲波的幅度、宽度或位置。按信号参量的取值方式不同,可把信号分为两类:模拟信号和数字信号。通常,按照信道中传输的是模拟信号还是数字信号,相应地把通信系统分为模拟通信系统和数字通信系统。模拟通信系统是利用模拟信号来传递信息,通常需把原始电信号(基带信号)变换成适合在信道中传输的信号,并在接收端进行逆变换。完成这种变换需采用调制器和解调器。而数字通信系统则是利用数字信号来传递信息。数字通信涉及的技术问题很多,其中主要有信源编码与译码、信道编码与译码、数字调制与解调、同步及加密与解密等。目前,无论是模拟通信还是数字通信,在不同的通信业务中都得到了广泛的应用。数据采集的相关内容在第5章已经进行了详细的介绍,本章将主要讨论数据传输技术。

由于数据传输系统的种类很多,涉及到的理论和设计方法也有较大差异。本章以一个8路模拟信号采集与单向传输系统为例,从数据传输系统的基本组成部分——数字调制解调器出发开始进行讨论,详细介绍了基本的数据传输系统的设计方案,其基本思想是通过训练,使读者建立数据采集与传输系统的设计思路,掌握基本的设计方法,从而能够在以后的实践中根据具体的设计需求来实现合理的设计。

6.2 数据采集与传输系统的设计

1. 设计任务

按照图6.1所示的框图设计一个用于8路模拟信号采集与单向传输系统。

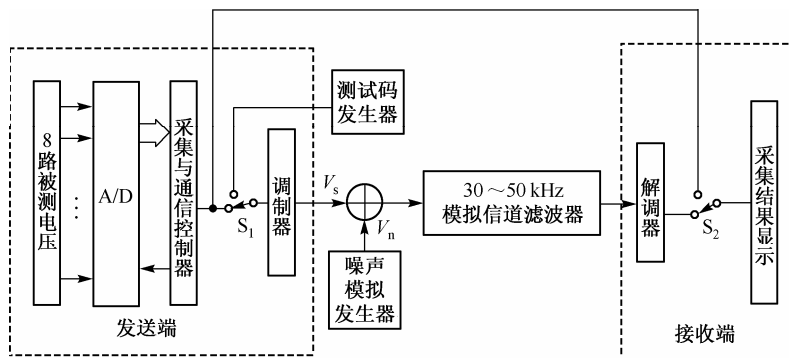


图 6.1 8 路模拟信号采集与单向传输系统方框图

2. 设计基本要求

- (1) 被测电压为 8 路 0~5 V 分别可调的直流电压。系统具有在发送端设定 8 路顺序循环采集与指定某一路采集的功能。
- (2) 采用 8 位 A/D 转换器。
- (3) 采用 3 dB 带宽为 30~50 kHz 的带通滤波器(带外衰减优于 60 dB/十倍频程)作为模拟信道。
- (4) 调制器输出的信号峰-峰值 V_{sp-p} 为 0~1 V 可变, 码元速率为 16 kb/s; 制作一个时钟频率可变的测试码发生器(如 0101...码等), 用于测试传输速率。
- (5) 在接收端具有显示功能, 要求显示被测路数和被测电压值。

3. 设计提高部分要求

- (1) 设计制作一个用伪随机码形成的噪声模拟发生器, 伪随机时钟频率为 96 kHz, 周期为 127 位码元, 生成多项式采用 $f(x)=1+x^3+x^7$ 。其输出峰-峰值 V_{np-p} 为 0~1 V 连续可调。
- (2) 设计一个加法电路, 将调制器输出 V_{sp-p} 与噪声电压 V_{np-p} 相加送入模拟信道。在解调器输入端测量信号与噪声峰-峰值之比 (V_{sp-p}/V_{np-p}), 当比值分别为 1, 3, 5 时, 进行误码测试。
- (3) 在 (V_{sp-p}/V_{np-p}) = 3 时, 尽量提高传输速率, 用上述 (2) 的测试方法, 检查接收数据的误码情况。

6.2.1 系统设计方案分析

数据采集与传输系统的设计难点主要集中在通信模型的建立上, 典型的数字通信系统框图如图6.2所示。对照数字通信系统模型及设计任务的要求, 系统设计中需考虑的问题有

- (1) 8 路数据采集电路的设计;
- (2) 调制方案的确定与电路设计;
- (3) 3 dB 带宽为 30~50 kHz 的带通模拟信道的设计;
- (4) 解调方案的确定与电路设计;
- (5) 噪声源产生电路的设计;
- (6) 时钟频率可变的测试码发生器设计;
- (7) 数据接收端显示系统设计。

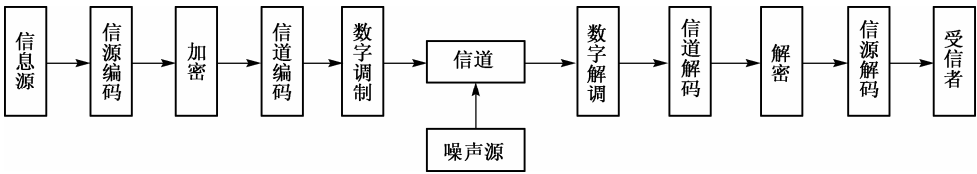


图 6.2 典型的数字通信系统框图

其中调制解调方案的确定与电路设计是设计的难点和切入点。此外, 为实现 8 路数据的采集和单向发送及接收显示, 发送端和接收端都需要使用一片单片机, 以控制数据采集、通信和结果显示。

6.2.2 调制方案的确定与相应数学模型的建立

调制方案的确定首先应分析信道条件和信号信噪比。设计要求系统传输的是数字信号, 码元传输速率为 16 kb/s, 信号被限定在 30~50 kHz 的带宽内。基带信号的带宽为 $B_m = 16\text{ kHz}$, 经调制后的能量主要分布在 $2B_m = 32\text{ kHz}$ 的频带内, 而噪声为窄带白噪声, 经过带宽仅为 20 kHz 的信道后信号与

噪声的能量大致相当。由香农公式 $C = B \lg(1 + S/N)$ 可知, 当信道容量 C 一定时, 增大信道的带宽, 可以降低对信噪比的要求; 反之, 当信道频带较窄时, 可以通过提高信噪功率比来补偿。可计算出: 当信号和噪声幅度的比值为 3:1 时, 信噪比约为 9, 信道传输信息的极限约为 66.5 kb/s; 当信号和噪声幅度的比值为 1:1 时, 信道传输信息的极限约为 40 kb/s。由以上分析可知, 系统设计首先要对传输的数据进行数字编码调制, 而系统的模拟信道的带宽仅为 20 kHz, 码元传输速率为 16 kb/s, 属于典型的高速窄带数字通信。因此, 在选择调制方案时应综合考虑信道和信噪比的情况, 进行相应的理论分析和计算, 从而合理地选择调制方式。

数字调制可分成线性调制和非线性调制两种。二进制振幅键控(2ASK)、频移键控(FSK)和相移键控(PSK)是常见的数字调制信号, 其中 2ASK 和 2FSK 属于线性调制。

1. 二进制振幅键控(2ASK)

2ASK 是利用代表数字信息“0”或“1”的基带矩形脉冲去键控一个连续的载波, 使载波时断时续地输出。有载波输出时表示发送“1”, 无载波输出时表示发送“0”。通常 2ASK 的调制原理示意图如图 6.3 所示, 其中载波信号为频率较高的正弦波, 开关电路受二进制数据信号 $S(t)$ 控制, 通过包络检波法解调。2ASK 波形示意图如图 6.4 所示。

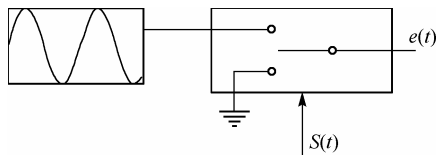


图 6.3 2ASK 的调制原理示意图

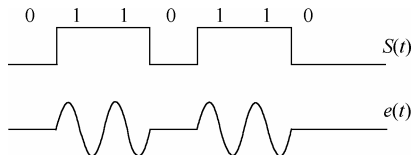


图 6.4 2ASK 波形示意图

一个 2ASK 信号可以表示成一个单极性矩形脉冲序列与一个正弦载波的相乘, 即

$$e(t) = \left[\sum a_n g(t - nT_s) \right] \cos \omega_c t \quad (6.1)$$

式中, $g(t)$ 是持续时间为 T_s 的矩形脉冲, 而 a_n 的取值服从下述关系:

$$\begin{cases} a_n = 0 & \text{概率为 } p \\ a_n = 1 & \text{概率为 } (1-p) \end{cases}$$

令 $S(t) = \sum a_n g(t - nT_s)$, 则得到其数学模型为 $e(t) = S(t) \cos \omega_c t$ 。

该信号有两种基本的解调方法: 非相干解调法(包络检波法)及相干解调法(同步检测法), 二进制振幅键控信号的接收系统组成方框图如图 6.5 所示。

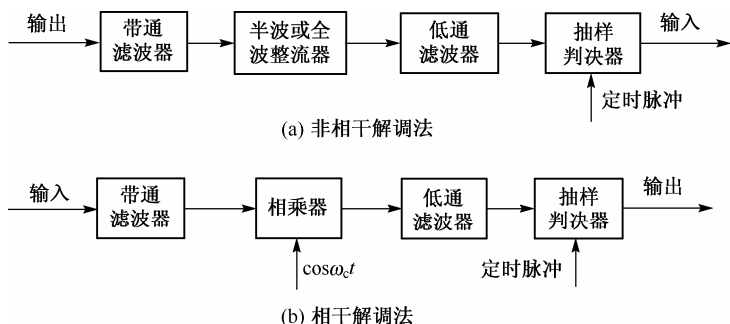


图 6.5 2ASK 信号的接收系统组成方框图

2ASK 是数字调制中出现最早的,也是最简单的方法,调制和解调电路都很简单,所占据的带宽是基带脉冲波形带宽的两倍;它的主要缺点是:系统抗噪声性能差,误比特率高。

2. 二进制频移键控(2FSK)

2FSK 是用载波的频率来传送数字消息的,即用所传送的数字消息控制载波的频率。通常 2FSK 信号的产生原理示意图如图6.6所示。2FSK 信号中的“0”符号对应载频 f_1 ,“1”符号对应载频 f_2 , 2FSK 利用矩形脉冲序列控制开关电路对两个不同的独立频率源进行选通,其波形示意图如图 6.7 所示。

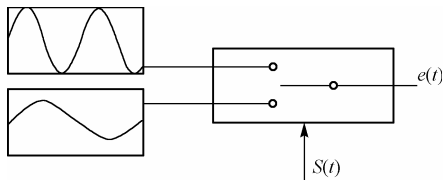


图 6.6 2FSK 信号的产生原理示意图

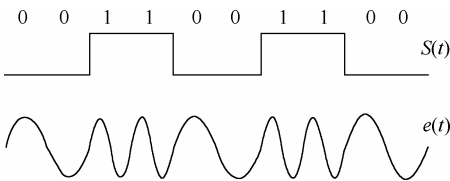


图 6.7 2FSK 波形示意图

根据以上 2FSK 信号的产生原理,已调信号的数学表达式可以写为如下形式:

$$e(t)=\sum a_n g(t-nT_s)\cos(\omega_1 t+\varphi_n)+\sum a_n' g(t-nT_s)\cos(\omega_2 t+\phi_n)$$

(6.2)

式中, $g(t)$ 为单个矩形脉冲; 脉宽为 T_s :

$$\begin{cases} a_n=0 & \text{概率为 } p \\ a_n=1 & \text{概率为 } (1-p) \end{cases}$$

a_n' 是 a_n 的反码,即若 $a_n=0$, 则 $a_n'=1$; 若 $a_n=1$, 则 $a_n'=0$ 。于是 $a_n'=0$, 概率为 $(1-P)$, $a_n'=1$, 概率为 P 。由 2FSK 的基本原理得知, 一个 2FSK 信号可以看成是两个不同载频的 2ASK 信号的叠加, φ_n 和 ϕ_n 分别是第 n 个信号码元(1 或 0)的初始相位。在频移键控中, φ_n 和 ϕ_n 不携带信息,通常可令 φ_n 和 ϕ_n 为零。

2FSK 信号的常用解调方法是如图 6.8 所示的非相干检测法和相干检测法。

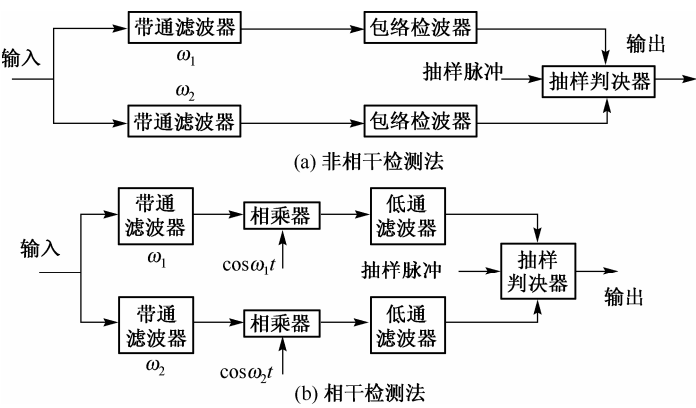


图 6.8 2FSK 信号的常用解调方法

2FSK 的优点是抗干扰能力强,能量主要集中在载频的较低处,是数字通信中使用比较广的一种方式。2FSK 的不足之处是其要求的带宽为 $2B+|f_2-f_1|$ (f_1 和 f_2 为 FSK 的两个载波频率), 频带利用率低,相等信噪比下误码率较高。

3. 二进制相移键控(2PSK)

2PSK 是受键控的载波相位按基带脉冲而改变的一种数字调制方式,通常 2PSK 信号的产生原理示意图如图 6.9 所示。其中,载波信号的相位随基带脉冲的改变而改变。二进制相移键控中,通常用相位 0 和 π 来分别表示“0”或“1”,其波形示意图如图 6.10 所示。

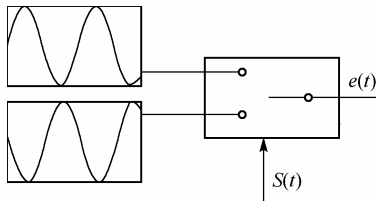


图 6.9 2PSK 信号的产生原理示意图

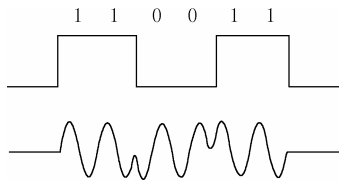


图 6.10 2PSK 波形示意图

2PSK 信号的数学模型如下:

$$e(t) = \sum a_n g(t - nT_s) \cos \omega_c t \quad (6.3)$$

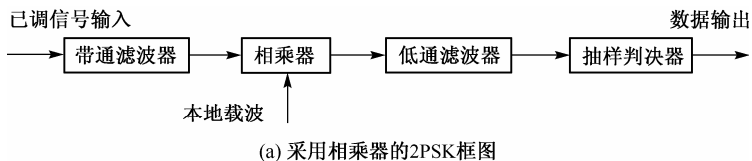
式中, $g(t)$ 是持续时间为 T_s 的矩形脉冲; a_n 的统计特性为

$$a_n = +1, \text{ 概率为 } p; \quad a_n = -1, \text{ 概率为 } (1-p)$$

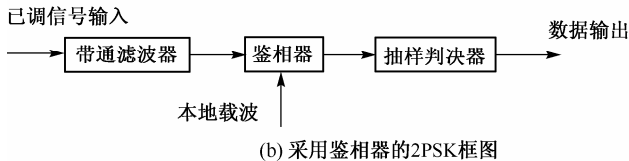
因此,在其一个码元持续时间 T_s 内观察时, $e(t)$ 为

$$e(t) = \cos \omega_c t, \text{ 概率为 } p; \quad e(t) = -\cos \omega_c t, \text{ 概率为 } (1-p)$$

对于 2PSK 的解调一般采用相干解调,相应框图如图 6.11 (a) 所示。考虑到相干解调在这里实际上起鉴相作用,故相干解调中的“相乘-低通”可用鉴相器代替,如图 6.11 (b) 所示。图中的解调过程实质上是输入已调信号与本地载波信号进行极性比较的过程,该方法又称为极性比较法解调。



(a) 采用相乘器的 2PSK 框图



(b) 采用鉴相器的 2PSK 框图

图 6.11 2PSK 信号的接收方框图

2PSK 的优点是:频带利用率高,相同信噪比下误码率最低;它的不足之处是:PSK 存在相位模糊问题,需要对源二进制信号进行差分编码,然后进行调相,即利用 DPSK 方法解决相位模糊问题。因此,在调制端和解调端都要增加很复杂的硬件电路,而且差分译码的时候可能引起误码扩散,导致系统误码率上升。除此之外,还有差分相干 2DPSK 和同步检测 2DPSK,虽然性能好,但是由于系统过于复杂,不便于实际实现。

4. 方案选择

综合上述三种方案,从系统误码率上看相干方式略优于非相干方式。从频带利用率来看,ASK 和 PSK 的频带宽度都是 $2B$ (B 为被调制二进制基带信号的带宽),即码元宽度为 T_s 时,2ASK 系统和 2PSK

系统的第一零点带宽为 $2/T_s$ ，即 $B_{2ASK} = B_{2PSK} = 2f_s$ 。其中， $f_s = 1/T_s$ 。由此可见，2ASK 系统和 2PSK 系统的信号传输带宽是码元速率的 2 倍(基带信号带宽的 2 倍)，FSK 则相对大一些。具体分析如下：

(1) 由前面的讨论已知，对相位不连续的 2FSK 信号，可以看成由两个不同载频的 2ASK 信号的叠加，因此，2FSK 频谱可以近似表示成中心频率分别为 f_1 和 f_2 的两个 2ASK 频谱的组合。采用 2FSK 方式时，2FSK 系统的第一零点带宽为 $|f_2 - f_1| + 2/T_s = |f_2 - f_1| + 2B$ ，其中 $|f_2 - f_1|$ 为两个载频之差。其传输带宽 $B_{2FSK} = |f_2 - f_1| + 2f_s > 32 \text{ kHz}$ ，超出了设计任务给出的模拟信道的带宽(20 kHz)。考虑到题目中模拟信道带宽较窄，而要求达到的码元传输速率较高，这将导致 FSK 信号的部分频谱分量不能通过信道传输，从而造成误码率的升高。

(2) 2ASK 系统所需硬件设备最简单，但由于它的误码率较高，一般不用于数字通信中。考虑设计任务给出的信道及噪声比要求，采用 2ASK 方式时一个码元内只包括两个周期的载波，所以采用包络检波法难以解调，实现的可能性很小。

(3) 对于 2PSK 系统，从频带利用率和抗干扰能力上(特别是抗加性高斯白噪声)看都比较理想，但由于 2PSK 系统存在倒相问题，使用中需要对调制信号进行差分编码，这会造成系统解调设备复杂的同时降低系统的抗干扰性能。

(4) 带通二进制键控系统中，每个码元只传输 1 bit 信息，其频带利用率不高，为了提高频带利用率，最有效的办法是使一个码元传输多个比特的信息。这就是多进制键控体制，例如在四进制频移键控(4FSK)中采用 4 个不同的频率分别表示四进制的码元，每个码元含有 2 bit 的信息。需要说明的是，采用多进制键控体制使一个码元传输多个比特的信息，用于提高频带利用率，这一结论并无错误，但就具体调制方式进行分析时可看出，由于 MFSK 的码元采用 M 个不同频率的载波，要求每个载频之间的距离足够大，使不同频率的码元相互正交(和使用 2FSK 时的条件相同)，所以它占用较宽的频带。设 f_l 为其最低载频， f_h 为其最高载频，则 MFSK 信号的带宽近似等于 $B_{MFSK} = f_h - f_l + \Delta f$ 。式中 Δf 为单个码元的带宽，它决定于信号传输速率。同时，为了得到相同的误码率，和二进制系统相比，接收信号信噪比需要更大，即需要更大的发送信号功率。

(5) 鉴于 FSK 调制的不足之处，有一种改进的调制方式，即最小频移键控方式(MSK)。MSK 方式是 FSK 信号的一种特殊方式，即当 $(f_2 - f_1) = n(1/2)T_b$ (T_b 为比特率)时，其相位始终保持连续变化且两载波的频率之差始终等于码元速率的 1/2，对于给定的频带，MSK 相比 FSK 能传输更高的比特率。该调制方式可使信号的波形没有突跳，使信号主瓣所占频带宽度比 2FSK 信号窄，在主瓣带宽之外，功率谱旁瓣的下降也更为迅速。因此，MSK 信号比较适合在窄带信道中传输，对邻道的干扰也较小，抗干扰性能也要优于 2FSK。针对设计任务，使用 MSK 调制应属于较好的方案，但 MSK 调制需要用到专门的调制芯片，实现成本较高。设计时可选择采用 EXAR 公司的单片函数发生芯片 XR2206 为核心构成调制电路，其实现难度不大。

(6) 如果仅考虑系统设计的可实现性，在降低设计指标的情况下，可使用 2FSK 调制的方案。2FSK 调制占用带宽为 $|f_2 - f_1| + 2/T_s = |f_2 - f_1| + 2B$ ，兼顾信道的带宽及为避免两个载频间的相互干扰，可选定两个载波频率分别为 32 kHz 和 48 kHz。

6.2.3 噪声模拟发生器的设计和模型分析

通常产生伪随机序列的电路为一反馈移位寄存器。一般地，线性反馈移位寄存器由于理论比较成熟，实现比较简单，实际中常常使用。由线性反馈移位寄存器产生出的最长的二进制数字序列称为最大长度线性反馈移位寄存器序列，通常简称为 m 序列。

一般的线性反馈移位寄存器的组成框图如图 6.12 所示，其中若 $C_i=0$ 表示此线断开， $C_i=1$ 则表示

接通。

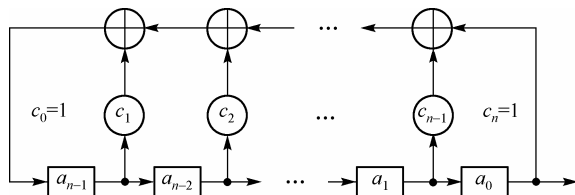


图 6.12 线性反馈移位寄存器的组成框图

由于 n 级移位寄存器共有 2^n 个的不同状态，除全 0 外，还剩下 2^n-1 个不同的状态。因此， n 级线性反馈移位寄存器产生的最长序列的周期为 2^n-1 。

设 n 级移位寄存器的初始状态为 $a_{-1}a_{-2}L a_{-n}$ ，经过一次移位后，状态变为 $a_0a_{-1}a_{-2}L a_{-n+1}$ 。经过 n 次移位后，状态为 $a_{n-1}a_{n-2}L a_1a_0$ 。再位移一次时，移位寄存器左端新得到的输入 a_n 为

$$a_n = c_1a_{n-1} \oplus c_2a_{n-2} \oplus L \oplus c_na_0 = \sum_{i=1}^n c_ia_{n-i} \pmod{2} \quad (6.4)$$

故对任意状态 a_k ，有

$$a_k = \sum_{i=1}^n c_ia_{k-i} \quad (6.5)$$

令

$$f(x) = c_0 + c_1x + c_2x^2 + L + c_nx^n = \sum_{i=0}^n c_ix^i \quad (6.6)$$

该方程即为特征方程(或特征多项式)。由 n 级移位寄存器构成的 m 序列发生器，其线性序列的最大长度为 $m=2^n-1$ ，题目要求的 m 码周期为 $127=2^7-1$ 位码元，故需要 7 级移位寄存器。因此选择 m 序列的生成多项式为 $f(x)=1+x^3+x^7$ ，其原理图如图 6.13 所示。

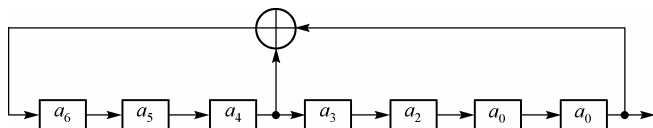


图 6.13 m 序列产生电路原理图

当各级触发器均处于 0 状态时，电路会停止产生序列信号，即产生了阻塞现象。这是由于序列发生器在 0 状态下不具有自启动特性，所以需要在反馈中增加一个防止全 0 的修正项，具体实现电路如图 6.14 所示。在输出端口加入一个衰减器，使输出电压可以在 $0\sim 1\text{ V}$ 范围内连续变化。

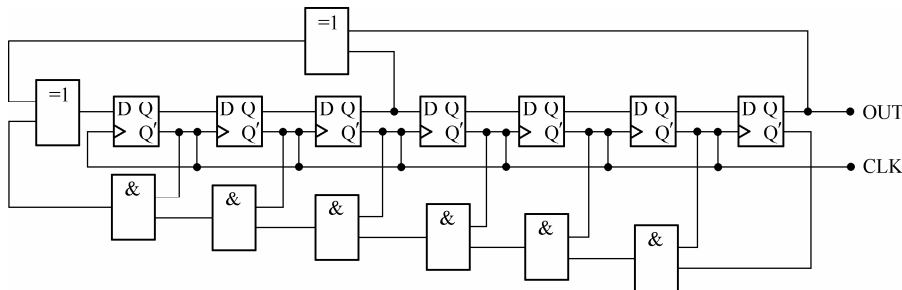


图 6.14 m 序列产生电路

6.2.4 系统各模块设计分析

根据以上分析与论证，在一片单片机上同时实现数据的收发比较困难，因此采用双 CPU 方案用于发送端和接收端的发送和接收控制，用两片可以精确设定波特率的 89C51 单片机来分别控制数据采集、调制、通信、解调和采集结果显示等功能的实现，采用 FSK 调制，8 路模拟信号采集与单向传输系统总体设计框图如图6.15 所示。

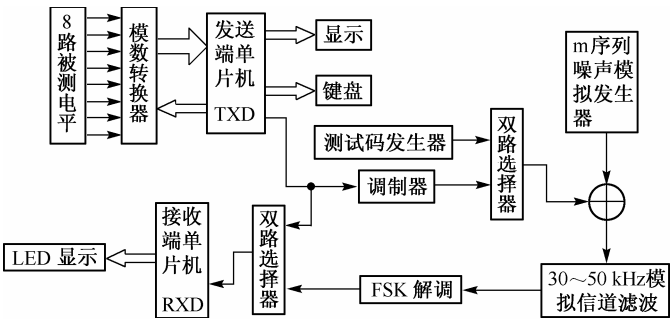


图 6.15 8 路模拟信号采集与单向传输系统总体设计框图

1. 8 路模拟信号的产生和A/D转换器电路设计

被测信号为 8 路 0~5 V 分别可调的直流电压，可通过可调电位器分压产生。A/D 转换采用逐次逼近 A/D 转换芯片 ADC0809，分辨率为 8 位，量化误差不超过 $\pm 1/2$ LSB，片内有带锁存功能的 8 路模拟多路开关，可对 8 路 0~5 V 的输入模拟电压信号进行转换，转换时间约 100 μ s。为减小测量误差，对信号源的参考电源进行了稳压处理，可采用直流基准稳压芯片 MAX883，参考实现电路如图 6.16 所示。

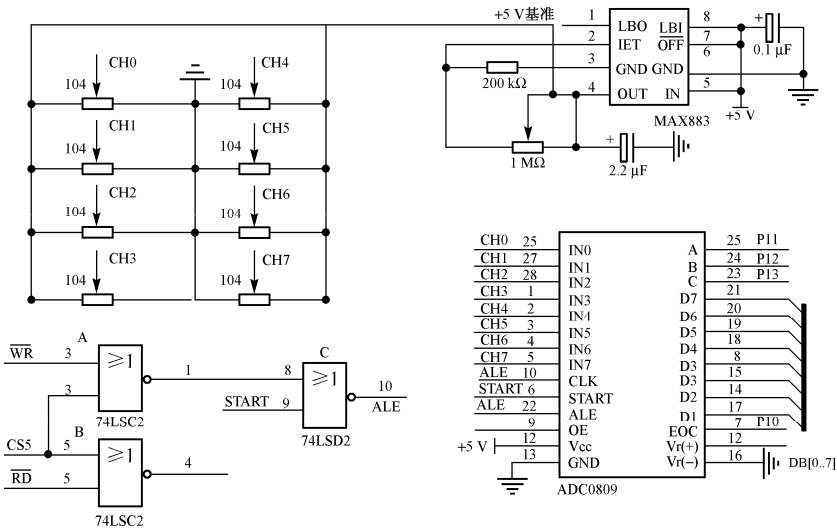


图 6.16 8 路模拟信号的产生和 A/D 转换电路

2. 发送端的采集与通信控制器

发送端的采集与通信控制器采用 89C51 单片机作为控制核心，接收来自 ADC0809 的数据，并利用单片机内置的专用串行通信电路将数据进行并-串转换后输出至调制器。发送端可采用键盘作为输入

控制,单片机通过接口芯片与键盘相连,由键盘控制采集方式是轮询采集或指定通道采集,同时也可以利用键盘进行其他扩展功能的切换。

3. 调制解调电路的设计

经过前面的方案论证,选择FSK调制方式,选择两个载波频率为32 kHz和48 kHz。调制电路可以由单片函数发生芯片为核心来构成,也可以利用CPLD来设计基带信号调制单元。

以后者为例,CPLD中基带信号调制单元是一个双路选择器,载波信号通过分频电路产生,基带信号接在双路选择器的选择端子上,如图6.17所示。当基带信号为电平“1”时CH2输出,当基带信号为电平“0”时CH1输出,从而实现2FSK的调制。

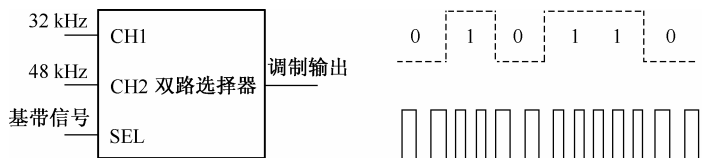


图 6.17 基带信号调制单元

解调可以利用锁相环电路实现,也可以采用专用的FSK解调器芯片实现,现以锁相环构成解调电路的方式为例。锁相环相当于一个中心频率可以跟踪输入信号频率变化的窄带滤波器。利用锁相环的跟踪功能,实现载波信号的同步提取。锁相环基本组成框图如图6.18所示。

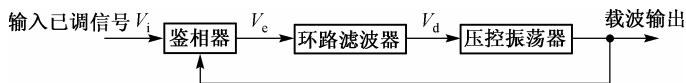


图 6.18 锁相环基本组成框图

如图6.18所示,设输入信号 V_i 的相位为零,鉴相器输出 $V_e(t)$ 也为零,则经过滤波器的输出信号 $V_d(t)=0$,压控振荡器(VCO)处于自由振荡状态,输出一个振荡频率为 f_v 、相位为 θ_v 的信号。当输入信号 V_i 的相位为 θ 时, θ 与 θ_v 通过相位比较器进行比较,输出一个与相位差 $\Delta\theta=\theta-\theta_v$ 的大小成比例的误差电压 $V_e(t)$,经过环路滤波器输出一个变化缓慢的控制电压 $V_d(t)$ 。此电压控制压控振荡器的输出相位,使 θ_v 向 θ 靠近,直到 $\Delta\theta$ 趋近于0或 $\Delta\theta$ 为常数时,环路才稳定下来,即进入锁定状态。环路锁定时,压控振荡器的输出频率 f_v 与输入信号的频率 f_i 相等。因此,如果 f_i 是一个高稳定的基准信号,则 f_v 也具有 f_i 的稳定度,从而提取高质量的载波。

利用74HC4046锁相环构成FSK解调电路,参考电路如图6.19所示。2FSK调制系统的两个载波频率分别为 $f_1=32\text{ kHz}$ 和 $f_2=48\text{ kHz}$,中心频率 $f_0=40\text{ kHz}$ 。在锁相环的前级设计比较器,将通过模拟信道的传输信号转换为TTL电平;在设计锁相环时,使其锁定在FSK的一个载频 f_2 上,对应输出高电平,而对另一载频 f_1 失锁,对应输出低电平,然后在锁相环的输出端连接一个截止频率为20 kHz的低通滤波器,滤除锁相环解调后输出信号的高频成分,最后用比较器对信号进行整形,即可输出波形较好的数字信号。

4. 3 dB带宽 30~50 kHz模拟信道滤波器模块

为在通带内获得最大平坦度,选择巴特沃斯带通滤波器。巴特沃斯低通滤波器不管阶数 N 为多少,所有特性曲线都通过-3 dB点,因此具有3 dB不变性,而且在通带内有最大平坦的幅度特性。巴特沃斯低通滤波器可通过变换关系式 $s = p + \Omega_0^2/p$ 变换成巴特沃斯带通滤波器,其中 s 是模拟低通滤波器

拉普拉斯变量($s = \sigma_1 + j\Omega_1$)， p 是模拟带通拉普拉斯变量($p = \sigma + j\Omega$)， Ω_0 是模拟带通的几何中心频率。设计指标为 $f_{c1} = 30\text{ kHz}$ ， $f_{c2} = 50\text{ kHz}$ ，阻带衰减 $\geq 60\text{ dB}$ /十倍频程。

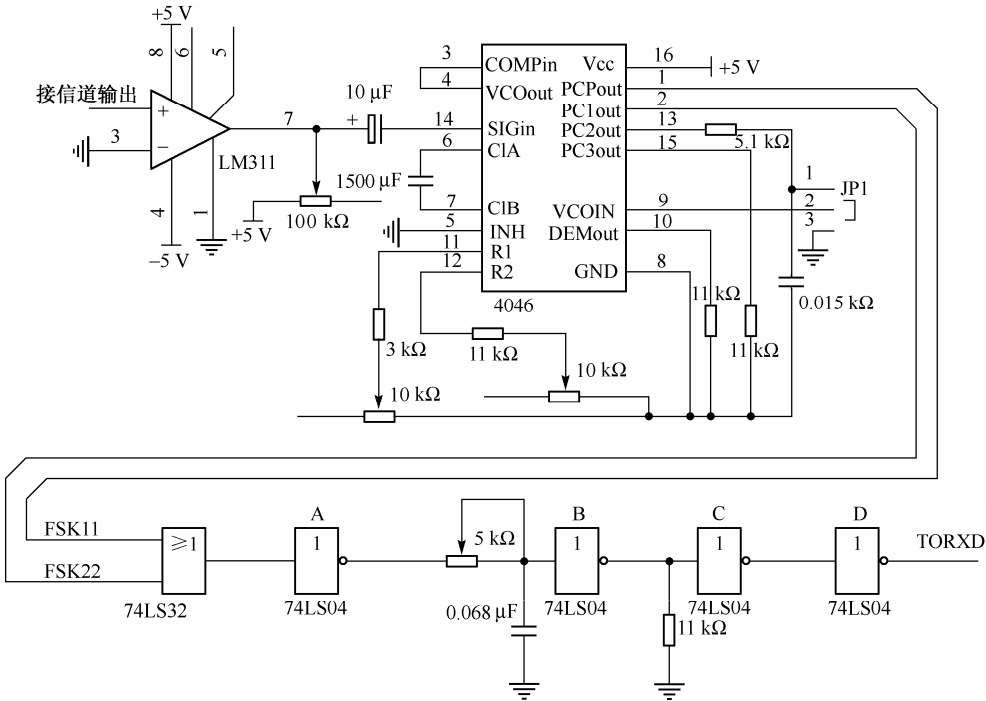


图 6.19 利用 74HC4046 锁相环构成 FSK 解调电路

1) 阶数计算

根据巴特沃斯型低通幅度函数，可得

$$20\lg(1+10^{2N})^{1/2} \geq 60\text{ dB}$$

解得 $N \geq 3$ 。这里取 $N = 4$ ，即滤波器选择 4 阶。

2) 电路选择

电路可选择两级二阶 MFB 带通滤波器级联形式，根据系统传输函数和巴特沃斯 4 阶多项式的表达式计算出组容元件值，参考电路图如图 6.20 所示。

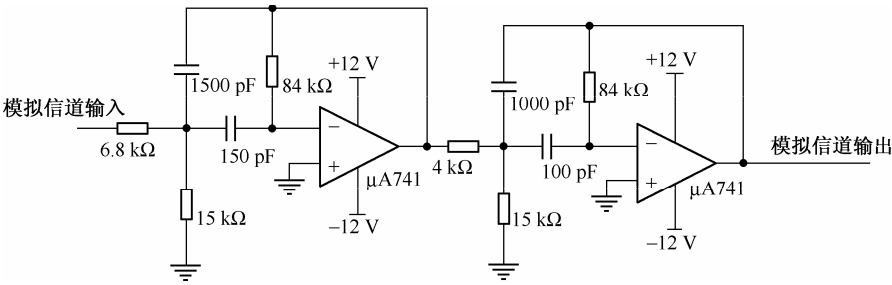


图 6.20 30~50 kHz 模拟信道滤波器

5. m序列噪声码发生器单元

根据前面的论证分析，该单元可利用 7 级反馈移位寄存器构造的伪随机序列产生电路作为模拟噪声发生器。需注意的是，为了产生连续序列波形，防止全零状态出现，将 7 级 D 触发器的 Q 端通过或

非门接到 D 触发器的置位端, 一旦出现全零的异常情况, D 触发器被置位, 电路恢复正常工作。在噪声输出端用电位器调节其峰-峰值在 0~1 V 变化, 通过一级射随器隔离后通过加法器实现与调制器输出信号的相加。

模拟噪声发生器可采用普通数字逻辑分立器件实现, 但所需分立器件多, 结构复杂。本方案使用 CPLD 来实现, 采用 Altera 公司的 EPM7128 芯片, 伪随机序列的生成多项式为 $f(x) = 1 + x^3 + x^7$, 此方案简单易行, 只需把编译好的文件下载到可编程器件中即可, 而且可以仿真, 在编译好文件后就可以看出所做的设计是否正确。

6. 时钟频率可变的测试码发生器设计

测试码主要用于测试传输速率, 对于码型没有特别的要求, 可以采用频率可调的方波信号(0101...码)。实现电路可采用波形发生器 ICL8038 及简单的外围电路构成, 这样电路结构简单, 输出频率连续变化, 但其不足之处是电路频率稳定性差, 对误码测试有影响。还可利用 CPLD 的分频模块产生不同频率的测试码。此方案的优点是没有多余的外围电路, 频率稳定性好; 不足之处是, 输出频率不能连续变化, 只能对所需测试的几个测试点进行测试。

考虑到系统设计的便利及资源利用率, 可采用 CPLD 实现基带信号的 2FSK 调制、m 序列噪声码发生器、测试码发生器和分频器等功能。CPLD 的内部逻辑模块如图 6.21 所示。

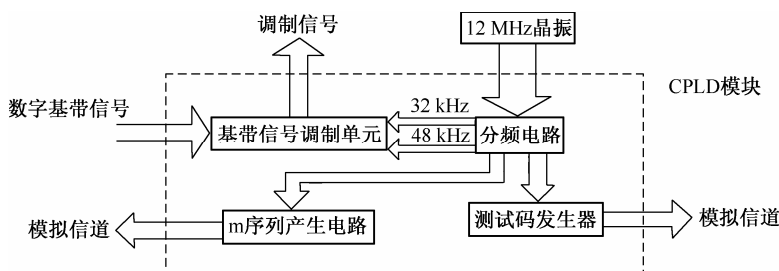


图 6.21 CPLD 的内部逻辑模块

6.2.5 系统软件设计

1. 系统软件功能

系统软件分为发送端软件模块和接收端软件模块两部分, 其主要功能包括:

- (1) 发送端可设定 8 路循环采集或者指定一路采集, 采集的同时显示当前通道号和相应电压值。
- (2) 软件过滤错误数据, 并支持一定的纠错功能。
- (3) 软件实现误码率测试: 系统附加测试信道, 使系统本身支持误码率测试与显示。
- (4) 软件实时设定波特率, 多挡可调。
- (5) 通过键盘设定噪声是否加入模拟信道。
- (6) 与 CPLD 的接口控制等。

2. 系统软件流程图

系统软件主要包括发送端软件模块和接收端软件模块, 其流程图分别如图 6.22 和图 6.23 所示。

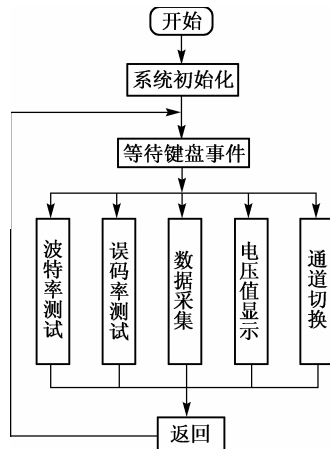


图 6.22 发送端软件模块流程图

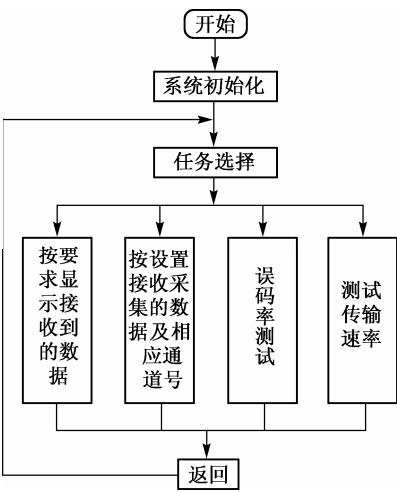


图 6.23 接收端软件模块流程图

6.3 本章小结

本章以一个 8 路模拟信号采集与单向传输系统为例，介绍了数据采集与传输系统的设计，通过数据采集模块、调制解调电路、模拟信道设计、测试码发生器电路、噪声模拟发生器电路等部分的分析与设计，使读者对数据采集与传输系统的基本组成和设计方法有一个较为全面的理解和掌握。调制解调技术是数据传输系统的关键技术之一，不同调制方式对系统设计的整体实现有很大的影响，而调制方式又是由信道的传播特性所决定的。一个好的调制方式应该在低接收信噪比的条件下提供尽可能小的误比特率。本章仅以 2FSK 方式为例来设计了调制解调电路，强调的是系统设计的可实现性。在实际的数据传输系统设计中，应根据实际的要求来选择合适的调制解调方法并设计相应的电路。

第 7 章 控制策略与算法的研究

7.1 引言

过程控制是工业自动化的重要分支。学习和掌握一些相对简单的过程控制理论和设计方法是十分必要的。

工业中的过程控制是指以温度、压力、流量、液位和成分等工艺参数作为被控变量的自动控制。一般而言,管理中采取的控制可以在行动开始之前、进行之中或结束之后进行,第一种称为前馈控制或预先控制,第二种称为过程控制或同期控制,第三种称为反馈控制或事后控制。本章主要讨论过程控制,过程控制系统一般由以下几部分组成:

- (1) 被控过程(或对象);
- (2) 用于生产过程参数检测的数据采集器与变送仪表;
- (3) 控制器;
- (4) 执行机构;
- (5) 报警、保护和连锁等其他部件。

过程控制有非常鲜明的特点,其主要特点如下:(1) 被控过程的多样性。其主要表现为过程控制所涉及的面非常广泛,控制对象不同控制过程的机理不同,执行机构也不同。因此,过程控制系统中的被控对象(包括被控量)是多样的,明显地区别于运动控制系统。(2) 控制方案的多样性。由被控过程的多样性决定了过程控制系统的控制方案必然是多样的。这种多样性包含系统硬件组成和控制算法及软件设计。(3) 被控过程属于慢过程且多属于参数控制。连续工业过程大惯性和大滞后的特点决定了被控过程为慢过程。被控过程是物流变化的过程,伴随物流变化的信息(物性、成分、温度、压力、流量、液位或物位)表征为被控过程状态的参数,也是过程控制系统的被控量。(4) 定值控制是过程控制的主要形式。在过程控制中,被控参数的设定值为一个定值,定值控制的主要任务是如何减小或消除外界干扰,使被控量尽量保持接近或等于设定值。

自从 1932 年奈奎斯特(Nyquist)发表有关反馈放大器的稳定性的论文以来,控制理论科学的发展已走过 70 多年的历程,其中前 30 年是经典控制理论的发展和成熟阶段,后 40 年是现代控制理论的形成和发展阶段。几十年来,过程控制策略与算法出现了三种类型:简单控制、复杂控制与先进控制。通常将单回路 PID 控制称为简单控制。PID 控制以经典控制理论为基础,主要用频域方法对控制系统进行分析设计与综合。为了满足复杂过程工业的一些特殊控制要求,过程控制的理论和技术不断进步,逐渐发展了串级控制、前馈控制、均匀控制和 Smith 预估控制等控制策略与算法,称为复杂控制。这些方法仍然以经典控制理论为基础,但在结构与应用上各有特色。从 20 世纪 80 年代开始,在现代控制理论和人工智能发展的理论基础上,针对工业过程本身的非线性、时变性、耦合性和不确定性,提出了许多行之有效的解决方法,如预测控制、模糊控制、自适应控制、人工神经网络控制等,常统称为先进过程控制。近年来,以专家系统、模糊逻辑、神经网络、遗传算法为主要方法的基于知识的智能处理方法已经成为过程控制的一个重要方向。

考虑到典型性和可获得设计模型的便利性,本章首先以一个特定对象(150 mm×80 mm×20 mm 的铁块)的温度控制为例讨论过程控制中的有关问题,然后针对被控对象可能存在的非线性、纯时滞的特

点, 讨论其控制策略及控制算法。由于过程控制具有很强的实践性, 控制策略和算法的有效性必须通过验证。因此, 控制系统的硬件电路的设计必须相对于算法是开放的, 硬件电路的设计方法也是本章讨论的重点。本章将通过以上讨论后给出一个恒温控制系统的设计实例。

7.2 基于非线性、纯时滞被控对象的控制策略及算法的分析

在自然和社会现象中, 许多系统的发展趋势或未来状态不仅与现状有关, 而且或多或少地与过去的发展历史有关, 这类现象称为滞后现象, 或称为遗传效应。在过程控制中, 常常用单容水槽和热交换器为例来说明时滞现象的产生。在单容水槽的液位控制中, 由于调节阀距水槽有一段距离, 调节阀开度的变化所引起的流入量变化 ΔQ_i 需经一段传输时间 τ , 才能对水槽液位产生影响, τ 即是纯延迟时间。在热交换器中, 被调量是加热物料的出口温度, 而控制量是载热介质, 当改变载热介质流量后, 对物料出口温度的影响必然要迟延一个时间, 即介质经管道传输所需的时间。这两个例子具有典型性, 其实, 几乎所有的被控对象都有时滞特性。而时滞被普遍认为是导致系统不稳定的一个原因, 时滞的产生会极大地破坏系统的控制性能, 比如, 在过程控制中, 由于纯时滞的存在, 使得被调量不能及时反映系统所承受的扰动, 即使测量信号到达调节器, 调节机构接收调节信号后立即动作, 也需要经过纯延迟时间 τ 以后, 才能波及被调量, 使之受到控制。因此, 这样的过程必然会产生明显的超调量和较长的调节时间, 系统的不确定度增大。所以, 具有纯时滞的过程被公认为是较难控制的过程, 其难控程度将随着纯时滞时间 τ 占整个过程动态的份额的增加而增加。一般认为, 当纯时滞时间 τ 与过程的时间常数 T 之比大于 0.3 时, 称该过程是具有大时滞的过程。从以上两个例子还可看出, 由于对象的变增益特性, 对象动态特性的描述只能用非线性方程。在实际的生产过程中, 精确的分析结果表明所有系统都是非线性的, 而线性系统只是一种近似和简化, 因此研究非线性系统更能发掘自然界的本质特征。随着生产和科学的发展, 非线性问题越来越多地成为人们关注的焦点问题之一。综上所述, 非线性系统普遍存在, 时滞是系统一般所固有的, 所以综合二者, 分析和研究非线性、时滞系统的控制方法及稳定性就显得相当重要。

为了解决非线性、纯时滞对象的控制问题, 许多学者在理论和实践上做了大量的研究工作, 相继提出了许多行之有效的控制方法, 主要有 Smith 预估计控制策略、大林(Dahlin)算法、自适应控制、预测控制、鲁棒控制、变结构控制、智能控制及各种复合控制策略。本节将分别对它们进行简要的介绍和分析。

7.2.1 Smith预估计控制和大林算法

Smith 于 1957 年提出了著名的预估控制算法(纯滞后补偿算法), Smith 控制方法是在系统给定通道的前向回路中设置一个并联补偿装置, 使经补偿后的等效对象不再含有纯滞后, 从而对这个滞后的等效对象施加 PI 或者 PID 调节。具有纯滞后对象的单反馈控制系统如图 7.1 所示。其中, $R(s)$ 为设定值; $E(s)$ 为偏差, $E(s) = R(s) - W(s)$; $D(s)$ 为调节器的传输函数; $Y(s)$ 为调节器输出; $G(s) = G_0(s)e^{-\theta s}$ 为被控对象的传递函数; $G_0(s)$ 为被控对象中不含纯时滞部分的传递函数; $e^{-\theta s}$ 为被控对象纯滞后部分的传递函数; $W(s)$ 为被调量, 又称为输出量。

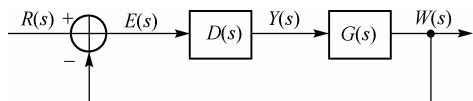


图 7.1 具有纯滞后对象的单反馈控制系统

为了提高这类大纯滞后系统的稳定性, 可在调节器 $D(s)$ 的两端并联一个反馈补偿网络, 用来补偿被控对象中的纯滞后部分, 这就是 Smith 预估器的原理。

这个反馈补偿网络称为 **Smith 预估器**，其传递函数为 $\tilde{G}_0(s)(1-e^{-\theta s})$ 。补偿后的系统框图如图 7.2 所示。

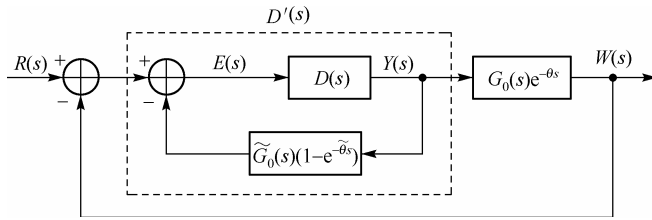


图 7.2 采用 Smith 补偿的大纯滞后补偿控制系统

图 7.2 中虚线框为带纯滞后补偿的调节器，其传递函数为

$$D'(s) = \frac{Y(s)}{R(s) - W(s)} = \frac{D(s)}{1 + D(s)\tilde{G}_0(s)(1 - e^{-\tilde{\theta}s})} \quad (7.1)$$

图 7.2 可简化为图 7.3 所示的单回路形式。

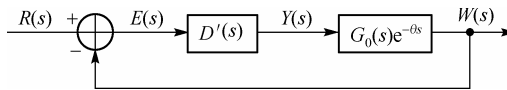


图 7.3 大纯滞后补偿控制系统的等效补偿

由图 7.3 可得补偿后的系统闭环传递函数为

$$\phi(s) = \frac{D(s)G_0(s)e^{-\theta s}}{1 + D(s)\tilde{G}_0(s)(1 - e^{-\tilde{\theta}s}) + D(s)G_0(s)e^{-\theta s}} \quad (7.2)$$

在满足条件 $\tilde{G}_0(s) = G_0(s)$ ， $\tilde{\theta} = \theta$ 时，系统达到完全补偿，此时系统的闭环传递函数为

$$\phi(s) = \frac{D(s)G_0(s)e^{-\theta s}}{1 + D(s)G_0(s)} \quad (7.3)$$

式 (7.3) 表明，在完全补偿的条件下，消除了被控对象的纯滞后部分对系统稳定性的不利影响，因为式中 $e^{-\theta s}$ 被等效地移至闭环控制回路之外，不影响系统的稳定性。由拉普拉斯变换的平移定理可以证明： $e^{-\theta s}$ 仅将控制过程在时间坐标上推移了纯滞后时间 θ ，控制系统的过渡过程及其他性能指标均与对象特性 $G_0(s)$ 不含纯滞后部分时完全相同。因此，对任何大纯滞后时间 θ ，系统都是稳定的。

然而，尽管 **Smith** 算法能够从理论上解决纯滞后系统的控制问题，但是它存在一些缺陷，如要求精确的过程模型、对实际对象的参数变化十分敏感、系统应对扰动的响应很差等。

大林算法是由美国 **IBM** 公司的 **Dahlin** (大林) 于 1968 年针对工业过程控制中的纯滞后特性而提出的一种控制算法。

大林算法的优点是过程比较简单，只要能设计出合适的且可以物理实现的数字调节器 $D(z)$ ，就能够有效克服纯滞后的不利影响。但它的缺点和 **Smith** 算法相同，需要一个准确的过程数学模型。当模型误差较大时，控制质量将严重恶化，甚至系统会变得不稳定。已有很多的学者通过研究 **Smith** 预估器、大林控制器及内模控制 (**IMC**) 的关系后，得到三者在一定程度上具有等价关系。在模型精确时，内模控制和大林控制器是完全等价的，而 **Smith** 预估器与内模控制器在结构上有相似性，因此可以在此基础上应用内模控制设计的方法来设计 **Smith** 预估器，以提高其鲁棒性能。内模控制结构示意图如图 7.4 所示。

7.2.2 自适应控制

20 世纪 70 年代初期, 由于空间技术和过程控制的发展, 自适应控制技术得到快速发展。自适应控制建立在系统数学模型参数未知的基础上, 随着系统行为的变化, 控制器会相应地改变其控制参数, 以适应系统特性的变化, 保证整个系统的性能指标达到令人满意的结果。自适应控制在时滞系统中的应用主要分为模型参考自适应控制和自校正调节器两大类。

通常模型参考自适应设计方法为: 基于局部参数最优化的设计方法和基于稳定性考虑的设计方法。参数最优化方法没有考虑系统的稳定性和鲁棒性。基于稳定性考虑的设计方法又分为根据李雅普诺夫稳定性衍生的算法和根据波波夫超稳定性理论的设计方法。这些设计方法都需要对实际过程有较准确的数学描述。也正因这一点, 该方法并未在实际过程控制中得到广泛的应用。但应该强调的一点是, 对于一般一阶或二阶系统, 上述设计都很成熟且有控制效果很理想的算法。可以很容易地比较这些算法, 选取较理想的一种作为这类问题的基础算法。一般模型参考自适应控制结构示意图如图 7.5 所示。

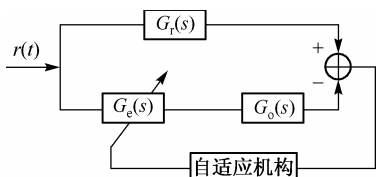


图 7.5 一般模型参考自适应控制结构示意图

7.2.3 预测控制

预测控制是一种基于预测过程模型的控制算法, 根据过程的历史信息判断将来的输入和输出。它强调模型的函数而非模型的结构, 因此, 状态方程、传递函数甚至阶跃响应或脉冲响应都可作为预测模型。预测模型能体现系统将来的行为, 设计者可以尝试不同的控制律用计算机仿真观察系统的输出结果。

预测控制是一种最优控制的算法, 根据补偿函数或性能函数计算出将来的控制动作。预测控制的优化过程不是一次离线完成的, 是在有限的移动时间间隔内反复在线进行的。移动的时间间隔称为有限时域, 这是与传统的最优控制最大的区别。传统的最优控制是用一个性能函数来判断全局最优化。而预测控制对于动态特性变化和存在不确定因素的复杂系统无须在全局范围内判断最优化性能, 因此这种滚动优化方法很适用于这样的复杂系统。

预测控制也是一种反馈控制的算法。如果模型和过程匹配错误, 或者是由于系统的不确定因素引起的控制性能问题, 预测控制可以补偿误差或在线辨识校正模型参数。

预测控制主要包括非参数模型和参数模型两类控制算法, 比较流行的控制算法有: 模型算法控制(MAC)、动态矩阵控制(DMC)、广义预测控制(GPC)、广义预测极点(GPP)控制、内模控制(IMC)等。

各类预测控制算法都有一些共同的特点, 归结起来有三个基本特征: (1) 预测模型, (2) 有限时域滚动优化, (3) 反馈校正, 这三步一般由计算机程序在线连续执行。

7.2.4 鲁棒控制

所谓鲁棒性, 是指标称系统所具有的某一种性能品质对于具有不确定性的系统集的所有成员均成立。如果所关心的是系统的稳定性, 那么就称该系统具有鲁棒稳定性; 如果所关心的是用于干扰抑制性能或用其他性能准则来描述的品质, 那么就称该系统具有鲁棒性能。

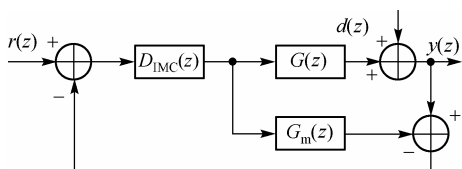


图 7.4 内模控制结构示意图

已有文献通过对模型加权函数的误差函数确定时滞, 然后与极点配置最优预报自校正 PID 控制器相结合, 使控制器随时滞及过程参数的变化而不断调整控制策略, 不但可以对参数时变、有扰动作用的大时滞系统进行控制, 而且也能应用于非最小相移系统, 对时滞系统能很好加以控制。

所谓鲁棒控制,就是设计一种控制器,使得当系统存在一定程度的参数不确定性及一定限度的未建模动态变化时,闭环系统仍能保持稳定,并具有一定的动态性能品质的控制。由于利用鲁棒控制来解决时滞系统的不确定性恰能发挥其特点,因而近年来有关时滞系统的鲁棒研究取得了很大的进展。

鲁棒控制的早期研究主要针对单变量系统在微小摄动下的不确定性,具有代表性的是 Zames 提出的微分灵敏度分析。然而,实际工业过程中故障导致系统中参数的变化不是无穷小摄动,而是有界摄动,因此产生了以讨论参数在有界摄动下系统性能保持和控制为内容的现代鲁棒控制。现代鲁棒控制着重研究控制算法的可靠性,其设计目标是找到在实际环境中为保证安全要求控制系统最小必须满足的条件。一旦设计好控制器,它的参数不需改变而且控制性能能够保证。

鲁棒控制方法在时间域或频率域中,一般要假设过程动态特性的信息和它的变化范围。一些算法不需要精确的过程模型,但需要一些离线辨识。一般鲁棒控制系统的设计是以一些最差的情况为基础,系统一般并不工作在最优状态。常用的设计方法有:INA 方法、同时镇定、完整性控制器设计、鲁棒控制、鲁棒 PID 控制及鲁棒极点配置、鲁棒观测器等。

7.2.5 变结构控制

变结构控制系统(VSS)的研究是由前苏联学者在 20 世纪 50 年代末开始的,但变结构在时滞系统中的应用只是在 20 世纪 90 年代以后才得到广泛的关注。变结构的优点在于系统分解和系统的滑动模态对某些干扰和参数摄动具有强鲁棒性,因此用它来控制一些参数变化剧烈、受扰动影响大的生产过程可以取得较好的控制效果。

变结构控制尽管具有控制速度快、鲁棒性好、对被控对象的参数变化和外界干扰不敏感等优点,但也经常因此造成振荡和失稳等问题,从而影响了变结构控制技术的推广,将预估控制引入变结构控制是解决此问题的较好方法。

7.2.6 智能控制

智能控制是以控制理论、计算机科学、人工智能、运筹学等学科为基础,扩展了相关的理论和技术,其中应用较多的有模糊逻辑、神经网络、专家系统、遗传算法等理论和自适应控制、自组织控制、自学习控制等技术。

智能控制技术的主要方法有模糊控制、基于知识的专家控制、神经网络控制和集成智能控制等。常用的优化算法有:遗传算法、蚁群算法、免疫算法等。

模糊控制以模糊集合、模糊语言变量、模糊推理为其理论基础,以先验知识和专家经验作为控制规则。其基本思想是用机器模拟人对系统的控制,就是在被控对象的模糊模型的基础上运用模糊控制器近似推理等手段,实现系统控制。在实现模糊控制时主要考虑模糊变量的隶属度函数的确定及控制规则的制定,二者缺一不可。

专家控制是将专家系统的理论与控制技术相结合,仿效专家的经验,实现对系统控制的一种智能控制。主体由知识库和推理机构组成,通过对知识的获取与组织,按某种策略适时选用恰当的规则进行推理,以实现对控制对象的控制。专家控制可以灵活地选取控制率,灵活性高;可通过调整控制器的参数,适应对象特性及环境的变化,适应性好;通过专家规则,系统可以在非线性、大偏差的情况下可靠地工作,鲁棒性强。

神经网络模拟人脑神经元的活动,利用神经元之间的联结与权值的分布来表示特定的信息,通过不断修正连接的权值进行自我学习,以逼近理论为依据进行神经网络建模,并以直接自校正控制、间接自校正控制、神经网络预测控制等方式实现智能控制。将神经网络的自学习机制和模糊控制机制相

结合构成的模糊神经网络控制器(FNC)既具有模糊逻辑的推理能力,又有神经网络很强的学习能力和非线性表达能力,是一种很好的时滞系统控制方法。

7.3 恒温控制系统的设计与实现

1. 设计任务

设计一个高精度恒温度控制系统,控制对象为一块 $150\text{ mm}\times 80\text{ mm}\times 20\text{ mm}$ 的铁块,其温度可以在一定的范围内由人工设定。

2. 设计基本要求

- (1) 温度设定范围为 $40^{\circ}\text{C}\sim 100^{\circ}\text{C}$,最小设定分度为 1°C 。
- (2) 用 3 位数字显示温度,分辨率为 0.1°C ,在温度设定范围内显示温度和标准温度之间的误差 $\leq 1^{\circ}\text{C}$ 。
- (3) 温度控制稳态误差的绝对值 $\leq 0.5^{\circ}\text{C}$ 。
- (4) LED 显示当前温度和人工任意设定的温度。

3. 设计提高部分要求

- (1) 温度控制稳态误差的绝对值 $\leq 0.1^{\circ}\text{C}$;在任意温度尽量减小超调量。
- (2) 系统有较强的抗干扰能力。使用一台电扇对铁块进行吹风,要求系统的温控性能不变。
- (3) 当设定温度从 50°C 突变到 70°C 时,超调量 $\leq 1.5^{\circ}\text{C}$,并尽量减少调节时间。
- (4) 通过串行口和 PC 通信,显示被控对象的温度响应曲线。

说明:

(1) 加热元件采用电烙铁芯, 220 V 交流供电,其加热功率应通过热量平衡计算及综合控制要求自行决定。

(2) 控制对象铁块的示意图如图 7.6 所示。加工方式为:在铁块的侧部中间钻一个 $\phi 5\sim\phi 6$ 的孔,能将电烙铁芯插入即可,深度 40 mm ;在铁块的正中间钻一个 $\phi 8$ 的孔,深度 15 mm ,不钻穿;供插入标准温度计进行计量检测;温度传感器可以安装在铁块左部距离边界 50 mm 处,钻孔方式和标准温度检测孔相同,上下对称。

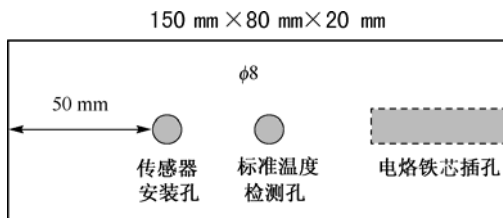


图 7.6 控制对象铁块的示意图

7.3.1 设计任务分析

根据题目要求,控制系统的硬件电路的设计必须相对于控制算法是开放的,采用模拟调节器显然不能适应这一要求。因此,系统设计必须采用直接数字控制(Direct Digital Control, DDC)系统。用 MCU 取代模拟调节器直接控制执行器,使被调量保持在给定值。其硬件结构以 MCU 为控制核心,前向通道由温度测量部分组成,后向通道由功率控制部分组成,DDC 系统的总体框图如图 7.7 所示。

本系统是一个典型的闭环控制系统,对象是一块 $150\text{ mm}\times 80\text{ mm}\times 20\text{ mm}$ 的铁块,铁的比热是 $0.46\times 10^3\text{ J}/(\text{kg}\cdot^\circ\text{C})$ 。通过对被控对象中的加热元件电烙铁芯的平均功率进行控制,达到对铁块温度控制的目的。系统中采用一片 MCS 8051 单片机作为主控制器,前向通道为测温部分,后向通道为控制部分。

为了实现对铁块的温度控制,首先要求出被控对象的数学模型。令加热器全功率开通,测试被控对象(铁块)的温度从室温分别上升到 40°C 、 50°C 、 60°C 、 70°C 、 80°C 、 90°C 时的热惯性和温度变化,可得到被控对象(铁块)的温升响应曲线,如图 7.8 所示。

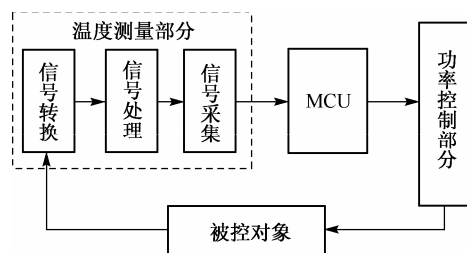


图 7.7 DDC 系统的总体框图

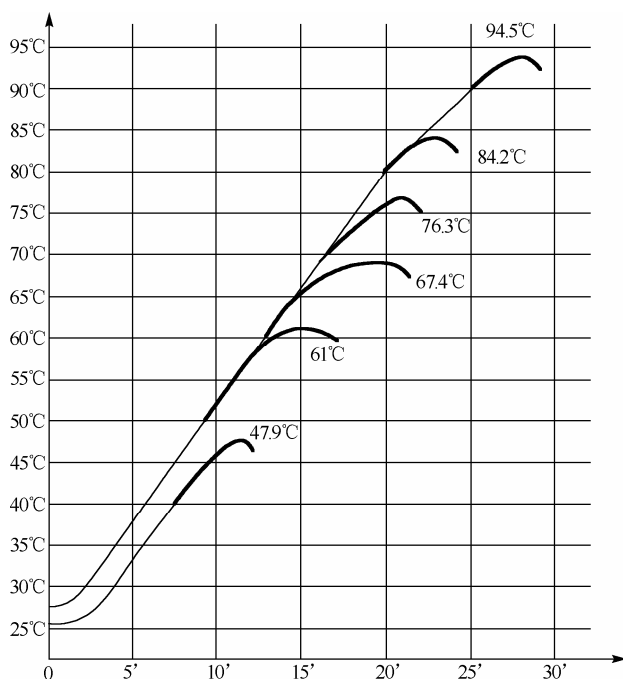


图 7.8 铁块的温升响应曲线

从响应曲线来看,当加热器全功率开通后,铁块的温度并不是即时跟随的,有一滞后时间。当铁块从室温被加热到预定温度时,虽然加热器已被关断,但受热惯性的影响,铁块温度依然有一上升区间(图中温升曲线瞄黑部分)。因此可看出被控对象具有非线性、滞后、大惯性的特点,在低温段惯性较大,在高温段惯性较小,被控对象的变增益特性明显。因此该控制对象为一个带有纯滞后的一阶惯性环节,其传输函数为

$$G(s) = \frac{K_p e^{-\theta s}}{\tau_1 s + 1} \quad \theta = NT \quad (7.4)$$

式中, K_p 为系数; τ_1 为被控对象的时间常数; θ 为纯滞后时间,一般假定它是采样周期 T 的整数倍; N 为正整数。

由于被控对象具有较大的纯滞后时间,它使系统的稳定性降低,过渡过程特性变坏,当被控对象

的纯滞后时间较大时,采用常规的 PID 类算法很难获得良好的控制性能,需要选择合适的控制算法才能满足设计要求。同时,为了达到良好的温度控制精度,要求系统的前置温度采样电路和功率控制电路有着良好的性能,高精度的信号测量和精确的功率控制是保证温度控制精度的关键。

7.3.2 系统硬件电路方案设计及分析

1. 温度信号采集

系统的前端通道即信号采集测温部分,此部分由信号转换、信号处理和信号采集三部分实现。就控制系统而言,控制精度不仅取决于算法,测量精度更是保证控制精度的关键,系统设计的第一要素是如何实现高精度的测量,也就是说,只有准确地测量才可以保证控制的精确。因此,前置信号处理、采集的测量电路的设计指标和电路的优化至关重要。

信号转换部分可采用温度传感器 LM35 将温度信号变换为电信号,同时考虑到传感器和后继电路之间匹配和抗干扰问题,可在温度传感器后接入由同相放大器构成的阻抗匹配、隔离电路。LM35 具有很高的工作精度和较宽的线性工作范围,它的输出电压与摄氏温度成线性比例,且无需外部校准或微调。从使用角度来说,LM35 与用开尔文标准的线性温度传感器相比更有优越之处,其相关特性如下:

- 工作电压: 直流 4~30 V;
- 工作电流: 小于 133 μA ;
- 输出电压: 0~1.0 V;
- 输出阻抗: 1 mA 负载时 0.1 Ω ;
- 精度: 0.5 $^{\circ}\text{C}$ (在+25 $^{\circ}\text{C}$ 时);
- 漏泄电流: 小于 60 μA ;
- 比例因数: 线性+10.0 mV/ $^{\circ}\text{C}$;
- 非线性值: $\pm 1/4^{\circ}\text{C}$;
- 校准方式: 直接用摄氏温度校准;
- 封装: 密封 TO-46 晶体管封装或塑料 TO-92 晶体管封装;
- 使用温度范围: -55 $^{\circ}\text{C}$ ~+150 $^{\circ}\text{C}$ 额定范围。

温控系统中常用两种方法采集前端对象的温度信号:一是采用模数转换器,温度传感器输出的电压或电流信号经过处理后,使用 A/D 转换器进行采样获得温度信号;另一种方法是采用 V-f 变换,将温度传感器输出的电压或电流信号转换为频率信号,通过频率计测频获得对应的温度信号。以下对这两种方法分别进行讨论。

1) 采用 ADC 芯片进行温度信号的采集

(1) ADC 芯片的选取

在第 5 章中已对数据采集及 A/D 转换器的字长的选取进行了详细的讨论。结合本节的训练任务由图 7.8 中被控对象的传输特性曲线可看出,温度信号是一慢变信号,被控对象的时间常数较大,因此对 A/D 转换器的转换速率要求不高,但对精度有较高的要求。由系统对误差(或精度)的要求确定字长时,考虑到 LM35 的性能指标及系统设计指标的要求,一般要求 A/D 转换器的精度应高于传感器的精度,使得系统的总误差尽量不是由 ADC 引入的。8 位 A/D 转换精度只能达到 $\pm 0.2\%$,显然满足不了系统精度的要求;12 位 A/D 转换精度为 $\pm 0.012\%$,可满足系统设计的指标要求,因此可选用 12 位 A/D 转换器 MAX197。MAX197 是一个多量程的 12 位 A/D 转换器,其主要特点如

下:

- 具有 12 位分辨率和 1/2 LSB 的线性度;
- 单一+5 V 供电;
- 可选择的输入电压范围为 $-10\sim 10\text{ V}$, $-5\sim 5\text{ V}$, $0\sim 10\text{ V}$, $0\sim 5\text{ V}$;
- 输入通道耐压至 $\pm 16.5\text{ V}$;
- 8 路模拟输入通道;
- $6\text{ }\mu\text{s}/\text{D}$ 转换时间, 100 kps (sample percent second) 采样速率;
- 内外采样模式可选;
- 内外时钟可选。

MAX197 的其他特点还包括具有 5 MHz 带宽的跟踪/保持电路、8+4 位并行接口、软件可选内部或外部时钟、可变的采集控制及内置 4.096 V 电压基准或可选的外部基准源。由于 MAX197 片内包含高精度的参考电压源和时钟电路, 使用标准的微处理器接口单元, 数据存取和总线释放的时序与大多数通用的微处理器相兼容。因此, MAX197 可以在不需要外部参考电压源和时钟电路及其他外部电路的情况下完成 A/D 转换功能, 应用非常方便。

(2) 信号处理部分的设计

由于 LM35 输出电压范围为 $0\sim 1\text{ V}$ (对应 $0\sim 100^\circ\text{C}$), 灵敏度为 $10\text{ mV}/^\circ\text{C}$, 系统设计的指标要求分辨率为 0.1°C (即 1.0 mV 信号电压必须能准确测量), 为了取得较高的测量精度, 必须合理设计输入信号量程 (对应于 A/D 转换器的动态范围), 应采用分段测量的方法。所谓“分段测量”, 就是将事先规定的测量范围划分为多个不同的量程, 然后在测量过程中根据读取的信号数据大小来选择不同的量程。这个动作非常类似于人工使用万用表来测量不同的电参数, 只是后者是手工拨动选择开关, 而前者是自动选择开关。分段测量的意义在于: 若信号输入量程过大 (高增益, 大尺度测量), 在测量信号的量值较大时将会产生信号失真; 若输入信号量程过小 (低增益, 小尺度测量), 在测量信号的量值较小时将会降低测量信号的信噪比。所以, 合理的设计应在噪声和失真度之间达到一个平衡。可以选取的方式包括程控可变增益同相放大器、程控可变增益差分放大器和数字控制的精密仪表放大器等。

① 方案一: 程控可变增益同相放大器

由于 MAX197 为多通道 A/D 转换器, 因此可选用 4 个放大倍数分别为 5 倍、7 倍、10 倍和 17 倍的同相放大器连同电子模拟开关一起组成程控可变增益同相放大器, 通过多路电子模拟开关 MAX333 控制各路的选择后送 MAX197 进行采样、量化。已知 LM35 输出温度信号电压范围是 $0\sim 1\text{ V}$, 且 A/D 转换器 MAX197 采用单极性 $0\sim 5\text{ V}$ 输入, 为了确保精度, 当 LM35 输出温度信号电压为 $0\sim 0.2\text{ V}$ 时, 选择 17 倍同相放大器; 当输出温度信号电压为 $0.2\sim 0.5\text{ V}$, 选择 10 倍同相放大器; 当 LM35 输出温度信号电压为 $0.5\sim 0.7\text{ V}$ 时, 选择 7 倍同相放大器; 当 LM35 输出温度信号电压为 $0.7\sim 1\text{ V}$ 时, 选择 5 倍同相放大器。所采用的电路如图 7.9 所示。

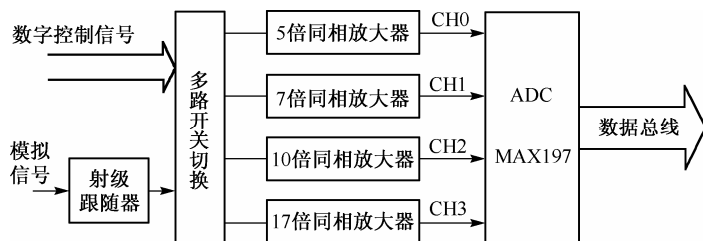


图 7.9 程控可变增益同相放大器

② 方案二：程控可变增益差分放大器

程控可变增益差分放大器由前置同相放大级和差分放大级组成。由宽带高精度运放 MAX400 构成前置同相放大级，考虑到和 LM35 温度传感器的阻抗匹配，前置同相放大级的单位增益为 1，连接成射级跟随器，差分放大级采用改进型差分放大电路，如图 7.10 所示。合理选择参数，使得差分放大器的输出 $V_{03} = 10(V_{01} - V_{02})$ ，考虑到和 A/D 转换器的缓冲和匹配，A4 设计为射级跟随器，A4 的输出电压 $V_0 = V_{03}$ 。假设基准电平 $V_{02} = 0\text{ V}$ ，当 LM35 输出温度信号电压为 $0 \sim 0.5\text{ V}$ 时，A3 输出电压 $V_{03} = V_0$ 为 $0 \sim 5\text{ V}$ ，满足 A/D 转换要求；假设基准电平 $V_{02} = 0.5\text{ V}$ ，当 LM35 输出温度信号电压为 $0.5 \sim 1\text{ V}$ 时，A3 输出电压 $V_{03} = V_0$ 为 $0 \sim 5\text{ V}$ ，同样满足 A/D 转换要求。由于小于 1.2 V 的低阻驱动电压基准源难以获得，可采用 16 位 D/A 转换器 MAX542 构成数控基准电压源。

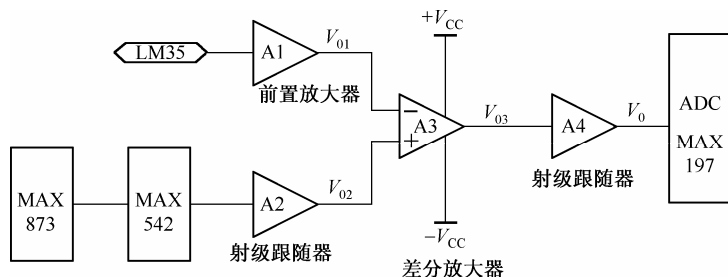


图 7.10 程控可变增益差分放大器电路

③ 方案三：数字控制的精密仪表放大器

MAX5426 是理想的可编程仪表放大器，通过两个数字输入端控制可编程仪表放大器的放大增益，可以为 1 倍、2 倍、4 倍或者 8 倍。考虑到该可编程仪表放大电路要求双极性的输入信号，而由传感器所变换得到的电信号是单极性信号，所以在放大电路前端需要增添单-双变换电路。

采用可编程放大器 MAX5426 实现测量量程四挡之间的切换：

- 若电压幅值为 $0 \sim 0.5\text{ V}$ ，则通过 MAX5426 的引脚 (D1D0) = 11 将增益设定为 8 倍。
- 若电压幅值为 $0.5 \sim 1\text{ V}$ ，则通过 MAX5426 的引脚 (D1D0) = 10 将增益设定为 4 倍。
- 若电压幅值为 $1 \sim 2\text{ V}$ ，则通过 MAX5426 的引脚 (D1D0) = 01 将增益设定为 2 倍。
- 若电压幅值大于 2 V ，则通过 MAX5426 的引脚 (D1D0) = 00 将增益设定为 1 倍。

数字控制的精密仪表放大器电路如图 7.11 所示。

信号处理部分的设计方案比较：

方案一中，通过多路电子模拟开关实现对 4 个具有不同放大倍数的同相放大器进行控制与切换，实现对不同的信号（尤其是小信号）的不同倍数放大来提高精度和信噪比。理想情况下这种方案是可以实现的，但是由于模拟开关存在接通电阻及各开关间的隔离度不够高，使得系统要求的精度难以达到。

方案二中，使用程控可变增益差分放大器，由单片机控制 D/A 转换器构成数控基准电压源，D/A 转换器输出的标准电压作为基准，利用差分放大器电路的工作原理，使其能够准确调节差分放大器的放大倍数，使得前向温度信号采集电路能够达到对温度信号的精密测量要求，对后续电路中温度的控制算法也不会有很大的影响。整个前置放大电路不仅改善了小信号的测量精度，而且较好地解决了传感器和前置放大器的匹配及 A/D 转换器和缓冲放大电路的匹配和缓冲问题。而且，这种设计方

计的分辨率不低于 1 Hz 即可满足测量对于分辨率的要求。假设本设计中温度测量范围大于 100℃, 为 0~150℃, 相应的温度传感器输出电压范围为 0~1.5 V。通过调整 V-f 变换电路, 可以使 LM331 的输入电压为 1 V 时, 输出频率为 1 kHz。此时 V-f 变换电路 0~1.5 V 的输入电压变化对应 0~1500 Hz 的输出频率变化。问题是就数字频率计而言, 虽然采用了等精度测频的方法, 但在测量较低频率信号时, 由于计数基数较小, 误差显然要大一些。因此, 应对输入信号做适当的变换, 数字频率计对 2.5~5.5 kHz 的频率信号有较高的测量精度, 相对测量时间也较短。所以可考虑对输入信号电压从 0~1.5 V 到 -5.5~-2.5 V 的线性转换, 以满足测量精度的要求。设计中可使用浮地式差分放大器完成 0~1.5 V 到 -5.5~-2.5 V 的电压转换。

图 7.13 为一浮地式差分放大器, 可以看到这种放大器和一般放大器不同, 那就是它是“浮地”的, 即放大器输出被“浮”在一基准电位上, 这个电位通常称为“模拟地”。其目的是通过调整放大器的输入参考基准电位, 使得输出电压 U_o 正好在给定的工作电压区间内。

其中 U_{11} , U_{12} 为传感器输出的差分信号, U_o 为两级运放的输出, U 为电压基准, A1 和 A2 为运算放大器。设 U_1 为 A1 的输出电压。则

$$\frac{U_1 - U_{11}}{R_2} = \frac{U_{11} - U}{R_1} \quad (7.5)$$

$$\frac{U_o - U_{12}}{R_4} = \frac{U_{12} - U_1}{R_3} \quad (7.6)$$

联立式(7.5)和式(7.6), 可得

$$U_o = \left(1 + \frac{R_4}{R_3}\right)(U_{12} - U_{11}) + \left(1 - \frac{R_2 R_4}{R_1 R_3}\right)U_{11} + \frac{R_2 R_4}{R_1 R_3}U \quad (7.7)$$

可见, 整个放大电路的输出包括三个部分:

第一部分: $\left(1 + \frac{R_4}{R_3}\right)(U_{12} - U_{11})$ 为传感器输出信号的差模放大增益, 这正是所期望的。

第二部分: $\left(1 - \frac{R_2 R_4}{R_1 R_3}\right)U_{11}$ 为放大器共模增益, 是所不期望的。

第三部分: $\frac{R_2 R_4}{R_1 R_3}U$ 为放大器调零输出, 与“浮地”有关系。

为了使比例差分放大器抑制共模增益, 令

$$\left(1 - \frac{R_2 R_4}{R_1 R_3}\right)U_{11} = 0 \quad (7.8)$$

即要求

$$R_2 R_4 = R_1 R_3 \quad (7.9)$$

实际设计时可选取 $R_1 = R_4$, $R_2 = R_3$, 因此整个电路输出为

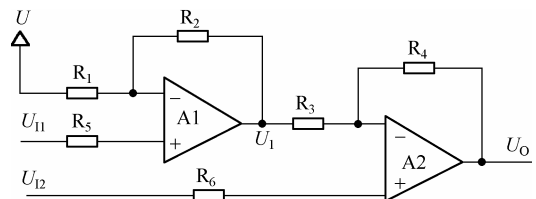


图 7.13 浮地式差分放大器

$$U_O = \left(1 + \frac{R_4}{R_3}\right)(U_{I2} - U_{I1}) + U \quad (7.10)$$

选取 $R_4 = R_3$, $U = -2.5 \text{ V}$, 并让 U_{I2} 接地, U_{I1} 接传感器输出电压即可满足设计要求; 传感器输出电压被放大两倍, 加上基准电压 U 后, 输出电压为 $-5.5 \sim -2.5 \text{ V}$, 对应于温度 $0 \sim 150 \text{ }^\circ\text{C}$ 。

而 $U = -2.5 \text{ V}$ 可以由 2.5 V 电压基准芯片 MAX873 和一个反相器产生, 参考实现电路如图 7.14 所示。

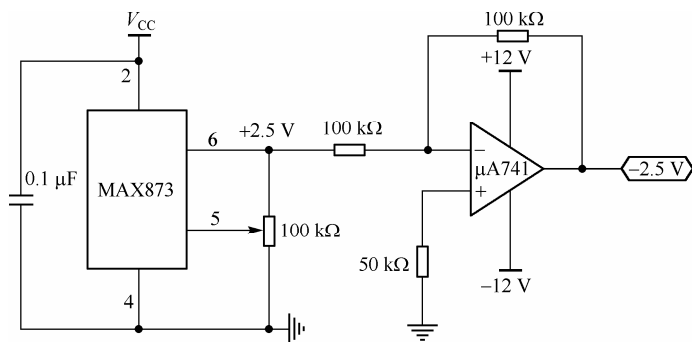


图 7.14 -2.5 V 电压基准电路

在设计前置通道电路时, 需要注意以下两点:

- (1) 选择共模抑制比高的精密集成运算放大器, 以提高电路的共模抑制比及增益线性度。
- (2) 选用低噪声高精度的电阻, 以减小由电阻噪声所引起共模抑制比下降。

电路设计时, 应采用共模抑制比高的集成运算放大器, 如 MAX400 或和其性能指标相同的其他集成运算放大器, 电阻可采用精度为 1% 的金属膜电阻。

2. 功率驱动控制

功率控制通常有两种方法: 一是调功, 通过控制发热器的通断来控制, 即控制加在功率器上的正弦波的波头数来控制功率, 也就是说, 在一定的门限时间内通过给定数目的波头数来实现对功率的控制, 对于本系统来说, 可以利用纯数字电路 CPLD 控制单位时间内加入电烙铁热丝的波头数, 从而调节电烙铁的输出平均功率; 二是调相, 通过在给定时刻控制导通角来精确控制导通时加在功率器上的电压幅值, 实现对功率器的精确均匀控制。分析这两种控制方法: 调功只能实现对功率的粗略控制, 而题目要求的是实现对功率的精确控制, 显然这种方法不是最好的选择; 而调相可以实现均匀精确的控制功率, 能够实现对功率的微调, 可准确地控制功率大小, 从而实现到达定值温度时的平滑过渡。

下面讨论全数字锁相功率调节器的设计。

要通过调相控制功率, 必须随时知道并记录 220 V 电源的相角, 从而准确地控制导通时刻, 故需要对 220 V 电源的相角同步信号进行跟踪记录。为此, 采用变压器提取 220 V 的同步电压信号, 并用比较器跟踪记录 220 V 电源的过零点, 从而输出与 220 V 电源同相、同频率的脉冲。为进行精细功率控制, 将半个周期的交流电分为 N 份(这里选用 $N = 255$), 其中 $1/N$ 既为调相的步长, 又定义为内部计时器的计数单位。每次计数在过零点开始, 当计数输出量与输入数字控制量相等时, 产生触发脉冲使可控硅开启; 下一个过零点关闭(可控硅特性应为随机开启, 过零关闭), 当改变输入数字控制量的大小时, 可控硅的导通相角随即对应改变, 从而实现对功率器上所加有效电压大小的控制。由于要求计数脉冲与 220 V 电源完全同步, 采取锁相环电路, 使计数脉冲与 220 V 电源的工频信号完全同步, 实现对功率器的发热功率从微小到可能的最大功率的平滑调节。

在传统的控制系统中，一般利用模拟锁相环电路提取市电的同步信号，从而实现同步功率调节器的功能，具体实现电路框图如图7.15所示。较为便利的设计方法是依照模拟锁相环的原理利用可编程逻辑器件，采用边沿检测电路实现同步信号的提取，在设计中省去了模拟锁相环器件，效果相同但简化了电路结构，设计效率更高。采用这一方法的全数字同步功率调节电路原理框图如图7.16所示，图中方框内的硬件电路部分可由 CPLD 器件实现。

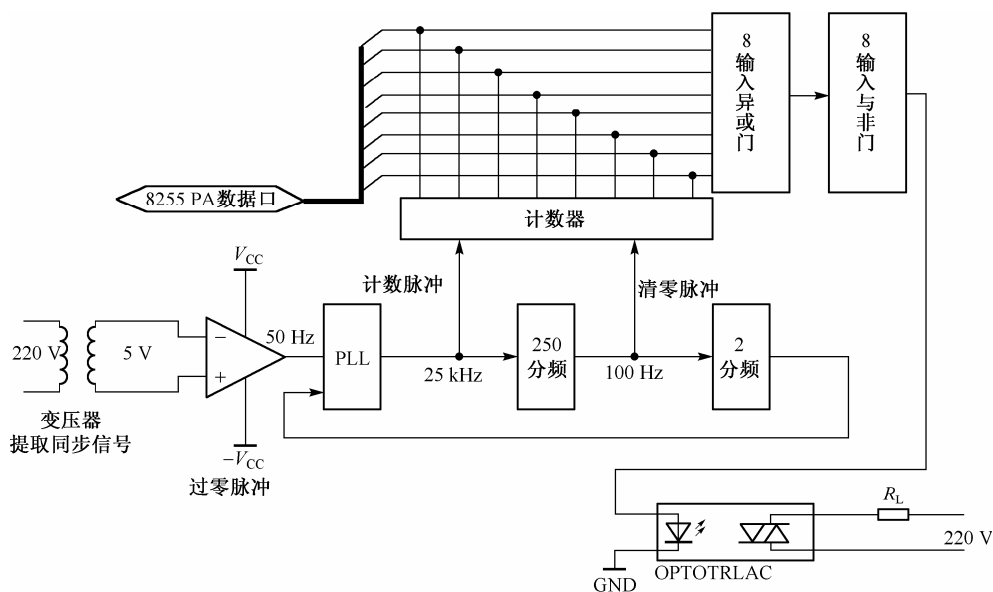


图 7.15 基于锁相环电路的同步功率调节器实现电路框图

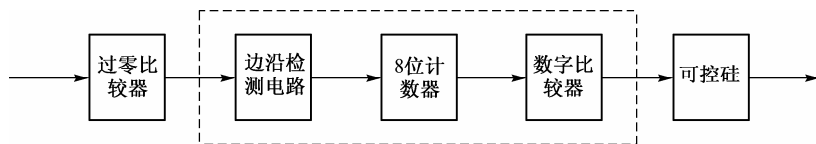


图 7.16 全数字同步功率调节电路原理框图

具体实现方法描述如下:

(1) 采用变压器将 220 V 市电转换为同步的峰值为 5 V 的正弦波电压信号, 然后经过过零比较器把正弦波电压信号转换为占空比 1:1 的 50 Hz 方波信号。

(2) 50 Hz 的方波信号通过边沿检测电路得到同相的 100 Hz 的窄脉冲信号, 其边沿检测电路如图 7.17 所示。

(3) 由晶振电路产生 1 MHz 的时钟信号, 对其 40 分频后产生 25 kHz 的频标信号。8 位计数器对 25 kHz 的频标信号进行计数, 计数器具备一个上升沿清零端, 该端的接入信号为 (2) 中得到的 100 Hz 的窄脉冲, 该窄脉冲以频率 100 Hz 对计数器清零, 使得计数的开始时刻为 50 Hz 市电信号的过零处, 从而保证严格同步。

(4) 8 位计数器的计数值输入到数字比较器中, 与单片机设置的预定值进行比较。每次计数在过零点开始, 当计数值与单片机输入的预定值相等时, 产生触发脉冲使可控硅开启, 下一个过零点关闭。每次改变输入数字控制量的大小, 即可改变导通时间。这样输出与市电同步的、周期性的方波信号去控制可控硅的导通角, 通过改变单片机输入值的大小可以方便地调节可控硅的导通角, 准确地实现功率调节。需要指出的是, 采用调相方式对电网会造成一定的污染, 相对于调功方式噪声耦合也大一些。全数字同步功率调节电路的信号时序关系如图 7.18 所示。

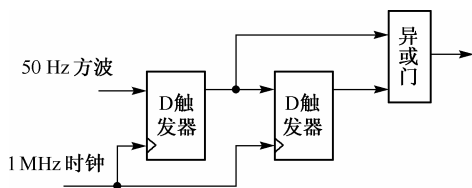
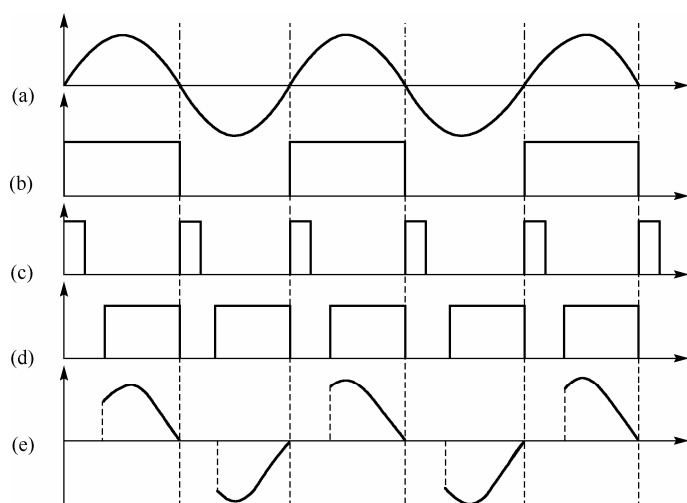


图 7.17 边沿检测电路



(a)—变压器输出交流信号；(b)—过零比较输出波形；(c)—清零脉冲；(d)—开关控制信号；(e)—过零关断的交流波形

图 7.18 全数字同步功率调节电路的信号时序关系

7.3.3 控制策略及算法实现与比较

由于被控对象是一个具有一定自衡能力的惯性系统，具有较大的纯滞后时间。采用常规的 PID 算法很难获得良好的控制性能，需要选择合适的控制算法才能满足控制要求。一个好的控制系统，应能在尽量短的时间内达到最佳控制点，使系统具有良好的动态性能和静态指标，其控制系统品质要求可以概括为稳、准、快。具体的工作品质指标包括最大偏差（即超调量，被调参数在过渡过程中偏离给定值的最大幅度）、过渡时间（从被调参数变化之时起，直到被调参数进入新的稳态值的 $\pm 5\%$ 所需要的时间）、静差（过渡过程结束后，被调参数的新稳定值与给定值之差）。针对被控对象特性，控制策略及算法实现如 7.2 节所述有多种方法，限于篇幅，选择以下三种控制算法来讨论它们的实现过程，并对实现结果做出比较。

1. 大林算法

大林算法的设计目标是：设计或选择合适的数字调节器，使整个闭环系统的传输函数是带纯滞后时间的一阶惯性环节，而且要求闭环系统的纯滞后时间等于被控对象的纯滞后时间，从而使得闭环系统的调节品质满足时滞被控系统的控制要求。由于对系统的模型结构与参数有严格的要求，因而消除了由于超调引起的系统不稳定等因素的影响。大林算法本身是在计算机发展的基础上提出的，它特别适用于直接数字控制 (DDC) 系统。

本节设计任务为：利用电阻丝加热—供测量用的简易的金属热平衡块系统，受控对象可看成是一个具有一定自衡能力的惯性系统，可用一阶惯性环节和一个延迟环节（零阶保持器）来近似。已知控制对象为一个带有纯滞后的一阶惯性环节，其传输函数为

$$G(s) = \frac{K_p e^{-\theta s}}{\tau_1 s + 1} \quad (7.11)$$

式中， τ_1 为被控对象的时间常数； θ 为被控对象的纯延时时间，设它为采样周期 T 的整数倍， $\theta = NT$ (N 为正整数)。

设含有纯滞后对象的计算机控制系统方框图如图 7.19 所示。其中， $R(s)$ 为设定值； $E(s)$ 为偏差；

$D(z)$ 为数字调节器; $Y(s)$ 为调节器的输出函数;
 $H(s)$ 为零阶保持器, $H(s) = (1 - e^{-Ts})/s$; $G(s)$ 为
 纯滞后对象的传输函数; $W(s)$ 为被测量。

由图7.19可看出, 被控对象与一个零阶保持器
 相串联, 相应的整个闭环系统的脉冲传递函数为

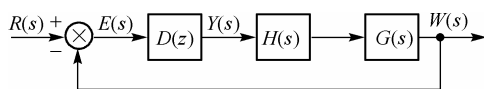


图 7.19 含有纯滞后对象的计算机控制系统方框图

$$\Phi(z) = \frac{W(z)}{R(z)} = \frac{z^{-N-1}(1 - e^{-T/\tau})}{1 - e^{-T/\tau}z^{-1}} \quad (7.12)$$

$$\text{代入} \quad D(z) = \frac{\Phi(z)}{G(z)[1 - \Phi(z)]} \quad (7.13)$$

$$\text{可得} \quad D(z) = \frac{1}{G(z)} \times \left[\frac{z^{-N-1}(1 - e^{-T/\tau})}{1 - e^{-T/\tau}z^{-N-1} - z^{-N-1}(1 - e^{-T/\tau})} \right] \quad (7.14)$$

$D(z)$ 就是要设计的数字控制器, 它可由计算机程序来实现。而式中 $G(z)$ 为广义对象的 z 传递函数, $G(z) = Z[H(s)G(s)]$, 由式(7.11)和式(7.12)经计算可得

$$G(z) = K_p z^{-N-1} \left[\frac{1 - e^{-T/\tau_1}}{1 - e^{-T/\tau_1}z^{-1}} \right] \quad (7.15)$$

$$\text{从而} \quad D(z) = \frac{(1 - e^{-T/\tau_1}z^{-1})(1 - e^{-T/\tau})}{K_p(1 - e^{-T/\tau_1})[1 - e^{-T/\tau}z^{-1} - (1 - e^{-T/\tau})z^{-N-1}]} \quad (7.16)$$

式中, T 为采样周期; τ_1 为被控对象的时间常数; τ 为闭环系统的时间常数。

式(7.16)经化简后得到

$$D(z) = \frac{A - Bz^{-1}}{1 - Cz^{-1} - (1 - C)z^{-N-1}} \quad (7.17)$$

式中, $A = \frac{1 - e^{-T/\tau}}{K_p(1 - e^{-T/\tau_1})}$; $B = Ae^{-T/\tau_1}$; $C = e^{-T/\tau}$ 。

因为 $D(z) = Y(z)/E(z)$, 则

$$[1 - Cz^{-1} - (1 - C)z^{-N-1}]Y(z) = (A - Bz^{-1})E(z) \quad (7.18)$$

对上式求 z 逆变换, 可得大林算法的数字控制器的最终表达式为

$$y(n) = Ae(n) - Be(n-1) + Cy(n-1) + (1 - C)y(n - N - 1) \quad (7.19)$$

式中, $y(n)$ 为 n 时刻的输出值; $e(n)$ 为 n 时刻的误差值; $e(n-1)$ 为 $n-1$ 时刻的误差值; $y(n - N - 1)$ 为 $n - N - 1$ 时刻的输出值。

由式(7.19)给出的大林算法, 可以方便地使用计算机对其进行编程。

单片机对铁块进行温度控制时, 由单片机检测被控对象的当前温度, 并与设定值相比较, 得到温度误差值 $e(n)$ 。单片机根据当前误差 $e(n)$ 、上次误差 $e(n-1)$ 及上次控制量 $y(n-1)$ 、前 $(N+1)$ 次控制量 $y(n - N - 1)$, 计算得到现行控制量 $y(n)$ 。因此, 当前控制量 $y(n)$ 控制电压序列值可以很容易地由计算机通过算法得到, 并且可通过适当选择系数 A, B, C 使 DDC 系统具有最小超调的系统响应。

下面讨论大林算法的参数整定。

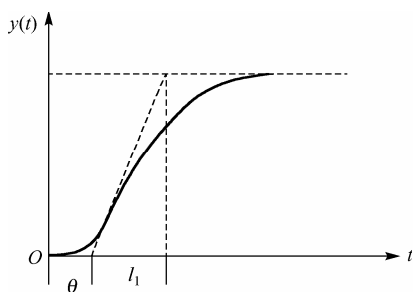


图 7.20 被控对象的飞升曲线

(1) 纯滞后时间参数 θ 和被控对象的时间参数 τ_1 的确定由被控对象的飞升曲线(如图 7.20 所示)确定对象的纯滞后时间参数 θ 和被控对象的时间参数 τ_1 。

(2) 采样周期 T 的确定 综合控制精度、超调量等指标选取合适的采样周期 T 。若 T 偏大,则取样稀疏,单位时间内控制点过少,势必造成较大的过冲量及系统控制误差;若 T 偏小,则对采样量化字长要求过高,而且对于有限字长的控制系统,过密的采样周期会使系统控制失败。

(3) N 值的确定 由 $N = \theta/T$ 可以确定 N 值。

(4) 被控对象放大系数 K 的确定 K 可由下列公式确定:

$$K = \frac{\frac{\Delta z \text{ (测量值的变化量)}}{z_{\max} - z_{\min} \text{ (测量仪器刻度范围)}}}{\frac{\Delta P \text{ (调节器输出变化量)}}{P_{\max} - P_{\min} \text{ (调节器的输出变化范围)}}} \quad (7.20)$$

(5) 闭环系统的时间参数 τ 的确定

由被控对象纯滞后部分的传输函数 $e^{-T/\tau}$ 计算 τ , τ 一般与 T 取同量级,在设计实验中需不断调整 τ 值,观察系统的响应图,使得闭环系统的指标达到最佳。

2. 基于最优化控制理论的智能控制法

判断控制效果的好坏,最重要的参数就是控制系统在不同时刻输出的偏差及偏差的变化率。控制过程即是对某一给定输入值,寻求对应最佳控制点的过程。根据上述原理,在重视超调指标的情况下,采用优化控制理论与智能控制相结合,寻优控制参数,可使智能控制的质量明显提高。

本系统以三个决策层和若干控制规则实现对系统的过程控制,不同的决策层对应不同的偏差带,控制规则表达为一组条件语句,状态条件和控制作用均为一组被量化的信息语言,共同构成微机的控制算法。

(1) 决策一

- ① IF $P=0$ AND $E > M_{\text{冲}}$, THEN 驱动值 $U = FFH$;
- ② IF $P=0$ AND $E < M_{\text{冲}}$, THEN 驱动值 $U = 00H$;
- ③ IF $E < 0$ THEN 驱动值 $U = 00H$;
- ④ IF $P=1$ AND $0 < E < M_{\text{误}}$ AND $\delta_N > \delta_{N-1}$ THEN 驱动器 $U_N = 0.618U_{N-1}$;
- ⑤ IF $P=1$ AND $0 < E < M_{\text{误}}$, AND $\delta_N < \delta_{N-1}$ THEN 驱动值保持;
- ⑥ IF $P=1$ AND $E > M_{\text{误}}$ THEN $b = U_{N-1}$, $a = U_N$, 驱动值 $U = b - 0.618(b-a)$;
跳转至决策二。

(2) 决策二

- ⑦ IF $E < 0$ THEN 驱动值 $U = 00H$;
- ⑧ IF $E > 0$ AND $T_K > T_{K-1}$, THEN a 保持, $b = U_K$ 驱动值 $U = b - 0.618(b-a)$;
- ⑨ IF $E > 0$ AND $T_K < T_{K-1}$, THEN b 保持, $a = U_K$, 驱动值 $U = b - 0.618(b-a)$;
- ⑩ IF $E > 0$ AND $T_K = T_{K-1}$, THEN 驱动值保持, 跳转至决策三。

(3) 决策三

- ⑪ IF $E \geq 0$ AND $E \leq M_{\text{误}}$, AND $T_K \geq T_{K-1}$, THEN 驱动值保持;
- ⑫ IF $E > 0$ AND $E < M_{\text{误}}$ AND $T_K < T_{K-1}$, THEN 驱动值 $+ \Delta$;

- ⑬ IF $E > 0$ AND $E > M_{\text{误}}$ THEN 驱动值 $+\Delta$;
 ⑭ IF $E < 0$ AND $|E| < M_{\text{误}}$ AND $T_K > T_{K-1}$, THEN 驱动值 $-\Delta$;
 ⑮ IF $E < 0$ AND $|E| < M_{\text{误}}$ AND $T_K \leq T_{K-1}$, THEN 驱动值保持;
 ⑯ IF $E < 0$ AND $|E| > M_{\text{误}}$ THEN 驱动值 $-\Delta$ 。

上述规则中, E 表示偏差, $E = G - T$, G 为温度给定值, T 为温度采样值, $P=0$ 表示温度未曾升至给定值, $P=1$ 表示温度曾升至给定值; $M_{\text{误}}$ 为各决策层中的允许误差值; δ 为温度过冲值, Δ 为微小数字增量。

驱动值数字范围 00H~FFH, 00H 和 FFH 分别对应可控硅最小、最大导通角。为了抑制全速升温后系统升温的严重超调, 规则中设置了 $M_{\text{冲}}$ 死区, 当升温进入该区后, 立即关断可控硅, 之后铁块靠热惯性升至给定值。由于最初控制驱动值偏离真值较远, 加之系统的纯时滞, 由偏差量和偏差变化率难以决策控制参数, 因此以超调量 δ 为寻优目标参数。三个决策层中, 采样周期各不相同, 决策二的采样周期较长。

本控制器不依赖于对象数学模型, 省掉了烦琐的参数整定环节。但实验表明, 此算法存在以下几个方面的缺点:

(1) 此算法中的三个决策不可逆。设计中控制的小铁块易受外界干扰, 一旦干扰产生, 程序将产生错误迭代, 错误一旦产生, 就再也无法回到正常的控制过程, 导致控制的彻底失败。故此算法只适用于被控物体不易受外界干扰的系统控制过程中。

(2) $M_{\text{冲}}$ 和 $M_{\text{误}}$ 参量的选择相当困难。 $M_{\text{冲}}$ 的选取要保证第一次上升过程有过冲, 而且过冲不能太大; 各个决策的 $M_{\text{误}}$ 的选取还要保证不同决策需要达到的控制目的。更大的问题在于: 对应于不同的起始温度、不同的设定温度、不同的外部环境, $M_{\text{冲}}$ 和 $M_{\text{误}}$ 所对应的值都不同。这样, 此算法只能适用于某一固定设定温度的控制; 在控制中, $M_{\text{冲}}$ 和 $M_{\text{误}}$ 保持不变。

(3) 三个决策的判断进入条件不易控制。决策一的目的在于使过冲量小, 决策一进入决策二的条件是温度变化的趋势较小, 驱动值范围缩小。决策二的目的是通过黄金分割法迭代寻优, 使转入决策三时的驱动值趋于自衡驱动。决策三的目的是在根据控制动态变化, 使驱动值在自衡驱动值的附近波动, 控制进入稳态。算法的决策转移条件, 往往不能使各个决策达到所预期的目的。因此, 此算法适用于被控对象热容积较大、能较好保持自衡态的被控对象的系统控制中。

3. 仿人智能控制法

(1) 仿人智能控制的基本思想

经典控制中的常规 PID 控制存在的主要缺点是难以解决稳定性和准确性之间的矛盾, 原因在于这种控制方式以不变的模式来处理变化多端的动态过程。

图 7.21 所示为典型的闭环控制系统在设定值扰动下的阶跃响应曲线。

相应曲线可划分为几个不同的阶段:

① OA 段: 这一段为系统在控制信号作用下, 由静态到动态再向稳态转变的关键阶段。由于系统具有惯性, 决定了这一段曲线只能呈倾斜方向上升。

为了获得好的控制特性, 在 OA 段应该采取变增益控制; 为了使系统上升的既快又不至于超调过大, 当系统输出上升接近稳态值而又相隔 m 时, 比例控制作用要降低, 使得系统借助于惯性继续上升, 既有利于减小超调而又不至于影响上升时间。

② AB 段: 系统输出值已经超过了稳态值, 向误差增大的方向变化, B 点误差到达最大值(负)。

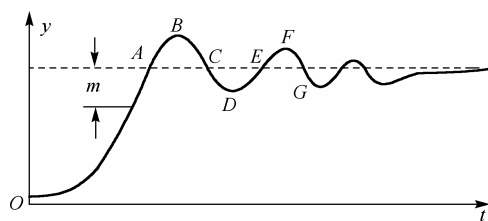


图 7.21 闭环控制系统在设定值扰动下的阶跃响应

在 *AB* 段, 控制作用应该尽力压低超调量, 除了采用比例控制外, 应加积分控制作用, 以便通过对误差积分加强控制作用, 使系统尽快回到稳态值。

③ *BC* 段: 在此段误差开始减小, 系统在控制作用下已呈现向稳态变化的趋势。这时若再继续施加积分控制作用, 势必造成控制作用太强, 而出现系统回调。

④ *CD* 段: 系统输出减小, 误差向相反的方向变化, 并达到正的最大值。此种情况应采用 *PI* 控制。

⑤ *DE* 段: 系统出现误差减小趋势, 控制作用不易太强, 否则会出现再次超调, 显然不应施加积分控制作用。

后面各段情况类同, 不再赘述。

由上述响应特性的分析可以看出, 控制系统的动态过程是不断变化的, 为了获得良好的控制性能, 控制器必须根据控制系统的动态特征, 不断地改变或调整控制决策, 以便使控制器本身的控制规律适应于控制系统的需要。

(2) 仿人智能控制行为的特征变量

为了有效地模拟人的智能控制行为, 并应用计算机实现智能控制, 必须通过一些变量来描述控制系统的动态特征, 表征其动态行为。

通常模糊控制中选用误差 *e* 和误差变化 Δe 作为输入变量, 模糊控制器的输出 *u* 可表示为

$$u = f(e, \Delta e)$$

(7.21)

误差 *e* 和误差变化 Δe 能较准确地反映出系统的动态特性。由这两个基本的模糊控制变量出发, 可引出其他的特征变量, 以便从动态过程中获取更多的特征信息, 并利用这些信息更好地设计仿人智能控制器。这些引出的特征变量中, 比较常见的是 $e \cdot \Delta e$ 。

$e \cdot \Delta e$ 可以方便地描述系统动态过程误差变化的趋势。设控制目标值为 *goal*, 当前测量值为 *T*, 则

$$e = \text{goal} - T, \quad e_{n-1} = \text{goal} - T_{n-1}, \quad e_n = \text{goal} - T_n$$
$$\Delta e_n = e_n - e_{n-1}$$

对应于图7.21所示的动态系统响应曲线的不同阶段, 特征变量的取值符号由表7.1给出。

表 7.1 特征变量符号变化

	<i>OA</i> 段	<i>AB</i> 段	<i>BC</i> 段	<i>CD</i> 段	<i>DE</i> 段
e_n	>0	<0	<0	>0	>0
Δe_n	<0	<0	>0	>0	<0
$e \cdot \Delta e$	<0	>0	<0	>0	<0

只要区分开 *OA* 段与其他 $e_n > 0$ 且 $e \cdot \Delta e < 0$ 的区段(如 *DE* 段), 就可完全由 e_n 和 $e \cdot \Delta e$ 的符号确定不同的响应阶段, 制定出不同的控制策略。实际上这个问题很易解决: 设置一个标志位, 在第一次达到控制目标值时置位, 就可以将不同的响应阶段完全区分开。

(3) 仿人智能控制的分段控制与参数整定

仿人智能控制的参数整定的过程是: 根据特征变量的动态变化, 分段整定参数。实际上, 仿人智能控制的基本方法就是在不同的控制阶段灵活地调整比例、积分和微分控制量, 因此仿人智能控制的参数整定也就是 *PID* 参数整定的过程, 但因为是分段控制, 所以参数的确定并没有 *PID* 参数整定那样严格, 也不需要全局性的考虑, 在操作上比较灵活简单。

常规的数字 *PID* 控制有位置式和增量式两种算法, 位置式较直观, 所以采用此种算法。

PID 控制器是一种线性的控制器, 其控制算法的模拟表达式为

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(t) dt + T_D \frac{de(t)}{dt} \right] \quad (7.22)$$

式中, $u(t)$ 为调节器的输出信号; $e(t)$ 为调节器的偏差信号; K_p 为调节器的比例系数; T_i 为调节器的积分时间常数; T_D 为调节器的微分时间常数。

由于 DDC 系统是一种时间离散控制系统, 因此, 为了方便计算机控制, 需要将模拟式进行离散化处理, 位置式方法就是离散化表示方法的一种, 其推导过程如下:

$$\int_0^n e(t) dt = \sum_{i=0}^n e(i) \Delta t = T \sum_{i=0}^n e(i) \quad (7.23)$$

$$\frac{de(t)}{dt} \approx \frac{e(k) - e(k-1)}{\Delta t} = \frac{e(k) - e(k-1)}{T} \quad (7.24)$$

由式(7.22)、式(7.23)和式(7.24), 可得数字 PID 位置型控制算法为

$$u(k) = K_p \left\{ e(k) + \frac{T}{T_i} \sum_{i=0}^n e(i) + \frac{T_D}{T} [e(k) - e(k-1)] \right\} \quad (7.25)$$

式中, k 为采样序号, $k=0, 1, 2, \dots$; T 为采样周期; $u(k)$ 为第 k 次采样时计算机的输出; $e(k)$ 为第 k 次采样时的偏差值; $e(k-1)$ 为第 $k-1$ 次采样时的偏差值。

所以, 第 k 次采样 PID 的输出为

$$\begin{aligned} u(k) &= u_p(k) + u_i(k) + u_d(k) \\ u_p(k) &= K_p e(k) \\ u_i(k) &= K_i \sum_{j=0}^k e(j) = K_i e(k) + K_i \sum_{j=0}^{k-1} e(j) = K_i e(k) + u_i(k-1) \\ u_d(k) &= K_D [e(k) - e(k-1)] \end{aligned} \quad (7.26)$$

式中, $K_i = \frac{T}{T_i} K_p$ 为积分系数; $K_D = \frac{T_D}{T} K_p$ 为微分系数。

设计中采用较简单的凑试法进行 PID 参数的整定时, 应综合考虑被控对象的惯性和温度变化的速率, 合适地选取控制周期, 以使采集的数据值能够反映出温度变化的趋势。

增大比例系数 K_p 将加快系统的响应速度, 在有静差的情况下有利于减小静差。但过大的比例系数会使系统有较大的超调量, 并产生振荡, 使系统趋于不稳定。 K_p 过小又会使系统的过渡时间增大。

增大积分时间 T_i , 有利于减小超调量, 减小振荡, 使系统更加稳定, 但系统静差的消除将随之变缓。

增大微分时间 T_D , 有利于加快系统响应速度, 使超调量减小, 稳定性增加, 但系统对扰动的抑制能力会衰减, 对扰动有较敏感响应。

在凑试时, 可参考以上参数对控制过程的影响趋势, 采用对参数实行先积分, 后比例, 再微分的整定步骤。

在过程控制中, 大多数被控对象都具有低通特性。因此, 有了被测量的采样周期经验数据, 比如被测参数为流量时采样周期 T 的经验数据为 $1 \sim 5$ s, 压力为 $3 \sim 10$ s, 液位为 $6 \sim 8$ s, 温度为 $10 \sim 20$ s, 成分为 $15 \sim 20$ s。根据本题的设计要求可确定采样周期为 10 s, 相应的控制策略如下:

- ① OA 段: $(e > 0) \&\& (e \cdot \Delta e < 0) \&\& (P = 0) \&\& (e \geq M)$ 时, 全开;
 $(e > 0) \&\& (e \cdot \Delta e < 0) \&\& (P = 0) \&\& (e < M)$ 时, 采用 PD 控制。
- ② AB 段: $(e \leq 0) \&\& (e \cdot \Delta e \leq 0)$ 时, 全关, $P = 1$ 。
- ③ BC 段: $(e \leq 0) \&\& (e \cdot \Delta e \geq 0) \&\& (|e| < K)$ 时, 采用 PD 控制。
- ④ CD 段: $(e > 0) \&\& (e \cdot \Delta e > 0)$ 时, 采用 PD 控制。
- ⑤ DE 段: $(e > 0) \&\& (e \cdot \Delta e < 0) \&\& (P = 1)$ 时, 全关。

由于被控对象在不同温度下有不同的热特性, 所以在控制中除采用在过程中分段处理的方法, 还应采用温度分段处理的方法; 对应于不同的过程时刻和不同的温度区间段, 采用不同的控制参数, 以期取得更好的控制效果。

4. 控制算法分析

根据前面讨论过的控制系统稳、准、快的品质要求, 一个理想的控制系统应同时满足较短的过渡时间、较小的超调量和较小的静差等要求。然而, 在实际情况下很难实现三种要求的同时满足。在对各种算法进行具体实现时, 根据所得数据和响应图进行对比分析, 有如下结论:

- (1) 用基于最优化控制理论的智能控制法较好地实现了温度控制, 其收敛速度较快, 即过渡时间较短, 但是温控过程不可逆。
- (2) 对比大林算法实现的温度控制图与仿人智能控制法实现的温度控制图, 可以很直观地看出大林算法与仿人智能控制法各自的特点:

- 从过渡时间的长短来看, 由于仿人智能法在开始加热的一段时间内以全开功率加热, 所以被控对象的温度上升速度, 明显快于大林算法, 即过渡时间较短。
- 从超调量的大小来看, 由于大林算法在控制过程中控制量变化较平稳, 所以功率输出均匀, 使其控制结果基本没有超调量; 而仿人智能法控制中较快的上升速度容易导致超调。
- 仿人智能法的微分量占较大比重, 所以对环境的变化比大林算法敏感, 易受环境影响, 而大林算法主要为积分叠加的过程, 控制均衡平稳, 抗干扰能力强。
- 仿人智能法的控制较滞后, 有变化后才有相应的控制给出, 所以控制过程中容易产生波动, 进入稳态过程较缓慢, 且不易保持稳态; 而大林算法的超前预测能力强, 易于进入并保持稳态。
- 仿人智能控制法的参数整定实际上是一个根据控制的情况不断修正参数的过程, 比大林算法烦琐的参数整定更为直观。

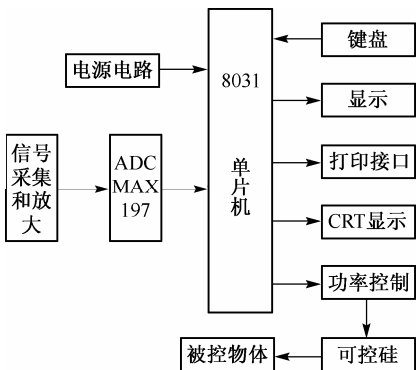


图 7.22 采用 ADC 采样方法系统总体框图

基于上面的算法比较和实验效果, 对于具有纯滞后的控制对象, 考虑到大林算法能够很好地解决时滞问题, 本节将以此算法的实现为例进行分析。

7.3.4 系统设计

1. 系统方框图

经过上面各种方案的比较和讨论, 在综合分析题目的要求及现有的条件, 可以使用 ADC 采样和 $V\text{-}f$ 变换的两种方法获得温度信号, 系统总体框图分别如图 7.22 和图 7.23 所示。

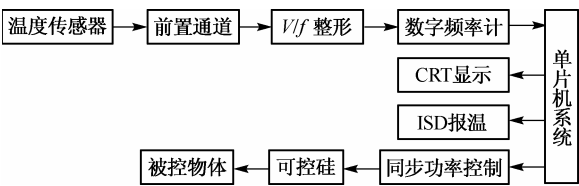


图 7.23 采用 V-f 变换方法系统总体框图

2. 系统软件流程和大林算法流程

系统软件整体流程图如图 7.24 所示，大林算法流程图如图 7.25 所示。

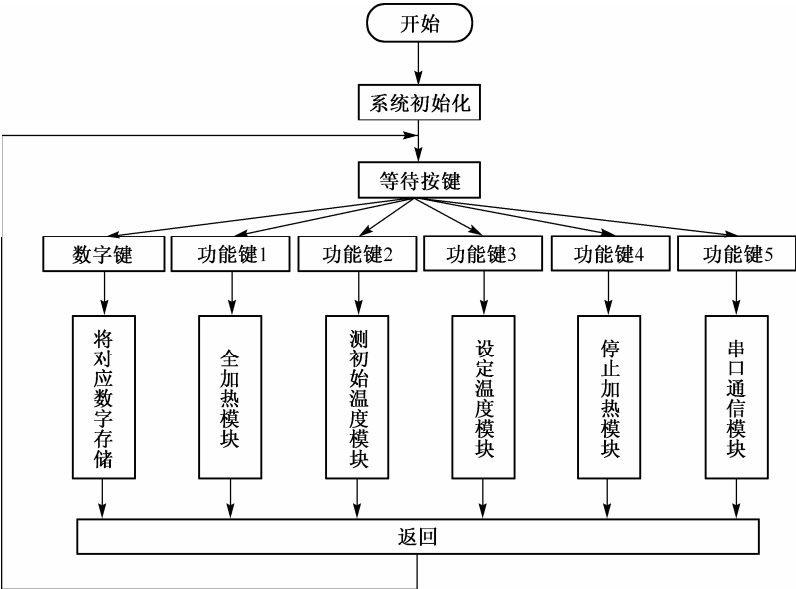


图 7.24 系统软件整体流程图

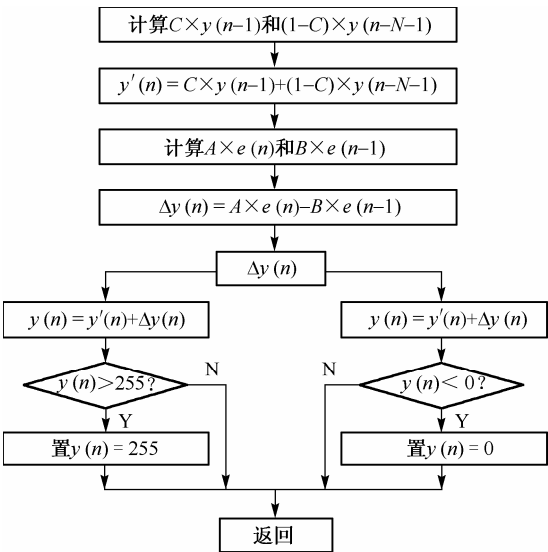


图 7.25 大林算法流程图

7.3.5 系统调整与性能测试

1. 参数调整

设定几个不同的控制温度，如 40℃, 50℃, 65℃, 70℃，在每个控制过程中调整大林算法的参数，从室温开始对被控对象加热控制，利用温度描述软件，绘出 4 个控制过程的曲线，并通过曲线可以观察到每个控制过程的超调量、过度时间和静差误差等，从而确定合适的大林算法的参数。

分段测试：基于上面的参数调整，采用适合的大林算法的参数，从室温开始对被控对象进行分段控制，即室温~40℃, 40~50℃, L，利用温度描述软件记录各段控制曲线，从而确定各段的超调量、过度时间和静差误差等，最终确定大林算法的参数值。

2. 电路调试及性能测试

1) 系统电路调试

采用先分别调试各单元模块，调通后再进行整机调试的方法，以提高调试效率。调试过程参考如下。

- (1) 单片机最小系统模块调试。
- (2) 温度信号采集模块调试。
- (3) 功率驱动器模块调试。
- (4) 整机调试。

2) 对象特性测试

(1) 铁块温度传输特性测试

测试方法：从室温加热到预定温度，分别测试从室温到 40℃, 50℃, 60℃, 70℃, 80℃, 90℃时被控对象的温度上升速率，记录曲线的形态及对应时间等数据。加热器满功率开通。

(2) 热惯性测试

测试方法：从室温将被控对象分别加热到 40℃, 50℃, 60℃, 70℃, 80℃, 90℃，记录不同终点温度的过冲量数值的大小。加热器满功率开通。

3) ADC 温度采样系统测试

为作比较，在采用 ADC 温度采样实现系统设计时，分别采用大林算法和基于最优化控制理论的智能控制算法完成系统的响应测试。

(1) 采用大林算法控制铁块温度(从室温开始加热)，设定温度为 60℃，测试结果如图 7.26 所示。其他设定温度情形有类似曲线。

(2) 采用智能控制法控制铁块温度(从室温开始加热)，由相关算法实现可得到温度图，限于篇幅具体的结果没有给出。

分析测试结果可得到如下结论：当分别采用大林算法和智能控制法两种算法时，温度上升曲线不尽相同。大林算法能够有效地控制温度，控制精度很高，控温的超调量很小，能很好地实现题目要求，但是到达设定温度所需的时间比较长。智能控制法的升温时间很短，但是超调量比较大，而且温度稳定时的静差也比较大，算法中的三个决策不可逆是智能控制法的主要问题。总的来说，大林算法和智能控制法都能实现对温度的控制，但大林算法比智能控制法有更好的控制效果。

4) V-f 温度采样系统测试

在采用 V-f 温度采样实现系统设计时，分别采用大林算法和仿人智能控制算法完成系统的响应测试，并分析、比较测试结果。

(1) 仿人智能控制算法的控制过程可由 CRT 显示的温度-时间曲线直观地反映出来，包括仿人智

能控制算法在相对较稳定环境中的测试曲线和其在环境温度变化较为显著的条件下的测试曲线。实验中使用电扇对铁块吹风。

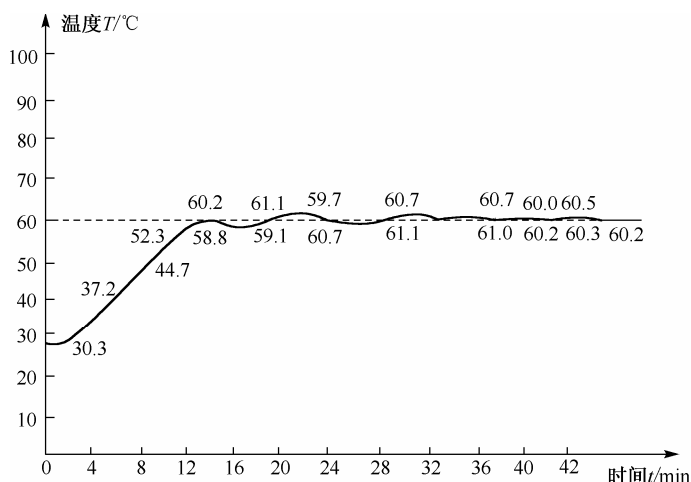


图 7.26 铁块温度系统的大林算法控制阶跃响应

(2) 大林算法的控制过程也可由 CRT 显示的温度-时间曲线直观地反映出来, 主要是测试大林算法在环境温度变化较为显著的条件下的阶跃响应曲线。

5) 测试误差分析

(1) 从热平衡观点看, 对于一个被调量为温度的对象, 流入和流出的热流量差额累积起来可以存储在对象中, 表现为对象平均温度水平的升高(如果流入量大于流出量), 此时, 对象的储蓄容积就是它的热容量。显然, 不同介质对象的热容量是不同的, 同为温度控制, 对它们的控制难度差别很大。在本题中加热电热丝带入的热量是流入量, 室温下空气热交换及对象热辐射带走的热量是流出量。在稳态下, 流入量与流出量保持平衡。所以, 无论是流入量或流出量, 作为引起被调量变化的原因, 都应看成是被控对象的输入量。而在本题中温度传感器和计量温度计的检测位置并不在同一个点上, 而热源又位于铁块的一端, 铁块的温度是不均匀的, 呈现一个温度梯度场, 因此, 计量温度计与被测温度的热平衡测量受到很多因素的影响。要提高精度, 必须采取相应的措施。

(2) 作为计量温度的水银温度计的时间常数比 LM35 温度传感器的时间常数要大得多, 除非采用标准电子温度计作为计量温度, 它们之间的时间常数大致相等, 否则为了准确地测试, 还应精确测定水银温度计的时间常数与作为变送仪表的 LM35 温度传感器的时间常数, 以计量温度计为标准修订参数, 减小测试误差, 并做出误差分析。

7.4 本章小结

本章介绍控制系统中的控制策略和算法, 着重研究了非线性、纯时滞控制系统中的各种控制算法。通过对一个实验室环境下的非线性、纯时滞控制系统的设计实例介绍, 说明了控制系统的硬件、算法和软件在设计时应当注意的问题, 这是读者学习本章时所应重点关注的内容。通过本章的学习, 希望读者能够将已有的信号与系统的知识应用到具体的设计实践中, 进而分析过程控制系统的特点。最后, 读者要养成理论分析和实验结果互动思考的好习惯, 通过实验结果检验自己的理论分析结果的正确性, 同时通过理论分析指导实验过程的科学设计。

第8章 简单测试仪器的设计与实现

8.1 引言

所谓的测试仪器大多是指一些具有测量功能的电子设备。在科研实践活动中客观、定量的观察必须借助于测量仪器,测量能够定量地获取一个量,在测量过程中,测量仪器极大地拓展了人们的感官所能感受的客观世界。因此,在复杂电子系统的设计实践中,简单测试仪器的设计与实现具有重要的意义。

本章以常见的几种测试仪器的设计为例,介绍它们的设计和实现方法,希望读者能以测量学的视角来看待和思考设计过程中的问题,对测量方法、测量误差的基本理论能有更清晰的认识。常见的测试仪器包括电气参数测量仪、元器件参数测量仪、时域测量仪、频域测量仪和数据域测量仪等。时域测量仪如数字频率计、数字相位测量仪等,它们的设计在前面的章节已经详细介绍过。

本章将要介绍的数字电容测量仪属于元器件参数测量仪的一种,数字电容测量仪的原理主要是将待测电容 C_x 转换为可测量物理量,如电压、频率、脉宽等,通过测量上述物理量的数值来间接测量电容的容值。

数字式工频多用表属于电气参数测量仪,它能够测量工频交流电的电压有效值、电流有效值、有功功率、无功功率、功率因素等参量。电流是七个基本国际单位(SI)之一,而在集总参数电路中,考虑到便利性,测量的主要参数是电压。因为在标准电阻的两端,只要测出电压值,即可求出电流和功率。因此,数字式工频多用表的设计主要是电压参数的测量和参量的转化。电压测量成为最基本的电气测量参数,电子设备的许多工作特性均可视为电压的派生量,如调幅度、波形的非线性失真参数等。在非电参量测量中,大多数物理量的传感器也都是以电压作为输出的。因此,电压测量是其他许多电参量、非电参量测量的基础。

示波器是一种基本的、应用广泛的时域测量仪,通过它能观察信号波形,测量信号的幅度、频率、周期等基本参量,测量脉冲的脉宽、占空比、上升/下降时间等参数,还能测量两个信号的时间和相位关系。由于数字示波器采用高速 A/D 转换器对待测信号取样,因此通过数字示波器不仅可以观测高频重复性的周期信号,还可以观测瞬间的单次脉冲,并且具有存储功能。因此,高速数据采集、存储和回放系统的设计成为数字示波器系统设计的难点。

逻辑分析仪属于数据域测量仪,它是数字电路调试和信号分析中不可缺少的工具。简易逻辑分析仪的设计实质上就是用双踪信号示波器作为逻辑分析结果的显示设备,用单片机控制逻辑信号采集和逻辑分析仪的各项功能操作,用 FPGA 处理逻辑信号并控制逻辑分析结果波形的点阵扫描,达到一般逻辑分析仪应有的功能和指标。

本章将以数字电容测量仪、数字式工频多用表、数字示波器和简易逻辑分析仪的设计为例来介绍简单测量仪器的设计与实现,旨在通过这一系列训练,使读者在掌握基本理论知识和设计方法的基础上,充分领悟复杂电子系统的设计思想,并学会使用合理的设计方法来完成简单电子仪器仪表的设计。

8.2 数字电容测量仪

1. 设计任务

设计并制作一个基于 MCS-51/52 系统的智能化电容测量仪。

2. 设计基本要求

测量范围：1 pF~10² μF；

准确度：1.0×10⁻³；

分辨率：1.0×10⁻³ pF；

线性度：0.99。

测量结果以 6 位 LED 显示，LED 显示单元最高位显示 C 以表示电容器测量，显示单位为 pF，以科学计数法显示测量结果。

8.2.1 测量原理分析与论证

电容的测量方法很多，主要可以分为下面两类。第一类方法是把电容量通过电路转换成电压量，然后把电压量经模数转换器转换成数字量并换算成电容量进行显示。第二类方法是将电容量转化为与时间或者频率有关的量，通过测量时间或频率求出待测电容的电容量。常见的电容测量方法有高频交流测量法、相位测量法和 C-T 测量法。

1. 高频交流测量法

高频交流测量法是基于传统测量技术的一种方法，其基本原理是：利用线性闭环运算放大器组成的高频测量电路，对被测电容 C_x 进行测量。

高频交流测量法测电容的原理图如图 8.1 所示，C_x 为被测电容，G_x 为 C_x 的等效并联电导，R_f 为反相反馈电阻，由于采用高速宽带低漂移放大器，同相端可直接接地。当给测量电路输入高频正弦波激励信号时，测量电路的输出电压和输入电压之间的复数关系为

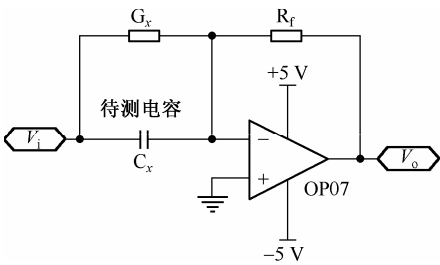


图 8.1 高频交流测量法测电容的原理图

$$V_o = -(j\omega C_x R_f + G_x R_f) V_i \tag{8.1}$$

式中， ω 为激励信号的角频率。测量电路对被测电容 C_x 的灵敏度与信号源的角频率成正比；而等效并联电导与 ω 无关，即测量电路受等效并联电导的影响极小。所以可以认为高频测量提高了对被测电容的灵敏度，一般可采用大于或等于 500 kHz 的激励信源。测量时，分别测出 V_o 与 V_i，联解求得 C_x 和 G_x。高频交流电容测量电路示意图如图 8.2 所示。

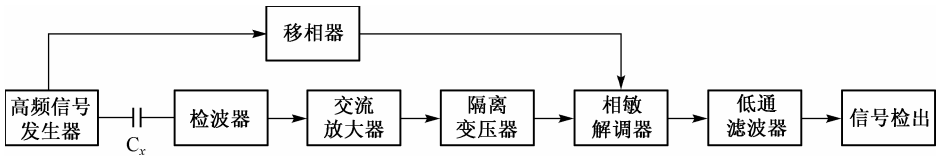


图 8.2 高频交流电容测量电路示意图

2. 相位测量法

所谓相位测量法，是指利用被测电容 C_x 对交流信号的相移作用，通过 CPU 对相位差进行测量来获得被测电容的容量值。其原理图如图 8.3 所示。

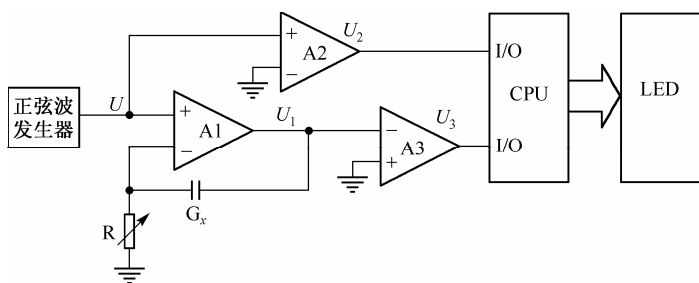


图 8.3 相位测量法测电容的原理图

设正弦激励信号 $u = u_m \sin \omega t$ 输入到 A_1 的同相端, A_1 为同相放大器, 反馈回路由 R 和 C_x 组成, A_1 的输出 U_1 为

$$U_1 = \frac{Z_R + Z_C}{Z_R} \cdot U \quad (8.2)$$

其中, Z_R 为纯电阻; $Z_C = 1/j\omega C$ 为纯容抗。由图 8.4 可见, 相位角 $-\pi/2$ 在虚轴负方向上, 所以 A_1 的放大系数 $(Z_R + Z_C)/Z_R$ 的相角是 Z_R 和 Z_C 矢量合成的相角 φ 。若定义 U 的初始相位为零, 则 U_1 滞后于 U 相位 φ 。 A_2 和 A_3 为过零比较器, A_2 为同相比较, 输出为 U_2 , A_3 为反相比较, 输出为 U_3 。测量时 CPU 只对正跳变检测, 即只检测 U 和 U_1 的零相位, 所以 U_2 和 U_3 正跳变的相差为 $\varphi + \pi$, 这有利用提高测量精度。

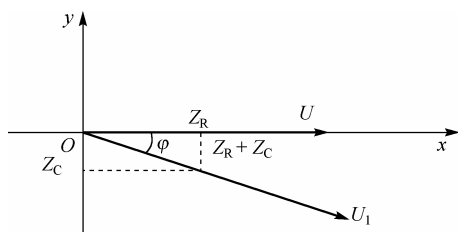
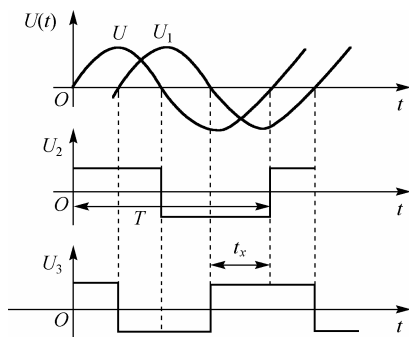


图 8.4 矢量参数图

图 8.5 比较器 A_2 和 A_3 的输出波形

比较器 A_2 和 A_3 的输出波形如图 8.5 所示, 可看出

- (1) CPU 测出 U_2 两次上跳的时间间隔为 T , 即为正弦信号的周期。
- (2) 测量 U_2 上跳到 U_3 上跳的间隔 t_x , 则相位差为

$$\varphi = \frac{t_x}{T} \times 2\pi - \pi \quad (8.3)$$

- (3) 由图 8.4 可知

$$\tan \varphi = \frac{|Z_C|}{|Z_R|} = \frac{1}{\omega R C_x} \quad (8.4)$$

所以

$$C_x = \frac{1}{\omega R \tan \varphi} = \frac{T}{2\pi R \tan(2\pi \times t_x / T - \pi)} \quad (8.5)$$

3. C-T变换法

C-T变换法的原理是将被测电容 C_x 的测量转换为时间参数的测量,而对时间参数的测量已在前面的章节中进行了介绍,因而这一方法可直接利用前面的设计结果。

C-T变换法测电容的原理图如图8.6所示。其中 A_1 为积分器, A_2 为比较器, E_1 和 E_2 为电容测量充放电定值电压源, C_x 为被测电容, S 为电子模拟开关, F 为反相器。假定被测电容 C_x 两端的初始电压等于零,当模拟开关 S 断开时, E_2 对被测电容积分,积分过程为定值积分,将一直积分到使 A_1 输出 $U_{A1} = E_1$ 为止。设此定值积分阶段积分时间为 T_1 , 则有

$$U_{A1} = -\frac{1}{R_2 C_x} \int_0^{T_1} E_2 dt = -\frac{E_2 T_1}{R_2 C_x} = E_1$$

$$T_1 = -E_1 R_2 C_x / E_2 \quad (8.6)$$

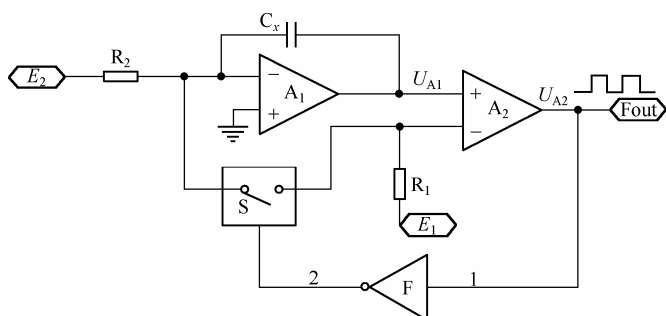


图 8.6 C-T变换法测电容原理图

T_1 阶段结束后,比较器 A_2 翻转,其输出跳变为低电平,经反相器 F 后控制模拟开关 S 闭合,使 E_1 接入积分回路。由于 $|E_1/R_1| > |E_2/R_2|$,故通过 C_x 的积分电流极性将发生变化,使得积分器反向积分。此时输出电压 U_{A1} 将从 E_1 值经过一段时间后被积分至 $0V$ 。设反向积分时间为 T_2 , 则有

$$E_1 = -\frac{1}{C_x} \int_0^{T_2} \left(\frac{E_2}{R_2} + \frac{E_1}{R_1} \right) dt$$

$$T_2 = \frac{E_1 R_1 R_2 C_x}{R_1 E_2 + R_2 E_1} \quad (8.7)$$

设比较器输出电压信号的周期为 T , 则有

$$T = T_1 + T_2 = -\frac{E_1 R_2 C_x}{E_2} + \frac{E_1 R_1 R_2 C_x}{R_1 E_2 + R_2 E_1}$$

$$C_x = -\frac{E_2 (R_1 E_2 + R_2 E_1) T}{E_1^2 R_2^2} \quad (8.8)$$

可求得

因此测量电路的灵敏度为

$$S = \frac{dT}{dC_x} = -\frac{E_1 R_2}{E_2} + \frac{E_1 R_1 R_2}{R_1 E_2 + R_2 E_1} \quad (8.9)$$

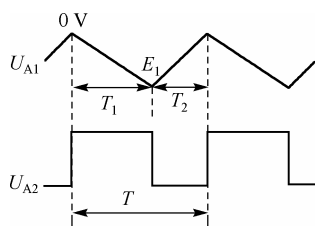


图 8.7 积分器与比较器的输出电压波形

上述过程中的积分器与比较器的输出电压波形如图8.7所示。

4. 方案比较分析

通过以上测量方法的介绍可看出电容数字测量仪可由多种方法设计。除了上面介绍的电路, 还有其他的設計方法, 例如可采用 555 集成定时器构成单稳态触发器电路, 单稳态触发器输出电压的脉宽为 $T_w = RC \ln 3 \approx 1.1RC$, 电路产生的脉宽可以从几个微秒到数分钟, 当 R 固定时, 改变电容 C , 则可以改变输出脉宽 T_w , 由 T_w 的宽度即可求出电容的大小。由于单稳态触发器的输出脉宽 T_w 与电容 C 成正比, 利用已掌握的数字化频率测量的知识和技术, 可以快速实现电容数字测量仪的设计。这一方法也是 C - T 变换法, 只是采样方式和上面介绍的双向积分采样的方法有所不同。下面就已讨论的设计方法做出分析和比较。

1) 高频交流测量法

该方法的优点是: 测量灵敏度高; 有较高的准确度; 可适用于微小电容器的测量。一般地, 电容的测量其准确度的要求不可能很高, 因此该测量方法在微小电容测量时具有很大的优势, 这对以微小电容 (1 pF 以下) 为基础的传感器技术, 如计算机断层扫描技术的应用而言有着重要的意义。

该方法的不足是: 高频交流测量电路相对于其他方法的测量电路, 其电路结构较为复杂, 电路需要高稳定度的高频信号发生器、高性能高频放大器和相敏解调器及滤波环路等。

2) 相位测量法

该方法的优点是: 利用 CPU 的运算能力对 T 和 t_k 进行测量, 可大幅度降低系统测量时信源的频率漂移所引起的误差; 由于是对过零信号进行检测, 可消除振幅漂移引起的误差; 测量误差主要来源于对 T 和 t_k 的测量时的计数误差, 但由于被测量信源的频率一般取几百赫兹, 而 CPU 的时钟频率 ($6 \sim 12 \text{ MHz}$), 因此 T 和 t_k 对应的计数值较大, 测量误差可降到 0.1% 以下。若利用前面章节的设计结果, 通过宽位高速计数单元电路, 提高测量精度有较大的余地。

该方法的不足是: 为提高精度, 相差 ϕ 要控制在一定范围, 这时需改变 R 值来进行量程转换。

3) C - T 变换法

该方法的优点是: C - T 变换法利用比较器输出电压信号的周期 T 和被测电容 C_x 成正比的对应关系, 基于双斜率积分原理, 实现电容器的测量。其电路结构简单, 通常的电容测量电路一般都要有测量激励信号源等电路, C - T 变换法不需要信号源, 相比而言 C - T 变换法的硬件开销较小。更重要的是, C - T 变换法的比较器输出电压信号周期的变化量 dT 和被测电容变化量 dC_x 之比等于常数, 通过测量输出信号周期变化量的方法间接测量电容器容值, 具有较高的灵敏度和线性度。而且积分电路具有时域平均作用, 电路对于共模干扰信号有很强的抑制能力, 使测量的稳定性有较大的提高。

该方法的不足是: 当被测电容 C_x 有较小容量时, 由于积分路径较小, 测量分辨率降低。当被测电容 C_x 有较大容量时, 测量电路跟随特性变差, 测量灵敏度降低。

由于前面的章节对时间参数测量做过详细的讨论, 考虑到设计的效率, 在已获较高精度的数字化频率测量工具的情况下, 采用 C - T 变换法应是一种较好的选择。本节就以 C - T 变换法为例讨论设计过程。

8.2.2 系统参数的计算

根据指标要求确定系统参数。由前面关于 C - T 测量法的分析可知, 比较器输出电压信号的周期及待测电容的电容量分别为

$$\begin{cases} T = T_1 + T_2 = -\frac{E_1 R_2 C_x}{E_2} + \frac{E_1 R_1 R_2 C_x}{R_1 E_2 + R_2 E_1} \\ C_x = -\frac{E_2 (R_1 R_2 + R_2 E_1) T}{E_1^2 R_2^2} = \frac{K}{F_x} \end{cases} \quad (8.10)$$

当占空比为 1:1 时, $T_1 = T_2$, 即满足 $E_2 = -E_1$, $R_2 = 2R_1$ 。由此可得出如下结论: C_x 与 F_x 在理论上成反比关系, 比例系数 K 由系统参量所决定。

指标要求 C_x 的范围为 $1 \text{ pF} \sim 10^2 \mu\text{F}$, $C_{x\text{max}}/C_{x\text{min}} = 10^8$, 如果频率测量工具可以对 $0.1 \text{ Hz} \sim 16 \text{ MHz}$ 之间的频率进行高精度测量, 测量精度为 10^{-6} , 即 $f_{x\text{max}} = 16 \text{ MHz}$, $f_{x\text{min}} = 0.1 \text{ Hz}$, $f_{x\text{max}}/f_{x\text{min}} = 10^9$, 则设计完全满足测量要求。但值得注意的是, 由于模拟开关打开和关断需要一定的建立时间, 这一时间会严重制约小电容的充放电速度。快速充放电过程与相对较慢的模拟开关的建立时间之间的矛盾, 导致系统对小电容测量的误差很大。

下面给出关于上述结论的量化指标并确定系统工作频率范围。如果使用的模拟开关为 MAX332, MAX332 的闭合时间为 600 ns , 关断时间为 450 ns , 由此确定 MAX332 模拟开关正常工作下, 开关闭合和断开的的时间间隔不得少于 600 ns 。相对于本系统, 要求测量频率不得高于 1.6 MHz 。事实上, 实验测试结果表明, 当测量频率高于 600 kHz 时, 电容测量精度就受到了很大影响。此外, 当频率计测量 1 Hz 以下频率时, 需要较长的测量时间。综合考虑上述因素, 可估算系统测量时频率范围在几 Hz 至几百 kHz 比较合适。但在这样的频率范围内却又无法覆盖被测电容范围 $1 \text{ pF} \sim 10^2 \mu\text{F}$ 的指标要求。可以采取的解决办法是设置两路电容充放电通道(两路通道在测量不同电容时的时常数基本相等): 一路通道的电阻值较大, 测量较小电容 ($0.1 \mu\text{F}$ 以下) 时, 频率范围满足几 Hz 至几百 kHz 的要求; 另一路通道的电阻值较小, 测量较大电容 ($0.1 \mu\text{F}$ 以上) 时, 频率范围也满足几 Hz 至几百 kHz 的要求。这就较好地解决了电容充放电过程与模拟开关开闭动作速度冲突的问题, 参考电路如图 8.8 所示。

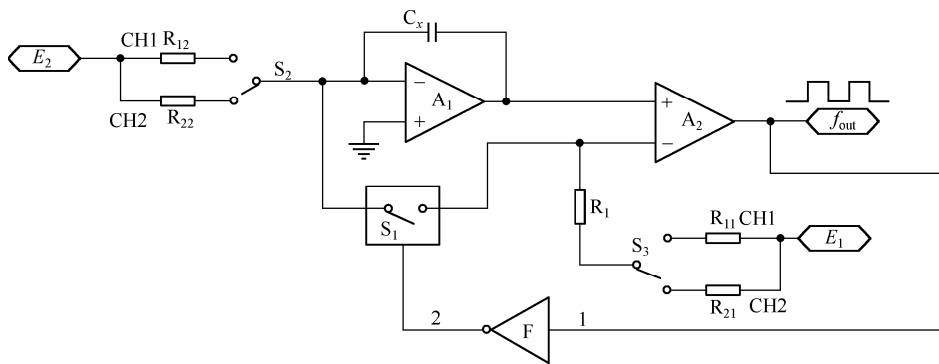


图 8.8 具有两路充放电通道的 C - T 测量改进电路图

图 8.8 中 R_{11} 和 R_{12} 构成一路电容充放电通道; R_{21} 和 R_{22} 构成另一路电容充放电通道。满足条件 $R_{11} \gg R_{21}$, $R_{12} \gg R_{22}$ (几十倍至几百倍)。当测量电容较小时, S_2 和 S_3 接通 CH1; 当测量电容较大时, S_2 和 S_3 接通 CH2。

8.2.3 系统硬件电路设计

1. 电压源模块

系统测量基准电压源 E_2 和 E_1 分别取 $+2.500 \text{ V}$ 和 -2.500 V , 由 MAX873 提供, 具体电路如图 8.9 和图 8.10 所示。

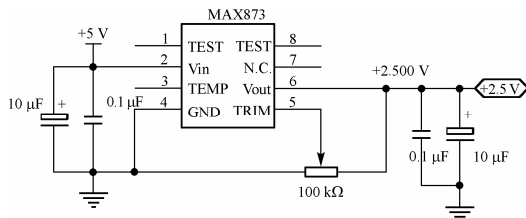


图 8.9 $+2.500 \text{ V}$ 电压源电路

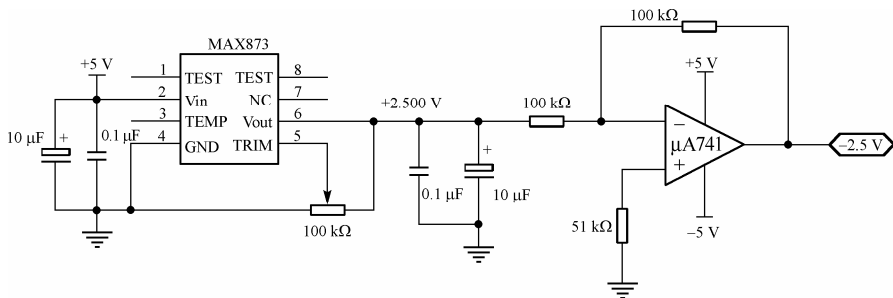


图 8.10 -2.500 V 电压源电路

2. 利用模拟开关切换充放电电路模块

如图8.11所示，CH1 和 CH2 分别由两路充放电电路组成。通路由 IN1 和 IN2 进行选择，当 PC4 为高电平时，R₂₁和 R₁₁接入通路；当 PC4 为低电平时，R₂₂和 R₁₂接入通路，完成通路间的切换。

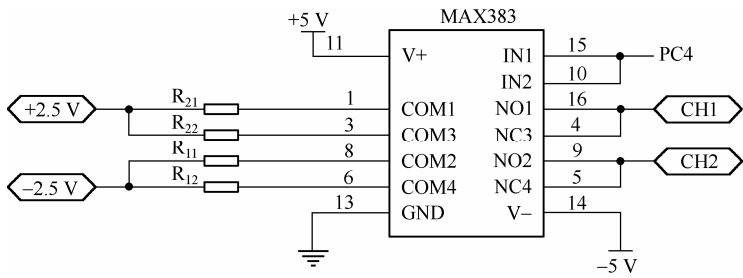


图 8.11 模拟开关切换充放电电路电路图

3. 频率测量模块

频率测量模块的设计在前面的章节已经详细讨论过，采用等精度测频的方式实现 C-T 变换的测量。

8.2.4 软件设计

系统设计应能实现两挡量程的自动切换，具体实现的思路是：程序初始化阶段，将模拟开关切换至小电容充放电通道，粗测 1 s 时间，如果测到的频率值小于 2 Hz，则说明被测电容的容值较大，这时自动将模拟开关切换至大电容充放电通道，完成量程的自动切换。

8.3 数字工频多用表

1. 设计任务

设计并制作一个能同时对一路工频交流电(频率波动范围为 50±1 Hz、有失真的正弦波)的电压有效值、电流有效值、有功功率、无功功率和功率因数进行测量的数字式多用表。

为便于设计与制作，设待测 0~500 V 的交流电压、0~50 A 的交流电流均已经过相应的变换器转换为 0~5 V 的交流电压，如图8.12所示。

2. 设计基本要求

(1) 测量功能及量程范围

- ① 交流电压：0~500 V；
- ② 交流电流：0~50 A；
- ③ 有功功率：0~25 kW；
- ④ 无功功率：0~25 kvar；
- ⑤ 功率因数(有功功率/视在功率)：0~1。

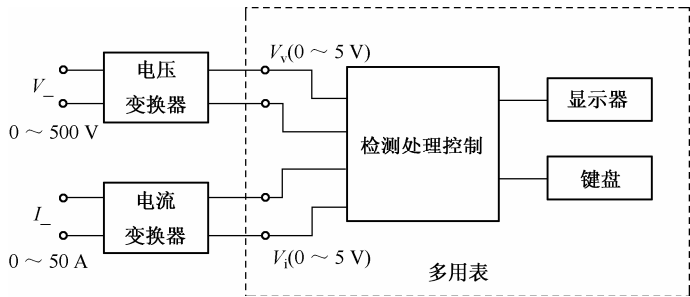


图 8.12 数字式多用表

(2) 准确度

- ① 显示为 5 位(0.000~4.999)，有过量程指示；
- ② 交流电压和交流电流： $\pm (0.8\% \text{ 读数} + 5 \text{ 个字})$ ，例如当被测电压为 300 V 时，读数误差应小于 $\pm (0.8\% \times 300 \text{ V} + 0.5 \text{ V}) = \pm 2.9 \text{ V}$ ；
- ③ 有功功率和无功功率： $\pm (1.5\% \text{ 读数} + 8 \text{ 个字})$ ；
- ④ 功率因数： ± 0.01 。

(3) 功能选择：用按键选择交流电压、交流电流、有功功率、无功功率和功率因数的测量与显示。

3. 设计提高部分要求

- (1) 用按键选择电压基波及总谐波的有效值测量与显示。
- (2) 具有量程自动转换功能，当变换器输出的电压值小于 0.5 V 时，能自动提高分辨率达 0.01 V。
- (3) 用按键控制实现交流电压、交流电流、有功功率和无功功率在测试过程中的最大值与最小值测量。
- (4) 其他(例如扩展功能，提高性能)。

8.3.1 系统设计方案分析和理论计算

1. 系统设计方案分析

设计要求对单相电的电参数进行测量，这些参数中电压和电流为基本量，其他参数是导出量。任务中，待测 0~500 V 的交流电压、0~50 A 的交流电流均已相应地转换为 0~5 V 的等效交流电压，其中等效电流信号是等效电压信号经过等幅移相网络得到的。基于此前提，本设计需要测得等效电压(电流)信号的真有效值、等效电压信号基波的真有效值和电压、电流两路信号的相位差，再按照工频信号有功功率、无功功率和功率因数等定义计算出各物理量值。因此，完成基本要求应主要解决以下两个关键问题：(1) 真有效值转换，其中包括等效电压和等效电流信号的真有效值的转换；(2) 电压信号、电流信号相位差的测量。另外，为完成提高部分要求中测量电压基波和总谐波的有效值，需测量等效电压信号基波的真有效值。由总谐波、基波和失真信号有效值之间的关系式，可以算出总谐波有效值。因此，此部分的关键问题是如何从失真信号中获得工频信号的基波。

相位差的测量方法在前面的章节已经详细地讨论过，可采用等精度的测量方法进行测量。50 Hz 基波信号的获得可以采用多阶的低通滤波器来实现，具体实现方法可以用分立器件来实现，也可以用专用滤波器来实现。

真有效值转换常见的方法有热电偶变换法、采样算法、模拟直接运算转换法和单片集成有效值转换器件法等。其中，采样算法和单片集成有效值转换器件法是最常用的方法。采样算法通过对周期信号进行快速采样获得多个离散值，再利用单片机的运算功能进行相关运算得到真有效值。采用采样算法的系统设计方案如图 8.13 所示。为保持采样间隔随信号频率的波动而变化，即把一个周期等时间间隔采样变为等相位采样，可采用锁相环电路，利用锁相环把信号进行 64 倍频，从而在需要采集信号的一个周期内产生 64 个脉冲，利用此脉冲信号作为单片机的外部中断信号，快速启动 A/D 转换器进行转换，实现数据采样。

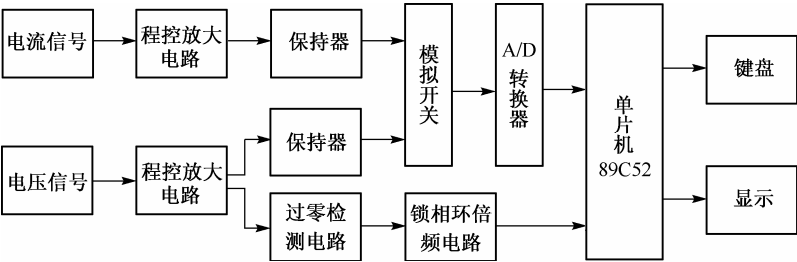


图 8.13 采用采样算法的系统设计方案

而单片集成有效值转换器件法是利用专用芯片实现交流信号的真有效值转换，输出直流信号，A/D 转换器可以直接对其进行采样。该方案的软件设计相对简单些，其系统设计方案如图 8.14 所示。本节以此方案为例进行讨论。

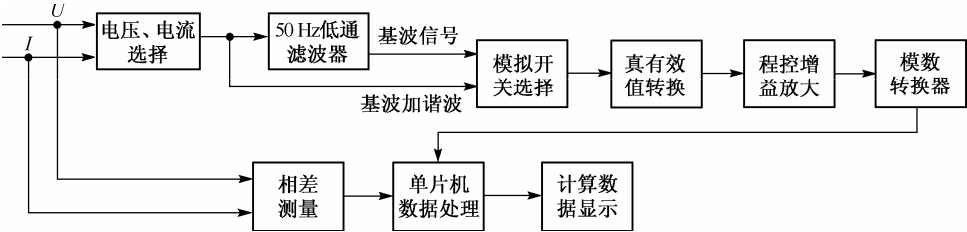


图 8.14 采用单片集成有效值转换器件法的系统设计方案

2. 理论计算

1) 功率及功率因数的计算

根据电工学原理的相关知识可知，当周期为 T ，电压信号 U 和电流信号 I 之间的相差为 Φ 时，交流电参量定义为

$$\text{电压有效值: } \bar{U} = \sqrt{\frac{1}{T} \int_0^T U^2(t) dt},$$
$$\text{功率因数: } \cos \Phi = P/S,$$
$$\text{无功功率: } P = \bar{U} \bar{I} \sin \Phi,$$

$$\text{电流有效值: } \bar{I} = \sqrt{\frac{1}{T} \int_0^T I^2(t) dt}$$
$$\text{有功功率: } P = \bar{U} \bar{I} \cos \Phi$$
$$\text{视在功率: } S = \bar{U} \bar{I}$$

因此，只要测得 U 和 I 的真有效值及两者的相差 Φ ，然后经过运算就可以方便地求得上述的各电参量值。

2) 电压基波有效值的计算

被测信号 $u(t)$ 是一个周期的时间函数, 除基波外, 还含有直流分量和各次谐波, 根据傅里叶级数理论, 可将其表示为

$$u(t) = \sum_{n=0}^{\infty} [b_n \cos(n\omega_1 t) + a_n \sin(n\omega_1 t)] \quad (8.11)$$

式中, $n=0, 1, 2, \dots$; a_n 和 b_n 分别为各次谐波的正弦项和余弦项的振幅; a_1 和 b_1 分别为基波的正弦和余弦分量的振幅; b_0 为直流分量。

由傅里叶级数理论, 可以求出 $u(t)$ 中的基波分量为

$$u_1 = b_1 \cos(\omega_1 t) + a_1 \sin(\omega_1 t)$$

将上式化简可以得到

$$\begin{aligned} a_1 &= \sqrt{2}\bar{U} \cos \alpha_1, & b_1 &= \sqrt{2}\bar{U} \sin \alpha_1 \\ \bar{U}_1^2 &= (a_1^2 + b_1^2)/2, & \tan \alpha_1 &= b_1/a_1 \end{aligned} \quad (8.12)$$

式中, $\bar{U}_1$ 为基波有效值; α_1 为 $t=0$ 时的基波分量的相角。谐波总有效值根据基波有效值可以算出为

$$\bar{U}_{\Sigma}^2 = \bar{U}^2 - \bar{U}_1^2 \quad (8.13)$$

式中, $\bar{U}_{\Sigma}$ 为谐波总有效值; $\bar{U}$ 为电压总有效值; $\bar{U}_1$ 为基波总有效值(假定不存在直流分量)。

8.3.2 各模块电路设计与分析

本系统主要由移相网络、程控增益放大电路、真有效值转换电路、50 Hz 低通滤波电路、A/D 采样电路和相差测量电路组成, 下面分别予以介绍。

1. 移相网络

在线性系统中, 当正弦交流电压施加在线性负载上时, 产生的电流也是一个与电压同频的正弦交流量。因此, 模拟电流信号可由模拟电压信号经过移相网络后得到。为此需设计等幅移相网络, 通过改变电位器的阻值大小可以将电压信号移相 ($-180^\circ \sim 0^\circ$) 后得到电流信号, 参考电路如图8.15所示。

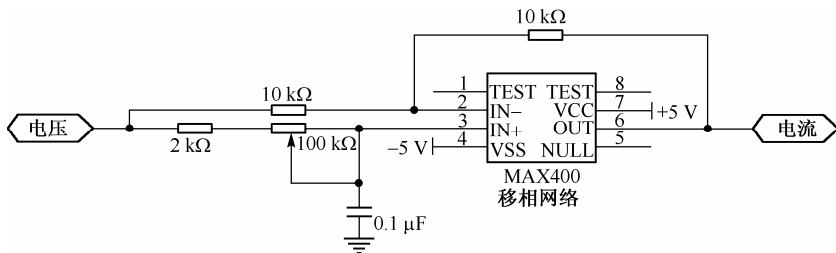


图 8.15 获得等效电压、电流信号的移相网络

2. 程控增益放大电路

为了保证系统测量精度, 考虑到输入信号幅值变化范围较大, 为满足对大小不同的电流信号和电压信号进行处理, 输入放大器电路应采用可编程放大形式。可采用精密电阻网络仪表放大器 MAX5426 和 MAX427 组成信号程控增益放大电路。该程控增益放大电路可以实现 1, 2, 4, 8 四个挡位的放大, 其特点是放大精度很高, 理论增益误差指标达到 0.025%。程控增益放大电路的设计框图如图8.16所示,

参考电路如图8.17所示。

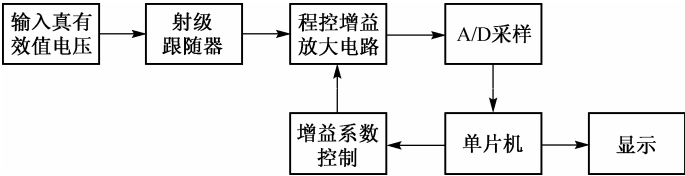


图 8.16 程控增益放大电路的设计框图

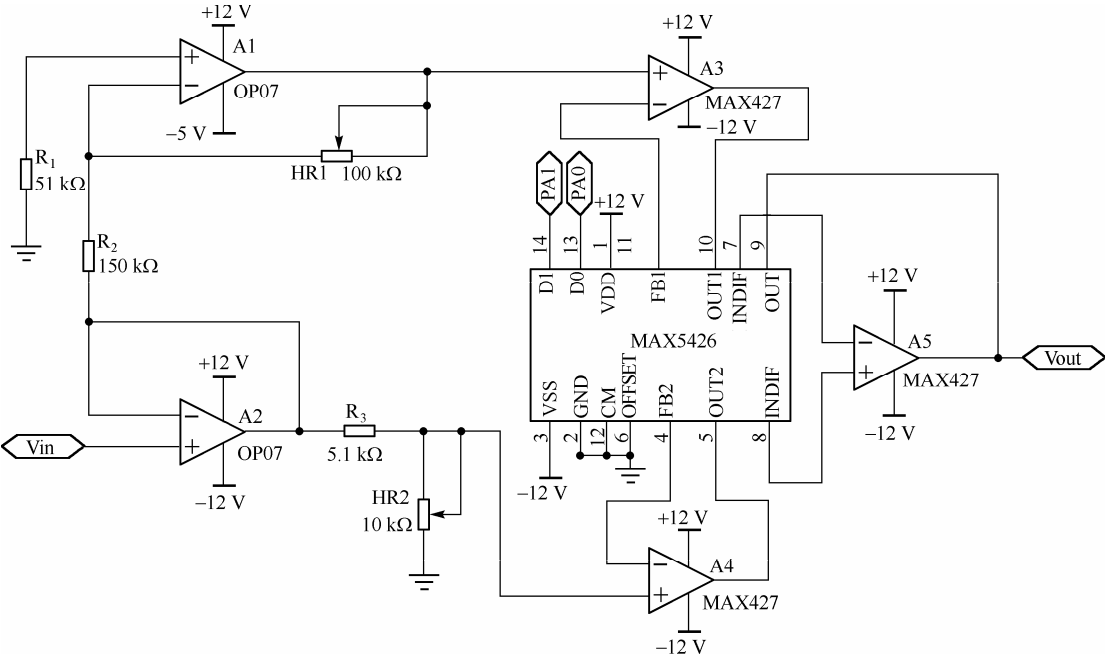


图 8.17 程控增益放大电路

输入信号与程控放大倍数选择对应关系如表8.1所示，其中增益系数选择 11, 10, 01 和 00 分别对应 8, 4, 2 和 1 倍。

表 8.1 输入信号与程控放大倍数选择对应关系

输入真有效值电平 V_{rms}/V	增益系数 (D1 D0) 选择
$0 \leq V_{rms} < 0.625$	11
$0.625 \leq V_{rms} < 1.25$	10
$1.25 \leq V_{rms} < 2.50$	01
$2.50 \leq V_{rms} < 5.00$	00

3. 真有效值转换电路

采用单片电压真有效值转换器 AD536 实现交流信号的真有效值转换,该芯片能计算包括交流和直流成分的任何复杂输入波形的真有效值并转换为直流信号输出，且与波形参数及其失真度大小无关，因而减小了设计难度。其参考电路如图8.18所示。实际应用时，AD536 转换后得到的电压真有效值有 0.5%的最大误差，且有交流纹波，在 AD536 的输出端接一个低通滤波器，可消除交流纹波，从而改善电路性能并获得稳定的直流信号。

4. 50 Hz低通滤波电路

为了得到 50 Hz 工频信号的基波，需设计 50 Hz 低通滤波器电路，较为便利的方法是采用开关电容滤波器。可使用 MAXIM 公司的 8 阶椭圆低通滤波器 MAX293，其中心频率 f_0 从 0.1 Hz~25 kHz，由外部时钟频率 f_c 控制，关系为 $f_r/f_0=100$ ，设计时外部时钟频率取 5.9 kHz(工频信号的频率波动范围为 50 ± 1 Hz)，使其截止频率为 50 Hz，高于 50 Hz 的电压谐波被衰减，获得基波。其参考电路如图8.19所示。

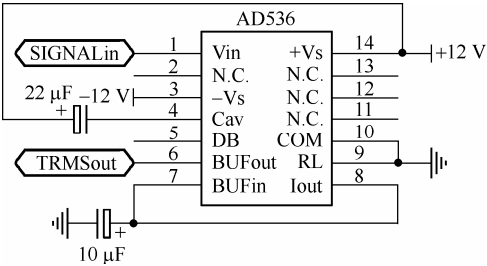


图 8.18 AD536 真有效值转换电路

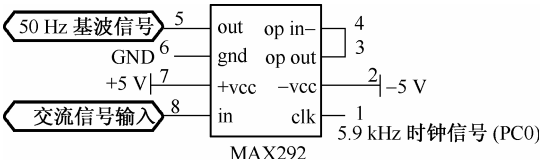


图 8.19 50 Hz 低通滤波电路

应注意的是，当采用开关电容滤波器时，所获取的基波信号的电平由于被叠加直流电平可能产生偏移，因此需在滤波电路的后端加设直流电平校正及隔离驱动电路，以保证信号的准确测量。

5. A/D采样电路

在第 5 章已对数据采集系统的设计做了详细的讨论，一般来说，微处理器选定后字长也就确定了。理论上，微处理器本身的运算精度并不受其字长的限制，因为可以通过宽字节浮点运算的方法来取得任意高的运算精度。也就是说，在实际应用中，运算的精度并不取决于微处理器本身，而取决于输入到微处理器的数据的有效位数的多少。若输入数据的有效位数较高，则经处理后的精度也就较高；反之，则较低。因此，A/D 转换器的字长(采样、量化后 A/D 转换器输出位数的多少)关系到系统测量的精度。考虑系统设计的指标要求，8 位 A/D 转换器的转换精度只能达到 $\pm 0.2\%$ (幅度分辨率为 0.391%)，12 位 A/D 转换器的转换精度为 $\pm 0.012\%$ (幅度分辨率为 0.0244%)。虽然基本部分对误差的要求不高，仅为 $\pm 0.8\%$ ，但考虑到系统的综合误差和信噪比要求，为了保证数据采集精度，应取高于 8 位的 A/D 转换器，设计使用高性能逐次逼近型 12 位 ADC MAX197。MAX197 具有转换精度高、接口简单、转换速率快等优点，MAX197 的主要特点在第 7 章中已有介绍，其参考电路如图8.20 所示。

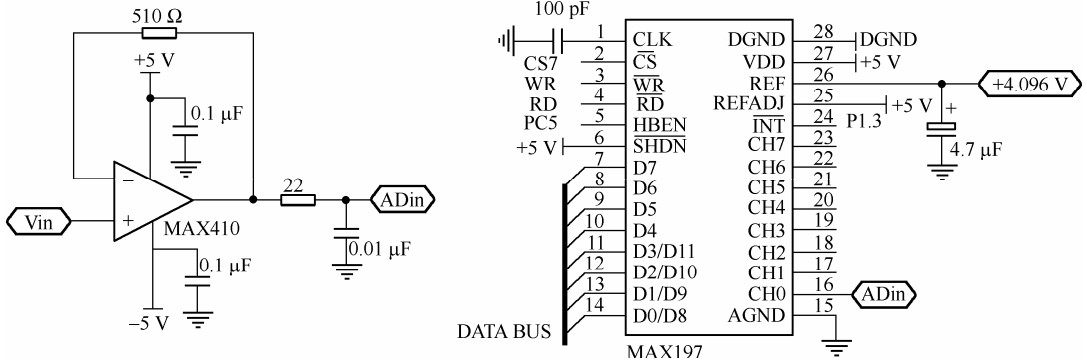


图 8.20 A/D 采样电路

6. 相差测量电路

利用等精度测量原理，可以实现高精度的相差测量，具体测量原理在第 3 章已经详细地讨论过，测量系统采用 1 MHz 的频标信号，其参考电路如图8.21 所示。

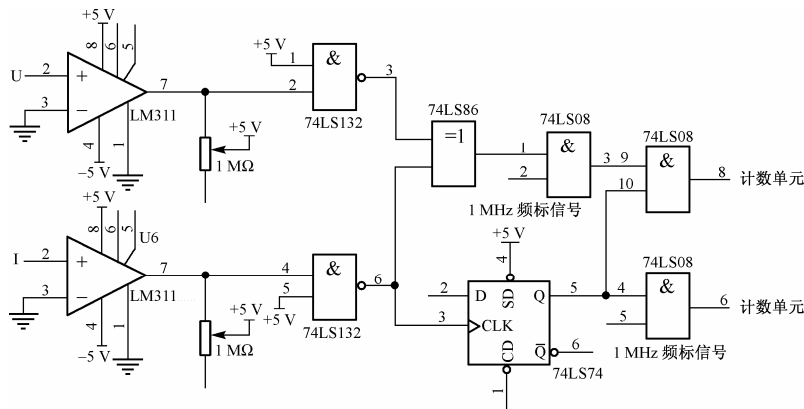


图 8.21 相差测量电路

7. 键盘设置

由于设计要求测量的参量比较多，因此键盘设置要充分考虑到设计的要求，参考的按键设置方式如表 8.2 所示。

表 8.2 键盘定义与按键功能

键 盘 定 义	按 键 功 能	键 盘 定 义	按 键 功 能
0	选择基波加谐波	Right	视在功率
1	选择基波	Enter	无功功率
空键 1	自动换挡	Left	有功功率
空键 3	测量电流	Up	测量相差
空键 2	测量电压	Down	功率因数
		Clear	清零停止；自检

8.3.3 系统软件设计

本系统软件设计主程序采用结构化的程序设计方法，功能模块各自独立。主程序可分为自检、初始化、键盘管理等模块，其参考流程图如图8.22所示。简要说明如下：

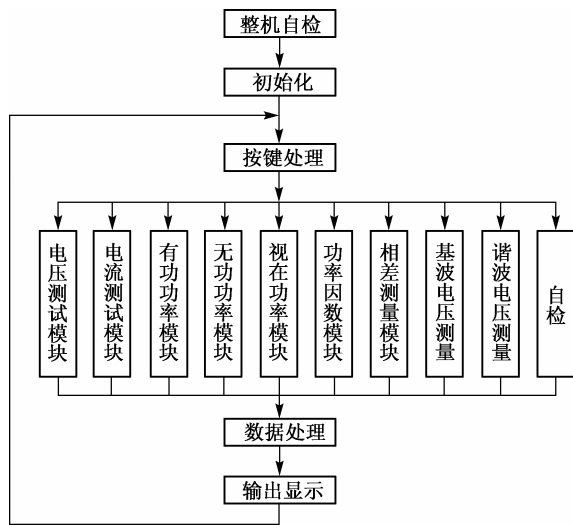


图 8.22 系统软件设计主程序参考流程图

- (1) 初始化模块完成对单片机系统及外围电路的初始化及设置。
- (2) 开机后仪表分挡实现自动校正，且自动调零。
- (3) 在系统运行的任何时候都可选择对系统进行自检操作，并给出相关自检结果信息。
- (4) 键盘管理的功能是根据按键的功能号和当前状态转入相应的状态，并执行有关功能模块，然后进入下一次循环。
- (5) 采用八位数码管显示，第 1 个数码管显示表示的物理量 (U 表示电压有效值， I 表示电流有效值， P 表示有功功率， Q 表示无功功率， N 表示功率因数， R 表示视在功率， Y 表示电压与电流的相差角)。第 8 个数码管显示度数标志，第 2~7 个数码管显示相应测量值的大小。

8.4 简易数字存储示波器

1. 设计任务

设计并制作一台用普通示波器显示被测波形的简易数字存储示波器，其示意图如图8.23所示。

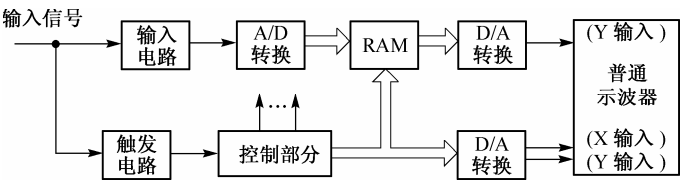


图 8.23 简易数字存储示波器示意图

2. 设计基本要求

- (1) 要求仪器具有单次触发存储显示方式，即每按动一次“单次触发”键，仪器在满足触发条件时，能对被测周期信号或单次非周期信号进行一次采集与存储，然后连续显示。
- (2) 要求仪器的输入阻抗大于 100 kΩ，垂直分辨率为 32 级/div，水平分辨率为 20 点/div。设示波器显示屏的水平刻度为 10 div，垂直刻度为 8 div。
- (3) 要求设置 0.2 s/div, 0.2 ms/div, 20 μs/div 三挡扫描速度，仪器的频率范围为 0~50 kHz，误差≤

5%。

- (4) 要求设置 0.1 V/div, 1 V/div 二挡垂直灵敏度, 误差 $\leq 5\%$ 。
- (5) 仪器的触发电路采用内触发方式, 要求上升沿触发、触发电平可调。
- (6) 观测波形无明显失真。

3. 设计提高部分要求

(1) 增加连续触发存储显示方式, 在这种方式下, 仪器能连续对信号进行采集、存储并实时显示, 且具有锁存(按“锁存”键即可存储当前波形)功能。

(2) 增加双踪示波功能, 能同时显示两路被测信号波形。

(3) 增加水平移动扩展显示功能, 要求存储深度增加一倍, 并且能通过操作“移动”键显示被存储信号波形的任一部分。

(4) 垂直灵敏度增加 0.01 V/div 挡, 以提高仪器的垂直灵敏度, 并尽力减小输入短路时的输出噪声电压。

8.4.1 简易数字存储示波器的系统设计方案

电子学中的信号大多都是时间变量信号, 可以用一个时间函数 $f(t)$ 来描述。要研究一个复杂的信号(波形), 最好的办法是把它显示出来, 即把看不见、摸不着的电磁现象、电磁信号转换成可以直接观察的图像。如果这种转换和显示是无失真的, 那么显示出的图像就是该电磁现象的全貌。它包含了电磁现象发生过程的全部信息, 从而可以进行深入的分析。示波器就是这样的一种基本的、应用最广泛的时域测量仪器, 它能测量信号的幅度、频率、周期等基本参量。常用的示波器有模拟示波器和数字存储示波器两类。与模拟示波器不同, 数字存储示波器(Digital Storage Oscilloscope, DSO)将被测模拟信号先送至高速 A/D 转换器进行取样、量化和编码, 编码后的数据存储到 RAM 中(写过程), 然后再将这些数字码从 RAM 中依次取出, 经 D/A 转换使其包络重现为原始的输入模拟信号(读过程)。因为数字存储示波器采用了实时取样技术, 所以既可观察重复的周期信号, 也可以观察单次信号。数字存储示波器的最大特点是: 可以存储和调用显示特定时刻信号, 具有实时显示和存储两种工作模式。简易数字存储示波器系统由信号调理电路、触发电路、A/D 转换电路、D/A 转换电路、X 和 Y 通道、控制器等部分组成。被测的输入信号, 经过调理、采样、量化后存入数据存储器, 然后在控制器的控制下, 从存储器读出数据并转换为模拟信号, 输入到普通示波器的 Y 通道; 同时系统还需要产生对应的扫描信号, 加入到通用示波器的 X 通道, 将被测的输入信号在通用示波器的荧光屏上显示出来。DSO 的实时采样工作方式决定了系统设计方案必须采用高速数据的采集和处理技术, 因而高速数据采集、存储和回放电路的设计将成为该系统设计的主要难点。

由于单片机时钟频率的限制, 数据采集过程必须由高速逻辑器件控制, 因此控制器的设计包括单片机设计和可编程逻辑器件设计两部分。利用可编程逻辑器件完成对信号的采集和存储控制, 承担底层控制; 利用单片机实现对可编程逻辑器件及整个系统的管理, 承担顶层控制和数据处理, 如从键盘输入选择采样速率、选择信号调理电路的增益、将存储的数字信号进行数据处理并恢复为模拟信号进行显示等操作。采用 CPLD 和单片机实现的简易数字存储示波器的总体设计方案如图8.24所示。

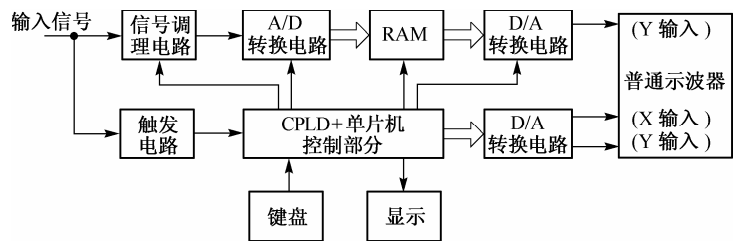


图 8.24 基于 CPLD 高速逻辑控制的简易数字存储示波器的总体设计方案

8.4.2 主要技术指标与设计参数计算

高速数据采集、波形存储与回放电路是数字存储示波器的主要组成部分，也是系统设计的难点。数字存储示波器的技术指标是系统设计中参数计算的依据，下面首先介绍一下数字存储示波器的主要技术指标，然后讨论有关部分的设计参数的计算。

1. 主要技术指标

(1) 最大采样速率

单位时间内完成一次完整 A/D 转换的最高次数称为仪器的最大采样速率(也称为最大数字化速率)，常以频率来表示。采样速率越高，说明仪器捕捉信号的能力越强。采样速率主要由 A/D 转换速

率来决定。采样速率通常用每秒的采样点即 Sa/s (sample/second) 来表示。在实际应用中, 实时采样速率可根据被测信号所设定的扫描时间因数 (Ts/div) 来推算, 其推算公式为

$$f = N/T$$

式中, N 为每格的取样数; T 为扫描时间因数 (s/div), 即扫描一格所占用的时间。

(2) 存储带宽 B

存储带宽与取样速率密切相关, 并取决于采集的方式: 重复采集或单次采集。重复采集是指利用等效时间取样技术 (包括顺序取样) 来测量快速的重复信号的采集方式, 存储的波形是输入信号波形的多次重复取样的合成, 这时的带宽称为模拟带宽或等效带宽。而单次采集采用实时取样技术来测量信号, 不论信号是重复的还是单次出现的, 都只做一次采集, 并以实时方式显示, 所以能观察单次信号和缓慢变化的信号, 这时的带宽又称为单次带宽或实时带宽。也就是说, 带宽既取决于最大采样速率, 也取决于所采用的显示恢复技术。当采用光点显示时, 需从存储器 RAM 中读出数据, 这时的取样密度 (速率) 表示为示波器屏幕 X 轴上显示的被测信号每格所对应的取样点数, 记为 “个/cm” 或 “个/div”。其最高带宽是奈奎斯特频率极限, 它是最大数字化采样频率 f_s 的 $1/2$ 。实际应用中, 为保证显示波形的分辨率, 往往要求增加更多的采样点, 一般取 $N = 4 \sim 10$ 倍或更多, 即存储带宽为 $B = f_s/N$ 。

(3) 分辨率

分辨率是反映存储信号波形细节的综合特性, 它包括垂直分辨率和水平分辨率。垂直分辨率与 A/D 转换器的分辨率相对应, 常以屏幕每格的分级数来表示。示波管平面是一个二维平面, 坐标刻度为 $8 \times 10 \text{ div}$, 如果采用 8 位的 A/D 转换器 (256 级), 则仪器垂直分辨率为 32 级/div, 水平分辨率由存储器的存储容量来决定, 如 1 K 字的存储容量为 1024 个字, 水平级数为 1024, 相当 10 位的分辨率。因此, 时间分辨率等于取样间隔, 它可以根据使用情况调节, 对存储的每一个波形来说, 可以规定需要多少存储空位。比如, 对于某一信号波形所用的存储容量为 200 个字节, 屏幕水平显示格为 10 格, 则仪器的水平分辨率为 $200/10 = 20$ 点/div。

(4) 存储容量

存储容量又称为记录长度, 它由采集存储器的最大存储容量来表示, 常以字 (Word) 为单位。数字存储器常采用 256, 512, 1K, 4K 等容量的高速 RAM 存储器。

(5) 扫描时间因数 S

扫描时间因数取决于来自 A/D 转换器的数据写入获取存储器的速度及存储容量, 而存储容量又称为存储长度, 通常又定义为获取波形的取样点数组, 用直接存放 A/D 转换后数据的存储单元数来表示。因此, 扫描时间因数为相邻两个取样点的时间仰角 (取样窗口) 与每个取样点数的乘积, 即 $Ts/div = (1/Sa)N$ 。其中, 实时采样时间 $1/Sa$ 既为写脉冲周期也为读脉冲周期, N 为屏幕中每格内对应的存储容量。一般用 S 表示扫描时间因数。可以看出, 在 A/D 转换器速率相同的条件下, 存储容量越大, 则扫描时间因数也越大。若 A/D 的转换速率为 20 MSa/s , 存储容量为 1 K, 则最快扫描时间因数为 $5 \mu s/div$; 若存储容量为 10 K, 则最快扫描时间因数为 $50 \mu s/div$ 。也就是说, 存储容量与水平分辨率在数值上有互为倒数的关系, 存储容量的选取直接影响取样时间窗口的大小, 时间窗口缩短会失去信号的重要部分, 时间窗口增大会使水平分辨率降低。

2. 设计参数的计算

(1) 存储深度

存储深度即为记录长度, 以 M 来表示, 根据题目要求, 其存储深度为 $20 \text{ 点/div} \times 10 \text{ div} = 200$ 点。

(2) 采样速率 f_s 与扫描时间因数 S

如前所述, 采样速率通常指示波器进行 A/D 转换的最高速率。在固定存储深度 M 的条件下, 采样速率 f_s 是随选用的扫描挡位而变化的, 与扫描时间因数 S 成反比。已知每格采样点数为 20, 则 $f_s = 20/S$ 。任务要求设置 0.2 s/div, 0.2 ms/div, 20 μ s/div 三挡扫描速度 (对应的采样速率分别为 100 Hz, 1 kHz, 1 MHz), 可以设计扫描时间因数从 20 μ s/div~200 ms/div, 共有 13 挡, 涵盖题目要求的 3 挡扫描时间因数。依据上式计算对应的采样速率, 如表 8.3 所示。

表 8.3 扫描时间因数 S 与采样速率 f_s 的关系对照表

S/div	20 μ s	40 μ s	100 μ s	200 μ s	500 μ s	1 ms	2 ms
f_s/kHz	1000	500	200	100	40	20	10
S/div	5 ms	10 ms	20 ms	50 ms	100 ms	200 ms	
f_s/kHz	4	2	1	0.4	0.2	0.1	

值得注意的是, 采样速率是通过与时基信号进行可调分频得到的。实际设计中, 需要对输入分频器的时基信号进行可调分频得到表 8.3 所列的采样速率。

(3) A/D 芯片的选取

A/D 芯片的位数取决于垂直分辨率, 垂直分辨率指标一般是指仪器内部采用的 A/D 转换器在理想情况下进行量化的比特数。指标要求垂直分辨率为 32 级/div, 示波器垂直满刻度为 8 格。垂直方向上应该有 $32 \times 8 = 256 = 2^8$ 量化级, 因此 A/D 转换器的位数不应该低于 8 位。此外, 从表 8.3 可知, A/D 转换器的采样速率不应低于 1 MHz。可选的 A/D 转换器很多, 考虑成本及接口的便利性, 本设计可选用 TI 公司的 8 位并行高速 A/D 转换器 TLC5510, 该芯片用单 5 V 供电, 额定输入信号范围为 0~2 V, 最高采样速率为 20 MSa/s, 内部带有采样保持电路和基准电阻, 控制简单。

(4) 程控放大器增益

程控增益放大器的放大倍数的要求应满足垂直灵敏度指标, 同时兼顾系统输入信号和 A/D 输入信号幅值范围。指标要求设置 0.01 V/div, 0.1 V/div, 1 V/div 三挡垂直灵敏度, 系统输入信号范围为 -4 V~+4 V; A/D 输入信号幅度为 0.6~2.6 V (采用内部基准源)。当示波器满度显示时, 被测信号的幅度 V_m 将分别是 $V_{i1} = 1 \text{ V/div} \times 8 \text{ div} = 8 \text{ V}$, $V_{i2} = 0.1 \text{ V/div} \times 8 \text{ div} = 0.8 \text{ V}$, $V_{i3} = 0.01 \text{ V/div} \times 8 \text{ div} = 0.08 \text{ V}$ 。

综合以上指标, 设置三挡放大倍数分别为 0.25、2.5 和 25, 其中 0.25 倍放大倍数对应 1 V/div 垂直灵敏度; 2.5 倍放大倍数对应 0.1 V/div 垂直灵敏度; 25 倍放大倍数对应 0.01 V/div 垂直灵敏度。

(5) 输入信号峰-峰值

可通过 89C51 读取并分析 RAM 中的数据, 判断最大值和最小值, 由式 (8.14) 可以计算出输入信号的峰-峰值:

$$V_{p-p} = [(D_{\max} - D_{\min}) / (255/8)] \times A$$

(8.14)

式中, $D_{\max}$ 为波形数据最大值; $D_{\min}$ 为波形数据最小值; A 为垂直分辨率 (V/div)。

(6) 垂直灵敏度误差

垂直灵敏度误差也称为垂直偏转因数误差, 其计算公式为

$$e = \frac{V_1 - V_2}{D} \times 100\% < 5\%$$

(8.15)

式中, e 为垂直灵敏度误差; V_1 为测量读数 (V); V_2 为校准信号每格电压值 (V); D 为校准信号幅度 (div)。需要说明的是, 按照误差理论, 有正误差和负误差两种情况。因此, 设计时应按照垂直灵敏度误差 $\leq |5\%|$ 进行信号通道误差分配和设计。

(7) 扫描误差

扫描误差是测量时间间隔的准确度，主要取决于定时信号的准确度及扫描电压的准确程度。定时信号由时基信号分频获得，而时基信号产生于晶体振荡器，其稳定度为 10^{-8} 以上，和题目要求的扫描误差小于 5%相比，可以忽略不计。定时信号由时基信号经 CPLD 分频后获得(定时信号影响采样速率的准确度)，其误差在 ns 量级，也可忽略不计。而扫描电压由后向通道的 D/A 及输出电路产生，若选用 8 位 D/A，其量化误差为 $1/2^8 \approx 0.4\%$ ，输出电路由于增益的非线性也使扫描电压产生误差，通常估计扫描电压误差小于 2%。对于扫描误差，一般用标准周期光脉冲进行校验，用示波器的 Δt 光标自动测量标准信号周期时间值与真值之间的百分比误差，即扫描误差。

8.4.3 系统各模块电路设计

数字存储示波器按其工作原理可分为波形取样与存储、波形显示、波形测量与波形处理等功能模块。下面分别进行讨论。

1. 前级信号处理电路设计

设计要求对两个被测信号同时进行显示，因此必须同时对两个被测信号进行采样、存储，可以采取交替方法或者双通道方法。交替方法采用一个高速率切换的模拟开关分别选通两路信号进入采样电路，两路波形数据存储在同一片存储器的奇、偶地址位，双踪显示时，先扫描奇数位，再扫描偶数位地址的数据，从而实现双踪显示。交替方案节省了一个 A/D 转换器和一片存储器，节约了成本，下面以这种方法为例进行讨论。

前级信号处理模块的设计利用模拟开关 MAX333A 构成单、双踪切换电路，并配合运算放大器实现程控增益放大电路，此模块的主要功能是控制两路信号的分时选通，并对输入信号的幅值进行程控放大，使输入信号的幅度满足模数转换器所要求的动态转换范围，并满足垂直灵敏度指标要求。CH1 和 CH2 两路波形信号分别经过 OP07 构成的跟随器，输入到模拟开关 MAX333A。由 CPLD 产生的地址信号的最低位 AR0 控制 CH1 和 CH2 的高速切换，分时采样两路信号。程控放大单元采用宽带运放构成放大电路，高频信号失真小，并且由精密电位器构成反相放大电路，完成输入信号的 0.25 倍、2.5 倍和 25 倍精确放大。A/D 转换器对输入信号的要求通常是单极性的，而被测模拟信号可能是双极性的。为了满足 A/D 转换器的要求，在后级运放实现 1.6 V 电平抬升，以保证输入到 A/D 转换器的信号电平大小与 A/D 转换器的 0.6~2.6 V 的动态转换范围相匹配。前级信号处理电路的设计框图如图 8.25 所示。三挡程控增益放大电路如图 8.26 所示。

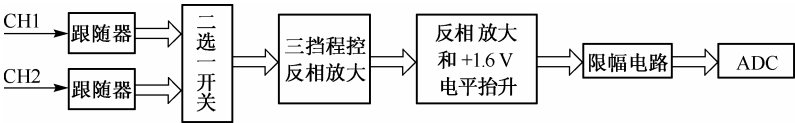


图 8.25 前级信号处理电路的设计框图

2. 数据采集电路设计

系统设计采用实时采样技术，重复采集和单次采集具有相同的带宽。A/D 转换器采用高速模数转换器 TLC5510，该芯片内含 S/H 电路，最高采样速率为 20 MSa/s。外部基准电压从引脚 REFT 和 REFB 接入，对于 2 V 输入信号而言，要求 $V_{REF} = 2\text{ V}$ ，可接入外部电压基准源也可采用内部参考电压源。TLC5510 的工作时序较为简单，时基信号经可调分频后获得采样时钟信号，在 CLK 信号的上升沿开采样，当输出使能端 OE 接低电平时，在 2.5 个 CLK 周期后，采样量化数据自动呈现在数据线上。设

度。可采用稳定度较高的 40 MHz 有源晶振，将其作为 CPLD 的时钟基准输入。由 CPLD 生成一个分频比可调的分频器，得到不同的采样时钟信号。这一模块除有源晶振需外接以外，其余部分均在 CPLD 中实现。

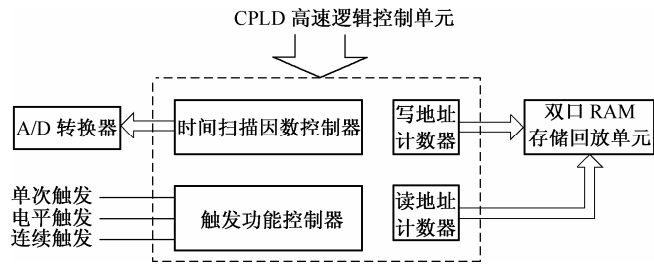


图 8.28 CPLD 高速逻辑控制单元框图

2) 触发功能控制器

在满足触发条件时，启动对被测信号进行采样，可实现单次触发、电平触发和连续触发功能。

3) 写地址计数器

用来产生写地址信号，它由 CPLD 生成二进制的计数器，计数器的位数由存储长度来确定。写地址计数器的计数频率与 A/D 转换器的采样时钟频率相同，其输出的写地址信号送至双口 RAM 的地址线。

4) 读地址计数器

用来产生读地址信号，它由 CPLD 生成二进制的计数器，其输出的读地址信号将数据从双口 RAM 中读出。

4. 双口RAM读写采样数据电路设计

A/D 采样量化数据存储于 RAM 中，选用双口 RAM 进行存储。双口 RAM IDT7164 有两组相互隔离的数据线、地址线、片选线和读写控制线，它们可以对 RAM 内部的存储单元同时进行读写操作，并且互不影响，这样就解决了数据的高速读写操作的问题。存储数据线与 A/D 采样量化输出数据线相连，读出数据线与列扫描 D/A 数据线相连。存储与读出的地址线受 CPLD 控制，具体连接关系如图 8.29 所示。

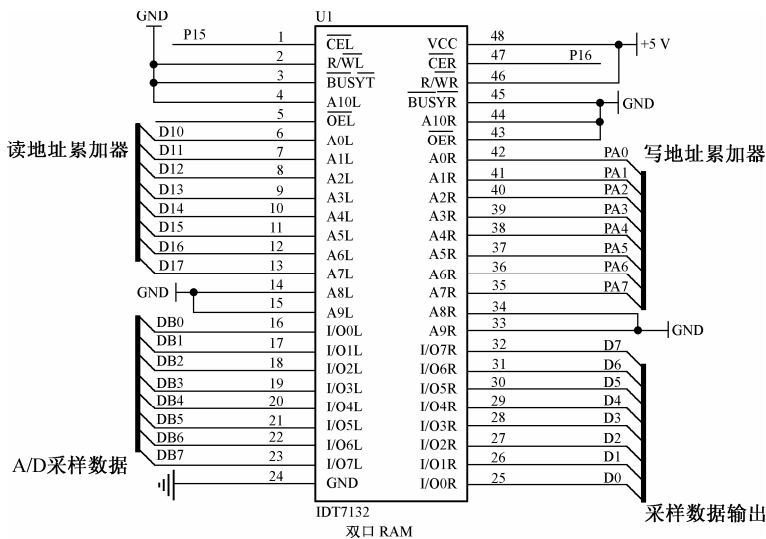


图 8.29 双口 RAM IDT7164 连接图

的上下平移。模拟开关 MAX333A 实现单/双踪切换功能，参考实现电路如图8.33所示。

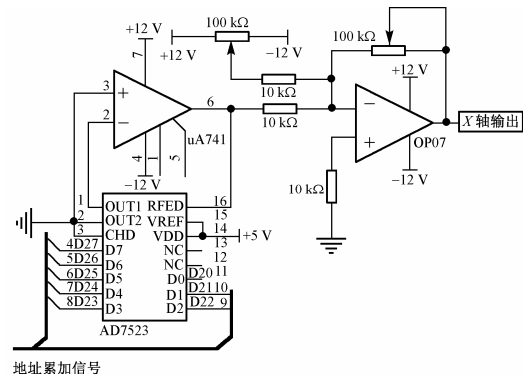


图 8.32 行扫描电路图

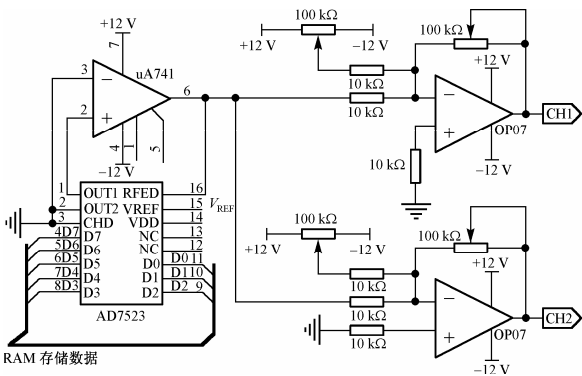


图 8.33 列扫描电路图

7. 输出信号与模拟示波器的连接

显示采用模拟示波器的 X-Y 方式。在 X-Y 方式下，示波器的垂直轴和水平轴的偏转电压由外部提供。屏幕上每一个位置都有对应的一个 X-Y 坐标。因此，只要提供波形的坐标数据，经 D/A 转换送至 X, Y 轴即可。显示时，地址计数器以固定的频率循环计数，地址信号直接送至行扫描 D/A 转换器，产生周期锯齿波，对应 X 轴偏转电压；双口 RAM 数据送至列扫描 D/A 转换器，对应 Y 轴偏转电压。X-Y 方式下的显示原理如图8.34 所示。

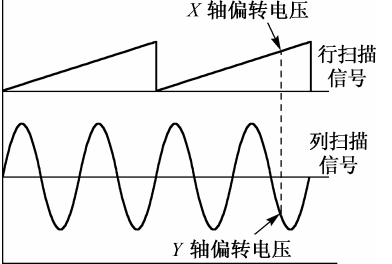


图 8.34 X-Y 方式下的显示原理图

8. 键盘设计

键盘是输入控制命令的人机接口，根据测量功能的要求，可设置扫描速度、垂直灵敏度、单次/连续、单踪/双踪、扩展(移动)/常态、锁存、启动/停止、上/下等功能键。对于单片机系统，还应设置一个 RESET 键，用于将系统复位成默认状态。

8.4.4 系统软件设计

简易数字存储示波器的系统软件主程序流程图如图8.35所示。在实际设计中，一个系统的硬件和软件共同完成一个任务，因此，软、硬件之间既有分工也有相互配合，系统设计要求在硬件设计时必须考虑有关软件的相互配合的问题。在前面的硬件设计中已对相关的软件设计做出了说明。在软件设计中，为了程序的简捷、运行可靠，采用自顶向下的编程方法，将任务划分为多个不同的功能模块，以减小编程的难度。限于篇幅，这里仅给出系统主程序流程图的设计。

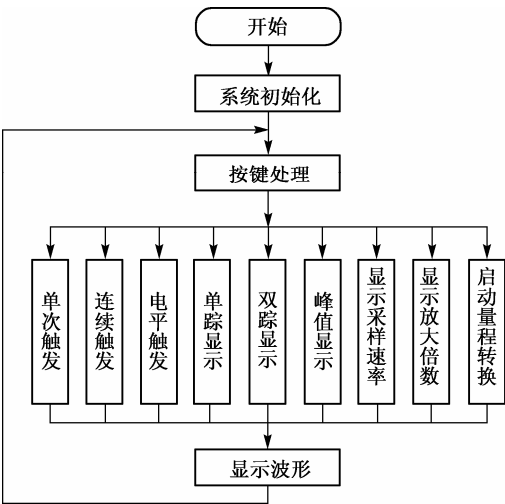


图 8.35 简易数字存储示波器的系统软件主程序流程图

8.5 简易逻辑分析仪设计

1. 设计任务

设计并制作一个 8 路数字信号发生器与简易逻辑分析仪，其结构框图如图 8.36 所示。

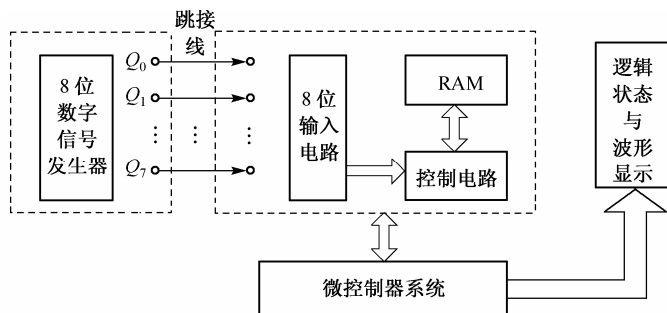


图 8.36 8 路数字信号发生器与简易逻辑分析仪结构框图

2. 设计基本要求

(1) 制作数字信号发生器

能产生 8 路可预置的循环移位逻辑序列，输出信号为 TTL 电平，序列时钟频率为 100 Hz，并能够重复输出。

(2) 制作简易逻辑分析仪

① 具有采集 8 路逻辑信号的功能，并可设置单级触发字。信号采集的触发条件为各路被测信号电平与触发字所设定的逻辑状态相同。在满足触发条件时，能对被测信号进行一次采集存储。

② 能利用模拟示波器清晰稳定地显示所采集到的 8 路信号波形，并显示触发点位置。

③ 8 位输入电路的输入阻抗大于 50 k Ω ，其逻辑信号门限电压可在 0.25~4 V 范围内按 16 级变化，以适应各种输入信号的逻辑电平。

④ 每通道的存储深度为 20 bit。

3. 设计提高部分要求

(1) 能在示波器上显示可移动的时间标志线，并采用 LED 或其他方式显示时间标志线所对应时刻的 8 路输入信号逻辑状态。

(2) 简易逻辑分析仪应具备 3 级逻辑状态分析触发功能，即当连续依次捕捉到设定的 3 个触发字时，开始对被测信号进行一次采集、存储和显示，并显示触发点位置。3 级触发字可任意设定(例如，在 8 路信号中指定连续依次捕捉到两路信号 11, 01, 00 作为三级触发状态字)。

(3) 触发位置可调(即可选择显示触发前、后所保存的逻辑状态字数)。

(4) 其他(如增加存储深度后分页显示等)。

8.5.1 任务分析与方案论证

逻辑分析仪属于数据域测试仪。在数据域分析中，通常关注的不是每条信号线上电压的确切数值，而只需要知道信号线上的电压是处于低电平还是高电平，以及各信号相互配合在整体上表示什么意义。通常认为，数据域分析是研究数据流、数据格式、设备结构和用状态空间概念表征的数字系统。逻辑分析仪的原理框图如图 8.37 所示，它主要包括数据捕获和数据显示两大部分。数据捕获部分包括信号

输入、采样、数据存储、触发产生和时钟选择等。外部被测信号送到信号输入电路，与门限电平进行比较，通过比较器整形为符合逻辑分析仪内部逻辑电平的信号。采样电路在采样时钟控制下对信号进行采样，采样获得的数据流送到触发产生电路进行触发识别。根据数据捕获方式，在数据流中搜索特定的触发字，当搜索到符合条件的触发字时，产生触发信号。数据存储电路在触发信号的控制下进行相应的数据存储。数据捕获完成后，由显示控制电路将存储的数据以适当的方式(波形或字符列表等)显示出来，以便对捕获的数据进行观察分析。时钟电路可以选择外时钟或内时钟作为系统的工作时钟。

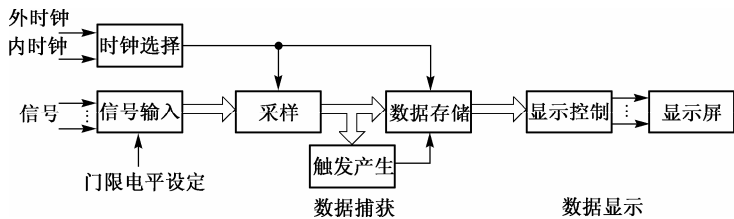


图 8.37 逻辑分析仪的原理框图

按照设计任务要求，设计分为数字信号发生器的设计和简易逻辑分析仪的设计两部分。数字信号发生器要求序列时钟频率仅为 100 Hz，且负载能力要求也不高，由 8 路信号产生电路、循环移位寄存器、100 Hz 时钟产生电路和逻辑信号输出部分构成。简易逻辑分析仪由信号输入、RAM 及其读 / 写控制、波形显示等部分组成。系统设计可以采用以下两种方案实现。

(1) MCU 方式。即采用 MCS-51 单片机作为系统核心，要求单片机完成基本的控制和分析处理，以及完成 8 路 TTL 电平数据的采集、存储和示波器的显示控制等。从题目分析来看，这一方案是可实现的，但从训练的角度来看，采用这一方案减小了训练难度，并且在工程实现上意义不大。因为实际的逻辑分析仪是以远高于信号变化的频率进行高密度采样的，以发现逻辑信号中的延迟和竞争冒险现象，这需要大容量的 RAM 和高速逻辑控制电路。

(2) MCU+FPGA 方式。考虑到 EDA 技术的应用已成为电子系统设计的主流，FPGA 的组合逻辑能力和时序实现能力较强，编程灵活，因此可以采用 FPGA 作为逻辑判断核心、MCS-51 单片机辅助控制的方式。单片机控制信号发生器，向 FPGA 发出 8 位移位信号，利用键盘向 FPGA 送触发字，实现人机交互功能。利用 FPGA 实现逻辑判断、波形存储及波形显示控制等。该方案既发挥了 FPGA 逻辑处理性能优越的特性，又发挥了单片机控制功能强的特点。下面就以此方案为例来讨论简易逻辑分析仪的设计。

8.5.2 理论分析和参数计算

根据取样方式的不同，逻辑分析仪可分为逻辑状态分析仪和逻辑定时分析仪两类，它们的组成原理基本相同。逻辑状态分析仪用于系统的软件分析，逻辑定时分析仪主要用于信号逻辑时间关系的分析。显然，设计任务要求的是逻辑定时分析仪的设计。下面首先介绍逻辑定时分析仪的主要技术指标，然后讨论有关部分的设计参数的计算。

1. 主要技术指标

(1) 定时分析最大速率

逻辑定时分析仪(以下统称为逻辑分析仪)在工作时考察两个系统时钟之间数字信号的传输状态和时间关系。因此，逻辑分析仪内部有时钟发生器(采样时钟)，在内部时钟的控制下记录数据，与被测系统异步工作，这是逻辑分析仪的主要特点。为了提高测量准确度和分辨率，要求内部时钟频率远高于被测系统的时钟频率。由此可看出，所谓的定时分析最大速率，是指逻辑分析仪工作在定时分析

方式时的最大数据采集速率，它可以是实际的采样时钟最高频率，也可以是等效采样速率。

(2) 通道数

通道数是指逻辑分析仪信号输入通道数量，它包括数据通道数量和时钟通道数量。通道越多，可同时观测的信号就越多。

(3) 触发方式

逻辑分析仪的触发方式越多，其数据窗口定位就越灵活。一般有随机触发、通道触发、毛刺触发、字触发等基本方式，还有一些触发附加功能，如延迟触发、限定触发、组合触发、序列触发等。

(4) 输入门限变化范围

输入门限变化范围是指逻辑分析仪前端探头输入信号门限的可变范围。一般逻辑信号门限电压为 $-2 \sim +5 \text{ V}$ ，其变化范围越大，则可测试的数字系统逻辑电平种类越多。

(5) 存储深度

存储深度是指每个通道可以存储的数据位数，单位为比特/通道，一般为几比特/通道至几十比特/通道。

2. 设计参数计算

(1) 定时分析数据采集速率

采样速率的高低与数据采集的质量有着十分重要的关系。为能得到更高的清晰度，大多采用非同步方式，用高于几倍被测系统频率的时钟频率进行采样。如果采样速率较低，系统的测量分辨率也会跟着降低。例如使用 1 MHz 的时钟脉冲，则每个时钟的周期为 $1 \mu\text{s}$ ；如果输入为比 $1 \mu\text{s}$ 更窄的脉冲，则检出的概率很小。由于任务要求序列时钟频率为 100 Hz ，字采集的间隔时间为 10 ms ，因此这一指标的实现比较容易。

(2) 存储深度

任务要求显示 8 路信号波形，每通道的存储深度为 20 比特/通道，设每个页面的存储深度为 M_1 ，则

$$M_1 = 8 \times 20 = 160 \text{ bit} = 20 \text{ Byte}$$

设按照扩展要求扩展存储页数为 n 页，设系统的存储深度为 M_n ，则

$$M_n = 20n \text{ Byte}$$

(3) 多级逻辑信号门限电压的设定

按照任务要求，逻辑门限电压可在 $0.25 \sim 4 \text{ V}$ 范围内按 16 级变化，即最低电压为 0.25 V ，最高电压为 4 V 。按 16 级变化，即其步长为 0.25 V 。因此，对应的 16 级门限电压为 0.25 V ， 0.5 V ， $\dots$ ， 3.75 V ， 4.00 V 。

8.5.3 系统各个模块的设计与实现

1. 数字信号发生器的设计

数字信号发生器由 8 路信号产生电路、循环移位寄存器、 100 Hz 时钟产生电路和逻辑信号输出部分构成。8 路信号产生电路是通过 8 路拨动开关预置输入信号的状态。采用 FPGA 实现循环移位逻辑信号序列产生单元，该单元集成了循环移位寄存器、 100 Hz 时钟产生电路和逻辑信号输出三部分。数字信号发生器的原理框图如图 8.38 所示。通过 8 路拨动开关预置输入的 8 位逻辑状态，在时钟控制下，根据 8 路拨动开关所预置的逻辑状态产生 8 路移位信号。

2. 逻辑分析仪的设计

该部分是本系统的核心部分,主要采用 FPGA 来实现对输入 8 路逻辑信号的采集和存储,从而实现 8 路逻辑状态分析,并将采集存储的波形数据输出供单片机读取。逻辑分析仪主要是由信号的输入通道部分、触发控制电路、存储电路和输出电路等部分构成。信号的输入通道部分的阻抗匹配和门限控制采用分立器件来实现。

(1) 信号输入调理部分

任务要求 8 位输入电路的输入阻抗大于 50 kΩ, 设计采用射极跟随器作为前向通道的输入电路,一方面可以达到高输入阻抗的要求;另一方面可以在被测信号源和所设计的逻辑分析仪之间起隔离作用,其电路如图 8.39 所示。输入电路运算放大器 A 采用 JFET 输入运算放大器,其输入阻抗可达 $10^{12} \Omega$ 。为了对信号源呈现为稳定负载,可在电路的输入端并联一个电阻 R, 这时等效输入电阻 R'_i 为 $R'_i = R \parallel R_i$ 。由于要求输入阻抗大于 50 kΩ, 又 $R_i \gg 100 \text{ k}\Omega$, 因此可以认为 $R'_i \approx R$, 按照要求取 $R = 100 \text{ k}\Omega$ 。

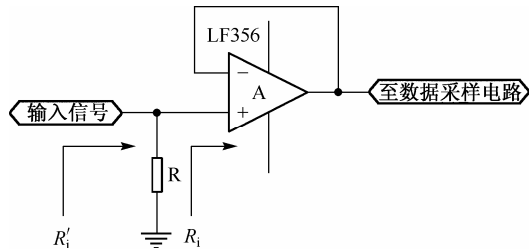


图 8.39 前向通道的输入电路

又任务要求逻辑信号门限电压可在 0.25~4 V 范围内按 16 级变化,设计采取由单片机控制 8 位 D/A 转换器 DAC0832 的数字量输入的方法来控制比较器的基准电压。DAC0832 的参考电压 V_{REF} 设为 5 V, 输入数据端接收预置数字控制量 D_{in} , 输出电压与所输入数据的关系式为

$$V_{\text{out}} = \frac{D_{\text{in}}}{256} \times V_{\text{REF}} = \frac{D_{\text{in}}}{256} \times V_{\text{in}} \tag{8.16}$$

因此通过改变 DAC0832 输入的数字量,就可以实现门限电平很精确地在 0.25~4 V 范围内按 16 级变化。输入通道的参考硬件电路如图 8.40 所示。

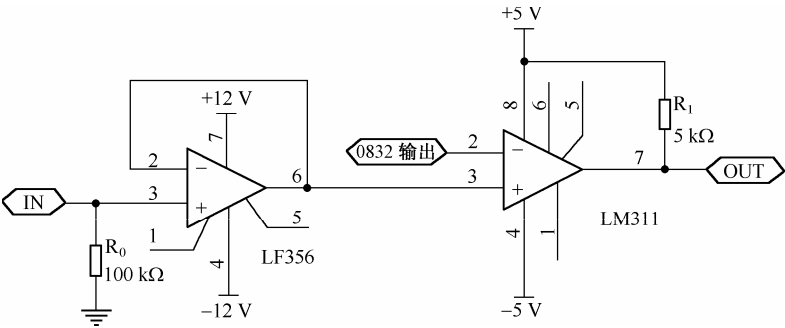


图 8.40 输入通道的参考硬件电路

(2) 触发电路部分

触发电路的功能是当满足触发字条件后产生一个触发信号,用以控制采样电路。触发电路有 3 种触发方式: 序列触发、并行触发和直接触发。

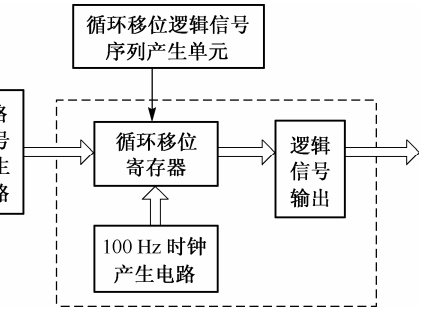


图 8.38 数字信号发生器的原理框图

① 序列触发 设置几个按预定次序排列的触发字,只有被测的系统信号按同样的顺序先后满足所有的触发字时,才产生触发信号。

② 并行触发 设置几个按预定次序排列的触发字,只有当这几路信号同时满足所设的触发条件时,才产生触发信号。

③ 直接触发 当触发电路接收到一个脉冲信号时,就将它当做触发信号直接送给触发控制电路处理。

根据任务要求,设计采用序列触发方式实现多级逻辑状态分析触发功能。

① 单级触发字触发 在正常时钟及数据流经过时,对采集的 8 路数据进行判断,如果与预先设定的值相符,则激活触发信号,对数据进行采集和存储,并记录触发点地址。

② 3 级逻辑状态分析触发 此方式需要对数据流实时采集,并且保留在 3 级移位寄存器中,将移位寄存器中的 3 级数据按照单级触发的判断方式进行判断,并对所有的判断结果进行“与”运算,最终产生触发使能信号,便可以得到 3 级触发。

此外,为了实现触发点位置可调(即可选择显示触发前、后所保存的逻辑状态字数),可以采用加窗技术,即将存储空间增加到 80 bit,但只是显示其中的 20 bit,由窗的位置决定显示哪些数据。

(3) 数据存储部分

所采集到的数据为一个字节,每一位的值代表相应路数的信息,在模拟示波器上显示时,需将 8 路信号分离开来,因此存储数据时需先对其进行处理,即先将采集到的数据分离成 8 bit,然后将每个 bit 填成一个字节(例如,字节的高 3 位表示该 bit 所在的输入路数,第 4 位为该 bit 值,低 4 位填 0),这样即可直接将数据送显,无需再对数据进行处理。数据存储可采用 FPGA 的内部 RAM,分别存储 8 路数据,这样可在时钟信号的上升沿时将数据分别存入相应地址的 RAM 中。

(4) 波形显示部分

依据示波器原理,若要真实显示被测信号的波形,则必须在 Y 通道输入被测信号的同时,在 X 通道输入一个随时间线性变化的电压,通常采用锯齿波电压。同时为了稳定显示波形,要求每个扫描周期所显示的信号波形在荧光屏上完全重合,即曲线形状相同,并在同一起始点。要满足这一点,扫描电压周期和被测信号周期必须成整数倍关系,这样才能保证每次扫描的起始点都对应在与信号电压相同的相位点上,使每次扫描显示的波形都重叠在一起。因此,可采用依次显示各列的方式,即横轴波形为锯齿波;纵轴依次显示第 1 路的第 1 个状态,第 2 路的第 1 个状态,……,第 7 路的第 8 个状态,第 8 路的第 8 个状态。若要较好地显示整个画面,则根据人眼的视觉惰性,每帧的刷新频率应大于 16 Hz。如果选 50 Hz,则横向扫描信号的每一个小锯齿波的频率应为 $50f$ (f 为采样后数字信号码元的速率)。

● 通过模拟示波器显示波形

用两个相互独立的数模转换器 DAC0832 输出两路模拟信号:一路输出锯齿波作为示波器的扫描信号对 X 轴进行扫描;另一路输出相应 Y 轴方向的逻辑高低电平状态,每 bit 的逻辑电平信号用 12 个点扫描,并采用同样的扫描方式显示其他 7 路信号波形及同步的 100 Hz 时钟信号。

● 显示时间标志线

固定时间轴(即 X 轴)坐标,对 Y 轴进行扫描画出相应的时间标志线,并可以随意地左右移动时间标志线的位置,移动步进为 X 轴方向的一个扫描单位。此外,通过 8 只单色发光二极管显示时间标志线所对应时刻的逻辑信号状态。单片机数据总线通过锁存器 74LS373 来控制发光二极管的亮灭情况,其参考硬件电路如图 8.41 所示。

● 显示触发点位置

由于触发点位置是在时钟信号的上升沿时显示,所以在时钟上升沿时对触发点处 Y 轴进行扫描以加强触发时刻时钟上升沿的亮度,从而清晰地显示触发点的位置,并可随着触发点的不同位置而改变

显示的位置。

● 分页显示

每页显示 8 路信号和同步 CLK 信号，显示的存储深度为 20 bit，分 4 页显示 RAM 中存储的 80 bit 数据，通过按键控制上、下翻页显示，根据 CLK 所在页码的不同而对 CLK 信号上升沿的触发标志进行消隐。

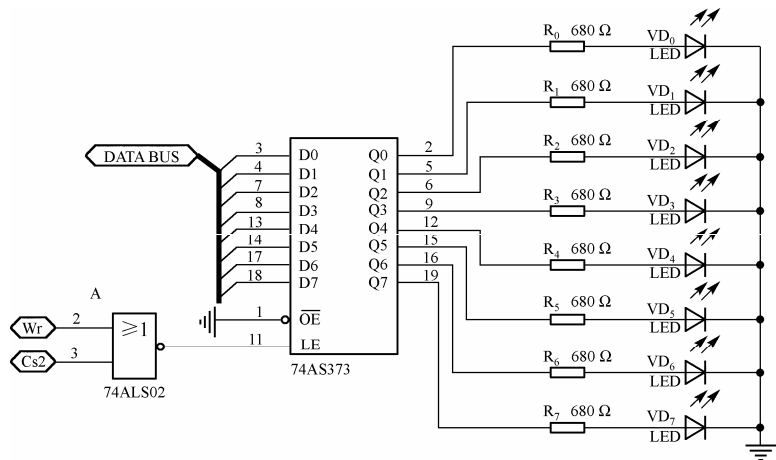


图 8.41 逻辑信号状态显示电路

8.5.4 系统软件设计

该系统采用单片机和 FPGA 共同完成对逻辑分析仪的软件控制。单片机负责读取键盘有关门限电压值、触发模式和触发字的设定，然后通知 FPGA 启动相应的数字信号发生器、触发判断、RAM 储存等模块，再由 FPGA 及 D/A 完成示波器显示功能，最后由单片机完成反馈时间标志线所对应的逻辑状态的显示。

1. FPGA程序设计

采用硬件描述语言 VHDL/Verilog HDL，用 FPGA 来实现逻辑分析部分，包括控制模块、触发控制电路模块、RAM 读/写模块等。

其中，控制模块根据控制字的状态，从单片机获取逻辑分析部分所需的各种控制量，如单级触发字、3 级触发字、掩码和所需保存的逻辑状态的数目。控制模块的控制方式可按照如表 8.5 所示的方式设置。

表 8.5 控制方式

控制字 CONT[3..0]	获取到的数据 DATAIN 的意义	
000	单级触发字 CODEW	
010	多级触发字	第一个触发字 CODE1
011		第二个触发字 CODE2
100		第三个触发字 CODE3
101	掩码 MASK	
110	所需保存的逻辑状态的数目 N	

当各路被测信号电平与触发字所设定的逻辑状态相同时，触发控制电路产生一个有效触发信号

TRIG。触发控制电路模块的工作状态表可参见表 8.6。

表 8.6 触发控制电路模块的工作状态表

当前状态	转换条件	下一状态	触发输出
IDLE	RESTART	START	0
	(NOT) RESTART	IDLE	0
START	CONW == CODE	SIGNAL	1
	CONW == CD1	MUL1	0
SIGNAL	CONW == CODE	HOLD	0
	OTHERS	SIGNAL	0
MUL1	CONW == CD2	MUL2	0
	OTHERS	START	0
MUL2	CONW == CD3	HOLD	1
	OTHERS		0
HOLD	CONW == CONTROL	START	0
	OTHERS	HOLD	0

说明：① 模块中状态机的转换条件
CONW—由输入的数据 DATAIN 和掩码 MASK 相与而得；CODE—单级触发字；
CD1, CD2, CD3—3 级触发字。

② 模块中状态机的特殊状态
IDLE—空闲态，其转换控制信号为 RESTART；
START—开始态，当满足 CONW==CODE 进入单级触发状态，当满足 CONW==CD1 进入 3 级触发状态；
HOLD—保持态，只有在外控制信号 CONTROL 控制下，才能再次进入状态机的状态转换。

③ 模块在状态机的输出
1—输出的触发信号有效，为高电平；0—输出的触发信号无效，为低电平。

在触发信号 TRIG 和预置保存的逻辑状态数目的控制下调整后端模块中 RAM 的写方式，其控制方式可参见表 8.7。

表 8.7 控制方式表

触发信号 TRIG	输出的后端控制		输出 FULL
	输 出	状 态	
0 无效	00	失调	0
1 有效	01	触发	0
1 有效	10	写满	1

说明：失调状态—在写信号的控制下，可以循环地向 RAM 中写信号；
触发状态—当接收到有效触发信号时，只能对 RAM 写满一次；
写满状态—不再向 RAM 中写信号，并输出已满信号 FULL。

数据输出部分，接到前端有效 FULL 信号后开始对 RAM 从触发位置开始进行读操作。

2. 单片机程序设计

单片机部分整体软件流程图如图8.42所示，波形显示部分软件流程图如图8.43所示。

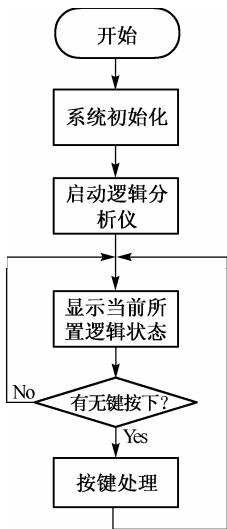


图 8.42 单片机部分软件整体流程图

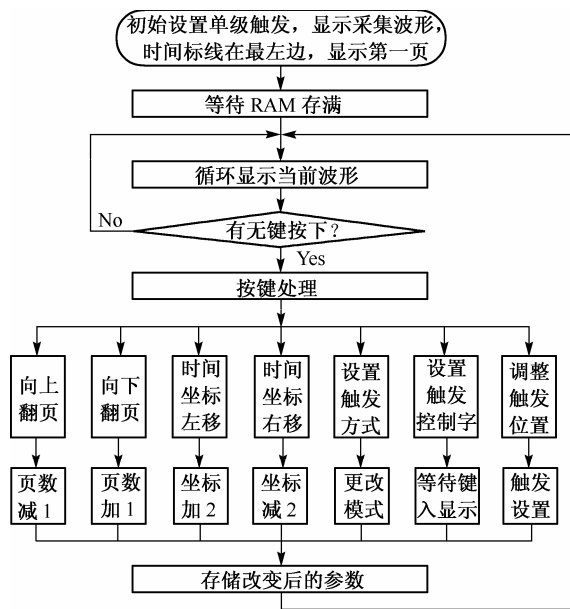


图 8.43 波形显示部分软件流程图

8.6 本章小结

本章以数字电容测量仪设计、数字式工频多用表设计、数字存储示波器设计和简易逻辑分析仪设计等例子介绍了简单测量仪器的设计方法与实现过程。

8.2 节主要讨论的是如何将电容参数的测量转化为时间参数的测量。另外，由于电子计数器具有测量精度高、速度快、自动化程度高等特点，很多物理量在测量过程中都可转化成时间参数进行测量。因此，设计的重点就是如何把电容容值参数转化为时间参数及如何实现时间参数的精确测量的问题。

8.3 节要求在掌握工频参数基本含义的前提下，完成对交流电压、交流电流、有功功率、无功功率、视在功率、功率因素等参数的测量。本设计的重点在于对工频参数的理解、真有效值转换和相位差测量方案的选取及电压参量的测量系统设计。通过设计可以看出，这些参数都是基于对电压的测量，因此，本节也充分体现了作为电子测量的一个主要参数——电压测量的重要性。

8.4 节主要讨论了简易数字存储示波器的设计。该系统设计中控制器的设计具有一定的难度，控制器采用两层控制，顶层用单片机来管理系统的任务调度、数据处理及人机交互等，底层由 FPGA 处理高速控制问题。对系统设计和实现中的软硬件分工的问题需认真周密考虑。系统设计核心就是如何实现高速数据采集、存储与回放系统的设计。

8.5 节主要讨论了基于单片机最小系统平台和 FPGA 模块联合实现简易逻辑分析仪设计的方法，包括了信号发生及处理、触发控制、数据存储及波形显示等模块的分析和实现。逻辑分析仪的设计实际上也是一个数据采集、存储和回放系统的设计，设计时，要充分利用单片机强大的控制功能和 FPGA 存储和逻辑处理的能力，分模块设计，最后再完成系统的整机联调和测试。

第9章 基于FPGA/CPLD的电路设计流程简介

9.1 引言

前面的章节介绍了复杂电子系统设计的一些基本思想、方法和平台，所讲授的内容均是基于传统单片机平台的设计方法，它是整个复杂电子系统设计的基础。然而，随着现代电子技术及芯片制造技术的飞速发展，越来越多的基于可编程器件 FPGA/CPLD、DSP 处理器、ARM 处理器的设计平台逐渐融入甚至替代传统基于单片机的设计平台。

可编程逻辑器件，尤其是 FPGA，由于它的诸多优点使得它成为现代电子系统设计中不可或缺的部分，随处可见它的身影，在前几章的设计训练中已有非常多的设计实例采用了可编程逻辑器件。现代的电子产品和数十年前的电子产品相比，体积越来越小，功能越来越强，价格却越来越低，其变化之大令人惊叹！这一变化得益于微电子技术的进步。我们应该清醒地看到，微电子技术的快速发展及 EDA 技术的引入使得现代电子产品的复杂性大大上升，电子产品的设计理念、方法及环境有了重大的变化。尤其是现代电子系统的设计方法与环境所发生的重大变化，是对电子信息类大学生的一个强有力的挑战。如何面对这一挑战已成为高等工程教育必须解决的问题。

可编程器件 FPGA 被广泛应用，是因为 FPGA 可以替代其他 PLD 或者各种中小规模数字逻辑芯片在数字系统中的应用，而且 FPGA 也是实现具有不同逻辑功能 ASIC 的有效办法。FPGA 芯片在每次加电或需要时从外部读入构造数据，由于其本身的结构，它具有以下优点：无限可重复编程，且编程速度快（毫秒级）；不需要专用的或价格高的编程设备，用户只需对装载构造数据的 PROM 或 EPROM 编程，不需要对 FPGA 本身进行编程；在系统加电运行时完成重构，需要时可让芯片重新编程，读入另一种设计的构造数据，完成系统的在线重构。

从本章开始到后续章节将按照由易到难、由简到繁的设计任务来安排训练内容，从不同的设计角度和不同的应用领域来介绍 FPGA 的设计和应用例子。本章是基础性的训练，主要内容是基于 FPGA/CPLD 的电路设计流程简介，将详细介绍基于 Quartus II 软件开发的基本流程、可编程器件的选择及硬件电路的设计。希望读者通过本章的学习，能够快速了解 FPGA 开发软件的基本操作流程，能够设计出基于 FPGA 器件的硬件开发平台，能够掌握现代电子系统设计常用的开发环境和设计模式。

9.2 FPGA设计软件

在复杂电子系统的设计中往往需要借助 EDA (Electronic Design Automation) 技术，常用的 EDA 软件按照主要功能或主要应用场合可大致分为：电子电路设计与仿真工具、PCB 设计软件、IC 设计软件、FPGA/CPLD 设计工具。FPGA 开发需要一些专用的 FPGA 工具软件，其功能包括 FPGA 程序的编写、综合、仿真及下载等。就整体而言，目前的 FPGA 工具软件可以分为两类：一类是 FPGA 芯片生产商直接提供的集成开发环境，如 Altera 公司的 Quartus II、Xilinx 公司的 ISE、Actel 公司的 Libero 及 Lattice 公司的 IspDesignExpert；另一类是其他专业的 EDA 软件公司提供的辅助软件工具，统称为第三方软件，如业内主流的仿真工具 Modelsim 和综合工具 Synplify/Synplify Pro，它们都可以嵌入到 Quartus II 或者 ISE 等集成开发环境中辅助完成仿真、综合等操作。

作为全球最大的两个 CPLD/FPGA 制造厂商 Xilinx 公司和 Altera 公司,其市场占有率已达到 80% 左右,其中 Xilinx 的市场占有率约为 50%,Altera 的市场占有率约为 30%。从这个数据上看,Xilinx 在这个市场占有率有主导地位,但是这个数据并不能说明全部情况。因为这些年 Altera 的市场份额不但来自他的 CPLD 公司,而且还来自 ASIC, ASSP 及处理器等其他技术产业的份额,所以从新技术增长率方面考虑,Altera 公司是相对高些的。Altera 公司是目前唯一批量供应低成本低功耗并得到广泛应用的 65 nm FPGA 产品(Cyclone III)的公司,同时 Altera 也是目前唯一推出 40 nm FPGA(Stratix IV)的公司。此外,Altera 也是目前唯一能够与 FPGA 配合使用同一软件(Quartus II)来开发 ASIC(HardCopy)的公司,这些都足以说明 Altera 在软硬件方面的综合实力。因此,在本书以后的章节中将主要介绍基于 Altera 公司 FPGA 器件的复杂电子系统的设计方案。

Quartus II 软件是 Altera 公司新开发的 EDA 软件设计平台,提供了全面的逻辑设计能力,其功能十分强大,包括支持 VHDL 和 Verilog 硬件描述语言的设计输入、基于图形的设计输入、系统功能仿真、系统时序仿真分析、布局布线、Nios II 嵌入式软核、SOPC Builder、DSP Builder 等。操作者使用 Quartus II 软件完成可编程器件的开发,包括设计输入(文本、图形和波形等设计方法自由组合)、项目编译(完成最小化逻辑综合、适配设计项目于单个器件或多个器件及形成编程和配置数据等工作)、设计校验(包括功能仿真、时序仿真、影响速度的关键路径的延时预测及多种系列器件混合使用的多器件仿真)及器件编程等过程,通过 Quartus II 软件的图形用户界面(GUI)菜单和工具条能轻松、直观地完成系统的设计与实现。图9.1给出了 Quartus II 软件的典型设计流程。

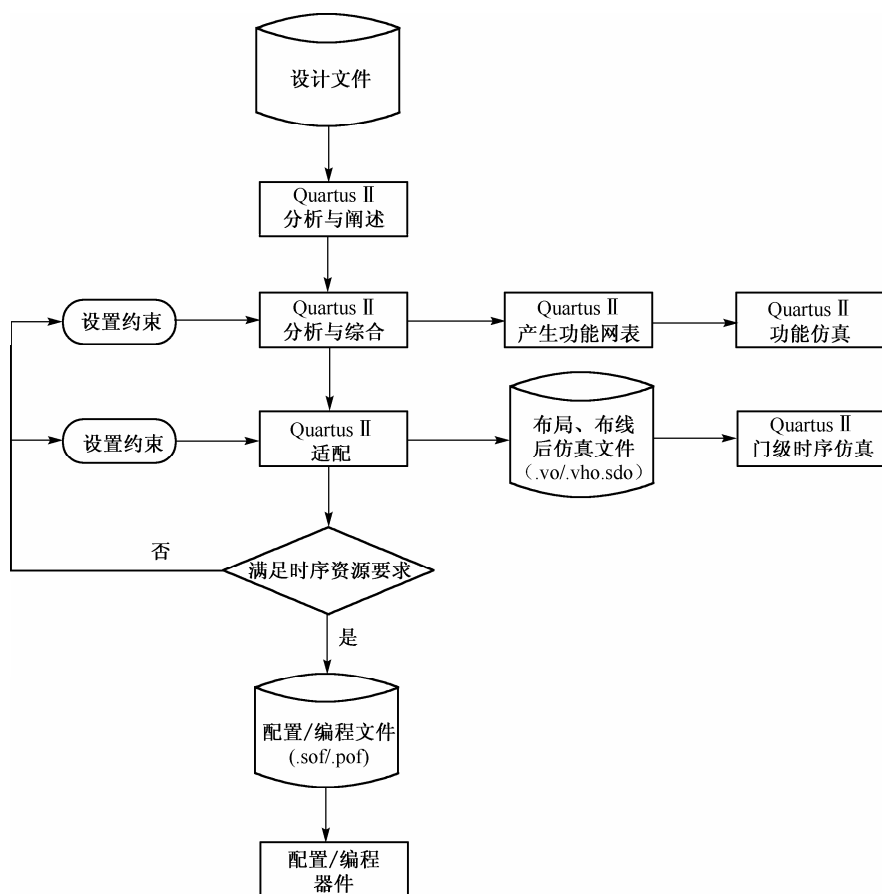


图 9.1 Quartus II 软件的典型设计流程

Quartus II 支持多种设计输入方法，其本身包含的编译器支持原理图式图形设计输入、文本编辑输入(如 AHDL, VHDL, Verilog) 和内存编辑输入(如 Hex, Mif)等。本节将以设计一个十六进制的计数器为例，分别介绍 HDL 设计输入和图形设计输入。

9.2.1 HDL设计输入

进入 Windows 后，双击 Quartus II 图标，如图9.2所示。

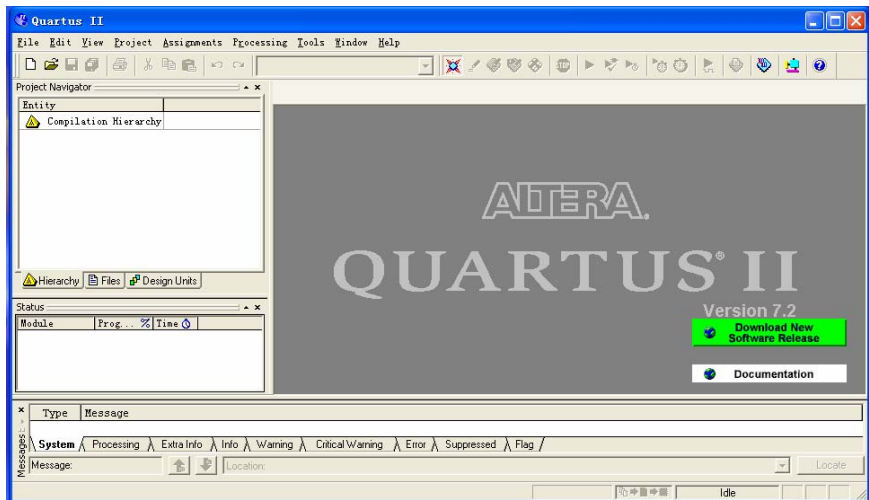


图 9.2 Quartus II 管理器

1. 工程建立

使用“New Project Wizard”，可以为工程指定工作目录、分配工程名称及指定最高层设计实体的名称，还可以指定要在工程中使用的的设计文件、其他源文件、用户库和 EDA 工具及目标器件(也可以让 Quartus II 软件自动选择器件)。

建立工程的步骤如下：

- (1) 选择“File”菜单下的“New Project Wizard”选项，如图9.3所示。

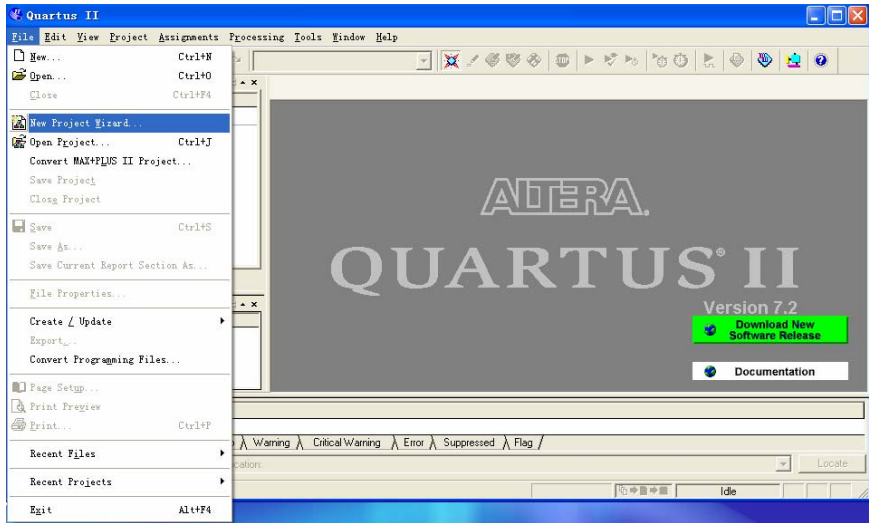


图 9.3 建立项目

(2) 输入工作目录和项目名称，如图 9.4 所示。可以直接单击“Finish”按钮，以下的设置过程可以在设计过程中完成。

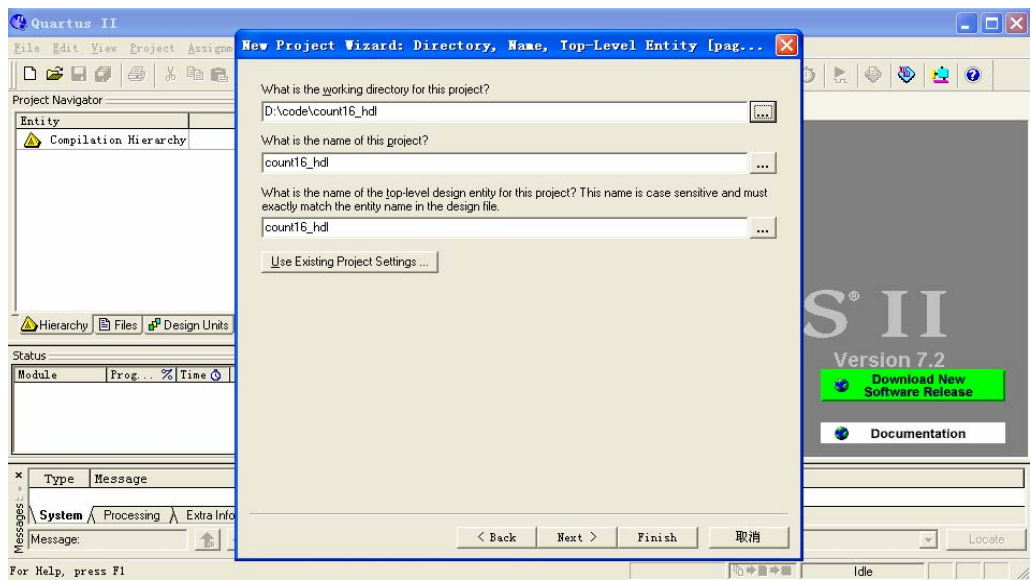


图 9.4 项目目录和名称

(3) 加入已有的设计文件到项目，可以直接单击“Next”按钮，设计文件可以在设计过程中加入，如图 9.5 所示。

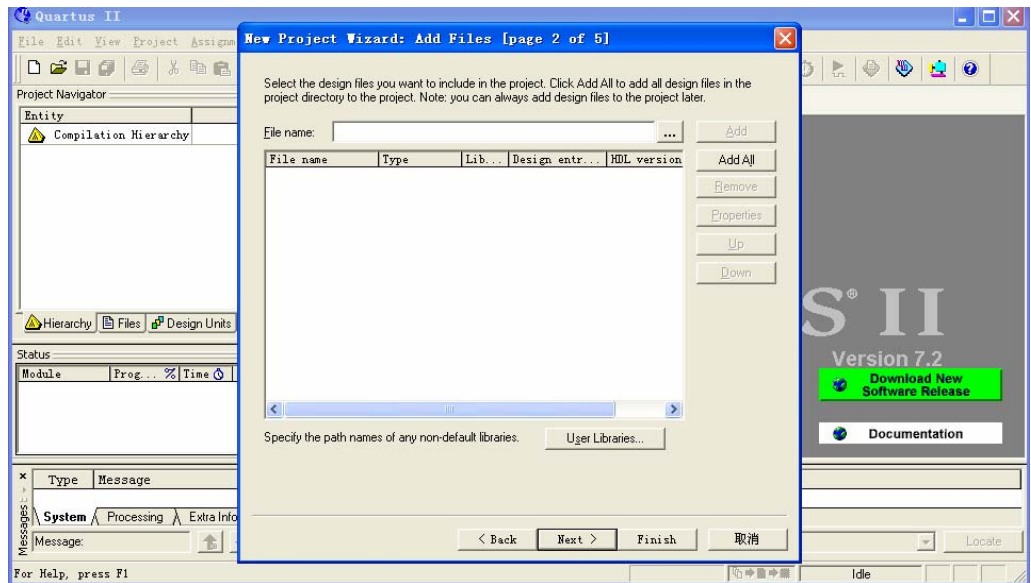


图 9.5 加入设计文件

- (4) 选择设计器件 EP2C35F672C6，如图 9.6 所示。
- (5) 选择第三方 EDA 综合、仿真和时序分析工具，如图 9.7 所示。
- (6) 建立项目完成，显示项目概要，如图 9.8 所示。

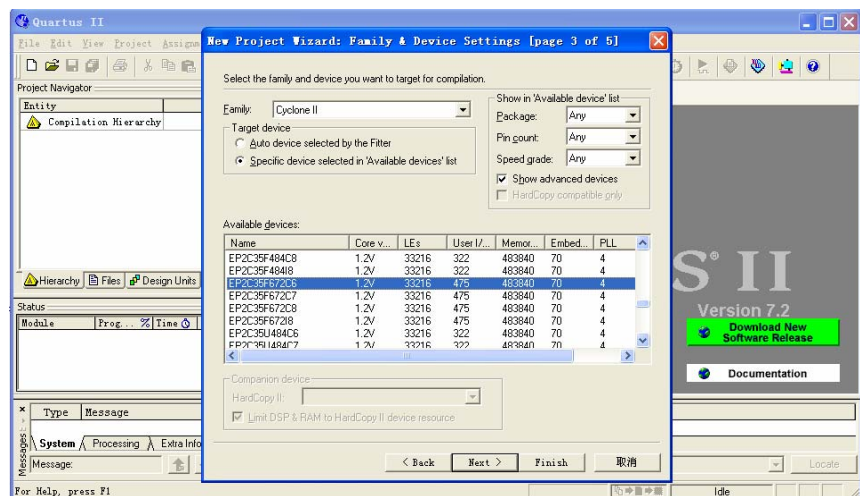


图 9.6 选择器件

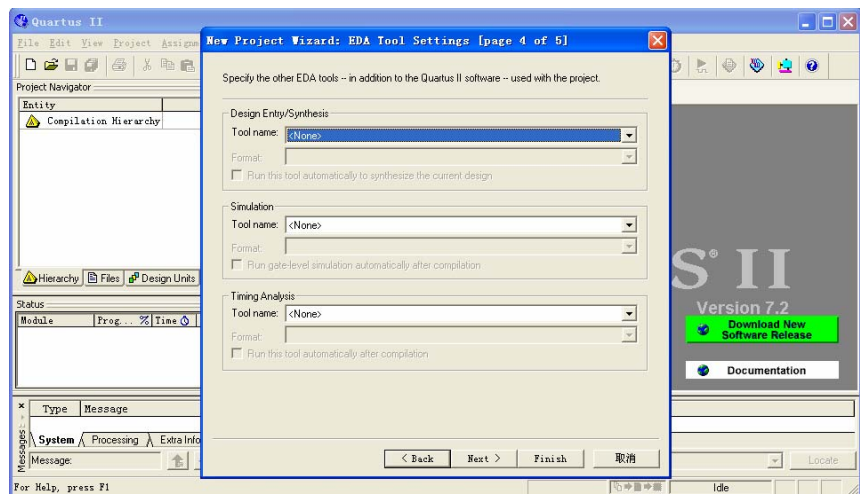


图 9.7 选择 EDA 工具

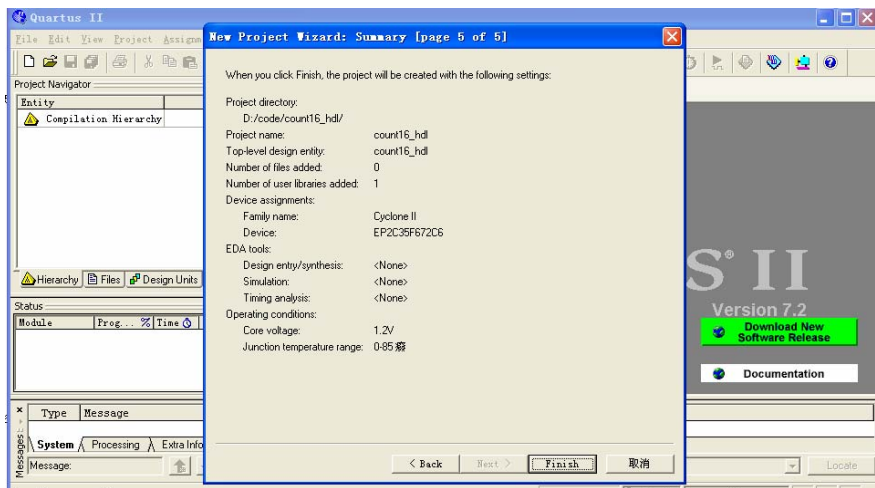


图 9.8 项目概要

2. HDL输入

HDL 输入的操作步骤如下：

(1) 选择“File”菜单下的“New”选项，在弹出的窗口中选择“Verilog HDL File”，如图9.9所示。

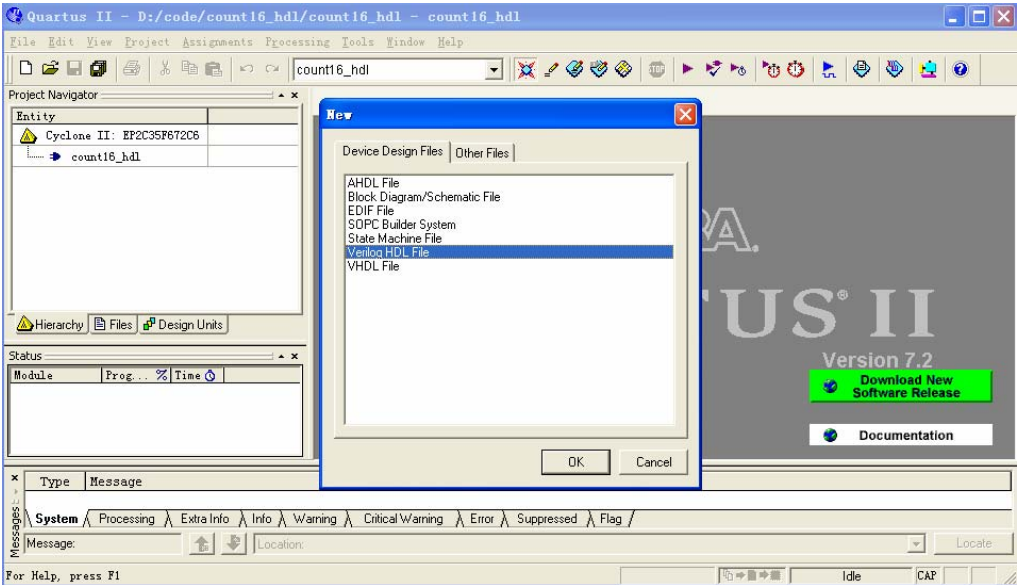


图 9.9 新建 Verilog HDL 文件

(2) 在空白的文本编辑区进行文本编辑，图9.10直接给出了一个十六进制计数器的程序。

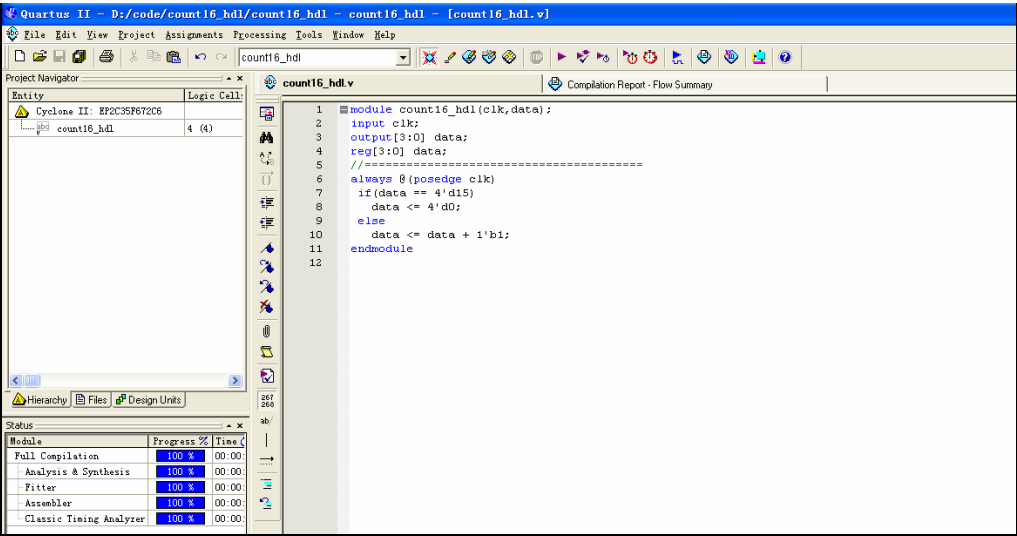


图 9.10 完成编辑后的屏幕

(3) 单击编译器快捷方式“▶”按钮，完成编译后，弹出菜单报告错误和警告数目，并生成编译报告，如图9.11所示。Quartus II 编译器的功能包括设计错误检查、逻辑综合、Altera 器件适配及为仿真、定时分析和器件编程产生输出文件。编译器首先提取项目设计文件之间的层次连接信息并检查基

本的设计输入错误，然后结合所有的设计文件生成能被高效处理的数据库。在编译过程中和编译后，用户都能在编译结果报告窗口看到结果。

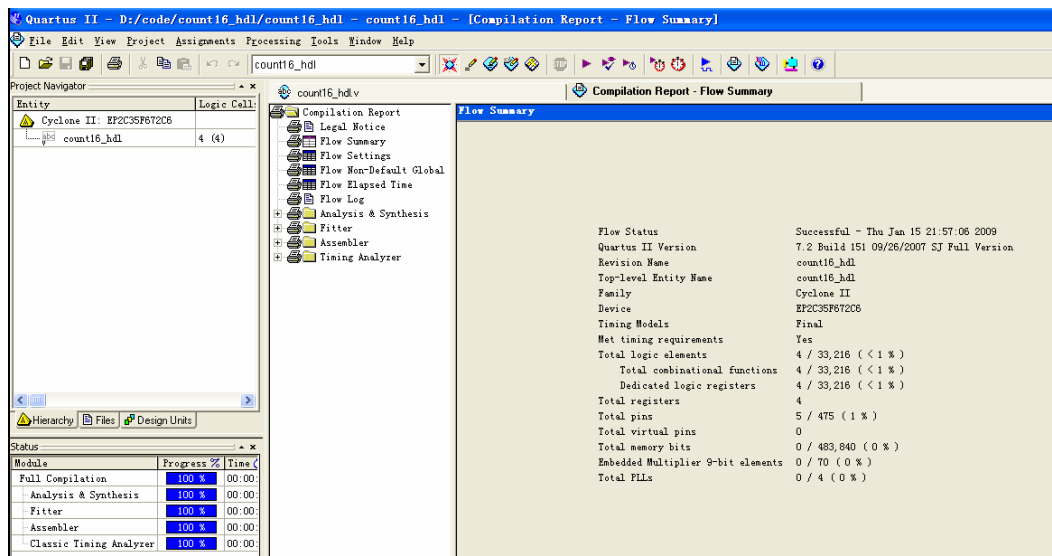


图 9.11 编译结果

(4) 根据硬件接口设计，对芯片引脚进行绑定。选择“Assignments”菜单下“Device”的“Pins”选项，进行引脚绑定，如图9.12所示。

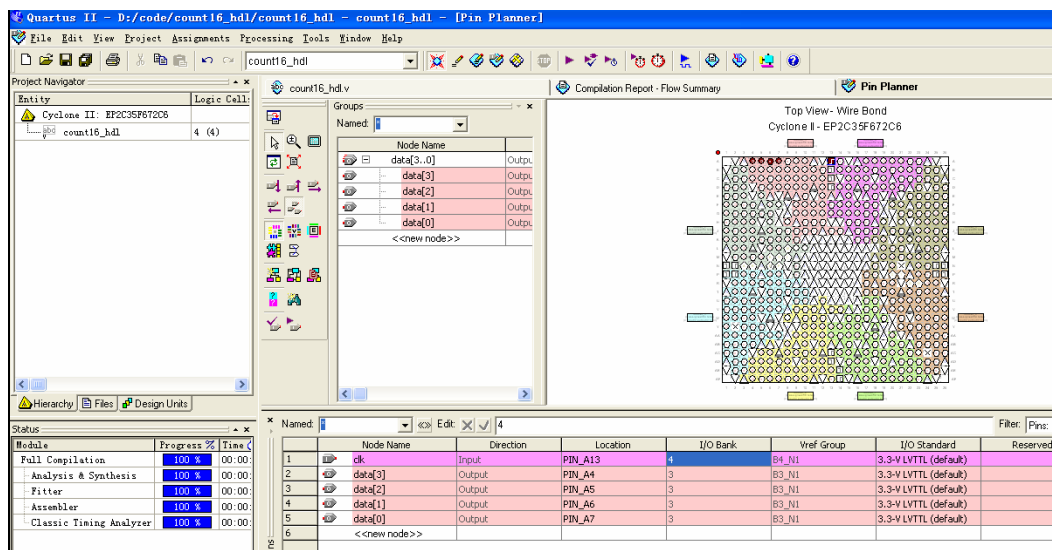


图 9.12 引脚分配界面

- (5) 完成所有引脚的分配，然后重新编译项目。
- (6) 对目标版适配下载，单击“ ”按钮，屏幕显示如图9.13所示。
- (7) 选择“Hardware Setup”，如图9.14所示。
- (8) 连接 USB 下载器。在硬件添加向导中，选择路径..\altera\72\quartus\drivers\usb-blaster\lx32，完成 USB 下载器的安装，在图9.15中选择“USB-Blaster”。

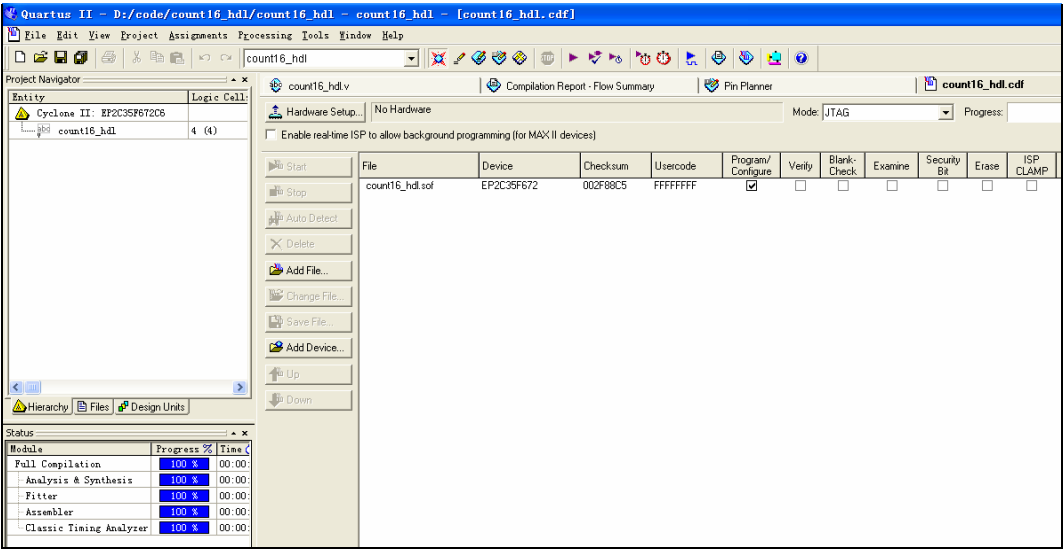


图 9.13 适配下载界面

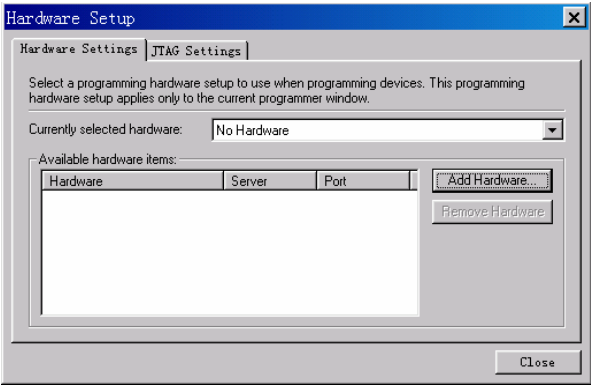


图 9.14 下载硬件设置

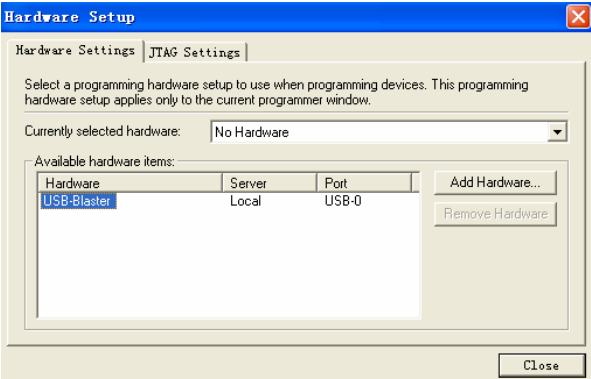


图 9.15 添加下载硬件

(9) 选择下载模式。在实验开发平台中，一般采用两种下载配置方式：AS 模式和 JTAG 模式。AS 模式对配置芯片下载，可以掉电保持，而 JTAG 模式对 FPGA 下载，掉电后 FPGA 信息丢失，每次上电都需要重新配置，如图 9.16 所示。

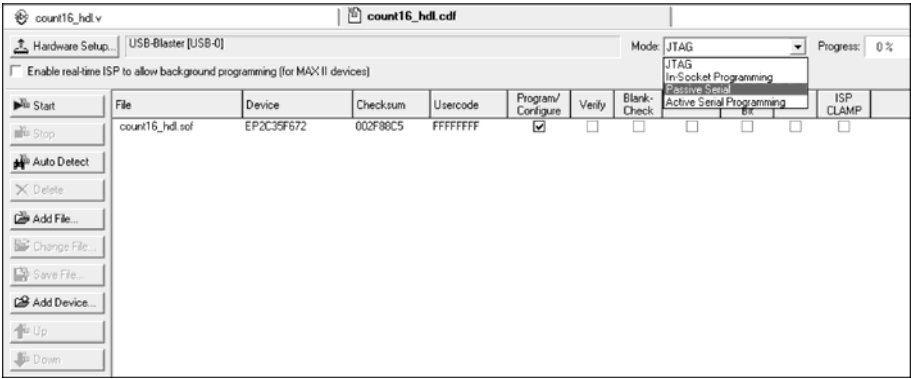


图 9.16 选择下载模式

(10) 选择下载文件和器件。JTAG 模式使用后缀为 sof 的文件，AS 模式使用后缀为 pof 的文件，进行下载配置操作。若使用 AS 模式，还需在“Assignments”菜单下“Device”→“Device& Pin Options”→“Configuration”中，选择对应的配置芯片，并再进行编译、下载，如图 9.17 所示。

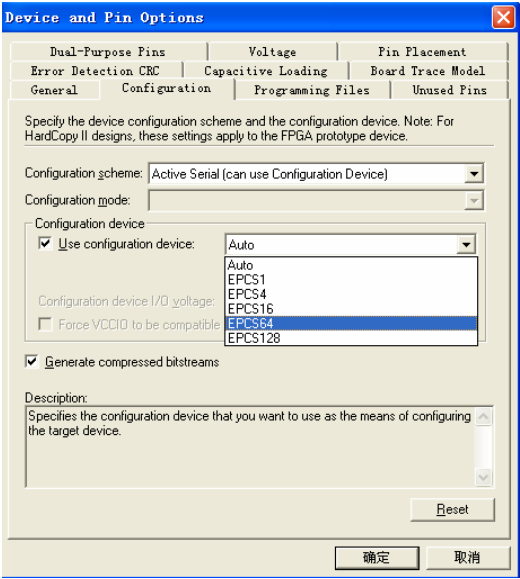


图 9.17 选择配置芯片

9.2.2 原理图设计输入

本节将介绍如何基于原理图利用“MegaWizard Plug-In Manager”来设计十六进制的计数器，并给出波形仿真的操作步骤。

1. 原理图输入

- (1) 按照 9.1 节的介绍，新建工程“count16_sch”，在主窗口单击“File”→“New”，选择“Block Diagram/Schematic File”选项，如图9.18所示。
- (2) 如图9.19所示，选择“Tools”→“MegaWizard Plug-In Manager”，进入如图9.20所示的界面。

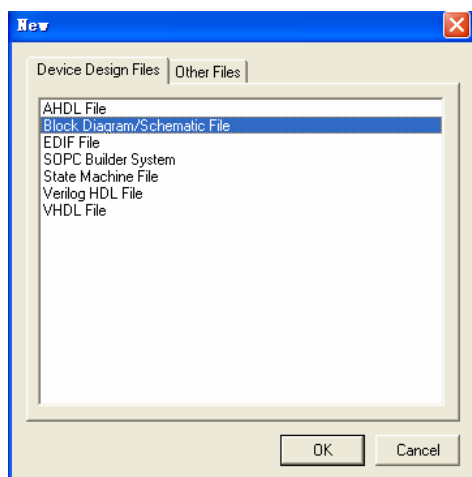


图 9.18 新建原理图文件

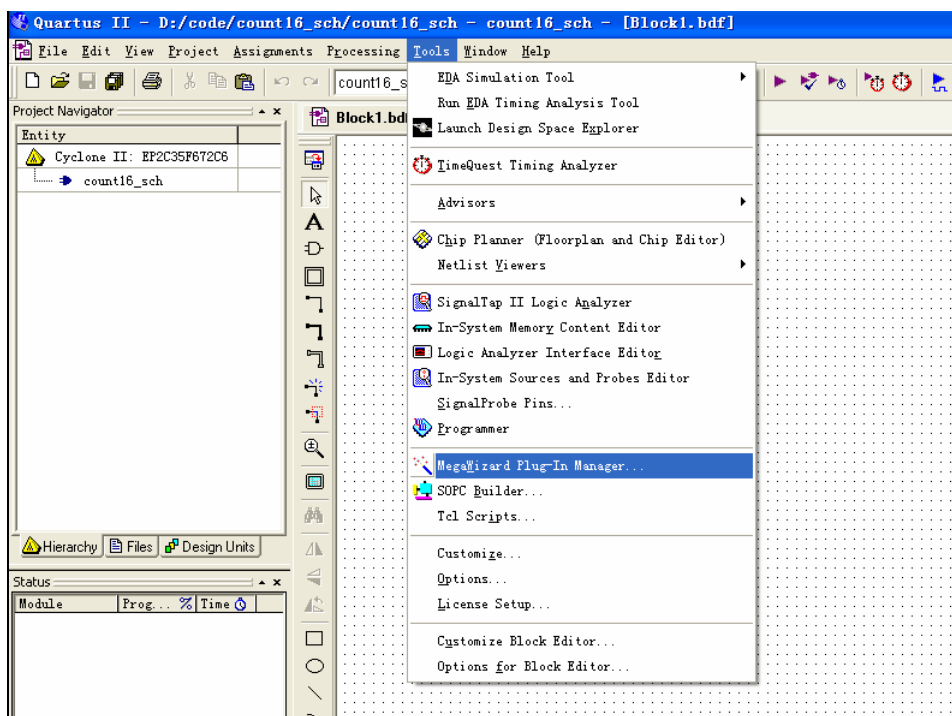


图 9.19 “MegaWizard Plug-In Manager” 选择界面

(3) 如图9.20所示, 在“Arithmetic”中选择“LPM_COUNTER”, 并在“what name do you want for output file”栏下面输入模块名, 在本例中命名为“count16”, 并单击“next”按钮。

(4) 在图9.21所示的界面中选择计数器的输出总线位宽为 4 bit, 计数器为加 1 计数。上述操作已经完成了一个基本的计数器设计, 因此可单击“finish”按钮, 完成设计。

(5) 双击.bdf 原理图文件, 如图9.22所示, 在弹出的“symbol”窗口中选择已配置完毕的“count16”模块, 释放鼠标, 将其放置到原理图文件中。

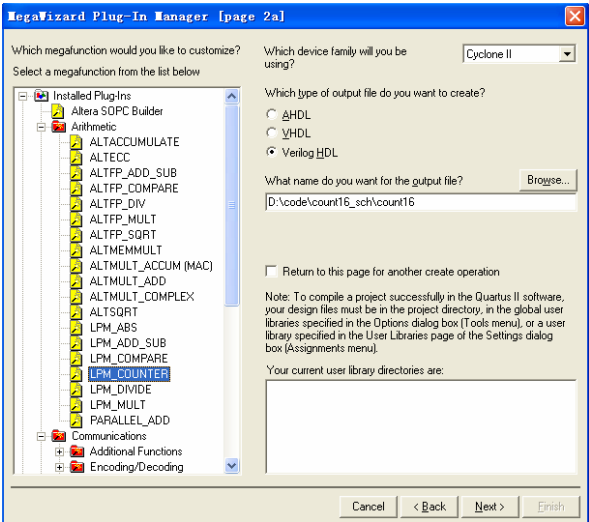


图 9.20 “MegaWizard Plug-In Manager” 设置界面 1

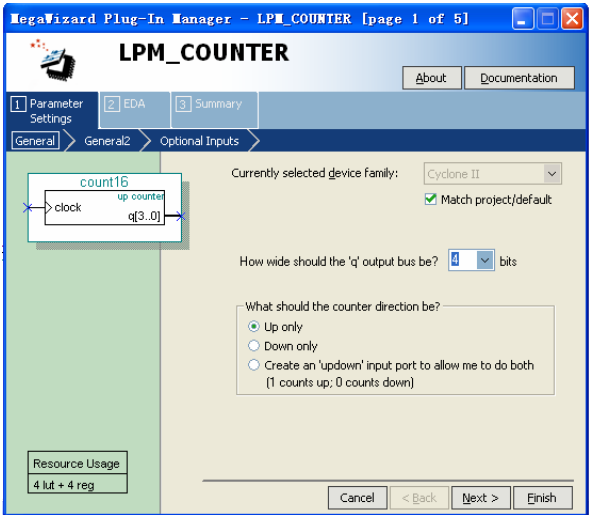


图 9.21 “MegaWizard Plug-In Manager” 设置界面 2

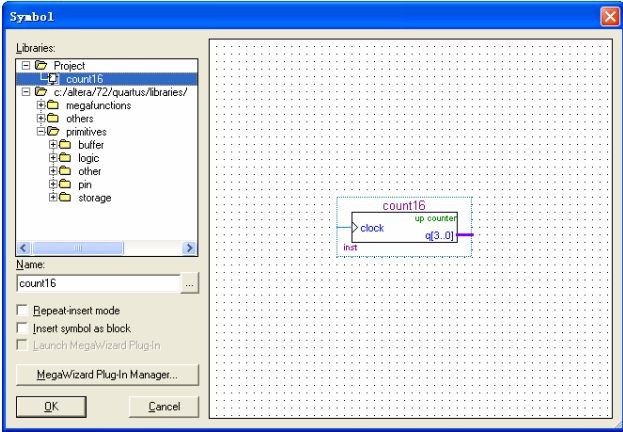


图 9.22 添加“count16”模块

(6) 双击.bdf 原理图文件, 在弹出的“symbol”窗口中输入“input”, 将其放置到原理图文件中, 改名为“clk”, 并将其与“count16”模块的输入端相连接; 同理, 加入“data[3..0]”输出端口并将其与“count16”模块的输出端相连接, 如图9.23所示, 将原理图文件保存为 count16_sch.bdf, 单击“”按钮, 完成编译。

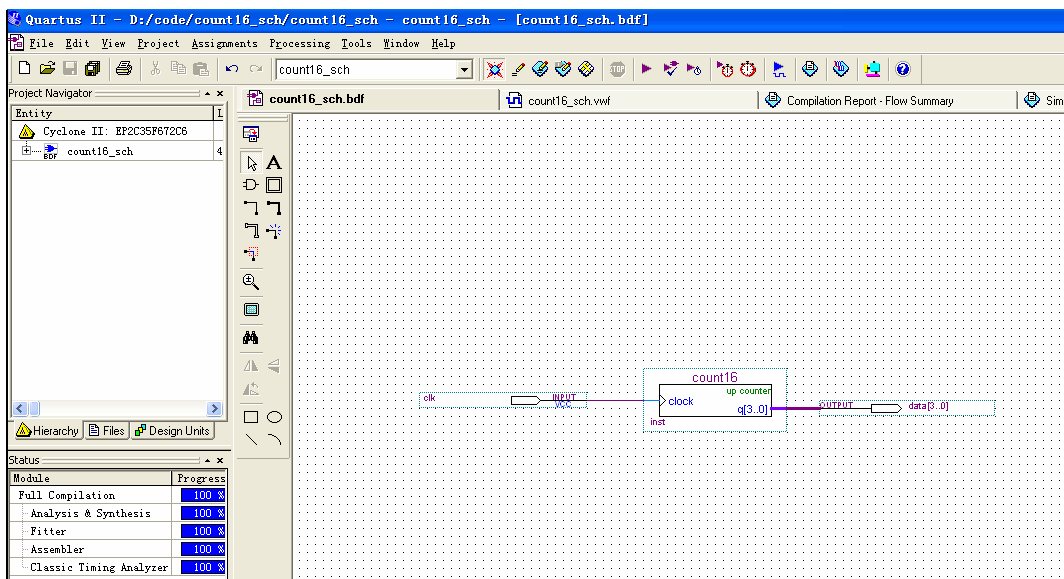


图 9.23 完成设计后的原理图

2. 波形仿真

以工程 count16_sch 为例, 介绍使用 Quartus II 软件自带的仿真器进行波形仿真的步骤。

(1) 在工程 count16_sch 中, 新建矢量波形文件, 如图9.24所示。

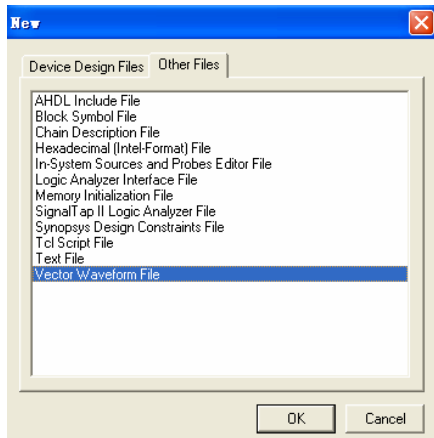


图 9.24 新建矢量波形文件

(2) 在建立的波形文件左侧一栏中, 单击鼠标右键, 在弹出的菜单中选择“Insert Node or Bus”, 如图9.25所示。

(3) 在出现的图9.26中选择“Node Finder”, 将打开“Node Finder”对话框, 本实验对输入、输出的引脚信号进行仿真, 所以在“Filter”中选择“Pins: all”, 单击“List”按钮, 如图9.27所示。

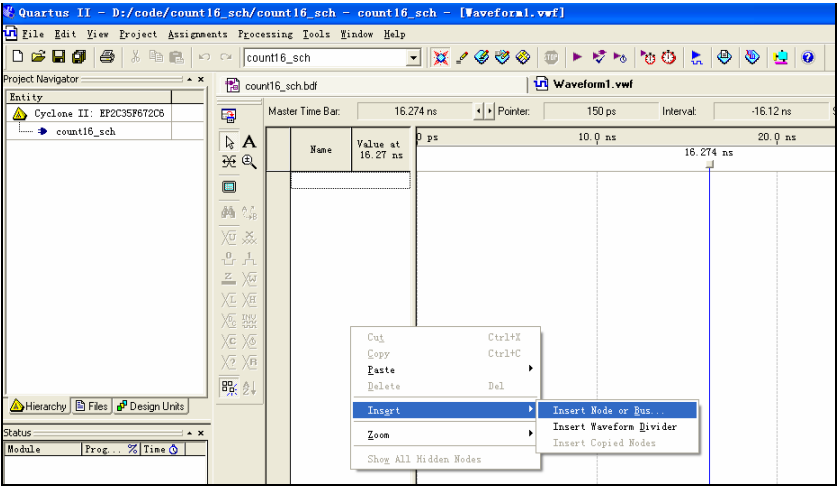


图 9.25 矢量波形文件节点加入

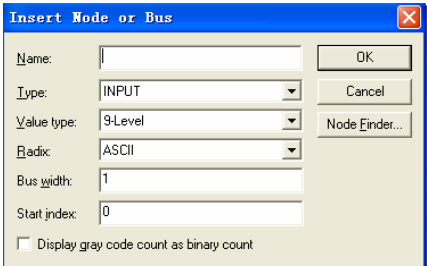


图 9.26 节点加入工具框

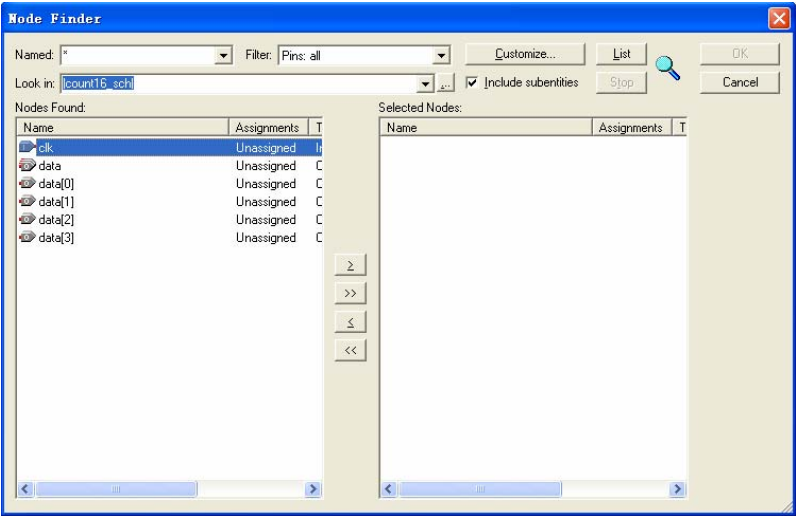


图 9.27 Node Finder 工具框

(4) 在图9.27左栏中选择需要进行仿真的端口，通过中间的按钮加入到右栏中，单击“OK”按钮，端口加入到波形文件中，如图9.28所示。

(5) 选择“clk”输入端，单击“”按钮，设置时钟频率为 100 MHz，保存为 count16_sch.vwf 文件，并单击“”按钮，开始仿真，结果如图9.29所示。

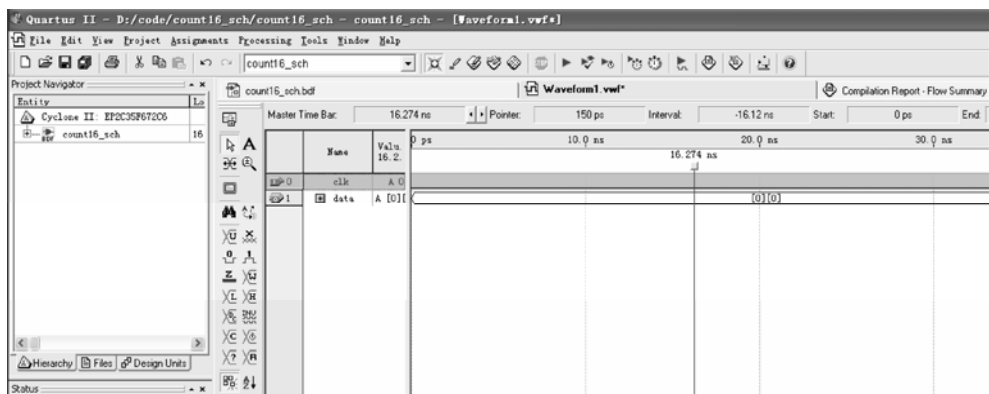


图 9.28 加入仿真节点后的波形图

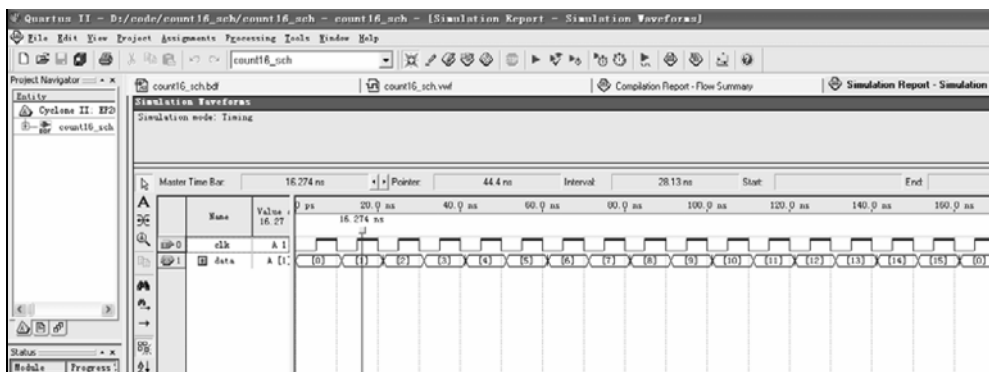


图 9.29 波形仿真结果

通过上述介绍,读者应当掌握基本的基于HDL和原理图设计的Quartus II操作步骤,并能够使用“Pin Planner”,“MegaWizard Plug-In Manager”,“Simulation”等工具辅助完成设计工作。其实,Quartus II软件中还有“SignalTap II Logic Analyzer”和“In-System Memory Content Editor”等实用工具,由于篇幅原因,这里不再介绍,感兴趣的读者可以到Altera网站 www.altera.com.cn 下载相关的Quartus II软件操作文档。

9.3 FPGA器件的使用

在现代电路设计过程中,越来越多地采用了EDA技术,利用可编程器件来实现传统方法中分立元器件所实现的组合电路和时序电路。可编程器件最终所构造的电路逻辑是利用专用FPGA设计软件,通过HDL硬件描述语言或原理图输入设计来实现的,由于该部分工作是基于软件的,因而这种设计方式具有可重复性且易于修改。在基于EDA技术的现代电路设计中,系统硬件平台的快速实现对于整个系统的成功设计具有至关重要的作用。所谓的硬件平台,是指针对训练的要求开发设计一种资源配置合理、性能稳定的实验装置,从而将应用系统设计工作的重心转移到软件设计上,硬件平台的设计应能满足训练题目多样化的要求,具有开放性、灵活性、便捷性。如何高效地构建一个基于可编程器件的硬件平台及如何选取可编程器件是初学者首先要解决的一个重要问题。

9.3.1 FPGA与CPLD器件的比较

目前在系统设计中,使用较多的可编程器件可分为两类:现场可编程门阵列(FPGA)和复杂可编程逻辑器件(CPLD)。以Altera公司的器件为例,一般来说,FPGA基于SRAM的架构,集成度高,

以查找表 LUT 为基本单元,内嵌有丰富的存储器,数字信号处理(DSP)块及各种高速接口等,并具有数据掉电易失性、需要有上电加载过程,在复杂算法、队列调度、数据处理、高性能设计、大容量缓存设计等领域中有广泛应用。CPLD 基于 E^2 PROM 工艺,集成度较低,以逻辑阵列块(LAB)为基本单元,具有非易失性,可以重复写入,在粘合逻辑、地址译码、简单控制、FPGA 加载等设计中有广泛应用。

虽然 FPGA 和 CPLD 都是可编程器件,但是由于它们在物理结构上的不同,因此具有各自的特点,其主要特点如下:

(1)CPLD 具有丰富的乘积项,从而适用于完成各种算法及组合逻辑;FPGA 具有丰富的触发器,从而适用于实现各种时序逻辑。

(2)CPLD 的连续式布线结构决定了其时序延时是均匀的和可预测的;FPGA 的分段式布线结构决定了其时序延时是不可预测的。

(3)CPLD 集成度较低,编程使用 E^2 PROM 或 FLASH 技术,无需外部存储器,使用简单;FPGA 集成度高,资源丰富,基于 SRAM 编程,编程信息断电消失,需要先将程序下载到外部存储器,上电后 FPGA 从外部存储器读取编程数据并写入 SRAM,从而完成 FPGA 配置,因此使用较为麻烦。

(4)CPLD 保密性好,能够对片内 E^2 PROM 或 FLASH 的配置程序进行加密操作;FPGA 保密性差,无法对片外存储器进行程序加密。

一般来说,初学者在进行系统设计时采用什么器件都可以。从设计和使用来看,两者的差别不是很大;从性价比等方面来看,建议采用 FPGA 器件,因为毕竟 FPGA 的资源丰富,有利于系统逻辑功能的不断增加和升级,至于 FPGA 器件程序断电不能保存的问题,可采用专用的、配套的用于 FPGA 配置的 E^2 PROM 器件,在系统上电后,FPGA 器件将自动从 E^2 PROM 中读取编程信息完成器件配置,无需人工干预。

9.3.2 FPGA器件的使用

如前所述,在采用 EDA 技术的现代电路设计中,由于整个系统核心器件 FPGA 的功能逻辑是利用专用 FPGA 开发工具,通过软件编程来实现的,因此前期基于 FPGA 的硬件平台建立是一个关键环节。如何快速搭建 FPGA 开发平台,需要解决下面两个问题。

(1)系统电源模块能否为 FPGA 器件提供准确的电压值和足够大的电功率。

以 Altera 公司 Cyclone II 系列 FPGA 为例,FPGA 工作的核心电压为 1.2 V,块电压可分别设置为 1.5 V、1.8 V、2.5 V 和 3.3 V,以满足不同接口电平的需要。Altera 公司的 Cyclone II 开发板上采用了 Micrel 公司的电源芯片 MIC49300 为其提供电压值 1.2 V、电流值 3 A 的核心电压,采用 MIC29502 通过电阻配置可为其提供电流值 5 A,电压值 1.5 V、1.8 V、2.5 V 和 3.3 V 可选的块电压。其实,Altera 公司针对每一个系列的器件都给出了设计参考文档,初学者在进行设计前可先查阅设计参考文档中有关器件核心电压、块电压、锁相环电压的相关介绍,然后参考该系列的评估板(原理图可在 Altera 网站上下载)中电源模块的设计,从而快速、准确地给出电源电路的设计,为系统平台的建立奠定一个良好的基础。

(2)FPGA 器件与配置器件的连接是否正确,FPGA 和配置器件的下载电路是否正确。

以 Altera 公司 Cyclone II 系列 FPGA 为例,可分别采用 JTAG、AS 和 PS 这三种方式来完成配置。在 JTAG 方式下,编程数据直接下载到 FPGA 的 SRAM 中,该操作速度快,但编程数据在 FPGA 断电后消失,因此适用于调试模式。在主动 AS 方式下,编程数据存放在 FPGA 片外的 EPCS 系列的 E^2 PROM 中,系统加电后,FPGA 器件引导配置操作过程,它控制着外部存储器和初始化过程,配置数据被同步在 DCLK 输入上,通过 DATA0 引脚 1 个时钟周期传送 1 位数据给 FPGA,从而完成 FPGA 的程序配置。由于 EPCS 系列 E^2 PROM 的价格比较便宜,因此在一般系统中常采用 AS 方式作为 FPGA 的配置方案。

在被动PS方式下,编程数据存放在FPGA片外的EPC系列的配置器件中,系统加电后,外部计算机或控制器控制配置过程,外部储存部件中的配置数据在DCLK上升沿锁存,通过DATA0引脚1个时钟周期传送1位数据给FPGA。但是由于EPC系列的器件相对较贵,因此PS方式并不推荐使用。

在系统设计中,一般采用JTAG模式+AS模式,这样可以用JTAG方式调试。当程序已经调试无误后,再用AS模式把程序烧到配置芯片中去。图9.30给出了Cyclone II系列的FPGA器件JTAG模式下的电路连接图,其中JTAG下载接口由4个必须的信号TDI, TDO, TMS和TCK及一个可选的TRST信号构成,当进行JTAG下载操作时,需将Altera公司提供的FPGA下载器与图9.30中的下载插座相连接。按照图9.16所示的操作,在Quartus II中选择JTAG下载模式,并单击“start”按钮完成下载。图9.31给出了Cyclone II器件在AS模式下与EPCS系列器件的连接方式及EPCS系列器件的在线下载接口示意图,同理在对EPCS系列器件进行在线配置时,需将Altera公司提供的FPGA下载器与图9.31中的下载插座相连接,按照9.2节介绍,在Quartus II中选择AS下载模式,并单击“start”按钮完成下载。

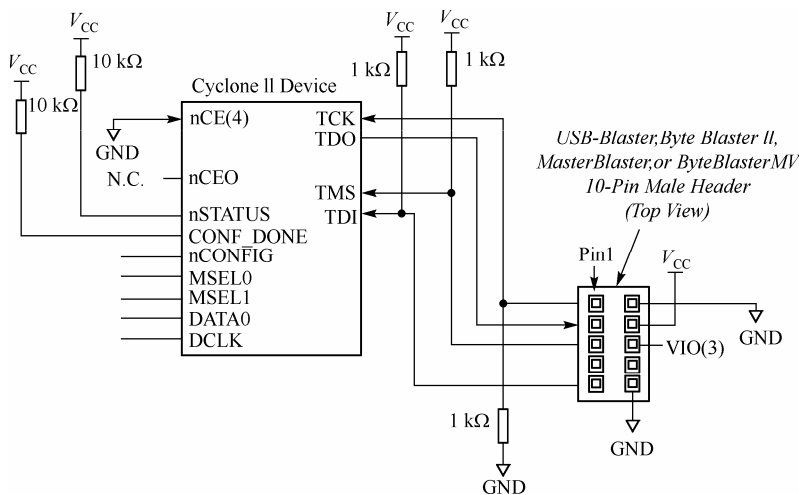


图 9.30 Cyclone II 系列的 FPGA 器件 JTAG 模式下的电路连接图

9.3.3 FPGA实验开发平台的介绍

由上述分析可知,FPGA 器件本身具有一定的灵活性,在实际系统设计中,所设计构造的逻辑电路通过 VHDL/Verilog 硬件描述语言进行描述,利用 Quartus II 软件下载到 FPGA 器件中,从而实现其功能。当系统功能改变时,若外设物理层接口不发生变化,那么只需要利用 VHDL/Verilog 硬件描述语言修改有关程序,并再次利用 Quartus II 软件下载新的配置文件到 FPGA 器件中,从而实现新的系统功能。因此,相比于传统的分立元器件的设计方式,基于 FPGA 的 EDA 设计是简捷、灵活的。

基于 FPGA 的 EDA 设计实践性较强,在理论学习的同时,必须加强实际动手能力的培养。考虑到 FPGA 器件本身具备的灵活性,一个高效的 FPGA 实验开发平台必须具有丰富的各种常用的外设物理层接口,使得使用者能够在开放的平台上针对不同的功能设计出不同的逻辑控制电路。基于上述考虑,作者研制出了一种能充分满足教学需要的 FPGA 实验开发平台,其结构布局图如图 9.32 所示。从图中可看出,整个实验平台不仅包含有四位一体的 LED 数码管、流水灯、LED 点阵、按键等人机交互外设,还包含 RS232 串行接口、USB2.0 接口等通信外设,以及 A/D 模数转换器、D/A 数模转换器、步进电机等其他外设。另外,在本实验平台中还提供了有 200 个输入、输出引脚的 IO 扩展槽,通过 I/O 扩展槽可以用堆栈方式连接功能扩展板(如存储器板),不仅能够进一步开展 SOPC, DSPBuilder 等

更高级的 EDA 实验，还可用于与自行开发的子系统进行连接，从而进行开放性的训练。本书后面章节中的绝大多数实验都是围绕着该实验平台来展开，并且在本平台上予以实现和验证的。

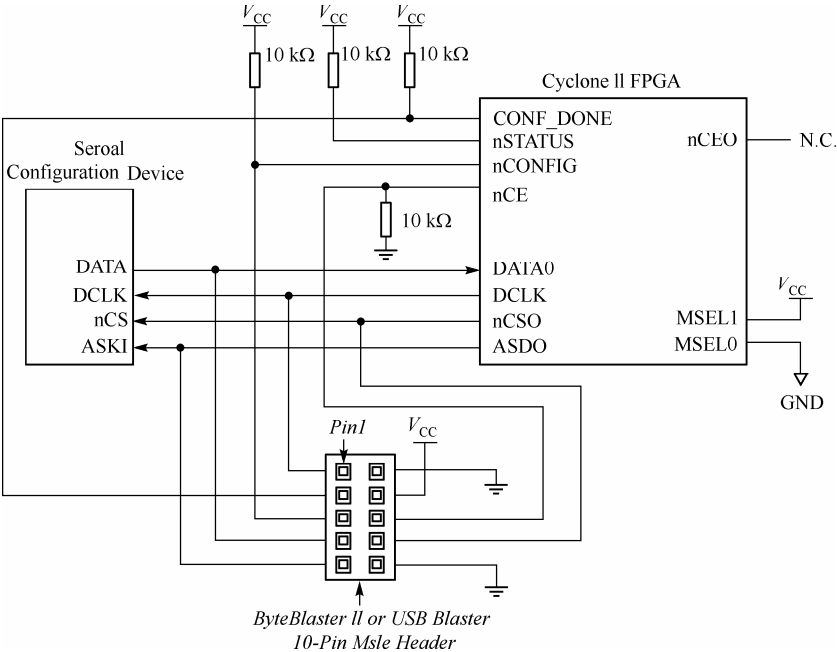


图 9.31 Cyclone II 器件在 AS 模式下与 EPCS 系列器件的连接方式及 EPCS 系列器件的在线下载接口示意图

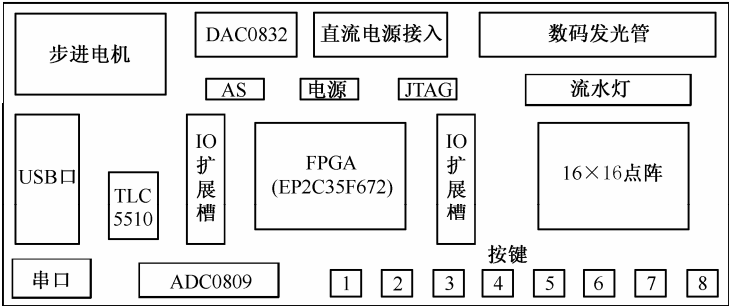


图 9.32 FPGA 实验开发平台的结构框图

9.4 本章小结

本章首先以 Altera 公司产品为例，在其开发软件 Quartus II 中使用硬件描述语言和原理图方式建立设计文件和进行仿真，详细说明了其设计流程，读者可以很容易通过有关的截图自己动手实践。为了加深读者对可编程逻辑器件内部逻辑结构的了解，本章从实际应用的角度出发比较了两类可编程逻辑器件——FPGA 和 CPLD 的不同特点及它们各自的应用领域。虽然本章的内容和训练比较简单，但它是后续章节的先导，其重要性不容忽视。本章在最后介绍了后续章节设计训练中将要使用的 FPGA 实验开发平台。

第 10 章 基于FPGA的外设控制电路的设计

10.1 引言

在传统的系统设计流程中，通常采用 MCS-51 单片机作为整个系统的主控制器，MCS-51 单片机实现整个系统外设的管理和控制功能。例如在 2.2 节介绍的通用单片机平台中，MCS-51 控制 ADC0809 完成数据的采集，控制 DAC0832 完成数模转换，扩展连接 8279 来实现 LED 数码管、键盘等外设的管理，完成人机交互。从工程设计的角度看，单片机的 I/O 端口数量较少，资源有限，而且单片机程序执行具有顺序性，因此单片机只能通过外接端口扩展芯片，并采用分时操作的方式来实现对整个外设的控制和管理。相对于单片机而言，FPGA 器件具有硬件速度更快、I/O 端口数目更多、并行处理能力更强的优点，能够利用丰富的可编程资源针对不同的外设构造相应的逻辑电路，从而实现对外设接口的管理及控制。

本章将分别介绍基于 FPGA 的 LED 数码管的显示、基于 FPGA 的 A/D 转换电路的控制与实现、基于 FPGA 的 D/A 转换电路的控制与实现、基于 FPGA 的 LED 点阵的控制与显示和基于 FPGA 的步进电机转速/方向控制的训练。通过这 5 个基础训练实例，希望读者能够了解并掌握常用外设电路的基本原理及工作时序，并能够通过 FPGA 设计构造组合时序电路及基本的状态机电路来完成对外设电路的控制，读者在学习的过程中应比较 FPGA 实现外设控制与单片机实现外设控制的区别，从而进一步体会 FPGA 器件的灵活性、并行性特点。

10.2 LED数码管的控制与显示

1. 设计任务

利用 FPGA 实现对单个 LED 数码管或者四位一体 LED 数码块(以下简称四位一体数码块)的控制与显示。

2. 设计基本要求

- (1) 设计 4 bit BCD 码到 7 段数码管的译码器，使得 7 段数码管能够显示十六进制数；
- (2) 基于动态扫描原理，利用 FPGA 实现四位一体数码块的显示控制。

本节首先介绍 LED 数码管和四位一体数码块的显示原理，并在熟悉和掌握显示原理的基础上分析和讨论利用 FPGA 实现 LED 显示控制的方法。其中，如何构造一个 4 bit BCD 码到 7 段数码管的译码器是设计的重点。在掌握了显示原理和设计方法后，实现显示模块的难度就不大了。该模块使用方便，在应用系统中需要显示功能时，可以直接移植使用。

10.2.1 LED数码管的显示原理

1. 单数码管的显示原理

以共阴极单数码管为例，如图10.1所示，其具有 9 个输入端口，其中 $\overline{\text{cs}}$ 端口为片选端，低电平有效；DP 端口为小数点输入端，高电平有效；a, b, c, d, e, f, g 端口分别对应数码管的 7 段段码，高电平

有效。当共阴极数码管的 $\overline{\text{cs}}$ 端输入低电平(‘0’)时,数码管选通,之后将根据 DP 和 a, b, c, d, e, f, g 端口输入的‘1’或‘0’电平来显示对应的字符。

2. 四位一体数码块显示原理

如图10.1所示,共阴极四位一体数码块共具有 12 个输入端口,其中每个数码管都共用小数点输入端 DP 及 7 段段码输入端 a, b, c, d, e, f, g, 且高电平有效。另外,数码块还具备 4 个片选输入端($\overline{\text{cs0}}$, $\overline{\text{cs1}}$, $\overline{\text{cs2}}$, $\overline{\text{cs3}}$), 分别对应每个数码管的选通, 低电平有效。基于第 2 章所介绍的动态显示原理, 依次扫描选中每个数码管的同时(每次仅有一个数码管被选中), 输入待显示图形的 7 段段码及小数点, 即可在四位一体数码块上得到需显示的图形。以显示“1234”为例, 其操作步骤如下:

(1) 仅选中 $\overline{\text{cs0}}$ 片选端, 输入‘4’对应的 7 段段码;

(2) 仅选中 $\overline{\text{cs1}}$ 片选端, 输入‘3’对应的 7 段段码;

(3) 仅选中 $\overline{\text{cs2}}$ 片选端, 输入‘2’对应的 7 段段码;

(4) 仅选中 $\overline{\text{cs3}}$ 片选端, 输入‘1’对应的 7 段段码;

(5) 重复步骤(1)~(4), 依次循环扫描, 只要扫描频率大于一定值, 利用人眼的视觉惰性, 就可在数码块上清晰地看到“1234”。

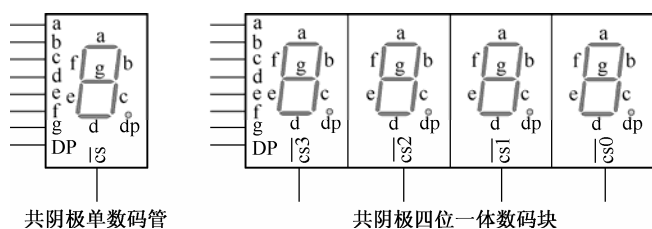


图 10.1 数码管和数码块示意图

10.2.2 FPGA实现LED显示控制

在数字逻辑电路中, 可用 74LS47(共阳极)/74LS48(共阴极)译码驱动电路来控制 LED 显示。在 FPGA 中, 可用硬件描述语言设计一个译码器, 即将要显示的字符译成 7 段码, 以实现单个数码管的显示。对于多个 LED 数码管, FPGA 实现显示控制时, 还需要考虑到片选信号的循环选通, 利用人眼的视觉暂留特性, 达到多个数码管同时显示的功能, 这其中控制好数码管之间的循环选通时间(分时显示刷新速率)是相当重要的。

10.2.3 模块功能实现

1. LED数码管译码模块

LED 数码管译码(decode)模块的主要功能是将 4 bit BCD 码转换成数码管的段码, 这是一个典型的译码电路。采用 Verilog 硬件描述语言的 case 语句来描述该项功能, 在实现过程中应注意数码管显示的字符与数码管段码的对应关系。其源代码如下:

```
module decode(sel_data, seg_data);
input[3:0] sel_data;    //输入的 4 位 BCD 码
output[7:0] seg_data;   //输出的数码管段码及小数点
reg[7:0] seg_data;      //seg_data={a, b, c, d, e, f, g, dp}

always @(sel_data)
```



```

begin
  case(sel_data)
    4'd0:seg_data=8'b11111100;
    4'd1:seg_data=8'b01100000;
    4'd2:seg_data=8'b11011010;
    4'd3:seg_data=8'b11110010;
    4'd4:seg_data=8'b01100110;
    4'd5:seg_data=8'b10110110;
    4'd6:seg_data=8'b10111110;
    4'd7:seg_data=8'b11100000;
    4'd8:seg_data=8'b11111110;
    4'd9:seg_data=8'b11110110;
    4'd10:seg_data=8'b11101110;
    4'd11:seg_data=8'b00111110;
    4'd12:seg_data=8'b10011100;
    4'd13:seg_data=8'b01111010;
    4'd14:seg_data=8'b10011110;
    4'd15:seg_data=8'b10001110;
    default:seg_data=8'b00000000;
  endcase
end

endmodule

```

2. 四位一体数码块显示模块

基于数码块的动态显示原理，四位一体数码块显示模块的实现需要解决下列问题：

- (1) 构造一个依次循环扫描模块，从而依次选通数码块中的每一个数码管；
- (2) 在选通每个数码管的同时，必须在数码块的段数据端口输入对应的段数据。

根据上述分析，四位一体数码块显示模块的组成框图如图 10.2 所示。cnt4 模块是一个 0~3 的加 1 循环计数器，其计数值 $q[1:0]$ 输入到 2 线-4 线译码器 74139M 的输入端，产生依次循环的 $\overline{cs0}$ 、 $\overline{cs1}$ 、 $\overline{cs2}$ 、 $\overline{cs3}$ 片选信号，并将这组片选信号接入到四位一体数码管的 4 个片选输入端； $data0x[3:0]$ 、 $data1x[3:0]$ 、 $data2x[3:0]$ 、 $data3x[3:0]$ 输入端分别给出四位一体数码块上每个数码管待显示字符的 BCD 码。由于 muxer 多路选择器模块的选择端与计数值 $q[1:0]$ 相连接，因此 muxer 多路选择器的输出为 2 线-4 线译码器 74139M 所选通的数码管上待显示字符的 BCD 码。最后 decode 模块将完成 BCD 码到数码管段码的转换，并将 segment[7:0] 段码值与四位一体数码管的 7 段数据端口及小数点端口相连接。在实际系统设计中，由于 FPGA 输出端口 segment[7:0] 的驱动能力有限，可利用三极管或者 74573 等集成电路来提高端口的驱动能力，从而增加数码块的显示亮度，达到更好的显示效果。

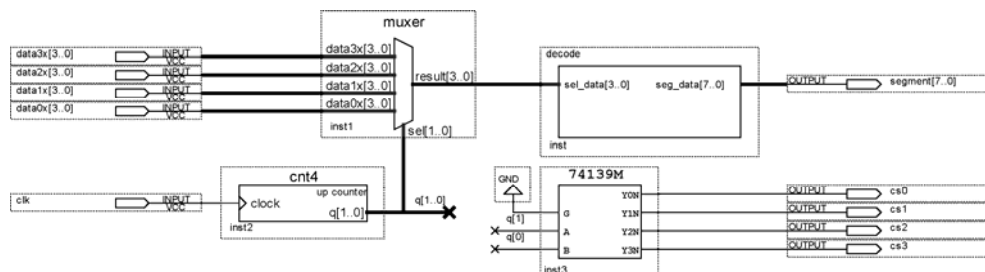


图 10.2 四位一体数码块显示模块的组成框图

读者可以参考第 2 章的相关内容确定显示时钟 `clk` 的速率等参数,还应当仔细体会本节的基于 FPGA 的 LED 显示设计和第 2 章中的基于 MCU 的 LED 显示设计之间的区别。

10.3 A/D转换电路的控制与实现

1. 设计任务

利用 FPGA 实现对 A/D 转换芯片的控制,完成一路模拟数据的采集,并将模数转换结果显示到四位一体数码块上。

2. 设计基本要求

- (1) 利用状态机设计 ADC0809 转换芯片的控制模块,完成一路模拟数据的采集;
- (2) 基于 10.2 节所介绍的四位一体数码块显示模块,将模数转换结果显示到数码块上。

要实现对某个芯片或电路的控制,首先必须了解和掌握该芯片或电路的工作时序,这样才能构造出满足芯片或电路工作时序的控制模块。因此,本训练的关键点是如何根据 A/D 转换芯片的工作时序来设计状态机,从而实现 FPGA 对 A/D 转换芯片的控制。本节首先介绍 A/D 转换芯片 ADC0809 的工作时序;然后在分析并掌握该时序的基础上,利用有限状态机来实现 ADC0809 控制模块的时序逻辑电路设计;最后将利用 10.2 节设计的四位一体数码块的显示模块,实现模数转换结果的显示。

10.3.1 ADC0809 模数转换芯片的工作时序

ADC0809 芯片的主要性能参数介绍如下:ADC0809 为 8 通道,8bit 的 A/D 转换芯片;是一个逐次逼近型的 A/D 转换器;外部供给基准电压;单通道转换时间为 $116\mu\text{s}$;带有三态输出锁存器,转换结束时,可由 CPU 打开三态门,读出 8 位的转换结果;有 8 个模拟量的输入端,可引入 8 路待转换的模拟量;芯片采用双列直插式封装,共有 28 个引脚。其工作时序如图 10.3 所示,各引脚的功能分 4 组简述如下。

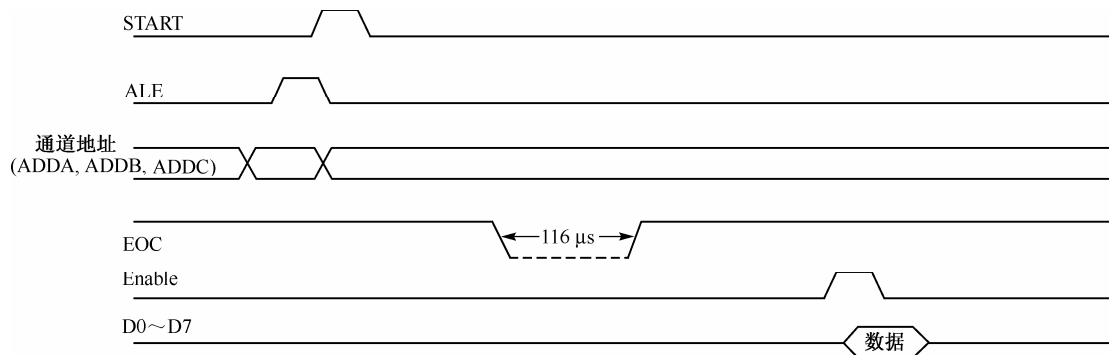


图 10.3 ADC0809 的工作时序图

(1) 模拟信号输入 `IN0~IN7`: `IN0~IN7` 为 8 路模拟电压输入线,加在模拟开关上,工作时采用时分的方式,轮流进行 A/D 转换。

(2) 地址输入和控制线(4 条): `ADDA`, `ADDB` 和 `ADDC` 为地址输入线,用于选择 `IN0~IN7` 上哪一路模拟电压送给比较器进行 A/D 转换; `ALE` 为地址锁存允许输入线,高电平有效。当 `ALE` 线为高电平时, `ADDA`, `ADDB` 和 `ADDC` 三条地址线上地址信号得以锁存,经译码器控制 8 路模拟开关通路工作。

(3) 数字量输出及控制线(11 条): `START` 为“启动脉冲”输入线,上升沿清零,下降沿启动 ADC0809

工作。EOC 为转换结束输出线, 该线高电平表示 A/D 转换已结束, 数字量已锁入“三态输出锁存器”。D0~D7 为数字量输出线, D7 为最高位。ENABLE 为“输出允许”线, 高电平时能使 D0~D7 引脚上输出转换后的数字量。

(4) 电源线及其他引线(5 条): CLOCK 为时钟输入线, 用于为 ADC0809 提供逐次比较所需的时钟信号, 一般为 640 kHz 的时钟脉冲。VCC 为+5 V 电源输入线, GND 为地线。+VREF 和-VREF 为参考电压输入线, 用于给电阻网络提供标准电压。一般+VREF 常和 VDD 相连, -VREF 常接地。

设计任务要求的是一路模拟数据的采集, 则可在 ADC0809 的地址线 ADDA, ADDB 和 ADDC 上给出相应输入通道的二进制编码, 任意选择一条指定的输入通道完成一路模拟输入信号的模数转换功能。参照 ADC0809 的工作时序, 控制模块控制 ADC0809 完成模数转换的工作流程如下:

(1) 控制模块同时产生“START”转换启动脉冲和“ALE”地址输入控制锁存脉冲, “START”下降沿启动 ADC0809 工作, “ALE”高电平锁存地址线上地址信号;

(2) 控制模块检测 ADC0809 的“EOC”转换结束标识线, 若“EOC”变为低电平, 则表示 ADC0809 正在进行模数转换操作;

(3) 控制模块检测到 ADC0809 的“EOC”转换结束标识线从低电平变为高电平, 则表示 ADC0809 模数转换结束;

(4) 控制模块输出“ENABLE”信号, 高电平有效, 并读取 ADC0809 数据线的数字值, 完成一次模数转换操作。

(5) 跳到步骤(1)启动下一次转换工作。

10.3.2 模块功能实现

利用 Verilog 语言设计 ADC0809 转换电路控制器的关键在于如何将 ADC0809 的工作时序抽象成状态转移图, 从而由有限状态机来实现。

有限状态机是由寄存器组和组合逻辑构成的硬件时序电路, 其状态只能在同一时钟跳变的情况下才能从一个状态转向另一个状态。有限状态机设计的一般步骤包括: 逻辑抽象, 得出状态转换图; 状态化简; 状态分配; 选定触发器的类型并求出状态方程、驱动方程和输出方程, 以及按照方程得出逻辑图。

参考 ADC0809 转换时序图, 依据 10.3.1 节的分析, 可得到如图 10.4 所示的 ADC0809 转换电路控制器的状态转移图, 状态机的每次跳变不但取决于各个输入值, 还取决于当前状态。

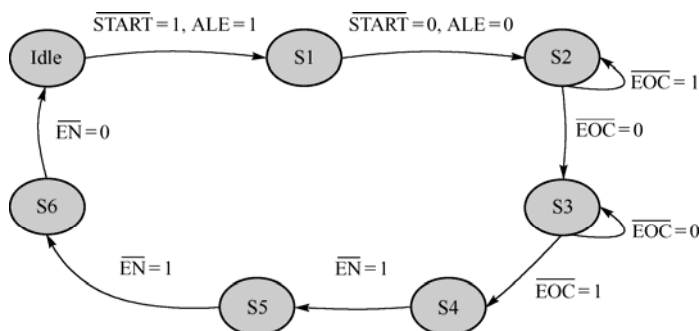


图 10.4 ADC0809 转换电路控制器的状态转移图

(1) Idle: 它是系统上电复位后的状态, 系统控制模块在输出一个时钟周期的“START”和“ALE”高电平后, 状态机跳转到 S1 状态;

(2) S1: 系统控制模块输出一个时钟周期的“START”和“ALE”低电平, “START”下降沿启动 ADC0809 工作, “ALE”高电平锁存地址线上地址信号, 状态机跳转到 S2 状态;

(3) S2: 系统控制模块检测 ADC0809 的“EOC”转换结束标识线, 若“EOC”变为低电平, 则表示 ADC0809 正在进行模数转换操作, 状态机进入 S3 状态, 否则状态机停留在 S2 状态;

(4) S3: 系统控制模块检测到 ADC0809 的“EOC”转换结束标识线从低电平变为高电平, 则表示 ADC0809 模数转换结束, 状态机跳转到 S4 状态, 否则状态机停留在 S3 状态;

(5) S4: 系统控制模块输出一个时钟周期的“EN”高电平, 状态机跳转到 S5 状态;

(6) S5: 系统控制模块输出一个时钟周期的“EN”高电平, 同时将 AD 转换后的数据 data_ad 输入给 data_h, 状态机跳转到 S6 状态;

(7) S6: 系统控制模块输出一个时钟周期的“EN”低电平, 状态机跳转到 Idle 状态, 开启下一次 A/D 采样状态。

该状态转移图(参见图 10.4)采用 Verilog 描述代码如下:

```
module ad_convert(rst,clk,eoc,data_ad,start,ale,enable,data_h);
//=====
input rst,clk,eoc;
input[7:0] data_ad;
output start,ale,enable;
output[7:0] data_h;
//=====
reg start,ale,enable;
reg[7:0] data_h;
reg[2:0] state;
parameter idle = 3'd0,s1=3'd1,s2=3'd2,s3=3'd3,s4=3'd4,s5=3'd5,s6=3'd6;
//=====
always @(posedge clk or negedge rst)
begin
    if(!rst)
        begin
            start <= 0;
            ale <= 0;
            enable <= 0;
            state <= idle;
        end
    else
        case(state)
            idle:begin
                start<=1;
                ale <= 1;
                enable<=0;
                state<=s1;
            end
            s1:begin
                start<=0;
                ale <= 0;
                enable<=0;
                state<=s2;
            end
            s2:begin
                if(eoc==0)
```

```

        state<=s3;
    else;
    end
end
s3:begin
    if(eoc==1)
        state<=s4;
    else;
    end
end
s4:begin
    enable<=1;
    state<=s5;
end
end
s5:begin
    enable<=1;
    data_h<=data_ad;
    state<=s6;
end
end
s6:begin
    enable<=0;
    state<=idle;
end
end
default:begin
    state<=idle;
end
end
endcase
end
endmodule

```

10.3.3 系统整体设计

在本系统设计过程中,可在顶层设计时采用原理图的设计方法,更加直观地显示各模块之间的连接关系,如图10.5所示,将 ad_convert 模块与四位一体数码块的显示模块相连接,即可完成 A/D 转换电路的控制与显示的设计。

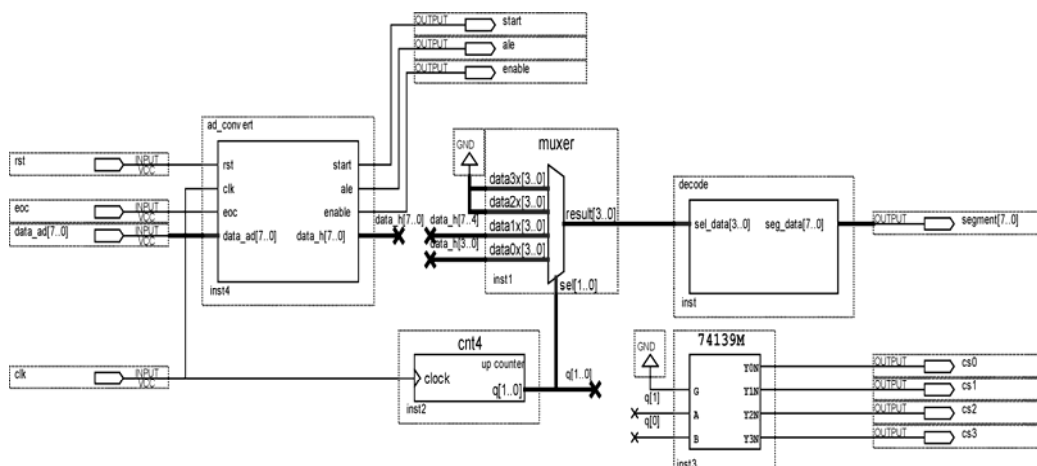


图 10.5 数据采集与显示系统的顶层设计图

10.4 D/A转换电路的控制与实现

1. 设计任务

利用 FPGA 实现对数模转换器 DAC0832 的控制, 使其能够输出正弦波、方波、锯齿波等信号。

2. 设计基本要求

(1) 掌握利用参数化模块库 (Library of Parameterized Modules, LPM) 设计 ROM 的方法, 并能够编辑 ROM 中存储的数据;

(2) 实现 FPGA 对数模转换器 DAC0832 的控制, 使其能够输出正弦波、方波、锯齿波等信号。

要利用 FPGA 实现对数模转换器 DAC0832 的控制, 首先应分析 DAC0832 的工作时序, FPGA 将根据时序关系来实现对 DAC0832 的控制。此外, 要输出正弦波、方波、锯齿波等信号, 其相应的波形文件需要预先存储起来。许多系列的 FPGA 芯片内嵌了存储阵列, 因此在 FPGA 中实现各种存储器, 如单/双端口 RAM、单/双端口 ROM、先进先出存储器 FIFO 等非常方便, 而且具有诸多优点。FPGA 中构造存储器主要有两种方法: 一是通过硬件描述语言如 VHDL, AHDL, Verilog HDL 等编程实现; 二是调用软件自带的库函数实现。调用库函数方法构造存储器与硬件描述语言输入方式相比, 更为方便、灵活、快捷和可靠。本训练的波形文件需要存储在 ROM 中, 因此利用 Atera 提供的参数化模块库 (LPM) 来完成 ROM 的构建, 也是本训练的重要环节之一。

10.4.1 D/A芯片的时序和控制方式分析

DAC0832 是一种电流型的并行数据位宽 8 bit 的数模转换芯片, 由于其操作简单、数据建立时间短 ($1\mu\text{s}$), 因此广泛应用于工业控制等领域。图10.6给出了 DAC0832 的结构框图, 其中,

(1) 数字量输入信号 $\text{DI7} \sim \text{DI0}$: DI0 为最低位, DI7 为最高位。

(2) 输入锁存允许信号 ILE : 高电平有效; 片选信号 $\overline{\text{CS}}$: 低电平有效; 写信号 1 信号 $\overline{\text{WR1}}$: 低电平有效。当 ILE , $\overline{\text{CS}}$ 和 $\overline{\text{WR1}}$ 同时有效时, 输入寄存器的输出随输入而变化。

(3) 转移控制信号: $\overline{\text{XFER}}$, 低电平有效; 写信号 2 信号 $\overline{\text{WR2}}$: 低电平有效。当 $\overline{\text{XFER}}$ 和 $\overline{\text{WR2}}$ 同时有效时, DAC 寄存器的输出随输入而变化。

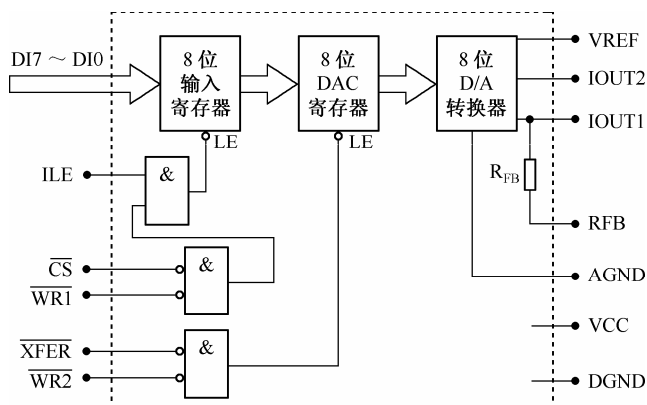


图 10.6 DAC0832 的结构框图

从上述分析可知, DAC0832 一般有三种连接方式: 直通方式 (两个寄存器均工作于直通状态)、单缓冲方式 (一个寄存器工作于直通状态, 另一个工作于受控锁存器状态) 和双缓冲方式 (两个寄存器均工

作于受控锁存器状态)。在单片机系统里,由于多个设备需要复用单片机的数据总线,因此往往采用如图10.7所示的单缓冲方式。

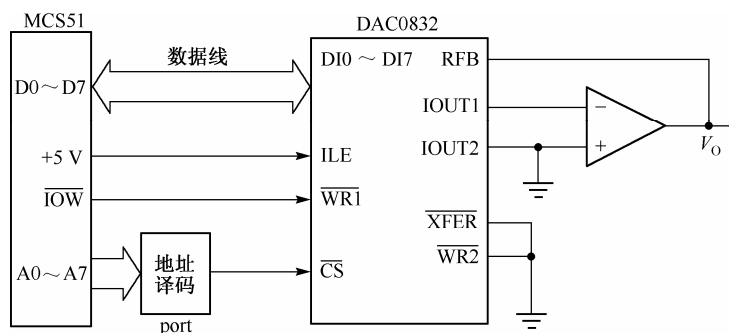


图 10.7 DAC0832 的单缓冲方式

在 FPGA 系统中,由于 FPGA 器件的 I/O 端口资源充足,且为了充分发挥 FPGA 的并行处理能力,可将 FPGA 的 I/O 端口直接与 DAC0832 相连接,并将 DAC 的 ILE 接高电平, $\overline{CS}$, $\overline{WR1}$, $\overline{WR2}$ 和 $\overline{XFER}$ 接低电平,使得 DAC0832 工作在直通模式下,此时 FPGA 将不停地将 8 bit 的变化数据输入到 DAC0832 的 8 位数字量输入线上, DAC0832 将输出相应的变化模拟量。

10.4.2 模块功能实现

在本设计中, DAC0832 输出模拟波形所对应的数字量将存放到 ROM 中。下面将介绍如何利用 Altera 提供的参数化模块库 (LPM) 来创建 ROM, 操作步骤如下。

步骤 1: 打开 MegaWizard Plug-IN Manager 工具, 如图10.8所示。

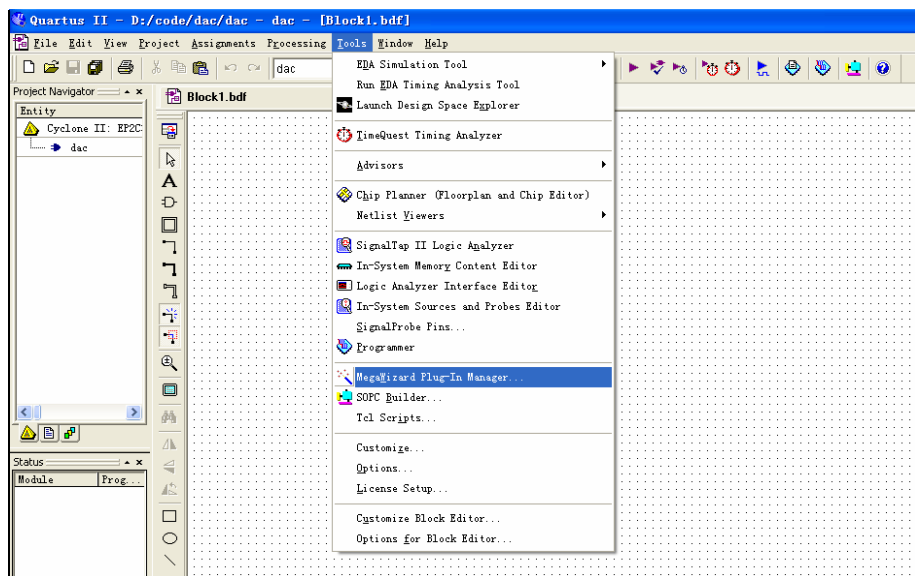


图 10.8 利用 LPM 创建 ROM 步骤 1

步骤 2: 选择生成一个定制项目, 如图10.9所示。

步骤 3: 新建文件为 LPM_ROM 类型, 如图10.10所示。

步骤 4: 选择构造 ROM 的参数要求, 如图10.11所示。

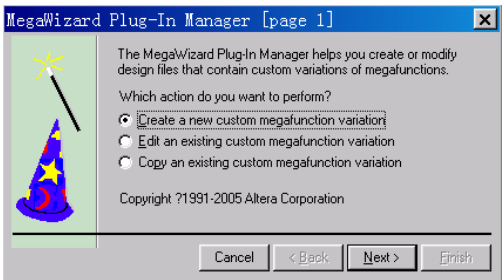


图 10.9 利用 LPM 创建 ROM 步骤 2

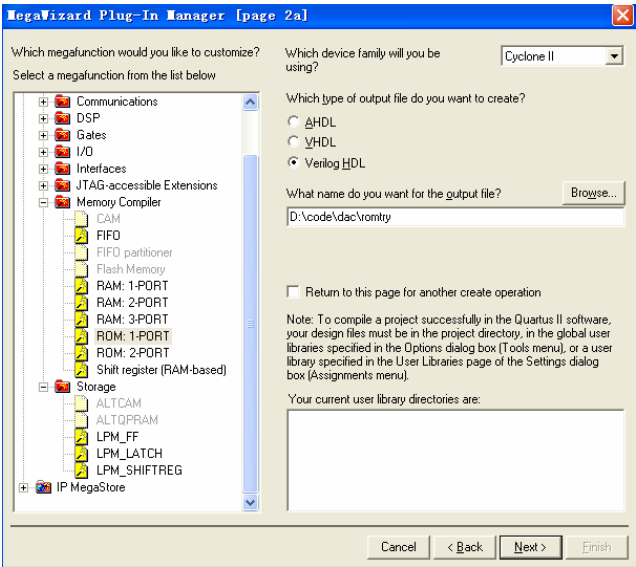


图 10.10 利用 LPM 创建 ROM 步骤 3

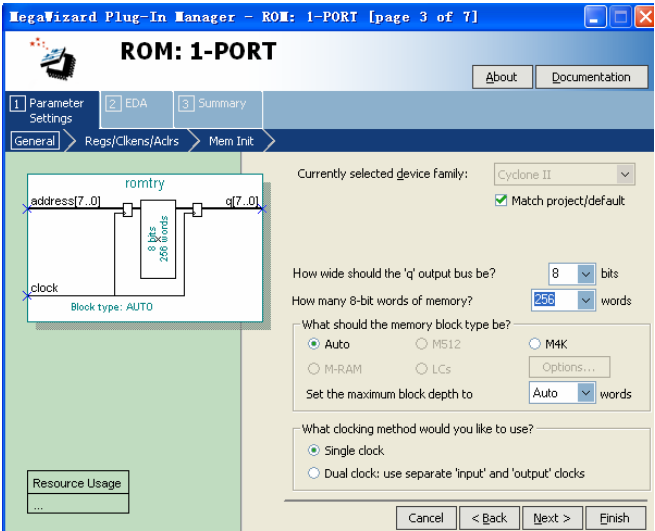


图 10.11 利用 LPM 创建 ROM 步骤 4

步骤 5: 选择输出是否锁存, 如图10.12所示。
步骤 6: 输入 ROM 的初始化文件名称 romboxin.mif, 如图10.13所示。

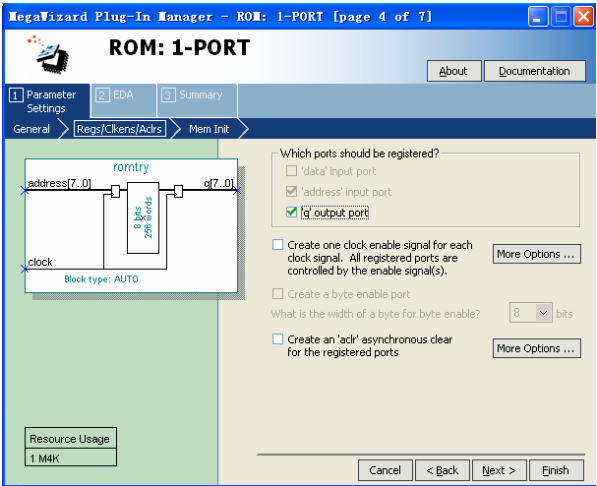


图 10.12 利用 LPM 创建 ROM 步骤 5

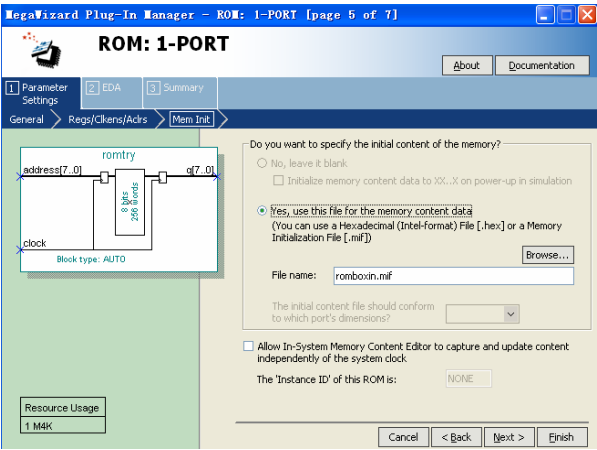


图 10.13 利用 LPM 创建 ROM 步骤 6

步骤 7: 结束向导, 如图10.14所示。

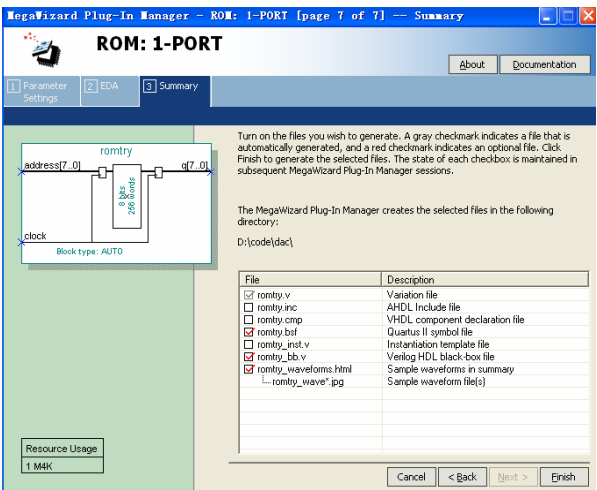


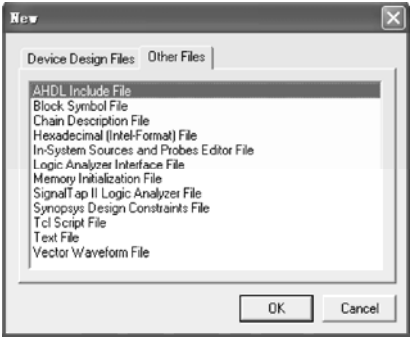
图 10.14 利用 LPM 创建 ROM 步骤 7

存储器初始配置文件的格式可以是 HEX 或 MIF 两种类型。本设计采样 MIF 类型文件，以产生方波初始信号为例，其格式如下：

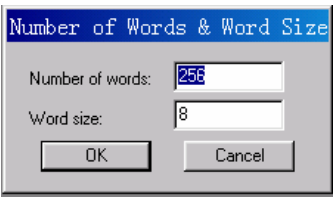
```
DEPTH = 256; % Memory depth and width are required %
WIDTH = 8;  % Enter a decimal number %
ADDRESS_RADIX = HEX; % Address and value radices are optional %
DATA_RADIX = HEX;    % Enter BIN, DEC, HEX, or OCT; unless    %
                    % otherwise specified, radices = HEX %
-- Specify values for addresses, which can be single address or range
CONTENT
BEGIN
[00..70]:    00;
[80..ff]:    ff;
END ;
```

Quartus II 中提供了初始文件创建和修改功能，其操作步骤如下：

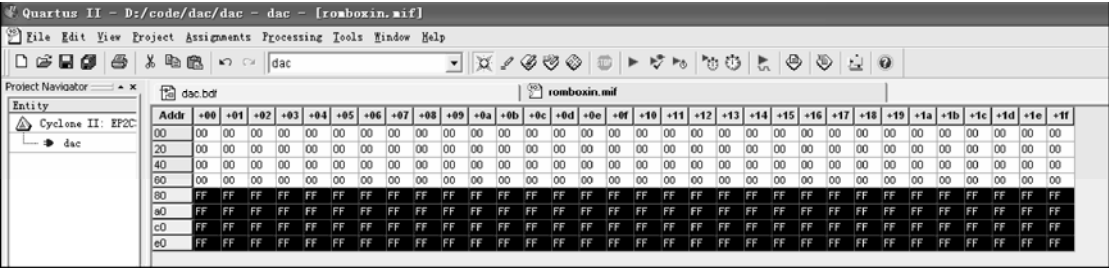
- 步骤 1：新建文件，如图10.15(a)所示，选择文件类型(选择 Memory Initialization File)。
- 步骤 2：选择 ROM 字宽度和深度，如图10.15(b)所示。
- 步骤 3：填充 ROM 单元，如图10.15(c)所示。



(a)



(b)



(c)

图 10.15 初始文件创建和修改

10.4.3 系统整体设计

在本系统设计过程中，可在顶层设计时采用原理图的设计方法，如图10.16所示，将所产生的 ROM 模块与利用 LPM 所产生的 cnt256 模块(0~255 的加 1 循环计数器)相连接，即可完成 D/A 转换电路的控制。通过替换配置文件改变波形生成模块中的初始数据，可得到锯齿波、方波、三角波和正弦波等多种波形。

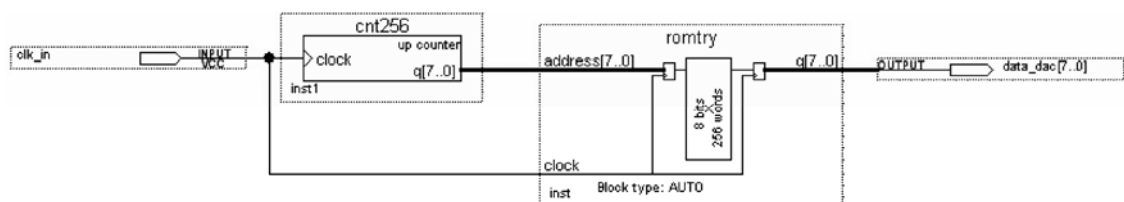


图 10.16 DAC0832 控制系统的顶层设计图

10.5 LED点阵的控制与显示

1. 设计任务

利用 FPGA 实现对 LED 点阵的控制与显示。

2. 设计基本要求

(1) 利用 16×16 点阵汉字字库提取程序，获取所需显示字符的十六进制数据，并将其存放在 LPM_ROM 模块中。

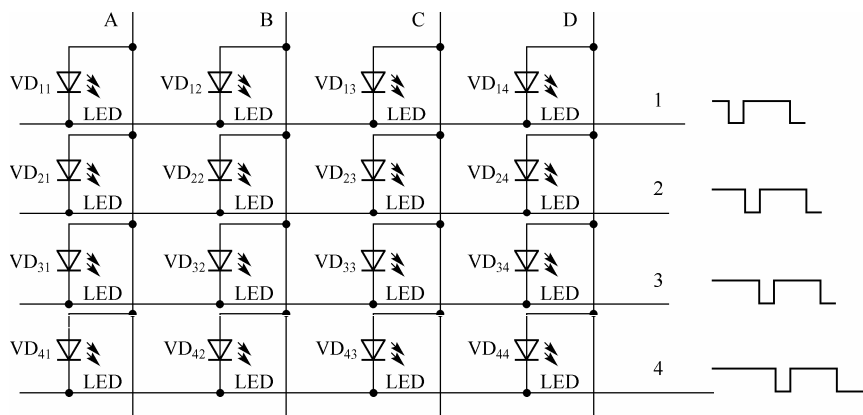
(2) 基于动态扫描原理，利用 FPGA 实现 LED 点阵的控制与显示。

本节将首先简单回顾第 2 章所介绍的 LED 点阵动态显示原理，然后给出在 FPGA 系统中最简捷的 LED 点阵控制模式，接着利用 16×16 点阵汉字字库提取程序获取“武汉大学”所对应的十六进制数据，最后构造模块实现 LED 点阵的控制与显示。

10.5.1 LED点阵的显示原理

在第 2 章中对于 LED 显示原理已有详细的介绍。常用的 LED 点阵显示采用动态扫描法，所谓动态扫描法，即将整屏画面分为若干部分分别进行刷新，利用人眼的视觉暂留现象而实现的一种显示方法。动态扫描采用分时选通技术，通过行驱动管的分时工作，使得每行 LED 的点亮时间占一帧内总时间的 $1/N$ ， N 为需要扫描的行数，只要扫描频率大于一定值，利用人眼的视觉惰性，即可以看到一幅完整的画面。

以共阴极 4×4 LED 点阵为例(参见图10.17)，LED 点阵的行线上加载扫描选通信号，列线上为数据输入，当行线上有一负脉冲选通信号时，列端四位数据中为“1”的发光二极管导通点亮。显示采用逐行扫描方式，数据端不断输入数据，行扫描按顺序逐行选通，扫描一个周期(4 次)产生一帧画面。

图 10.17 4×4 共阴极 LED 阵列

10.5.2 模块功能实现

1. 16×16 LED点阵的硬件设计

考虑到 FPGA 芯片 I/O 口驱动能力的不足，在系统设计中需将 FPGA 的行选通输出引脚和列数据输出引脚经驱动电路驱动后再与 LED 点阵相连接。驱动包括数据驱动和行驱动。行驱动承载的脉冲电流要大一些，设计时应先计算通过单个发光二极管的脉冲电流的大小，然后计算出一行上全亮时的总的脉冲电流的大小，根据计算结果设计驱动电路，以保证 LED 点阵的显示亮度适合观察。

在本系统设计中，由于采用的 16×16 LED 点阵所需驱动电流较小，所以 FPGA 的 16 位列数据输出引脚经两个 74HC245 锁存驱动后与 LED 点阵的 16 位列数据端口直接相连接；FPGA 的 4 位行选通输出引脚经 4 线-16 线译码器 74HC154N 译码(低电平有效)后连接 LED 点阵的 16 位行选通端口。

2. LED点阵的控制与显示模块

如图10.18所示，整个系统由三个模块构成：datarom 模块里面存放着待显示字符“武汉大学”的十六进制码；source1 模块用来产生 datarom 模块所需要的地址数据及 LED 点阵所需要的行选通编码信号(该编码信号经 4 线-16 线译码器后为 16 bit 的 LED 点阵行选通信号)；freq_divide 模块将输入的 50 MHz 的时钟信号分频得到合适的 1 kHz 扫描时钟信号。

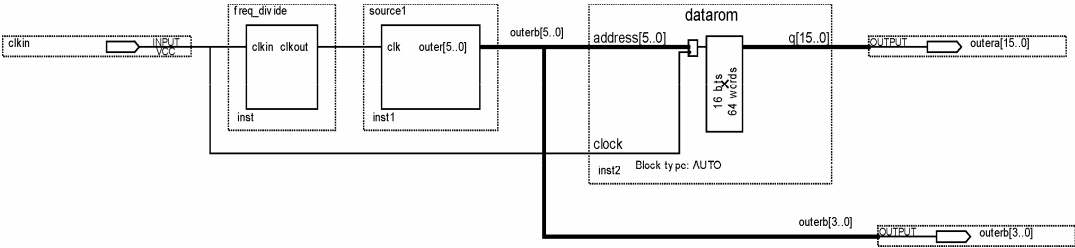


图 10.18 16×16 LED 点阵的控制与显示模块框图

- datarom 模块：利用专用的 16×16 点阵的字符字库提取软件提取待显示字符“武汉大学”所对应的十六进制码，如图10.19所示，将提取的十六进制码构造成 mif 文件，其中地址“000~00F”中的数据对应字符“武”，地址“010~01F”中的数据对应字符“汉”，地址“020~02F”中的数据对应字符“大”，地址“030~03F”中的数据对应字符“学”，并利用 LPM_ROM 模块构建一个存放着待显示 4 帧字符数据的 ROM。

Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f
00	0040	0050	3F48	0040	FFFE	0040	0440	0420	27A0	2420	2420	2410	2712	380A	E006	0002
10	2000	1000	17FC	0208	8208	4910	4910	1110	10A0	20A0	E040	20A0	2118	260E	2604	0000
20	0100	0100	0100	0100	0100	FFFE	0100	0280	0280	0240	0440	0420	0810	1018	200E	4004
30	0108	108C	0CC8	0890	7FFE	4004	8FE8	0040	0080	7FFE	0080	0080	0080	0080	0280	0100

图 10.19 字符“武汉大学”所对应的十六进制数据

- source1 模块：在实际显示过程中，考虑人眼的观察效果，在本设计中每个字符所对应的数据将重复扫描显示 16 次再转到下一帧的数据，因此在 source1 模块中设计了一个 10 位的地址加 1 计数器 outerbuf[9:0]，输出端口 outerb[5:0]={outerbuf[9:8], outerbuf[3:0]}；其中 outerb[5:0]为所存放列数据 ROM 的地址，与 datarom 模块的地址相连接，outerb[3:0]为行选通编码信号，与外设的 4 线-16 线译码器 74HC154N 相连接。
- freq_divide 模块：在本设计中，由于每帧字符数据扫描 16 次后才扫描显示下一帧字符数据，因此

基于人眼的视觉惰性,两个不同的字符数据之间的显示频率应小于 50 Hz;另一方面,若点阵的扫描频率过低,则每次扫描显示的字符将会出现闪烁的情况,从而不利于人眼的观察。`freq_divide` 模块产生了 1 kHz 点阵扫描时钟频率,从而实现了每个字符的显示频率大于 50 Hz(实际值为 $1000/16 \approx 66$ Hz),不同字符间的显示频率小于 50 Hz [实际值为 $1000/(16 \times 16) \approx 3.9$ Hz],保证了人眼的正常观察。

10.6 步进电机的转速/方向控制

1. 设计任务

利用 FPGA 实现对步进电机转速和方向的控制。

2. 设计基本要求

- (1) “手动”控制电机的“运转/停止”,电机以低速启动;低速停止。
- (2) 当电机处于“运转”状态时,可“动态调节”电机的转速和方向。
- (3) 电机转速加到最大时(12.5 转/秒),动态加速无效;电机转速减到最小时(1.25 转/秒)动态减速无效。

- (4) 系统第一次上电时,电机处于停止状态;系统上电时,默认的旋转方向为顺时针。

步进电机是利用脉冲信号控制的电机装置,步进电机每次接收到一组脉冲信号(脉冲电流),便旋转一个角度,即步进角。本节首先将简单介绍步进电机的工作原理,在熟悉和掌握步进电机工作原理的基础上分析和讨论利用 FPGA 实现步进电机控制的基本方法,其中环形脉冲发生器的设计是关键。读者在学习过程中应重点理解如何产生环形脉冲序列,如何控制步进电机的旋转方向,如何调整步进电机的转速及如何去除按键抖动。

10.6.1 步进电机的工作原理

步进电机是一种“脉冲/角度”变换装置,由一组称为“定子”的固定线圈绕组和一个称为“转子”的运动部件所构成。若给定子绕组输入一个“适当”的电流脉冲[即使得这些绕组中的某 1 个(或 2 个)处于“通电状态”,其他绕组处于“断电状态”],转子就会按“规定”的方向转动一个预定的角度,即电机“走了一步”。显然,在保证电机处于“不失步”的前提下,所输入电流脉冲的频率越高,电机的转速就越高;若希望改变电机的转动方向,只要改变电机定子绕组的通电顺序(即改变输入脉冲的“相位”)即可。

图 10.20 是该四相反应式步进电机工作原理示意图。开始时,开关 S_B 接通电源, S_A , S_C 和 S_D 断开, B 相磁极和转子 0,3 号齿对齐,同时,转子的 1,4 号齿就和 C, D 相绕组磁极产生错齿, 2, 5 号齿就和 D, A 相绕组磁极产生错齿。当开关 S_C 接通电源, S_B , S_A , S_D 断开时,由于 C 相绕组的磁力线和 1,4 号齿之间磁力线的作用,使转子转动, 1,4 号齿和 C 相绕组的磁极对齐。而 0,3 号齿和 A, B 相绕组产生错齿, 2,5 号齿就和 A, D 相绕组磁极产生错齿。以此类推, A, B, C, D 四相绕组轮流供电,则转子会沿着 A, B, C, D 方向转动。

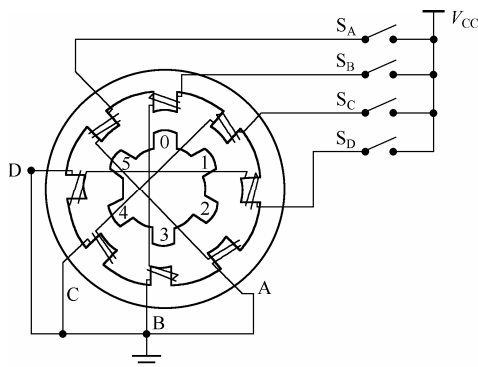


图 10.20 四相反应式步进电机工作原理示意图

应当注意的是，步进电机有一个技术参数：空载启动频率，即步进电机在空载情况下能够正常启动的脉冲频率。如果脉冲频率高于该值，则电机不能正常启动，可能发生丢步或堵转。在有负载的情况下，启动频率应更低。如果要使电机达到高速转动，脉冲频率应该有加速过程，即启动频率较低，然后按一定加速度升到所希望的高频(电机转速从低速升到高速)。因此步进电机需低速启动运转，若高于一定速度就无法启动，并伴有啸叫声。

在本训练中使用四相步进电机，控制端有 4 个：SMA, SMB, SMC 和 SMD。其对应的脉冲分配表如表 10.1 和表 10.2 所示(假设电机按顺时针方向旋转)，表中的“1”表示使对应的相线圈绕组处于“通电状态”，表中的“0”表示使对应的相线圈组处于“断电状态”。

表 10.1 四相单 4 拍脉冲分配表

节拍编码	SMA	SMB	SMC	SMD	节拍编码	SMA	SMB	SMC	SMD
n	1	0	0	0	$n+2$	0	0	1	0
$n+1$	0	1	0	0	$n+3$	0	0	0	1

表 10.2 四相单 8 拍脉冲分配表

节拍编码	SMA	SMB	SMC	SMD	节拍编码	SMA	SMB	SMC	SMD
n	1	0	0	0	$n+4$	0	0	1	0
$n+1$	1	1	0	0	$n+5$	0	0	1	1
$n+2$	0	1	0	0	$n+6$	0	0	0	1
$n+3$	0	1	1	0	$n+7$	1	0	0	1

显然，若步进电机使用单 8 拍的工作方式，每变换一次通电脉冲，电机转子所转过的角度(即步距角)是使用单 4 拍方式的一半。

10.6.2 模块功能实现

1. 系统整体设计

该系统由时钟分频模块 PIN1M、按键控制与去抖动模块 key、按键控制与去抖动模块 key1 及步进电机控制模块 motor_step1 组成，系统的顶层设计框图如图10.21 所示。

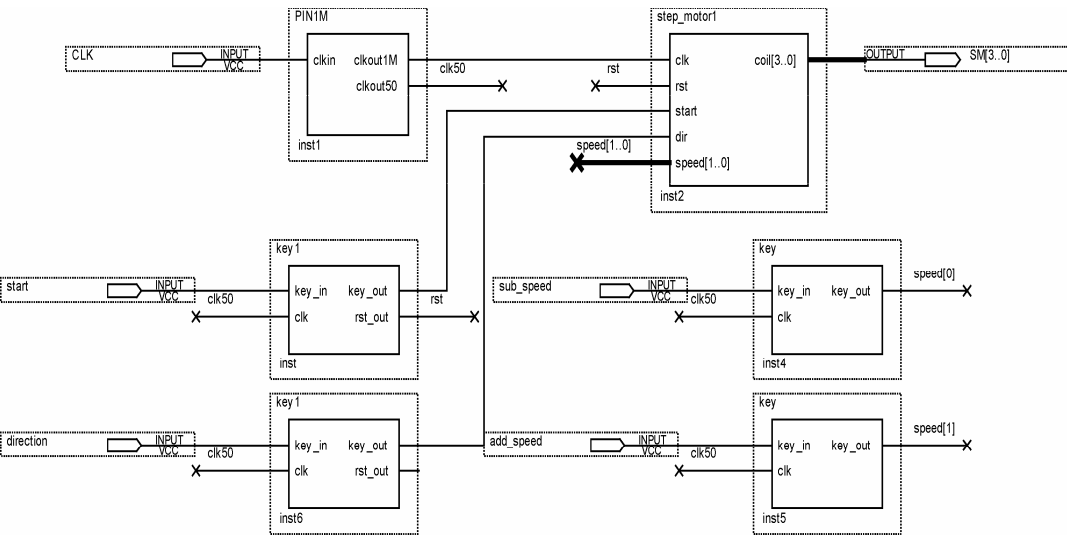


图 10.21 系统的顶层设计框图

2. 时钟分频模块PIN1M

设计训练所采用的系统平台的输入时钟频率为 50 MHz, 在时钟分频模块 PIN1M 中, 通过设计两个计数分频器来产生频率分别为 1 MHz 和 50 Hz 的两个时钟信号, 其中频率为 1 MHz 的时钟信号用做步进电机控制模块 motor_step1 的输入时钟, 频率为 50 Hz 的时钟信号用做按键控制与去抖动模块 key 和按键控制与去抖动模块 key1 的工作时钟。

3. 按键控制与去抖动模块key, key1

为了实现系统功能, 需要用到“start”, “direction”, “add_speed”和“sub_speed”4个按键, 其功能如下。

- 按键“start”: 系统上电后, 按下该键, 启动步进电机工作; 再次按下该键, 停止步进电机工作, 周而复始。
- 按键“direction”: 系统上电启动步进电机工作后, 电机作顺时针旋转, 按下该键后电机改为逆时针旋转; 再次按下该键, 电机改为顺时针旋转, 周而复始。
- 按键“add_speed”: 每按下此键, 步进电机的旋转速度增加 1.25 转/秒, 速度增加到 12.5 转/秒后, 此键失效。
- 按键“sub_speed”: 每按下此键, 步进电机的旋转速度减小 1.25 转/秒, 速度减小到 1.25 转/秒后, 此键失效。

在硬件平台中, 通过按键实现控制功能是十分常见的, 但是由于按键是机械触点, 当机械触点断开、闭合时会有抖动, 为使每一次按键动作只做一次有效响应, 就必须考虑去除抖动。一般来说, 按键的抖动可以采用硬件或软件的方法消除, 硬件去抖通常可采用 RS 触发器或采用具有预置和清除功能的 D 触发器, 接入按键开关回路对开关信号整形来消除抖动。软件去抖采用延时读开关信号的方法来实现消除抖动, 其方法是: 根据人按键速度的统计, 一般按键时间大于 100 ms, 以按键按下所产生开关信号占空比的 50% 计算, 按下时间大于 50 ms。假设按键按下将产生高电平, 若高电平持续时间小于 50 ms 即可判断为抖动信号, 若高电平持续时间大于 50 ms 即可判断为有效按键信号。

在 FPGA 的设计中, 常用的去抖动方式也有两种: 加固定延时方式和移位寄存器锁存方式。

(1) 加固定延时方式: 即当系统检测到按键按下的高电平后, 并不是立即做出判断, 而是经过 20 ms 的延时后, 再一次检测按键值, 如果依然为高电平, 则判断为按键信号, 否则判断为抖动信号。该方法主要应用于软件去抖。

(2) 移位寄存器锁存方式: 利用 3 个移位寄存器, 寄存器的工作时钟周期为 20 ms, 通过异或门检测出按键的上升沿, 并持续输出一个周期的高电平, 从而产生了一个由按键触发的上升脉冲。该方法主要应用于利用 FPGA 等可编程器件来实现按键去抖动, 因此在本设计中采用该方法。

根据系统功能要求, 按键“start”和“direction”被按下后 FPGA 产生一个持续的电平信号, 而按键“add_speed”和“sub_speed”被按下后 FPGA 产生一个上升沿脉冲。另外, 当“start”被按下后 FPGA 产生一个正脉冲的复位信号“rst”。可参考的 key1 模块的核心代码如下:

```
always @(posedge clk)
begin
    key_inbuf1 <= key_in;
    key_inbuf2 <= key_inbuf1;
    key_inbuf3 <= key_inbuf2;
    if((!key_inbuf3) && key_inbuf2 && key_inbuf1)
```

```

begin
    key_out <= ~key_out;
    rst_out <= 1'b1;
end
else
    rst_out <= 1'b0;
end
end

```

4. 步进电机控制模块motor_step1

步进电机控制模块的核心部分就是环形脉冲发生器，而环形脉冲发生器设计的目的实质上是设计一个状态机，以达到在输出端有序给出四相单 8 拍脉冲的目的，如表 10.2 所示的从 n 时刻到 $n+7$ 时刻的脉冲电压。脉冲电压周期(脉冲频率)的改变将控制步进电机的旋转速度，脉冲电压的循环方向(脉冲相位)的改变将控制步进电机的旋转方向。

一般情况下，可以设计一个状态机以实现状态从 n 时刻到 $n+7$ 时刻周期性脉冲电压序列的有序变化；反之亦然。但更加简捷的做法为设计一个环形移位寄存器，其工作时钟频率可调，移位方向可变，可参考的核心代码如下。其中 $s_0 \sim s_7$ 对应 n 时刻到 $n+7$ 时刻的四相单 8 拍脉冲：1000, 1100, 0100, 0110, 0010, 0011, 0001, 1001。dir 为按键“direction”产生的旋转方向控制信号，rst 为按键“start”所产生的复位信号。

```

parameter
    s0=4'b0001, s1=4'b0011, s2=4'b0010, s3=4'b0110, s4=4'b0100, s5=4'b1100, s6=4'b1
    000, s7=4'b1001;
always @(posedge clk_scan or posedge rst)
begin
    if(rst)
        begin
            state <= {s0,s1,s2,s3,s4,s5,s6,s7};
        end
    else if (start)
        begin
            if(dir)
                state <= {state[27:0],state[31:28]};
            else
                state <= {state[7:0],state[31:8]};
        end
    end
    assign coil = state[31:28];
end

```

步进电机的工作频率由一个可变分频系数的分频器产生，每按下一次“add_speed”按键，分频器的计数最大值就减小 5000，当计数器的计数值减小到 5000 后，按键“add_speed”失效；每按下一次“sub_speed”按键，分频器的计数最大值就增加 5000，当计数器的计数值增加到 50 000 后，按键“sub_speed”失效。由于分频器的计数值的取值范围为 5000~50 000，步进为 5000，所以步进电机的工作频率为 $\frac{10^6}{2 \times 50\,000 \times 8} \sim \frac{10^6}{2 \times 5000 \times 8}$ 转/秒，即 1.25~12.5 转/秒，步进为 1.25 转/秒，从而满足系统的性能要求。可参考的核心代码如下：


```
always @(posedge clk)
begin
    if(cnt == 9999)
        begin
            cnt <= 0;
            clk_buf <= ~clk_buf;
        end
    else
        cnt <= cnt + 1'b1;
    end
reg[19:0] comp;
always @(posedge clk_buf or posedge rst)
begin
    if(rst)
        comp <= 50000;
    else if(start)
        begin
            if(speed == 2'b10)
                begin
                    if(comp < 50000)
                        comp <= comp + 5000;
                    end
                else if(speed == 2'b01)
                    begin
                        if(comp > 5000)
                            comp <= comp - 5000;
                        end
                    end
            end
        else
            begin
                if((comp > 5000) && (comp < 50000))
                    comp <= comp + 5000;
                else
                    comp <= 50000;
                end
            end
reg[19:0] cnt1;
reg clk_scan;
always @(posedge clk)
begin
    if(cnt1 == comp)
        begin
            cnt1 <= 0;
            clk_scan <= ~clk_scan;
        end
    else
        cnt1 <= cnt1 + 1'b1;
    end
end
```

10.7 本章小结

本章介绍了基于 FPGA 的基本外设控制电路的设计、实现原理和方法，并以具体的实例进行相关的讨论。本章主要讨论了基于 FPGA 的七段 LED 显示原理和七段 LED 数码块显示的设计过程、ADC0809 的转换时序和利用 FPGA 进行控制的过程、DAC0832 的工作原理和利用 FPGA 进行控制的方法、利用 FPGA 实现数模转换时的关键技术、利用 FPGA 实现 LED 点阵的控制及基于 FPGA 的步进电机控制。本章的重点是通过学习所提供的设计实例，掌握 FPGA 基本的设计与开发方法。本章在介绍这些设计实例时，还给出了它们相应的程序和实现框图以供参考，希望读者通过本章有关设计实例的学习，能够掌握较复杂的 FPGA 设计过程中的一些基本方法，并且对 FPGA 的工作特点能有更深入的认识。当然，由于理解、使用 HDL 程序的能力及设计习惯的不同，实现设计的风格也是不同的。所以，设计实例中给出的程序仅起引导和参考的作用，用以启发读者对于设计过程的理解和思考，进而提高自身的创造性设计能力。

第 11 章 基于FPGA的协议转换电路设计

11.1 引言

单片机、ARM、DSP 等微控制器内部大多都集成了 RS232 和 USB2.0 (Universal Serial Bus) 等通信协议模块,能够很方便地与其他控制器进行基于标准协议的通信。而对于 FPGA 来说,由于其器件结构本身并不包含这些通信协议模块,所以一般情况下设计者需要调用 IP 核或自行设计通信协议模块以实现与其他系统之间的基于标准协议的通信。Altera 公司提供了大量的 IP 核,设计者可以通过参数化设置界面方便地调用 IP 核,但是 IP 核中的大多数功能模块往往需要付费的 License 支持。在某些应用场合中,系统间的通信往往只需要采用某种通信协议中的一部分,例如基于 FPGA 的系统 A 和基于单片机的系统 B 短距离的通信连接时,它们之间所采用的异步串行通信并不需要完整的通用异步收发器 (Universal Asynchronous Receiver-Transmitter, UART) 功能,其中 RS232 标准中的联络控制信号线可以省略,仅保留收、发数据线和地线即可满足串行通信的要求,这样极大地降低了实现 UART 功能时的 FPGA 编程难度,设计者从而能够很容易利用 HDL 语言构造简易的 UART 功能模块来完成系统间的通信。

UART 具有构造简单、使用方便的特点,但因为其通信方式的原因,数据传输速率较低。在数据通信中除了 UART 之外,USB2.0 传输协议同样得到了广泛应用,其传输速率远远超过了 IEEE1394 接口进行视频传输的 400 Mb/s,达到了 480 Mb/s,而且具有即插即用的 PnP (Plug and Play) 功能、可进行菊花链式的级联(通过 USB HUB 进行外围扩展)、可串联多达 127 个 USB 设备等优点。应用该协议可支持实时语音、音频和视频数据的传输。根据 USB2.0 的协议规范,采用硬件描述语言实现符合该协议的功能控制器,利用 FPGA 构造基于 USB2.0 的从设备数据收发模块,可实现从设备与 PC (主设备) 之间的高速通信。

本章将分别介绍基于 FPGA 的简易 UART 功能的实现及基于 FPGA 的 USB2.0 通信电路的实现。通过这两个实例的学习,读者除了进一步巩固如何利用状态机来构造时序逻辑电路模块外,还应学会如何在设计中利用最有效的方法来实现电路功能。

11.2 简易UART接收模块的设计与实现

1. 设计任务

利用 FPGA 构造简易的 UART 接收模块,该模块接收 PC 发送的 RS232 串口数据,并显示在四位一体数码块上。

2. 设计基本要求

(1) 构建简易的 UART 接收模块,能够正确接收 PC 发送的 RS232 串口数据,其数据格式为: 1 位起始位、8 位数据位、1 位停止位、无奇偶校验位,波特率设定为 115 200 Baud/s。

(2) 基于动态扫描原理,将接收的串口数据显示在四位一体数码块上。

UART 允许在串行链路上进行全双工的通信。串行外设用到的 RS232-C 异步串行接口,一般采用专用的集成电路即 UART 实现,如 8250, 8251 和 NS16450 等芯片都是常见的 UART 器件,有时系统

设计不需要使用完整的 UART 的功能和某些辅助功能，或者设计上用到了 FPGA/CPLD 器件，那么就可以将所需要的 UART 功能集成到 FPGA 内部，使用硬件描述语言将 UART 的核心功能集成，从而使整个设计更加紧凑、稳定且可靠。

本训练要求在实验开发平台上实现一个简易的 UART 接收功能模块，首先必须了解 UART 的基本通信原理，在分析 UART 的基本发送和接收时序的基础上，设计简易的 UART 接收模块，并运用前面章节设计的结果，将接收的数据显示到四位一体的数码块上。

11.2.1 UART的基本通信原理

UART 即通用异步收发器，是计算机系统中常见的一个外设，常简称为串口。所谓的串行通信，是指数据的各位逐位依次传送。串行通信的优点是传输线少、通信距离长、成本低；其缺点是速度慢、效率低。

串口通信的概念非常简单，串口按位 (bit) 发送和接收数据，尽管比按字节 (Byte) 的并行通信速率慢，但是串口通信方式可以在使用一根线发送数据的同时用另一根线接收数据。其电路简单并且能够实现远距离通信。比如 IEEE488 定义并行通信状态时，规定设备线总长不得超过 20 m，并且任意两个设备间的并行通信线长度不得超过 2 m；而对于串口而言，长度可达 1200 m。典型的串口用于 ASCII 码字符的传输时，通信使用 3 根线完成：(1) 地线，(2) 发送线和 (3) 接收线。由于串口通信是异步的，端口能够在 1 根线上发送数据同时在另一根线上接收数据。其他线用于握手信号的传递，但不是必须的。串口通信最重要的参数是波特率、数据位、停止位和奇偶校验。对于两个进行通信的端口，以下参数必须匹配：

(1) 波特率 波特率是指每秒传输的符号的个数，单位为波特 / 秒 (Baud/s)。对于二进制传输，它在数值上等于信息传输速率，但两者的物理意义不同。例如假设一台设备的数据传输速率为 240 字符 / 秒，异步通信方式时，字符格式为：1 位起始位，8 位数据位，1 位停止位，则波特率为 $240 \times 10 = 2400 \text{ Baud/s}$ ，其信息传输速率为 2400 b/s。每一个二进制代码位的传送时间为波特率的倒数： $T_d = 1/2400 \approx 0.417 \text{ ms}$ 。异步通信的波特率一般在 50 ~ 19 200 b/s 之间。

(2) 数据位 数据位是指异步通信时要传送的数据的具体内容，可以是 5 位、7 位和 8 位等。异步通信方式用一个起始位表示一个字符的开始，用停止位表示字符的结束，数据位则在起始位之后、停止位之前，这样就构成了一帧 (如图 11.1 所示)。在异步通信中，每个数据都以特定的帧形式传送，数据在通信线上一位一位地串行传送。



图 11.1 异步通信的字符格式

(3) 停止位 用于表示发送一个数据的结束，用逻辑高电平表示，占 1 位、1.5 位或 2 位，其中，1.5 位是指时间概念。需说明的是，在异步通信中字符和字符之间是异步的，但接收到字符的起始位后，该字符内的各位的时间是确定的，或者说位与位之间的脉冲间隔是相同的，即字符内部的各位是同步传送 (位同步) 的。因此，停止位不仅仅是表示传输的结束，并且提供了计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度就越大，但其代价是数据传输率的降低。

(4) 奇偶校验位 用于有限差错检测。当数据不需要进行奇偶校验时, 此位可省略。

本节设计方案采用的字符结构为: 1 位起始位, 8 位数据位, 1 位停止位, 无奇偶校验位, 如图 11.1 所示。

UART 主要由波特率发生器、发送部分和接收部分等组成。UART 发送部分的用途是将准备输出的并行数据按照基本 UART 帧格式转化(拼装)为 TXD 信号串行输出。UART 接收部分接收 RXD 串行信号, 并将其转化为并行数据。波特率发生器用做专门产生一个远远高于波特率的本地时钟信号对输入 RXD 不断采样, 使接收器与发送器保持同步。

UART 的通信协议的物理层为 RS232-C 标准: RS232-C 标准对两个方面做了规定, 即信号电平标准和控制信号线的定义。

(1) 信号电平标准 RS232-C 采用 EIA 电平规范, 使用负逻辑规定逻辑电平, 信号电平与通常的 TTL 电平不兼容。RS232-C 将 $-15 \sim -5 \text{ V}$ 规定为“1”, $+5 \sim +15 \text{ V}$ 规定为“0”, 因此为了能够将计算机接口与终端 TTL 器件连接, 必须在 RS232-C 与 TTL 电路之间进行电平的变换。实现这种变换的方法可用分立元件, 也可用集成电路芯片。目前较为广泛地使用集成电路转换器件, 如 MC1488, SN75150 芯片来完成 TTL 电平到 RS232 电平的转换, MAX232 芯片可完成 TTL $\longleftrightarrow$ EIA 双向电平转换。在本节设计任务中采用 MAX232 来完成电平转换工作。

(2) 控制信号线 RS232-C 是一种有 25 条传输线构成的总线, 然而通常的用于传输 ASCII 码字符的 UART 通信时, 只需要使用 3 根线: 地线 GND、发送线 TxData 和接收线 ReData。在本设计任务中就仅需利用 GND, TxData 和 ReData 这三根线来实现简易的 UART 串口通信, RS232 标准中的联络控制信号线可以省略。

11.2.2 系统总体分析

本训练主要是实现一个简易的 UART 接收功能模块, 接收并在 LED 上显示 PC 发送的 RS232 串口数据。由前面的介绍可知, 系统的顶层模块主要由 UART 接收器、波特率发生器和 LED 显示模块组成。设计所要完成的功能为: UART 接收器接收异步信号, 并完成串/并的转换及显示。系统设计中, 顶层设计可采用原理图的设计方法, 其中 divfreq 模块和锁相环模块一起构成波特率发生器, 产生接收模块 re 的工作时钟 clk。波特率发生器产生的时钟频率不是波特率时钟频率, 而是波特率时钟频率的 16 倍, 目的是为在接收时对异步的串行数据进行精确采样, 其频率为 $1\,843\,200 \text{ Hz}$ [$115\,200 (\text{波特率}) \times 16$]; 显示模块 disp4 (在 10.2 节中已有介绍) 完成接收数据的显示, 整个系统的顶层设计如图 11.2 所示。

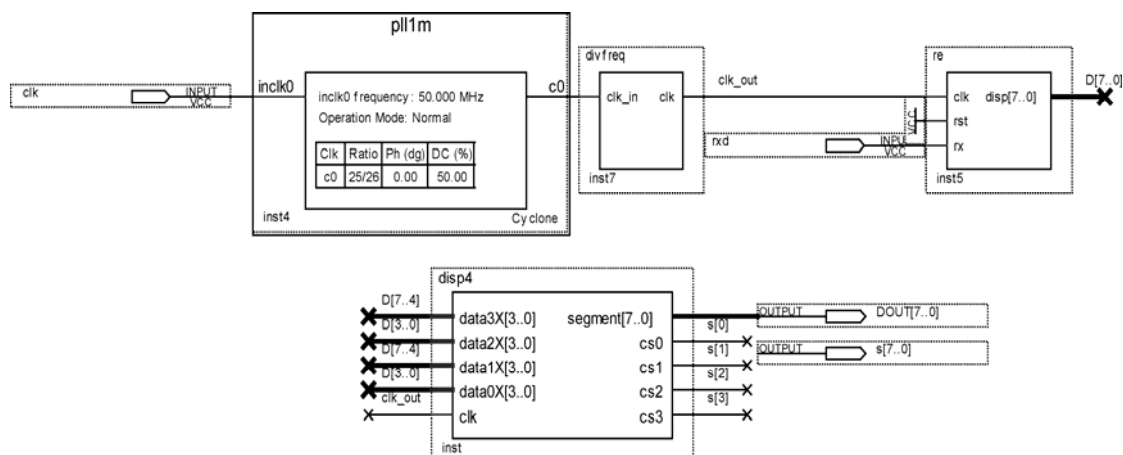


图 11.2 简易 UART 接收和显示系统的顶层设计图

11.2.3 关键模块设计

为了准确地接收到 PC 发送的串口数据,接收模块的时钟 clk 是发送时钟的 16 倍(即 $f_{clk} = 115\,200 \times 16$) Hz,以保证接收模块在位判决时每次的取样判决都在码元的正中间,以减小传送差错。串行数据帧和接收时钟是异步的,发送来的数据由逻辑 1 变为逻辑 0 可以视为一个数据帧的开始。此时,接收器先要捕捉起始位,确定 RXD 输入电平由 1 到 0 的变化。依照位判决的规则设计,当检测到逻辑电平 0 时,并不立即确定为起始位,而要在 8 个时钟周期后再次取样,如果依然是逻辑电平 0,即认为是正常的起始位,而不是噪声干扰。然后在每隔 16 个时钟周期采样接收数据,并由移位寄存器接收数据并完成串并转换,最后输出数据到显示端口。依据 PC 发送数据格式,串口接收模块的工作流程如下:

(1) Idle: 上电或复位后模块进入 Idle 状态,在该状态下,当模块检测到 RXD 端的低电平时,进入 S1 状态;否则在 Idle 状态等待。

(2) S1: 8 个时钟周期后,再一次检测 RXD 端的电平值,若输入电平为低,则表示接收到 1 位起始位,进入 S2 状态;若输入电平为高,则进入 Idle 状态,继续检测起始位的到来。

(3) S2: 16 个时钟周期后,将 RXD 端的电平值赋值给 temp 变量($temp \leftarrow \{rx, temp[7:1]\}$),循环 8 个周期后, temp 值即为 RXD 端接收到的一帧串口数据,模块进入到 s3 状态。

(4) S3: 16 个时钟周期后,判断 RXD 端的电平值,若输入电平为高,则可判断为接收到串口数据的停止位,表示一帧数据接收完毕,且接收数据有效,并将 temp 的数据赋值给 DISP 端口;若输入电平为低,则接收数据无效, temp 值清零,模块进入 Idle 状态,开始新一轮的接收操作。

接收模块 re 采用 Verilog 描述的参考代码如下:

```
module re(clk,rst,rx,disp);
input clk,rst,rx;
output[7:0] disp;
reg[7:0] disp;
reg[3:0] state,cnt,num;
reg[7:0] temp;
reg[10:0] count;
parameter idle=0,s1=1,s2=2,s3=3;
always @(posedge clk or negedge rst)
begin
    if(!rst)
        begin
            state <= idle;
            disp <= 8'd0;
            temp <= 8'd0;
            cnt <= 4'd0;
            num <= 4'd0;
        end
    else
        begin
            case(state)
                idle: begin
                    temp <= 8'd0;
                    if(rx == 1'b0)
```

```
        state <= s1;
    else
        state <= idle;
    end
s1: begin
    if(cnt == 4'd7)
        begin
            cnt <= 4'd0;
            if(rx == 1'b0)
                state <= s2;
            else
                state <= idle;
            end
        end
    else
        begin
            cnt <= cnt + 1'b1;
            state <= s1;
        end
    end
s2: begin
    if(cnt == 4'd15)
        begin
            cnt <= 4'd0;
            temp <= {rx,temp[7:1]};
            if(num == 4'd7)
                begin
                    num <= 4'd0;
                    state <= s3;
                end
            else
                begin
                    num <= num + 1'b1;
                    state <= s2;
                end
            end
        end
    else
        begin
            cnt <= cnt + 1'b1;
            state <= s2;
        end
    end
s3: begin
    if(cnt == 4'd15)
        begin
            cnt <= 4'd0;
            state <= idle;
            if(rx == 1'b1)
```

```
        disp <= temp;//
    end
    else
        cnt <= cnt + 1'b1;
    end
    default:
        state <= idle;
    endcase
end
end
endmodule
```

11.3 基于USB2.0 数据收发模块的设计与实现

1. 设计任务

利用 FPGA 和 CY7C68013 芯片构造基于 USB2.0 的从设备数据收发模块,通过“EZ-USB Control Panel”软件实现从设备与 PC(主设备)之间的通信。

2. 设计基本要求

(1) 构建基于 USB2.0 的从设备数据收发模块,采用块传输方式完成与 PC 之间的数据交换;

(2) 依据 CY7C68013 的时序关系,从设备数据收发模块的读速率应能达到 160 Mb/s,写速率应能达到 133 Mb/s。

本训练是利用 FPGA 和 CY7C68013 芯片构造基于 USB2.0 的数据收发模块。由于 CY7C68013 是符合 USB2.0 标准的微控制器,集成了符合 USB2.0 的收发器、串行接口引擎(SIE)、增强型 8051 内核及可编程的外围接口,因此 CY7C68013 与 FPGA 接口的逻辑粘接方式及实现逻辑粘接的 Verilog HDL 编程成为本设计的关键点。本节将首先阐述 USB 通信的特点及 USB2.0 规范中的 4 种不同传输类型,然后介绍 CY7C68013 的基本功能,最后结合 FPGA 和 CY7C68013 芯片快速构建出基于 USB2.0 的从设备数据收发模块,以块传输方式实现与 PC 之间的数据交换,读速率达到 160 Mb/s,写速率达到 133 Mb/s。

11.3.1 USB2.0 简介及传输类型

在 IBM PC 最早的设计中,就已经使用并行接口及 RS232 串行接口。这不仅加速了原始 PC 设计的流程,并且允许将已经在市面上出现的打印机和调制解调器立即连接到 PC 上。但随着计算机和外围设备的发展,并行接口和 RS232 串行接口已逐渐成为通信的瓶颈。USB 规范的出现,突破了旧接口的限制,不仅具备快速通信的能力,而且其灵活性令它可以取代各种外围设备所使用的接口,成为了目前最常用的 PC 的连接端口。

在传输速度方面,USB 支持 3 种总线速率:低速(low speed)的 1.5 Mb/s、中速(full speed)的 12 Mb/s 和高速(high speed)的 480 Mb/s。这些速率是指总线所支持的信号的位速率,具体到某个 USB 设备可期望的数据传输速率会低一点。因为除了数据之外,总线还必须携带状态、控制和错误检查信号。另外,由于多个 USB 设备可能在分享总线,所以理论上单一传输的最大速率在高速时大约为 53 Mb/s,在中速时大约为 1.2 Mb/s,在低速时则为 800 b/s。

之所以采用 3 种速率,主要考虑到以下几个方面原因。

对于低速而言,首先低速设备通常造价低廉,此外对鼠标和其他需要柔性电缆连接的设备来说,

低速的电缆由于不需要高密度的同轴屏蔽线包裹,因此会更加柔软,使得使用者能方便地移动操作设备。应用领域为键盘、鼠标、游戏机等。

中速与现有的串行接口和并行接口的速率具有可比性,或性能更好,因此可以取代这些接口。应用领域为电话、广播、音频、麦克风、影像压缩等。

高速的传输速率达到了 480 Mb/s,和串口 115 200 b/s 的速率相比,相当于串口速率的 4 000 多倍,完全能够满足需要大量数据交换的外设的要求。应用领域为影像、存储设备、高速数据传输等。

在传输类型方面,USB 定义了 4 种传输类型。

(1) 控制传输 可靠的、非周期性的、由主机软件发起的请求或者回应的传送。控制传输通常用于命令事务和状态事务。在控制传输过程中,携带 USB 规范定义请求,主机借此来了解设备或配置设备,设备必须在端点 0(Endpoint 0)的默认管道支持控制传输。

(2) 中断传输 小规模数据的、低速的、固定延迟的传送。中断传输经常用于数据必须在指定时间内传输完的场合。一般应用包括键盘、鼠标、游戏手柄及集线器的状态报表等。中断传输与其他 USB 传输类型一样,只发生在主机轮询设备的时候,并不是由设备触发的硬件中断。但是这种传输“好像是中断”,因为它能够保证主机以最小的耽搁接收或发送数据。

(3) 等时传输 在主机和设备之间的周期性的、连续的通信,一般用于传送与时间相关的信息。等时传输用于必须以固定速率抵达或在指定时刻抵达、可以容忍偶发错误的流数据传输场合,它可以保证大量数据尽快地通过总线,但数据不一定要以固定速率来传输。与块传输不同的是,一旦等时传输开始,主机保证可以在预期的时间内完成传输任务,一般应用场合包括实时的语音和音乐。

(4) 块传输 非周期性的、大包的、可靠的数据传送。块传输适用于时间不重要的传输,可以传输大量的数据而不会阻塞总线,因为它会让其他传输类型先执行,等待可以传输的时间。在一个闲置的总线中,块传输是最快的传输类型;另外块传输同样会检测错误,一般应用包括从主机传送数据到打印机,从扫描仪传送数据给主机,以及读/写磁盘等。

USB 的信号传输和电源连接是通过一种四线的电缆实现的,四根信号线分别为 D+, D-, V_{BUS} 和 GND。

- V_{BUS} 和 GND: V_{BUS} 使用 +5 V 电源; GND 为电源地线,可向设备提供电源。
- D+ 和 D-: D+ 和 D- 为差分数据信号。USB 数据传输中没有专门的时钟信号线,时钟被调制后与差分数据一同被传送出去,时钟信号被转换为 NRZI 码,并填充了比特以保证转换的连续性,每一数据包中附有同步信号以使得接收方可还原出原时钟信号。

11.3.2 USB 控制器 CY7C68013

由于 USB 协议的复杂性,在开发 USB 从设备时,通常采用智能的 USB 外围设备控制器。该控制器必须能够检测和反映 USB 连接端口的事件,提供存储传输数据的方式,以及接收并处理接收到的数据。

各种 USB 控制器的芯片有不同的特性:有些控制器只需要寄存器来存储和提取 USB 数据;有些还包括管理和传送描述符给主机、设置数据交换的数值,以及确定传送适当的联络信息包等功能;有些 USB 控制器包含有内置的 CPU;有些则是依靠外部的 CPU 来处理非 USB 通信的工作。所有的 USB 控制器都有一个或多个 USB 连接端口、传输数据用的缓冲区、寄存器及 I/O 等。如果是内置 CPU 的 USB 控制器,还会包括内置的内存,或是外部内存的接口。

在 USB 从设备与主机正常通信之前,主机必须检测并配置设备,所以用户必须编写固件程序代码,响应主机的控制请求命令并且能够提供描述符,使主机了解设备的功能,从而配置设备并与设备进行正常通信。另外,固件还要有完成正常的数据传输功能。

Cypress 公司的 EZ-USB FX2 芯片 CY7C68013 是一款性能较高的 USB2.0 微控制器。在单芯片上集成了 USB2.0 收发器、SIE(串行接口引擎)、增强的 8051 微控制器和可编程的外围接口。EZ-USB FX2 中的智能 SIE 可以处理大部分 USB2.0 协议,使得微控制器可以专注于应用层设计,从而减少了开发时间、确保了 USB 的兼容性,其内部结构框图如图 11.3 所示。为叙述简单起见,下文使用 FX2 代指 USB 控制器。

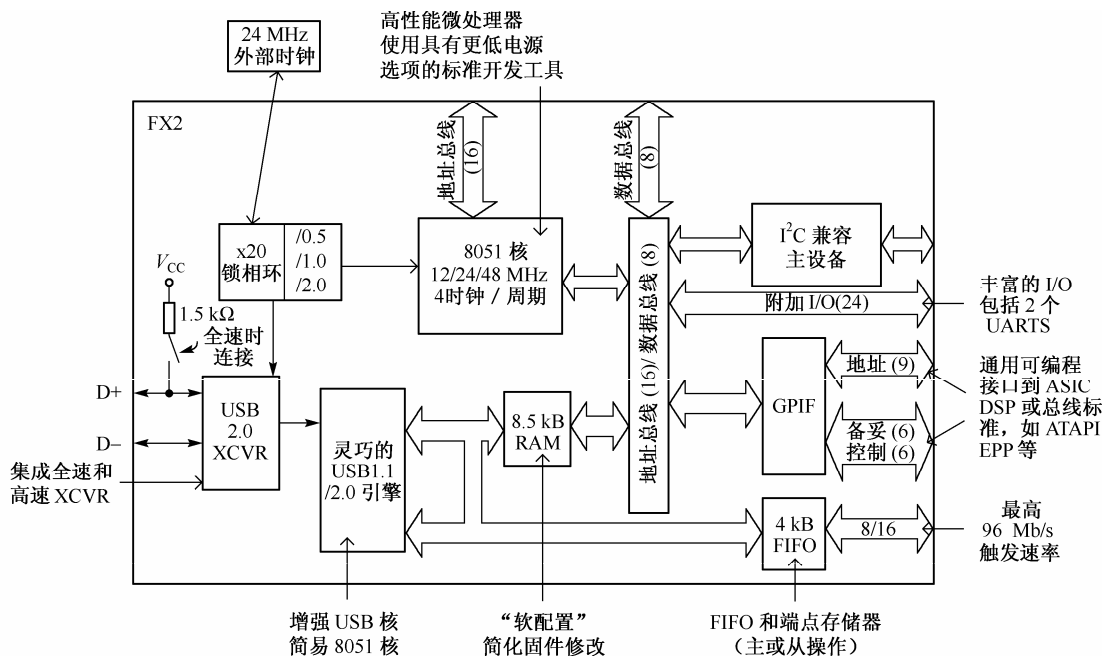


图 11.3 CY7C68013 的内部结构框图

(1) CY7C68013 的主要特性如下:

- 带有增强的 8051 内核,性能可达到标准 8051 的 5~10 倍,且与标准 8051 的指令完全兼容。
- 集成度高,芯片内部集成微处理器、RAM、SIE 等多个功能模块,从而减少了多个芯片接口部分需要时序配合的麻烦。
- 采用软配置,在外设未通过 USB 接口接到 PC 之前,外设上的固件存储在 PC 上;而一旦外设连接到 PC 上,PC 则先询问外设是“谁”(即读设备描述符),然后将该外设的固件下载到芯片的 RAM 中,这个过程称为再枚举。这样在开发过程中,当固件需要修改时,可以先在 PC 上修改好,然后再下载到芯片中。
- 具有易用的软件开发工具,该芯片开发系统的驱动程序和固件的开发和调试相互独立,可加快开发的速度。

(2) CY7C68013 有三种可用的接口模式: Ports(端口模式)、GPIF Master(GPIF 主控,可编程接口模式)和 Slave FIFO(从 FIFO 模式)。其中,后两种模式利用其内部集成的、可以独立于微处理器而自动处理 USB 事务的硬件(USB 核),数据的传输通过执行 USB 本身的协议来完成,微处理器可不参与数据传输,从而使数据的传输速率大大地提高,同时也简化了固件代码的编写。由于克服了微处理器这个带宽“瓶颈”,后两种方式广泛应用于大批量的数据传输,如图像、视频等信号的采集。

- 在“端口”模式下,所有 I/O 引脚都可作为 8051 的通用 I/O 口。作为一种最基本的数据传输

方式,其数据传输主要由固件程序完成,需要 CPU 的参与,因此数据传输速率比较低,适用于传输速率要求不高的场合。

- “GPIF 主控”接口模式使用 PORTB 和 PORTD 构成通向 4 个 FX2 端点 FIFO(EP2, EP4, EP6 和 EP8)的 16 位数据接口。GPIF 作为内部的主控制器与 FIFO 直接相连,并产生用户可编程的控制信号与外部接口进行通信。同时, GPIF 还可以通过 RDY 引脚采样外部信号并等待外部事件。由于 GPIF 的运行速率比 FIFO 快得多,因此其时序信号具有很好的编程分辨率。另外, GPIF 既可以使用内部时钟,也可以使用外部时钟。
- 在“从 FIFO”模式下,外部逻辑或外部处理器直接与 FX2 端点 FIFO 相连。在这种模式下, GPIF 不被激活,因为外部逻辑可直接控制 FIFO;外部主控端既可以是异步方式,也可以是同步方式,并可以为 FX2 接口提供自己的独立时钟。

(3) 固件程序的功能和基本架构

固件程序是从设备中微控制器及其外围电路正常工作必不可少的部分,其作用就是辅助硬件完成以下几种功能。

- 初始化工作,包括设置一些特殊功能寄存器的初值以获得所需的设备属性或者功能,例如开中断、使能端点、配置端口等;
- 辅助硬件完成设备的重新枚举过程,包括模拟设备的断开与重新连接,对接收到的设置包进行分析判断,从而对主机的设备请求做出适当的响应,完成主机对设备的配置任务;
- 对中断的处理;
- 数据的接收和发送;
- 外围电路的控制。

Cypress 公司为了简化和加速用户使用 EZ-USB FX2 芯片进行 USB 外设的开发过程,提供了一个完整的固件程序的架构,用户只需要提供一个 USB 描述符表,添加其他端点接收和发送数据的通信代码及控制外围电路的程序代码。

在本设计任务中,将采用块传输类型,利用 CY7C68013 的从 FIFO 方式,基于 FPGA 构造一个能够与 PC 完成数据交换的从设备数据收发模块。

11.3.3 基于 USB2.0 从设备数据收发模块的实现

Slave FIFO 方式是从机方式,在该方式下 FX2 的 CPU 不直接参与 USB 数据处理,而是简单地把 FX2 作为 USB 和外部数据处理逻辑(如 ASIC, DSP 和 SIE 控制器)之间的通道,数据流并不经过 CPU,而是通过 FX2 的 FIFO 直接传输。FIFO 由外部控制器控制,外部控制器可像普通 FIFO 一样对 FX2 的多层缓冲 FIFO 进行读写;与此同时, FIFO 提供外部控制器所需的时序信号、握手信号(满、空等)和输出使能等信号。FX2 的 Slave FIFO 工作方式可设为同步或异步工作方式;工作时钟可选为内部产生或外部输入。

在本设计中,采用 Slave FIFO 从机方式实现 FPGA 对 FX2 的控制, CY7C68013 与 FPGA 的连接示意图如图 11.4 所示。修改 FX2 芯片提供的基本固件程序,设置端点 EP2 为输出端点,大小为 1KB, FlagA 为 EP2 缓存的空标志,低电平有效;端点 EP6 为输入端点,大小为 1KB, FlagC 为 EP6 缓存的满标志,低电平有效。

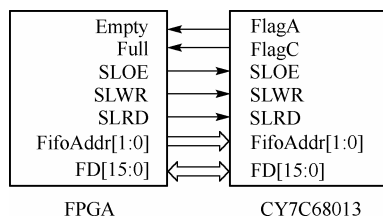


图 11.4 CY7C68013 与 FPGA 的连接示意图

参照 CY7C68013 的 Slave FIFO 下的工作时序, FX2 读数据过程也就是 PC 发送数据经过 CY7C68013 后传到 FPGA 的过程, 其过程如下:

① 分配 FIFOADR[1:0]=00, 这时 FIFO 指针会指向输出端点, 表明使用端点 EP2; 检查 FIFO 是否空, 当 Empty=1 时表示 FIFO 不空, 转到②, 否则保持在①;

② 赋值 SLOE=0, 使双向数据线 FD 在输出状态, 采样 FD 数据线上的数据, 并在 SLRD 的上升沿使 FIFO 指针自动加 1, 跳转到③;

③ 假如有更多的数据需要读, 转到②, 否则转到①。

FPGA 向 FIFO 写数据是读数据的逆过程, 也就是 FPGA 把数据写入 FX2 的 FIFO, PC 从 FIFO 从读取数据, 其编程过程如下:

① 分配 FIFOAFR[1:0]=10, FIFO 指针指向输入端点, 检查 FIFO 的满标志是否为 1, 假如 Full=1, 表示 FIFO 不满, 转到②, 否则保持在①;

② 把外部数据 indata 放在 FD 上, 同时将 SLWR 拉高, 以使 FIFO 指针自动加 1, 然后转到③;

③ 假如有更多的数据要传输, 转到②, 否则转到①。

在 USB2.0 从设备数据收发模块的测试过程中, 可以设计这样一个流程: 首先 PC 向从设备发送“43 A4”的启动命令; 从设备 CY7C68013+FPGA 在接收到“43 A4”命令后, 启动向 PC 写数据的操作, 以块传输方式持续向 PC 写入从“0001H”开始, 以“0001H”为步进的十六进制的数; PC 端可通过“EZ-USB Control Panel”观测到写入的整个数据。整个流程的 Verilog 参考代码如下, 工作时钟频率为 100 MHz。

```
module USBControl(
    CLK,           //系统时钟      100 M
    RST,           //复位          '0'
    FLAGA,         //EP2 输出空标志    1
    FLAGC,         //EP6 输入满标志    1
    SLOE,          //FIFO 输出允许     0
    SLWR,          //FIFO 写允许       0
    SLRD,          //FIFO 读允许       0
    FD,            //USB 双向数据线    16 bit
    PKTEND,        //USB 控制信号      1
    FIFOADDR       //FIFO 地址选择     2 bit
);
input RST, CLK;
input FLAGA, FLAGC;
output SLOE, SLWR, SLRD, PKTEND;
output[1:0] FIFOADDR;
inout[15:0] FD;
reg[15:0] PCD;
reg SLOE, SLWR, SLRD;
reg[1:0] FIFOADDR;
reg[15:0] FDR;
reg[4:0] present_state;
reg flag;
reg[2:0] delay_cnt;
parameter
idle=5'd0, RDS1=5'd1, getflag_1=5'd2, getflag_2=5'd3, getflag_3=5'd4, delay=5'd19,
delay1=5'd20, WRS1=5'd9, wrdata_1=5'd10, wrdata_2=5'd11, wrdata_4=5'd13, wrdat
a_5=5'd14, wrdata_6=5'd15;
//=====
always @(posedge CLK or negedge RST)
```

```

begin
  if(!RST)                //复位,"0"有效
  begin
    SLWR<=1'b1;
    SLOE<=1'b1;
    SLRD<=1'b1;
    flag<=1'b0;
    PCD<=16'h0;
    FDR<=16'h0;
    delay_cnt<=3'h0;
    FIFOADDR<=2'b00;
    present_state<=idle;
  end
  else
  begin
    case(present_state)
    idle:
      begin
        flag<=1'b0;
        SLWR<=1'b1;
        SLOE<=1'b1;
        SLRD<=1'b1;
        FDR<=16'h0;
        PCD<=16'h0;
        if(FLAGA==1'b1)
          present_state<=RDS1;
        else
          present_state<=idle;
        end
      RDS1:
        begin
          SLOE<=1'b0;
          flag<=1'b0;
          FIFOADDR<=2'b00;
          delay_cnt<=delay_cnt+1'b1;
          if(delay_cnt>=3'h2)
            begin
              delay_cnt<=3'h0;
              present_state<=getflag_1;
            end
          else
            present_state<=RDS1;
          end
        getflag_1:
          begin
            SLRD<=1'b0;
            present_state<=getflag_2;
          end
        getflag_2:
          begin
            PCD<=FD;
            present_state<=getflag_3;
          end
        getflag_3:
          begin
            SLRD<=1'b1;

```

```

        if(PCD==16'h443)
            present_state<=WRS1;
        else
            present_state<=idle;
        end
WRS1:
    begin
        if(FLAGC)
            begin
                SLWR<=1'b1;
                FIFOADDR<=2'b10;
                present_state<=wrd1;
            end
        else
            present_state<=WRS1;
        end
    wrd1:
        begin
            flag<=1'b1;
            SLWR<=1'b0;
            present_state<=delay1;
        end
    delay1:
        begin
            delay_cnt<=delay_cnt+1'b1;
            if(delay_cnt>=3'h2)
                begin
                    delay_cnt<=3'h0;
                    present_state<=wrd2;
                end
            else
                present_state<=delay1;
            end
        wrd2:
            begin
                FDR<=FDR+1'b1;
                present_state<=wrd4;
            end
        wrd4:
            begin
                SLWR<=1'b1;
                present_state<=wrd5;
            end
        wrd5:
            begin
                delay_cnt<=delay_cnt+1'b1;
                if(delay_cnt>=3'h3)
                    begin
                        delay_cnt<=3'h0;
                        present_state<=wrd6;
                    end
                else
                    present_state<=wrd5;
                end
            end
    end
end

```

```

wrdata_6:
    begin
        SLWR<=1'b1;
        present_state<=WRS1;
    end
default:
    present_state<=idle;
endcase
end
end
assign PKTEND=1'b1;
assign FD=(flag)?FDR:16'hzzzz;
//=====
Endmodule

```

如图 11.5 所示,在设计过程中,首先在“EZ-USB Control Panel”软件的“Bulk/Int”选项“pipe”栏中选择“0:Endpoint 2 OUT”,并在“Hex Bytes”栏中输入“43 A4”,以启动向 PC 写数据的操作;然后在“Bulk/Int”选项“pipe”栏中选择“2:Endpoint 6 IN”,在“Length”栏中输入“512”,然后单击“Bulk/Int”按钮,PC 将以块传输方式从 CY7C68013 中读写 512 字节的数据,并显示在对话框中,其结果与理论值一致。经过实验验证,PC 若持续读取大容量数据,基于块传输的 SLVAE FIFO 的异步传输模式,速率可接近 133 Mb/s。

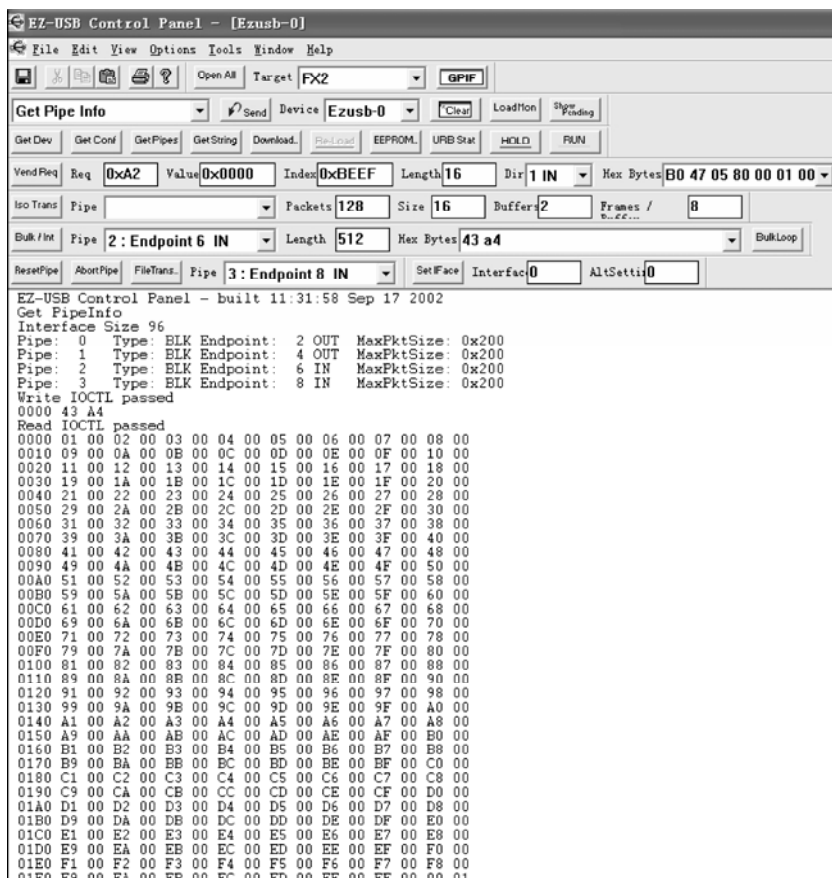


图 11.5 EZ-USB Control Panel 软件截图

11.4 本章小结

本章分别介绍了基于 FPGA 的 UART 接收器和 USB2.0 数据收发模块的设计与实现，这些内容的学习有着重要的意义，因为 FPGA 作为逻辑粘合部件，能为各种处理器提供基于各类总线的外设接口，如高速设备与低速设备之间的接口等应用。另外，本章介绍的电路的时序和状态较前面章节更加复杂，希望读者通过本章的学习能够更进一步加深对 EDA 设计的理解，进一步掌握复杂逻辑的 FPGA 设计方法。

第 12 章 基于FPGA的FFT算法实现

12.1 引言

时域是信号在时间轴上随时间变化的总体概括,其动态信号 $x(t)$ 是描述信号在不同时刻取值的函数。而在频域中,自变量是频率,即横轴是频率,纵轴是该频率信号的幅度,也就是通常说的频谱图,频谱图描述了信号的频率结构及频率与该频率信号幅度的关系。由于信号往往在频域比其在时域更加简单和直观,所以大部分信号分析的工作是在频域中进行的。而傅里叶变换可以实现信号在时间域和频率域之间的相互转换,它可以将时域中的信号 $x(t)$ 映射为频域中的 $X(f)$,使人们能更直观地了解和获取信号的有关信息。另外,由于傅里叶变换是线性变换,它也使得对系统的分析更加简单。

然而,对于连续时间信号,为了获取傅里叶(逆)变换,无论在时域还是在频域都要对连续函数进行积分运算,而且其积分上、下限是从 $-\infty$ 到 $+\infty$,积分是不收敛的。显然,要使用数字信号处理器件来实现这些变换,必须把计算范围从无限区间收缩到一个有限区间(注意,在时间域和频率域都应限制于有限范围)。此外,还必须把连续函数改换成离散数据(注意,同样是在时间和频率域两方面都应当是离散样值)。DFT 为傅里叶分析的数字实现提供了理论依据,但是它的计算复杂度大,一般只能用于信号的离线处理。而 FFT 算法的出现为 DFT 的快速实现提供了可能,它利用了原始变换矩阵的冗余性,将长序列的 DFT 分解为短序列的 DFT,大大减少了运算量。FFT 算法虽然在信号理论方面没有什么新的贡献,但它将 DFT 的计算速率提高了大约 $N/\lg N$ 倍。作为一种快速算法,FFT 算法应用领域十分广泛,它被广泛应用在通信、语音、图像、工业测量等领域。

FFT 算法的硬件实现方案主要有以下三种形式: DSP(通用数字信号处理器); FFT 专用芯片; FPGA(现场可编程门阵列)。DSP 器件实现处理速度快、灵活性强,适合于多种复杂的 DSP 算法,例如在通信系统中的信道编解码等算法。采用专用的 FFT 处理芯片,虽然处理速率能达到要求,但其可扩展性较差。FPGA 具有硬件结构可重构的特点,适合于算法结构固定、运算量大的前端数字信号处理。最新推出的 FPGA 产品都采用多层布线结构,具有更低的核心电压、更丰富的 I/O 引脚和更多的逻辑单元(LES),内置嵌入式 RAM 资源,内部集成多个数字锁相环和多个嵌入的硬件乘法器,所有这一切都使得 FPGA 在数字信号处理领域显示出自己特有的优势。

本章将首先介绍有关傅里叶变换的基础知识,着重介绍在进行傅里叶变换时,如何对感兴趣的信号进行一定时间的观察和采样,采样序列被存储后如何从采样序列数据中算出频谱。对于傅里叶逆变换,如何实现从频谱出发算出时间数据序列。接着对 FFT 相关理论进行分析,然后详细讨论如何在 FPGA 上实现离散傅里叶变换的快速实现算法 FFT。

12.2 设计任务的提出与傅里叶变换的理论分析

1. 设计任务

本章的训练任务是对 FFT 算法在 FPGA 上的实现进行研究,包括算法比较、选取和具体的设计、仿真。考虑到训练难度及系统设计的可实现性,系统设计采用 Cooley-Tukey FFT 快速实现算法。在应

用 FPGA 实现 FFT 的具体设计时,应在进行算法比较后,确定设计的总体方案,并完成相应的单元模块设计,其中包括:蝶形运算单元设计、块浮点单元设计、地址产生单元设计、双口 RAM 乒乓结构设计、旋转因子计算、ROM 读取操作设计、时序控制单元设计,最后设计出 512 或 1024 点数的复数的 FFT 处理系统。

2. 设计基本要求

理解 FFT 的原理及应用;并对 Cooley-Tukey FFT 快速实现算法进行分析和研究;在 FPGA 上完成基-4 Cooley-Tukey FFT 算法的设计;用 Quartus II 软件进行仿真此设计,并与 MATLAB 计算的理论值进行误差比较。

3. 提高部分要求

要求在 FPGA 上实现 FFT 算法的一种特殊形式——Goertzel 算法的设计。

12.2.1 离散傅里叶变换

傅里叶变换的离散性和周期性在时域与频域中表现出巧妙的对称关系,即周期性的连续时间函数,其傅里叶变换为离散的非周期函数(傅里叶级数,离散频谱);而非周期的离散时间函数,其傅里叶变换为连续的周期函数(抽样信号的频谱呈周期性)。就傅里叶变换的离散性与周期性之间的关系,可以分为下面 4 种情况进行讨论。

1. 连续时间与连续频率(FTC)

对于连续时间信号的傅里叶变换,首先考虑非周期连续时间信号的频谱。根据定义,连续时间函数的傅里叶变换可以表示为

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft} dt \quad (12.1)$$

相应的傅里叶逆变换为

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft} df \quad (12.2)$$

这种情况下时间函数及其变换函数的波形如图 12.1(a)所示,这里的 $x(t)$ 和 $X(f)$ 都是连续非周期的。

2. 连续时间与离散频率

当连续时间信号为周期信号时,其傅里叶变换具有离散特性,呈现为冲激序列。这种情况下,表示信号频谱的另一种方法是写成傅里叶级数的形式,这对变换可表示为

$$X(kf_1) = \frac{1}{T_1} \int_0^{T_1} x(t)e^{-j2\pi kf_1 t} dt \quad (12.3)$$

和

$$x(t) = \sum_{k=-\infty}^{\infty} X(kf_1)e^{j2\pi kf_1 t} \quad (12.4)$$

其中, $x(t)$ 表示周期为 T_1 的周期性连续时间函数, f_1 表示离散频率之间的频率间隔, T_1 和 f_1 之间满足关系

$$f_1 = 1/T_1 \quad (12.5)$$

这种情况下时间函数及其变换函数的波形如图 12.1(b)所示,其中 $x(t)$ 是连续周期的, $X(f)$ 为离

散非周期的。此时和前一种情形相比, f 用 kf_1 代替, $e^{j2\pi ft}$ 用 $e^{j2\pi kf_1 t}$ 代替, $X(f)$ 用 $X(kf_1)$ 来表示。

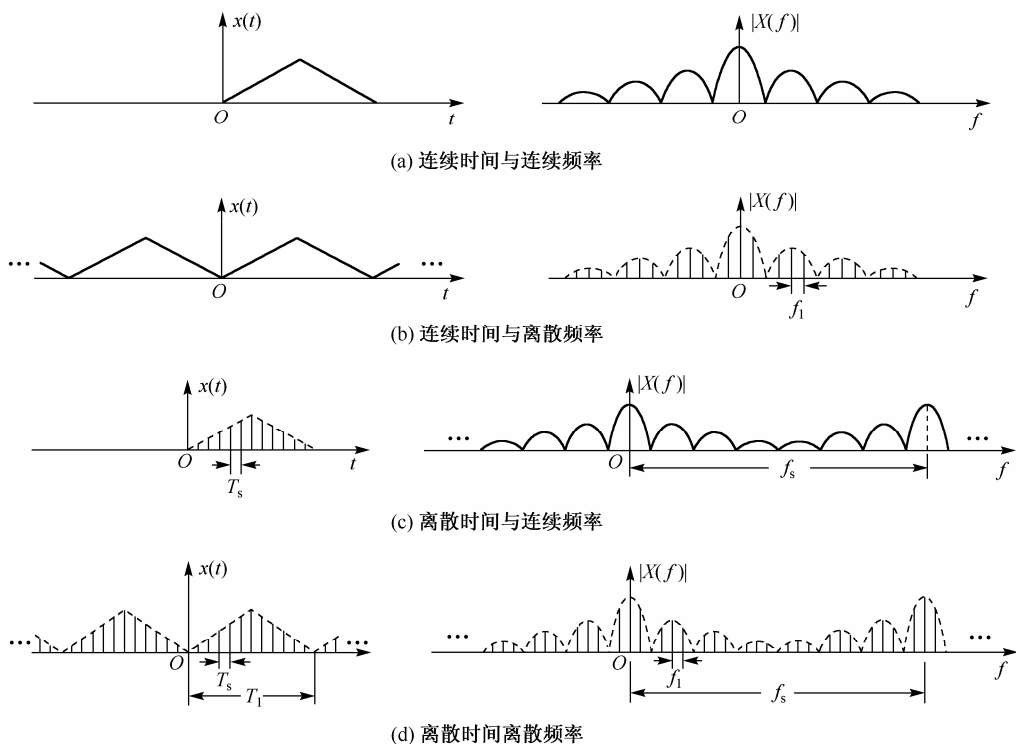


图 12.1 傅里叶变换的各种形式

3. 离散时间与连续频率(FTD)

此种情形即所谓离散时间傅里叶变换(DTFT)。非周期的离散时间函数 $x(nT_s)$ 的傅里叶变换 $X(f)$ 呈现为周期性的连续函数, 它们之间的关系为

$$X(f) = \sum_{k=-\infty}^{\infty} x(nT_s) e^{-j2\pi n f T_s} \quad (12.6)$$

和

$$x(nT_s) = \frac{1}{f_s} \int_0^{f_s} X(f) e^{j2\pi n f T_s} df \quad (12.7)$$

其中, $x(nT_s)$ 表示采样周期为 T_s 的非周期离散时间函数, f_s 表示频域信号重复的周期, T_s 和 f_s 之间满足关系

$$f_s = 1/T_s \quad (12.8)$$

这种情况下时间函数及其变换函数的波形图如图 12.1(c)所示, 这里的 $x(nT_s)$ 是离散非周期的, $X(f)$ 是连续周期的。

4. 离散时间与离散频率(DFT)

这是数字信号处理算法研究的情形, 周期性离散时间函数的傅里叶变换为周期性离散频率函数, 这一对变换可写为

$$X(kf_1) = \sum_{n=0}^{N-1} x(nT_s) e^{-j2\pi nkT_s f_1} \quad (12.9)$$

和

$$x(nT_s) = \frac{1}{N} \sum_{k=0}^{N-1} X(kf_1) e^{j2\pi nkT_s f_1} \quad (12.10)$$

其中, 离散周期时间信号 $x(nT_s)$ 的采样时间间隔为 T_s , $T_1 = NT_s$ 为时间信号的周期, 离散周期频率信号 $X(kf_1)$ 的频率间隔为 f_1 , $f_s = Nf_1$ 为频域中信号的周期。根据前面第(2)和第(3)种情形的推导很容易得到下面的关系:

$$T_s f_1 = 1/N \quad (12.11)$$

和

$$T_1 f_s = N \quad (12.12)$$

其中, T_1 为离散周期时间信号 $x(nT_s)$ 的重复周期, f_s 为离散周期频率信号 $X(kf_1)$ 在频域中的重复周期。

通过对以上 4 种不同情况的讨论, 可以看出, (1) 连续时间信号的傅里叶变换(FTC), 频谱函数 $X(f)$ 是一个具有幅值和相位的复函数, 在时间上是连续的。(2) 和 FTC 不同的是, 离散信号的傅里叶变换(FTD)是一个周期函数, 它以采样频率为周期重复。一方面, 一个周期时间函数有一个离散的频谱(傅里叶级数), 而与此同时, 通过采样形成的离散时间函数的频谱是周期的。(3) FTD 只有乘上采样间隔 T_s 后才与 FTC 在大小与单位上相等。也就是说, 相乘后的频谱是周期的, 如果满足采样定理, 位于基本间隔内的离散信号的频谱与连续信号的频谱相同。(4) 为能在计算机中处理和应用, 必须对连续信号进行采样、量化, 即要从连续时间信号过渡到离散时间信号, 因此, 应从 FTD 过渡到离散傅里叶变换(DFT)。也就是说, 实践中如果进行“数字信号分析”, 实际总是指的是 DFT。所以应从时限信号角度来解释离散时间离散频率的傅里叶变换对。由此看来, 式(12.9)中待分析的信号可以认为是时间长度为 NT_s 的时限信号, 因此对于 $t > 0$ 和 $t < NT_s$, 信号均为 0, 仅对有限个函数值进行累加, 累加的时域范围不再是 $-\infty$ 到 ∞ , 而是 $0 \leq n \leq N-1$, 为了使得频域中计算得到的频谱也是离散的, 需将时限信号在时域中进行周期延拓得到一个周期信号。从而, 上面的变换对可以进一步写成下面的形式:

$$X(kf_1) = \sum_{n=0}^{N-1} x(nT_s) e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (12.13)$$

和

$$x(nT_s) = \frac{1}{N} \sum_{k=0}^{N-1} X(kf_1) e^{j\left(\frac{2\pi}{N}\right)nk} \quad (12.14)$$

为了研究方便, 常将时间间隔 T_s 和频率间隔 f_1 省去, 即

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (12.15)$$

和

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\left(\frac{2\pi}{N}\right)nk} \quad (12.16)$$

式(12.15)和式(12.16)表示的变换对是常用的 DFT 形式, 本章后面将介绍如何在 FPGA 上快速实现它。

12.2.2 傅里叶变换的相关讨论

1. 离散时间信号的傅里叶变换(FTD)

离散时间傅里叶变换的时域信号实际上是使用冲激函数对模拟信号在时域内进行周期性采样,即所谓理想采样。注意,这里讨论的模拟时间信号的持续时间是假设时间向前和向后都是无限长的,然而实际中数字信号分析总是指的是 DFT,时域信号对象应该是时限信号的采样值。由 12.2.1 节中的第 1 种和第 3 种情况讨论结果,可知理想采样后信号的频谱是采样前模拟信号的频谱以周期为 $1/T_s$ 进行搬移的结果, T_s 为时域信号的采样周期。从图 12.1 (a) 和图 12.1 (c) 中可以看出,欲使得采样之后的信号能够完全恢复采样前的信号,要求采样前的模拟信号带宽小于采样频率的一半,即采样频率大于模拟信号带宽的两倍,前者说明了为什么需要在 A/D 采样之前进行滤波,将高频的不需要的信号在采样之前滤除,后者即是所谓的基带采样定理。基带采样定理表明为了能够使用数字信号处理技术,必须滤除掉一些含有信息量较小的频率信息,即在频域中只能处理一部分的频率带宽,从图 12.1 (c) 中可以看到能够处理的带宽为 $-f_s/2 \leq f \leq f_s/2$ 。综上所述,时域信号的采样周期决定了数字信号处理器能够处理的信号带宽。

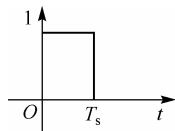


图 12.2 抽样保持电路的矩形脉冲

上面讨论的是使用冲激函数进行理想采样,采样信号的频谱是模拟信号的频谱的周期延拓。但是实际中不可能产生冲激函数进行采样,而是近似认为是使用一个矩形脉冲(如图 12.2 所示)进行采样,即所谓采样保持电路。

此时采样后信号的频谱为

$$X'(f) = \sum_{n=-\infty}^{\infty} X(f - nf_s) \text{Sa}(\pi f / f_s) e^{-j\pi n f / f_s} \quad (12.17)$$

其中 Sa 定义为 $\text{Sa}(\pi f / f_s) = \frac{\sin(\pi f / f_s)}{\pi f / f_s}$ 。

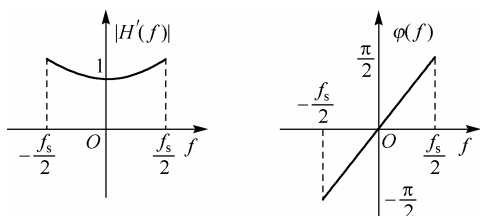


图 12.3 补偿低通滤波器的特性

可以看出,零阶抽样保持信号的频谱的基本特征仍然是 $X(f)$ 频谱以采样频率 f_s 为周期重复,但是要乘上 $\text{Sa}(\pi f / f_s)$ 函数,此外还附加了延时因子 $e^{-j\pi f / f_s}$ 。当 $X(f)$ 频带受限且满足采样定理时,为复原 $X(f)$ 频谱,在接收端需要进行补偿,即所谓 Sinc 校正,补偿滤波器的频域形式如式 (12.18) 所示,补偿低通滤波器的特性如图 12.3 所示。

$$H'(f) = \begin{cases} \frac{e^{j\pi f / f_s}}{\text{Sa}(\pi f / f_s)} & |f| \leq f_s/2 \\ 0 & |f| > f_s/2 \end{cases} \quad (12.18)$$

2. 离散傅里叶变换(DFT)

前面的讨论中提到对信号在数字域中进行处理时必须对信号的带宽进行限制,即数字域只能处理有限带宽的信号,但这只是必要条件,若要能够使用数字信号处理器进行信号处理,还必须对信号的时间长度进行限制,即只能处理有限时间、有限带宽内的信号。离散傅里叶变换是处理有限时间、有限带宽信号的一种算法,是数字信号处理中采用的算法,而其他各种形式的傅里叶变换只能应用在分析之中。

图 12.1(d) 可以看成是对一个周期为 T_1 的波形以采样频率为 $1/T_s$ 进行采样得到的结果, 这里实际处理的信号时间长度是 T_1 。又 $f_1 = 1/T_1$ 成立, 即时域信号的截断处理影响了频域中谱信号的频率分辨率。DFT 算法产生的相邻两个频率成分之间的频率间距 f_1 (亦即频率分辨率 Δf) 与测量时间成反比, $\Delta f = 1/NT_s$, $\Delta\omega = 2\pi/NT_s$ 。也就是说, 时域中可以利用的数据 N 越多, 总的测量时间 $T_1 = NT_s$ 越长, 频谱分辨率 $\Delta\omega$ 或 Δf 也就越高。进一步的分析可以看出, 任何时域信号在频域中的分辨能力和时域的分辨能力之间存在着相互制约的关系, 要想在频域中得到较高的频率分辨率, 则时域中信号必须持续较长的时间, 即不能够分析局部时域信号的精细频率特性; 反之亦然。

综合上述讨论可以得出以下结论: 为了在数字域中处理信号, 需要在频域中和时域中对信号进行截断处理。频域和时域的截断限度可以由采样频率和采样持续时间确定。频域中的截断处理可以通过一个频域中的滤波器在频域中选取信号的主要能量, 时域中的截断处理可以通过一个时间窗函数在时域中选取信号的主要能量。但是, 非理想频域滤波器会对信号的频谱产生混叠效应, 时间的有限长度窗函数会使得信号的频谱产生能量泄露。另外, 非理想冲激采样过程本身也会对信号的频谱产生影响, 所以在恢复信号时需要对这些影响进行校正。

3. 时间窗函数与滤波器设计

1) 时间窗函数的设计

信号的时域内截断可以看做信号与一个时间窗函数的乘积, 因此时间截断信号序列的频谱是原信号序列频谱与时间窗函数谱的卷积, 表现为原信号频谱能量的泄露。与 A/D 前置滤波器的设计类似, 时间窗函数的设计包含了时间窗函数的长度和时间窗函数的形状两方面设计。

为能对时间窗函数长度的影响做出说明, 举例如下: 考虑一个受激谐振回路中衰减振荡的电流过程为 $x(t) = 2e^{-t/\tau} \sin(2\pi ft)$ [如图 12.4(a) 所示], 其中 $\tau = 3\text{ s}$, $f = 0.5\text{ Hz}$, 采样信号频率为 5 Hz , 假设进行理想采样, 采样波形如图 12.4(b) 所示, f 对应的数字频率为 $0.2\pi\text{ rad/s}$ 。

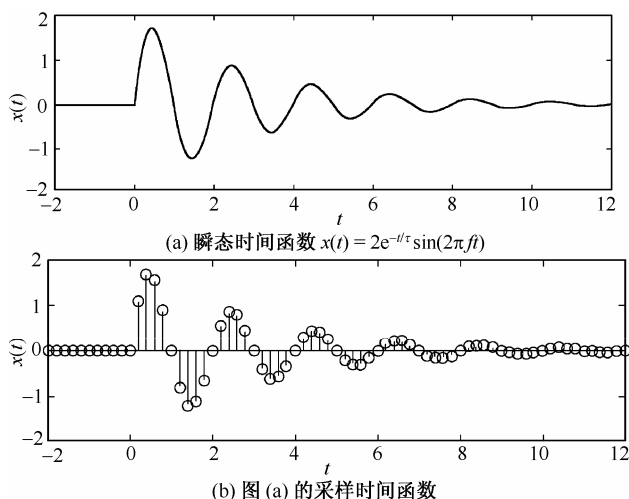


图 12.4 衰减振荡过程信号时域波形

$t < 0$ 时, $x(t) = 0$;

$t > 0$ 时, $x(t) = 2e^{-t/\tau} \sin(2\pi ft)$ 。衰减常数 $\tau = 3\text{ s}$ 。

图 12.5 中给出了采样时间分别为 12 s , 4 s 和 2 s 时所得序列的 DFT 幅度谱, 分别对应了周期信号

的 6 个周期、2 个周期和 1 个周期。若是正弦信号可以持续无穷长时间, 则对应的 DFT 幅度谱是在 $0.2 \pi \text{ rad/s}$ 位置上的冲激脉冲, 但是这里由于采样时间的截断效应, 频域中能量扩散到了冲激脉冲的两边, 采样时间越短, 扩散越厉害, 当采样时间变得和周期信号的周期相当时, DFT 得到的频谱能量峰值位置已经开始偏移, 这会使得频谱的估计产生错误, 其原因在上面的分析中已有说明。当采样定理得到满足, 频谱函数就不会重叠而变形。但采样定理并没有就单个频谱部分相互间能做多好的区分做出断言, 对于频谱分辨率起作用的不是采样频率 f_s , 而是总的测量时间 NT_s 。因此, 在通信和各种数字信号处理应用场合都要求处理的采样信号时间足够长, 才能正确估计信号的频率, 并且根据要求能够区分的频率分辨率选择适当的采样时间。现代数字信号处理为了解决基本 DFT 方法的频率分辨率低的问题, 又提出了很多新的算法, 有兴趣的读者可以参考有关文献。

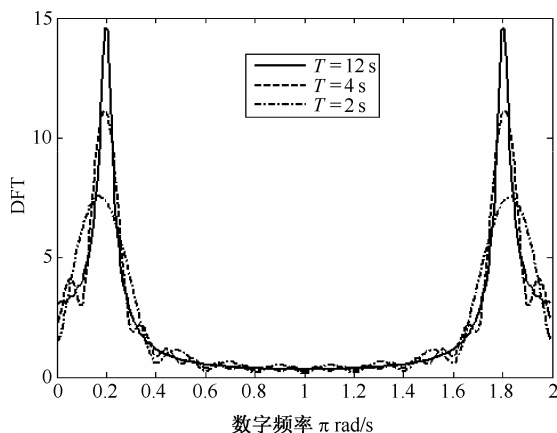


图 12.5 不同采样时间下的 DFT 幅度

以上讨论了窗函数长度对频谱分析的影响, 这里进一步讨论窗函数的形状对频谱分析的影响。图 12.1(d) 可以看成是一个矩形窗函数对一个非周期函数进行加窗的结果, 对矩形窗加窗之后的信号进行采样, 再以周期 T_1 在时间轴的两端进行周期延拓。也就是说, 一个非时限信号经 DFT 后出现频谱的展宽, 原因在于经变换后得到的并非是信号自己的频谱, 而是与窗函数的频谱进行卷积后的结果。因此, 如果测量时间也即时间窗函数的长度不是信号周期的整数倍, 就将出现虚假的谱线。矩形窗函数陡峭的边缘显得特别不利, 矩形窗函数的突然开启和关闭的特性会使其频谱成分在很宽的频带内分布, 根据信号与系统中的知识知道, 矩形窗函数作用于信号之后会使得信号的频谱在很宽的频带内产生能量泄露, 因此需要改进矩形窗函数。常见的 FFT 窗函数包括 Hanning 窗、Hamming 窗、Blackman-Harris 窗和 Kaiser-Bessel 窗等窗函数。各种窗函数的频域波纹曲线如图 12.6 所示。

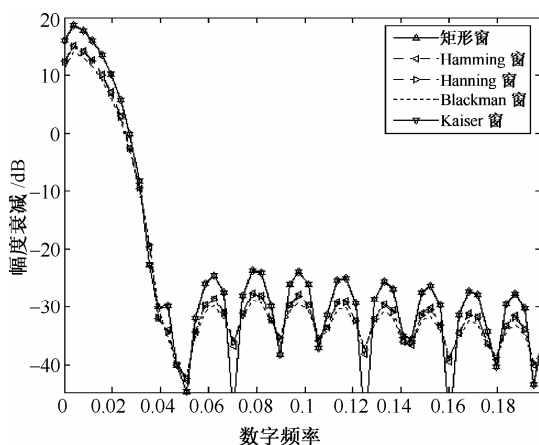


图 12.6 各种窗函数的频域波纹曲线

2) A/D 前置滤波器设计

信号频谱在频域中的截断是通过 A/D 前置滤波器来实现的。图 12.1(c) 中信号在频域中的表示是近似的，因为对于有限长序列的频谱在频域中是无限的，从而采样后得到的序列信号是引入了噪声的，即频谱混叠效应，这可以通过图 12.7(a) 看出。为了减小混叠效应，需要在 A/D 之前设置一个滤波器，以减小带外能量对频带内信号的影响。基带采样定理要求采样信号的频率比信号带宽大于两倍以上即可，但通过图 12.7 可以看出，在实际系统设计中采样频率若仅比信号带宽大两倍是不适合的，因为这会让 A/D 前置滤波器的设计变得非常困难，可能会在采样过程中引入很大的混频干扰。

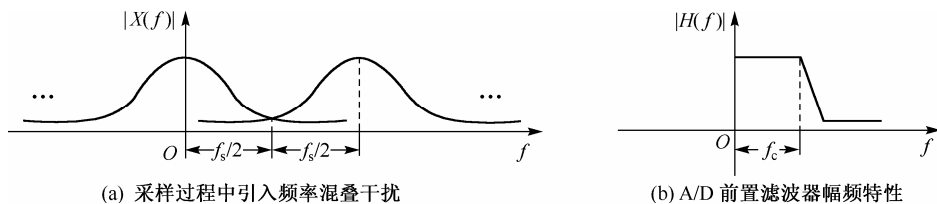


图 12.7 频率混叠效应和 A/D 前置滤波器设计

A/D 前置滤波器的设计包含了滤波器带宽和滤波器种类(即滤波器在通带和阻带的表现)两个方面来设计。对于非周期的模拟低通信号，工程上一般采用 5~10 倍带宽的采样频率，这样可以方便 A/D 前置滤波器的参数设计，另一方面设计者可以根据需要和成本复杂度的考虑选用适合的滤波器形式。所设计的前置滤波器要求能够大大衰减感兴趣频带之外频谱的能量，关于具体的滤波器参数设计读者可以参考有关文献。

12.2.3 基于FPGA的FFT算法设计的相关讨论

1965 年，快速傅里叶变换(Fast Fourier Transform, FFT)的提出，根本地改变了傅里叶变换的地位。FFT 算法成为数字信号处理中最基本的计算技术，其应用十分广泛，在信号处理领域扮演的角色越来越重要，因此研究 FFT 有着重要的现实意义。

在工程实践中，实现数字信号处理主要有三种形式：一种是使用单片机或数字信号处理器(DSP)，通过软件编程来实现；另外两种是应用专用集成电路芯片(Application Specific Integrated Circuit, ASIC)或可编程大规模集成电路来实现。第一种形式利用软件编程来实现，虽然有很大的灵活性，但受 DSP 本身性能及程序指令顺序执行的限制，难以实现高速、大规模运算；专用集成电路芯片或可编程大规模

模集成电路却可以实现很高的运算速度,非常适合高速信号处理系统的应用。

在过去很长一段时间,DSP处理器是DSP应用系统核心器件的唯一选择。尽管DSP处理器具有通过软件设计能适用于实现不同功能的灵活性,但面对当今快速变化的DSP应用市场,特别是面对现代通信技术的发展,DSP处理器在处理速度上略显不足。面向DSP的各类专用集成电路芯片虽然可以解决并行性和速度的问题,但其高昂的开发费用、耗时的设计周期和不灵活的纯硬件结构,使得DSP的ASIC解决方案日益失去其实用性。

FPGA芯片具备在线可编程能力,具有硬件结构在线可重构的特点,适合于算法固定、运算量大的前端数字信号处理。近几年,随着现场可编程门阵列(Field Programmable Gate Array, FPGA)技术的迅速发展,采用并行度更大、速度更快的FPGA芯片来实现FFT已成为必然趋势。FPGA技术的关键就是利用强有力的设计工具来缩短开发周期,提供元器件的优质利用性,降低设计成本,并能够并行处理数据,容易实现流水线结构,且升级简便,提高设计的灵活性,这些都是非常适合实现FFT算法。

因此,自主研发基于FPGA芯片的FFT算法系统,把FFT实时性的要求和FPGA芯片设计的灵活性结合起来,实现并行算法与硬件结构的优化配置,提高FFT处理速度,满足现代信号处理的高速率、高可靠性要求,已成为系统级设计的重要选择方案之一。

与DSP相比,FPGA实现FFT的主要优越性有:

(1) FPGA实现数字信号处理最显著的特点就是高速性能好。FPGA有内置的高速乘法器和加法器,尤其适合于乘法和累加等重复性的DSP任务。

(2) FPGA的存储量大。DSP内部一般没有大容量的存储器,而FFT实时处理运算时需要存储大量的数据,DSP只能外接存储器,这样会使运算速度下降,同时电路也会更复杂和不稳定。目前,高档的FPGA中有巨量的高速存储器,不用外接存储器便可实现FFT实时处理运算,其速度更快,电路更简单,集成度和可靠性也大幅度提高。

(3) FPGA是硬件可编程的,比DSP更灵活。DSP往往需要外部的接口和控制芯片配合工作,FPGA则不需要,这样使得硬件更简单和小型化。

(4) 在比较FPGA和DSP时,一个极为重要的系统参数是输入/输出(I/O)带宽。除了一些专用引脚外,FPGA上几乎所有的引脚均可供用户使用,这使得FPGA信号处理方案具有非常高性能的I/O带宽。大量的I/O引脚和多块存储器可让系统在设计中获得优越的并行处理能力。

需要说明的是,自1965年美国库利(J.W.Cooley)和图基(J.W.Tukey)提出FFT快速算法以来,FFT已有了多种算法。从FFT算法理论的发展上看,主要有两个方向:

(1) 组合数FFT算法,针对FFT变换点数 N 等于2的整数次幂,如基-2算法、基-4算法、基-8算法、基-16算法、实因子算法、分裂基算法及任意组合因子算法,利用系数的周期性和对称性,使长序列的DFT分解成更小点数的DFT,从而大大减少运算工作量。

(2) N 不等于2的整数次幂的算法,以维诺格兰德(S.Winograd)为代表的一类傅里叶变换算法,如Winograd Fourier Transform Algorithm,简称WFTA算法;Polynomial trans-brm Fourier Factor Transform Algorithm,简称PFTA算法,利用下标映射和数论及近代数学的知识,去掉级间的旋转因子,从而减少运算量。

和Cooley-Tukey FFT算法相比,PFTA和WFTA在运算量上占优势,用的乘法器比Cooley-Tukey算法少,但控制复杂,控制单元实现起来相对麻烦。在硬件实现中,需要考虑的不仅仅是算法运算量,更重要的是算法的复杂性、规整性和模块化。控制简单、实现规整的算法在硬件系统实现中要优于仅仅是在运算量上占优的算法。就组合数FFT算法而言,分裂基算法具有一定的优势,综合了基-2和基-4的运算特点,但其蝶式运算结构在控制上要复杂一些,因此基-2和基-4算法是目前普遍采取的两种算法。

12.3 基于FPGA的FFT算法的实现

12.3.1 FFT基础知识

考虑输入序列 $x(0), L, x(N-1)$ 的 DFT 变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (12.19)$$

式中, $W_N^{nk} = e^{j\frac{2\pi nk}{N}}$, $N = N_1 N_2$ 。将时域索引用下式表示:

$$n = An_1 + Bn_2 \bmod N \begin{cases} 0 \leq n_1 \leq N_1 - 1 \\ 0 \leq n_2 \leq N_2 - 1 \end{cases} \quad (12.20)$$

式中, $A, B \in \mathcal{C}$ 是常数。利用这个索引变换可以将输入序列变成下面二维的形式:

$$[x(0) \ x(1) \ L \ x(N-1)] = \begin{bmatrix} x(0, 0) & x(0, 1) & L & x(0, N_2 - 1) \\ x(1, 0) & x(1, 1) & L & x(1, N_2 - 1) \\ M & M & O & M \\ x(N_1 - 1, 0) & x(N_1 - 1, 1) & L & x(N_1 - 1, N_2 - 1) \end{bmatrix} \quad (12.21)$$

相应地将频域中的索引使用下面的式子表示:

$$k = Ck_1 + Dk_2 \bmod N \begin{cases} 0 \leq k_1 \leq N_1 - 1 \\ 0 \leq k_2 \leq N_2 - 1 \end{cases} \quad (12.22)$$

式中, $C, D \in \mathcal{C}$ 是常数。如何根据具体的 N_1 和 N_2 选择 A, B, C 和 D 的一般算法可以参考有关文献。

有了上面的预备知识, 下面讨论 Cooley-Tukey FFT 算法。Cooley-Tukey FFT 算法是所有 FFT 算法中最为通用的, 因为 N 可以任意地进行因数分解。令 $A = N_2$ 和 $B = 1$, 得到下面的映射结果:

$$n = N_2 n_1 + n_2 \begin{cases} 0 \leq n_1 \leq N_1 - 1 \\ 0 \leq n_2 \leq N_2 - 1 \end{cases} \quad (12.23)$$

令 $C = 1$ 和 $D = N_1$, 得到下面的映射结果:

$$k = k_1 + N_1 k_2 \begin{cases} 0 \leq k_1 \leq N_1 - 1 \\ 0 \leq k_2 \leq N_2 - 1 \end{cases} \quad (12.24)$$

将 n 和 k 代入 W_N^{nk} 得到

$$W_N^{nk} = W_N^{N_2 n_1 k_1 + N_1 N_2 n_1 k_2 + n_2 k_1 + N_1 n_2 k_2} \quad (12.25)$$

由于 W 是 $N_1 N_2$ 阶的, 可以得到 $W_N^{N_1} = W_{N_2}$, $W_N^{N_2} = W_{N_1}$, 因此 W_N^{nk} 可以进一步化简为

$$W_N^{nk} = W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_1} W_{N_2}^{n_2 k_2} \quad (12.26)$$

从而可以将一维的 DFT 变为二维的 DFT:

$$X(k_1, k_2) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left(W_N^{n_2 k_1} \sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} \right) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} X'(k_1, n_2) \quad (12.27)$$

式中, $X'(k_1, n_2) = W_N^{n_2 k_1} \sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} = W_N^{n_2 k_1} X''(k_1, n_2)$ 。

下面给出完整的 Cooley-Tukey FFT 算法。 $N = N_1 N_2$ 点 DFT 可以通过下列步骤实现:

- (1) 根据式 (12.22) 计算输入序列的变换;
- (2) 计算 N_2 个长度为 N_1 的 DFT, 得到 $X''(k_1, n_2)$;
- (3) 在第一级变换输出 $X''(k_1, n_2)$ 上应用旋转因子 $W_N^{n_2 k_1}$;
- (4) 计算 N_1 个长度为 N_2 的 DFT;
- (5) 根据式 (12.23) 计算输出序列的索引变换。

12.3.2 基- r Cooley-Tukey FFT 算法

Cooley-Tukey FFT 算法的一个重要优点就是 N 的因子分解可以任意选取, 考虑 $r=2$, $N = r^K = 2^K$, 则时域和频域的索引映射为

$$n = 2^{K-1} n_1 + L + 2n_{K-1} + n_K \quad (12.28)$$

$$k = k_1 + 2k_2 + L + 2^{K-1} k_K \quad (12.29)$$

选择 $r=2$ 带来的好处是可以尽量减少旋转因子的计算 (此时为 1 或 -1), 此时的 DFT 运算一般使用蝶形图的形式表示。图 12.8 给出了按频率抽取 DFT 蝶形运算示意图 ($N=8$)。

图 12.8 中算法在频域中将最初的 DFT 分成更短的 DFT, 这种方法叫做频率抽取 (Decimation In Frequency, DIF) 方法。典型的输入值是按照顺序出现的, 而频率输出值是按位逆序的。还可以构造按时间抽取 (Decimation In Time, DIT) 方法, 首先将所有的输入序列按位逆序分开, 则得到的频率值都是按顺序出现的。

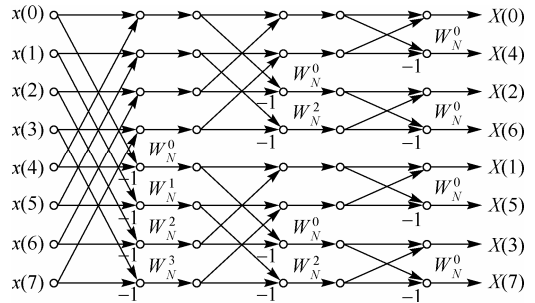


图 12.8 按频率抽取 DFT 蝶形运算示意图 ($N=8$)

12.3.3 基-2 Cooley-Tukey FFT 算法的 FPGA 实现

基-2 蝶形处理器由一个复数加法器、一个复数减法器和一个旋转因子的复数乘法器组成。计算量最大的复数乘法通常是由 4 次实数乘法和 2 次加 / 减法运算实现, 但已有文献中提到只需 3 次实数乘法和 3 次加 / 减法也可以实现复数乘法, 因为可以将一个操作数预先储存。

1. 乘法器的实现

欲计算复数乘法 $(X + jY)(C + jS) = R + jI$, 其中 C 和 S 是可以预先计算得到的, 可以在一个存储表中预先存储 C , $C+S$ 和 $C-S$, 然后计算 $E = X - Y$, $Z = C \cdot (X - Y)$, 则得到 $R = (C - S) \cdot Y + Z$, $I = (C + S) \cdot X - Z$ 。

下面给出 VHDL 实现乘法器的部分代码:

```

LIBRARY lpm;
USE lpm.lpm_components.ALL;
...
ENTITY ccmul IS
...
PORT(clk: In std_logic; --系统时钟
      x_in, y_in, c_in: In std_logic_vector(W-1 downto 0); --X, Y, C 输入
      cps_in, cms_in: In std_logic_vector(W1-1 downto 0); --C+S, C-S 输入
      r_out, i_out: Out std_logic_vector(W-1 downto 0) --R, I 输出
);
END ccmul

ARCHITECTURE behv OF ccmul IS
...
sub_1: lpm_add_sub
GENERIC MAP( ... );
PORT MAP( ... );
...
mul_1: lpm_mult
GENERIC MAP( ... );
PORT MAP( ... );
...
END behv;

```

2. 蝶形处理器的实现

Cooley-Tukey FFT 算法快速实现 DFT 由一系列的蝶形处理单元实现, 如图 12.9 所示。

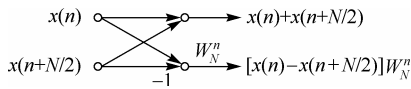


图 12.9 按频率抽选蝶形运算流程图

欲在 FPGA 中使用定点方法实现乘法和加法运算, 计算之前需要进行定标。为了使得整个运算过程中所有运算结果和操作数的定标相同, 图 12.9 所示的蝶形运算可以表示为

$$D_{\text{Re}} + jD_{\text{Im}} = (\text{Re}(x(n) + x(n+M)) + j\text{Im}(x(n) + x(n+M))) / 2 \quad (12.30)$$

$$E_{\text{Re}} + jE_{\text{Im}} = (\text{Re}(x(n) - x(n+M)) + j\text{Im}(x(n) - x(n+M))) / 2 \quad (12.31)$$

其中临时结果 $E_{\text{Re}} + jE_{\text{Im}}$ 再乘以旋转因子。

整个蝶形处理器的 VHDL 参考代码如下:

```

LIBRARY lpm;
USE lpm.lpm_components.ALL;
...
PACKAGE mul_package IS --用户自定义库, 利用前面的复数乘法单元程序
COMPONENTS ccmul
    GENERIC( ... );
    PORT( ... );
END COMPONENT;
END mul_package.ALL;

```

```

...
ENTITY bfproc IS
  GENERIC( ... );
  PORT(clk : In std_logic;
        Are_in, Aim_in, c_in, Bre_in, Bim_in : In std_logic_vector(W-1 downto 0);
        cps_in, cms_in : In std_logic_vector(W1-1 downto 0);
        Dre_out, Dim_out, Ere_out, Eim_out : out std_logic_vector(W-1 downto 0)
        );
END bfproc;

ARCHITECTURE behv of bfproc IS
  ...
  ccmul_1:ccmul
    GENERIC MAP( ... );
    PORT MAP (...);
  END behv;

```

上面给出了基-2 DFT 快速实现方法, 根据 Cooley-Tukey FFT 算法很快可以得到混合基 DFT 快速算法。

12.3.4 离散傅里叶逆变换 (IDFT) 的快速计算方法

只要把 DFT 运算中的每一个系数 W_N^{nk} 变换成 W_N^{-nk} , 最后再乘以常数 $1/N$, 则可以将前述的 FFT 快速算法用来计算 IDFT。例如, 可以直接由按频率抽选的信号流图出发, 把 W_N^{nk} 换成 W_N^{-nk} , 并且在每列(级)运算中乘以因子 $1/2$, 就可以得到 IDFT 的流图。

上面给出的方法虽是思路清晰, 编程方便, 但还是要改动 DFT 模块的程序, 下面讨论一种不用改动 DFT 模块程序的方法实现 IDFT。

由 IDFT 公式可以得到

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} = \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*(k) W_N^{nk} \right]^* = \frac{1}{N} \left\{ \text{DFT} \left[X^*(k) \right] \right\}^* \quad (12.32)$$

这种方法基于 FPGA 实现时可以利用前面得到的 DFT 实现模块, 先将 $X(k)$ 取共轭, 就可以直接利用 DFT 子模块, 然后再将运算结果取一次共轭, 并乘以 $1/N$, 即可得到 $x(n)$ 值。

12.3.5 改进的 DFT 实现方法

上面给出了基本的 DFT 的 Cooley-Tukey FFT 算法在 FPGA 上的实现方法, 随着技术和工程实践的发展, 又提出了一些改进的算法, 可以进一步提高运算速度、精度和减小资源的消耗等。

一般来说, 如果算法使用基-4 DFT 实现, 虽然使用的资源多了一些, 但速度上的好处足以弥补。如果资源充足, 使用基 16、基 8 或基 16/8 复用模块, 速度可以大大提高。一般 FFT 的 FPGA 实现简单框图如图 12.10 所示。

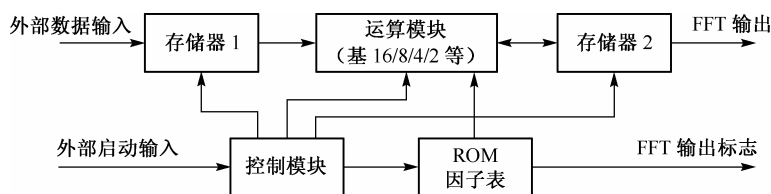


图 12.10 FFT 的 FPGA 实现简单框图

最后, 根据运算过程中对数据位数取位和表示形式的不同, 可以将 FFT 分为浮点 FFT、块浮点 FFT 和定点 FFT。它们在实现时对于系统资源的要求是不同的, 而且有着不同的适用范围。

浮点 FFT 是基于数据表示为浮点的基础之上的, 即数据是由一个纯小数和一个因子组成的, 输入要转成纯小数和因子的浮点表示形式, 所有计算过程中保证结果不会溢出, 而输出要变成所需大小的定点表示形式。只要因子位数足够大, 浮点 FFT 计算是不会溢出的。而定点 FFT 的所有计算过程中都是定点运算, 如果各个过程的截位规则不适当, 很容易出现溢出, 必须要有溢出控制。块浮点 FFT 是介于浮点 FFT 和定点 FFT 之间的一种运算机制, 它是根据过程输入数据的大小, 在计算之前进行控制(数据上移一个比特或下移一个比特, 或乘以一个特定因子), 可以保证不溢出, 但一般也需要溢出控制。

浮点 FFT 运算没有溢出, 信号平均信噪比高, 但由于因子的运算必然导致电路复杂, 实现困难。定点 FFT 运算实现简单, 难以保证不溢出, 需要统计得出合适的截位规则, 否则产生溢出将导致输出结果严重错误。块浮点 FFT 由于每个过程(包括最后输出前)结束后有一统计控制过程, 延时较大, 但是可以保证不溢出而且电路又相对浮点 FFT 来说简单得多。

应根据应用的具体要求, 选择合适的 FFT 实现方式。如果要求精度, 并且要解决频域中严重的单频干扰, 就必须使用浮点 FFT, 使用数据位数很大的定点 FFT 和块浮点 FFT 也能解决这个问题, 但位数的确定十分困难。如果不要求高精度, 逻辑资源和 ROM 比较紧张, 可考虑定点 FFT 运算。如果输入信号在频域集中于几个点上或者对精度要求一般, 则可以慢速处理, 则采用块浮点 FFT 运算就能够保证这几点的信噪比, 而忽略其他点处的信噪比。

12.4 Goertzel算法及其在FPGA上的实现

12.4.1 基本Goertzel算法

很多应用都是只需要进行单音调检测或有限个音频检测, 例如双音多频信号(DTMF)解码、呼叫过程(拨号音、忙音等)解码、频率响应测试(发送一个音调, 同时将结果读回)。在频率响应测试中, 如果在一定频率范围内进行测量, 那么得到的频响曲线中可能会包含丰富的信息, 例如从电话线的频响曲线可以知道线上是否有负载线圈(电感)。FFT 是一种常用来分析系统频谱特性的方法, 但采用 FFT 方法一次运算将给出所有设计频带内的频率特性, 由于计算点数较多、耗时也大, 而且很多嵌入式系统都不具备进行连续实时 FFT 处理的能力, 这时一般采用 FFT 算法的一种特殊形式——Goertzel 算法。

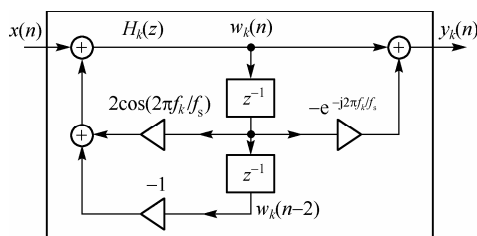


图 12.11 Goertzel 算法的直接 II 型实现

Goertzel 算法一次只计算一个频率点上的频谱特性。采用 Goertzel 算法能得出与常规 DFT 或 FFT 相同的实部和虚部结果。如果需要的话, 还可以从该频率实部和虚部中算出幅度和相位信息。本节将介绍基本的 Goertzel 算法和一种 Goertzel 优化算法及其在 FPGA 上的实现。

Goertzel 算法可以看成是一个高 Q 的窄带滤波器, 如图 12.11 所示, 滤波器的传递函数为

$$H_k(z) = \frac{1 - W_N^k z^{-1}}{(1 - W_N^k z^{-1})(1 - W_N^{k*} z^{-1})} = \frac{1 - e^{j2\pi k/N} z^{-1}}{1 - 2\cos(2\pi k/N)z^{-1} + z^{-2}} \quad (12.33)$$

其中, $H_k(z)$ 为滤波器的传递函数; $W_N^k = e^{j2\pi k/N}$, $k = fN/f_s$; f 为信号频率; f_s 为系统采样频率; N 为样本点数。

从而感兴趣频率的 FFT 值可用下式计算:

$$X(k) = y_k(N-1) = w_k(N-1) - e^{-j2\pi f_k/f_s} w_k(N-2) \quad (12.34)$$

递推部分 $w_k(n)$ 满足关系

$$w_k(n) = x(n) + 2\cos(2\pi f_k/f_s)w_k(n-1) - w_k(n-2)$$

其中, $n=0, 1, 2, L, N-1$; $w_k(-1) = w_k(-2) = 0$ 。

实际应用常常只需要信号的能量而无需相位信息。因此

$$|X(k)|^2 = w_k^2(N-1) - 2\cos(2\pi f_k/f_s)w_k(N-1)w_k(N-2) + w_k^2(N-2) \quad (12.35)$$

为了计算一个频点的能量 $|X(k)|^2$, 递推部分需要进行 N 次迭代运算, 非递推部分只要运算一次。

12.4.2 Goertzel 优化算法

基本 Goertzel 算法在实际应用中存在着两个缺陷。第一, 由前面分析可知 Goertzel 算法含有递归计算, 存在着输出对输入的直接反馈, 因此使用定点算法实现时要先通过理论的方法或统计的方法确定输出的范围, 以保证递归的结果不会溢出。第二, Goertzel 算法实际上是一个 IIR 滤波器, 它的递推运算结构不能充分利用已有的硬件资源。现代高速数字信号处理器内部都集成了常用算法的运算结构单元, 比如乘法累加单元, 它可以有效地实现 FIR 型的滤波器运算。针对 Goertzel 算法的上述缺陷, 下面介绍一种改进计算结构的 Goertzel 算法。

下面推导改进结构的 Goertzel 算法计算公式。令

$$\Delta = 2\cos(2\pi f_k/f_s) \quad (12.36)$$

则递推部分可记为

$$w_k(n) = x(n) + \Delta w_k(n-1) - w_k(n-2) \quad (12.37)$$

其中, $n=0, 1, 2, L, N-1$; $w_k(-1) = w_k(-2) = 0$ 。

利用式 (12.36) 可以得到

$$w_k(n) = \sum_{m=0}^n a_m x(n-m) + w_k(-1) + w_k(-2) = \sum_{m=0}^n a_m x(n-m) \quad (12.38)$$

其中 a_m 满足关系

$$\begin{cases} a_{i+2} = \Delta a_{i+1} - a_i & 0 \leq i \leq n-2 \\ a_0 = 1, a_1 = \Delta \end{cases} \quad (12.39)$$

求解式 (12.38) 可得到

$$a_m = \frac{\Delta + \sqrt{\Delta^2 - 4}}{2\sqrt{\Delta^2 - 4}} \left(\frac{\Delta + \sqrt{\Delta^2 - 4}}{2} \right)^m + \frac{-\Delta + \sqrt{\Delta^2 - 4}}{2\sqrt{\Delta^2 - 4}} \left(\frac{\Delta - \sqrt{\Delta^2 - 4}}{2} \right)^m \quad (12.40)$$

代入 $\Delta = 2\cos(2\pi f_k/f_s)$ 得到

$$a_m = \sin(m+1)\omega / \sin \omega \quad (12.41)$$

其中, $\omega = 2\pi f_k / f_s$; $m = 0, 1, L, n$ 。

由式(12.37)可以得到 Goertzel 算法的改进计算结构:使用两个 FIR 结构来实现式(12.34)中的递归部分,如图12.12所示。

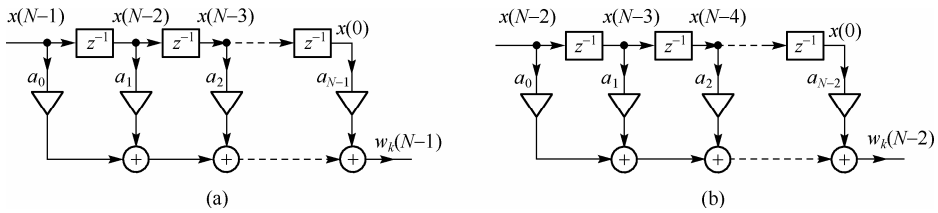


图 12.12 改进计算结构的 Goertzel 算法

$$w_k(N-1) = \sum_{m=0}^{N-1} a_m x(N-1-m) = \sum_{m=0}^{N-1} a_{N-1-m} x(m) \quad (12.42)$$

$$w_k(N-2) = \sum_{m=0}^{N-2} a_m x(N-2-m) = \sum_{m=0}^{N-2} a_{N-2-m} x(m) \quad (12.43)$$

为了能够将改进的 Goertzel 算法应用到定点系统中,需要对数据进行定标,即要预先知道运算过程中结果的最大值,这要求能够事先确定 a_m 在系统参数范围内的取值范围。下面举例说明如何利用 MATLAB 仿真得到 a_m 取定的范围,进而得到算法运算结果的范围。

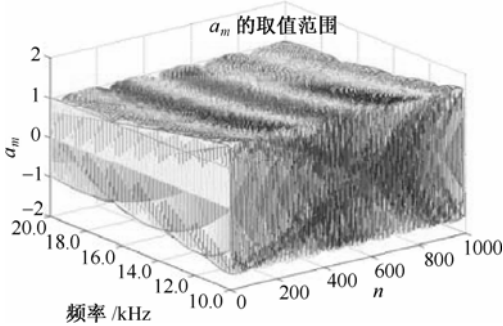


图 12.13 a_m 的取值范围

假设在某通信系统中,采样频率 $f_s = 100 \text{ kHz}$,系统前端带通滤波器的带宽为 $15 \sim 19.5 \text{ kHz}$,计算时使用 10 ms 时间段内的样本,即 1000 点样本。使用 MATLAB 计算出 a_m 的值域,如图12.13所示。

从图 12.13 中可以发现在所给系统参数下 a_m 的范围在 $(-2, 2)$ 。因此

$$w_k(N-1) = \sum_{m=0}^{N-1} a_m x(N-1-m) \leq N |a_m|_{\max} |x(n)|_{\max} \leq 2N |x(n)|_{\max} \quad (12.44)$$

$$w_k(N-2) = \sum_{m=0}^{N-2} a_m x(N-2-m) \leq N |a_m|_{\max} |x(n)|_{\max} \leq 2(N-1) |x(n)|_{\max} \quad (12.45)$$

$x(n)$ 是由模数转换器(ADC)量化得到,ADC 的输出范围决定了它的取值,故由式(12.43)和式(12.44)可以确定递归的结果取值范围,从而可以据此正确完成数的预先定标。

上面使用了解差分方程的方法推导了改进计算结构的 Goertzel 算法,事实上式(12.36)还可以看成是一个 AR(2)过程,下面使用另一种方法将其推广到一般的 AR(p)过程。已知

$$x(n) = -\sum_{k=1}^p a_k x(n-k) + bw(n) = -a_1 x(n-1) - a_2 x(n-2) - L - a_p x(n-p) + bw(n) \quad (12.46)$$

令 $\mathbf{X}(n) = [x_1(n) x_2(n) L x_p(n)]^T$, 其中 $x_1(n) = x(n)$, $x_2(n) = x_1(n-1)$, $\dots$, $x_p(n) = x_1(n+1-p)$, 则可以得到下面的关系式成立:

$$X(n) = \Phi(n, n-1)X(n-1) + \Gamma(n, n-1)w(n) \quad (12.47)$$

其中, $\Phi(n, n-1) = \begin{bmatrix} -a_1 & -a_2 & L & -a_p \\ 1 & 0 & L & 0 \\ M & L & 0 & M \\ 0 & L & 1 & 0 \end{bmatrix}$; $\Gamma(n, n-1) = \begin{bmatrix} b \\ 0 \\ M \\ 0 \end{bmatrix}$ 。即

$$X(n) = \Phi(n, -1)X(-1) + \sum_{i=0}^n \Phi(n, i)\Gamma(i, i-1)w(i) \quad (12.48)$$

其中, $\Phi(k, j) = \Phi(k, k-1) \cdot \Phi(k-1, k-2) \cdot L \cdot \Phi(j+1, j)$, $j = -1, L, k$ 。

因此 Goertzel 算法可以看成是一种特殊的 AR(2) 过程, 其中

$$\Phi(k, k-1) = \Phi = \begin{bmatrix} \Delta & -1 \\ 1 & 0 \end{bmatrix} \quad (12.49)$$

$$\Gamma(k, k-1) = \Gamma = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (12.50)$$

其中, $k = 0, 1, L, n$; $X(-1) = [0 \ 0]^T$ 。

12.4.3 Goertzel 算法在 FPGA 上的实现

本节将根据改进的计算结构提出一种 Goertzel 算法的硬件实现结构。图 12.11 所示的 Goertzel 算法直接 II 型结构计算频点能量时包含了复数乘法运算, 为了达到实时应用, 需要 5 个乘法单元和 4 个加法单元, 而且存在着负数乘法和减法运算, 因此需要使用补码, 这些都会使得硬件结构变得繁杂。

如果使用图 12.12 所示的改进后的计算结构, 即可以得到图 12.14 所示的算法硬件实现框图。可以看到, 它只包含了两个乘法单元和两个加法单元。为了保证实时应用, 数据缓冲区使用了两个存储区相互切换。通过比较, 容易看出本节提出的 Goertzel 算法使用硬件实现时更加简洁高效。

改进后的 Goertzel 算法的计算结构, 大大提高了计算效率。通过上面的分析可以得出下面的结论:

(1) 算法是以存储空间来换取计算效率的提高。这是在时间关键而存储空间不是瓶颈的应用场合中常用的一种提高计算效率的方法。

(2) 算法通过数据的重新组织大大简化了计算结构。Goertzel 算法中含有加、减、乘和复数运算, 改进后的算法只包含有加法和乘法运算, 便于使用硬件实现。这实际上仍是以存储空间的代价换取的。

(3) 核心算法的优化在实时系统开发中占有重要地位。定点处理器一般都要求核心算法采用定点运算, 并采用最适合于硬件的计算结构。另外本算法可使用汇编进行优化, 进一步减小运算时间。

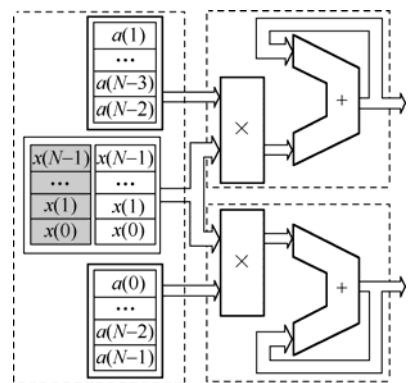


图 12.14 改进后的 Goertzel 算法的硬件实现框图

12.5 本章小结

本章首先介绍了 DFT 的有关基本概念,说明使用数字信号处理有哪些优点和局限性,其优点是在数字域中可以根据需要选择合适的采样频率和持续时间,在频域中和时域中对信号进行截断处理,有利于设计者对系统的精度和成本进行控制;直接使用 DFT 方法进行分析的缺点是其直接计算量很大,不适合实时应用。另外,本章对所谓的时频测不准原理做了简单的说明,即在傅里叶变换的应用中不可能在时间和频率上对信号同时做精确的测量。

本章接着介绍了 DFT 的快速实现原理,即 Cooley-Tukey FFT 算法,并进一步说明了如何在 FPGA 上使用定点方法实现 FFT 算法,给出了 FFT 最基本的蝶形运算单元的实现代码清单和 FFT 算法的 FPGA 实现框图及流程。

最后,本章提到了 FFT 算法的一种特殊形式——Goertzel 算法,它一次只计算信号在一个频点上的信息,这在有些应用场合下有重要意义;本章还进一步介绍了一种改进的 Goertzel 算法计算结构及其 FPGA 实现。

DFT 在各种数字信号处理中起着核心作用,而 DFT 的快速算法——FFT 已成为数字信号处理的最基本的技术之一。因此,本章的训练有着非常重要的意义,希望读者能通过对基本理论的深入学习,自主研发和设计基于 FPGA 芯片的 FFT 算法系统,掌握 FFT 在工程应用及实践中的设计方法。

第 13 章 基于FPGA的其他复杂电路设计与实现

13.1 引言

通过前面章节的学习,读者对 FPGA 的基础知识已经有了一定的了解,熟悉了 FPGA 的开发流程,并且对数字逻辑有一定深度的理解。本章在此基础上进一步介绍 FPGA 在复杂系统中的应用。

本章首先以电子密码锁的设计为例,较为详细地介绍了电子密码锁的原理,在此基础上分析和讨论基于 FPGA 建立简单的密码锁模型的过程,并对模型的实现方法及设计要求做出了说明。希望读者通过训练,能够掌握如何利用状态机来描述一个复杂的行为及如何在 FPGA 上实现状态机的方法。在这个例子中,还详细讨论了如何利用 FPGA 实现人机接口电路,包括键盘按键的消抖、利用 FPGA 控制 LED 的显示等电路模块。

本章接下来以基于全数字锁相环(All Digital Phase-Locked Loop, ADPLL)的频合器的设计为例进行讨论,这个设计是建立在 FPGA 平台基础上的。随着锁相环(Phase Lock Loop, PLL)的理论研究日趋完善,其应用范围遍及整个电子技术领域,如信号处理、调制解调、时钟同步、电子测量、频率合成等应用领域。采用常规器件的 PLL 频率合成器的设计在第 4 章中已有相关的讨论,读者应当对锁相环的基本理论有了一定认识。早期的锁相环全部由模拟电路组成,随着大规模、超高速数字集成电路的发展及计算机的普遍应用,集成锁相环和数字锁相环技术日趋成熟,出现了全数字锁相环。全数字锁相环在结构和工作方式上和模拟锁相环有着很大的不同,但其工作原理和模拟锁相环是相同的。因此,了解和掌握全数字锁相环的设计方法,对于加深锁相环理论的认识有着积极的意义。本章在介绍全数字锁相环结构的基础上,详细分析了如何在 FPGA 中实现全数字锁相环各个模块的设计,在此基础上引导读者完成一个频率合成器系统的完整设计。通过这个例子,读者会发现 ADPLL 在 FPGA 上实现时所有的组成模块都很简单,大多是一些功能各异的计数器,但是将这些简单的器件经过适当地组合,就可以构成一个功能强大的应用系统。需要说明的是,对整个模块(全数字锁相环)进行数学抽象,进而分析它的极限性能的过程是很复杂的,限于篇幅本章仅给出了一般情况下的数学分析。

通过本章的学习,希望读者能够学会使用状态机来分析一个复杂行为,学会使用数学工具来描述复杂的系统,并将复杂的系统分解为若干个简单且易实现的模块,从而完成设计。

13.2 基于FPGA的电子密码锁的设计与实现

1. 设计任务

利用 FPGA 设计和实现电子密码锁。

2. 设计基本要求

- (1) 密码锁具有设置和修改密码功能;
- (2) 采用串行十进制输入方式输入密码,密码位数在 4 位到 10 位之间;
- (3) 输入口令全部正确即可完成开锁动作;
- (4) 错误输入解锁口令次数不能大于 3 次,否则进入死锁状态,并发出报警信号;

- (5) 密码锁的所有动作和状态都有相应的人机交互显示提示, 对输入口令有显示和消隐功能;
- (6) 开锁后应能再次上锁;
- (7) 死锁后的处理机制。

电子密码锁能够事先设置一组或多组的口令, 当破解者(cracker)试探超过一定次数时, 密码锁进入死锁状态并发出报警。与普通机械锁相比, 它有许多独特的优点: 保密性好, 防盗性强, 可以不用钥匙, 记住密码即可开锁等。国内外厂家已推出很多密码锁的专用集成电路, 如 5G058, CIPH0, CIPH1, CIPH2 等芯片。5G058 是通过扰乱按键的顺序来设置口令, CIPH0 等则是通过按键输入口令, 并且可以设置多组口令和超级用户口令等功能。随着技术的发展, 市场又推出了用于密码锁控制的新型芯片如 AT89C2051 等, AT89C2051 有一对能读写串行 E²PROM 的专用接口, 将设置的密码存入 E²PROM 中, 克服了旧式密码锁断电后密码丢失的缺点。它可以同时设置四组密码, 每组有上亿种组合。

本节要求设计一个基于 FPGA 的简单的密码锁模型, 能够进行简单的密码锁工作过程模拟。由于 FPGA 具有 ISP 功能, 当设计需要更改时, 如增加口令位数和更改口令权限管理时, 只需更改 FPGA 中的控制和接口电路, 利用 EDA 工具将更新后的设计下载到 FPGA 中即可, 无须更改外部电路的设计, 这就大大提高了设计的效率。此外, 采用 FPGA 纯硬件构造的电子密码锁电路与微处理器控制的电子密码锁电路相比, 具有更高的工作可靠性。

13.2.1 电子密码锁原理

为提高密码锁的保密性, 充分发挥电子技术的优势, 电子密码锁设计应从以下几个方面予以考虑。

1. 编码总量的设定

电子密码锁随机开锁成功概率定义为

$$p = 1/N_T \quad (13.1)$$

其中, p 为随机开锁成功概率; N_T 为密码编码总量。显然要使密码锁的安全保密性高, 则应使 p 尽可能趋近于零, N_T 尽可能大。但 N_T 越大, 电路相应越复杂, 密码的记忆与操作也越麻烦, 故 N_T 应有合理的上限和下限。下限 N_{TL} 的选择应使密码落在随机开锁可能成功的操作时区以外。若每一次开锁操作时间为 t , 为便于做随机试验, 将 N_T 分成 n 段并期望在 $1/n$ 段的 $1/2$ 处开锁成功。此时 $p = 1/2n$, 则随机开锁试验期望成功的时间为

$$T_r = N_T \times t/2n \quad (13.2)$$

假定系统设计不考虑误码输入的保护, 电子锁在无保护情况下使操作人员可任意做随机开锁试验的时间为 T_{EN} , 则

$$N_{TL} = (2n \times T_{EN}/t) \times x \quad (13.3)$$

其中 x 为最低安全系数。即

$$T_{EN} = (N_{TL} \times t/2n)/x \quad (13.4)$$

显然, 若使 $N_T \geq N_{TL}$, 则 $T_r > T_{EN}$, 使分段随机试验不易成功。

由此可看出, 电子密码锁的编码总量设定是系统设计安全性、保密性的首要技术指标。一般来说, 当 N_T 选定后, N_T 的上限 N_{TH} 原则上越大越安全, 但一般设计时取 $N_{TH} = (10 \sim 10^3) N_{TL}$ 较合理。

2. 编码制式的选取

编码制式应根据 N_T 的大小选取, 可分为以下三种情况。

- (1) 密码各位都可重码: $N_{T1} = a^i$ 。

- (2) 密码非相邻位可重码： $N_{T2}=a \cdot (a-1)^i$ 。
- (3) 密码任何位都不可重码： $N_{T3}=a \cdot (a-1) \cdot \cdots \cdot (a-i+1)$ 。

其中 a 为基数， i 为位数， a 和 i 的选取应满足 $N_{T3} \geq N_{TL}$ ， $a=2, 3, 4, 10, 12, 14, 16$ 。

现以 $a=10, i=6$ 为例，可以算出 $N_{T2}=0.59N_{T1}$ ， $N_{T3}=0.15N_{T1}$ ，所以编码制式考虑能否重码对密码编码总量 N_T 有很大影响。

另外，相同制式下不同的基底对编码总量 N_T 也有很大的影响，而且基底的选择会影响到硬件电路的设计。设计题目要求密码位数为 4 到 10 位，可考虑密码锁采用十进制编码，为简化电路设计，密码锁口令采用对串行脉冲计数的方式输入。

3. 误码输入的保护措施

如前所述，电子密码锁的设计应考虑自身的安全保密性，由于捕捉密码的实验是随机的，若要使 $p=1/N_T$ 趋近于 0，必须采取误码输入保护措施。要求设计的误码输入次数不超过三次，误码输入次数超过三次后系统应关闭主控电路，拒绝大于三次的密码输入，并且系统进入死锁状态。系统正常状态的恢复可以采用以下三种方式。

- (1) 延时后输入二级密码管理方式，即由管理员级密码作为正常输入的开锁密码，然后重新定义开锁密码。
- (2) 使用一个按键使系统重新恢复到正常状态。
- (3) 系统掉电恢复。

一般来说，第一种方式具有管理方便、安全性高等优点，适合于安全性要求较高的场所，相对设计难度也大一些。第二种方式在设计实现上较为方便，但安全性也较低。

13.2.2 系统设计描述

密码锁功能的实现需要一个人机交互的平台，因此首先应对系统的输入、输出信号进行分析以确定键盘、显示单元的设计，进而对系统的信号控制处理模块做出设计描述。根据设计要求可知，电子密码锁设计要求能串行输入数字口令，口令全部正确才开锁，输入三次口令之后还不能正确开锁，进入死锁状态。密码锁的输入键应该有密码设置键、串行数据输入键、开锁键和确认键，输出应该有开锁、死锁提示等。设计还要求密码锁可以对输入的口令消隐，开锁后能再次上锁，进入死锁状态能够通过特殊功能键恢复到常态。因此，密码锁的输入键还应该有消隐功能键、上锁键和复位键。此外，电子密码锁设计要求有 5~8 位 7 段 LED 显示，对所有的按键操作都有相应的功能提示显示，这样密码锁输出信号应该包括 LED 数码管的片选信号和段选通信号。

综上所述，密码锁的输入键应包括复位键、上锁键、密码键、确认键、开锁键、消隐键和串行数据输入键；输出信号应包括死锁提示、按键操作所对应的功能提示及相关信息的显示、LED 的片选信号和段选通信号。选取电子密码锁在设置口令和开锁时输入 8 位十进制数据(题目要求密码位数 4~10

位)，采用串行累加计数的方式输入数据。可参考的密码锁输入、输出信号示意图如图 13.1 所示。具体说明如下：SDIN 为串行数据输入键，电路内部设计一个十进制加法器对串行脉冲计数，计数值作为口令输入。考虑到所有按键的可靠响应，可以使用 RS 触发器及对键盘的低速扫描来减小按键抖动影响。

系统共有 8 个输入键：CLEAR、LOCK、SET、CRACK、CR、SEC、SDIN 和 CLK，各键的功能如下。

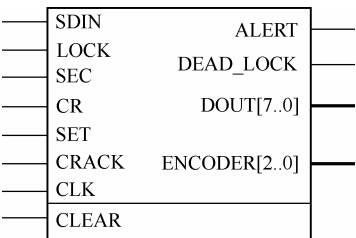


图 13.1 密码锁输入、输出信号示意图

CLEAR: 总复位功能键。任何时候按下该键,系统恢复到原始状态(设置的密码不变),系统上锁,等待输入开锁密码。

LOCK: 上锁键。当正确解锁后,再次按下该键,系统重新上锁,原先设定的密码不变;若用户在正确解锁后没有按下该键,超时一定时间后系统自动重新上锁,且原先密码不变。

SET: 设置密码键。按下该键后,等待输入密码。

CRACK: 解锁键。按下该键后,表示后面输入的数据是解锁密码。当输入开锁密码后,如果输入密码正确,则会在数码管显示“OPEN”字样,否则会提示“AGAIN”或者提示“ERROR”(解锁错误次数已经达到三次,之后系统进入死锁状态,只有按下 CLEAR 键后系统才能恢复正常)。

CR: 输入确认键。无论是在解锁还是上锁时,每输入一位密码值后,按下该键予以确认密码,因此设置密码时必须按下该键才能使该位密码有效。不论是在解锁还是上锁状态,CR 键在确认所输入的密码时,显示模块应显示该次输入的数据对应的是第几位密码。

SEC: 消隐功能键。每按一次该键,输入的密码会在显示和消隐之间切换。

SDIN: 串行数据输入键。

CLK: 系统的控制时钟信号。

有关输出信号的功能定义如下。

ALERT: 解码时输入密码错误小于三次时,显示错误输入次数并给出报警信号,可以用来驱动 LED 和蜂鸣器等。

DEAD_LOCK: 解码时输入密码错误次数等于三次时,显示错误提示符号并给出死锁状态信号,可以用来驱动 LED 或蜂鸣器等(即报警信号由 ALERT 与 DEAD_LOCK 共同产生)。

DOUT[7..0]: LED 数码管的段选通信号,用于显示有关数字和状态提示符。

ENCODER[2..0]: LED 数码管的片选信号。

13.2.3 密码锁各模块的设计

密码锁系统的结构框图如图 13.2 所示。信号处理模块完成输入信号的检测和处理,并输出控制信号。显示控制电路完成输出显示功能。密码口令信号的输入采用按键输入方式。因为机械开关不可避免存在抖动,为避免抖动影响,使用分频器对 CLK(1 MHz)进行分频,分频器的一路输出对按键进行扫描检测,另一路输出控制 LED 的扫描速率。

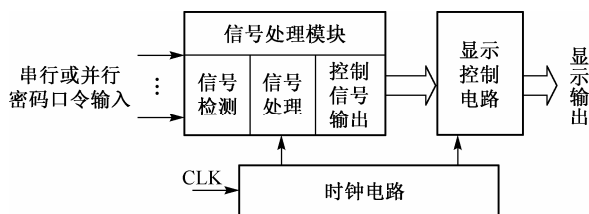


图 13.2 密码锁系统的结构框图

图 13.3 是密码锁系统的顶层示意图,包括信号处理模块 s_pw、显示模块 dis 和分频电路 div1。下将说明各个模块的具体设计。

1. 信号处理模块的设计

信号处理模块完成信号的检测、处理和控制输出,使用状态机来描述这部分的功能。图 13.4 是信号处理模块的状态转换图。

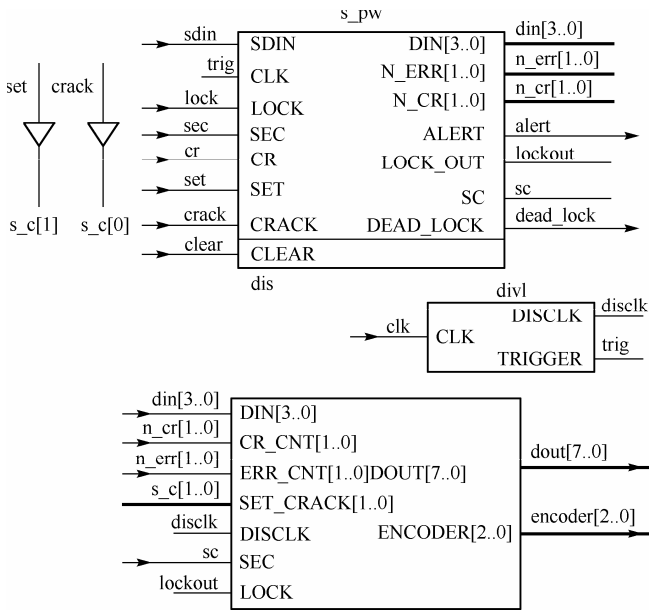


图 13.3 密码锁系统的顶层示意图

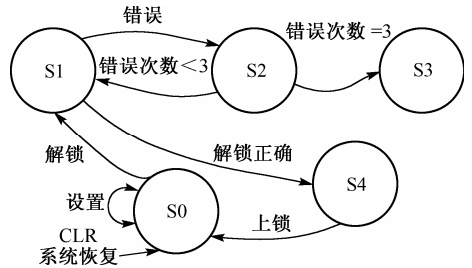


图 13.4 信号处理模块的状态转换图

S0: 初始态, 允许口令设置操作, 按 CLEAR 键后返回 S0 状态。S1: 解锁状态, 按 CR 键时进入解锁状态, 允许不大于三次的解锁错误。S2: 错误状态, 若解锁错误则系统进入错误状态, 判断解锁错误次数是否达到三次, 进而确定下一状态。S3: 死锁状态, 解锁错误次数达到三次, 进入死锁状态并报警。S4: 开锁状态, 解锁正确即进入开锁状态, 按 LOCK 键后上锁并进入初始态 S0。按键检测程序很简单, 这里不再赘述。参考程序如下:

```
parameter st0=3'b000, st1=3'b001, st2=3'b010, st3=3'b011, st4=3'b100;
...
always @(negedge trig)
begin
    if(!i_CR)
    begin
        ...
    end
    else
    begin
        ...
    end
end
always(currentstate)
case (currentstate)
st0:
begin
    if(!i_SET) ...
    else if(!i_CRACK)
        currentstate <= st1;
    end
st1:
begin
    if (!i_CRACK ) ...
```

```

        end
    st2:
        begin
            ...
        end
    st3:
        begin
            ...
        end
    st4:
        begin
            ...
        end
    endcase

```

2. 显示模块设计

显示模块使用 5 位 LED 指示密码锁的工作状态, 并能对输入的口令显示消隐。下面是显示模块的部分参考程序。

```

always @(posedge disclk)
begin
    if(cnt==3'b100)
        cnt<=3'b000;
    else
        cnt<=cnt+1;          //编码计数器
    end
always @(posedge disclk)
    if(cnt==3'b000)
        begin
            encode<=3'b000;
            if(!c) ...
        else
            dout <= 8'b00000000; //消隐显示
        end
    else if(cnt==3'b001)
        begin
            encode <= 3'b001; ...
        end
    else if(cnt==3'b010)
        begin
            encode <= 3'b010; ...
        end
    else if(cnt==3'b011)
        begin
            encode <= 3'b011; ...
        end
    else if(cnt==3'b100)
        begin
            encode <= 3'b100; ...
        end
end

```

程序中 cnt 为编码计数器, 实际上每个值代表一个状态, 表明操作哪一位 LED。

3. 分频电路的设计

图13.5是分频模块的顶层示意图,其中模块FDIV模块是一个20分频的分频器,CLK是系统时钟,频率为1 MHz。

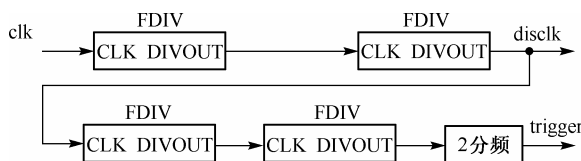


图 13.5 分频模块的顶层示意图

显示的帧频为 $10^6 / (20 \times 20 \times 5) = 500$ 帧/秒。考虑到实际按键的速度,按键检测的时钟频率不能太高也不能太低,实际操作实验表明 3 Hz 左右的扫描信号有很好的效果,按键的扫描频率为 $10^6 / (20 \times 20 \times 20 \times 20 \times 2) = 3.1$ Hz。

下面是 FDIV 模块的参考程序。

```
always @(posedge clk)           //20 分频
    if(d==4'b1001)
        begin
            d = 4'0000;
            dout = ~dout;
        end
    else
        d = d+1;
    assign divout = dout
```

13.3 基于全数字锁相环的频率合成器的设计

1. 设计任务

利用 FPGA 设计一个基于全数字锁相环的频率合成器。系统使用硬件描述语言在 Quartus II 软件平台进行仿真,并最终在 FPGA 平台上实现。

2. 设计基本要求

- (1) 输入频率范围为 1 Hz~10 MHz;
- (2) 输出频率范围为 1 Hz~10 MHz, 频率分辨率为 1 Hz;
- (3) 输出信号相位以输入信号为基准, 输出信号与输入信号的最大相差可控;
- (4) 用户设定频率的最大频差可控。

13.3.1 全数字锁相环的性能分析

4.2 节已经对锁相环的基本原理等内容进行相关介绍,在此不再赘述。

根据锁相环的实现方式和各个组成部分中的信号,可以将锁相环分为模拟锁相环、数字锁相环、混合信号锁相环、全数字锁相环和软件锁相环等。所谓全数字锁相环(All Digital Phase Lock Loop, ADPLL),就是环路部件全部数字化,即采用数字鉴相器(Digital Phase Detector, DPD)、数字环路滤波器(Digital Loop Filter, DLF)、数控振荡器(Digital Controlled Oscillator, DCO)构成的锁相环。与传统的模拟锁相环相比,全数字锁相环避免了模拟锁相环存在的温度漂移、零点漂移和易受电压变化影响等

缺点,在抗干扰能力和可靠性方面都有着明显的优势。随着现场可编程门阵列(FPGA)的工作频率和集成度的提高,用户已经可以在单芯片上实现所需参数的全数字锁相环,常用的全数字锁相环有 74xx297 系列。

数字锁相环的性能分析主要包括以下几个方面。

(1) 线性性能。数字鉴相器、数字滤波器、数控振荡器都是基于数字逻辑来实现的,均是严格的线性系统,从而整个数字锁相环也是一个线性系统。

(2) 捕获性能。锁相环的捕获性能主要涉及两个方面:一是捕获时间的长短;二是捕获带的宽度。捕获性能优劣直接关系到该锁相环的应用范围。能够使环路进入锁定的最大固有频差称为环路捕获带,两倍的捕获带称为捕获范围,使环路完成频率捕获过程所需的时间称为频率捕获时间。捕获带的宽度主要由环路参数决定,而捕获时间不仅与环路的参数有关,还与初始频差密切相关,初始频差越大,需要的频率牵引过程越长。

(3) 相位误差和频率误差。数字锁相环的输出频率由数控振荡器的输入频率分频得到,故数字锁相环的频率误差与数控振荡器的输入频率频偏及分频系数有关。对于一般的频率源稳定度都在 10^{-8} 数量级上(采用晶振作为参考频率源),所以数字锁相环的频率误差很小。数字锁相环的相位最大误差与中心频率 f_0 和数控振荡器的输入时钟频率 f_d 的比值相关。具体计算公式为 $\Delta\theta = (f_0/f_d) \times 2\pi$,其中 $\Delta\theta$ 是相对于输入信号来计算的,如输入信号频率 $f_0 = 2\text{ kHz}$,数控振荡器的输入时钟频率 $f_d = 2\text{ MHz}$,则最大相差为 $\Delta\theta = (1/1000) \times 2\pi$ 。对于中心频率较低的数字锁相环系统,最大相位误差可以控制得很小。最大频率误差与输出频率 f_i 和数控振荡器的输入时钟频率 f_d 的准确度相关。具体计算公式为 $\Delta f = (f_i/f_d) \times \Delta f_d$ 。其中 Δf_d 为时钟频率 f_d 的最大频率偏差,如设定输出信号频率 $f_i = 2\text{ kHz}$,数控振荡器的输入时钟频率 $f_d = 2\text{ MHz}$, $\Delta f_d = 10\text{ Hz}$,则输出信号最大频差为 $\Delta f = 0.01\text{ Hz}$ 。

(4) 最大中心频率。数字锁相环主要是利用比输入中心频率高很多的一个参考频率来控制输出信号的频率与相位。简单来说,实现数字锁相环必须要外接一个高频信号,此信号的频率要求与系统相位误差和中心频率的大小有关。

13.3.2 系统整体设计

本训练要求设计一个基于全数字锁相环的频率合成器,要求实现鉴相器、环路滤波器和数控振荡器等各模块及系统整体的设计。系统使用硬件描述语言在 Quartus II 软件平台进行仿真,并最终在 FPGA 平台上实现。本训练的频率合成器系统主要包括全数字锁相环(数字鉴相器、数字滤波器、数控振荡器)和自适应数字分频器等部分,其系统框图如图 13.6 所示,其工作过程如下。

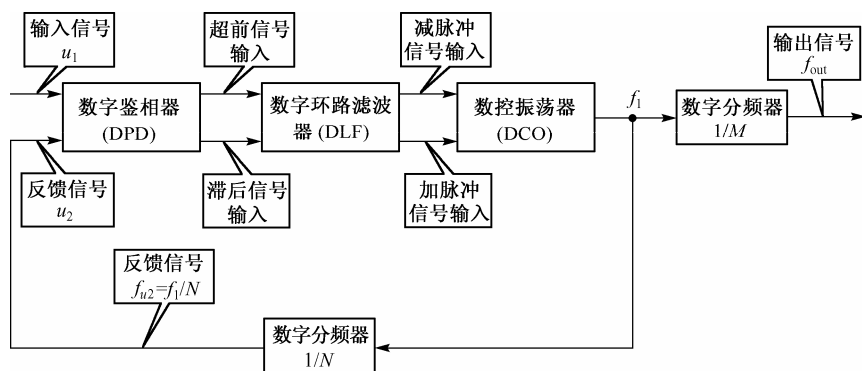


图 13.6 全数字频率合成器的系统框图

(1) 对输入参考信号高电平和低电平分别进行计数,测定输入信号频率,计算获得自适应数字分频器的分频系数 N 。

(2) 输入参考信号和反馈信号在数字鉴相器中进行相位比较,判断反馈信号是超前还是滞后及超前或滞后的时间长短,得到相位差信号。

(3) 数字环路滤波器将相位差信号转换为相应的加减脉冲信号。

(4) 加减脉冲信号控制数控振荡器的脉冲输出频率并调整反馈信号的频率和相位,调整后的数控振荡器的输出信号通过 N 分频反馈到环路再次与输入信号鉴相。

(5) 重复(2),(3)和(4)过程,最终达到相位锁定。锁定状态时,参考信号和反馈信号同频同相。

(6) 根据设定的输出信号频率计算分频系数 M ,系统锁定后,输出与输入信号同相、频率自定义的信号。

本训练要求输入频率范围为 $1\text{ Hz}\sim 10\text{ MHz}$,一般频率合成器因为内部锁相环的捕获带限制,对输入频率范围都有严格限制。设计时,可以通过改进数字锁相环的数字鉴相器和数控振荡器来提高频率合成器的应用范围。设计时可以由数控振荡器的输入时钟信号分频获得,通过设定不同的分频系数来获得对应的不同输出信号频率。

13.3.3 全数字锁相环设计

1. 全数字鉴相器

鉴相器是锁相环的关键部件,它的主要功能是检测输入信号与压控振荡器输出信号的相差和频差,并将它们放大,鉴相器的性能决定着锁相环的精度和稳定度。和模拟锁相环鉴相器不同的是,全数字锁相环中的数字鉴相器全部采用数字电路,电路内部只有“0”和“1”两种工作状态,不存在零点漂移,抗干扰能力强,具有传统模拟鉴相器不可比拟的优越性。

FF 计数器鉴相器是全数字锁相环电路中比较常用的一种鉴相器。此外,奈奎斯特率鉴相器(NPRD)、过零鉴相器、希尔伯特变换鉴相器和数字平均鉴相等也都是全数字锁相环中常用到的鉴相器。图13.7给出了由简单 JK 触发器鉴相器演变而来的 FF 计数器鉴相器结构及其工作波形。参考输入信号 u_1 和数字振荡器 DCO 的输出信号 u_2 是二进制信号,它们用以设置或复位边缘触发的 RS 触发器,触发器的 Q 输出逻辑“1”的时间和相位误差 θ_e 成正比。利用信号 Q 选通高频时钟信号进入(向上)计数器,因此计数器的值 N 和相位误差 θ_e 成正比。图中,计数器在信号 u_1 的正沿处复位,高频时钟信号的频率通常是 Mf_0 , f_0 是参考信号的频率,而 M 是较大的正整数。

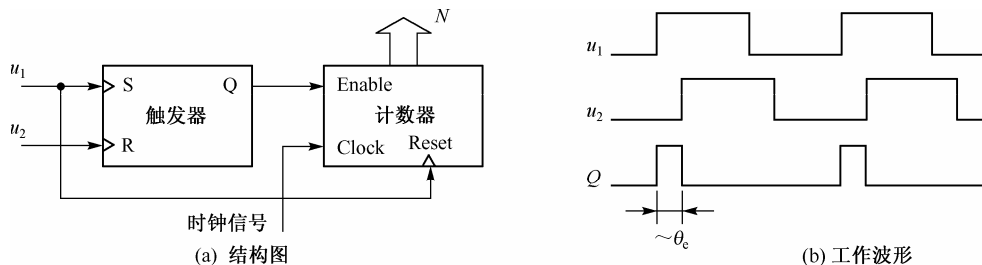


图 13.7 FF 计数器鉴相器结构及其工作波形

2. 全数字环路滤波器

前面提到了多种鉴相器,但并非所有的环路滤波器都可以和各种类型的全数字鉴相器相互组合使用,因此在设计 ADPLL 时必须考虑哪种类型的环路滤波器可以和前面讨论的各种不同鉴相器相配合使用。

最简单的环路滤波器是由普通的加/减计数器 (UP/DN counter) 组成的, 加/减计数环路滤波器最好和上/下脉冲鉴相器一起使用, 它也可以与 EXOR 或 JK 触发鉴相器联合使用。加/减计数环路滤波器的结构和工作波形如图 13.8 所示。首先需要—个脉冲成型网络, 把 UP 脉冲和 DN 脉冲转化为计数器的时钟和方向信号 ($\overline{\text{UP}}/\text{DN}$)。鉴相器每产生一个上脉冲, 加/减计数器的计数值就加 1; 鉴相器每产生一个下脉冲, 加/减计数器的计数值就减 1。环路滤波器的 n 比特并行输出 u_f 记为 N 。因为计数值 N 是向上脉冲和向下脉冲的权值之和 (向上脉冲的权值为 +1, 向下脉冲的权值为 -1), 故可以近似认为这种简单环路滤波器是具有转换功能的积分器。

K 计数器是最常用的数字环路滤波器之一。这种环路滤波器通常和 EXOR 或者 JK 触发器一起工作, 如图 13.9 所示。 K 计数器由两个相互独立的计数器组成, 这两个计数器通常称之为“加 (UP) 计数器”和“减 (DN) 计数器”。实现时, 这两个计数器总是向上计数的, K 为这两个计数器的模, 由“模 K 控制”输入端进行控制, 一般是 2 的整数次幂。计数器时钟信号 (K 时钟) 的频率为 ADPLL 的中心频率 f_0 的 M 倍。 $\overline{\text{UP}}/\text{DN}$ 信号控制 K 计数器的运行, 当该信号为高电平时, DN 计数器开始工作, 同时 UP 计数器的值保持不变; 与之相反, 当它为低电平时, UP 计数器计数而 DN 计数器保持不变。超过 $K-1$ 时, 两个计数器的计数值都会返回到 0, 同时输出进位或借位脉冲信号。UP 计数器的最高位作为进位输出, 而 DN 计数器的最高位作为借位输出, 这样当 UP 计数器的值大于或等于 $K/2$ 时, 进位端为高电平; 当 DN 计数器的值大于或等于 $K/2$ 时, 借位端输出为高电平。进位和借位信号的上升沿可以用来控制振荡器的频率。

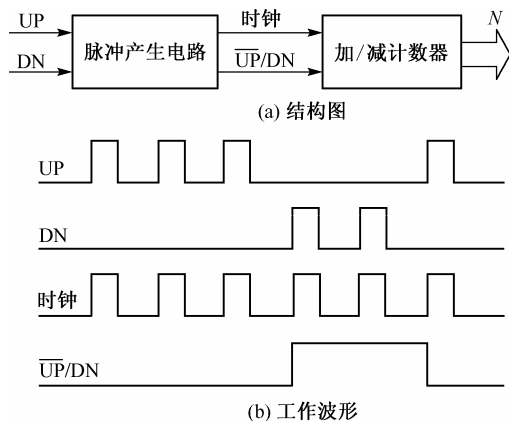


图 13.8 加/减计数环路滤波器的结构和工作波形

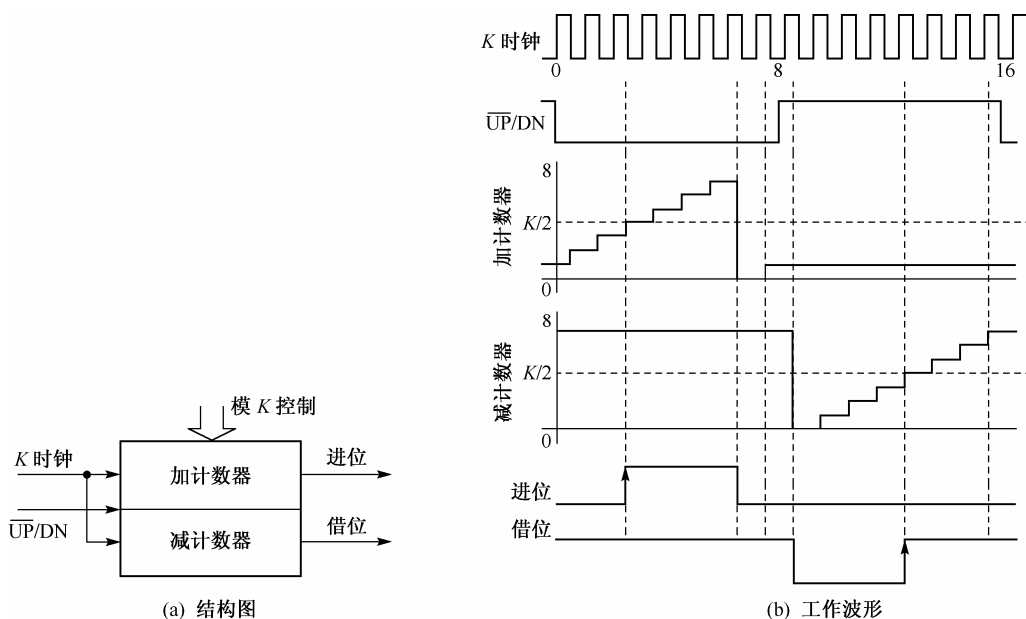


图 13.9 K 计数器环路滤波器的结构

图13.9(b)是假定 K 时钟频率为中心频率的 16 倍 ($M=16$)，设定计数器的模为 8 时 K 计数器环路滤波器的工作波形。其中 $\overline{UP/DN}$ 由鉴相器的输出信号控制。当锁相环的输入信号和输出信号的相位相同时， $\overline{UP/DN}$ 是一个占空比为 50% 的方波，此时进位脉冲和借位脉冲相互抵消。若锁相环的输入信号和输出信号的相位不相同，此时 $\overline{UP/DN}$ 就不再是一个对称的方波信号了。假设在输入信号 u_1 的一个周期内，当 $\overline{UP/DN}$ 信号处于低电平的时间比处于高电平的时间长时，平均来说，UP 计数器就会比 DN 计数器接收到更多的时钟脉冲，单位时间内进位平均数就会比借位平均数大，反之亦然。当 $\overline{UP/DN}$ 端的信号始终为低电平时，UP 计数器就会一直处于工作状态；而当 $\overline{UP/DN}$ 端的信号始终为高电平时，DN 计数器就会一直处于工作状态。

根据工作波形和各信号逻辑关系，给出 K 计数环路滤波器的 Verilog HDL 参考程序如下：

```
module kcount(clk,up,dn,en,b,a,dec,inc);
input  clk,up,dn,en,b,a;
output dec,inc;
reg [3:0] kdata,countdata;
reg [1:0] temp;
reg inc,dec;
always @(b,a)
begin
    temp <= {b,a};
    case (temp[1:0])
        2'b00: kdata <= 4'b0001;
        2'b01: kdata <= 4'b0010;
        2'b10: kdata <= 4'b0100;
        2'b11: kdata <= 4'b1000;
        default: kdata <= 4'b0001; //判断 b, a 端口值取计数 K 值
    endcase
end
always @(posedge clk)
if(en)
begin
    inc <= 0;
    dec <= 0;
    if(up)
begin
        if(countdata<kdata)
            countdata <= countdata+1 ;
        else
begin
            dec <= 1;
            countdata <= 4'b0000; //超前信号模 K 计数
        end
    end
else if(dn)
begin
        if(countdata<kdata)
            countdata <= countdata+1 ;
        else
begin
            inc <= 1;

```

```

countdata <= 4'b0000;          //滞后信号模 K 计数
end
end
end
endmodule

```

另一种常用的数字环路滤波器是所谓的 N -before- M 环路滤波器, 这种滤波器有很强的非线性, 如图 13.10 所示, 这里由于篇幅限制就不分析它的工作原理, 有兴趣读者可以参考有关文献。

对于 N 位并行输入的数字环路滤波器, 可以将对应的模拟传输函数使用数字滤波器实现。假设模拟滤波器的传输函数为 $F(s) = \frac{U_f(s)}{U_d(s)}$, 变换到数字域中为 $F(z) = \frac{U_f(z)}{U_d(z)}$, 因此可以用下面时域形式的数字滤波器实现:

$$u_f(n) = b_0 u_d(n) + b_1 u_d(n-1) + b_2 u_d(n-2) + L - a_1 u_f(n-1) - a_2 u_f(n-2) - L \quad (13.5)$$

上面的 IIR 滤波器如何在 FPGA 实现, 可以参考有关文献。

3. 数字压控振荡器

最简单的数字压控振荡器(DCO)就是可变模 N 分频计数器 DCO, 如图 13.11 所示。数字滤波器的 N 位输出控制 N 分频计数器的分频参数, DCO 的输入时钟为固定的高频振荡器信号。

另一类 DCO 称为增减(ID)计数器, 在实际设计中经常被采用, 如图 13.12(a) 所示。这个数字振荡器与产生进位和借位脉冲的环路滤波器(前面已经讨论)一起使用。该类型数字振荡器有三个输入信号: 输入时钟信号(ID 时钟)、增量(INC)输入和减量(DEC)输入。从图 13.12(a)中可以看出 ID 计数器的工作过程, ID 计数器对进位和借位输入的上升沿敏感。在缺少进位和借位脉冲的情况下, ID 计数器只是简单地把 ID 时钟频率除 2, 每两个 ID 时钟产生一个输出脉冲。

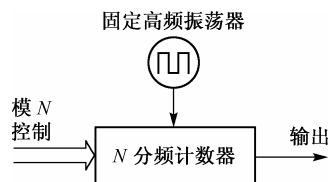


图 13.11 可变模 N 分频计数器 DCO 模块图

需要说明的是, 图 13.12(a) 所示数字振荡器中实际包含了一个翻转触发器(Toggle-FF), 这在图 13.12(a) 中没有画出, ID 计数器的输出(ID_{out})由逻辑功能 $ID_{out} = ID_{clock} \cdot \overline{\text{Toggle-FF}}$ 获得。当在 ID 计数器的 INC 端有一个进位脉冲, 且翻转触发器置高时处理信号; 如果在翻转触发器为低时, 进位端得到进位信号, 那么在 ID 时钟的下一个上升沿且翻转触发器置高时处理信号。翻转触发器在变高之后的下一个 ID 时钟的上升沿开始变低, 并且低电平信号持续两个 ID 时钟周期; 当翻转触发器为高时, 如果进位端有进位信号, 那么在 ID 时钟的下一个上升沿到来时, 翻转触发器置低, 并且在两个 ID 时钟周期内保持低电平, 这使得下一个 ID_{out} 脉冲在时间上提前了一个 ID 时钟周期。相应地, 借位端仅在翻转触发器处在低电平时有效。一旦借位端触发, 在 ID 时钟的下一个上升沿到来时, 翻转触发器将置高并持续两个 ID 时钟周期, 因此, 下一个 ID_{out} 脉冲延迟了一个 ID 时钟周期。

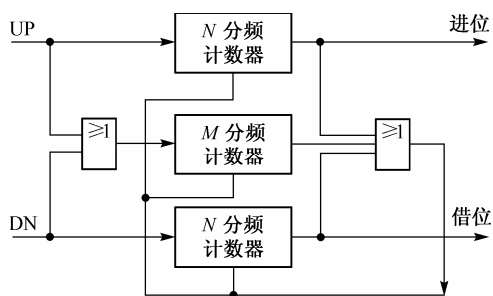


图 13.10 N -before- M 环路滤波器

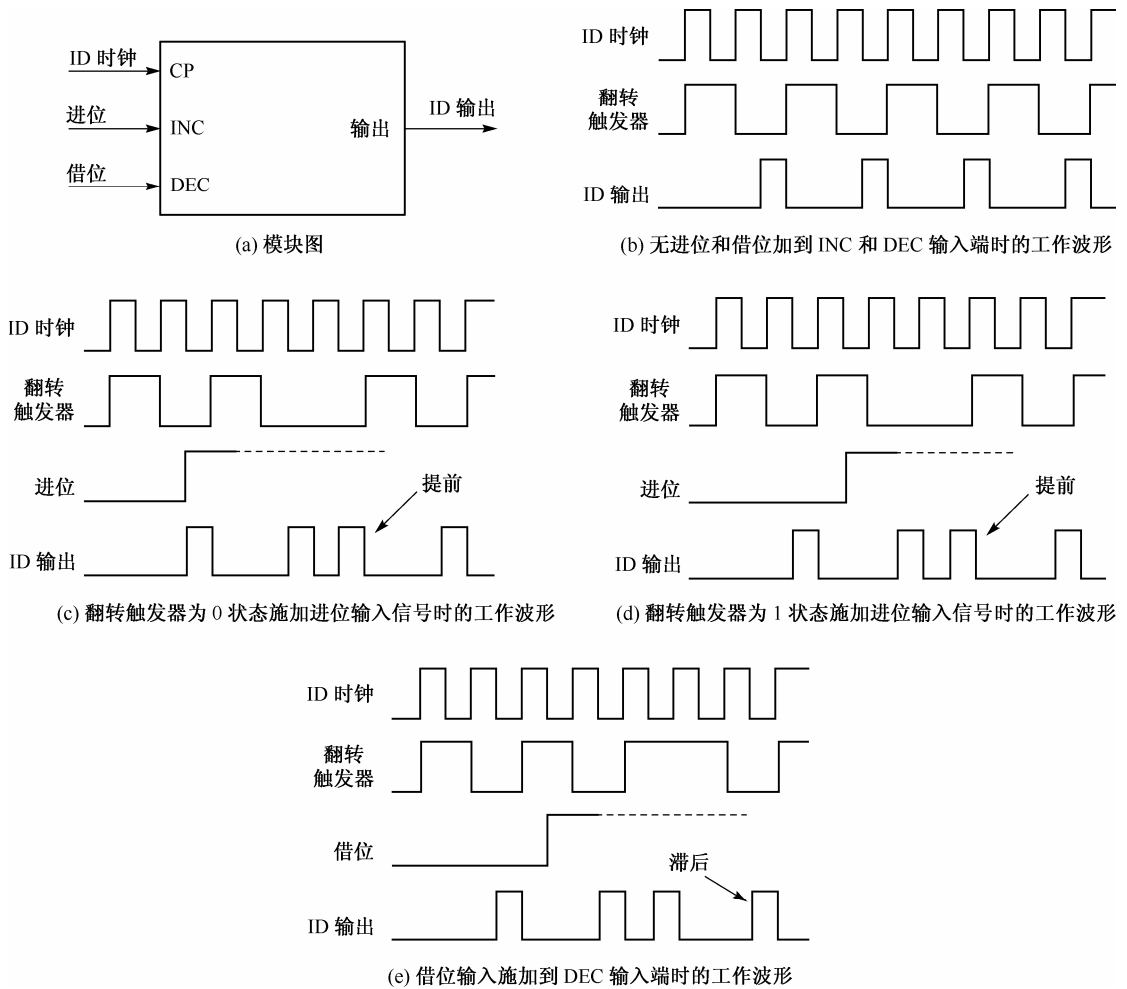


图 13.12 ID 计数器 DCO 的模块图及其在各种情况下的工作波形

上述设计的 ID 计数器的输出频率不可能和 ID 时钟频率一样高，最高只能达到输入时钟频率的 $2/3$ ，最低只能达到输入时钟频率的 $1/3$ ，这将限制 ADPLL 的同步引入范围。可以使用改进的设计方法以增大 ADPLL 的同步引入范围，即在翻转触发器置高，且进位端获得进位信号时，在 ID 时钟的下一个上升沿到来时，翻转触发器置低，并且一直持续置低状态直至进位信号消失，然后在 ID 时钟的下一个上升沿翻转。如此可以使得 ID 计数器在有进位信号时，将不断加入半个时钟周期直至进位信号消失为止。相应地，此方法可以应用到借位端信号。这样，ID 计数器的输出频率高端可以达到接近时钟频率，其输出频率低端可以到 1 Hz。图13.12中 (b)，(c)，(d)和(e)给出了 ID 计数器 DCO 的各种情况下的工作波形。

根据工作波形和各信号逻辑关系，给出 ID 计数器数字环路滤波器的 Verilog HDL 参考代码如下：

```
module dco (IDclk,dec,inc,IDout);
input IDclk,dec,inc;
output IDout;
reg tff,up_set,dn_set;           //tff 为翻转触发器信号
always @(posedge IDclk)
    if(dec ^~ inc)
        begin
```

```

tff <= ~tff;
up_set <= 0;
dn_set <= 0;                                //没有加减信号
end
else
  if(inc & (!dec))
    begin
      up_set <= 1;                            //有加脉冲信号
      tff <= 0;
    end
  else if((!inc)& dec)
    begin
      dn_set <= 1;                            //有减脉冲信号
      tff <= 1;
    end
  end
  assign IDout = (~IDclk)&(~tff);
endmodule

```

4. ADPLL的整体设计

最常用的 ADPLL 结构如图 13.13 所示。它包含两个鉴相器：一个 EXOR 门和一个 JK 触发器，环路滤波器由前面讨论的 K 计数器构成，数字振荡器由前面讨论的 ID 计数器构成，使用图中的 ADPLL 结构实现锁相环功能。

(1) ADPLL 数学模型的建立与分析

结合模拟和数字锁相环的理论分析，可以得到 ADPLL 的相位和误差传递函数，图 13.14 为 ADPLL 的数学模型。

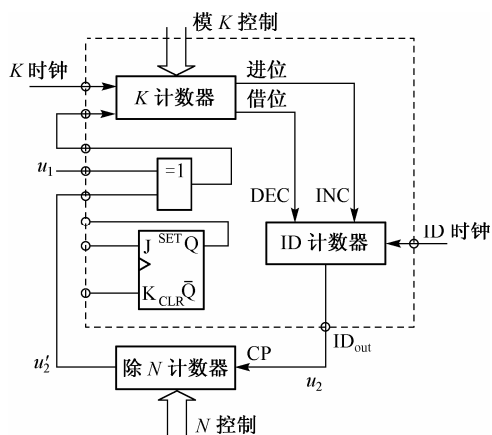


图 13.13 最常用的 ADPLL 结构(电路基于 74xx297 型 IC)

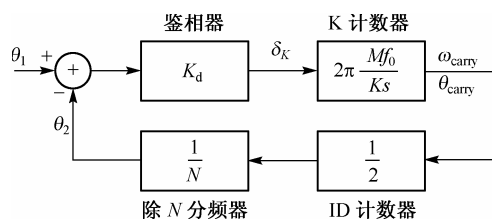


图 13.14 ADPLL 的数学模型

鉴相器可以看做增益为 K_d 的模块，输出占空比因子 δ_K 作为 K 计数器的输入 $\overline{\text{UP/DN}}$ ，

$$\delta_K = (\theta_1 - \theta_2) K_d \quad (13.6)$$

对于异或门鉴相器，相差为 $\pi/2$ 时， $\delta_K = 1$ ，相差为 $-\pi/2$ 时， $\delta_K = -1$ 。因此，对于异或门鉴相器，其增益 $K_d = 2/\pi$ ，同理可得边沿控制鉴相器的增益为 $K_d = 1/\pi$ 。

K 计数器的输出信号频率为

$$f_{\text{carry}} = \delta_K \frac{Mf_0}{K} \quad (13.7)$$

其中, f_0 为锁相环的中心频率。对应的角频率为

$$\omega_{\text{carry}} = \delta_K 2\pi \frac{Mf_0}{K} \quad (13.8)$$

相位是角频率对时间的积分:

$$\theta_{\text{carry}} = \int \omega_{\text{carry}} dt \quad (13.9)$$

所以 K 计数器的相位传递函数为

$$K(s) = \frac{\theta_{\text{carry}}(s)}{\delta_K(s)} = 2\pi \frac{Mf_0}{Ks} \quad (13.10)$$

对于基本 ID 计数器 DCO 可以看成是增益为 $1/2$ 的模块(改进型的 ID 计数器也可以得到相应的结果)。除 N 计数器可以看成是增益为 $1/N$ 的模块, 因此 ADPLL 的相位传递函数为

$$H(s) = \frac{\theta_2(s)}{\theta_1(s)} = \frac{\omega_0}{\omega_0 + s} \quad (13.11)$$

其中, $\omega_0 = \frac{K_d \pi M f_0}{KN}$ 。

系统的相差传递函数为

$$H_e(s) = 1 - H(s) = \frac{s}{\omega_0 + s} \quad (13.12)$$

可以看出, 上述 ADPLL 为一阶系统, 时间常数为 $\tau = \frac{1}{\omega_0} = \frac{KN}{K_d \pi M f_0}$ 。为了获得最小波纹, 对于异或门鉴相器和边沿控制鉴相器, K 的模值分别取为 $M/4$ 和 $M/2$ 。另外减小 N , 可以减小 ADPLL 的稳定时间。

(2) ADPLL 的同步范围

简单分析可以得到 ADPLL 工作的频率范围, 当 K 计数器一直上升计数时, ADPLL 产生最大的输出频率, 此时进位脉冲频率为

$$f_{\text{max}} = f_0 \frac{M}{K} \quad (13.13)$$

因为每个进位脉冲给 ID 计数器输入增量都会使 ID_{out} 信号增加 $1/2$ 周期, 所以 ID 计数器的输出频率增加

$$\Delta f_{\text{IDout}} = f_0 \frac{M}{2K} \quad (13.14)$$

又因为 N 分频计数器使频率减少为 $1/N$, 所以 ADPLL 可以处理的最大频率偏移为

$$\Delta f_{\text{H}} = f_0 \frac{M}{2KN} \quad (13.15)$$

式(13.15)即是 ADPLL 能够同步的最大频率范围。式(13.15)要求 ID 计数器能够处理所有的进位和借位信号, 这要求 $N > N_{\text{min}} = 3M/2K$ 。若 N 不能满足这个条件, 此时 ADPLL 的同步范围将被限

定在

$$\Delta f_H = f_0/3 \quad (13.16)$$

(3) 设计步骤

图13.15给出了ADPLL的设计流程。根据要求,可以选择使用JK触发器作为鉴相器,选择 $M = 2K$ 使得波纹最小。当选择 $M = 2N$ 时,可以使得设计的电路尽量简单,此时 K 计数器和 ID 计数器的时钟完全相同,因而可以从同一信号源获取。图13.15中第6步是计算 $K = f_0/\Delta f_H = 4$,此时 $N = K = 4$,满足 $N > 3M/2K = 3$,以上即完成了相应参数的设计。具体的硬件描述语言程序可以参考本节前面介绍的相关内容和其他的相应参考文献。

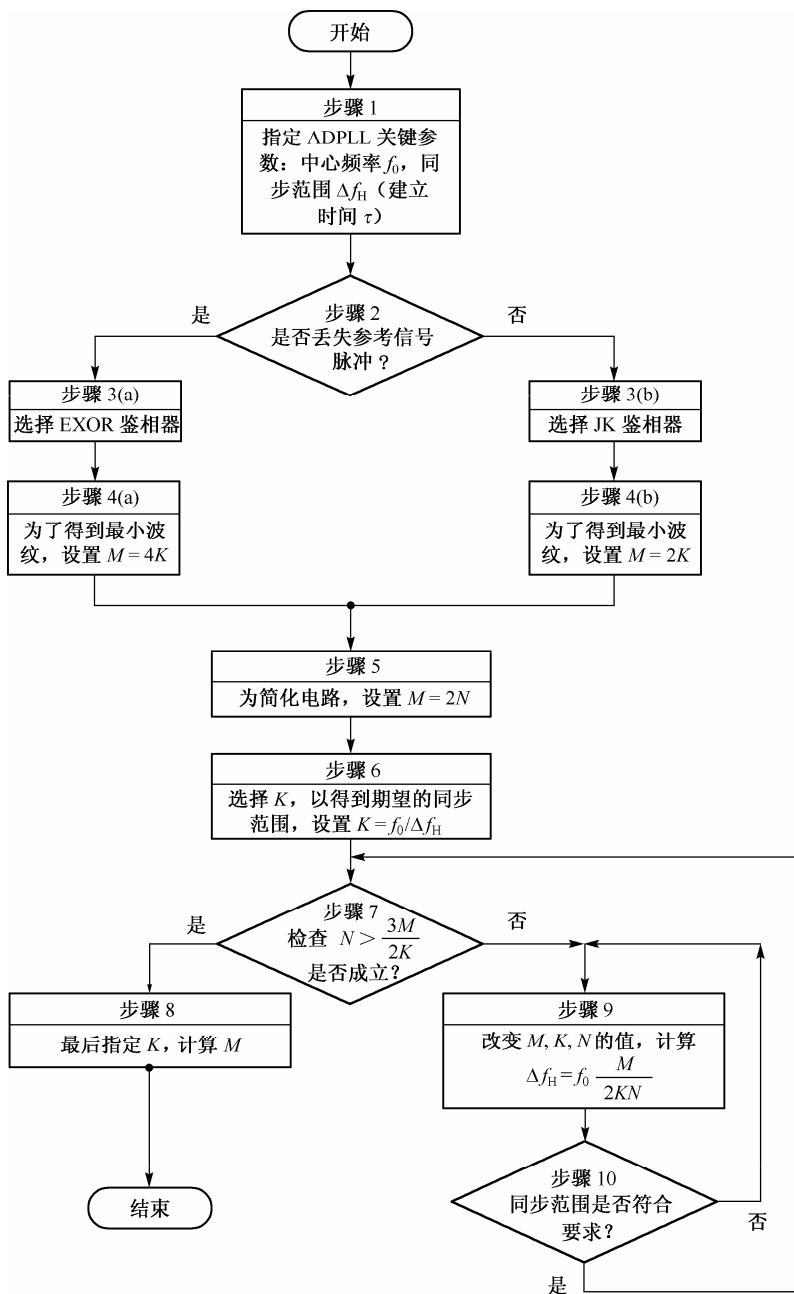


图 13.15 ADPLL 的设计流程

13.3.4 自适应频合器的设计

所谓自适应频合器,即能够自适应地根据输入信号的频率调整环路参数,使得输出信号和输入信号同步。在本训练的频率合成器设计中,分频器的设计与实现是系统设计的重点之一。由于本训练要求的输入频率范围为 1 Hz~10 MHz,而一般频率合成器因为内部锁相环的捕获带限制,对输入频率范围都有严格限制,因此需要设计自适应分频器。改进传统频率合成器的实现方法,使得输入信号频率范围不再受到锁相环同步带的限制。自适应分频器的具体实现方法是,首先对输入频率的高电平和低电平分别计数,得到输入信号的频率信息;然后根据计数值获得对应的分频系数 N ,分频后获得反馈信号 u_1 ,这样可以保证反馈信号与输入信号频率相同。对于整个系统来说,不同输入频率的信号和反馈信号均能实现频率跟踪,即系统克服了一般数字锁相环同步带的限制。

13.3.5 频合器系统仿真分析测试

在 FPGA 设计过程中,仿真是对设计进行检验的可靠方法,一般基于 FPGA 前端设计的仿真包括功能仿真和时序仿真。功能仿真是最基本的仿真实验,其主要目的是验证设计文件的逻辑功能是否正确,是否满足设计要求。时序仿真是通过观察和计算各信号之间的时间延迟,以检验时序是否正确,时序仿真可以有效地分析设计中可能存在的竞争与冒险,从而确定设计的实际工作性能。

本训练的仿真包括局部模块仿真和系统仿真。局部模块仿真是对环路各组成部件的仿真,包括数字鉴相器模块、数字环路滤波器模块、数控振荡器模块等,其目的是验证环路各部件的功能是否能够实现,同时也测得一些性能指标。系统仿真是把环路各部件组合起来成为一个系统后,再对这个系统进行仿真,其目的同样是验证系统的功能是否能够实现,并能够得到系统的一些性能指标。系统仿真时主要是测量锁相环的捕获和锁定特性,整个系统的输出信号频率和相位。其方法是把锁相环的各个部件连接成为一个闭环系统,然后在环路的输入端加入预设信号,观察环路输出端的波形。判断环路是否锁定,除了观察输出信号与输入信号的相位是否稳定地保持一致这一方法之外,还可以通过观察中间信号来分析和判断系统设计中可能出现的问题。

1. 频率合成器基本功能验证

设输入信号的频率为 2 kHz,输出频率选定为 200 Hz、2 kHz、20 kHz 和 200 kHz 四种情况做仿真,以验证系统是否能正确捕获和锁定输入信号的频率与相位,以及能否输出预设定频率的信号。

以输入信号的频率 2 kHz,设定输出频率 2 kHz 为例进行讨论,验证频率合成器中数字锁相环的基本功能,仿真截图如图 13.16 所示。其中 a2、a1 选择输出频率,在 IDclk 和 Kclk 为 2 MHz 的情况下,00、01、10、11 分别对应输出频率 200 Hz、2 kHz、20 kHz、200 kHz。Flag_lock 为 0 代表追踪过程,为 1 代表锁定, u_1 为输入信号, u_2 为反馈信号,allout 引脚为输出信号引脚。

对图 13.16 进行放大,可得图 13.17。

由图 13.16 和图 13.17 可以看出,在系统初始阶段,反馈信号 u_2 超前于输入信号 u_1 ,在经过大约 2.5 ms 的追踪时间后,系统输入与输出达到锁定,将系统入锁标志信号 flag_lock 置高。此时输出信号 allout 与输入信号 u_1 同频同相 [相差 $\Delta\theta \leq (1/1000) \times 2\pi$]。

2. 输入信号频率突变

输入信号频率突变的仿真是为了验证系统输入信号频率发生变化时系统的跟踪特性。例如,将系统输入频率由 2 kHz 在某一时刻(如 5 ms)变成 2.5 kHz,输出频率依然设定为 2 kHz, IDclk 和 Kclk 依

然设定为 2 MHz 的情况进行仿真。

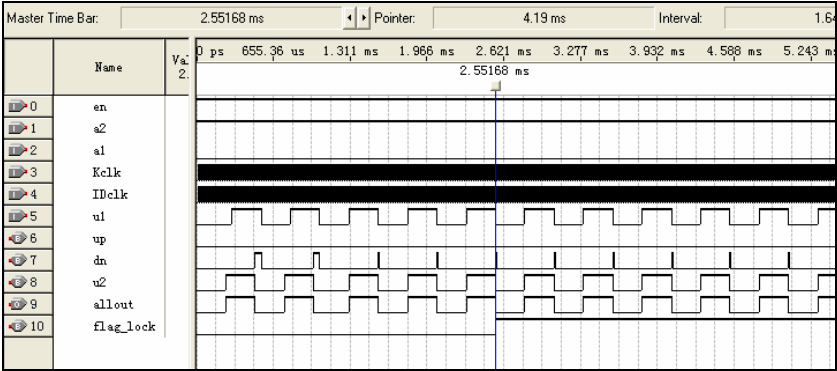


图 13.16 系统整体仿真截图 1

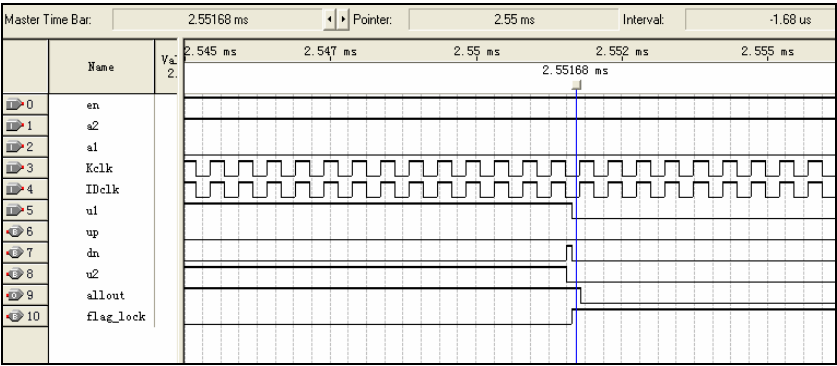


图 13.17 系统整体仿真截图 1 的放大图

3. 输出信号频率范围

对系统输出频率高端进行仿真验证。例如,可以先对系统输入频率为 1 MHz 的信号,设定输出频率为 1 MHz 的情况进行仿真,设定 Kclk 和 IDclk 频率为 20 MHz;再对输入频率为 1 MHz 的信号,设定输出频率为 10 MHz 的情况进行仿真,设定 Kclk 和 IDclk 频率为 80 MHz。

4. 输出信号频率步进

对系统输出频率低端及输出频率步进进行仿真验证。例如,系统输入信号频率为 1 Hz,设定输出频率为 4 Hz,设定 Kclk 和 IDclk 频率为 1 kHz,得到仿真截图如图13.18所示。

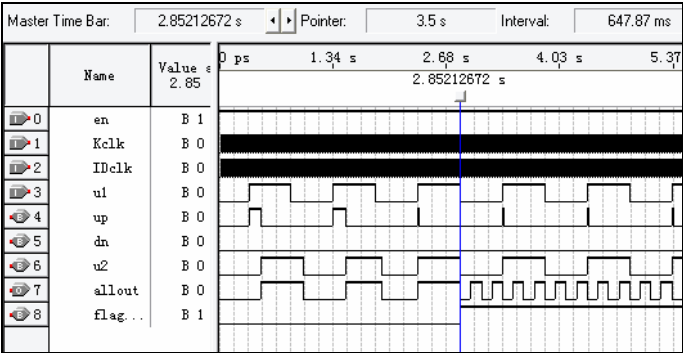


图 13.18 系统整体仿真截图

由图13.18可以看出,在系统初始阶段,反馈信号 u_2 滞后于输入信号 u_1 ,在经过大约 2.8 s 的追踪时间后,系统输入、输出信号相位锁定,将系统入锁标志信号 `flag_lock` 置高,从此时刻开始,系统输出端 `allout` 将稳定输出与输入信号 u_1 同相的 4 Hz 信号。

13.4 本章小结

本章主要介绍了 FPGA 的两种复杂应用设计,即在 FPGA 上进行电子密码锁设计和基于全数字锁相环的频合器的设计。电子密码锁设计本身有很高的逻辑复杂度,而且设计涉及到很多基于 FPGA 的人机接口电路设计,在 FPGA 的应用中有一定的代表性,本章主要讨论了电子密码锁系统的一个基本模型,读者在熟悉了 this 基本模型的基础上可以进一步改善并丰富密码锁的功能,如口令的保护、多等级密码设计及融合已有的密码算法等。

全数字锁相环在通信、自动控制等场合有着重要而广泛的应用,根据前面的介绍,在各种不同的应用场合,ADPLL 的各个组成部分可以采用不同的实现方式,因而对它们的分析也会有很大的不同,但是它们的基本原理和上面的分析还是相似的。基于全数字锁相环的频合器的设计运用了自顶向下的系统设计方法,在系统整体设计完成后,把频率合成器的整体电路分解为功能模块单元,即数字鉴相器、数字环路滤波器、数控振荡器和自适应数字分频器等 4 个部分,然后逐一设计和实现每个功能模块单元。当所有的单元都调试通过以后,再做系统级的仿真测试,这样做的好处是,出现问题容易解决,而且对于复杂模型往往由于代码的抽象度很高,使得在代码编译时不仅耗时且占用很多资源,并容易出错。从较低层的功能模块单元入手应该是较好的设计实现方法。

一门新技术的应用是无止境的,也是会超出当初它被开发出来的预期。限于篇幅,作者在这里不可能将本章中所给出的两个设计实例的所有设计方案和实现方法都给出来,而且所给出的参考设计方案不一定就是最优的,只希望读者能通过对典型应用的学习和理解,能够深入地思考设计过程中的问题,分析对比不同方法之间的差异和优劣,知道如何从系统设计的难点入手,在整体上把握设计要素,获得科学、合理的设计思想和分析复杂问题的方法。

第14章 基于FPGA的JPEG图像压缩

14.1 引言

近二十年来,科学技术取得了飞速的发展,由计算机技术所带来的信息革命使人类由工业化的社会进入到了信息化的社会。多媒体技术和 Internet 互联网技术的广泛应用加速了全球高速公路的建设,与此同时需要存储、传输和处理的信息的数量成指数级地增加。例如,考虑全高清分辨率(1920×1080)、全屏幕显示(Full Screen)、真彩色(True Color 24 位)、全动作(Full Motion, 25 帧/秒)的图像序列,播放 1 秒钟的视频画面数据量为 $1920 \times 1080 \times 3 \times 25 = 155\,520\,000$ 字节,相当于存储一亿五千万个汉字所占用的空间。如此庞大的数据量,给图像的传输、存储及读出造成了难以克服的困难。在基于 Internet 的多媒体通信中,即使面向最终用户的局域网速率达到了 1000 Mb/s,如此高的传输速率仍无法完全满足视频通信等应用的要求,因此需要研究各种图像压缩算法。

图像压缩机制通常分为两种:有损压缩和无损压缩。在无损压缩中,人们关心的是精确重建,获取没有信息丢失的数据。无损压缩是对文件本身的压缩,其原理和其他数据文件的压缩一样,是对文件的数据存储方式进行优化,采用某种算法表示重复的数据信息,文件可以完全还原,不会影响文件内容,对于数码图像而言,也就不会使图像细节有任何损失。由于无损压缩只是对数据本身进行优化,所以压缩比例有限,无损压缩通常被用于文本文件的压缩中。有损压缩是利用了人类对图像或声波中的某些频率成分不敏感的特性,允许压缩过程中损失一定的信息,虽然不能完全回复原始数据,但是所损失的部分对理解原始图像的影响较小。有损压缩换来了大得多的压缩比,广泛应用于语音、图像和视频数据的压缩,常见的声音、图像、视频压缩基本都是有损的。有损压缩的标准过程是变换编码。其基本思想是用一个和原来不同的数字基来表示数据,在这种新的表示下,数据的相关性能够显露出来或被拆开;在这种新的基下,大部分的系数都接近零,可以忽略,于是可以将余下的信息存储在一个较小的数据包中,对非零数据进行无损的编码来实现。

JPEG (Joint Photographic Experts Group) 是为静止的灰度(色彩)连续变化的图像编码压缩而制定的标准。JPEG 定义了两类压缩算法:一种是离散余弦变换(DCT);另一种是空间预测。DCT 算法偏重于图像的视觉效果,用该算法得到输出图像的视觉效果相当好,其与原图像的视觉效果几乎一样。鉴于 FPGA 系统具有高速、高可靠性、设计开发周期短、在线可重构性等许多优点,适合于开发各种满足用户要求、灵活性高的图像压缩处理及传输设备。本章中将主要介绍和分析如何利用 FPGA 器件来实现分辨率为 1920×1080 的灰度图像的 JPEG 压缩。

14.2 JPEG图像压缩的原理与实现

1. 设计任务

利用 FPGA 构造 JPEG 图像压缩算法,实现分辨率为 1920×1080 的灰度图像的有损压缩。

2. 设计基本要求

- (1) 图像压缩比大于 151;
- (2) 图像压缩峰值信噪比大于 30 dB。

14.2.1 JPEG图像压缩算法分析

JPEG 图像压缩算法流程大致如下：首先把大小为 1920×1080 的图像帧分成 8×8 的像素块，然后对每一块做 DCT，DCT 变换模块把数据从空间域变换到频率域，并去除数据的冗余度；量化器用加权模块产生对人眼优化的 DCT 系数；“Z”形扫描提取直流和低频分量以利于图像压缩；Huffman 编码将量化的 DCT 系数的熵最小化，完成图像信号的压缩。简单地说，这种方法的思想是：把大量的冗余数据简化为较小的、真正意义上的数据，删除极少视觉效果的信息，并且利用数据空间特征进一步压缩数据。JPEG 图像压缩编码流程如图 14.1 所示，其解压缩编码过程如图 14.2 所示。

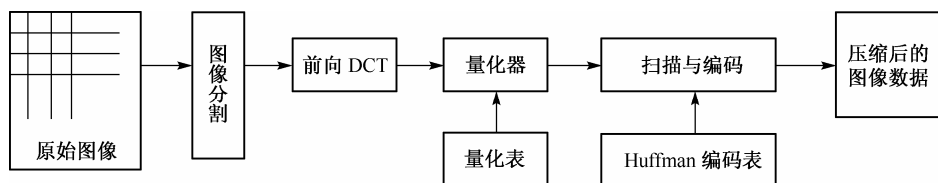


图 14.1 JPEG 图像压缩编码流程

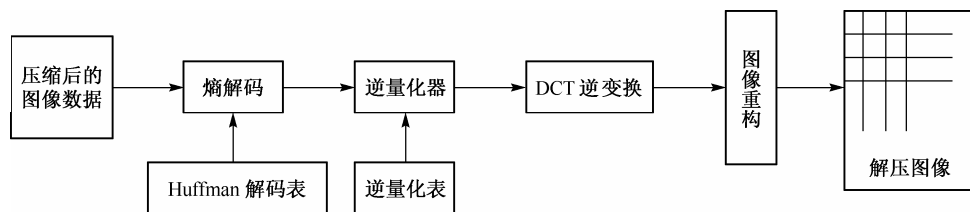


图 14.2 JPEG 图像解压缩编码流程

基本的 JPEG 图像压缩系统算法为顺序编码，其主要步骤如下：

(1) 将原始图像分割为 8×8 的像素块，根据从左到右、从上到下的光栅扫描方式进行排序，对每个图像块独立地做 DCT 变换。

(2) 所有的变换系数都用由用户定义的规格化数组表示，且规格化数组对所有的图像块固定不变。规格化数组中每个元素都是一个 12 位的二进制整数，作为每幅图像都需要的信息头的一部分被传送到接收端。心理和视觉对比敏感函数可以作为开发规格化数组的指南，根据其感知的重要性对每个系数加权。

(3) 二维 DCT 数组的左上角的系数被当做直流系数，它正比于空间块的平均亮度。在量化之后，以无损 DPCM 方案对这一系数编码。

(4) 交流系数量化后产生许多零，特别是对较高的频率。为利用得到的这些零的优点，采用一种“Z”形扫描方式重新排序，把二维 DCT 系数排成一维向量。这种重排系数近似地按照它们能量的平均值由大到小地排列，也是按照空间频率增加的方式排列，其目的是创造零值的长行程。

(5) 对交流系数进行游程编码，向量中每个非零系数首先被合并为一个 8 位二进制数 I ($0 \leq I \leq 255$) 其形式为 $I = \text{“NNNNSSSS”}$ ，其中最后 4 个有效位“SSSS”定义了系数幅度值的范围。在基本系统中，范围为 K 的系数幅度值落在 $(2^{k-1}, 2^k - 1)$ 和 $(-2^k + 1, -2^{k-1})$ 之间，且 $0 \leq k \leq 10$ 。在给定范围后，必须把附加的 K 位数据发送出去，以指明在这个范围中系数的符号和幅度。在合成值中的 4 个高位“NNNN”，给出了当前系数相对先前非零系数的位置，即非零系数间的行程，由“NNNN”指定的行程范围是 $0 \sim 15$ 。如果表示连续 16 个 0 系数行程用符号 $I = \text{“11110000”}$ ，若 0 系数行程超过 16，则要用多个编码，并且最多只用 3 次编码 ($I = \text{“11110000”}$)。此外，一个特殊的符号 ($I = 0, \text{EOB}$) 表示图像块向量中剩余

的系数都是零。因此, 编码符号集共包含 162 个符号, 即 10 个范围×16 种行程值+2 个附加符号。每个图像块的输出符号用 Huffman 编码, 并且允许有附加位。附加位用来指定符号和提取每一范类中系数的幅值。

(6) 对彩色图像的每个成分单独编码。

(7) 在解码端, 编码的二进制位流被 Huffman 解码, 量化的 DCT 系数的二维数组被恢复。把这个二维数组乘以对应的规格化矩阵, 则每个系数被反规格化。对反规格化的数组进行 DCT 逆变换, 就得到了近似的原始图像块, 即重构的图像块。重构图像块的误差依赖于量化的度量, 它是受规格化矩阵控制的。

根据 JPEG 编码的原理, 要利用 FPGA 器件来实现分辨率为 1920×1080 的灰度图像的 JPEG 压缩, 可采用自顶向下的设计方法。首先把 JPEG 编码器分成几个子模块, 每个子模块再向下分成若干个更小的子模块, 这些子模块级联起来, 共同完成 JPEG 编码任务。本设计中 JPEG 编码器分成几个大模块来实现: 图像分割模块、离散余弦变换模块、量化与游程编码模块、熵编码模块等。在这些模块的基础上, 再进行顶层的控制设计。本节接下来将对各个模块的设计与实现进行详细分析。

14.2.2 图像分割

原始图像的大小为 1920×1080, 分割为 8×8 的像素块, 并作为二维 DCT 的输入, 具体图像分割示意图如图 14.3 所示。对单色图像, 分割后只有亮度(或灰度)图像块序列。而对彩色图像的处理, 需要把 RGB 分量分别提取进行图像分割, 然后把各个分量按照次序排放, 分别为 R 分量图像块序列、G 分量图像块序列和 B 分量图像块序列。

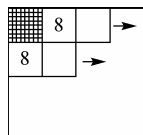


图 14.3 将图像分割为 8×8 的像素块示意图

14.2.3 离散余弦变换

离散余弦变换(DCT)是一种与傅里叶变换紧密相关的数学运算。在傅里叶级数展开式中, 如果展开的函数是实偶函数, 那么其傅里叶级数中只包含余弦项, 将其离散化可导出余弦变换, 因此成为离散余弦变换。

DCT 是正交变换, 可以将 8×8 的图像的空间表达式转换为频率域, 只需要少量的数据点表示图像; DCT 产生的系数很容易被量化, 因此能获得很好的块压缩。DCT 算法有很好的性能, 如它可采用快速傅里叶变换进行高速的运算, 因此它在硬件和软件中都很容易实现; 而且 DCT 算法是对称的, 所以利用 DCT 逆算法可以用来解压图像。

在图像压缩中, 采用 8×8 的图像块, 其原因有两个: 计算量和像素之间关系的数量。许多研究已经表明, 从统计学上平均来说, 在 15 或 20 个像素之后, 像素间的相关性开始下降。也就是说, 一列相似的像素通常会持续 15 到 20 个像素那么长。

采用 8×8 的图像块, 二维 DCT 正变换的公式如下:

$$F(u, v) = \frac{1}{4} c(u) c(v) \left[\sum_{i=0}^7 \sum_{j=0}^7 f(i, j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} \right] \quad (14.1)$$

DCT 逆变换采用如下公式:

$$f(i, j) = \frac{1}{4} c(i) c(j) \left[\sum_{u=0}^7 \sum_{v=0}^7 F(u, v) \cos \frac{(2u+1)i\pi}{16} \cos \frac{(2v+1)j\pi}{16} \right] \quad (14.2)$$

其中

$$\begin{cases} c(i) = c(j) = 1/\sqrt{2} & i, j = 0 \\ c(i) = c(j) = 1 & i, j \text{ 为其他值} \end{cases}$$

$f(i, j)$ 表示位于 (i, j) 坐标处的像素亮度。这样经过 DCT 正变换所得到的结果就是 DCT 系数，我们把 $F(0, 0)$ 称为直流系数(DC)，而把其他系数则称为交流系数(AC)。二维离散余弦变换核具有可分离特性，所以正变换核和逆变换核均可以将二维变换分解成一系列一维变换(行、列)进行计算，即

$$F(u, v) = \frac{1}{2}c(u)\left[\sum_{i=0}^7 G(i, v)\cos\frac{(2i+1)u\pi}{16}\right]$$

(14.3)

$$G(i, v) = \frac{1}{2}c(v)\left[\sum_{j=0}^7 f(i, j)\cos\frac{(2j+1)v\pi}{16}\right]$$

(14.4)

二维 DCT 变换分解成一维 DCT 变换的示意图如图14.4所示，并且其运算结果与行列变换先后次序无关。

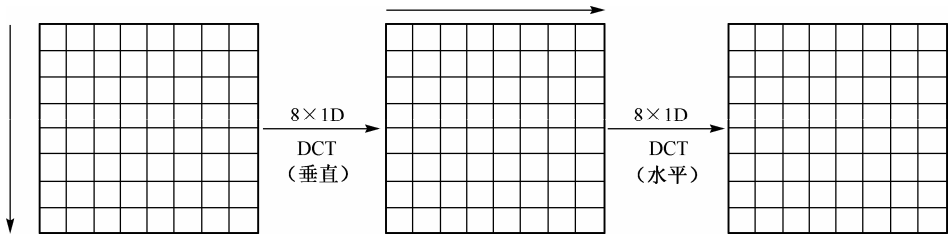


图 14.4 二维 DCT 变换分解为一维 DCT 变换的示意图

对一维 DCT 变换有多种算法，本设计可采用一种快速离散余弦变换，计算量较小，易于硬件实现，其具体实现过程如图14.5所示。

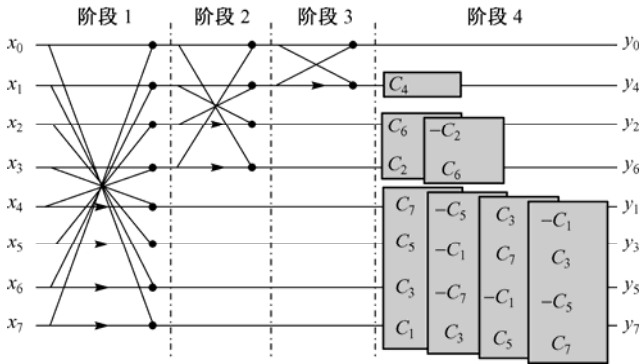


图 14.5 一维 DCT 快速变换的实现过程

在图14.5中，输入数据 $x_i(i = 0,1,L,7)$ ，经过运算后得到的数据 $y_j(j = 0,1,L,7)$ ，必须再对 y_j 做处理，即 $y_0/2\sqrt{2}, y_j/2 (j = 1, 2,L,7)$ ，也就得到了一维 DCT 系数。另外，对运算过程中涉及到的常数 $1/2\sqrt{2}$ 和 $c_x = \cos(x\pi/16) (x = 1,2,L,7)$ ，根据精度要求来取其量化值。在 FPGA 芯片中可采用二端口输入加法单元、二端口输入减法单元、二端口输入乘法单元、四端口输入乘加单元和八端口输入乘加单元来实现上述运算。

二维 DCT 变换是在一维 DCT 变换基础上实现的，具体流程如图14.6所示。

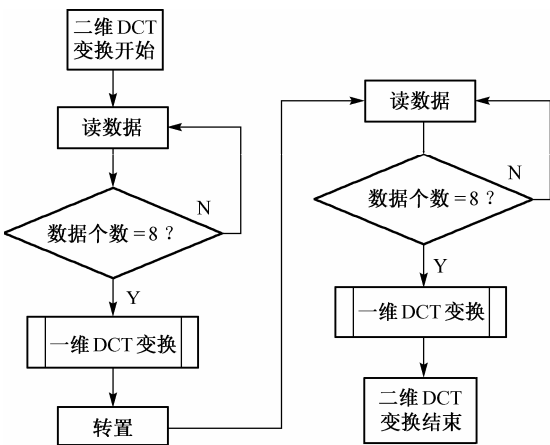


图 14.6 二维 DCT 变换流程图

本设计使用 HDL 语言完成 8×8 的像素块二维 DCT 变换，并进行仿真。为了方便调试，此处选取原始图像中一个 8×8 的像素块(如表14.1所示)，在 Quartus II 7.2 开发环境中分别进行功能仿真，得到的仿真结果可参考图14.7。

表 14.1 8×8 亮度抽样像素块

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	139	144	149	153	155	155	155	155
8	144	151	153	156	159	156	156	156
16	150	155	160	163	158	156	156	156
24	159	161	162	160	160	159	159	159
32	159	160	161	162	162	155	155	155
40	161	161	161	161	160	157	157	157
48	162	162	161	163	162	157	157	157
56	162	162	161	161	163	158	158	158

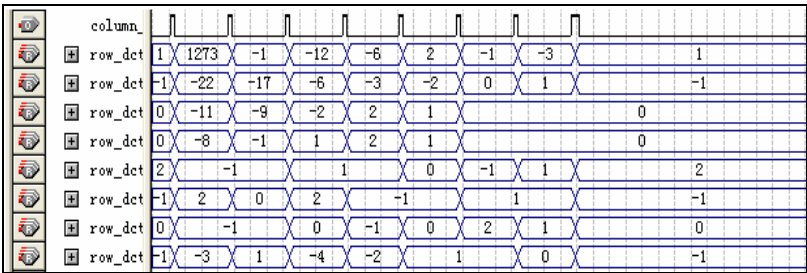


图 14.7 8×8 的采样图像像素块的二维 DCT 系数仿真结果

14.2.4 量化与游程编码

DCT 将 8×8 的图像块变换到频域中时，数值集中在矩阵左上角，即 8×8 的图像经过 DCT 变换后，低频分量都集中在左上角，高频分量分布在右下角。由于低频分量包含了图像的主要信息(如亮度)，而高频成分与之相比就不那么重要了，所以可以忽略高频分量，从而达到图像压缩的目的，这样就减少了数据量，因为在定义整个矩阵时，只有小部分是真正有意义的。很明显，如果删除一些离 DC 系数较远的值，就会损失一些信息，但是由于人眼对低频较敏感而对高频不太敏感，在大部分情况下人眼能够推知信息或感觉不到丢失的信息。因此，量化的主要目的是去除高频部分。简单地说，就是降低数据精度以减少存储数据所需的位数，从而达到数据压缩目的的过程。量化过程试图确定什么信息可以安全地消失，而没有任何明显的视觉保真度损失。量化可以使数据比特率下降。在图像压缩的

算法中，使用的是线性均匀量化器，DCT输出矩阵通过量化来减少系数的精度，从而提高压缩率。一般来说，如果矩阵中所有的值都表示出来的话，DCT 系数就处于全精度状态。随着删除矩阵中离 DC 系数较远的 AC 系数，DCT 系数的精度就降低了，图像压缩算法的目的是以不超过达到预期图像质量所必需的精度来表示 DCT 系数。

JPEG 基本算法中包含一套量化表，它们是基于人眼对各种空间频率的灵敏度，从广泛的实验中得到的。在量化用的代码中，较低空间频率的精度要高于较高频率的精度。一个典型的量化变换具有大部分零值部分，尤其是在较高频率时。设计者如果想在频谱的不同区域达到特定的精度要求，可以采用自己的量化表，调整量化表中数值即可改变图像的压缩比及图像压缩的峰值信噪比，表 14.2 所示的亮度量化表所对应的压缩比大于 15:1，图像压缩峰值信噪比大于 30 dB。对黑白图像，可以直接用亮度表进行量化；对彩色图像，需要把 RGB 各个分量提取出来分别量化。JPEG 推荐的量化表分为亮度量化表和色度量化表，分别如表 14.2 和表 14.3 所示。但考虑用 FPGA 实现图像压缩算法时，用乘法运算来实现除法操作比较方便。故该方案中对 JPEG 亮度量化表进行取倒数，然后四舍五入取 12 位宽的值经过转置后制作一个新的亮度量化表，如表 14.4 所示。

表 14.2 亮度量化表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

表 14.3 色度量化表

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

表 14.4 实际使用的 12 位宽的亮度量化表

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	256	341	293	293	288	171	84	57
8	372	341	315	241	186	117	64	45
16	410	293	256	186	111	74	53	43
24	256	216	171	141	73	64	47	42
32	171	158	102	80	71	51	40	37
40	102	71	72	47	38	39	34	41
48	80	68	59	51	40	36	34	40
56	67	74	73	66	53	45	41	41

JPEG 推荐的量化矩阵用 $Q(u, v)$ 表示，而该方案设计的量化矩阵用 $Q^*(u, v)$ 表示，经过二维 DCT 变换后的系数用 $F(u, v)$ 表示，则量化系数 $F^*(u, v) = F(u, v)/Q(u, v)$ 并取整。量化的实现流程如图 14.8 所示。在实际设计中，需要根据数据要求的精度来对量化表进行修订。

四位二进制编码。

表 14.5 亮度 DC 系数的 Huffman 编码表

亮度 DC 范围	码 长	码 字	亮度 DC 范围	码 长	码 字
0	2	00	6	4	1110
1	3	010	7	5	11110
2	3	011	8	6	111110
3	3	100	9	7	1111110
4	3	101	10	8	11111110
5	3	110	11	9	111111110

表 14.6 DC 系数对应的范围

DC 范围	DC 系数	DC 范围	DC 系数
0	0	9	−511, …, −256, 256, …, 511
1	−1, 1	10	−1023, …, −512, 512, …, 1023
2	−3, −2, 2, 3	11	−2047, …, −1024, 1024, …, 2047
3	−7, …, −4, 4, …, 7	12	−4095, …, −2048, 2048, …, 4095
4	−15, …, −8, 8, …, 15	13	−8191, …, −4096, 4096, …, 8191
5	−31, …, −16, 16, …, 31	14	−16383, …, −8192, 8192, …, 16383
6	−63, …, −32, 32, …, 63	15	−32767, …, −16384, 16384, …, 32767
7	−127, …, −64, 64, …, 127	16	32768
8	−255, …, −128, 128, …, 255		

对 AC 系数编码，表14.7给出了 AC 系数用比特表示的长度。将该长度与 0 行程组合构成符号 1，对符号 1 采用 Huffman 编码。符号 2 可用可变长度整数代码进行编码，若为负值，则采用二进制补码形式。表14.8中给出了 AC 系数的亮度 Huffman 编码表。

表 14.7 AC 系数对应的范围

AC 范围	AC 系数	AC 范围	AC 系数
1	−1, 1	6	−63, …, −32, 32, …, 63
2	−3, −2, 2, 3	7	−127, …, −64, 64, …, 127
3	−7, …, −4, 4, …, 7	8	−255, …, −128, 128, …, 255
4	−15, …, −8, 8, …, 15	9	−511, …, −256, 256, …, 511
5	−31, …, −16, 16, …, 31	10	−1023, …, −512, 512, …, 1023

表 14.8 AC 系数的亮度 Huffman 编码表

行程/尺寸	码 长	码 字
0/0	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000

(续表)

0/8	10	111110110
0/9	16	111111110000010
0/A	16	111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	1111110110
1/6	16	111111110000100
1/7	16	111111110000101
1/8	16	111111110000110
1/9	16	111111110000111
1/A	16	111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	111111110001001
2/6	16	111111110001010
2/7	16	111111110001011
2/8	16	111111110001100
2/9	16	111111110001101
2/A	16	111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	111111110001111
3/5	16	111111110010000
3/6	16	111111110010001
3/7	16	111111110010010
3/8	16	111111110010011
3/9	16	111111110010100
3/A	16	111111110010101
4/1	6	111011
4/2	10	1111111000
4/3	16	111111110010110
4/4	16	111111110010111
4/3	16	111111110010110
4/4	16	111111110010111
4/5	16	111111110011000
4/6	16	111111110011001
4/7	16	111111110011010
4/8	16	111111110011011
4/9	16	111111110011100

(续表)

行程/尺寸	码 长	码 字
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	1111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	11111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	11111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101
8/1	9	111111000
8/2	15	111111111000000
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011
8/9	16	1111111110111100

(续表)

行程/尺寸	码 长	码 字
8/A	16	111111110111101
9/1	9	111111001
9/2	16	111111110111110
9/3	16	111111110111111
9/4	16	111111111000000
9/5	16	111111111000001
9/6	16	111111111000010
9/7	16	111111111000011
9/8	16	111111111000100
9/9	16	111111111000101
9/A	16	111111111000110
A/1	9	11111010
A/2	16	111111111000111
A/3	16	111111111001000
A/4	16	111111111001001
A/5	16	111111111001010
A/6	16	111111111001011
A/7	16	111111111001100
A/8	16	111111111001101
A/9	16	111111111001110
A/A	16	111111111001111
B/1	10	111111001
B/2	16	111111111010000
B/3	16	111111111010001
B/4	16	111111111010010
B/5	16	111111111010011
B/6	16	111111111010100
B/8	16	111111111010110
B/9	16	111111111010111
B/7	16	111111111010101
B/A	16	111111111011000
C/1	10	111111010
C/2	16	111111111011001
C/3	16	111111111011010
C/4	16	111111111011011
C/5	16	111111111011100
C/6	16	111111111011101
C/7	16	111111111011110
C/8	16	111111111011111
C/9	16	111111111100000
C/A	16	111111111100001

(续表)

行程/尺寸	码 长	码 字
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	111111111110000
E/7	16	111111111110001
E/8	16	111111111110010
E/9	16	111111111110011
E/A	16	111111111110100
F/0 (ZRL)	11	11111111001
F/1	16	111111111110101
F/2	16	111111111110110
F/3	16	111111111110111
F/4	16	11111111111000
F/5	16	11111111111001
F/6	16	11111111111010
F/7	16	11111111111011
F/8	16	11111111111100
F/9	16	11111111111101
F/A	16	11111111111110

图14.12给出了 AC 系数编码流程。如图14.12所示，可以对 0 值 AC 系数的个数进行计数，直到遇到第一个非零的 AC 系数。但下面两种情况需要考虑：

(1) 当连续的零值 AC 系数的个数超过 16，用 ZRL 代码对这 16 个连续零值 AC 编码，同时游程计数复位为零。

(2) 当发现一个非零的 AC 系数，就按下面的方式构造一个复合 8 bit 数值。首先，找到表示 AC 系数所需要的位数，然后这个位数被编码成 BCD 值。这一 BCD 值就形成了 8 bit 复合值中的最低 4 bit，遇到非零系数之前的零值 AC 系数的个数生成了 8 bit 复合值中的高 4 bit。

重复上面的编码过程，直到全部 63 个 AC 系数都用这种方式处理完毕。如果最后一个 AC 系数是零，就用块结束(EOB)代码。

为下一步 Huffman 编码方便，把 DC 系数和 AC 系数分开处理，采用两个数据通道送入缓冲区。

连续重复的字符用两个符号表示，即符号 1 和符号 2。在本方案设计中，符号 1 为 8 位，符号 2 为 12 位，在实际工程项目中根据设计需要可以进行位宽调整。符号 1 的高 4 位表示两个非零值之间连续零的个数，低 4 位表示下一个非零值的比特数。符号 2 表示下一个非零值的实际值。其中行程编码的格式如表 14.9 所示。

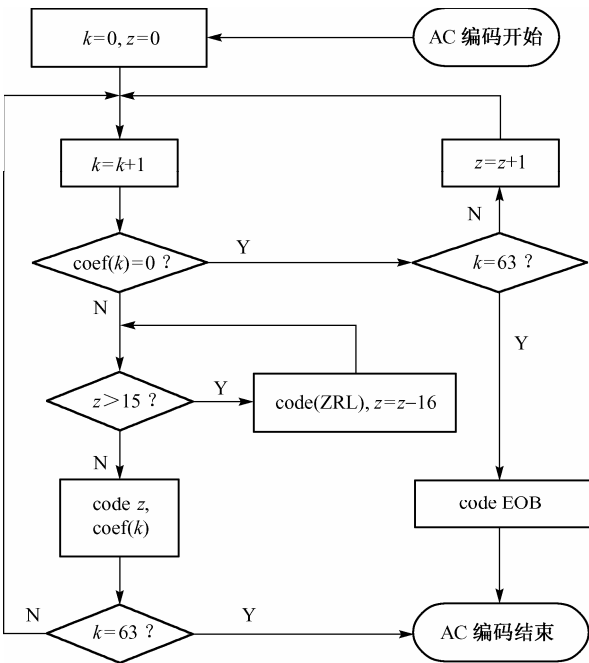


图 14.12 AC 系数编码流程

表 14.9 行程编码的格式

符号 1 及描述	7	6	5	4	3	2	1	0				
	零行程				范围							
符号 2 及描述	非零系数实际值											
	11	10	9	8	7	6	5	4	3	2	1	0

对采样的 8×8 像素图像块用已经设计成的游程编码模块进行仿真，其中 AC 系数图像块结束标志 EOB 在游程编码中用 0 表示，仿真数据按照十六进制显示，由于数据仿真结果跨越时间较长，采取截取两端部分结果，中间部分仿真结果省略。得到截取两端部分仿真结果如图14.13所示。

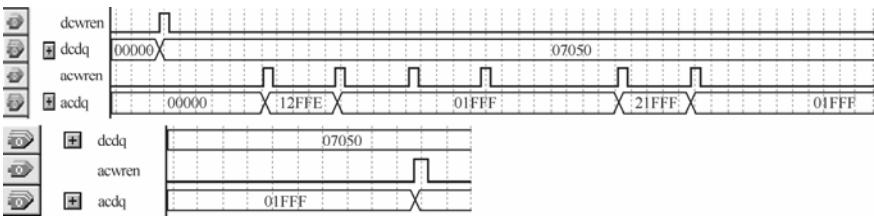


图 14.13 采样的 8×8 像素图像块的游程编码部分仿真结果

Huffman 编码及流程如图14.14所示。具体实现过程是：把前面对 DC 系数和 AC 系数的游程编码根据 Huffman 编码表进行 Huffman 编码，然后再对 Huffman 编码的码字进行合并为 16 位输出，在块结束标志 EOB(1010)到来时，输出过程结束。

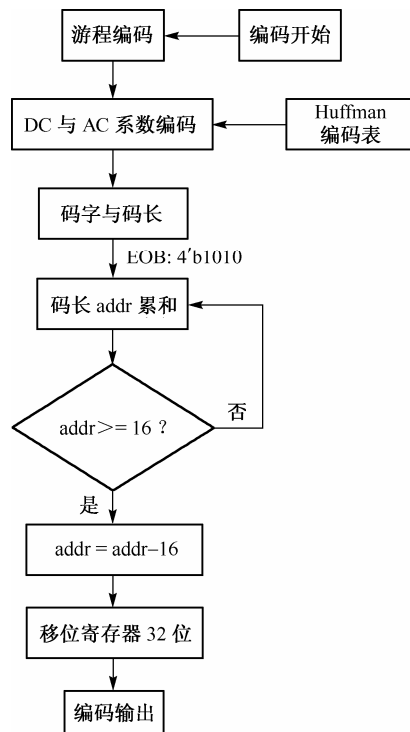


图 14.14 Huffman 编码及流程

14.3 本章小结

本章首先简单介绍了 JPEG 图像压缩算法的基本思想和主要步骤，然后详细说明了图像分割、离散余弦变换、量化与游程编码及熵编码的基本原理和它们在 FPGA 上的实现模型，并给出了 Quartus II 上的仿真结果。通过本章的学习，读者在掌握基本的图像处理知识的同时，更应学会如何将复杂的处理算法分解成各个相对简单的算法模块，学会如何利用基本的加、乘操作在 FPGA 上高效实现算法。

第 15 章 基于FPGA的神经网络对数-S形函数的设计与实现

15.1 引言

人工神经网络(Artificial Neural Network)作为人工智能的一个重要分支,被广泛地应用到金融、保险、交通、航天、国防等关系到国计民生的领域中。人工神经网络的研究途径有三种:利用传统的计算机进行模拟的数字模拟方法、采用光学器件的光学方法和采用大规模集成电路的电子学方法。采用电子学方法研究人工神经网络的基本构思是直接模仿人脑神经系统的结构,用非线性元件来模拟神经元,树突和轴突对应于连接单元的导线,突触则对应于连线的电阻,所得的即是人工神经网络芯片。其实人工神经网络理论模型本身就要求发展神经网络型计算机来实现,但迄今为止,由于条件的限制,这方面的工作还主要集中在传统计算机的软件模拟实现上。大多数学者认为,要使人工神经网络更快、更有效地进行大规模计算,关键在于其超大规模集成电路硬件的实现。正是基于上述考虑,神经网络的数字 VLSI 技术近年来发展很快,已成为人工神经网络研究的一个重要分支。

在神经网络的数字 VLSI 技术中,激活函数的硬件设计与实现是一个重点;而在激活函数的硬件实现中,对数-S 形函数(简称 S 形函数)的高效、精确的实现是一个难点。本章首先将介绍神经网络的一些基础知识及 S 形函数的基本特性,然后将阐述基本的坐标旋转数字计算机(Co-Ordinate Rotation Digital Computer, CORDIC)算法,最后在该算法的基础上提出了一种混合的 CORDIC 算法,并利用 FPGA 来设计与实现 S 形函数。读者在学习过程中应重点理解如何利用简单的加、乘、移位等运算去实现复杂的指数、双曲正弦、倒数等运算。

15.2 设计任务的提出与神经网络的基础知识

1. 设计任务

利用 FPGA 构造神经网络的 S 形函数模块。

2. 设计基本要求

- (1) S 形函数模块的计算误差平均值小于 0.1%, 最大值小于为 0.2%;
- (2) S 形函数模块的最高时钟频率大于 50 MHz, 流水线级数小于 10 级。

15.2.1 人工神经网络

神经系统的基本构造单元是神经细胞,也称为神经元。它和人体中其他细胞的关键区别在于具有产生、处理和传递信号的功能。每个神经元都由三部分组成:树突、细胞体和轴突。树突是树状的神经纤维接收网络,它将电信号传送到细胞体,细胞体对这些输入信号进行整合并进行阈值处理。轴突是单根长纤维,它把细胞体的输出信号导向其他神经元。一个神经细胞的轴突和另一个神经细胞树突的结合点成为突触。神经元的排列和突触的强度确立了神经网络的功能。简单神经网络及其简化结构如图15.1所示。

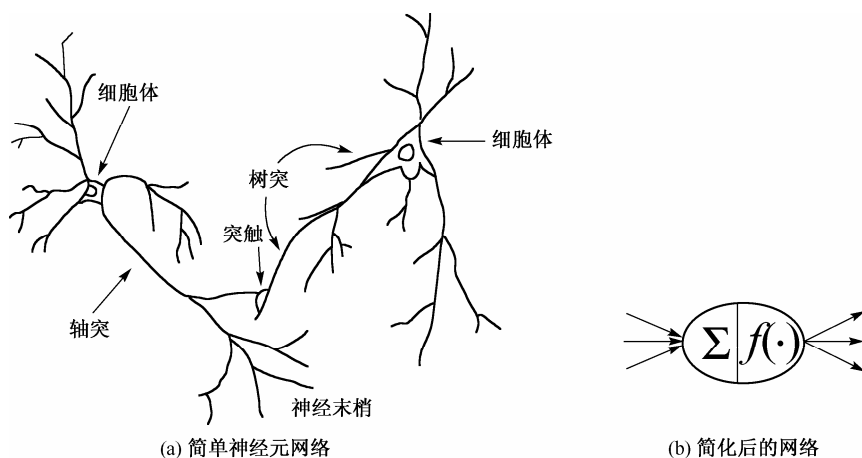


图 15.1 简单神经网络及其简化结构图

人们通过对人脑神经网络的初步认识,尝试构造出人工神经元以组成人工神经网络系统来对人的智能甚至是思维行为进行研究。经过几十年的努力和发展,已涌现出上百种人工神经网络模型。它们的网络结构、性能、算法及应用领域各异,但均是根据生物学事实衍生出来的。由于其基本处理单元是对生物神经元的近似仿真,因此被称为人工神经元。人工神经元的主要结构单元是信号的输入、综合处理和输出单元,其输出信号的强度大小反映了该单元对相邻单元影响的强弱。人工神经元之间通过互相连接形成网络,称为人工神经网络。神经元之间互相连接的方式称为连接模式,相互之间的连接度由连接权值体现。在人工神经网络中,改变信息处理过程及其能力,就是修改网络权值的过程。

15.2.2 神经元模型

神经元是神经网络操作的基本信息处理单元,图15.2给出了神经元的模型,它由三部分组成,是神经网络的设计基础。

(1) **连接链** 对应于生物神经元的突触,其连接强度由各连接上的权值表示,权值为正表示激活,为负表示抑制。

(2) **加法器** 用于求输入信号被神经元的相应突触的加权的和,这个操作构成一个线性组合器。

(3) **激活函数** 起非线性映射作用并将神经元输出幅度限制在一定范围内[一般限制在(0, 1)或(-1, +1)之间]。

此外,图15.2所示的神经元模型还包括一个外部偏置 b , 根据其为正或为负,相应地增加或降低激活函数的网络输入。

以上作用可分别用式(15.1)式(15.2)表达出来:

$$n = \sum_{j=1}^r p_j \cdot w_j \quad (15.1)$$

$$a = f(n + b) \quad (15.2)$$

式中, $p_1, p_2, \dots, p_r$ 为输入信号; $w_1, w_2, \dots, w_r$ 为神经元的权值; n 为线性组合结果; b 为偏置; $f(\cdot)$ 为激活函数; a 为神经元的输出。

激活函数是一个神经元及网络的核心。网络解决问题的能力与功效除了与网络的结构有关,在很大程度上取决于网络所采用的激活函数,本文将介绍最基本、最常用的三种。

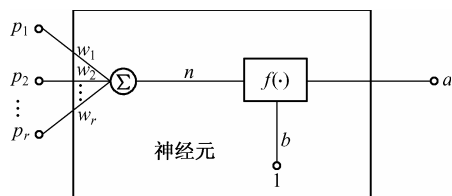


图 15.2 多输入单个神经元模型

(1) 阈值函数

阈值函数如图 15.3(a) 所示, 可表示为

$$f(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad (15.3)$$

(2) 分段线性函数

分段线性函数如图 15.3(b) 所示, 可表示为

$$f(v) = \begin{cases} 1, & v \geq 1 \\ \frac{1}{2}(1+v), & -1 < v < 1 \\ 0, & v \leq -1 \end{cases} \quad (15.4)$$

(3) Sigmoid 函数

此函数的形状为 S 形, 在构造的人工神经网络中是最常用的激活函数。它是严格的递增函数, 在线性与非线性行为之间显现出较好的平衡。它的一个例子是 **logistic 函数** (即 S 形函数), 如图 15.3(c) 所示, 可表示为

$$f(v) = \frac{1}{1 + \exp(-v)} \quad (15.5)$$

从图 15.3(c) 中可看出, 该激活函数的值域是 (0, 1)。有时候也期望激活函数的值域是 (-1, +1), 这时可采用 **tansig 函数**, 其如图 15.3(d) 所示, 可表示为式

$$f(v) = \frac{1 - \exp(-v)}{1 + \exp(-v)} \quad (15.6)$$

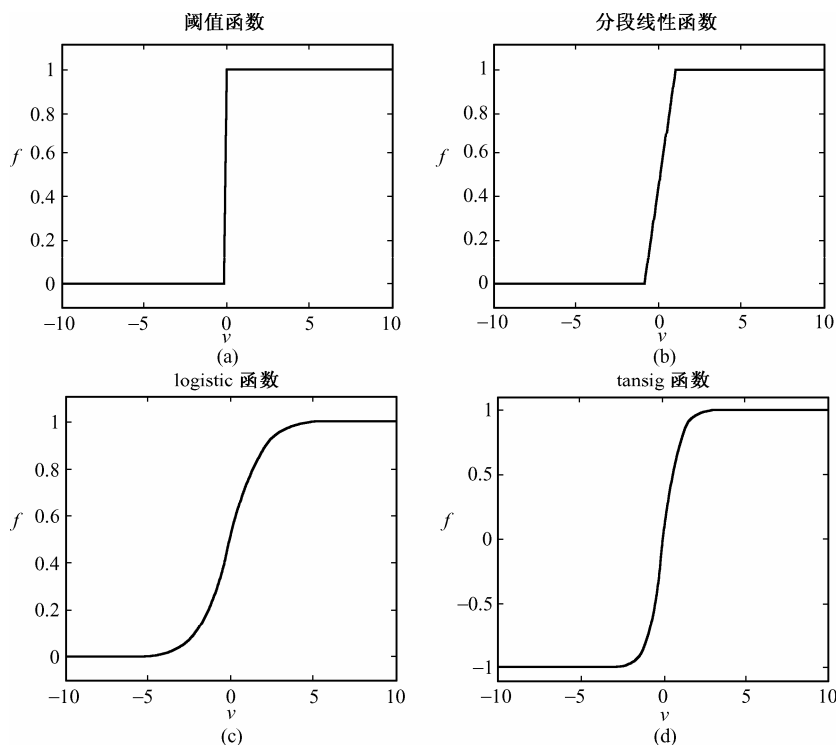


图 15.3 常用的激活函数

15.2.3 神经网络结构

一般来说，有多个输入的单个神经元并不能满足实际应用的要求，在实际应用中需要有多多个并行操作的神经元。神经网络中多个神经元的构造方式与训练网络的学习算法紧密相连，因此除单元特性外，网络的构造方式也是人工神经网络的一个重要特性。本节将介绍三种不同的基本网络结构。

1. 单层前向网络

在分层网络中，神经元以层的形式组织。在最简单的分层网络中，源节点构成输入层，直接投射到神经元的输出层。如图15.4所示，输入层和输出层各有三个节点，这样的网络称为单层网。注意，“单层”指的是神经元的输出层，源节点的输入层并不考虑在内，因为该层不具有执行计算的功能。

2. 多层前向网络

多层前向网络与单层前向网络的区别在于：多层前向网络含有一个或多个的隐含层(或称为隐藏层)，其中计算节点被相应地称为隐含(隐藏)神经元。通过加入一个或多个隐含层，使得网络能提取出更高序的统计，尤其当输入层规模庞大时，隐含神经元提取高序统计数据的能力便显得格外重要。

输入层的源节点提供激活模式的元素(输入向量)，组成第一层(第一隐含层)神经元的输入信号，第一层的输出信号作为第二层的输入信号，其余层类似。每一层的输入都是上一层的输出，最后输出层给出相对于源节点的激活模式的网络输出。如图 15.5 所示是一个 3-4-3 结构的前向神经网络，其由含有 3 个源节点的输入层、含有 4 个神经元的隐含层和含有 3 个神经元的输出层所组成。

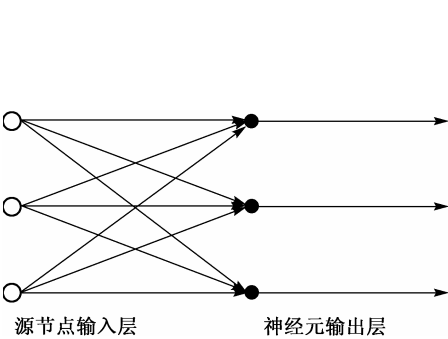


图 15.4 三输入-三输出的单层前向神经网络

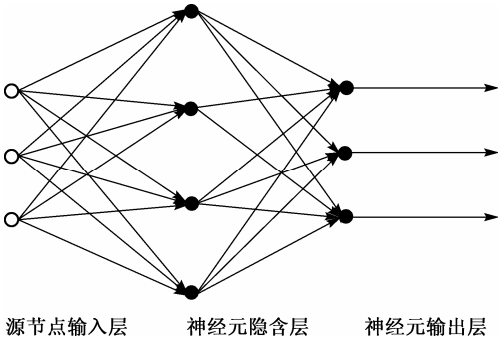


图 15.5 3-4-3 结构的前向神经网络

15.3 坐标旋转数字计算机(CORDIC)算法

15.3.1 CORDIC算法

坐标旋转数字计算机(CORDIC)算法是一种高效的迭代算法，它通过一系列简单的移位和加/减法迭代来实现矢量极坐标的旋转。由于该算法采用的运算符简单(移位和加/减)，易于硬件实现，因此被广泛地应用于乘法、除法、双曲线函数等超越代数函数的数字硬件设计和实现中。

CORDIC 算法由 Jack Volder 于 1959 年首先提出，用于计算在平面卡笛儿坐标系 (x, y) 和极坐标系 (R, θ) 之间进行自由坐标变换的乘法。CORDIC 算法的基本原理如下：设初始向量 V_0 [其极坐标为 (R_0, θ_0)]，平面卡笛儿坐标系 (x_0, y_0)] 旋转 α 后得到向量 V_1 [其极坐标为 (R_1, θ_1)]，平面卡笛儿坐标系 (x_1, y_1)]，即

$$x_0 = R_0 \cos \theta_0 \quad (15.7)$$

$$y_0 = R_0 \sin \theta_0 \quad (15.8)$$

$$R_1 = R_0 \quad (15.9)$$

$$x_1 = R_1 \cos \theta_1 = R_1 \cos(\theta_0 + \alpha) = R_1 (\cos \theta_0 \cos \alpha - \sin \theta_0 \sin \alpha) \quad (15.10)$$

$$y_1 = R_1 \sin \theta_1 = R_1 \sin(\theta_0 + \alpha) = R_1 (\sin \theta_0 \cos \alpha + \cos \theta_0 \sin \alpha) \quad (15.11)$$

将式 (15.7)、式 (15.8) 和式 (15.9) 分别代入式 (15.10) 和式 (15.11)，得到

$$x_1 = x_0 \cos \alpha - y_0 \sin \alpha = \cos \alpha (x_0 - y_0 \tan \alpha) \quad (15.12)$$

$$y_1 = x_0 \sin \alpha + y_0 \cos \alpha = \cos \alpha (x_0 \tan \alpha + y_0) \quad (15.13)$$

式 (15.12) 和式 (15.13) 中乘法运算 $x_0 \tan \alpha$ 和 $y_0 \tan \alpha$ 就可以用移位操作来代替，且任意一个给定的角度 α 都可以分解为一系列更小的微旋转。每次迭代相当于将向量旋转了一个微角度：约定 δ_k 代表向量的旋转方向，+1 代表逆时针方向，-1 代表顺时针方向；为了在硬件上实现方便，约定第 k 次旋转的角度 β 的正切值为 2 的倍数，即

$$\beta_k = \arctan 2^{-k} \quad (15.14)$$

$$\cos \beta_k = \sqrt{\frac{1}{1 + 2^{-2k}}} \quad (15.15)$$

若令 $N_k = \cos \beta_k$ ，其中 N_k 称为校正因子，则运算迭代式可表示为

$$x_{k+1} = \cos \beta_k (x_k - y_k \tan \beta_k) = N_k (x_k - \delta_k \cdot y_k \cdot 2^{-k}) \quad (15.16)$$

$$y_{k+1} = \cos \beta_k (x_k \tan \beta_k + y_k) = N_k (\delta_k \cdot x_k \cdot 2^{-k} + y_k) \quad (15.17)$$

为了避免在每次迭代运算中都进行校正因子的运算，可以在迭代结束后一起计算校正因子所带来的缩放效应，因此式 (15.16) 和式 (15.17) 可以简化为

$$x_{k+1} = x_k - \delta_k \cdot y_k \cdot 2^{-k} \quad (15.18)$$

$$y_{k+1} = \delta_k \cdot x_k \cdot 2^{-k} + y_k \quad (15.19)$$

第 k 次角的迭代如式 (15.20) 所示：

$$z_{k+1} = z_k + \delta_k \arctan(2^{-k}) \quad (15.20)$$

如果 α 是经过 K 次 CORDIC 迭代后总的旋转角度，则有

$$\alpha = \theta_K - \theta_0 = \sum_{k=0}^{K-1} \delta_k \cdot \beta_k \quad (15.21)$$

若 K 次 CORDIC 迭代后 $\theta_K \rightarrow 0$ ，则式 (15.18)、式 (15.19) 和式 (15.20) 可分别表示为

$$x_K = x_0 \cos z_0 - y_0 \sin z_0 \quad (15.22)$$

$$y_K = y_0 \cos z_0 + x_0 \sin z_0 \quad (15.23)$$

$$z_K \rightarrow 0 \quad (15.24)$$

则 K 次 CORDIC 迭代后总的校正因子 N 为

$$N = \prod_{k=1}^K N_k = \prod_{k=1}^K \sqrt{\frac{1}{1+2^{-2k}}}$$

(15.25)

若 $x_0 = N, y_0 = 0$ ，由于每次迭代时没有考虑校正因子的影响，则可得到

$$x_K = \cos z_0$$

(15.26)

$$y_K = \sin z_0$$

(15.27)

$$z_K \rightarrow 0$$

(15.28)

通过上面分析可看出，将所需要的角度值作为 z_0 输入，由式(15.26)、式(15.27)和式(15.28)所得到的迭代结果 x_K 和 y_K 就是所需要的三角函数值。迭代的总次数 K 由所期望的精度所决定，CORDIC 是位递归算法，即每完成一次迭代，结果的精度大约提高一位。当然，在定点运算中由于有限精度造成的截断也会带来近似误差。

表15.1 给出了 $0 \leq k \leq 15$ 所对应的迭代序列，从表中可看出其所覆盖的角度只有 $-90^\circ \sim +90^\circ$ ，而一般情况下要求覆盖的角度为 $-180^\circ \sim +180^\circ$ ，根据三角函数的变换公式，可采用如下方法处理 $-90^\circ \sim +90^\circ$ 范围外的角度。

- (1) 旋转 180° : $(x, y) \rightarrow (-x, -y), \theta = \theta - 180^\circ$
- (2) 旋转 90° : $(x, y) \rightarrow (y, -x), \theta = \theta - 90^\circ$

表 15.1 迭代序列所对应的角度

k	2^{-k}	β_k (度数)	k	2^{-k}	β_k (度数)
0	1	45	8	0.0039062	0.223811
1	0.5	26.565051	9	0.0019531	0.111906
2	0.25	14.036243	10	0.0009766	0.055953
3	0.125	7.125016	11	0.0004883	0.027976
4	0.0625	3.576334	12	0.0002441	0.013988
5	0.03125	1.789911	13	0.0001221	0.006994
6	0.015625	0.895174	14	0.0000610	0.003497
7	0.0078125	0.447614	15	0.0000305	0.001749

J. Walter 于 1971 年推广了 CORDIC 算法，将圆周 ($m = 1$)、线性 ($m = 0$) 和双曲线 ($m = -1$) 变换都包括进来。扩展的 CORDIC 算法方程分别如式(15.29)和式(15.30)所示：

$$\begin{bmatrix} X_{k+1} \\ Y_{k+1} \end{bmatrix} = N_k \begin{bmatrix} 1 & -m\delta_k 2^{-k} \\ \delta_k 2^{-k} & 1 \end{bmatrix} \begin{bmatrix} X_k \\ Y_k \end{bmatrix}$$

(15.29)

$$Z_{k+1} = Z_k + \delta_k \theta_k$$

(15.30)

如表15.2所示，参数 m 可取值-1、0 和 1，分别代表双曲线、线性和圆周三种极坐标变换模式，与其对应的迭代角度 θ_k 分别为 $\arctan h 2^{-k}, 2^{-k}$ 和 $\arctan 2^{-k}$ 。每种变换模式的两个旋转方向都是确定的。对向量化 ($Y_K \rightarrow 0$) 而言，具有原点的向量 (X_0, Y_0) 根据 $\delta_k = -\text{sign}(Z_k)$ 的值进行旋转，直到 Y_K 迭代收敛到 0；对旋转 ($Z_K \rightarrow 0$) 而言，具有原点的向量 (X_0, Y_0) 根据 $\delta_k = -\text{sign}(X_k \cdot Y_k)$ 的值进行旋转，直到 Z_K 迭代收敛到 0。 N_k 是模校正因子，用于校正第 k 次迭代所引起的模值改变，一般在每次迭代过程

中先不考虑 N_k 的模校正作用, 在迭代结束后用整个迭代过程的总模校正因子 $N = \prod_{k=1}^K N_k$ 对输出结果进行校正。

表 15.2 CORDIC 算法的操作模式

m	$Z_K \rightarrow 0$ (旋转)	$Y_K \rightarrow 0$ (向量化)	θ_k
1 (圆周)	$X_K = N(X_0 \cos Z_0 - Y_0 \sin Z_0)$ $Y_K = N(Y_0 \cos Z_0 + X_0 \sin Z_0)$	$X_K = N\sqrt{X_0^2 + Y_0^2}$ $Z_K = Z_0 + \arctan(Y_0/X_0)$	$\arctan 2^{-k}$
0 (线性)	$X_K = X_0$ $Y_K = Y_0 + X_0 Z_0$	$X_K = X_0$ $Z_K = Z_0 + Y_0/X_0$	2^{-k}
-1 (双曲线)	$X_K = N(X_0 \cosh Z_0 + Y_0 \sinh Z_0)$ $Y_K = N(Y_0 \cosh Z_0 + X_0 \sinh Z_0)$	$X_K = N\sqrt{X_0^2 - Y_0^2}$ $Z_K = Z_0 + \arctan h(Y_0/X_0)$	$\arctan h 2^{-k}$

CORDIC 算法的本质是利用一组常数的角度基底 $\{\theta_k, k \in [0, K]\}$ 去逼近任意一个旋转角度, 为了确 CORDIC 算法的收敛性, 这组常数的角度基底必须满足式 (15.31) 所示的不等式关系:

$$\theta_0 - \sum_{k=1}^{K-1} \theta_k \leq \theta_{K-1} \quad (15.31)$$

对于圆周模式或者线性模式而言, 由于 $\arctan(2^{-(k+1)}) \geq 0.5 \arctan(2^{-k})$ 或者 $2^{-(k+1)} \geq 0.5 \cdot 2^{-k}$, 满足式 (15.31) 所示的不等式关系, 因此在这两种模式下, CORDIC 算法的收敛性是满足的; 但对于双曲线模式而言, 由于并不是所有的 $\arctan h(2^{-(k+1)}) \geq 0.5 \arctan h(2^{-k})$, 因此不满足式 (15.31) 所示的不等式关系, 从而使得该模式下的收敛性无法得以保证。一个相对简单的解决方法是: 当迭代次数 $k = 4, 13, K, j, 3j+1, K$ 时, 相应的迭代进行两次, 即在双曲线模式下, 由式 (15.32) 给出了此时的旋转角度基底。

$$\{\theta_0, \theta_1, L, \theta_{K-1}\} = \{\arctan h 2^0, \arctan h 2^{-1}, L, \arctan h 2^{-4}, \arctan h 2^{-4}, L, \arctan h 2^{-K+1}\} \quad (15.32)$$

当采用不同的操作模式并赋予不同的初值时, CORDIC 算法可用于计算各种不同的超越代数函数。例如:

(1) 当 $m = -1, X_0 = 1/N, Y_0 = 0, Z_0 = \theta$ 时, 可计算出对应的双曲正弦 $X_K = \cosh(Z_0)$ 、双曲余弦 $Y_K = \sinh(Z_0)$ 及指数 $e^{-Z_0} = X_K - Y_K$;

(2) 当 $m = 0, X_0 = X, Y_0 = 1, Z_0 = 0$ 时, 可计算出对应的倒数函数 $Z_K = 1/X$ 。这两种函数都将用于神经网络 S 形函数的设计中。

15.3.2 混合CORDIC算法

在利用 CORDIC 算法来实现基本超越代数函数计算时, 所需的计算精度越高, 相应的迭代次数就越多。但迭代次数过多, 将导致计算效率下降。虽然流水线技术在大规模集成电路设计中的引入适当地提高了算法的计算效率, 但是当集成电路对流水线级数提出限制时, CORDIC 算法的计算效率成为算法实现的主要问题, 需找出可行的解决方法。

从原则上来说, CORDIC 算法通过多次迭代可得到对应任意分辨率的输入数据的结果, 而传统的查找表 (LUT) 技术由于受到查找表容量随着分辨率增长而迅速增长的限制, 往往适用于低分辨率的输入数据范围。将传统的 LUT 和基本的 CORDIC 算法相结合的混合 CORDIC (Hybrid CORDIC) 算法解决了计算效率和计算精度的矛盾。利用该方法, 根据需要适当分配 LUT 和基本 CORDIC 算法迭代运算所占的比例, 从而能够高效率、高精度地计算对应的输出值。

15.4 基于FPGA的对数-S形函数模块的硬件设计与实现

1. S形函数模块的系统总体设计

分析 S 形函数输入与输出的关系,可得到当输入变量 $Z > 8$ 或 $Z < -8$ 时, S 形函数输出值趋近于 1 或 0。因此在本设计中定义输入信号范围为 $-8 < Z < 8$, 对应的输出信号的范围为 $0 < S < 1$ 。

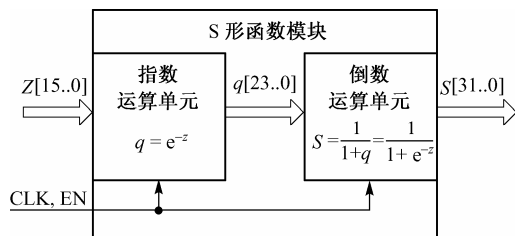


图 15.6 S 形函数的系统设计框图

如图 15.6 所示,采用自顶向下的设计方法, S 形函数的寄存器传输级(RTL)模块被划分为指数运算单元子模块和倒数运算单元子模块。S 形函数的功能模块的输入和输出数据分别采用了两种数据格式:输入数据用 16 bit 字长来表示,其中 1 bit 用于符号位,3 bit 用于表示输入数据的整数部分,12 bit 用于表示输入数据的小数部分;输出数据用 32 bit 字长来表示,其中 1 bit 用于符号位,1 bit 用于表示

输出数据的整数部分,30 bit 用于表示输出数据的小数部分。输入和输出数据采用不同的数据格式的主要原因在于 S 形函数的输出范围 $0 < S < 1$, 如果其输入和输出采用相同的数据格式,那么输出信号的大部分数据位将会一直为 0, 所以输入和输出应采用不同的数据格式。

设计的 S 形函数模块在流水线优化前完成一次正常计算需要经过 24 级流水线。指数运算子模块包含 16 级流水线,其中 1 级流水线用于输入数据的锁存,1 级流水线用于查找表操作,13 级流水线用于双曲线模式下 CORDIC 算法的旋转迭代,1 级流水线用于总模校正因子的校正及指数 $e^{-Z_0} = X_K - Y_K$ 的计算;倒数运算子模块包含 8 级流水线,其中 1 级流水线用于数据的归一化,1 级流水线用于查找表操作,1 级流水线用于乘法功能的实现,4 级流水线用于线性模式下的 CORDIC 算法的向量化迭代,1 级流水线用于数据的反归一化。

2. 指数运算单元子模块

考虑到输入信号 Z 的范围,指数运算单元子模块的输出信号 q 的数据格式用 24 bit 字长来表示,其中 1 bit 用于符号位,11 bit 用于表示输出信号的整数部分,12 bit 用于表示输出信号的小数部分。

依据 Hybrid CORDIC 算法的基本原理,在指数运算单元子模块中构建了一个 $2 \cdot 24 \cdot 2^4$ bit 容量的 ROM,该 ROM 里面存储着双曲正弦和双曲余弦的值。输入信号 Z 的 1 bit 符号位和 3 bit 的整数位作为 ROM 的地址输入,通过读取 ROM 中 24 bit 的存储值,将直接获得与输入数据整数部分对应的双曲正弦和双曲余弦,从而节省了利用基本 CORDIC 算法所需要的最大迭代次数为 7 次的迭代时间。

输入数据 12 bit 的小数部分决定了接下来的 13 次迭代(其中 $k=4$ 的迭代进行了两次),如图 15.7 所示,每次迭代运算仅仅用到移位和加、减法,13 级基于流水线的迭代操作后,通过基于移位的总模校正运算和减法运算即可获取对应的指数值,从而实现了指数运算单元子模块的功能。

3. 倒数运算单元子模块

倒数运算单元子模块的设计同样也是基于 Hybrid CORDIC 算法的,在该子模块中包含有一个 ROM 单元,通过进行查找表操作来读取里面存储的对应于小数部分高 10 bit 的倒数数值,接下来再进行 CORDIC 迭代操作,从而计算对应整个输入的准确的倒数数值。

该子模块由加法子单元、数据归一化和反归一化子单元、查找表子单元三部分组成。其中,加法子单元用于计算输入信号 q 与 1 的和;数据归一化和反归一化子单元在倒数计算前将利用移位运算把

数据转换到 $1 \leq q+1 < 2$ ，在倒数计算后同样利用移位运算将实际的计算结果还原；查找表子单元实际上是一个容量为 $12 \cdot 2^{10}$ bit 的 ROM，归一化后的数据的高 10 bit 小数部分的值将作为该 ROM 的地址输入，读取里面存储的对应的倒数值。

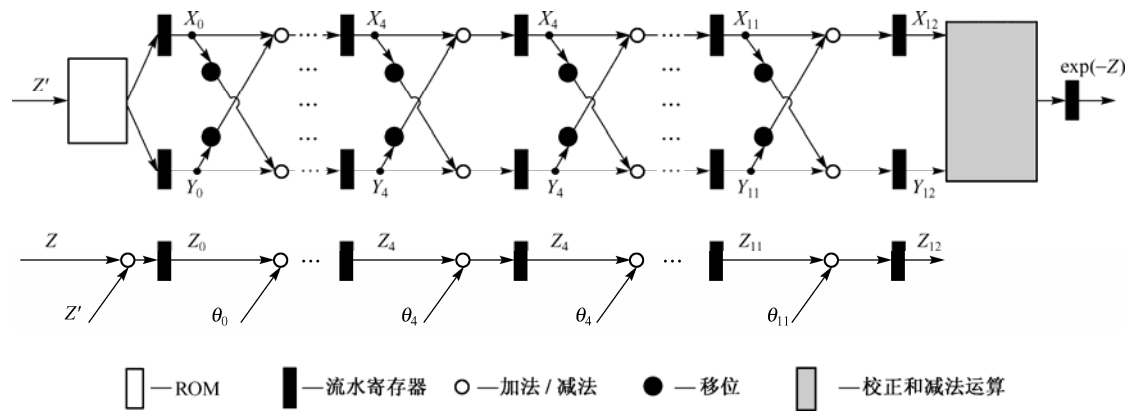


图 15.7 基于 Hybrid CORDIC 算法的指数运算单元子模块的框图

CORDIC 算法在线性向量化的模式下，通过迭代操作，最后的结果 Z 将逼近输入 X 的倒数值。第 k 次迭代时，根据旋转方向， Z 将加上或者减去 2^{-k} 。由于 Z 的初始值为 0，在经过 k 次迭代后将得到 k bit 小数宽度的倒数值。本设计中经过查找表操作得到的值对应于 Z 在 k 次迭代后的值（可在设计中取 $k=10$ ），因此式 (15.29) 可转换为

$$Y_{k+1} = Y_0 - l \cdot X \tag{15.33}$$

从式 (15.33) 可看出，原有的 k 次迭代操作被一次 LUT 操作所代替，因此大大地提高了计算效率。由于目前流行的 FPGA 器件中一般都含有一定数量的专用乘法器单元，因此 16×16 bit 的专用乘法器和 32 位的加法/减法器将用于求解 Y_{k+1} 的运算。将 Y_{k+1} 和 1 的值作为初始值赋值给接下来的线性向量化的 CORDIC 操作，然后按照图 15.7 所示的迭代操作进行 4 次迭代，最终得到输入 X 的精确的倒数值。

4. 综合与仿真

利用 Verilog HDL 硬件描述语言实现的 S 形函数功能模块采用 Altera 公司 Cyclone II 系列的 FPGA EP2C35F672C8 作为目标芯片，使用 Altera 公司的 FPGA 设计工具 Quartus II 7.2 进行综合、时序分析、仿真。

表 15.3 是 S 形函数功能模块综合后的时序分析结果，从表中可得到所设计的 S 形函数模块的最高时钟频率约为 72.8 MHz，其最大延时来自于乘法器运算子单元，耗时约为 13.7 ns。图 15.8 是该模块的后仿真波形示意图，为了便于观察，设定 S 形函数模块工作在 50 MHz 时钟频率下，从零点为起点在模块的输入端 Z 依次给出任意 10 个输入值 ($-8 < Z < 8$)，经过 24 级流水线操作，在 S 形函数模块的输出端 sigmoid_out 得到对应的 10 个输出值。从图中可看出整个模块的流水线操作正常，为了比较功能模块输出值与实际值的误差，可用计算器去直接计算这 10 个对应的输出值（取输出值的 4 位有效数字），结果如表 15.4 所示。

表 15.3 S 形函数功能模块综合后的时序分析结果

S 形函数功能模块类型	最大工作频率	最大延时	流水线级数
S 形函数模块	72.8 MHz	13.7 ns	24

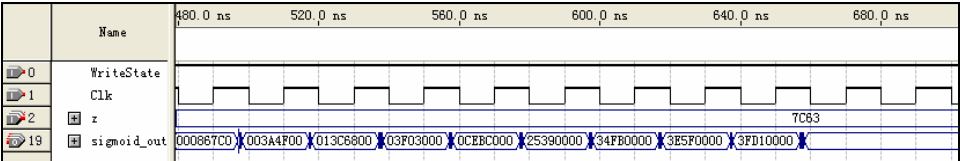


图 15.8 S 形函数功能模块的后仿真波形示意图

表 15.4 Quartus II 7.2 后仿真结果和计算器直接计算结果比较

输入值 (Hex/Dec)	Quartus II 输出值 (Hex/Dec)	直接计算输出值 (Dec)
86CB/-5.7554	0008670/0.0005130	0.0005127
A5D3/-5.6355	003A4F00/0.003559	0.03556
C126/-3.9282	013C6800/0.01931	0.01930
D465/-2.7253	03F03000/0.06153	0.06150
EA00/-1.3750	0CEBC000/0.2019	0.2018
0539/0.3264	25390000/0.5816	0.5809
1918/1.5684	34FB0000/0.8278	0.8276
3A57/3.6462	3E5F0000/0.9745	0.9746
5AFF/5.6873	3FD10000/0.9971	0.9966
7C63/7.7742	40000000/1.0000	0.9994

许多学者在 CORDIC 算法的实现上开展了大量的研究工作。近年来，一种被称为分段线性逼近的思想应用于 S 形激活函数的硬件实现中，Myers, Alippi, Amin 和 Basterretxea 分别介绍了 4 种采用该思想在段内利用不同逼近法则来实现 S 形函数的方法。为了更加直观地比较本设计模块与文献 [61～64] 中实现的 S 形函数模块的计算精度，可利用 MATLAB 软件编写程序(考虑了字节的截断误差)去获取本设计模块计算误差的平均值和最大值，其结果如图 15.9 和表 15.5 所示，由于 Hybrid CORDIC 的计算结果逼近计算器直接计算结果，因此图 15.9(a) 中 Hybrid CORDIC 的计算结果曲线与计算器直接计算结果曲线几乎重合，图 15.9(b) 给出了 Hybrid CORDIC 的计算结果与计算器直接计算结果之间的相对误差曲线。其中图 15.9(b) 是图 15.9(a) 对应的相对误差。从表 15.5 可看出，本设计模块的计算误差平均值为 0.05%，最大值为 0.19%，其计算精度远远高于文献 [61～64] 中设计的 S 形函数的计算精度，满足系统设计指标要求。

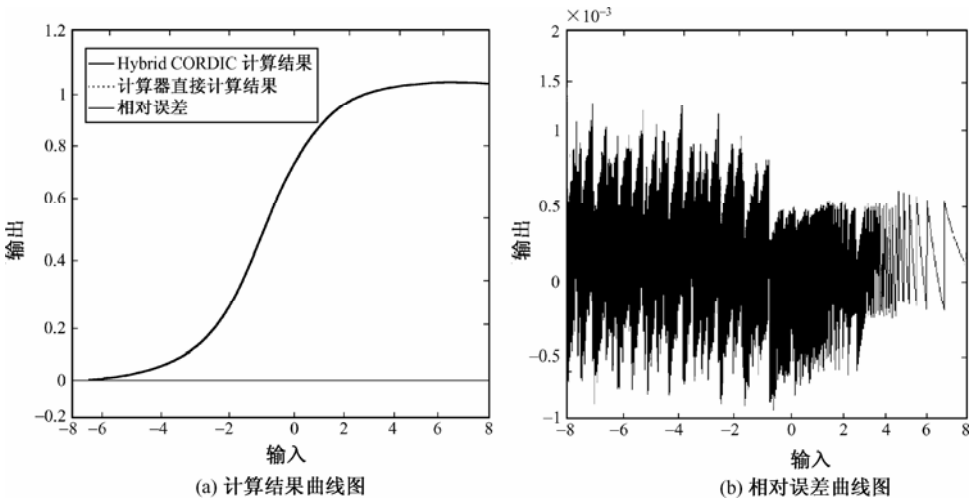


图 15.9 Hybrid CORDIC 算法仿真结果及相对误差

表 15.5 S 形函数功能模块的平均误差和最大误差

采用的方法	平均误差(相对)	最大误差(相对)
文献 [61]	0.0247	0.0490
文献 [62]	0.0087	0.0189
文献 [63]	0.0059	0.0189
文献 [64]	0.0077	0.0222
Hybrid CORDIC	0.0005	0.0019

5. 流水线优化

基于 Hybrid CORDIC 算法所设计的 S 形函数的计算精度的提高是以增加迭代次数为代价的,在优化前的设计中,虽然采用 LUT 技术节省了一定的迭代次数,但 S 形函数的功能模块进行一次计算依然需要经过 24 级流水线。为了提高计算效率,需要在不降低电路最高工作频率的前提下,对流水线的级数进行优化。通过 Quartus II 的时序分析可知,优化前的 S 形函数的功能模块的最大时延来自于乘法器运算单元子模块。在流水线的优化过程中,为保证流水线最大时延不变或者稍稍增加的情况下,将多级低延时的流水线合并为一级流水线。在本优化中,将指数运算单元子模块中的 13 级双曲线模式下 CORDIC 算法的旋转迭代流水线操作和 1 级总模校正因子校正及指数 $e^{-Z_0} = X_K - Y_K$ 计算的流水线操作合并为 4 级流水线操作;将倒数运算子模块中 1 级用于数据归一化的流水线操作和 1 级用于查找表的流水线操作合并为 1 级流水线操作,将 4 级用于线性模式下的 CORDIC 算法的向量化迭代的流水线操作合并为 1 级流水线操作。基于上述分析,优化后 S 形函数的功能模块的流水线级数从 24 级减少到了 10 级,且最高工作频率依然大于 50 MHz,从而进一步满足了系统设计指标要求。

15.5 本章小结

本章在分析和讨论传统 CORDIC 算法优、缺点的基础上,给出了利用混合 CORDIC 算法来设计并实现 S 形激活函数的方法,该方法结合了查找表和坐标旋转数字计算机迭代算法的优点,解决了计算效率和计算精度的矛盾。利用该方法,根据任务需要适当分配 LUT 和基本 CORDIC 算法迭代运算所占的比例,从而能够高效率、高精度地计算对应的输出值。在该思想的基础上,采用超高速集成电路硬件描述语言和流水线技术构造了 S 形函数的寄存器传输级模块,并在现场可编程门阵列上给予硬件实现。读者通过本章的学习应理解如何利用简单的加、乘、移位等运算去实现复杂的指数、双曲正弦、倒数等运算;与之同时,读者在功能模块设计中应充分运用流水线的设计思想来实现模块功能,并能够进行流水线优化设计,从而提高模块的工作效率。

第 16 章 基于TMS320C55XX系列DSP的 系统硬件和软件设计

16.1 引言

数字信号处理是通信、语音、图像等工程技术领域的一门关键基础学科，它以数字的形式对信号进行分析、采集、合成、变换、滤波、估计、识别等加工处理以提取有用的信息。与模拟信号处理相比，数字信号处理具有全软件方式实现、模块间不存在阻抗匹配问题、抗干扰能力强和易于集成等优点。在近三十年里，数字信号处理技术在理论和工程应用中都取得了长足的进步。

数字信号处理是以众多学科为理论基础的，所涉及的范围极其广泛，它包含算法的研究和算法实现两部分。常见的数字信号处理算法，如语音与图像的压缩编码、识别和鉴别，信源的加密与解密，通信信号的调制与解调，通信信道的估计与均衡，神经网络等智能算法，这些算法的研究目标是如何以最小的运算速度和存储器空间来完成给定的信号处理任务。数字信号处理的实现是指用硬件、软件或软硬件结合的方法来实现各种算法，各种数字信号处理方法可以在通用处理器上实现、在各种专用数字信号处理 ASIC 上实现、在各种通用的可编程数字信号处理器上实现或在包含有通用数字信号处理器的 ASIC 上实现，应用最广泛的还是在各种通用数字信号处理器上的实现方式。

数字信号处理器是一种专门为快速实时实现各种数字信号处理算法而设计的、具有特殊结构的微处理器。自从 20 世纪 80 年代初数字信号处理器芯片出现以来，DSP 技术已取得了突飞猛进的发展，DSP 芯片的性能不断提高，价格却不断下降。DSP 的应用领域也在不断的扩展，从军用、航天等高端产品到各种民用产品，今天人们的生活中已经到处可以见到数字信号处理器应用的例子，在我国 DSP 技术也正以极快的速度被应用到国民经济的各个领域。

目前，我国的 DSP 产品主要来自海外。1983 年 TI 公司的第一代产品 TMS32010 最先进入中国市场，以后 TI 公司通过提供 DSP 培训课程，使该公司 DSP 产品的市场份额不断扩大。现在 TI 公司的 DSP 产品约占据国内 DSP 市场份额的 80% 以上而遥遥领先，其余的市场份额由 Lucent, AD, Motorola, ZSP 和 NEC 等公司占有。本章通过一个水声通信系统的硬件和软件开发例子，详细介绍了 DSP 的发展现状、TI 公司 DSP 芯片的开发技术及基于 TMS320C55XX 系列 DSP 系统的硬件和软件的开发过程。

16.2 TMS320C55XX系列DSP简介

16.2.1 DSP芯片的特点

DSP 芯片的结构特点在很大程度上体现了 DSP 算法的需求。下面介绍 DSP 芯片在结构上的主要特点。

1. 算术单元

(1) 硬件乘法累加器

数字信号处理算法中的数据操作具有高度重复的特点，特别是乘、加操作。专门的硬件乘法累加

器是 DSP 芯片区别于通用微处理器 (General-Purpose Processors, GPP) 的一个重要标志。在 GPP 内通过微程序实现的乘、加操作往往需要 100 多个时钟周期, 非常费时; 而专门的硬件乘法累加器单周期实现乘、加操作, 并用累加器寄存器来处理多个乘积的累加。

(2) 多功能单元

在 CPU 内部设置多个并行操作功能单元 (ALU、乘法器和地址产生器等), 使 DSP 芯片在相同时间内能够完成更多的操作, 提高了程序执行速度。

2. 总线结构

早期的微处理器内部大多采用冯·诺依曼 (Von-Neumann) 结构: 统一的程序和数据空间, 共享的程序和数据总线。取指和取操作数分时进行, 造成传输通道上的瓶颈。而 DSP 内部采用的是程序空间和数据空间分开的哈佛 (Harvard) 结构, 允许同时取指令 (来自程序存储器) 和取操作数 (来自数据存储器)。很多 DSP 芯片内部有两套或两套以上的内部数据总线, 这种总线结构称为修正的哈佛结构。多总线结构可以保证一个机器周期内多次访问程序空间和数据空间。

TI 公司的 C6000 系列 DSP 芯片采用了甚长指令字 (VILW) 结构, 片内提供了 8 个独立的运算单元、256 位的程序总线、两套 32 位数据总线和一套 32 位的 DMA 专用总线。灵活的总线结构大大缓解了数据瓶颈对系统性能的限制。

3. 片内存储器

DSP 面向的是数据密集型应用, 存储器的访问速度对处理器的性能至关重要。现代通用微处理器内部一般集成有高速缓存器 (cache), 但是片内不设存储数据的 RAM 和存储程序的 ROM。这是因为通用微处理器的程序一般很大, 片内存储器不会给处理器性能带来更大的改善。而 DSP 算法的特点是需要大量简单的计算却程序短小, 程序存放于片内能有效减少指令传输时间, 缓解芯片外总线接口的压力。另外, 在内部集成存储数据的 RAM, 能够匹配访问速度, 防止总线竞争, 缓解了 DSP 芯片的数据瓶颈。

4. 流水线结构

DSP 芯片内部的流水线结构将指令分解为取指令、译码、取操作数和执行等几个阶段, 每一个阶段称为一级流水。在程序运行过程中, 不同指令的不同阶段在时间上是重叠的, 流水线结构提高了指令执行的整体速度, 有助于保证数字信号处理的实时性。图 16.1 给出四级流水线操作示意图, 在执行本条指令的同时, 还依次完成后面 3 条指令的取操作数、译码和取指, 将指令周期降低到最小值。

理想情况下, 一条 k 段流水线能在 $k+(n-1)$ 个周期内处理 n 条指令。其中前 k 个周期用于完成第一条指令的执行, 其余 $n-1$ 条指令的执行需要 $n-1$ 个周期。而在非流水线处理器上执行 n 条指令, 需要 nk 个周期。当指令条数 n 较大时, 可认为每个周期内执行的最大指令数为 k , 即流水线理想情况下效率为 1, 但由于流水线的填充和排空, 这种理想情况很难达到。

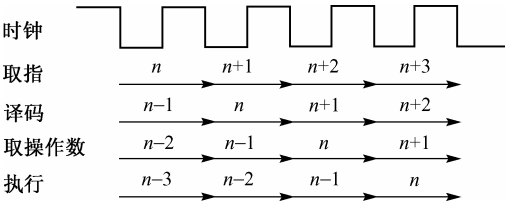


图 16.1 四级流水线操作示意图

16.2.2 TI公司DSP

世界上第一个单片 DSP 芯片是 1978 年 AMI 公司宣布的 S2811, 1979 年美国 Intel 公司发布的商用可编程器件 2920 是 DSP 芯片的一个重要里程碑, 这两种芯片内部都不是现代 DSP 芯片所必须的单

周期芯片。1980 年日本 NEC 公司推出的 μ PD7720 是第一个具有乘法器的商用 DSP 芯片。第一个采用 CMOS 工艺生产浮点 DSP 芯片的是日本的 Hitachi 公司，它于 1982 年推出了浮点 DSP 芯片。1983 年日本的 Fujitsu 公司推出的 MB8764，其指令周期为 120 ns，且具有双内部总线，从而处理的吞吐量发生了一个大的飞跃。而第一个高性能的浮点 DSP 芯片是 AT&T 公司于 1984 年推出的 DSP32。

在这么多的 DSP 芯片种类中，最成功的是美国得克萨斯仪器公司(Texas Instruments, TI)的一系列产品。TI 公司在 1982 年成功推出了第一代 DSP 芯片 TMS32010 及其系列产品 TMS32011, TMS32C10/C14/C15/C16/C17 等，之后相继推出了第二代 DSP 芯片 TMS32020, TMS320C25/C26/C28、第三代 DSP 芯片 TMS32C30/C31/C32、第四代 DSP 芯片 TMS32C40/C44、第五代 DSP 芯片 TMS32C50/C51/C52/C53 及集多个 DSP 内核于一体的高性能 DSP 芯片 TMS32C80/C82 等。

自 1980 年以来，DSP 芯片得到了突飞猛进的发展，DSP 芯片的应用越来越广泛。从运算速度来看,MAC(一次乘法和一次加法)时间已经从 20 世纪 80 年代初的 400 ns(如 TMS32010)降低到 40 ns(如 TMS32C40)，处理能力提高了 10 多倍。DSP 芯片内部关键的乘法器部件从 1980 年的占模区的 40%左右下降到 5%以下，片内 RAM 增加一个数量级以上。从制造工艺来看，1980 年采用 4 μ m 的 N 沟道 MOS 工艺，而现在则普遍采用亚微米 CMOS 工艺。DSP 芯片的引脚数量从 1980 年的最多 64 个增加到现在的 200 个以上，引脚数量的增加，意味着结构灵活性的增加。此外，随着 DSP 芯片的发展，DSP 系统的成本、体积、质量和功耗都有很大程度的下降。

表16.1给出了 TI 公司生产的 DSP 芯片详细划分,还可将该公司的 DSP 芯片分成 C2000 系列、C5000 系列和 C6000 系列，它们都有各自不同的应用领域。

表 16.1 TI 公司的 DSP 芯片系列

	所属系列	位 宽	出现时间	备 注
定点 DSP	TMS320C1x 系列，第一代	16 bit	1982 年前后	
	TMS320C2x 系列，第二代	16 bit	1987 年前后	
	TMS320C5x 系列，第五代	16 bit	1993 年	
	TMS320C54x 系列，第七代	16 bit	1996 年	
	TMS320C24x 系列，第七代	16 bit	1996 年	
	TMS320C6x 系列，第七代	32 bit	1997 年	
	TMS320C55x 系列，第七代	16 bit	2000 年	功耗最低
浮点 DSP	TMS320C3x 系列，第三代	32 bit	1990 年	
	TMS320C4x 系列，第四代	32 bit	1990 年	
	TMS320C67x 系列，第七代	64 bit	1998 年	速度最快
多处理器 DSP	TMS320C8x 系列，第六代	32 bit	1994 年	

TMS320 C2x 和 TMS320 C4x 称为 C2000 系列，主要用于数字控制系统。
TMS320 C54x 和 TMS320 C55x 称为 C5000 系列，主要用于功耗低、便于携带的通信终端。
TMS320 C62x 和 TMS320 C64x 和 TMS320 C67x 称为 C6000 系列，主要用于高性能复杂的通信系统，如移动通信基站。

16.2.3 DSP芯片选型

设计 DSP 应用系统时，选择 DSP 芯片是非常重要的一个环节。只有选定了 DSP 芯片，才能进一步设计外围电路和系统的其他电路。总的来说，DSP 芯片的选择应根据实际应用系统的需要而确定。

DSP 芯片的综合性能指标除了与芯片处理能力直接相关外,还与片内、片外数据传输能力相关。DSP 芯片的数据处理能力通常用处理速度来衡量;数据传输能力用内部总线和外部总线的配置,以及总线或 I/O 口的数据吞吐率来衡量。

一般来说,选择 DSP 芯片时应考虑如下诸多因素。

(1) DSP 芯片的运算速度。运算速度是 DSP 芯片的一个最重要的性能指标,也是选择 DSP 芯片时所需要考虑的一个主要因素。DSP 芯片的运算速度可以用以下几种性能指标来衡量:

- 指令周期,就是执行一条指令所需要的时间,通常以 ns 为单位。
- MAC 时间,即一次乘法加上一次加法的时间。
- FFT 执行时间,即运行一个 N 点 FFT 程序所需的时间。
- MIPS,即每秒执行百万条指令。
- MOPS,即每秒执行百万次操作。
- MFLOPS,即每秒执行百万次浮点操作。
- BOPS,即每秒执行十亿次操作。

(2) DSP 芯片的价格。

(3) DSP 芯片的硬件资源。

(4) DSP 芯片的开发工具。

(5) DSP 芯片的功耗。

(6) 其他的因素,如封装的形式、质量标准、生命周期等。

16.3 TI DSP开发集成环境CCS简介

16.3.1 CCS的简介

CCS(Code Composer Studio)是 TI 公司推出的用于开发 DSP 芯片的集成开发环境,采用 Windows 风格界面,集编辑、编译、链接、软件仿真、硬件调试及实时跟踪等功能于一体,极大地方便了 DSP 系统的开发与设计,是目前使用最为广泛的 DSP 开发环境之一。

CCS 有两种工作模式,即软件仿真器和硬件在线编程。软件仿真器工作模式可以脱离 DSP 芯片,在 PC 上模拟 DSP 的指令集和工作机制,主要用于前期算法实现和调试。硬件在线编程可以实时运行在 DSP 芯片上,与硬件开发板相结合进行在线编程和调试应用程序。

图16.2是 CCS 的主要组件示意图。

CCS 的主要功能如下:

(1) 工程项目管理工具对用户程序实行项目管理。在生成目标程序和程序库的过程中,建立不同程序的跟踪信息,通过跟踪信息对不同的程序进行分类管理;

(2) 具有集成可视化代码编辑界面,用户可通过其界面直接编写 C 语言程序、汇编语言程序、.cmd 文件等;

(3) 高性能的编辑器支持汇编文件的动态语法加亮显示,使用户很容易阅读代码,发现语法错误;

(4) 含有集成代码生成工具,包括汇编器、C 编译器、链接器等,将代码的编辑、编译、链接和调试等诸多功能集成到一个软件环境中;

(5) 基本调试工具具有装入执行代码(.out)、查看寄存器、存储器、反汇编、变量窗口等功能,并支持 C 源代码级调试;

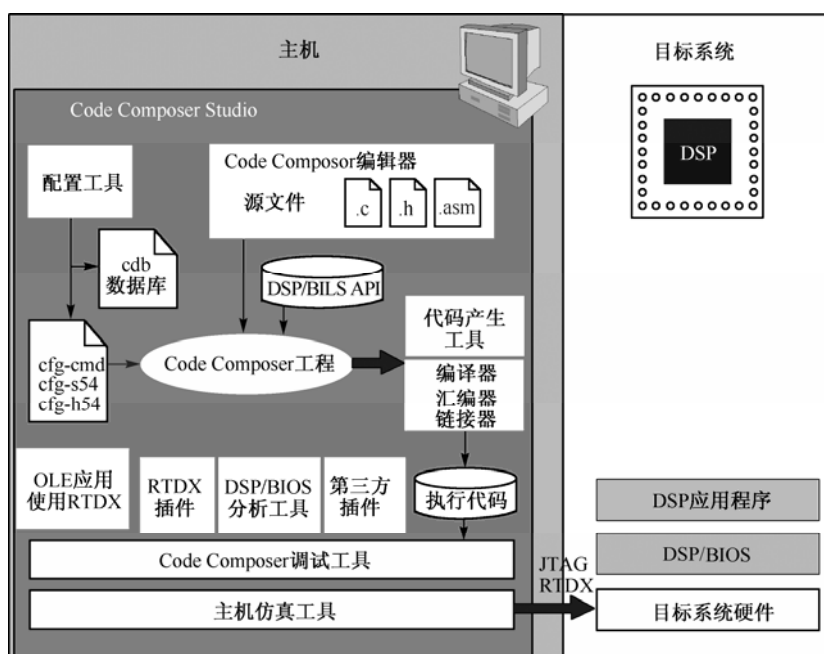


图 16.2 CCS 主要组件示意图

(6) 探针工具 (probe point)，可用于算法的仿真、数据的实时监控等；

(7) 断点工具，能在调试程序的过程中，完成硬件断点、软件断点和条件断点的设置；

(8) 分析工具，包括模拟分析和仿真器分析，可用于模拟和监视硬件的功能、评价代码执行的时钟；

(9) 数据的图形显示，可以将运算结果用图形显示，包括显示时域 / 频域波形、眼图、星座图、图像等，并能进行自动刷新；

(10) 提供 GEL 工具，利用 GEL 扩展语言，用户可以编写自己的控制面板 / 菜单，设置 GEL 菜单选项，可以方便直观地修改变量和配置参数；

(11) 支持多 DSP 调试；

(12) 支持 RTDX (Real Time Data Exchange) 技术，可在不中断目标系统运行的情况下，实现 DSP 与其他应用程序 (OLE) 的数据交换；

(13) 提供 DSP/BIOS 工具，增强对代码的实时分析能力，减小开发人员对硬件资源熟悉程度的依赖性。

16.3.2 CCS 的安装与使用

下面介绍 CCS2.2 及其升级包的安装，并简单介绍 CCS 的使用。

(1) 双击 SETUP.EXE，开始 CCS 的安装，如图 16.3 所示，选择安装 Code Composer Studio。

(2) CCS 安装完成后，开始安装 CCS2.2 下 C5000 系列的升级包，如图 16.4 所示。

图 16.5 是 CCS2.2 升级包安装过程的一个画面。CCS 的升级包可以在 ti 的官方网站 www.ti.com 上下载。

(3) 假设 CCS 的安装路径为 C:\ti，将 C:\ti\c5000\dsplib 中的有关库文件和批处理文件复制到 C:\ti\c5000\cgtools\lib 目录下，并覆盖相同的文件，如图 16.6 所示。

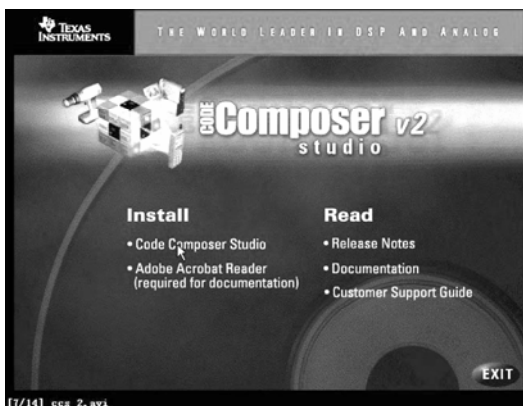


图 16.3 CCS 安装界面

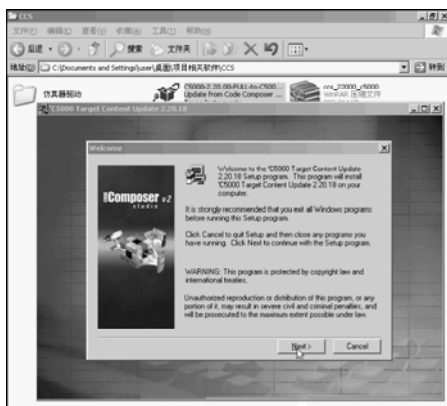


图 16.4 CCS 升级包的安装



图 16.5 CCS 2.2 升级包安装过程中的一个

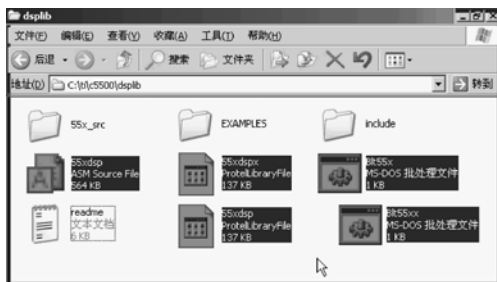


图 16.6 复制并覆盖有关的库文件和批处理文件

同时将 C:\ti\c5000\dsplib\include 下的相关头文件复制到 C:\ti\c5000\cgtools\include 目录下,并覆盖相同的文件,如图16.7所示。



图 16.7 复制并覆盖有关的库文件和批处理文件

(4) CCS 安装完成后，会在桌面上建立两个图标：CCS2 (‘C5000) 和 Setup CCS2 (‘C5000)。双击 Setup CCS2 (‘C5000) 对 CCS 的工作模式进行配置，如果是配置成硬件在线编程模式，需要先安装有关硬件仿真器的驱动，具体安装过程和设置请参考硬件仿真器厂商提供的资料。

CCS 设置好后，可以直接启动 CCS 或是在桌面上双击 CCS2 (‘C5000)图标启动 CCS，如图16.8所示，选择菜单“project”→“open”打开一个项目。

(5) 示例项目中使用了 DSP/BIOS™，如图16.9所示。DSP/BIOS™是 TI 公司为了方便 DSP 系统的开发而提供的一个简易嵌入式操作系统，它大大方便了用户编写多任务应用程序，减小了编程时对硬件的依赖性，使得用户能够把更多精力放在算法的高效实现上。读者可以参考有关的文献书籍以掌握这个高效工具。

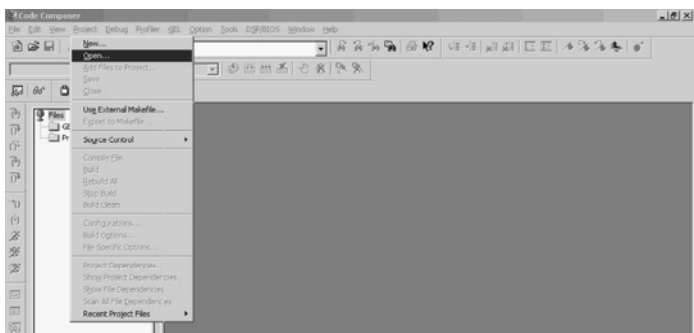


图 16.8 启动 CCS，打开一个项目

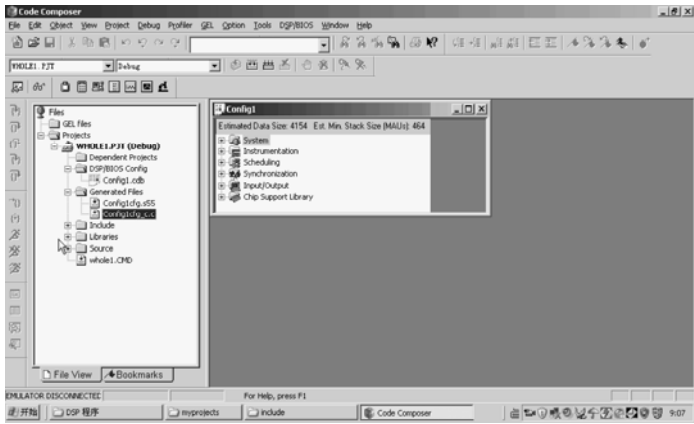


图 16.9 示例项目中使用的 DSP/BIOS™

(6) 程序的调试。开发者可以像在其他集成窗口开发环境中一样进行项目的调试，如图 16.10 所示。“Debug”菜单下有一些常用的调试方式，“Profiler”菜单下有一些工具可以使得用户能够分析程序的性能，当用户分析整个程序中哪些函数是关键函数时这个工具是非常有用的，而且利用这些工具可以很方便地比较各种算法的速度等性能。

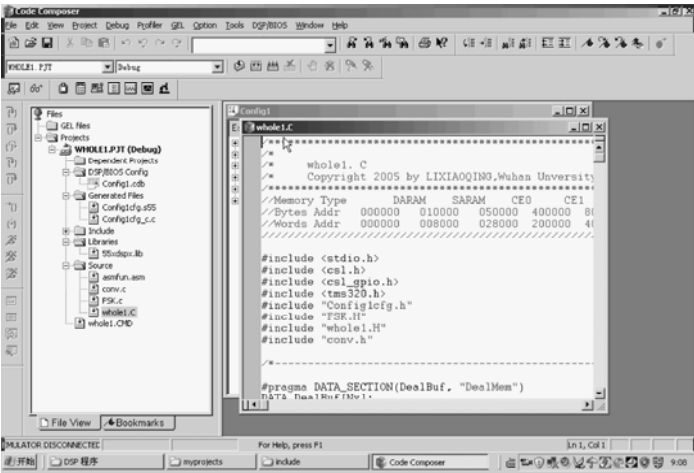


图 16.10 程序的调试

(7) 装载和硬件调试。与单片机的硬件仿真不同，DSP 硬件仿真需要将调试通过的程序装载到 DSP

硬件平台上进行硬件仿真，如图 16.11 所示，选择“File”菜单下的“Load Program...”命令装载项目生成的*.out 文件。硬件调试和(6)中介绍的调试方法基本一样。

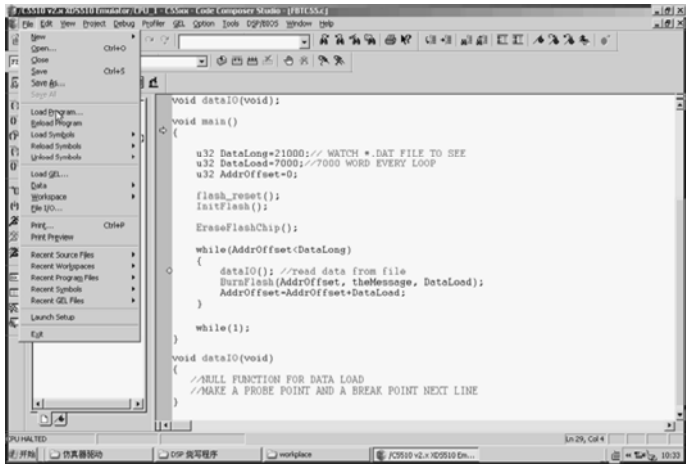


图 16.11 装载与硬件调试

16.4 基于DSP的水声通信终端设计

16.4.1 水声通信系统介绍

由于水声数字通信系统只是数字通信系统在水声信道环境下的特殊应用，因此其基本系统组成与一般数字通信系统大体相同，图 16.12 给出了水声数字通信系统的组成方框图。

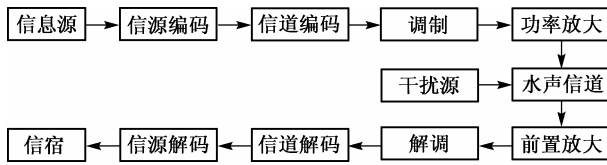


图 16.12 水声数字通信系统组成方框图

在实际应用系统中，为解决控制和信号处理的任务分配问题，一般采用 DSP 处理器和单片机协同工作的方法。使用 DSP 处理器专注于系统信号处理算法的实现，充分利用它的高速数据处理能力，以达到实时性要求；终端中添加的 MCU 单元负责进行控制和人机交互方面的工作。

水声数字通信系统终端的整体结构框图如图 16.13 所示。下面简单介绍基于 DSP 的水声通信系统终端关键模块的实现方法。

16.4.2 前端信号处理模块

1. 前置放大电路

前端信号处理模块用于完成换能器信号的采集和调理。此模块的结构框图如图 16.14 所示。

水声换能器的输出信号为差分信号，选取仪表放大器 AD620 完成前端信号提取和一级放大。仪表放大器具有低失调电压、低失调电压漂移、低噪声和低功耗等显著特点，AD620 芯片只用一只外部电阻就能设置放大倍数 1~1000。假设 G 表示放大倍数，设置电阻表示为 R_G ，两者满足公式如下：

$$R_G = \frac{49.4}{G - 1} \text{ k}\Omega \tag{16.1}$$

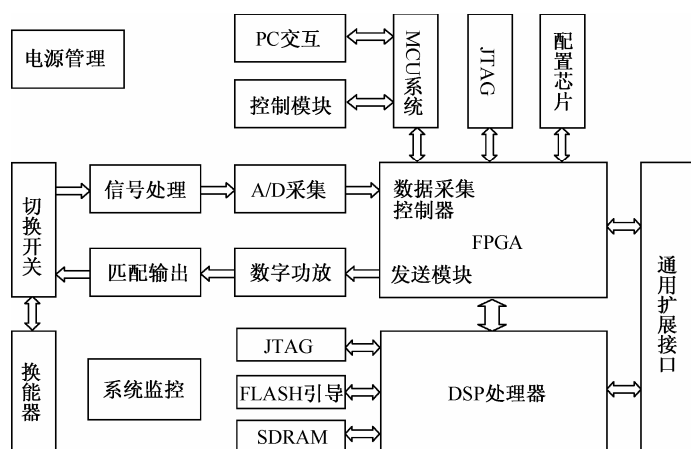


图 16.13 水声数字通信系统终端的整体结构框图

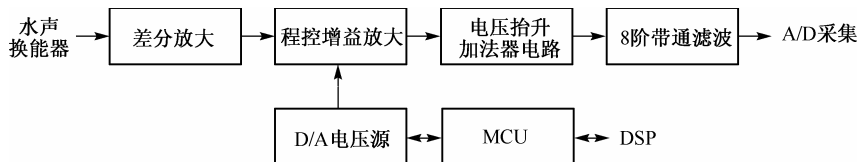


图 16.14 前端信号处理模块结构框图

在实际应用中, AD620 可以采用共模屏蔽驱动接法, 如图 16.15 所示。对换能器的屏蔽给予适当的驱动, 可减小电缆电容和杂散电容造成的差分相移, 保证交流共模抑制比不下降, 减小输入端的噪声干扰。

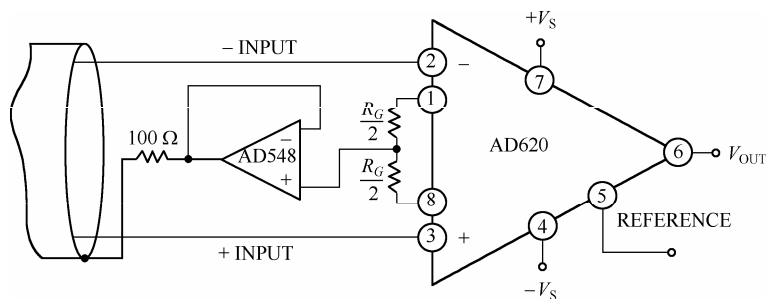


图 16.15 AD620 共模屏蔽驱动接法示意图

2. 程控增益放大电路

选取程控增益放大芯片进行增益控制, VCA610 是宽带连续可变电电压控制增益放大芯片。它具有高输入阻抗线性 dB 控制增益, 增益控制范围达到 80 dB (−40~+40 dB), 线性对应于控制电压 0~2 V。增益计算公式为

$$G(\text{dB}) = -40 - 40V_C$$

转换成电压放大倍数为

$$G = 10^{-2(V_C+1)} \quad (16.2)$$

假设使用数字的方法产生控制电压, 电压步距为 0.1 V。

$$G_1 = 10^{-2(V_{C1}+1)}, \quad G_2 = 10^{-2(V_{C2}+1)}$$

$$V_{C1} = V_{C2} - 0.1$$

$$\frac{G_2}{G_1} = \frac{10^{-2(V_{C2}+1)}}{10^{-2(V_{C1}+1)}} = 10^{2(V_{C1}-V_{C2})} = 10^{2(V_{C1}-V_{C2})} = 10^{0.2}$$

即程控增益放大器的各挡放大倍数为

$$L, (10^{0.2})^{-1}, 1, 10^{0.2}, (10^{0.2})^2, L$$

设 A/D 量化范围为 0~3 V, $3V/10^{0.2} \approx 1.89$ V, 即在控制电压步进 0.1 V 的情况下, VCA610 的输出信号可被调整在 1.89~3 V 的峰值区间。

3. 前端滤波器电路

抗混叠滤波器用来滤除连续信号中高于折叠频率的频谱分量, 使采样信号基本频谱在低频段中尽可能不混有连续信号的高频分量, 保证采样信号的基本频谱部分和连续信号频谱尽可能接近。

考虑到设计效率及可靠性等因素, 滤波器设计采用开关电容滤波器。MAX7490 是具有两个等同滤波模块、低功耗、低电压、宽动态范围的二阶开关电容滤波器。两等同结构分别通过 2~4 个外部电阻, 可编程为二阶带通、低通、高通、带阻功能滤波器。可以通过二阶滤波模块级联而成四阶滤波器; 同样, 更高阶滤波器可以通过多片 MAX7490 级联而成。MAX7490 的引脚如图 16.16 所示。

MAX7490 的主要特点如下:

- Q 值精度高达 $\pm 0.2\%$, 时钟-中心频率误差为 $\pm 0.2\%$;
- 双两阶模块;
- 16 引脚 QSOP 封装;
- 单电源供电;
- 内部或外部时钟编程;
- 低通、高通、带通、带阻多种功能;
- 时钟-中心频率比率为 100:1;
- 内部采样-中心频率比率为 200:1;
- 中心频率高达 40 kHz;
- 方便级联;
- 低功耗关闭模式小于 1 μ A 的电流。

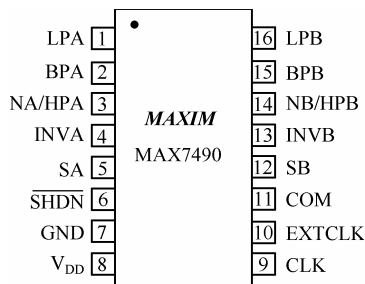


图 16.16 MAX7490 的引脚图

16.4.3 A/D 数据采集接口

对于数字通信系统而言, A/D 转换器的选取在考虑精度要求的同时还应满足实时通信对于转换速率的要求, 由于本设计的水声系统声波工作频段为 15.5~18.5 kHz, 因此要求 A/D 转换器的采集速率不低于 100 ksps (sample per second)。可选取 TI 公司的 TLV1572 A/D 转换器。TLV1572 是 TI 公司生产的高速十位串行 A/D 转换芯片, 要求提供的供电电压在 3~5 V 之间, 最小参考电压为 2.7 V。在 5 V 供电时, 最高转换速度为 1.25 Msps, 在 3 V 供电时最高转换速度为 625 ksps。为了使其达到最大的转换速率, 两种情况下所提供的时钟分别为 20 MHz 和 10 MHz。TLV1572 采用的是 8 引脚的 SOIC 封装, 如图 16.17 所示。它可以通过 3 个或 4 个串行口线直接与 DSP 或其他数字微处理器串口相连, 不需要外加逻辑, 但是转换速度受 SCLK 供给时钟的限制。当 $\overline{CS}$ 为高电平时, A/D 芯片各引脚处于三态状态。在 $\overline{CS}$ 由高电平变低电平时, TLV1572 检测 FS 引脚的状态来确定工作模式。若 FS 为低电平时, 则为 DSP 模式 (DSP mode), 若 FS 为高电平时, 则为 SPI/QSPI 兼容微处理器模式 (μ C mode)。

本系统设计 ADC 的采集速率为 100 ksps, DSP 的 McBSP 通道与 A/D 芯片 TLV1572 的接口采用图 16.18 所示的方式, 可通过修改 FPGA 内部的时序模块灵活地设置采样速率。

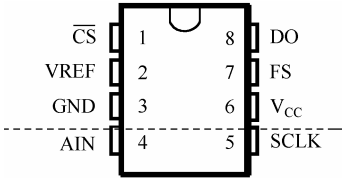


图 16.17 TLV1572 的引脚脚图

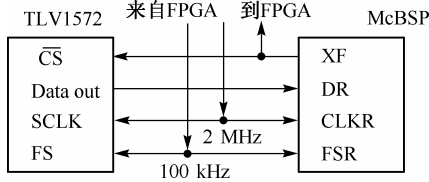


图 16.18 McBSP 与 TLV1572 的接口图

FPGA 内部构造的 McBSP 通道与 TLV1572 的接口仿真波形如图16.19所示。

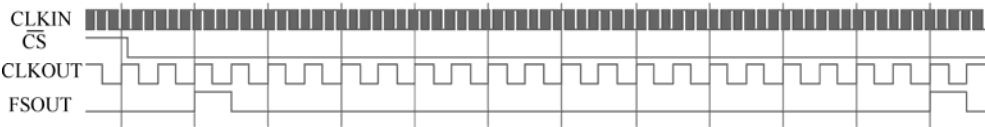


图 16.19 FPGA 内部构造的 McBSP 通道与 TLV1572 的接口仿真波形图

16.4.4 DSP信号处理模块

DSP 信号处理模块由 TI 公司 TMS320VC5510A 处理器和 Altera 公司的 Cyclone 系列 FPGA EP1C6 及其外围器件组成。

DSP 信号处理模块主要完成数据的 A/D 转换控制、采样数据的 DMA 获取及运行相应的信号处理算法。所设计的数据处理模型应当通用性强，只需修改核心算法部分就可以用于不同算法的研究。该模块结构如图16.20所示。

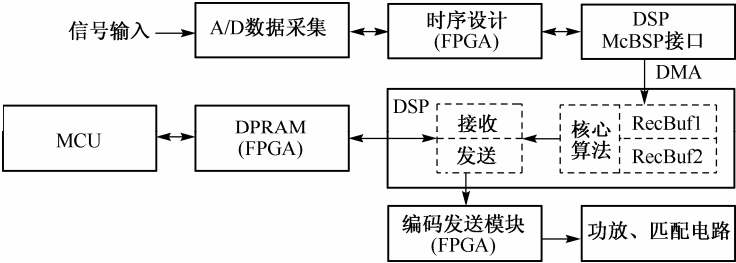


图 16.20 DSP 信号处理模块结构框图

高速串行 A/D 转换芯片的转换时序由 FPGA 内部构造电路实现，串行数据输出到 DSP 的多通道缓冲串行接口 McBSP。DSP 内部编程设置直接存储器存取通道 DMA，McBSP 接收到的数据被直接移动到设定的 DSP 内部存储空间，并且空间地址自动增加。当接收数据达到设定的一帧 (frame) 的元素 (element) 数目时，DMA 产生帧结束中断。帧结束中断处理程序完成两个 DSP 内部存储空间的交替和 DMA 使能，并且标志完成接收数据块的有效。核心算法的运算要求在下一帧接收中断结束前完成；当检测到一个存储空间数据有效就开始调用算法对该空间数据进行处理，在此期间，另一个存储空间正在进行 DMA 数据存储。

字节地址(Hex) 存储块划分 块长度 (字节)

000000	DARAM(8块)	65 536
010000	SARAM(32块)	262 144
050000	外部-CE0	3 866 624
400000	外部-CE1	4 194 304
800000	外部-CE2	4 194 304
C00000	外部-CE3	4 161 536
FF8000	ROM	32 768
FFFFFF	外部-CE3	

MP/MC=0 MP/MC=1

图 16.21 存储器空间映射图

下面讨论一下有关 DSP 外部存储器扩展的相关内容。
DSP 芯片 C5510/5510A 支持统一的存储空间映射(程序和数据的访问是针对一个相同的物理空间)，存储器空间映射如图16.21所示。

DSP 使用专门的外部存储器接口 (EMIF) 来扩展存储器单元。图 16.22 为 EMIF 的接口连线图。C5510/5510A 的 EMIF 有 22 位的地址总线、32 位的数据总线、4 个片选输出和多种控制信号, 能支持多种外部存储器器件, 如表 16.2 所示。

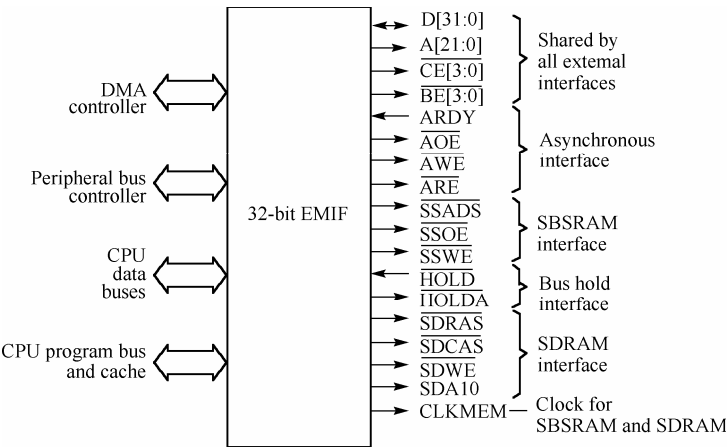


图 16.22 EMIF 的接口原理图

表 16.2 EMIF 接口的存储器类型

存储器类型	存 储 器	EMIF 时序特征	优 点
异步	SRAM, EPROM, Flash	可编程时序	可以提供高度灵活的存储器件时序
同步	同步突发 SRAM (SBSRAM)	运行于 1/2 CPU 时钟速率或 CPU 时钟速率	可以支持高速的相同吞吐量的存储器
	同步 DRAM (SDRAM)	运行于 1/2 CPU 时钟速率或 CPU 时钟速率	可以支持高速的、高密度的 SDRAM

TMS320VC5510 DSP 的外部存储器接口 EMIF 能够提供高度灵活的接口方式, 每个片选空间都可以连接不同类型的存储器件, 单独设置读写时序等参数。它支持的接口方式有: 异步接口、同步突发 SRAM (SBSRAM) 接口和同步动态 RAM (SDRAM)。每种接口方式都支持程序代码的访问、32 比特数据访问、16 比特数据访问和 8 比特数据访问。

系统使用异步接口扩展 FLASH 存储器空间和 FPGA 内部双口 RAM 空间, 使用同步接口扩展 SDRAM 存储器空间。下面分别对异步接口和 SDRAM 接口的连接方法、读写时序等给出详细的说明。

(1) 异步接口

EMIF 支持 32 比特、16 比特和 8 比特数据的异步访问 (8 比特的异步写操作不被允许)。EMIF 接口与访问的异步存储器之间的信号连线如图 16.23 所示。

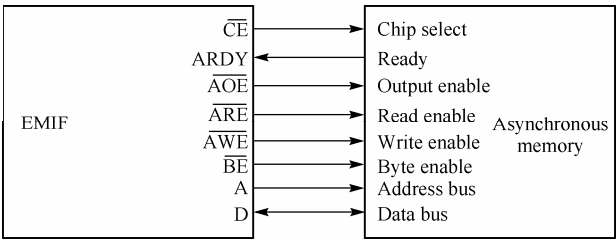


图 16.23 EMIF 连接与访问的异步存储器之间的信号连线图

EMIF 异步接口时序的可编程参数有 5 个。

- 建立周期 (Setup Period): 地址、片选使能 ($\overline{CE}$) 和字节使能 ($\overline{BE}$) 信号有效之后, 读选通信号

($\overline{\text{ARE}}$)或写选通信号($\overline{\text{AWE}}$)下降沿之前的 CPU 时钟周期。对于异步读操作,它也是输出使能信号($\overline{\text{AOE}}$)有效之后, $\overline{\text{ARE}}$ 下降沿之前的 CPU 时钟周期。

- 选通周期(Strobe Period):从读/写有效到无效(读选通或写选通下降沿到上升沿)之间的 CPU 时钟周期。
- 保持周期(Hold Period):读/写信号上升沿后,地址和字节使能信号保持有效的 CPU 时钟周期。对于异步读操作,它也是 $\overline{\text{ARE}}$ 信号上升沿后,输出使能信号有效的 CPU 时钟周期。
- 延长保持时间(Extended Hold):延长的保持时间是插入的附加的 CPU 周期,它需要插入的情况是:(a) EMIF 的下一访问地址在与当前访问空间不属于同一 CE 空间;(b) 下一访问与当前一次访问的方向相反,如当前操作为读操作,而下一操作为写操作。
- 超时值(Time-out Value):设定的一个单一的超时值对读、写操作都有效。在一次操作中,内部时钟对 ARDY 信号采样的低电平期间(表明存储器未准备好)的 CPU 时钟周期计数,如果计数超过超时值,EMIF 在总线错误状态寄存器中记录错误。若 EMIF 全局控制寄存器中

ARDYOFF 标志置 1,则 ARDY 信号不被采样,超时条件无效。

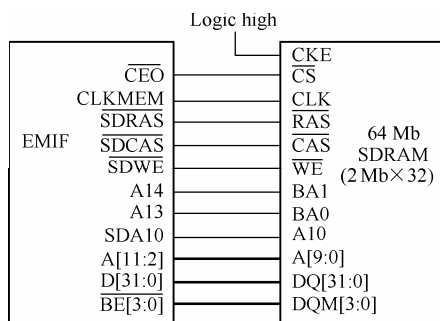


图 16.24 EMIF 与 64 Mb SDRAM
2 Mb×32 阵列接口图

(2) SDRAM 接口

EMIF 支持与高密度高速同步 DRAM 存储器的无缝连接,SDRAM 按照行和列的方式来组织存储阵列,并通过行列地址对数据进行读和写。TMS320VC5510 支持 16 位或 32 位宽度的 64 Mb 或 128 Mb 的 SDRAM 芯片。对于 SDRAM,EMIF 访问速度可以被配置为 1/2 或 1 倍的 DSP CPU 时钟周期。下面以扩展 32 位宽度的 64 Mb SDRAM 为例,给出接口示意图,如图 16.24 所示。

16.5 卷积码编解码实现

上面介绍了基于 TMS320VC5510 的水声数字通信系统的硬件设计,水声数字通信系统中为了抗多径衰落和干扰噪声的影响,还应当采用信道编码,本节将重点介绍基于 TMS320VC5510 的卷积码的编解码实现。

16.5.1 卷积码编码

考虑(2, 1, 7)卷积码,两个生成多项式分别为 $\text{POLYA} = 0x6d$, $\text{POLYB} = 0x4f$ 。

编码输入 16 位数据,输出 32 位编码数据。输入数据的 MSB 首先进入编码器,PLOYA 的编码结果首先输出,PLOYB 的编码结果紧接 PLOYA。

图 16.25 是编码器的实现寄存器级框图。图中所示的数据流顺序是 C 语言软件实现时采用的(C 语言采用左移实现编码),相应地在 FPGA 中硬件实现编码时也最好是采用这种编码结构。

D0~D15 用做 16 位数据的暂存器,在时钟的控制下,数据位向左移动,左端补 0。

编码器由 7 级移位寄存器组成: D21, D20, D19, D18, D17, D16 和 D15, 因此图 16.25 还可以画成图 16.26 所示。

7 级移位寄存器可表示编码所有状态。规定数据移出端为 MSB,数据移入端 LSB(参见图 16.26),假设(D21, D20, D19, D18, D17, D16, D15)=(1000000),则对应的状态为 S_{64} 。

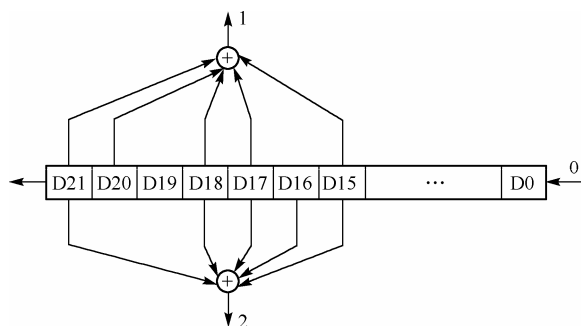


图 16.25 编码器的寄存器级框图

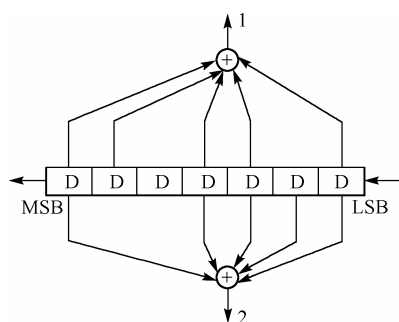


图 16.26 编码器等效框图

卷积码编码在上面移位寄存器左移 16 位之后，应当继续左移 7 位使得状态归到 0 状态，即所谓迫零编码，这样可以提高纠错的性能。

图 16.26 所示的编码器可以直接使用硬件实现，也可以软件通过位操作的方法或是查找表实现。

系统设计采用 (2, 1, 7) 卷积码， $k=7$ ，即编码 7 个数据时可能经历所有的状态。图 16.27 是 (2, 1, 7) 卷积码编码的网格图。

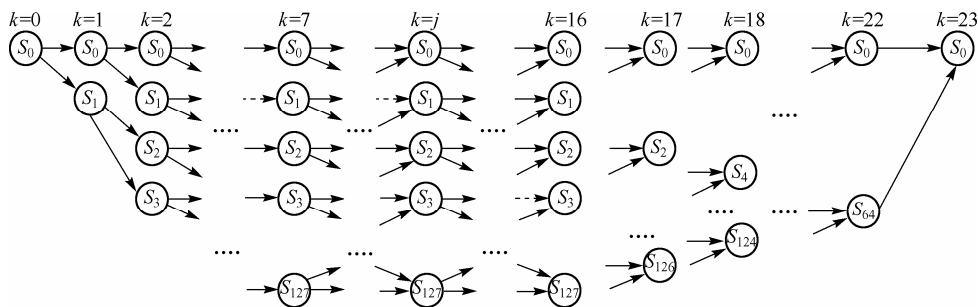


图 16.27 (2, 1, 7) 卷积码编码的网格图

16.5.2 卷积码解码

1. 编码特点

观察图 16.26 所示的编码器，在输入相同情况下，不论 MSB=0 或 MSB=1，只要 (D20, D19, D18, D17, D16, D15) 相同，则对应的下一个状态也是相同的，因为 MSB 被移出了寄存器 (对应的编码输出可能不相同)。因此，根据图 16.27 有下面的关系。

(1) 当 MSB=0 时，假设对应的状态为 S_n ($0 \leq n \leq 63$)，若移入为 1 时，对应的下一个状态为 S_{2n+1} ；若移入为 0 时，对应的下一个状态为 S_{2n} 。

(2) 当 MSB=1 时，可假设对应的状态为 S_{n+64} ($0 \leq n \leq 63$)，若移入为 1 时，对应的下一个状态为 S_{2n+1} ；若移入为 0 时，对应的下一个状态为 S_{2n} 。

综合以上，有图 16.28 所示的关系。图中实线表示输入 0 的路径，虚线表示输入 1 的路径。

关系 (1) 和 (2) 是从图 16.28 的左端出发得到的，相应地从图 16.28 的右端出发可得出下面的关系 (3) 和 (4)。

(3) 观察图 16.28 的右端发现，编码器的任一状态 (经历了全部状态以后) 有两个路径输入，一个路径输入对应的前一级状态号小于 64，另一个路径输入对应的状态号为 $64 \leq n+64 \leq 127$ 。

若 n 为偶数，则对应的前面两个状态为 S_n 和 S_{n+64} 。

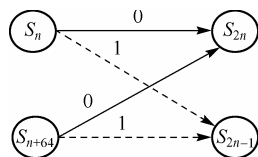


图 16.28 编码状态之间关系

若 n 为奇数, 则对应的前面两个状态可能为 $S_{(n-1)/2}$ 和 $S_{64+(n-1)/2}$ 。

(4) 观察图 16.28 还可以发现, 偶数状态号都是由输入 0 引起的, 奇数状态号都是由输入 1 引起的。

利用关系(1)和(2)可以生成度量矩阵, 预先得到每一状态在不同输入时的下一个状态输出, 而关系(3)和(4)可以直接用于维特比解码算法中。

注意, 上面的状态号的标示是按照本节前面规定的表示方法, 若采用不同的表示方法, 则得到的关系也将不同, 这一点在程序编写时要特别注意。

2. 维特比译码

卷积码译码采用维特比译码算法, 使用 C 语言实现。

下面先简述维特比译码算法的主要步骤:

(1) 从某一单位时间 $j = m$ 开始, 对进入每一状态的所有分支的部分路径计算其部分路径度量。对每一个状态挑选并存储一条有最大度量的部分路径及其部分度量值, 称此部分路径为留选路径。

(2) j 增加 1, 把此时刻进入每一状态的所有分支度量, 和同这些分支相连的前一时刻的留选路径的度量相加, 得到了此时刻进入每一状态的留选路径, 加以存储并删去其他所有路径, 因此留选路径延长了一个分支。

(3) 若 $j < l + k$, 则重复上面步骤, 直至译码器得到了最大路径度量的路径。由维特比算法得到的路径一定是最大似然路径, 沿着维特比算法得到的路径返回即可解码。

上面 k 表示卷积码的卷积长度, l 表示输入数据的长度, 本系统设计中 $k = 7$, $l = 16$ 。

对于二进制对称信道而言, 计算和寻找有最大度量的路径等价于寻找与 R 有最小汉明距离的路径, 即寻找

$$\arg \min d(R, C_j) \quad j = 1, 2, L, 2^{k+l} \quad (16.3)$$

其中, R 为接收到的序列; C 为发送序列。

3. 译码流程

(1) 初始化。求出对应每一状态的两条路径的编码输出, 并存储结果。

(2) 对于第 j 级, 考虑所有可能状态的所有部分分支的度量, 并对每一状态保留一个分支度量最大的部分路径, 即留选路径。

(3) 当 $j = l + k$ 时, 接收数据部分分支比较完毕。比较所有状态的度量值, 选取度量最大(汉明距离最小)的路径返回, 即完成解码。

16.5.3 交织器及维特比译码算法的部分程序代码

采用(2, 1, 7)卷积码编码, 相关长度为 7 位, 输入 16 位的数据后再输入 7 位 0 以使卷积码的编码状态归 0, 即有 16 位数据输入, 编码 46 ($2 \times (16 + 7)$) 位数据输出。编码效率为

$$\eta = 16/46 = 34.8\%$$

卷积码的纠错能力和错误的位置有关系。在图 16.26 所示编码器中, 最先输出的数据的纠错能力最强, 最末输出的数据纠错能力最弱。软件仿真时得到编码的纠错能力为 12~6 bit (包含上、下限)。因此, 在编码的输出端加一个交织器, 以改善纠正突发错误的能力。

图 16.29 为系统设计中与卷积码配套使用的块交织器示意图。卷积码采用了迫零卷积码, 交织器中最后一位不参与交织, 因为最后一位是使编码器归零的输出, 对于接收端可以直接确定它为 0。

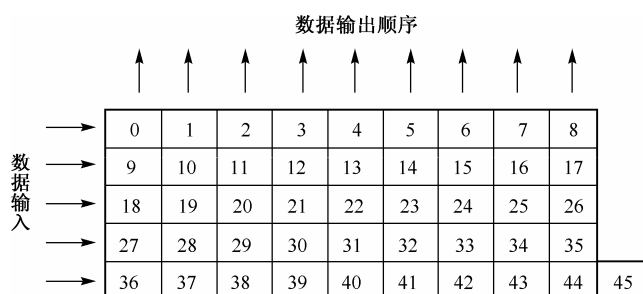


图 16.29 卷积码对应的块交织器示意图

下面是卷积码的维特比译码算法的状态度量、路径选择和回溯部分程序。

```

#include "math.h"
#include "va.h"
#include "conv.h"

#define K 7 //
#define L 16 //data transfer length
#define POLYA 0x006d //code generator polynomials
#define POLYB 0x004f

// n -----> 2n
// - ->
// -
// - ->
// n+64 -----> 2n+1
int cov_mettab[2][128]; //初始化的度量表，即编码的输出
int cov_state[L+K+1][128]; //度量状态，多保留一位 0 状态
int inbuf[L*2+K*2]; //接收到的符号
//int Rec[L]; //接收到的数据
//int outbuf[L]; //解码输出
int decout; //解码 16 位输出
int outbuf[L+K];
void symbol2rec(int *, int *); //第一个指针指向接收到的符号数组，第二个指针指向
//组装之后的缓冲区

void getmet(void);
void aacs(int *);
int backsearch(int *);
int compare(int, int);
int power2(int);
int power2(p)
{
    int i;
    int temp;
    temp=1;

    for(i=0; i<p; i++)
        temp=temp*2;

    return temp;
}

int compare(a, b)

```

```

{
    //int i;
    int cnt;
    int temp;
    temp = a^b;
    switch(temp)
    {
        case 0: cnt=0; break;
        case 1: cnt=1; break;
        case 2: cnt=1; break;
        case 3: cnt=2; break;
        default: cnt = 127;
    }
    return cnt;
}

void getmet(void)
{
    int e,i;
    int temp;

    for(i=0;i<64;i++){
        temp = i;
        temp = temp << 1;           //input 0
        e = Partab[temp & POLYA] << 1;
        e |= Partab[temp & POLYB];
        cov_mettab[0][temp]=e;

        temp = temp + 1;
        e = Partab[temp & POLYA] << 1;
        e |= Partab[temp & POLYB];
        cov_mettab[0][temp]=e;
    }

    for(i=64;i<128;i++){
        temp = i;
        temp = temp << 1;
        e = Partab[temp & POLYA] << 1;
        e |= Partab[temp & POLYB];
        cov_mettab[1][2*(i-64)]=e;

        temp = temp + 1;
        e = Partab[temp & POLYA] << 1;
        e |= Partab[temp & POLYB];
        cov_mettab[1][2*(i-64)+1]=e;
    }
}

void acs(int Rec[])
{
    int i,j,n;
    int temp;
    int preindex,premet0,premet1;
    int m0,m1;

```

```

for(i=0;i<128;i++)          //initialization for state 0
{
    cov_state[0][i]=0;
}
for(i=1;i<=K;i++)           //decoding for state number that smaller than 64
{
    temp=power2(i);
    //m0=compare(Rec[i],mettab[0][0]);
    for(j=0;j<temp;j++)
    {
        if(j%2)              //here for state number that is smaller than 64, only
            n=(j-1)>>1; //have the one half of the butterfly architecture
        else
            n=j>>1;
        preindex = n << 8; //calculate the previous state number (index)
        cov_state[i][j] = compare(Rec[i-1], cov_mettab[0][j]) + (cov_state[i-1]
[n] & 0x00ff);

        cov_state[i][j] = preindex + cov_state[i][j];
    }
}

for(i=K+1;i<=L+K;i++)
{
    for(j=0;j<128;j++)
    {
        if(j%2)
            n=(j-1)>>1;
        else
            n=j>>1;
        premet0 = cov_state[i-1][n] & 0x00ff; // get the met of the previous state
        premet1 = cov_state[i-1][n+64] & 0x00ff;
        m0=compare(Rec[i-1],cov_mettab[0][j]) + premet0;
        m1=compare(Rec[i-1],cov_mettab[1][j]) + premet1;
        //premet0 and premet1 index 0 & 1 represent that the previous state
        //index is bigger or smaller than 64
        //the same meaning for cov_mettab[ ][ ]'s upper index
        if(m0<m1)
        {
            preindex = n;
            preindex = preindex << 8;
            cov_state[i][j]=preindex + m0;
        }
        else
        {
            preindex = n+64;
            preindex = preindex << 8;
            cov_state[i][j]=preindex + m1;
        }
    }
}
}

```



```
int backsearch(int outbuf[]) //Note that the selection operation of standard ACS
{                               //is implemented here
    int i,j;
    int index;
    int preindex;
    int mettemp,temp;
    int rtn;

    mettemp=cov_state[L+K][0] & 0x00ff;

    index=0;
    for(i=1;i<128;i++)
    {
        temp = cov_state[L+K][i] & 0x00ff;
        if(mettemp>temp)
        {
            index=i;
            mettemp=temp;
        }
    }
    outbuf[L+K-1]=index%2;

    for(j=L+K-1;j>=1;j--)
    {
        preindex = cov_state[j+1][index];
        preindex = preindex >> 8;
        index = preindex;
        outbuf[j-1]=index%2;
    }
    rtn=0;
    for(i=0;i<L;i++)
        rtn=rtn+(outbuf[i]<<(L-1-i));
    return rtn;
}
```

16.6 本章小结

本章首先介绍了数字信号处理器的有关基本概念,数字信号处理器是为数字信号处理算法专门设计的一种微处理器,它使用硬件方式实现数字信号处理算法中常见的运算结构,并且采用改进哈佛结构解决存储器寻址的速度瓶颈限制。然后介绍了 TI 公司的 DSP 产品系列,并说明了如何根据需求选择合适的 DSP 芯片。最后介绍了基于 TMS320VC5510A 的水声数字通信的硬件设计和基于卷积码的水声信道编码,希望通过这个例子读者能够掌握基于 DSP 系统设计的一般方法,并了解在设计过程中需要注意的问题。

第 17 章 嵌入式操作系统

17.1 引言

前面的章节介绍了基于各种嵌入式处理器的硬件系统设计,随着技术的发展,处理器的功能变得越来越强大,要让所有的系统开发人员每开发一个新的系统时都从处理器的硬件开始学习,掌握新处理器的所有寄存器操作和汇编语言编程已是越来越不现实的。而嵌入式系统的应用却越来越广泛,设计过程越来越复杂,但新产品的上市时间要求越来越短,如何将各种基于 PC 平台的软件算法模块快速地移植到新的嵌入式平台上,已成为嵌入式产品开发人员迫切需要解决的问题。

嵌入式系统的发展过程大致经历了如下 4 个阶段。

(1) 无操作系统的嵌入式算法阶段

这一阶段的嵌入式系统是以单芯片为核心的系统,具有与一些监测、伺服、指示设备相配合的功能。一般没有明显的操作系统支持,而是通过汇编语言编程对系统进行直接控制。主要特点是系统结构和功能都相对单一,针对性强,无操作系统支持,几乎没有用户接口。

(2) 简单监控式的实时操作系统阶段

这一阶段的嵌入式系统主要以嵌入式处理器为基础、以简单监控式操作系统为核心。系统的特点是:处理器种类繁多,通用性比较弱;开销小,效率较高;一般配备系统仿真器,具有一定的兼容性和扩展性;用户界面不够友好,主要用来控制系统负载及监控应用程序的运行。

(3) 通用型嵌入式实时操作系统阶段

以通用型嵌入式实时操作系统为标志的嵌入式系统,如 VxWorks, pSos 和 Windows CE 就是这一阶段的典型代表。这一阶段嵌入式系统的特点是:能运行在各种不同的微处理器上;具有强大的操作系统的功能,如具备了文件和目录管理、多任务、设备驱动支持、网络支持、图形窗口及用户界面等功能;具有丰富的 API 和嵌入式应用软件。

(4) 以 Internet 为标志的嵌入式系统

伴随着通用型嵌入式实时操作系统的发展,面向 Internet 网络和特定应用的嵌入式操作系统正日益引起人们的重视,成为重要的发展方向。嵌入式系统与 Internet 的真正结合、嵌入式操作系统与应用设备的无缝结合代表着嵌入式操作系统发展的未来。

嵌入式实时操作系统应用十分广泛,包括数据通信、信息家电、航空航天、工业控制、生物医学电子、船舶工程、计算机外设、电信设备、交通运输、国防武器控制等领域,已经形成 IT 产业争夺的重点领域,它所带来的工业年产值已超过万亿美元。基于嵌入式实时操作系统的嵌入式系统开发已成为一种潮流。

嵌入式实时操作系统通过对处理器寄存器的抽象和系统外设驱动的添加裁剪,使得应用软件开发人员可以不用去关心嵌入式系统的硬件平台,从而专心于新的应用开发编程;同时嵌入式实时操作系统的使用也使得快速移植已有软件成果成为可能。嵌入式实时操作系统下的应用程序开发不同于普通应用开发的前后台模式,它使用更加先进的编程思想,使得系统能够更加有效地响应各种事件,使各种硬件平台的实时性能得到最大的发挥。另外,模块化的编程也使得系统更加稳定,而稳定性是某些嵌入式应用环境中首要考虑的因素。

本章将介绍有关嵌入式实时操作系统的基本知识，通过本章的学习，读者应掌握以下内容：嵌入式实时操作系统的基本概念；常见的嵌入式实时操作系统；嵌入式实时操作系统 $\mu\text{C}/\text{OS-II}$ 的移植。

17.2 嵌入式实时操作系统的基本概念

17.2.1 嵌入式实时操作系统的特点

由于微电子技术的进步和发展，单片系统硬件的规模越来越大、功能越来越强，从而给运行嵌入式实时操作系统提供了物质条件，于是出现了很多具有不同特点及不同应用领域的嵌入式实时操作系统。另一方面，嵌入式实时操作系统采用先进的编程思想，为软件开发人员提供了友好的开发平台：它一般可以提供多任务调度、维护管理、任务间通信和同步及内存管理等重要服务，从而使得应用程序易于设计、维护和扩展。因此，采用嵌入式实时操作系统开发的产品将会更加可靠，开发周期更加短。基于嵌入式实时操作系统的应用开发是当今嵌入式领域最热门的技术之一。

通常，把必须在有限时间内完成的任务称为实时任务，用来完成实时任务的系统称为实时系统。如果一个系统必须在极严格的期限内完成实时任务，否则就会产生灾难性的后果，那么这样的实时系统就称为硬实时系统。相对来说，如果系统完成任务的期限要求不是十分严格的，那么这种系统就称为软实时系统。对于硬实时系统来说，超过期限计算出来的结果是没有任何价值的；而软实时系统对于超时具有一定的容忍度，超过允许期限得到的运算结果不会完全没有意义，只是这个结果的可信度要有某种程度的降低，或者由此造成的后果还可以容忍。相对而言，实时操作系统的设计难度要大一些，而嵌入式实时操作系统与其他操作系统相比有如下特点。

- 微型化。嵌入式系统存储空间一般不会很大，电池的容量也较小，外部设备具有多样化，因而不会允许嵌入式操作系统占用较多的资源，所以在保证应用功能的前提下，嵌入式操作系统的规模越小越好。
- 实时性。由于对嵌入式操作系统的共同要求是能快速响应事件，具有较强的实时性，所以嵌入式实时操作系统的内核都是可剥夺型的。
- 可裁剪性。嵌入式操作系统运行的硬件平台多种多样，其宿主对象更是五花八门，所以要求嵌入式操作系统中提供的各个功能模块是可以让用户根据需要选择使用的，即它应具有良好的可裁剪性。
- 高可靠性。嵌入式系统广泛应用于军事武器、航空航天、交通运输、重要的生产设备领域，所以要求嵌入式系统必须有极高的可靠性。
- 易移植性。为适应多种多样的硬件平台，嵌入式操作系统应可以在不做大量修改的情况下稳定地运行于不同的平台上。

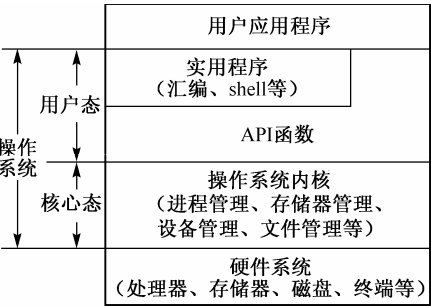


图 17.1 一种操作系统的层次关系

下面将介绍有关操作系统的基本概念，图17.1给出了一种可能的操作系统的层次关系。在后面章节里将分别介绍 $\mu\text{C}/\text{OS-II}$ 、Linux 和 VxWorks 等嵌入式操作系统的基础知识及它们在嵌入式系统上移植的例子。

17.2.2 嵌入式实时操作系统的相关概念

1. 操作系统的内核是由中断驱动的

在操作系统中，进程管理、内存管理及计算机硬件抽象层(HAL)中的程序模块组成了操作系统的核心部分，这

个核心部分称为操作系统的内核。由于操作系统内核的重要性，为了防止用户程序错误调用内核模块而使系统崩溃，所以对操作系统的内核必须加以特殊保护，使用户的应用程序不能直接调用内核模块。解决这个问题一个办法就是使用中断的方法，因为高级语言一般不支持中断，从而使得基于高级语言的应用程序不能直接调用内核模块。另外，进程切换的实现也要利用中断来实现，因为进程切换的实质是处理器执行流程的转移，而这种转移不应由正在执行的进程决定。

图17.2给出了处理器的中断处理流程，可以看到执行流程在整个中断处理过程中发生了两次跳转，一次是发生在响应中断并保护断点之后，另一次是发生在中断服务程序返回时。尤其需要注意的是第二次跳转时，根据返回指令 RTE 或 IRTE 之前保护的断点是否进行修改过，这次跳转的目的地是截然不同的：如果没有进行修改，则返回原来被中断了的程序；如果进行了修改，则会跳转到另一个程序中。操作系统正是利用了处理器中断过程的这两次跳转的特点，实现了操作系统所需要的系统调用和进程切换这两个功能。

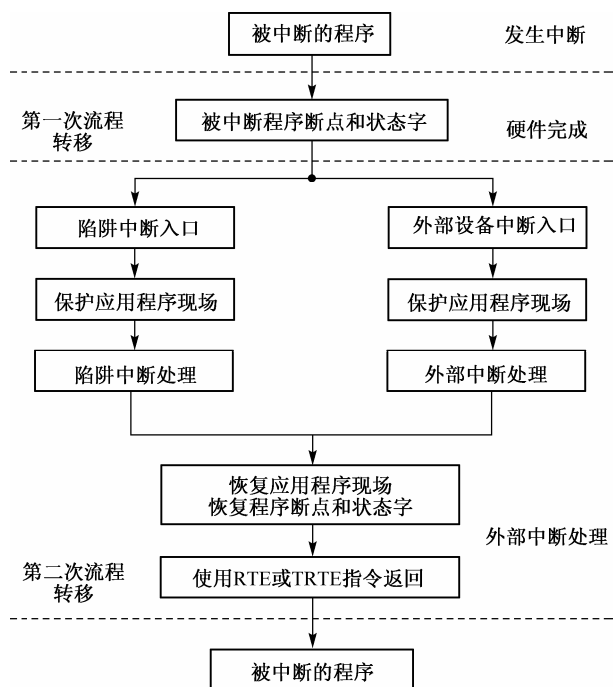


图 17.2 处理器的中断处理流程

2. 进程和线程的基本概念

如何合理地分配计算机的硬件和软件资源，使应用程序能高效、安全地运行，一直是嵌入式研究所致力解决的问题，为此引入了进程这个概念。在进程的基础上，为进一步提高系统的并发性能，又引入了更小的运行单位——线程。线程也称为任务，是一个简单的程序，该程序可以认为 CPU 完全只属于自己。嵌入式操作系统由于应用环境的限制，大多数是以线程(任务)为单位进行资源的分配，关于这一点，读者在阅读有关计算机操作系统和嵌入式操作系统的书籍时要注意区分。

典型地，每个任务都是一个无限循环，它总是处于以下 5 种状态之一：休眠态、就绪态、运行态、挂起态和被中断态，如图 17.3 所示。休眠态下该任务驻留在内存中，但并不被多任务内核所调度。就绪态意味着该任务已经准备好，可以运行了，但是由于更高优先级的任务正在运行，还暂时不能运行。运行态是指该任务已经掌握了 CPU 的控制权，正在运行中。挂起态也称为等待状态，指该任务正在等

待某一事件的发生(例如等待某外设的 I/O 操作, 等待某共享资源由暂不能使用状态转入能使用状态, 等待定时脉冲的到来或等待超时信号的到来以结束目前的等待等)。最后, 发生中断时, CPU 提供相应的中断服务, 原来正在运行的任务暂不能运行, 就进入了被中断状态。

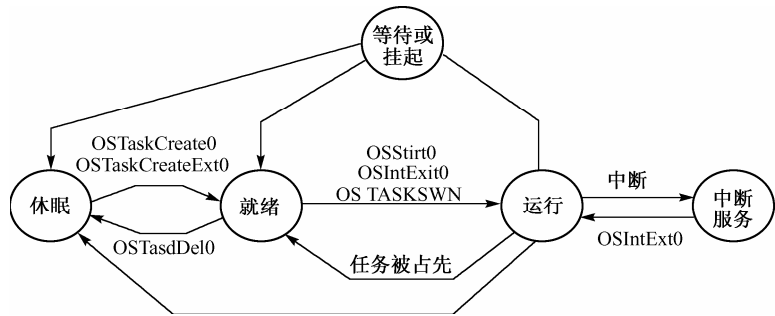


图 17.3 任务状态的转换

(1) 任务切换

当多任务内核决定运行另外的任务时, 它保存正在运行的当前状态(context), 即 CPU 寄存器中的全部内容。这些内容保存在任务的当前状态保存区(task’s context storage area), 也就是任务自己的栈区之中, 如图17.4所示。入栈工作完成之后, 就把下一个将要运行的任务的状态从该任务的栈中装入 CPU 的寄存器, 并开始下一个任务的运行。这个过程就称为任务切换。任务切换过程增加了 CPU 的额外负荷。

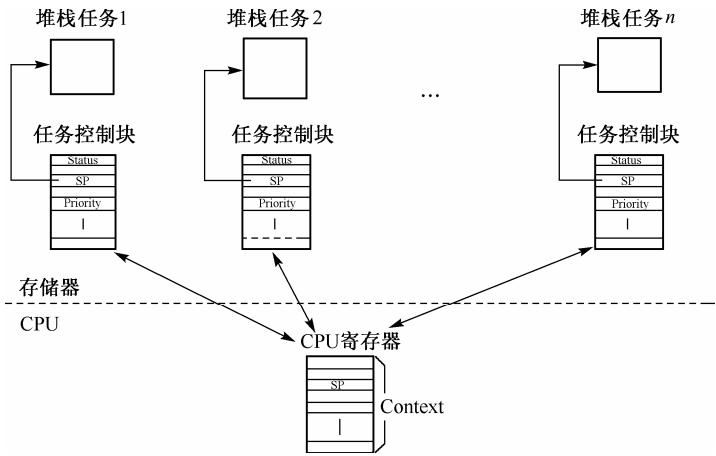


图 17.4 多任务切换

(2) 任务调度

调度(dispatcher)是内核的主要职责之一, 是内核决定该轮到哪个任务运行, 多数实时内核是基于优先级调度算法的。每个任务根据其重要程度的不同被赋予一定的优先级, 基于优先级的调度算法是指 CPU 总是让处于就绪态且优先级最高的任务先运行, 然而究竟何时让高优先级的任务掌握 CPU 的使用权, 要看是非占先式的内核还是占先式的内核。

非占先式(non-preemptive)内核要求每个任务自我放弃 CPU 的使用权。中断服务可以使一个高优先级的任务由挂起态变为就绪态, 但中断服务以后控制权还是回到原来被中断了的那个任务, 直到该任务主动放弃 CPU 的使用权, 那个高优先级的任务才能获得 CPU 的使用权, 如图17.5(a)所示。非占

先式内核的优点是中断响应快,几乎不需要使用信号量保护共享数据,但是最大的缺点是其任务响应时间不确定,不知道什么时候最高优先级的任务才能得到 CPU 的控制权,完全取决于应用程序什么时候释放 CPU。商业软件几乎没有非占先式内核。

绝大多数商业实时内核都是占先式(preemptive)内核,最高优先级的任务一旦就绪,就总能得到 CPU 的控制权。当一个运行着的任务使一个比它优先级更高的任务进入了就绪态,当前任务的 CPU 占用就被剥夺了,或者说当前任务被挂起了,那个高优先级的任务立刻得到了 CPU 的控制权。如果是中断服务子程序使一个高优先级的任务进入了就绪态,中断完成时,被中断了的任务将被挂起,优先级高的任务开始运行,如图 17.5 (b) 所示。

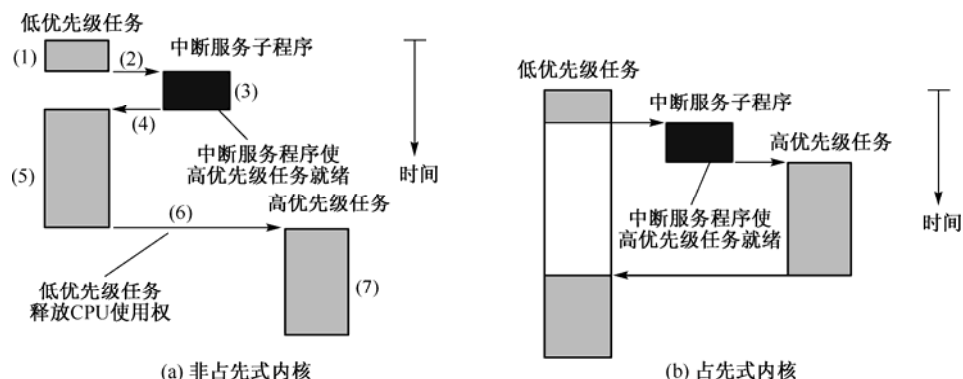


图 17.5 内核的任务调度

(3) 进程(任务)间的同步与通信

为实现各进程间的合作与无冲突的运行,在进程之间必须建立一些制约关系,这些制约关系可以分为直接制约关系和间接制约关系。直接制约关系源于进程之间的合作,间接制约关系源于对资源的共享。操作系统必须解决这两个问题:一是相关进程在执行上要有先后的次序,一个进程要等其伙伴发来通知,或建立了某个条件后才能继续执行,否则只能等待;二是各进程间应该具有一种互斥关系,即对于某个共享资源,如果一个进程正在使用,则其他进程只能等待该资源被释放后才能使用。

进程之间的这种制约性的合作运行机制称为进程间的同步,系统中进程的同步是依靠进程与进程之间互相发送消息来保证的。在操作系统中,是使用信号量、邮箱(消息邮箱)和消息队列这些被称为事件的中间环节来实现进程之间通信的,如图 17.6 所示。



图 17.6 两个进程使用事件进行通信示意图

3. 代码临界区和函数的重入性

代码的临界区也是临界资源,指处理时不可分割的代码,一旦这部分代码开始执行,则不允许任何中断打入。为了确保临界区代码的执行,在执行临界区之前要关闭中断,而临界区代码执行完毕之后要立即开启中断。

可重入型函数可以被一个以上的任务调用,而不必担心数据的破坏。可重入型函数任何时候都可以被中断一段时间以后又再次被调用,而相应数据不会丢失。可重入型函数或者只使用局部变量,即变量保存在 CPU 的寄存器或堆栈中,或者使用予以保护的全局变量。使用以下技术可以将函数编写为可重入型的:使用局部变量,调用函数前关闭中断;用信号量禁止该函数在使用过程中被再次调用。

4. 宏内核和微内核

操作系统的内核在设计上可以分为宏内核和微内核两种。宏内核的内部可以被分为若干模块，但是在运行时，它是一个独立的二进制映像，模块间的通信不是通过消息传递而是通过调用其他模块中的函数来实现的。微内核中，用以完成系统调用功能的程序模块通常只是进行简短的处理，而把其余工作通过消息传递给内核之外的进程来处理。微内核的基本思想就是要保持操作系统的内核尽可能小，这样就便于在不同硬件系统平台上进行移植。微内核的另一个优点是不需要的模块可以不加载到内存中，从而有效地利用内存。

5. 其他

典型的操作系统还包括存储器管理、I/O 设备管理和文件管理等模块，但是不同的嵌入式操作对这些部分的处理不尽相同，操作系统在不同的硬件平台上进行移植时是需要考虑这些问题的。受篇幅限制，这里就不给出有关内容的介绍了，感兴趣读者可以参考有关文献。

17.3 常见嵌入式实时操作系统介绍

本节将介绍一些常见的嵌入式实时操作系统，包括它们的起源、发展和特点，让读者对各种常见的嵌入式实时操作系统有一个快速的了解，从而能为自己的项目开发选择合适的操作系统。这里都是一些关于各种操作系统的一般性介绍，对于具体的各种操作系统的移植过程，读者可以参考相应的书籍和操作系统手册。为了让读者对移植的过程有概要性的理解，将在17.4节中给出一个 $\mu\text{C}/\text{OS-II}$ 的移植实例，所以将 $\mu\text{C}/\text{OS-II}$ 实时操作系统的介绍一并放到17.4节。

1. μCLinux 简介

为使讨论问题有针对性，在讨论 μCLinux 和 RTLlinux 系统时，首先对标准Linux系统直接在嵌入式应用中存在的局限性做出分析。

(1) 大多数嵌入式处理器不采用虚拟内存技术

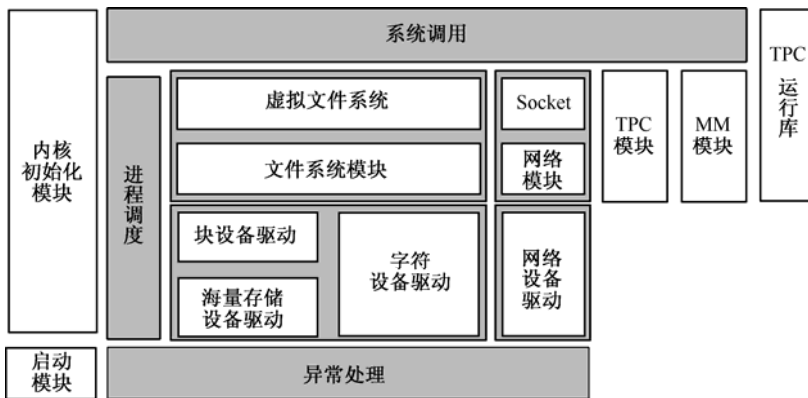
标准Linux系统的内存管理是基于虚拟内存技术的，虚拟内存是一个很费时的操作且没有时间上的保证，而且大多数的嵌入式处理器也都不支持虚拟内存技术，所以标准Linux系统直接移植到嵌入式处理器上要对操作系统进行大幅度的修改。

(2) 标准Linux系统不支持硬实时系统

标准Linux在进程调度和相关算法中是以进程的平均占用处理器时间为目标进行设计的，且其内核是非占先式的，因此不支持硬实时系统。

μCLinux 是标准Linux的一个变种，是一个源码开放的GNU产品，面向没有MMU(Memory Management Unit)的硬件平台。这类的CPU较著名的有Dragon Ball、ARM7及Intel i960，它们除了都没有MMU外，低价、低速、低功耗为其共同的特色，这样的CPU通常用在中高阶的嵌入式系统中(如gateway、PDA等)。 μCLinux 与标准Linux系统主要的区别在于两者的内存管理机制和进程调度管理机制，同时为了适应嵌入式应用的需求，它采用了romfs文件系统，并对Linux上的C语言库glibc做了简化。 μCLinux 的结构示意图如图17.7所示。

内核初始化模块主要负责异常、中断、内存分页、驱动程序、调度器的初始化工作，并启动init进程进行多任务环境。内核启动模块负责启动内核，是一个与硬件高度相关的模块，其主要任务是Chip Selector初始化和系统堆栈初始化，自Flash存储器解压Linux映像并将其装入内存，最后把系统控制权交给内核。

图 17.7 μ CLinux 的结构示意图

μ CLinux 的内核有两种可选的运行方式：可以在 Flash 上直接运行，也可以加载到内存中运行。Flash 运行方式把内核的可执行映像烧写到 Flash 上，系统启动时从 Flash 的某个地址开始逐句执行，这种方法实际上是很多嵌入式系统采用的方法。内核加载方式把内核的压缩文件存放在 Flash 上，系统启动时读取压缩文件并在内存里解压，然后开始执行，这种方式相对复杂一些，但是运行速度可能更快（RAM 的存取速率要比 Flash 高）。同时，这也是标准 Linux 系统采用的启动方式。

μ CLinux 系统采用 romfs 文件系统，相对于 ext2 文件系统要求更少的空间。空间的节省首先来自内核支持 romfs 文件系统需要更少的代码，其次 romfs 文件系统相对简单，建立文件系统超级块（superblock）需要更少的存储空间。romfs 文件系统不支持动态擦写保存，对于需要动态保存的数据采用虚拟 RAM 盘的方法进行处理（RAM 盘采用 ext2 文件系统）。

μ CLinux 是针对没有 MMU 的处理器而设计的，即 μ CLinux 不能使用处理器的虚拟内存管理技术。 μ CLinux 采用存储器的分页管理方法，系统在启动时把实际存储器进行分页，在加载应用程序时程序分页加载。但是由于没有 MMU 管理，所以实际上 μ CLinux 采用实存储器管理策略（Real Memory Management）。 μ CLinux 系统对于内存的访问是直接的，所有程序中访问的地址都是实际的物理地址，操作系统对内存空间没有保护，各个进程实际上共享一个运行空间。一个进程在执行前，系统必须为进程分配足够的连续地址空间，然后全部载入主存储器的连续空间中。由于应用程序加载时必须分配连续的地址空间，而不同硬件平台下可一次成块（连续地址）分配的内存大小限制是不同的，所以开发人员在开发应用程序时必须考虑内存的分配情况并关注应用程序需要运行空间的大小。另外，由于采用实存储器管理策略，用户程序同内核及其他用户程序在一个地址空间，程序开发时要保证不侵犯其他程序的地址空间，以使得程序不至于破坏系统的正常工作，或导致其他应用程序的运行异常。

μ CLinux 的多进程管理通过 vfork 来实现。这意味着 μ CLinux 系统 fork 调用完程后，要么子进程代替父进程执行（此时父进程已经 sleep）直到子进程调用 exit 退出，要么调用 exec 执行一个新的进程，这个时候将产生可执行文件的加载，即使这个进程只是父进程的副本，这个过程也不能避免。当子进程执行 exit 或 exec 后，子进程使用 wakeup 把父进程唤醒，父进程继续往下执行。

要使用 μ CLinux 开发嵌入式系统，必须至少具备下列套件：

- μ CLinux 原始码；
- 开发工具（编译器、链接器、…）；
- 应用程序函数库（ANSI-C、数学函数库、…）；
- RAM-Disk image source（这个可以自行制作）。

这些套件都可以在 μ CLinux 的官方网站（<http://www.ucLinux.org>）上查找到。

2. RTLinux

RTLinux (RealTime Linux) 是由美国 New Mexico Tech 的 Victor Yodaiken 等人开发的基于标准 Linux 的嵌入式操作系统, 在电信、工业自动化和航空航天等实时领域已经有成熟应用。RTLinux 把标准 Linux 内核作为微内核的一个进程看待, 并赋予它最低优先级, 使实时进程可以随时剥夺 Linux 内核的处理器使用权, 用这种巧妙的技术解决了标准 Linux 过长的中断反应时间和任务切换反应时间的缺点, 实现了对实时的支持。它能够创建精确运行的符合 POSIX.1b 标准的实时进程; 并且作为一种遵循 GPL V2 协议的开放软件, 可以在 GPL V2 协议许可范围内自由地、免费地使用、修改和再发布。New Mexico Tech 的 RTLinux 研制者们为该技术申请并获得了专利, 在该技术的影响下, 又出现了一系列基于 Linux 的其他嵌入式实时操作系统, 如 RTAI 和 ADEOS 等。RTLinux 系统结构如图 17.8 所示。

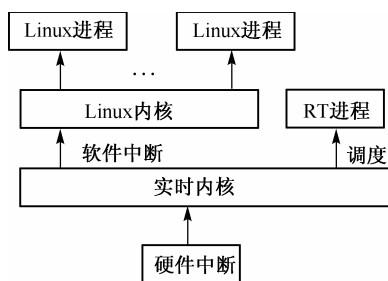


图 17.8 RTLinux 系统结构

RTLinux 有两种中断: 硬中断和软中断。软中断是常规 Linux 内核中断, 它可无限制地使用 Linux 内核调用; 硬中断是安装实时 Linux 的前提。RTLinux 将 Linux 源码中所有的 cli, sti, iret 指令分别用宏 S_CLI, S_STI 和 S_IRET 替换, 引入虚拟层截取所有的硬件中断, 分割 Linux 系统与硬件中断之间的直接联系。当 RTLinux 虚拟层接收到与实时处理有关的硬件中断时, 立即启动执行相应的实时中断服务程序; 而接收到与实时处理无关的中断时, 先保存相应的信息, 等到 RTLinux 内核空闲时通过软中断传递给 Linux 内核去处理, 这样就使得 RTLinux 内核不受各种软、硬件中断的影响, 不会造成时间上的不可预

测性。同时又区别于其他实时处理方案, 它并未对操作系统的内核做结构性的修改, 因此并不会妨碍 Linux 操作系统的进一步发展和变化。

从图 17.8 可以看出, RTLinux 拥有两个内核, 这就意味它有两组单独的 API, 一个用于 Linux 环境, 另一个用于实时环境。此外, 为保证实时进程与非实时 Linux 进程的数据交换, RTLinux 引入了 RT-FIFO 队列。RT-FIFO 被 Linux 视为字符设备, 最多可达 150 个, 最大的 RT-FIFO 数量在系统内核编译时设定。

RTLinux 仿真 Linux 内核中的中断控制器, 这样即使在 CPU 发生中断, 同时 Linux 内核请求被取消的情况下, 关键的实时中断也能够保持激活。研究报告显示, 这种方法在高速的处理器上能够获得低于 10 μ s 的中断反应时间, 其优势在于实时和非实时的线程是被分离的。关键的实时函数会在固定的 RTOS 环境下运行, 从而不受普通 Linux 内核的时间影响。

RTLinux 程序运行于用户空间和内核态两个空间。RTLinux 提供了应用程序接口, 借助这些 API 函数将实时处理部分编写成内核模块, 并装载到 RTLinux 内核中, 运行于 RTLinux 的内核态。非实时部分的应用程序则在 Linux 下的用户空间中执行。

3. VxWorks

VxWorks 操作系统是美国 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统, 凭借良好的持续发展能力、高性能的内核及友好的用户开发环境, 在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中, 如卫星通信、军事演习、弹道制导、飞机导航等。在美国的 F-16、FA-18 战斗机、B-2 隐形轰炸机和爱国者导弹上, 甚至连 1997 年 4 月在火星表面登陆的火星探测器上也使用到了 VxWorks。VxWorks 有以下特点。

- 可靠性。稳定、可靠一直是 VxWorks 的一个突出优点。自从对中国的销售解禁以来, VxWorks 以其良好的可靠性在中国赢得了越来越多的用户。
- 实时性。VxWorks 的实时性做得非常好, 其系统本身的开销很小, 进程调度、进程间通信、中断处理等系统公用程序精练而有效, 它们造成的延迟很短。VxWorks 提供的多任务机制中对任务的控制采用了优先级抢占 (Preemptive Priority Scheduling) 和轮转调度 (Round-Robin Scheduling) 机制, 也充分保证了可靠的实时性, 使同样的硬件配置能满足更强的实时性要求, 为应用的开发留下更大的余地。
- 可裁减性。VxWorks 由一个体积很小的内核及一些可以根据需要进行定制的系统模块组成。VxWorks 内核最小为 8 KB, 即便加上其他必要模块, 所占用的空间也很小, 且不失其实时、多任务的系统特征。由于它的高度灵活性, 用户可以很容易地对这一操作系统进行定制或做适当开发, 来满足自己的实际应用需要。

VxWorks 操作系统由以下部件组成。

- 内核。VxWorks 提供了以下的功能: 多任务调度(采用基于优先级抢占方式, 同时支持同优先级任务间的分时间片调度), 任务间的同步, 进程间通信机制, 中断处理, 定时器和内存管理机制。
- I/O 系统。VxWorks 提供了一个快速灵活的与 ANSI C 兼容的 I/O 系统, 包括 UNIX 标准的 Basic I/O, Buffer I/O 及 POSIX 标准的异步 I/O。VxWorks 包括以下驱动程序: 网络驱动、管道驱动、RAM 盘驱动、SCSI 驱动、键盘驱动、显示驱动、磁盘驱动、并口驱动等。
- 文件系统。VxWorks 支持 4 种文件系统: dosFs, rt11Fs, rawFs 和 tapeFs。支持在一个单独的 VxWorks 系统上同时并存几个不同的文件系统。
- 板级支持包 (Board Support Package, BSP)。板级支持包向 VxWorks 操作系统提供了对各种电路板的硬件功能操作的统一的软件接口, 它是保证 VxWorks 操作系统可移植性的关键, 它包括硬件初始化、中断的产生和处理、硬件时钟和计时器管理、局域和总线内存地址映射、内存分配等。每个板级支持包括一个 ROM 启动 (Boot ROM) 或其他启动机制。
- 网络支持。它提供了对其他 VxWorks 系统和 TCP/IP 网络系统的“透明”访问, 包括与 BSD 套接字兼容的编程接口、远程过程调用 (RPC)、SNMP (可选项)、远程文件访问及 BOOTP 和代理 ARP, DHCP, DNS, OSPF, RIP 等。
- 目标代理 (Target Agent)。目标代理遵循 WBD (Wind Debug) 协议, 允许目标机与主机上的 Tornado 开发工具相连。在目标代理的默认设置中, 目标代理是以 VxWorks 的一个任务 tWdbTask 的形式运行的。Tornado 目标服务器 (Target Server) 向目标代理发送调试请求。调试请求通常决定目标代理对系统中其他任务的控制和处理。
- 实用库。VxWorks 提供了一个实用例程的扩展集, 包括中断处理、看门狗定时器、消息登录、内存分配、字符扫描、线缓冲和循环缓冲管理、链表管理和 ANSI C 标准库。
- 基于目标机的工具。在 Tornado 开发系统中, 开发工具是驻留在主机上的。但是也可以根据需要将基于目标机的 Shell 和装载卸载模块加入 VxWorks。Tornado 开发环境如图 17.9 所示。

4. Windows CE

Windows CE 是 Microsoft 为 PDA Set-Top-Box 等用户化嵌入式系统开发的 32 位多任务多线程操作系统平台, 它具有 Windows 的 GUI、可 ROM 化、整合的电源管理、标准的通信协议及可与其他 Windows 应用软件共享信息等优点, 这些为开发者带来极大的益处。

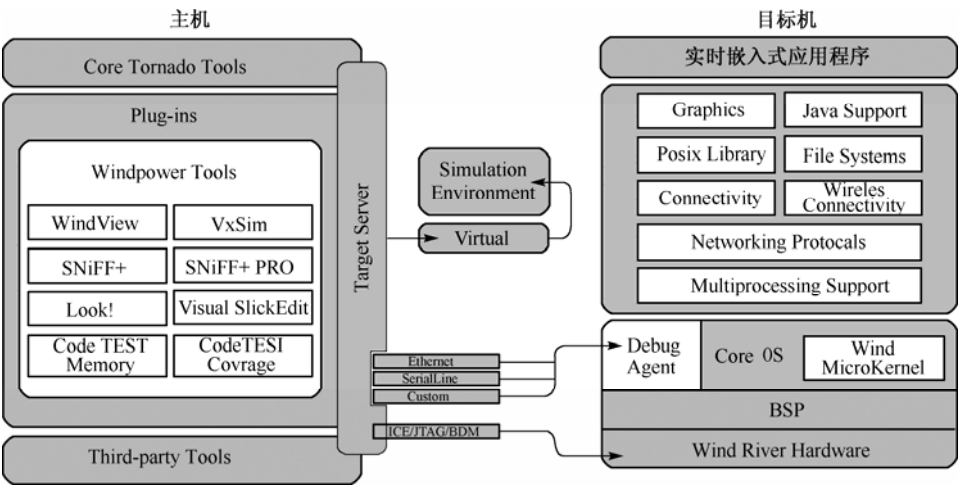


图 17.9 Tornado 开发环境

Windows CE 包括 4 个模块，提供了操作系统最重要的功能：内核、对象存储、制图、开窗口、事件字系统 (GWES) 和通信。Windows CE 还包含其他可选模块，支持诸如管理可安装设备的驱动程序和 COM 的任务。Windows CE 支持各种外围设备，还可以支持 Windows 98 和 Windows NT 普遍使用的 Win32 API 的子集和几种附加的编程接口，包括组件对象模型 (COM)、Microsoft 基本类库 (MFC)、Microsoft ActiveX 控制和 Microsoft 活动模板库 (ATL)。

图17.10给出了基于 Windows CE 的目标平台的结构示意图。一个基于 Windows CE 的嵌入式系统可分为四个层次，从底层到上层分别是硬件层、OEM 层、操作系统层和应用程序层。硬件层是系统的硬件，包括微处理器和各种周边设备。OEM 层是一个硬件抽象层，它提供了硬件和操作系统之间的接口，操作系统要访问具体的硬件就可以通过 OEM 层提供的 API 进行访问，而不必直接与硬件打交道。操作系统层中有 Windows CE 的组件，用户可以根据自己系统的需要进行定制，这样可以减小内存需求，使系统性能达到最佳。应用程序层是用户为特定的嵌入式系统开发的应用程序。

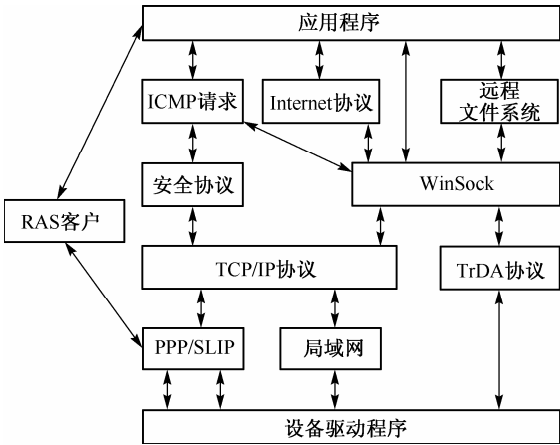


图 17.10 基于 Windows CE 的目标平台的结构示意图

Windows CE 界面与普通 PC 上使用的 Windows 系统很相近，而且微软也把开发工具和现有的 Visual 系列工具集成在一起了，但基于 WinCE 的开发和普通 Windows 开发还是有很大的区别的。因为 Win CE 针对小型移动设备，而这类装置是千差万别的，开发者必须了解目标设备和部署自己程序的方法。

Microsoft Embedded Visual Tools 3.0 为构建移动应用程序提供了一个入门级的集成开发环境, 包括必要的编译器、调试和平台文档。Embedded Visual Tools 3.0 包括 Microsoft Embedded Visual C++ 3.0 (eVC 3.0) 和 Embedded Visual Basic 3.0 (eVB 3.0)。这两种工具都是独立的开发环境, 不需要运行任何其他开发环境。如果你只使用其中的 C++ 语言编程, 可以选择下载单独的开发工具 Embedded Visual C++ 4.0。Visual Studio .NET 2003 将安装 .NET Compact Framework, 这是专门为资源有限的设备设计的。开发人员可以使用新的 C# 语言或者 Visual Basic .NET 语言来开发移动和嵌入式设备。由于 Microsoft 只提供 Windows CE 内核, 它必须通过开发工具来开发可运行在目标平台上的运行系统, 因此几乎所有外部的硬件驱动需要用户根据所选择的芯片来写驱动程序, 这又给开发者带来很大的麻烦。

5. pSOS

pSOS 是集成系统有限公司研发的产品, 该公司成立于 1980 年, 产品在成立后不久推出, 是世界上最早的实时系统之一, 也是最早进入中国市场的实时操作系统。该公司在 2000 年 2 月 16 日与 WindRiver Systems 公司合并, 被并购的同时包括其旗下的 DIAB-SDS, Doctor Design 和 TakeFive Software, 两公司合并之后已成为小规模公司林立的嵌入式软件业界里面的巨无霸。从 VxWorks 5.5 开始, 已将 pSOS 的主要特点融入 VxWorks 中。pSOS 的主要缺点在于其上下文切换时间长, 实时性不强, 采用的集成开发环境 Sniff+ 与产品兼容性不好, 部分关键功能无法使用。

从结构上看, pSOS 是一个由标准软组件组成的可裁剪的实时操作系统, 它包含单处理器支持模块 (pSOS+)、多处理器支持模块 (pSOS+m)、文件管理器模块 (pHILE)、TCP/IP 通信包 (pNA)、流式通信模块 (OpEN)、图形界面、JAVA 和 HTTP 等。pSOS 的各功能模块完全独立, 开发者可根据应用需求扩展系统功能和存储容量。pRISM+ 是 pSOS 的全集成开发环境, 包含 C/C++ 编辑浏览器、调试器、编译器和图形运行分析工具。

pSOS 推出时间较早, 因此它比较成熟, 目前可以支持很多类型的处理器, 其板级支持包 BSP 也比较全, 能支持各种类型的评估板, 同时支持 ISO9660, MS-DOS, NFS 等多种文件系统, 目前全球大约有 3500 万个设备上使用的是该操作系统。pSOS 曾经在实时操作系统领域销售额占世界第一, 近年来市场占有率有所下降。pSOS 在中国有着广泛的市场, 目前有八十几家企业或研究所在使用 pSOS 实时操作系统, 主要应用领域包括通信、航天、信息家电及工业控制。

6. QNX

QNX 是由加拿大 QNX 软件系统有限公司开发的一个建立在微内核和完全地址空间保护基础之上的实时操作系统, 它具有模块化、裁剪自如和易于扩展的特点。QNX 独特的微内核和消息传递结构为其运行和开发期间提供了丰富的接口。QNX 是唯一可以将实时 POSIX 环境外加一个完全的窗口系统安装在 1 MB 以下的闪储或只读存储器上的操作系统。

QNX 具有完全的可伸缩性, 应用在很宽广的领域中 (从消费电子产品到工业控制系统), 甚至其嵌入式窗口系统 Photon micro GUI 也是基于微内核结构, 因此很容易在任何目标系统上完成一个在尺寸和功能上均衡的图形界面。QNX 操作系统服务通过一组可选的协作进程——每个进程运行在它们自己的 MMU 的保护地址空间, 通过高性能的进程间消息传递通信机制可以访问所需要的服务 (文件系统, 设备 I/O 等)。这些服务的模块化给予了 QNX 高度的可伸缩性。QNX 以消息传递作为所有操作系统服务的基础, 使上下文切换时间很严格。

在传统方式中, 驱动程序在内核模式下运行并且有完全的权限访问系统存储区, 最终驱动程序和其他系统处理中的代码可能覆盖其他危险的区域, 这是传统的操作系统共同的内核问题。在充分利用 X86 芯片上的 MMU 后, QNX 通过其独特的结构可以提供非常高的可靠性。在 QNX 中, 设备驱动程

序运行在用户空间,拥有指定的系统权限,驱动程序不能在其地址空间外面写任何区域,因此内核缺陷非常少。除了将驱动程序运行在自己的地址保护空间外,QNX也支持软件看门狗通知,当程序中错误导致应用崩溃时,这个功能可使系统恢复。

QNX超越了TCP/IP和其他传统的连接标准,可以提供无缝的图形连接到Windows 95, Windows NT和X Windows,这样的做法使开发者可以在熟悉的桌面环境下创建QNX应用。除符合POSIX外,QNX为Win32、微软新CIFS协议和其他一些系统提供了移植工具。

QNX比TCP/IP或其他的传统网络服务提供了更高的网络透明度。实际上,QNX将整个LAN改变为单一逻辑机器。通过使用网络透明的消息传递可以访问所有的系统资源,可以很轻松地增加和减少系统资源和网点,无须分配具体的应用在其他处理器。

17.4 嵌入式实时操作系统 μ C/OS-II及其移植

μ C/OS-II是由Jean J. Labrosse于1992年编写的一个源码开放的嵌入式多任务实时操作系统。早期这个系统称为 μ C/OS,1999年Jean J. Labrosse推出了 μ C/OS-II,并在2000年得到了美国联邦航空管理局关于商用飞机的、符合RTCA DO-178B标准的认证,证明了 μ C/OS-II具有足够的稳定性和安全性。 μ C/OS-II是用C语言和汇编语言来编写的,其中绝大部分功能是用C语言编写的,只有极少部分与处理器密切相关的代码是用汇编语言编写的,用户在移植时也只是需要修改这些与处理器相关部分的代码,它可以很方便地移植到8位、16位和32位嵌入式处理器上。 μ C/OS-II构思巧妙,结构精练,具备了实时操作系统的全部功能,同时又是源码开放的,非常适合初次接触嵌入式实时操作系统的学生、嵌入式系统开发人员和爱好者学习,并且已经在很多处理器上得到了移植。 μ C/OS-II目前除了内核外,还提供商业化的文件系统 μ C/FS、图形系统 μ C/GUI及任务调试工具 μ C KA和 μ C View。读者如果希望了解关于 μ C/OS-II的更多信息,可以访问Jean J. Labrosse先生的官方网站<http://www.uCOS-II.com>。

17.4.1 μ C/OS-II的基本组成

1. μ C/OS-II的体系结构

μ C/OS-II的体系结构如图17.11所示。 μ C/OS-II作为一个微内核,它只对处理器和系统的时钟进行了抽象和封装,没有对其他硬件进行抽象,因而很容易移植到其他嵌入式平台上。 μ C/OS-II是基于优先级的可剥夺型内核,系统中的所有任务都有一个唯一的优先级别,所以适合应用在实时性较强的场合。 μ C/OS-II区分用户空间和系统空间,所以它适合应用在比较简单的处理器上。

μ C/OS-II的文件结构简单清晰,可以分为三类:(1)是与应用程序相关的文件,如includes.h,os_config.h;(2)是与计算机硬件相关的文件,如os_cpu.h,os_cpu_a.asm,os_cpu_c.c;(3)是系统内核的各种服务文件,如os_core.c,os_flag.c,os_mbox.c,os_mem.c,os_mutex.c,os_q.c,os_sem.c,os_task.c,os_time.c,ucos_ii.c,ucos_ii.h。

μ C/OS-II中习惯把线程称为任务或实时任务, μ C/OS-II中每个任务都有一个唯一的优先级,用来表示该任务在抢夺处理器时所具有的优先权力。 μ C/OS-II共有64个任务优先级别,数字0表示的任务优先级最高,最低优先级OS_LOWEST_PRIO可以由用户自己定义(但要小于64),则此时系统中可供使用的优先级别一共有OS_LOWEST_PRIO+1个,系统总是把最低优先级别OS_LOWEST_PRIO赋给由系统创建的一个任务——空闲任务。下面将依次简单介绍 μ C/OS-II系统中的任务管理、任务调度、时间管理、任务之间的通信和同步及内存管理。

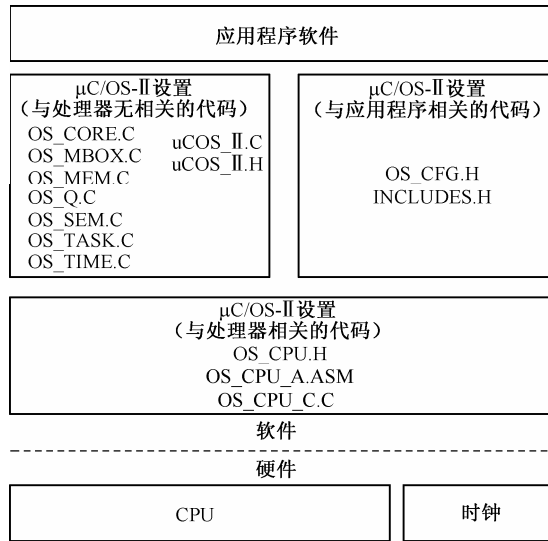


图 17.11 μC/OS-II 的体系结构

2. 任务管理

任务看起来与任何 C 函数一样，具有一个返回类型和一个参数，只是它从不返回。任务的返回类型必须被定义成 void 型。任务必须是下列两种形式之一。

```

void YourTask (void *pdata)
{
    for( ; ;){
        /* 用户代码 */
        OSMboxPend();
        OSQPend();
        ...
    }
}
或
void YourTask (void *pdata)
{
    /* 用户代码 */
    OSTaskDel(OS_PRIO_SELF);
}

```

- 建立任务：想让 μC/OS-II 管理用户的任务，用户必须先建立任务。用户可以通过传递任务地址和其他参数到以下两个函数之一来建立任务：OSTaskCreate() 或 OSTaskCreateExt()。任务可以在多任务调度开始前建立，也可以在其他任务的执行过程中被建立。在多任务调度开始 [即调用 OSStart()] 前，用户必须至少建立一个任务。任务不能由中断服务程序 (ISR) 来建立。
- 删除任务：删除任务是指任务将返回并处于休眠状态，并不是说任务的代码被删除了，只是任务的代码不再被 μC/OS-II 调用。通过调用 OSTaskDel() 就可以完成删除任务的操作。OSTaskDel() 一开始应确保用户所要删除的任务并非空闲任务，因为删除空闲任务是不允许的。OSTaskDel() 还应确保用户不是在 ISR 中去试图删除一个任务，因为这也是不被允许的。调用此函数的任务可以通过指定 OS_PRIO_SELF 参数来删除自己；OSTaskDel() 会保证被删除的任务是确实存在的。

- 请求删除任务：有时候，如果任务 A 拥有内存缓冲区或信号量之类的资源，而任务 B 想删除该任务，这些资源就可能由于没被释放而丢失。在这种情况下，用户可以想办法让拥有这些资源的任务在使用完资源后，先释放资源，再删除自己。用户可以通过 `OSTaskDelReq()` 函数来完成该功能。发出删除任务请求的任务（任务 B）和要删除的任务（任务 A）都需要调用 `OSTaskDelReq()` 函数。用户的应用程序需要决定在什么样的情况下删除任务。
- 改变任务优先级：`μC/OS-II` 允许用户动态地改变任务的优先级，用户建立任务的时候会分配给任务一个优先级，在程序运行期间，用户可以通过调用 `OSTaskChangePrio()` 来改变任务的优先级。
- 挂起任务：任务挂起是一个附加功能，任务在被挂起的同时也在等待延时的期满，而继续等待的延时期满，任务转入就绪状态。挂起任务可通过调用 `OSTaskSuspend()` 函数来完成，被挂起的任务只能通过调用 `OSTaskResume()` 函数来恢复。任务可以挂起自己或者其他任务。
- 获得有关任务的信息：用户的应用程序可以通过调用 `OSTaskQuery()` 来获得自身或其他任务的信息。实际上，`OSTaskQuery()` 获得的是对应任务的任务控制块(OS_TCB)中内容的副本。用户能访问的 OS_TCB 的数据域的多少决定于用户的应用程序的配置(参看 OS_CFGH)。

3. 任务调度

`μC/OS-II` 总是运行进入就绪态任务中优先级最高的那一个，这个工作是由任务调度器(scheduler)完成的。`μC/OS-II` 包含两种调度器，任务级的调度是由函数 `OSSched()` 完成的，中断级的调度是由另一个函数 `OSIntExt()` 完成的。`μC/OS-II` 任务调度所花的时间是常数，与应用程序中建立的任务数无关。

`OSSched()` 的所有代码都属临界段代码。在寻找进入就绪态的优先级最高的任务过程中，为防止中断服务子程序把一个或几个任务的就绪位置位，中断是被关掉的。`μC/OS-II` 中的 `OSSched()` 用 C 语言写的，是为增加可读性、可移植性和将汇编语言代码最少化。但为了缩短调度的时间，用户可以自己用相应的汇编语言改写。

`μC/OS-II` 中，中断服务子程序的示意码清单如下所示。

用户中断服务子程序名：

```
保存全部 CPU 寄存器；  
调用 OSIntEnter 或 OSIntNesting 直接加 1；  
执行用户代码做中断服务；  
调用 OSIntExit()；  
恢复所有 CPU 寄存器；  
执行中断返回指令；
```

调用脱离中断函数 `OSIntExit()` 标志着中断服务子程序的终结，`OSIntExit()` 将中断嵌套层数计数器减 1。当嵌套计数器减到零时，所有中断包括嵌套的中断就都完成了，此时 `μC/OS-II` 要判定有没有优先级较高的任务被中断服务子程序(或任一嵌套的中断)唤醒了。

4. 时间管理

`μC/OS-II` (其他内核也一样) 要求用户提供定时中断来实现延时与超时控制等功能。这个定时中断称为时钟节拍，它应该每秒发生 10 至 100 次。时钟节拍的频率是由用户的应用程序决定的。时钟节拍的频率越高，系统的负荷就越重。

- 任务延时函数：调用 `OSTimeDly()` 会使 `μC/OS-II` 进行任务调度，将当前任务挂起，开始执行下一个优先级最高的就绪态任务。任务调用 `OSTimeDly()` 后，一旦规定的时间期满或者有其

他的任务通过调用 `OSTimeDlyResume()` 取消了延时，它就会马上进入就绪状态。注意，只有当该任务在所有就绪任务中具有最高的优先级时，它才会立即运行。

- 按时分秒延时函数：`OSTimeDly()` 虽然是一个非常有用的函数，但用户的应用程序需要知道延时时间对应的时钟节拍的数目。用户可以使用定义全局常数 `OS_TICKS_PER_SEC` (参看 `OS_CFG.H`) 的方法将时间转换成时钟段。与 `OSTimeDly()` 一样，调用 `OSTimeDlyHMSM()` 函数也会使 $\mu\text{C}/\text{OS-II}$ 进行一次任务调度，并且执行下一个优先级最高的就绪态任务。任务调用 `OSTimeDlyHMSM()` 后，一旦规定的时间期满或者有其他任务通过调用 `OSTimeDly-Resume()` 取消了延时，它就会马上处于就绪态。同样，只有当该任务在所有就绪态任务中具有最高的优先级时，它才会立即运行。
- 系统时间：无论时钟节拍何时发生， $\mu\text{C}/\text{OS-II}$ 都会将一个 32 位的计数器加 1。这个计数器在用户调用 `OSStart()` 初始化多任务和 4 294 967 295 个节拍执行完一遍的时候从 0 开始计数。在时钟节拍的频率等于 100 Hz 的时候，这个 32 位的计数器每隔 497 天就重新开始计数。

5. 任务之间的通信

$\mu\text{C}/\text{OS-II}$ 中，任务间的共享数据和通信可以使用以下方法：临界区、调度器上锁或开锁、信号量、邮箱和消息队列。

- 临界区： $\mu\text{C}/\text{OS-II}$ 为了处理临界段代码需要关中断，处理完毕后再开中断，这使得 $\mu\text{C}/\text{OS-II}$ 能够避免同时有其他任务或中断服务进入临界段代码。 $\mu\text{C}/\text{OS-II}$ 定义两个宏来关中断和开中断，这两个宏调用分别是：`OS_ENTER_CRITICAL()` 和 `OS_EXIT_CRITICAL()`。因为这两个宏的定义取决于所用的微处理器，故在文件 `OS_CPU.H` 中可以找到相应宏定义，每种微处理器都有自己的 `OS_CPU.H` 文件。
- 调度器上锁或开锁：给调度器上锁函数 `OSSchedlock()` 用于禁止任务调度，直到任务完成后调用给调度器开锁函数 `OSSchedUnlock()` 为止。调用 `OSSchedlock()` 的任务保持对 CPU 的控制权，即使有个优先级更高的任务进入了就绪态。然而，此时中断是可以被识别的，中断服务也能得到（假设中断是开着的）。`OSSchedlock()` 和 `OSSchedUnlock()` 必须成对使用。
- 信号量： $\mu\text{C}/\text{OS-II}$ 中的信号量由两部分组成：一个是信号量的计数值，它是一个 16 位的无符号整数（0 到 65 535 之间）；另一个是由等待该信号量的任务组成的等待任务表。用户要在 `OS_CFG.H` 中将 `OS_SEM_EN` 开关量常数置成 1，这样 $\mu\text{C}/\text{OS-II}$ 才能支持信号量。 $\mu\text{C}/\text{OS-II}$ 提供了 5 个对信号量进行操作的函数，它们是 `OSSemCreate()`，`OSSemPend()`，`OSSemPost()`，`OSSemAccept()` 和 `OSSemQuery()` 函数。图 17.12 说明了任务、中断服务子程序和信号量之间的关系。图中用钥匙或者旗帜的符号来表示信号量。如果信号量用于对共享资源的访问，那么信号量就用钥匙符号，符号旁边的数字 N 代表可用资源数。对于二值信号量，该值就是 1。如果信号量用于表示某事件的发生，那么就用旗帜符号，这时数字 N 代表事件已经发生的次数。从图中可以看出，`OSSemPost()` 函数可以由任务或者中断服务子程序调用，而 `OSSemPend()` 和 `OSSemQuery()` 函数只能由任务程序调用。
- 邮箱：邮箱是 $\mu\text{C}/\text{OS-II}$ 中另一种通信机制，它可以使一个任务或者中断服务子程序向另一个任务发送一个指针型的变量。该指针指向一个包含了特定“消息”的数据结构。为了在 $\mu\text{C}/\text{OS-II}$ 中使用邮箱，必须将 `OS_CFG.H` 中的 `OS_MBOX_EN` 常数置为 1。 $\mu\text{C}/\text{OS-II}$ 提供了 5 种对邮箱的操作：`OSMboxCreate()`，`OSMboxPend()`，`OSMboxPost()`，`OSMboxAccept()` 和 `OSMboxQuery()` 函数。图 17.13 描述了任务、中断服务子程序和邮箱之间的关系，这里用符号“T”

表示邮箱。邮箱包含的内容是一个指向一条消息的指针。一个邮箱只能包含一个这样的指针(邮箱为满时)或者一个指向 NULL 的指针(邮箱为空时)。从图 17.13 中可以看出,任务或者中断服务子程序可以调用函数 `OSMboxPost()`,但是只有任务才可以调用函数 `OSMboxPend()` 和 `OSMboxQuery()`。

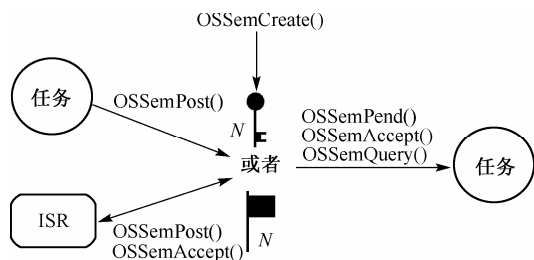


图 17.12 任务、中断服务子程序和信号量之间的关系

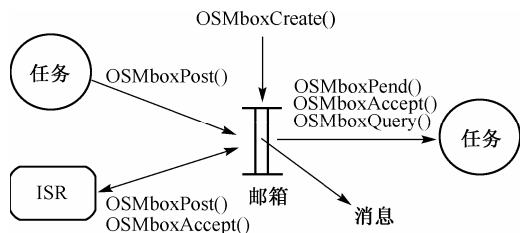


图 17.13 任务、中断服务子程序和邮箱之间的关系

- 消息队列：消息队列是 $\mu\text{C}/\text{OS-II}$ 中另一种通信机制，它可以使一个任务或者中断服务子程序向另一个任务发送以指针方式定义的变量。因具体的应用有所不同，每个指针指向的数据结构变量也有所不同。为了使用 $\mu\text{C}/\text{OS-II}$ 的消息队列功能，需要在 `OS_CFG.H` 文件中将 `OS_Q_EN` 常数设置为 1，并且通过常数 `OS_MAX_QS` 来决定 $\mu\text{C}/\text{OS-II}$ 支持的最多消息队列数。 $\mu\text{C}/\text{OS-II}$ 提供了 7 个对消息队列进行操作的函数：`OSQCreate()`、`OSQPend()`、`OSQPost()`、`OSQPostFront()`、`OSQAccept()`、`OSQFlush()` 和 `OSQQuery()`。图 17.14 是任务、中断服务子程序和消息队列之间的关系。其中，消息队列的符号很像多个邮箱。实际上，可以将消息队列看成是多个邮箱组成的数组，只是它们共用一个等待任务列表。每个指针所指向的数据结构是由具体的应用程序决定的。 N 代表了消息队列中的总单元数。当调用 `OSQPend()` 或者 `OSQAccept()` 之前，调用 N 次 `OSQPost()` 或者 `OSQPostFront()` 就会把消息队列填满。从图 17.14 中可以看出，一个任务或者中断服务子程序可以调用 `OSQPost()`、`OSQPostFront()`、`OSQFlush()` 或者 `OSQAccept()`，但是只有任务才可以调用 `OSQPend()` 和 `OSQQuery()` 函数。

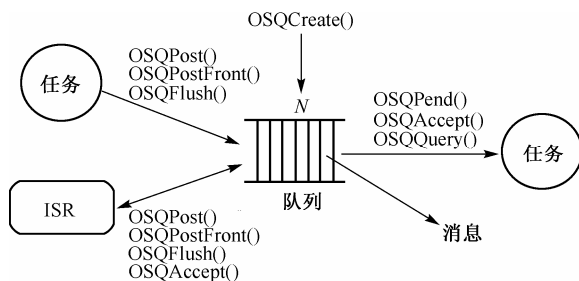


图 17.14 任务、中断服务子程序和消息队列之间的关系

6. 内存管理

在 $\mu\text{C}/\text{OS-II}$ 中，操作系统把连续的大块内存按分区来管理，每个分区中包含有整数个大小相同的内存块。利用这种机制， $\mu\text{C}/\text{OS-II}$ 对 `malloc()` 和 `free()` 函数进行了改进，使得它们可以分配和释放固定大小的内存块，这样 `malloc()` 和 `free()` 函数的执行时间也是固定的。

- 分配一个内存块：应用程序可以调用 `OSMemGet()` 函数从已经建立的内存分区中申请一个内存块，该函数的唯一参数是指向特定内存分区的指针，该指针在建立内存分区时，由

OSMemCreate() 函数返回。

- 释放一个内存块：当用户应用程序不再使用一个内存块时，必须及时地把它释放并放回到相应的内存分区中，这个操作由 OSMemPut() 函数完成。值得注意的是，OSMemPut() 函数并不知道一个内存块是属于哪个内存分区的。
- 使用内存分区：图 17.15 演示了如何使用 $\mu\text{C}/\text{OS-II}$ 中的动态分配内存功能及进行消息传递的例子。第一个任务读取并检查模拟输入量的值，如果其超过了一定的阈值，就向第二个任务发送一个消息。错误处理程序是该例子的中心。任何任务和中断服务子程序都可以向该任务发送出错消息。

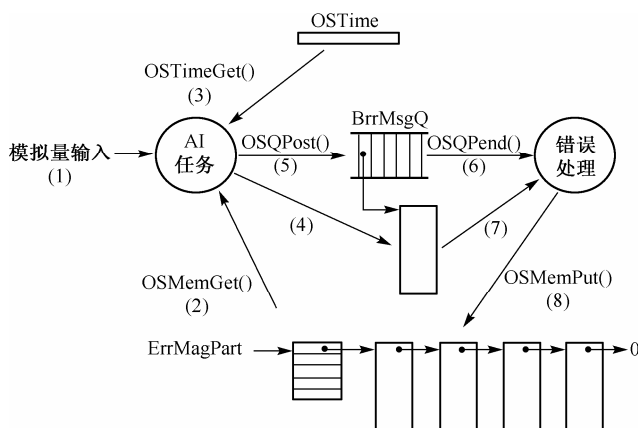


图 17.15 使用 $\mu\text{C}/\text{OS-II}$ 中的动态内存分配及进行消息传递的例子

17.4.2 $\mu\text{C}/\text{OS-II}$ 的移植

由于 $\mu\text{C}/\text{OS-II}$ 在设计时就已经充分考虑了可移植性，所以 $\mu\text{C}/\text{OS-II}$ 的移植相对来说是比较容易的。本节的内容主要是帮助读者更好理解移植过程，有关移植的具体的例子可以在 Jean L. Labrosse 的官方网站上找到。另外，也有很多的相关文献介绍了关于移植的例子，读者在实际开发嵌入式系统时应当借鉴已有的成功实例。

要使 $\mu\text{C}/\text{OS-II}$ 正常运行，处理器必须满足以下要求：

- (1) 处理器的 C 编译器能产生可重入代码；
- (2) 用 C 语言就可以打开和关闭中断；
- (3) 处理器支持中断，并且能产生定时中断（通常在 10~100 Hz 之间）；
- (4) 处理器支持能够容纳一定量数据（可能是几千字节）的硬件堆栈；
- (5) 处理器有将堆栈指针和其他 CPU 寄存器读出和存储到堆栈或内存中的指令。

移植 $\mu\text{C}/\text{OS-II}$ 需要一个 C 编译器，并且是针对用户用的 CPU 的。因为 $\mu\text{C}/\text{OS-II}$ 是一个可剥夺型（占先式）内核，用户只有通过 C 编译器来产生可重入代码；C 编译器还要支持汇编语言程序。所用的 C 编译器还必须提供一个机制来从 C 中打开和关闭中断。一些编译器允许用户在 C 源代码中插入汇编语言，这就使得插入合适的处理器指令来允许和禁止中断变得非常容易。还有一些编译器实际上包括了语言扩展功能，可以直接从 C 中允许和禁止中断。如果读者理解了处理器和 C 编译器的技术细节，移植 $\mu\text{C}/\text{OS-II}$ 的工作是非常简单的。移植工作包括以下几个内容：

- (1) 用 #define 设置常量的值 (OS_CPU.H)；
- (2) 声明 10 个数据类型 (OS_CPU.H)；
- (3) 用 #define 声明 3 个宏 (OS_CPU.H)；

- (4) 用 C 语言编写 6 个简单的函数(OS_CPU_C.C);
- (5) 编写 4 个汇编语言函数(OS_CPU_A.ASM)。

1. OS_CPU.H的编写

μC/OS-II 不使用 C 语言中的 short, int 和 long 等与编译器相关的数据类型, 而代之以移植性强的整形数据类型, 这样既直观又便于移植, 不过这就使得这部分内容成了必须移植的代码。OS_CPU.H 的程序清单如下所示。

```
#ifndef OS_CPU_GLOBALS
#define OS_CPU_EXT
#else
#define OS_CPU_EXT extern
#endif
/*
*****
*
*           数据类型
*           (与编译器相关)
*****
*/
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;           /* 无符号 8 位整数 */      (1)
typedef signed char    INT8S;           /* 有符号 8 位整数 */
typedef unsigned int    INT16U;         /* 无符号 16 位整数 */
typedef signed int      INT16S;         /* 有符号 16 位整数 */
typedef unsigned long   INT32U;         /* 无符号 32 位整数 */
typedef signed long     INT32S;         /* 有符号 32 位整数 */
typedef float           FP32;           /* 单精度浮点数 */      (2)
typedef double          FP64;           /* 双精度浮点数 */
typedef unsigned int    OS_STK;         /* 堆栈入口宽度为 16 位 */
/*
*****
*
*           与处理器相关的代码
*****
*/
#define OS_ENTER_CRITICAL()    ??? /* 禁止中断 */      (3)
#define OS_EXIT_CRITICAL()     ??? /* 允许中断 */
#define OS_STK_GROWTH          1 /* 定义堆栈的增长方向: 1=向下, 0=向上 */ (4)
#define OS_TASK_SW()           ???
```

μC/OS-II 中的一些关键系统函数用系统模式保护起来, 即用户应用程序不使用特殊方式是不能调用这些函数的。于是, 如何使用应用程序在需要时可以由用户模式进入系统模式来调用 μC/OS-II 系统底层的功能函数, 是移植时必须考虑的, 在 17.4.1 节中提到可以使用中断的方法, 而不同的处理支持的中断是不一样的, 这是移植过程中的一个关键的地方。

2. OS_CPU_C.C的编写

μC/OS-II 的移植实例要求用户编写 6 个简单的 C 函数: OSTaskStkInit(), OSTaskCreateHook(), OSTaskDelHook(), OSTaskSwHook(), OSTaskStatHook() 和 OSTimeTickHook()。唯一必要的函数是 OSTaskStkInit(), 其他 5 个函数必须要声明但没必要包含代码。

图17.16显示了 OSTaskStkInt() 放到正被建立的任务堆栈中的内容, 在这里假定堆栈是从下往上增长的, 但下面的讨论同样适用于从上往下增长的堆栈。

在用户建立任务的时候, 用户会传递任务的地址 pdata 指针、任务的堆栈栈顶和任务的优先级给 OSTaskCreate() 和 OSTaskCreateExt()。为了正确初始化堆栈结构, OSTaskStkInt() 只要求刚才提到的前三个参数和一个附加的选项, 这个选项只能在 OSTaskCreateExt() 中得到。

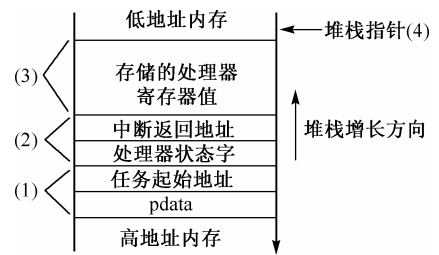


图 17.16 堆栈初始化

3. OS_CPU_A.ASM的编写

μC/OS-II 的移植实例要求用户编写 4 个简单的汇编语言函数: OSStartHighRdy(), OSCtxSw(), OSIntCtxSw() 和 OSTickISR()。

如果用户的编译器支持插入汇编语言代码的话, 用户就可以将所有与处理器相关的代码放到 OS_CPU_C.C 文件中, 而不必再拥有一些分散的汇编语言文件。

(1) OSStartHighRdy()

OSStartHighRdy() 必须调用 OSTaskSwHook(), 因为用户正在恢复最高优先级任务的寄存器。而 OSTaskSwHook() 可以通过检查 OSRunning 来知道是 OSStartHighRdy() 在调用它 (OSRunning 为 FALSE) 还是正常的任务切换在调用它 (OSRunning 为 TRUE)。OSStartHighRdy() 还必须在最高优先级任务恢复之前和调用 OSTaskSwHook() 之后设置 OSRunning 为 TRUE。

```
void OSStartHighRdy (void)
{
    Call user definable OSTaskSwHook();
    Get the stack pointer of the task to resume:
        Stack pointer = OSTCBHighRdy->OSTCBStkPtr;
    OSRunning = TRUE;
    Restore all processor registers from the new task's stack;
    Execute a return from interrupt instruction;
}
```

(2) OSCtxSw()

任务级的切换是通过发软中断命令或依靠处理器执行陷阱指令来完成的。中断服务程序、陷阱或异常处理例程的向量地址必须指向 OSCtxSw()。如果当前任务调用 μC/OS-II 提供的系统服务, 并使得更高优先级任务处于就绪状态, μC/OS-II 就会借助上面提到的向量地址找到 OSCtxSw()。在系统服务调用的最后, μC/OS-II 会调用 OSSched(), 并由此来推断当前任务不再是要运行的最重要的任务。

```
void OSCtxSw(void)
{
    保存处理器寄存器;
    将当前任务的堆栈指针保存到当前任务的 OS_TCB 中:
        OSTCBCur->OSTCBStkPtr = Stack pointer;
    调用用户定义的 OSTaskSwHook();
    OSTCBCur = OSTCBHighRdy;
    OSPrioCur = OSPrioHighRdy;
    得到需要恢复的任务的堆栈指针:
        Stack pointer = OSTCBHighRdy->OSTCBStkPtr;
    将所有处理器寄存器从新任务的堆栈中恢复出来;
```

执行中断返回指令;

}

(3) OSIntExit()

OSIntExit() 通过调用 OSIntCtxSw() 来从 ISR 中执行切换功能。因为 OSIntCtxSw() 是在 ISR 中被调用的, OSIntCtxSw() 必须要清理堆栈, 这样被中断的任务的堆栈结构内容才能满足需要, 所有的处理器寄存器都被正确地保存到了被中断的任务的堆栈之中。

```
void OSIntCtxSw(void)
```

```
{
```

调整堆栈指针来去掉在调用

OSIntExit();

OSIntCtxSw() 过程中压入堆栈的多余内容;

将当前任务堆栈指针保存到当前任务的 OS_TCB 中:

OSTCBCur->OSTCBStkPtr = 堆栈指针;

调用用户定义的 OSTaskSwHook();

OSTCBCur = OSTCBHighRdy;

OSPrioCur = OSPrioHighRdy;

得到需要恢复的任务的堆栈指针:

堆栈指针 = OSTCBHighRdy->OSTCBStkPtr;

将所有处理器寄存器从新任务的堆栈中恢复出来;

执行中断返回指令;

```
}
```

(4) OSTickISR()

$\mu\text{C}/\text{OS-II}$ 要求用户提供一个时钟资源来实现时间的延时和期满功能。时钟节拍应该每秒钟发生 10~100 次。为了完成该任务, 可以使用硬件时钟, 也可以从交流电中获得 50/60 Hz 的时钟频率。但是, 若在 $\mu\text{C}/\text{OS-II}$ 开始执行第一个任务前时钟节拍中断就发生了, 则 $\mu\text{C}/\text{OS-II}$ 的运行状态将不确定, 用户的应用程序也可能会崩溃。

```
void OSTickISR(void)
```

```
{
```

保存处理器寄存器;

调用 OSIntEnter() 或者直接将 OSIntNesting 加 1;

调用 OSTimeTick();

调用 OSIntExit();

恢复处理器寄存器;

执行中断返回指令;

```
}
```

17.5 本章小结

本章首先介绍嵌入式操作系统中的一些基本概念, 说明了嵌入式操作系统与普通操作系统之间的区别与联系; 随后介绍了一些常见的嵌入式操作系统, 给出了各种常见嵌入式操作系统的发展历史及各自的优缺点, 让读者对嵌入式操作系统有一个概略性的认识, 并为嵌入式操作系统的选择提供一些参考; 最后介绍了 $\mu\text{C}/\text{OS-II}$ 及其移植过程, 着重说明了嵌入式操作系统在移植过程中所要注意的问题。通过本章的学习, 读者应对嵌入式操作系统在嵌入式开发过程中的作用有一个基本的了解, 应认识到基于嵌入式操作系统的嵌入式开发是未来电子产品设计的趋势所在。

第 18 章 数码相机伴侣系统的设计与实现

18.1 引言

在信息化的今天，数码相机和各种手持式终端设备已得到了广泛地使用，这些设备大大地丰富和方便了人们的生活。然而用户在户外使用时，时常会出现存储卡容量不足的情况。随身携带笔记本电脑虽能解决问题，但不便于携带。轻巧实用的数码相机伴侣可将存储卡中数据转存到内置硬盘中，并可随时查看、编辑，解决了用户在使用过程中的烦恼。数码相机伴侣系统的设计，是一个典型的复杂电子系统的例子，是本章主要讨论的问题。在本设计中，选择什么样的控制器，如何搭建一个 PDA 软硬件平台，成为系统设计的关键所在。

ARM (Advanced RISC Machines) 既是一个公司的名字，也是一类微处理器的通称。采用 ARM 知识产权 (IP) 的微处理器，即通常所说的 ARM 微处理器。ARM 处理器以其低功耗、低成本和高性能等特点在嵌入式系统中占据了领先地位。目前 ARM 处理器主要型号包括 ARM7, ARM9, ARM9E, ARM10E, SecureCore, StrongARM, Xscale 等。其中 ARM7 低价位、低功耗，属于低端 ARM 处理器，主要用于工业控制、Internet 设备、网络和调制解调器设备、移动电话等。ARM9 高性能、低功耗，主要用于无线设备、仪器仪表、安全系统、机顶盒、高端打印机、数字照相机和数字摄像机的设计中。ARM9E 属于综合处理器，具有增强的 DSP 处理能力，主要用于下一代无线设备、数字消费品、成像设备、工业控制、存储设备和网络设备。ARM10E 采用了新的体系结构，高性能、低功耗，主要用于下一代无线设备、数字消费品、成像设备、工业控制、通信和信息系统。

本章将以 ARM9 为控制核心的便携系统——数码相机伴侣的设计为例，详细分析复杂电子系统设计中利用 ARM 处理器来构建相应系统的过程，讨论基于 ARM 处理器的应用系统的设计思想、设计方法及其实现细节。

18.2 数码相机伴侣系统的设计

1. 设计任务

利用 ARM9 处理器设计并制作一个具有读取存储卡、编辑照片等功能的数码相机伴侣。

2. 设计基本要求

- (1) 存储卡读写；
- (2) 硬盘读写；
- (3) 触摸屏操作；
- (4) 照片浏览。

3. 设计提高部分要求

- (1) USB 相机即插即用；
- (2) 电源管理；

- (3) 音视频播放;
- (4) 游戏功能;
- (5) 网络功能。

18.2.1 系统分析

针对本训练要求, 数码伴侣系统软硬件需要具备以下特点。

(1) USB 主机控制器

USB 相机即插即用功能, 以及外接多合一读卡器都需要 USB 主机控制器。虽然可以通过单独的 USB 主机控制芯片完成, 但若芯片具有内置控制器, 则可减小电路板的体积和提高电路的可靠性。

(2) 高分辨率液晶显示

由于需要进行照片的显示和编辑, 所以对液晶显示功能提出较高要求。

(3) 硬盘驱动功能

系统需要将数据从存储卡转移到具有更高容量的便携式硬盘中, 系统需要能够驱动 IDE 硬盘。

(4) 较低的功耗

由于数码相机伴侣是作为手持设备使用的, 功耗也是一个必须考虑的问题。

(5) 较快的运行速度

本训练要求对多媒体功能方面提出了一定要求, 芯片必须有一定的多媒体处理能力。

(6) 丰富的软件开发资源

本训练功能实现时需要大量软件开发工作, 如果没有足够的可用软件资源, 那么会大大延长开发周期, 需要采用拥有大量软件资源的操作系统(如 Windows CE, Linux 等)。具有开源免费的特点, 是嵌入式开发的首选。

综合考虑成本、功耗及软件资源等因素, 本方案采用三星公司基于 ARM9 的 S3C2410 微处理器。S3C2410 处理器内置 SD 读卡单元, 并可通过 USB 接口扩展多合一读卡器使用; 有丰富的 I/O 资源可扩展 IDE 接口; 芯片主频可达 202 MHz, 支持高分辨率显示输出, 为多媒体功能提供基础; 能够运行标准 Linux 操作系统, 可使用 Linux 下丰富的软件资源。

18.2.2 硬件系统设计

1. S3C2410 处理器简介

S3C2410 处理器基于 ARM920T 内核, 提供了丰富的片上资源, 包含一套比较完整的通用系统的外围设备, 使得基于 S3C2410 处理器的系统体积和功耗都较小。片上集成的功能主要包括以下方面:

- 具有 16 KB 指令缓存、16 KB 数据缓存和 MMU 单元;
- 外部存储控制器 SDRAM 控制和片选逻辑;
- LCD 控制器支持 STN 和 TFT 屏幕;
- 具有外部请求引脚的 4 通道 DMA;
- 具有串口(UART)、红外接口和 SPI 接口;
- IIC 总线控制器的 SD 主机接口和通道 IIS 总线控制器的 Multi-Media 卡接口;
- USB 主机和 USB 设备接口;
- PWM 计时器和内部计时器;
- 看门狗电路;
- 可配置多达 117 个通用 I/O 接口 24 个外部中断源;

- 电源管理包括常规、低速、空闲和断电等模式；
- ADC 和触摸屏接口；
- 具有日历功能的实时时钟 RTC；
- 具有锁相环的片上时钟发生器。

可见，S3C2410 芯片上集成的外围设备已能够完成本训练要求，为调试方便及扩展网络应用的需要，在设计时可添加以太网控制器以提供网络功能。

2. 系统结构

系统结构框图如图 18.1 所示，由于 S3C2410 处理器功能已很完善，所以外围电路很少，主要包括：

- 电源与晶振；
- 需要提供 5 V 电压供液晶屏使用，3.3 V 电压供外部接口使用，处理器核心电压则为 1.8 V，主时钟为 12 MHz 晶振；
- 存储器；
- 包含两片 DRAM HY57V561620 (共 64 MB) 和 Intel 闪存芯片 E28F128J3A (16 MB)。
- 音频电路；
- 使用 UDA1341TS 解码芯片；
- 以太网控制器；
- 为网络扩展及调试方便，电路中加入了以太网接口，使用 100 兆位以太网芯片 DM9000；
- 其他接口。

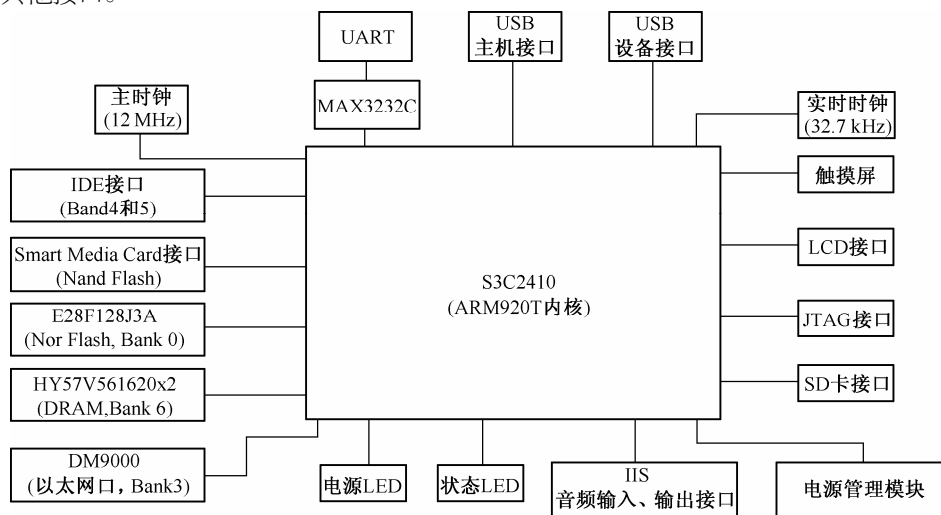


图 18.1 系统结构框图

主要为芯片内置控制器的接口，包括触摸屏电路接口、液晶显示输出接口、USB 主机及设备控制器接口、IDE 总线接口和串口通信接口等。

18.2.3 软件系统设计

1. 软件设计任务分析

通常一个嵌入式 Linux 系统软件开发涉及 4 个层次：

- 引导程序 (Boot Loader)。初始化硬件和加载系统；

- 内核(Kernel)。特定于嵌入式板子的定制内核及内核的启动参数；
- 文件系统(File System)。包括根文件系统和建立于内存设备之上文件系统；
- 用户应用程序。完成应用功能的程序，有时还包含用户图形界面(GUI)。

图18.2是一个同时装有引导程序、内核的启动参数、内核映像和根文件系统映像的固态存储设备的典型嵌入式 Linux 系统空间分配结构图。

通过本训练，读者将了解到引导程序、驱动程序、文件系统及用户应用程序 4 个方面软件开发的基本概念和方法。

引导程序	启动参数	内核	文件系统及用户应用程序

图 18.2 典型嵌入式 Linux 系统空间分配

2. 开发环境

由于目标 ARM 系统与开发用主机(通常为 X86 系统)不能在二进制代码级兼容，所以需要主机上创建一套交叉编译环境。工具链由一套编译、汇编和链接程序和程序库等组件组成。这些组件包括：

- Binutils。Binutil 提供了一系列用来创建、管理和维护二进制目标文件的工具程序，如汇编(as)、链接(ld)、静态库归档(ar)、反汇编(objdump)、elf 结构分析工具(readelf)、无效调试信息和符号的工具(strip)等。通常，Binutils 与 Gcc 是紧密相集成的，没有 Binutils 的支持 Gcc 不能正常工作。
- Gcc。Gcc(GNU Collect Compiler)是一组编译工具的总称，主要完成预处理和编译，另外提供了与编译器紧密相关的运行时库的支持，如 libgcc_s.so, libstdc++.so 等。
- Glibc。Glibc 是 GNU 发布的 C 运行库。Glibc 是 Linux 系统中最底层的应用程序接口(API)，几乎其他任何的运行库都会依赖于 Glibc。Glibc 除了封装 Linux 操作系统所提供的系统服务外，它本身也提供了许多必要功能的实现，如字符串处理、信号处理、文件目录操作、共享库动态加载器、动态内存管理等。
- Linux。Linux 内核源码，对于 2.4 内核还需要安装补丁以支持 S3C2410 芯片。

通常配置编译环境有三种方法：

- (1) 分步编译和安装交叉编译工具链所需要的工具和库，最终生成交叉编译工具链。
- (2) 通过 Crosstool 脚本工具来实现编译生成交叉编译工具链。
- (3) 直接下载安装已制作好的交叉编译工具链。

目前 ARM 系统开发资源已非常丰富，无须通过第一种或第二种方法从源代码编译获得，可以直接下载到经测试稳定工作的编译工具链可执行程序。第一种方法相对比较困难，适合想深入学习构建交叉工具链的读者。配置好编译环境后，先运行 make menuconfig 配置内核，再运行 make zImage 命令，即可得到所需要的内核镜像文件。

除编译工具外，开发时常用的工具还包括：

- 串口通信工具。Linux 系统下可使用 Minicom 工具，Windows 系统下可选用超级终端或 DNW 软件等。
- tftp 工具。Linux 系统下可通过 xinetd 设定 tftp 服务，Windows 系统下则需要安装相应工具软件，如 tftpd32。
- flash 烧写工具。可使用 sjf2410(三星公司提供)或 Flash Programmer 等工具；也可使用调试工具将需要烧写内容传输到目标板内存中后，编写程序将内存中内容写入闪存。

3. 引导程序

引导程序是系统加电后运行的第一段软件代码。在嵌入式系统中,通常并没有像 BIOS (Basic Input/Output System) 那样的固件程序,因此整个系统的初始化就完全由引导程序完成:通过这段引导程序初始化硬件设备,建立内存空间映射,设置启动参数,从而将系统的软硬件环境带到一个合适的状态。引导程序完成自己的任务后,会将控制权移交给操作系统。嵌入式系统中,引导程序除了完成对主要硬件如 CPU, SDRAM, FLASH 和串口等进行了初始化外,通常还可以提供下载文件、浏览目录、烧录 flash 及启动系统等。实际上,一个功能较强大的引导程序已经相当于一个微型的操作系统了。

显然,引导程序依赖于硬件实现,因此在嵌入式系统中建立一个通用的引导程序几乎是不可能的,每种不同的 CPU 体系结构有不同的引导程序(有些引导也支持多种体系结构的 CPU,比如 U-Boot 同时支持 ARM 体系结构和 MIPS 体系结构)。除依赖于 CPU 的体系结构外,引导程序也依赖于具体的板级设备的配置。对于两个不同的嵌入式电路板而言,即使它们使用同一种 CPU,要想让运行在一块板子上的引导程序也能运行在另一块板子上,通常也都需要修改引导程序的源代码。

目前在 S3C2410 上使用比较广泛的引导程序有 U-Boot 和 VIVI,它们已提供了对 S3C2410 处理器的支持,只需要根据电路配置情况修改相应代码即可引导系统。

(1) U-Boot 移植简介

① 建立工程

U-boot 是德国 DENX 小组开发的用于多种嵌入式处理器的引导程序,可以通过官方网站 <http://www.denx.de/wiki/U-Boot/SourceCode> 下载得到源代码。该程序遵守 GPL 协议。由于 U-Boot 已支持 S3C2410 芯片(board\smdk2410),所以建立编译环境后即可运行如下命令。

```
$ make smdk2410_config
$ make
```

因编译环境不同,通常需要做如下修改。

- 把目录 arm920t 下 config.mk 中 abi=apcs-gnu 删除。
- 修改 processor.h 中 debug_insn 定义。

```
union debug_insn
{
    u32 arm_mode;
    u16 thumb_mode;
}
```

修改代码并编译通过后,可以得到具有可运行在目标板并具有自启动功能的 U-Boot 镜像文件。由于还未对板载其他设备(闪存芯片、网络芯片等)提供支持,所以还不具备一般意义上嵌入式系统引导程序的功能。

② 修改及配置

针对本训练硬件方案而言,需要修改 U-Boot 源码以支持闪存芯片 EP28F128J3A (Intel) 和以太网芯片 DM9000 (Davicom)。U-Boot 工程中提供了大量常用芯片的驱动程序,故通过少量修改和配置即可完成对 U-Boot 对这两款芯片的支持。

(a) 闪存芯片驱动

在 board\smdk2410 的文件夹中原有闪存驱动(flash.c)是针对 AMD 公司闪存芯片,需要修改成支持 EP28F128J3A 的驱动。驱动写法可参考 board\cmi\flash.c 文件,需要注意的是,由于 cmi 中的闪存

驱动在写入时要交换字节,所以需要修改其中 `write_short()` 和 `wirte_buff()` 函数。另外,由于 E28F128J3A 中块的大小是 128 KB,故 `FLASH_BLOCK_SIZE` 应设置为 0x20000。

E28F128J3A 的配置参数在文件 `include/configs/smdk2410.h` 中,将与闪存配置相关的参数修改成如下内容。

```
#define PHYS_FLASH_1 0x00000000 /*chip select 1 */
#define PHYS_FLASH_SIZE 0x01000000 /* 16 MB */
#define CFG_FLASH_PROTECTION
#define CFG_FLASH_BASE PHYS_FLASH_1
#define CFG_MONITOR_BASE PHYS_FLASH_1
#define CFG_MAX_FLASH_BANKS 1/* max number of memory banks */
#define CFG_MAX_FLASH_SECT 128/* max number of sectors on one chip */
#define CFG_FLASH_ERASE_TOUT (2*CFG_HZ) /* Timeout for Flash Erase */
#define CFG_FLASH_WRITE_TOUT (2*CFG_HZ) /* Timeout for Flash Write */
#define CFG_ENV_IS_IN_FLASH 1
#define CFG_ENV_ADDR (PHYS_FLASH_1 + 0x60000)
#define CFG_ENV_SIZE 0x20000 /* Total Size of Environment Sector */
```

(b) 以太网芯片驱动

U-Boot 工程中已经包含了 DM9000 的驱动(`drive/net/dm9000.c`),需要将该文件加入工程并修改 DM9000 配置参数。

- 修改 `drivers/Makefile`, 在其中加入 `dm9000.o`;
- 在 `lib_arm/board.c` 中加入

```
#ifdef CONFIG_DRIVER_DM9000
#include "../drivers/dm9000x.h"
#endif
```

- `include/configs/smdk2410.h` 中将原 CS8900 配置改为 DM9000。

```
#define CONFIG_DRIVER_DM9000 1
#define CONFIG_DM9000_BASE 0x10000300 /*片选*/
#define DM9000_IO CONFIG_DM9000_BASE
#define DM9000_DATA (CONFIG_DM9000_BASE+4)
#define CONFIG_DM9000_USE_16BIT 1
#define CONFIG_DM9000_DEBUG 1
```

(c) 启动参数

使用 U-Boot 工程中的默认设置是不能启动 Linux 内核的,需要在 `smdk2410.h` 文件加入三个宏定义才能将参数信息传入 LINUX 的 TAG 区。

```
#define CONFIG_SETUP_MEMORY_TAGS
#define CONFIG_INITRD_TAG
#define CONFIG_CMDLINE_TAG
```

启动参数的配置实例如下:

```
#define CONFIG_BOOTDELAY 3
#define CONFIG_BOOTARGS "initrd=0x30800000,0x200000 rw root=/dev/ram
init=/linuxrc \
console=ttyS0,115200"
#define CONFIG_ETHADDR 08:00:3e:26:0a:5b
```

```
#define CONFIG_NETMASK      255.255.255.0
#define CONFIG_IPADDR      192.168.0.100
#define CONFIG_SERVERIP    192.168.0.1
#define CONFIG_BOOTFILE    "boot"
#define CONFIG_BOOTCOMMAND "tftp; bootm"//启动参数，根据需要修改
#define CFG_LOAD_ADDR      0x30800000 /*将内核搬运到内存中的默认地址*/
#define CFG_PROMPT          "SMDK2410 #"
```

此时编译出来的 U-Boot 镜像既可以引导闪存中 Linux 镜像，也可以通过网口远程加载 Linux 镜像启动系统。

(2) VIVI 移植简介

VIVI 是韩国 MIZI 公司开发的引导程序，适用于 ARM9 处理器，其对 S3C2410 支持非常完善，并支持从 Nand Flash 上引导系统。同 U-Boot 一样，VIVI 有两种工作模式：启动加载模式和下载模式。启动加载模式可以在一段时间后(该时间可修改)自行启动 Linux 内核，这是 VIVI 的默认模式。在下载模式下，VIVI 为用户提供一个命令行接口，通过接口可以使用 VIVI 提供的一些命令，如表 18.1 所示。

表 18.1 VIVI 常用命令说明

命 令	功 能
Load	把二进制文件载入闪存或内存
Part	操作 MTD 分区信息，显示、增加、删除、复位、保存 MTD 分区
Param	设置参数
Boot	启动系统
Flash	管理闪存，如删除闪存中的数据等

当完成 VIVI 的源代码修改后需进行配置和编译，以生成烧入闪存的二进制文件。由于 VIVI 要用到 kernel 的一些头文件，所以需要准备好 kernel 的源代码，并修改/vivi/Makefile 里的一些变量设置(变量实际取值需根据编译环境调整)。

```
LINUX_INCLUDE_DIR = /usr/local/arm/2.95.3/include
CROSS_COMPILE = /usr/local/arm/2.95.3/bin/arm-linux-
ARM_GCC_LIBS = /usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3
```

设置完毕并保存后，进入 vivi 目录，执行“make menuconfig”进行配置。最后运行“make”开始编译，编译结束后在 vivi 目录里面会生成文件“vivi”，这个就是后面要烧写到 Nand Flash 中的引导程序。

4. 系统内核

Linux 内核是一个庞大而复杂的操作系统的核心，不过尽管庞大，却通过子系统和分层的方式进行了很好的组织。

图18.3所示的最上面是用户(应用程序)空间，这是用户应用程序执行的地方。用户空间之下是内核空间，Linux 内核正是位于这里。内核的作用实际上是一个资源管理器，被管理的资源包括进程、内存和硬件设备，内核负责管理并裁定多个竞争用户对资源的访问(既包括内核空间也包括用户空间)，Glibc 也在这里。它提供了连接内核的系统调用接口，还提供了在用户空间应用程序和内核之间进行转换的机制。这一点非常重要，因为内核和用户空间的应用程序使用的是不同的保护地址空间。每个用户空间的进程都使用自己的虚拟地址空间，内核则占用单独的地址空间。

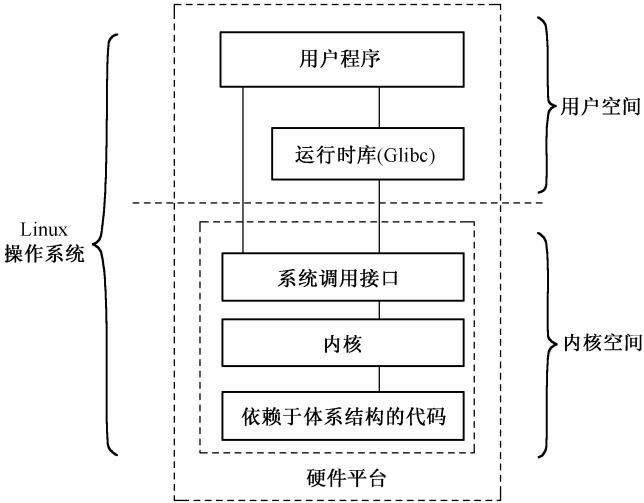


图 18.3 Linux 操作系统的基本体系结构

Linux 内核可以进一步划分成 3 层。最上面是系统调用接口，它实现了一些基本的功能，例如读和写。系统调用接口之下是内核代码，可以更精确地定义为独立于体系结构的内核代码。这些代码是 Linux 所支持的所有处理器体系结构通用的。在这些代码之下是依赖于体系结构的代码，构成了通常称为 BSP (Board Support Package) 的部分。这些代码必须考虑硬件体系结构才能正常操作并实现更高效率。linux/arch 子目录定义了内核源代码中依赖于体系结构的部分，其中包含了各种特定于体系结构的子目录。Linux2.6 内核已提供了 S3C2410 支持，Linux2.4 通过补丁也可支持该芯片，本训练不再探讨这部分的移植方法。

设备驱动程序是 Linux 内核的重要组成部分，控制了操作系统和特定硬件设备之间的交互。Linux 源码提供了一个驱动程序子目录 (linux/drivers)，这个目录又进一步划分为各种支持设备，如 Bluetooth, I2C 和 UART 等。Linux 操作系统中设备驱动模块的功能是扩展内核的功能，主要完成两部分任务：一个是系统调用，另一个是处理中断。系统调用是指对设备的操作过程，如 open, read, write, ioctl 等操作，设备驱动程序所提供的这组入口点由几个结构向系统进行说明，分别是 file_operations 数据结构、inode 数据结构和 file 数据结构。内核通过 file 结构识别设备，通过 file_operations 数据结构提供文件系统的入口点函数，也就是访问设备驱动的函数，结构中的每一个成员都对应着一个系统调用。在嵌入式系统开发中，一般仅实现其中几个接口函数：read, write, open, ioctl 及 release 就可以完成应用系统需要的功能。写驱动程序的重要任务之一就是完成 file_operations 中的函数指针。

Linux 系统中的设备分为三类：块设备、字符设备和网络设备。

- 块设备通常指需要以块 (block) 的方式写入的设备，如 IDE 硬盘等。
- 字符设备通常指可以直接读、写，没有缓冲区的设备，如并口等。
- 网络设备通常指网络设备访问的 BSD socket 接口，如网卡等。

对这三种设备驱动程序，设计时会有一定区别。接下来以触摸屏和 IDE 硬盘驱动为例，简要介绍 Linux 下字符设备和块设备驱动程序编写和修改的一般方法。

(1) 触摸屏驱动分析

数码伴侣采用四线电阻式触摸屏，其包含上下叠合的两个透明层，两层由具有相同表面电阻的透明阻性材料组成，通常用一种弹性材料将两层隔开。当触摸屏表面受到的压力 (如通过笔尖或手指进行按压) 足够大时，顶层与底层之间会产生接触。电阻式触摸屏采用分压器原理来产生代表 X 坐标和 Y

坐标的电压。如图 18.4 所示,分压器是通过将两个电阻进行串联来实现的。上面的电阻连接正参考电压,下面的电阻接地。两个电阻连接点处的电压测量值与下面那个电阻的阻值成正比。为了在电阻式触摸屏上的特定方向测量一个坐标,需要对一个阻性层进行偏置:将它的一边接 V_{REF} ,另一边接地。同时,将未偏置的那一层连接到一个 ADC 的高阻抗输入端。当触摸屏上的压力足够大,使两层之间发生接触时,电阻性表面被分隔为两个电阻。它们的阻值与触摸点到偏置边缘的距离成正比。触摸点与接地边之间的电阻相当于分压器中下面的那个电阻。因此,在未偏置层上测得的电压与触摸点到接地边之间的距离成正比。

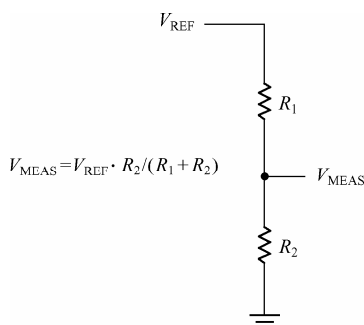


图 18.4 电阻式触摸屏工作原理

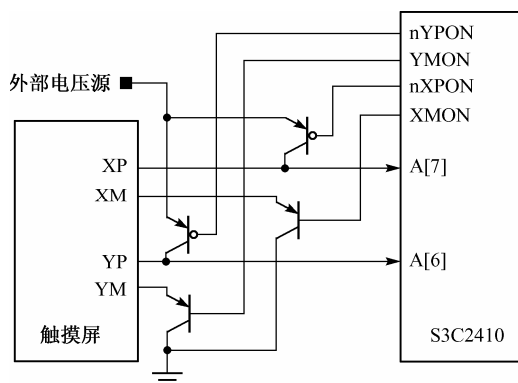


图 18.5 触摸屏接口电路

以三星公司提供的触摸屏驱动(linux/drivers/char/s3c2410-ts.c)为例分析,在该程序中有三个重要的数据结构:用于表示笔触点数据信息的结构 TS_RET,表示触摸屏控制器信息的结构 TS_DEV 及驱动程序与应用程序接口 file_operations 结构(s3c2410_fops)。

```
typedef struct {
    unsigned short pressure;
    unsigned short x;
    unsigned short y;
    unsigned short pad;
} TS_RET;
```

TS_RET 结构体中的信息就是驱动程序提供给上层应用程序使用的信息,用来存储触摸屏的返回值。上层应用程序通过接口从底层驱动中读取信息,并根据得到的值进行操作。

```
typedef struct {
    unsigned int PenStatus;
    TS_RET buf[MAX_TS_BUF];
    unsigned int head, tail;
    wait_queue_head_t wq;
    spinlock_t lock;
} TS_DEV;
```

TS_DEV 结构用于记录触摸屏运行的各种状态, PenStatus 包括 PEN_UP, PEN_DOWN 和 PEN_FLEETING。buf [MAX_TS_BUF]是用来存放数据信息的事件队列, head 和 tail 分别指向事件队列的头和尾。程序中的笔事件队列是一个环形结构,当有事件加入时,队列头加 1;当有事件被取走时,队列尾加 1;当头尾位置指针一致时读取笔事件的信息,进程会进入睡眠。wq 等待队列,包含一个锁变量和一个正在睡眠进程链表。当有好几个进程都在等待某件事时, Linux 会把这些进程记录到

这个等待队列。它的作用是当没有笔触事件发生时，阻塞上层的读操作，直到有笔触事件发生。lock 使用了自旋锁，用于保护临界资源。MAX_TS_BUF 的值为 16，即在没有被读取之前，系统缓冲区中最多可以存放 16 个位置信息。

```
static struct file_operations s3c2410_fops = {
    owner: THIS_MODULE,
    open: s3c2410_ts_open,
    read: s3c2410_ts_read,
    release: s3c2410_ts_release,
    poll: s3c2410_ts_poll, };
```

s3c2410_fops 就是内核对驱动的调用接口，将驱动函数映射为标准接口。代码中这种特殊表示方法是 GNU 编译器的一种特殊扩展，它使用名字进行结构字段的初始化，它的好处体现在结构清晰，并且避免了结构体发生变化带来的问题。

表 18.2 列出了触摸屏驱动中主要函数的名称和功能。驱动程序在装载时，会向系统注册两个中断处理函数：一个中断发生在触摸屏接触或释放时，另一个中断发生在模数转换完毕。触摸屏驱动程序的运转即基于这两个中断的处理。在实际开发中，可通过对该驱动加入软件滤波和软件去抖来获得更好的使用效果：对于抖动，原因在于触摸屏刚接触的一瞬间测量电压不稳定，造成获取坐标不精确。可通过在收到触摸中断处理中设定定时器以延迟 20 ns 开始数模转换解决；软件滤波则是对同一次接触进行多次读取坐标并取平均值来得到更好的坐标精度，即在 s3c2410_get_XY() 函数中增加相应代码。

表 18.2 触摸屏驱动中主要函数的名称和功能

函 数 名	功 能
s3c2410_ts_init()	初始化并注册设备包括配置 I/O 引脚，注册中断，生成设备节点等
s3c2410_ts_exit()	从系统中删除设备，撤销中断和设备注册信息
s3c2410_ts_open()	初始化队列、定时器等资源
s3c2410_ts_release()	释放定时器等资源
s3c2410_ts_read()	将内核空间数据复制到用户空间，当没有数据时休眠进程
s3c2410_get_XY()	获取触摸屏触点坐标
s3c2410_isr_adc()	模数转换中断处理函数
s3c2410_isr_tc()	触摸中断处理函数

(2) 硬盘驱动分析

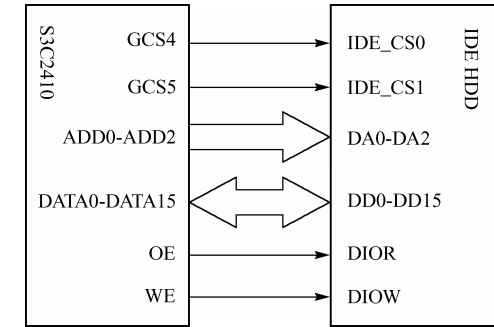


图 18.6 S3C2410 扩展 IDE 接口示意图

IDE (Integrated Drive Electronics) 规范是从 IBM PC/AT 上使用的 ATA 接口发展而来的。ATA 接口规范定义了信号电缆和电源线的电器特征、互连信号的电器和逻辑特征，还定义了存储设备中可操作的寄存器及命令和协议。IDE 磁盘驱动器与早期的 ATA 驱动器相比，增加了任务文件寄存器，包括数据寄存器、状态寄存器，以及反映地址的驱动器号、磁头号、道号和扇区号寄存器等。

在寄存器方面，IDE 规范定义了两组寄存器：命令寄存器和控制寄存器。命令寄存器用来接收命令和传送数据，控制寄存器用来控制磁盘操作。常用的寄存器包括数据寄存器、命令寄存器、驱动器 / 磁头寄存器、柱面号寄存器、扇区号寄存器、扇区数寄存器和状态寄存器。

在数据传输方面, IDE 接口规范定义了两种数据传输方式: 可编程 I/O (PIO) 方式和 DMA 方式。PIO 传送方式下, CPU 对控制器的访问都是通过 PIO 进行的, 包括从控制器读取状态信息和错误信息, 以及向控制器发送命令和参数。在一次 PIO 数据传输过程中, CPU 先选址, 然后使读 / 写信号有效, CPU 或控制器放置数据到数据总线, 控制器或 CPU 读取数据, 操作完成后, 释放总线, 这样一次数据传输完成。DMA 方式, 即直接内存访问, CPU 把缓冲区的地址与需要读 / 写的长度告诉外设, 外设准备好后向 CPU 发出一个 DMA 请求, 要求 CPU 暂停使用内存, 获得同意后就直接在内存和外设之间传输数据, 完成后再把对内存的访问权归还给 CPU。

由于 S3C2410 处理器使用 DMA 时序与 IDE 硬盘不同, 通常使用 PIO 方式操作硬盘。PIO 方式读 / 写速度较 DMA 方式慢, 但对本例已达到实用需要, 方案采用 PIO 方式读 / 写 IDE 硬盘。

在 Linux 系统中, IDE 硬盘属于块设备, 块设备驱动程序的编写与字符设备很类似, 也包括了注册和使用两个部分。但与字符设备驱动不同的是, 块设备驱动程序包括一个请求 (request) 队列。块设备数据结构 `blk_dev_struct` 定义如下:

```
struct blk_dev_struct {
    request_queue_t request_queue;
    queue_proc *queue;
    void *data;
};
```

Linux 源码中已包含 IDE 硬盘的完整驱动, 只需要根据目标板配置做少量修改: (1) 修改硬盘控制器寄存器访问基地址; (2) 修改 Bank 宽, 由于 IDE 硬盘中命令寄存器与数据寄存器宽度不同, 前者 16 位, 后者 32 位。所以在访问不同寄存器时, 要重新设置 Bank 宽度, 才能让数据被正确写入相应寄存器。Linux 系统对硬盘读写可用伪代码表示如下:

```
DO_RW_DISK(COMMAND)
{
    Set_Registers0;
    if(COMMAND=READ){
        Set read_intr as interrupt process function
        Send WIN_READ or WIN_MULTREAD command to Command register
    }
    if(COMMAND=WRITE){
        Send WIN_WRITE or WIN_MULTWRITE command to Command register
        Get the status of Status register and set DRQ bit
        Set write_intr as interrupt process function
        Send data to buffer in the disk
    }
}
```

5. 文件系统建立

这里的文件系统指在一个物理设备上的文件组织和目录, 它构成了 Linux 系统上所有数据的基础, Linux 程序、库、系统文件和用户文件都驻留其中, 因此它是 Linux 操作系统中庞大复杂且又是最为基本和重要的资源。需要指出的是, Linux 系统中的文件不仅包括普通的文件和目录, 每个和设备相关的实体也被映射为文件, 如磁盘、终端等。Linux 下的文件是操作系统服务和设备的、简单而又统一的接口。在本训练中, 操作系统将从闪存启动, 常见装载文件系统的方法有如下三种:

① Ramdisk。Ramdisk 方法将制作好的文件系统压缩后写入闪存, 启动时, 由引导程序装载入内

存解压缩后挂载到系统根节点。该方法实现简单，但由于在内存中非压缩保存，会占用到嵌入式系统宝贵的内存资源。

② Cramfs。Cramfs 是一个很简单的文件系统，并具有很高的压缩率，可以直接从闪存上运行而不需要装载入内存。但 Cramfs 是只读的，对于需要运行时修改的目录(如/etc, /var, /tmp)则会造成不便。

③ JFFS2。JFFS2 是一种专门针对闪存并具有读 / 写功能的存储格式，支持从闪存上运行。与前两种存储方式相比，JFFS2 移植相对复杂。

由于本训练不需要保存数据到闪存中，并且设计方案拥有足够的内存空间，可使用较简单的 Ramdisk 或 Cramfs 格式存储。通常在嵌入式 Linux 系统中受到存储空间限制，并不需要像桌面系统那样庞大的文件系统，需要开发者选择能满足需求的最少数量的文件制作文件系统。三星厂家提供的开发包中，包含 Cramfs 格式的示例文件系统，可在此基础上增 / 减文件供数码相机伴侣系统使用。也可以重新建立文件系统，步骤如下。

(1) 构造基本目录

建立文件系统时，可按照表18.3所示的基本目录结构构造并填充文件。需要注意的是，这里使用的可执行文件及库文件都需要交叉编译以运行在 ARM 平台。

表 18.3 Linux 系统基本目录结构

bin	用于存放 Linux 系统常用执行文件，如 mv, ls, mkdir 等
dev	用于存放 Linux 系统中使用的外部设备文件
etc	用于存放 Linux 系统管理时用到的各种配置文件和子目录
lib	该目录存放系统动态链接共享库
mnt	硬盘、读卡器等设备的挂载点
proc	用于存放系统核心与执行程序所需的信息
sbin	用于存放系统管理员常用程序
tmp	用于存放不同程序执行时产生的临时文件
usr	用户应用程序和文件的保存目录
var	服务日志信息保存目录
home	Linux 系统默认的用户根目录
root	超级用户登陆时的主目录

(2) 配置脚本

系统初启时会运行一些列脚本来配置和初始化系统。这些脚本及配置文件非常重要，如果配置不当会影响系统启动。这些脚本主要放置在 etc/rc.d 中，独立运行的系统服务启动脚本则放置在 etc/rc.d/init.d 文件夹下。读者可参考桌面系统中相关配置文件进行定制。

(3) 文件系统打包

使用工具将整理好的目录结构打包生成 Ramdisk 文件或 Cramfs 文件，并烧写进闪存。

6. 应用程序开发

设计良好的用户图形界面(GUI)能为使用者提供友好便利的界面，方便了非专业用户使用，将使用者从烦琐的命令中解脱出来，仅需按动屏幕或按键即可完成操作。随着嵌入式系统配置的提升，越来越多的嵌入式产品采用了图形界面。嵌入式系统中受硬件资源限制，用户图形界面要求直观、可靠、资源占用少。另外，由于嵌入式系统的多样性，嵌入式图形界面也需要具有高度可移植性和可裁剪性，以适应不同的应用场合。嵌入式图形界面开发系统一般具有以下特点：体积小；效率高；可移植性强；可靠性高。

支持 Linux 的图形开发系统比较多,常用的有 MiniGUI, MicroWindow 和 Qt/Embedded 等。在嵌入式环境下, GUI 系统程序开发与桌面应用程序开发类似,如绘图函数库、字型库和事件处理机制等都是嵌入式图形系统所要面临的问题。但在整体设计上必须考虑到嵌入式系统本身的特点,如分辨率低、运算资源少等。

在桌面 Linux 系统中,目前主要使用基于 X 标准的用户图形界面,但对于嵌入式系统来说, X 系统过于庞大和低效。所以目前主流嵌入式图形界面并不局限于 X 标准,其更强调系统的运行效率。在嵌入式 Linux 中有大量免费或商业图形界面库供选择。

(1) MiniGUI

MiniGUI 是国人做得最为成功的开源项目之一,它是在 Linux 控制台上运行的多窗口图形操作系统,可以在以 Linux 为基础的应用平台上提供一个简单可行的 MiniGUI 支持系统。MiniGUI 典型应用只有 300 KB 左右,可以应用在电视机顶盒、工业控制系统等诸多场合。MiniGUI 对中文的支持非常完善,其他字符集也可以轻松加入。

MiniGUI 开发的主要目标就是为基于 Linux 的实时嵌入式系统提供一个轻量级的图形用户界面支持系统。MiniGUI 为应用程序定义了一组轻量级的窗口和图形设备接口。利用这些接口,每个应用程序可以建立多个主窗口,然后在这些主窗口中创建按钮、编辑框等控件。MiniGUI 还为用户提供了丰富的图形功能,帮助开发人员显示各种格式的位图并在窗口中绘制复杂图形。

(2) Nano-X

Nano-X 由 MicroWindows 项目更名而来,该开源项目的宗旨就是针对体积小的装置建立一套先进的视窗环境。在 Linux 桌面上通过交叉编译可以很容易地制作出 Nano-X 的程序。MicroWindows 能直接对显示设备进行操作,能够在没有任何操作系统支持的情况下独立运行。因此 Nano-X 就十分小巧,便于移植到各种硬件和软件系统上。然而 Nano-X 系统升级维护工作缓慢,并缺乏足够文档支持,影响了该系统的推广使用。

(3) Qt/Embedded

Qt 是一个跨平台的 C++ 用户图形界面库,由挪威 TrollTech 公司出品,目前包括 Qt、基于 FrameBuffer 的 Qtopia Core、快速开发工具 Qt Designer 和国际化工具 Qt Linguist 等部分。Qt 支持包括 Linux 系统在内的所有的 Unix 系统,并支持 WinNT/Win2k, Windows 95/98 平台。Qt 还具有下列一些优点:优良的跨平台特性;面向对象封装;丰富的 API;支持 2D/3D 图形渲染;支持 OpenGL;大量的开发文档。

Qt/Embedded 是在嵌入式环境下所使用的 Qt。目前市面上基于 Linux 操作系统的掌上电脑几乎都采用 Qt/Embedded 开发图形界面。Qt/Embedded 可以直接在缓冲帧(framebuffer)上显示图形,反应的速度更快,这对硬件和容量都有限制的嵌入式环境非常重要。

(4) OpenGL ES

OpenGL ES 是一个轻量级的、无授权费用的嵌入式图形标准,可为很多嵌入式系统和设备提供图形 API 轮廓,包括手持无线设备、汽车和航空显示设备、多媒体消费电子设备,如高级数字电视、机顶盒和游戏控制台。

在常用图形界面开发系统中,以 Qt 文档及软件资源最为丰富,但其对系统硬件要求也最高。考虑到本训练采用硬件方案拥有 64 MB 内存和 202 MHz 主频,能够轻松运行 Qt 应用程序,方案采用 Qt/Embedded 开发数码相机伴侣应用软件。

① 搭建 Qt/Embedded 开发环境

由于本训练中用 Qt/Embedded 开发的应用程序最终会发布到安装有嵌入式 Linux 操作系统的设备上,所以使用装有 Linux 操作系统的 PC 来完成 Qt/Embedded 开发是最理想的环境。此外,Qt/Embedded

也可以安装在 Unix 或 Windows 系统上。以安装到 Linux 操作系统为例, 步骤如下: 下载并配置 Qt/Embedded 安装包; 下载并配置 Qt 的 X11 版安装包; 设置环境变量。

安装完成后, 进入 Qt/Embedded 安装目录下的 demos/chip, 运行 “qvfb &” 和 “./chip -qws”, 如果安装正确, 则此时可以看到示例程序的图像。Qt/Embedded 在 Linux 平台下开发时, 可以独立运行, 也可以使用虚拟缓冲帧的方式运行。对于后者, 其实就是用了一个 X11 应用程序虚拟了一个缓冲帧。可以通过制定设备的宽度、高度和色彩深度, 虚拟出来的缓冲帧将和物理的现实设备在每个像素上保持一致。这样软件开发人员可以在 PC 上获得目标机上运行效果, 使软硬件可同时开发, 加快了软件开发调试速度。运行 Qt 的虚拟缓冲帧工具的方法是在 Linux 图形模式下运行命令 qvfb。当 Qt 嵌入式应用程序要把显示结果输出到虚拟缓冲帧时, 在命令行运行程序时, 在程序名后加上 -qws 选项即可。

② 可视化程序开发

Qt 提供了大量的类和函数来创建 GUI 程序。Qt 既可用于创建“主窗口”式界面, 包含菜单栏、工具栏和状态栏等; 也可以用来创建“对话框”式界面, 使用按钮与选卡来呈现信息。Qt 支持单文档视图和多文档视图, 还支持拖动和剪贴板。Qt 甚至支持工具栏的拖曳、移动、托盘浮起等, 并且这些功能是内建的, 不需要额外代码。使用 Qt 编写可视化应用程序非常简单方便, 并可大大简化应用程序编写工作量。如下例数行代码就可生成一个“Hello World”可视化窗体。

```
#include <qapplication.h>
#include <qlabel.h>
Int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    QLabel *hello = new QLabel( "<font color=blue>Hello World!" ,0);
    hello->show();
    return app.exec();
}
```

虽然基于 Qt 能够大大加速可视化应用程序开发周期, 但对于本训练应用程序工作量仍非常庞大。如图片浏览编辑功能软件、多媒体播放软件、游戏软件等, 每一类都需要精心设计和较长的开发测试周期。幸运的是, 在 Linux 下有大量开源免费代码可供使用(包括 PDA 开发软件套件), 通过这些资源的整合利用, 可迅速搭建一套掌上系统软件环境。

18.3 本章小结

S3C2410 处理器是一款基于 ARM920T 内核的 32 位片上系统, 具有体积小、功耗低、资源丰富、扩展能力强等优良特性, 广泛应用于工业控制及消费类电子产品中。本训练基于 S3C2410 处理器完成了一个掌上系统的设计, 通过分类细化训练内容, 让读者掌握基于 Linux 操作系统的嵌入式设备开发的一般方法和原理。希望读者在对这些方法掌握的基础上, 能够完成任意此类掌上系统或工控系统的研发。

第 19 章 电子系统工程实现中的问题

19.1 概述

一个电子系统，除需要达到预期的功能、指标外，还涉及到其他若干工程技术问题，如电磁兼容、可靠性等问题。实际上，这些也是系统设计时所应考虑的性能指标。在完成一个电子系统的方案设计与电路设计以后，接下来的工作是实物制作，涉及到的工作包括：搭建电路、测试功能与指标，部件与整体的结构设计及在所需工作条件下做运行试验等，这期间会遇到与计算机模拟结果不一致的许多问题，往往要对原方案及电路设计不断做出修改。另外，还要更多地考虑到电子产品在以后付诸生产制造时所面临的各种工艺技术问题，乃至日后的使用维护、市场营销等诸多问题。因此可以说，电子系统的工程实现阶段，是处在承前启后的中间地位，是一个理论与实践相结合的重要环节。工程实现的过程涉及较广的知识面，其最终质量相当大程度上依赖于研制者的经验、技巧，必须在制作实践过程中不断学习总结与积累。

本章介绍设计模拟、数字及智能化电子系统时所涉及的工程实现方面的有关问题，如抗电磁干扰、电磁兼容性、热特性、可测性、可靠性及其他一些有特殊要求的电子系统在实现时要考虑的问题。有的应作为一种工程设计理念，还有不少是体现为实现过程中的一些技术与工艺问题，也有的需从方案、电路设计本身或元件选用方面来考虑。

19.2 电子系统的抗干扰设计

19.2.1 电磁干扰与电磁兼容问题

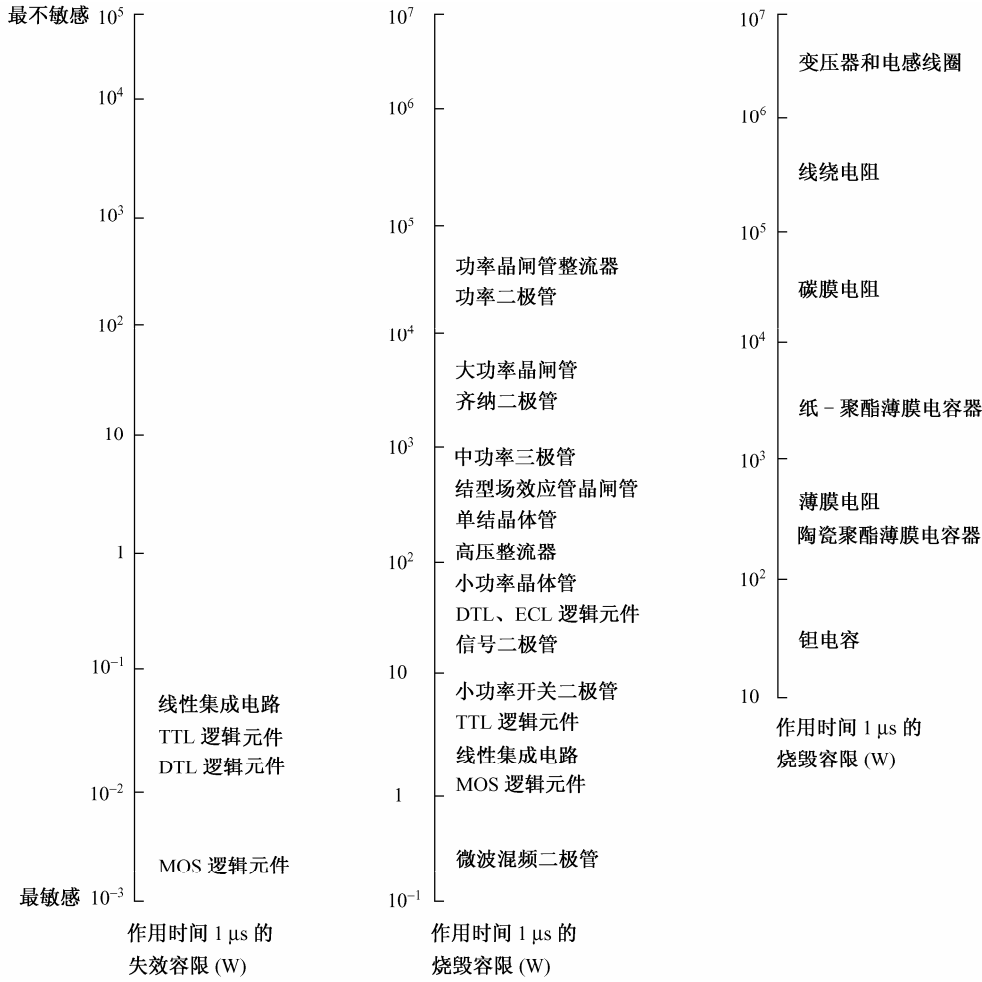
电子设备的周围充满着自然的及人为的电磁干扰信号，而电子设备本身对于其他的设备而言又是一个干扰信号源。提高设备的抗干扰能力，同时降低电子设备本身对周围电磁环境的污染，这是近几十年来备受重视的电磁兼容(EMC)问题。已经制定了相应的国际、国家及行业标准和规范，这些标准包括对电磁干扰的控制要求、安全限值、测量方法等，在设计电子系统时应参照执行。

干扰对电子设备可形成不同程度的危害，轻则可使设备的性能指标下降，重则可使设备不能正常工作，甚至可使机内较为脆弱的半导体器件击穿或烧毁。表 19.1 和表 19.2 列出了部分元器件静电放电易损值和失效烧毁容限。值得注意的是，在干燥的气候条件下，人体可带有上万伏的感应静电。此时，如果触摸甚至靠近 MOS 类高阻抗器件，很可能导致器件损坏。

表 19.1 常见半导体器件的静电放电易损值

器件类型	对静电放电的易损值/V	器件类型	对静电放电的易损值/V
肖特基二极管	300~2500	JFET	140~7000
肖特基 TTL	1000~2500	CMOSFET	100~200
双极晶体管	380~7000	CMOS	250~3000
ECL	500~1500	GaAsFET	100~300
可控硅	680~1000	EPROM	100

表 19.2 部分元器件静电放电易损值和失效烧毁容限



19.2.2 干扰的类型

1. 自然干扰源

宇宙射线、太阳黑子、耀斑、雷电等伴随的电磁活动，这些干扰信号常常造成通信、广播的中断，甚至设备的损坏。图19.1示出了自然界电磁噪声的频谱分布。各类噪声还有其时空分布。

2. 人为干扰源

对电子设备本身无用的信号即可视为干扰信号，如电台发射的无线电波、可控硅电力电子装置对工业电源的污染、车辆的点火电火花污染等。在同一个电子设备内部，各部分之间产生相互干扰，使得设备不能按预期设计的性能工作。

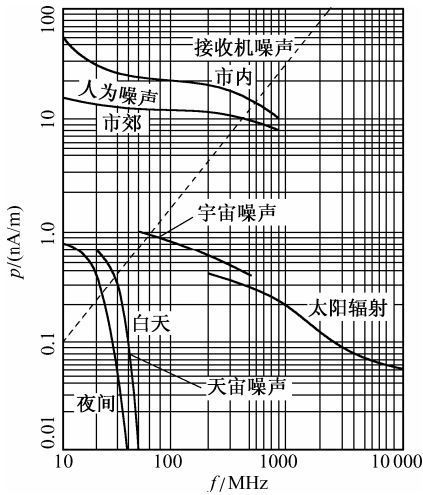


图 19.1 磁噪声特性

19.2.3 干扰传播的途径

干扰信号作用于电子设备，有传导和辐射两种途径。电子设备中的导线、元器件、结构体等都能形成传导和辐射耦合通道。它们有时能起着天线的作用，能够发射和接收干扰电磁波。图 19.2 中对各种类型的传导耦合和辐射耦合做了归纳与划分。

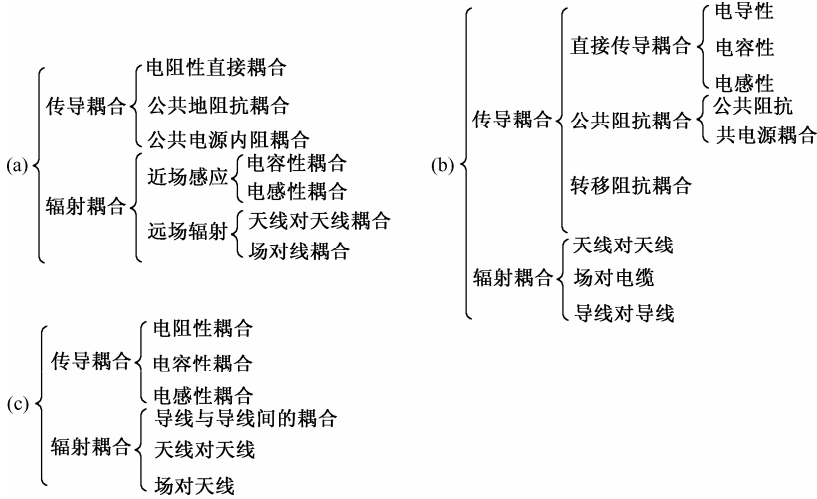


图 19.2 传导耦合和辐射耦合的划分

19.2.4 抗干扰设计方法

对不同的干扰信号应采取不同对策。干扰信号本身占据一定频带及空间方位，有的出现于一定时间段，有的有不同的传播途径，针对这些不同的特点，可采用不同的抗干扰方法：一种是将其拒之门外，常用电磁屏蔽技术阻隔干扰信号的传播通道，或设法在时域、频域及方位上使干扰信号与电子设备本身的工作信号分开，常用的载波技术就可以有效地从频率上将两者分开；另一种办法是“阻塞不如开导”，采用旁路、吸收等方法来消除干扰信号，这往往可获得很好的效果，表 19.3 列出了电磁兼容控制的一些策略。

表 19.3 电磁兼容控制策略

传输通道抑制	空间分离	时间分隔	频域管理	电气隔离
滤波	地点位置控制	时间共用准则	频谱管制	变压器隔离
屏蔽	自然地形隔离	雷达脉冲同步	滤波	光电隔离
搭接	方位角控制	主动时间分隔	频率调制	继电器隔离
接地	电磁场矢量方向控制	被动时间分隔	数字传输	DC/DC 变换
布线			光电转换	电动-发电机组

以下是电子系统工程实现中常用的抗干扰方法。

(1) 屏蔽。采用良导体材料制成电及电磁屏蔽罩，其性能与材料种类及厚度等有关，应该进行专门的设计计算。通常用高导磁材料，如铍莫合金等制成磁屏蔽罩。屏蔽罩的散热孔大小应经过计算。常利用设备外壳做屏蔽，应注意与电路地实现良好的搭接。若采用塑料机箱，在必要时应在内层喷涂金属作为屏蔽层。

(2) 注意元件的安装位置与角度。特别是变压器、电感线圈等产生的磁场具有特定的方向性，应考虑它们对其他电路的影响。

(3) 采用双绞线、同轴电缆作为长距离信号的传导线(参见图19.3)。

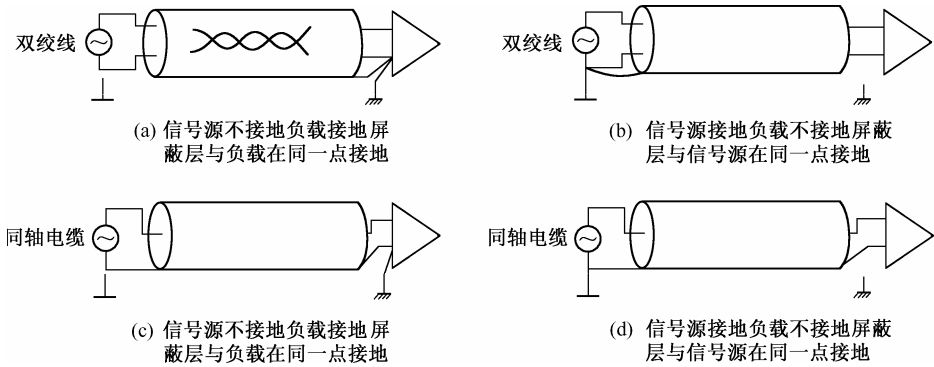


图 19.3 采用双绞线、同轴电缆作为长距离信号的传导线及接地方案

(4) 采用变压器隔离(参见图19.4)。

(5) 采用光电隔离(参见图19.5)。

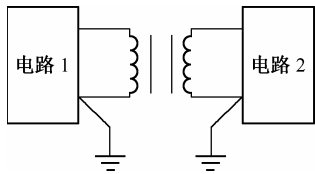


图 19.4 采用变压器隔离

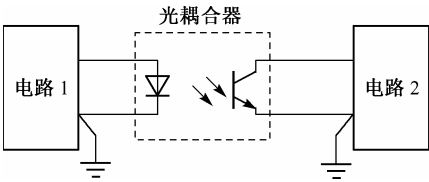


图 19.5 采用光电隔离

(6) 滤波。采用滤波器使信号与干扰在频谱上分离。为了减少宽带噪声干扰，在可能的条件下，使电路的各个环节呈窄带特性，以限制噪声。对来自电网中的干扰，可通过在进线处的 LC 低通滤波来抑制。特定频带的干扰信号采用陷波滤波器容易滤除。

(7) 接地。电子设备内部电路的地线与电源线是所有电路的公共部分。各部分电路的接地方式可归纳为如图19.6所示的几种。由于各部分电路的工作电流通过电源与引线而相互耦合，从而形成串扰，因此印刷板电路的地线与电源引线必须精心设计。

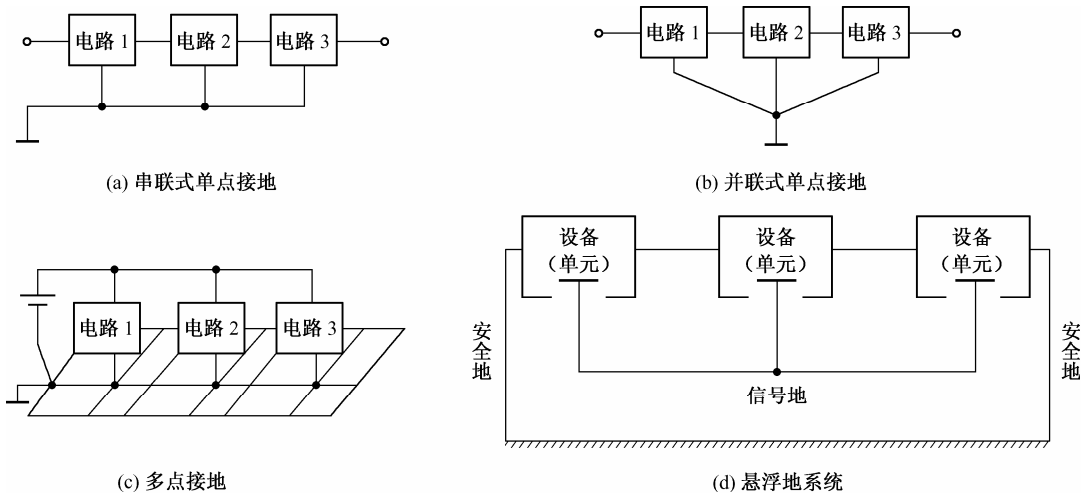


图 19.6 各种接地方式

串联式单点接地因公共部分跨越的电路多,各部分电路工作信号通过公共地线部分的电路起到了相互耦合的作用。如果各部分电路呈由小信号至大信号逐级级联的形式,最好采用各部分电路分别并联接地。各部分电路与公共电源线相连时,其间应加电源去耦低通滤波,而使高电平、大电流的电路部分(往往是整个电路的输出部分)最接近供电电源(参见图 19.7)。数字电路与模拟电路应分别接地。在设计印刷板电路时,应将数字电路与模拟电路分别连至各自的地线上,再连到电源的地线上,如图 19.8 所示。

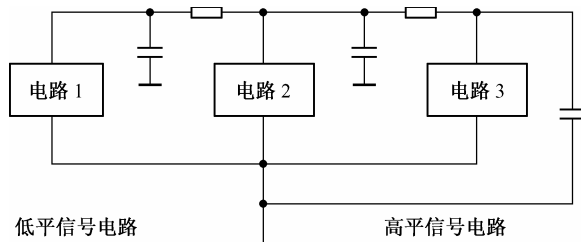


图 19.7 多级电路的供电

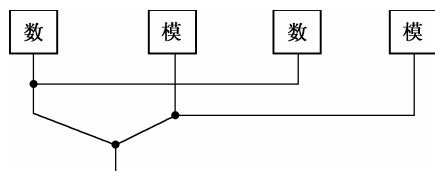


图 19.8 数字地与模拟地分开引出

19.3 电子设备热设计

电子系统内几乎都含有对温度敏感的半导体器件,温度对电子设备的影响包括:使性能下降,不能稳定可靠工作,直至器件受损或毁坏。热设计的目标,就是使设备的整机温度和内部各部分电路元件之间的温差分布保持在一个控制值之内。

器件的工作温度范围只有几十或一百多摄氏度,温度太低,电路不能正常工作,通常允许经一定时间的预热升温,达到规定的性能指标才使用。而实际中遇到的更多的问题是,要解决因大规模芯片和大功率电路自身发热,以及高温环境下设备的散热降温问题。可以通过温度补偿,采用正负温度系数互补器件设计电路、设计恒温控制系统、选用宽温范围工作的器件等方法来改善电路的温度特性。这里的热设计,主要是指对设备的发热器件(包括有源的和无源的)及整个设备的温度进行控制而采取的措施。

19.3.1 功率器件的散热

对系统内主要的发热器件,多采用安装散热片或局部风冷的办法。应该对大的发热元件留有较大的周围空间,采用易于散热的安装形式,例如大功率的晶体管或大瓦数的电阻元件,不应该采取紧贴印刷板的方式安装。

根据散热片的材料种类、形状、与发热器件的接触方式(直接贴紧或加绝缘层,常用云母片或硅脂等导热而不导电材料作为绝缘层)及不同器件的管型,可依据热设计的专用手册计算出合理的散热片大小(表面积)。在散热器上附加风扇能获得更好的效果。

用于大型功率器件和设备的散热方法还有:

- (1) 冷板,即通过流体作为热交换的装置,流体可使用空气(用风扇驱动)、水或其他制冷剂。
- (2) 相变冷却系统,利用介质相变(从液态变为汽态)时可吸收大量热量的原理工作。
- (3) 利用半导体制冷器件散热。

这些散热方法常用于大功率发射管及大功率激光管的散热上。其中水冷法用于 500A 以上电流的器件,相变冷却系统具有最高的冷却效率。

通过 EDA 工具,可以根据所设计的 PCB 上的温度分布效果图,对元件排列位置和密度进行调整。元件发热,可形成自然对流,PCB 竖立安装通常比横卧安装的散热效果要好。

19.3.2 整机的散热

常用的方法是机箱开通风孔, 安装排风扇。在利用排风扇排热时, 应注意风路的设计, 使主要的发热元件处于风路上, 此时机壳的通风孔并非越多越好, 必须与风路设计统一考虑, 同时兼顾通风孔对电磁干扰等问题的影响。许多电子设备带有金属外壳, 常利用来兼做散热器, 将功率器件直接安装在外壳上, 但应注意电绝缘和安全问题。

19.4 可靠性设计

可靠性设计是指在设计电子系统时, 不仅要关心系统的功能, 还应将可靠性列为一项要达到的指标。设备的可靠性是指设备能在规定的时间内完成规定的功能。设备的故障带有随机性, 可用平均故障时间 MTBF 来描述:

$$\text{MTBF} = \frac{\text{总工作时间}}{\text{故障次数}} \quad (\text{h}) \quad (19.1)$$

现在, 国内外电子行业都已经把 MTBF 作为定量评价产品质量的主要标准之一。

整机的可靠性与它所使用的元器件的可靠性、元器件的数量、电路的设计质量、系统的可靠性结构类型等有关。元器件的可靠性用失效率表示。失效率是指工作到某时刻尚未失效的产品, 在该时刻后单位时间内发生失效的概率。一般记为 λ , 它也是时间 t 的函数, 故也记为 $\lambda(t)$, 称为失效率函数, 有时也称为故障率函数或风险函数。失效率的观测值是在某时刻后单位时间内失效的产品数与工作到该时刻尚未失效的产品数之比, 即

$$\hat{\lambda}(t) = \frac{\Delta N_f(t)}{N_s(t)\Delta t} \quad (19.2)$$

元器件的失效率参数由生产厂家提供, 应按照电子系统对可靠性的要求和成本控制要求等选择使用。

从可靠性角度来看, 众多的元件在系统中处于串联和并联两种结构状态。若 N 个元件中任意一个失效, 都会引起整个系统的失效, 则称它们处于可靠性串联, 若 N 个元件全部失效, 才会引起整个系统的失效, 则称它们处于可靠性并联。对于可靠性串联, 整机的 MTBF 与元器件的失效率及元器件数 N 的关系为

$$\text{MTBF} = 1 / \sum_{i=1}^N \lambda_i \quad (19.3)$$

对于可靠性并联, 则有

$$\text{MTBF} = \frac{1}{\lambda} \sum_{i=1}^n \frac{1}{i} \quad (19.4)$$

大多数民用产品采用可靠性串联结构, 除了选用失效率低的器件外, 应尽量采用集成度高的器件, 以减少器件总数。对于重要的系统、部件和电路, 可采用可靠性并联结构设计或采用两种结构混合的设计方法。提高可靠性的其他结构设计方法还有冗余、旁待(备份)等。

改进基本电路设计, 合理选取元件的工作参数, 对易损器件(半导体器件等)采取保护措施, 对提高可靠性是必要的。工艺结构设计对提高可靠性也很重要, 以下是一些要注意的问题。

(1) 隔振和缓冲

电子设备在运输、发射或以外跌落等过程中将承受不同程度的振动与冲击力,这对结构设计和大质量元件的安装提出了专门的隔振与缓冲的要求。应尽量避免采用悬臂式和容易引起引力集中的结构形式。结构件尽可能选用屈服强度、极限强度及延伸率较高的塑料材料。刚度较大的隔振器缓冲效果好,刚度小的隔振器的隔振性能好,常用的有橡胶隔振器和金属弹簧隔振器。

(2) 腐蚀与防护设计

电子设备与大气、土壤、海水等接触,都将产生不同程度的腐蚀。特别是长期处于盐雾、酸雾、高温下工作的电子设备及在野外、地下长年工作的无人值守的电子设备,其电子元件和结构都应考虑采取防护设计,通常采用表面涂覆、镀层、发蓝、发黑、防锈油、浸渍、密封、灌封等方法。

(3) 防尘、防爆

电子设备中的开关、继电器、电刷等带触点部分的元件,染尘后将影响性能,这些部位的工作过程中常引起电强火花,在矿井中和有易燃气体的场所将引起火灾、爆炸,对应用在这些特殊环境中的电子设备,不应采用上述元件来设计电路,并且通常对整机做全封闭处理。

19.5 数字电路的可测试性设计

可测试性设计,就是以改善逻辑电路的可测性、易测性和可诊断性为目标的设计。

对于规模较小、PCB 走线密度不高的电路系统,可通过探针或设计专用的针床(位置与被测点相互对齐的一组探针)对电路实行在线的功能性测试,模拟电路各部分信号流向往往是很清楚的,可逐步测试各点信号。对于复杂的数字系统,电路的调试往往成为大的问题,甚至严重到无法测试和无法对故障进行检测修理。因此在数字电路及系统的设计阶段,更应考虑测试、调试和维修的方便,即应具有可测性或易测性。

以下的方法与经验有助于改善电路的可测性。

(1) 使用针床。针床是一组探床,其位置与被测 PCB 上的测点——对应,可用来输入多个数据和获得多个输出数据;

(2) 使用多路器,即多路分配器和多路选择器,它们可以由 N 个控制信号经译码,实现 2^N 个节点的选择;

(3) 使用串行移位寄存器,将欲增加的控制点和观测点接至附加的寄存器的输入和输出上,这些寄存器接成串行移位形式,这样通过很少的数据线和控制线,可以移入、移出大量的激励和响应数据;

(4) 对闭环回路预留开断跳线点;

(5) 应尽量将一个大电路分成若干个小电路,以减少总的测试矢量数(测试码以电路规模的平方和立方增加);

(6) 少用异步时序电路,少用单稳态电路;

(7) 少用可调元件和少用非规范元件;

(8) 设置状态复位功能;

(9) 不同逻辑阈值电子电路分区排版,数字电路与模拟电路分区排版。

随着电路规模的增大、VLSI 电路的大量运用、多层 PCB 和表面贴装(SMD)技术的广泛应用,电路板走线密度越来越高,通过增加测点来改善电路的可测性已很困难。而随着电路速度向数 GHz 迈进,传统的时域和数据域测量已几乎不可能。于是提出了各种可测性设计技术,如内建自测试技术、边界扫描测试技术(BST)等。

边界扫描测试已成为很受欢迎的测试方法，已由“联合测试行动小组”制定了 JTAG 标准，并由 IEEE 宣布为 IEEE1149.1——“测试访问口及边界扫描设计”工业标准。

图19.9是一个在核心逻辑的每个输入、输出端增加了一个边界扫描(BS)单元，以构成具有边界扫描测试功能的 IC 电路，各个 BS 内部的寄存器串联成移位寄存器形式构成一条边界扫描通路，各个 BS 单元既与芯片内核心逻辑相连，又经缓冲器与外部引脚相连。当这些 IC 安装在 PCB 上时，可通过这些 BS 单元获取芯片引脚外 PCB 上铜线的信息。在片内边界扫描控制电路的控制下，编好的测试数据从 TDI 移入，从 TDO 获取测试结果，根据移入、移出的数据实现对 PCB 故障和芯片逻辑功能的检查。其中，TCK 为时钟，TMS 为方式选择线。尽管这种方法要增加 BS 电路，但对于 VLSI 或大规模电路系统，这种额外开销相对是很少的。这种设计仅需一个四总线的测试访问口(还有一个可选线为测试复位 TRST)，比起需要大量增加测点的方法这是十分重要的优点。

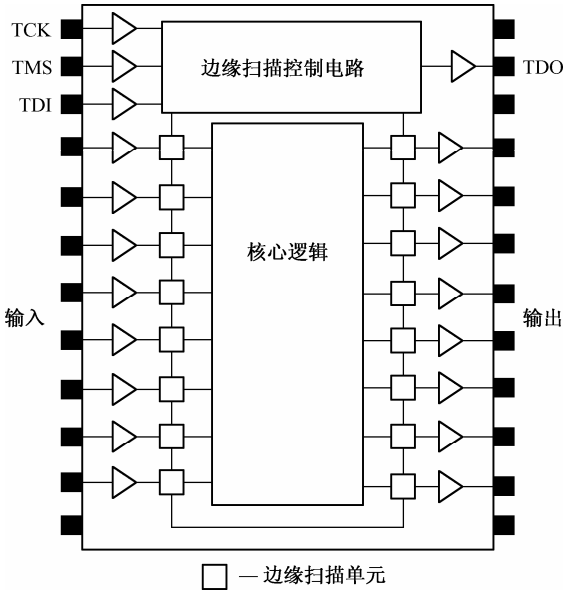


图 19.9 具有边界扫描测试功能的 IC 电路内部结构

边界扫描测试技术能用于器件级、电路板级和系统级的可测性设计。图19.10 为由三片带 BST 功能的 IC 构成的 PCB，各 IC 的 TDO 与 TDI 相串联。测试访问口通过 4(或 5)根线与 PCB 上所有 IC 相连，实现边界扫描测试。其中 TCK 和 TMS 并行送到各个 IC (图19.10中未画出)。对 PCB 电路的测试包括两个方面。

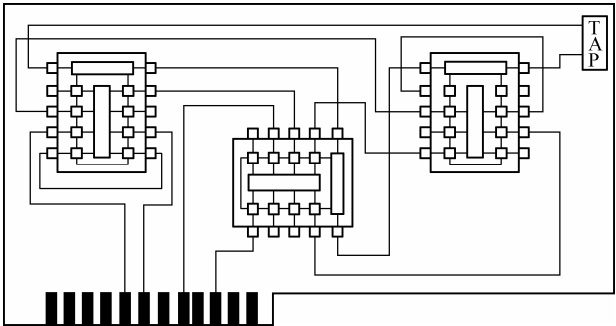


图 19.10 在 PCB 中一个完整的边界扫描链

(1) PCB 电路连线的测试

连接于两个 IC 的 I/O 引脚之间的 PCB 走线是否不通, 可以通过 IC 中的 BS 单元的输出(发端)与输入(收端)信号进行检测, 这种测试又称为(IC 芯片的)外部测试。从芯片输出端的 BS 单元移出的激励数据, 经 PCB 铜线传到另一个输入端的 BS 单元读出, 即可判断铜线的通断。由于铜线与 BS 单元之间还有焊点和 IC 内部从管芯到引脚之间的连线两个环节, 所以被测结果也包含了可能的虚焊与管内引线由于机械或焊接而断开的信息。

(2) 对接入通路的所有 IC 进行功能测试

这时, 激励数据通过 BS 施加到 IC 的输入端, 而从输出端捕获响应数据, 这与 PCB 引线测试情况相反。这种测试称为内部测试。

(3) 测试非扫描器件

对于非扫描器件组成的电路, 若其输入、输出与边界扫描器件相连, 则可利用边界扫描器件的 BS 单元构成“虚拟通道”, 用做与之相联的输入端的激励并对输出端观察。

很多可编程器件(PLD)在编程后, 接着就是进行测试, 也采用了边界扫描测试, 与 IEEE1149.1 标准兼容。这样电路编程和测试使用的是同一接口, 这给电路系统的设计带来了很大方便 [对于编程功能, 还要一个“编程禁止”信号, 因而这是一个五线接口(当选 TRST 时为六线)]。

19.6 印制电路板(PCB)的设计与装配

电子系统大多采用印制电路板(PCB)结构, 在系统的实现过程中, PCB 的设计、装配和调试占据了很大的工作量。

19.6.1 PCB 的设计

PCB 设计是系统实现的重要环节, 在很大程度上影响产品的性能和整个设备的综合指标。

1. 设计工具

PCB 的专业设计工具很多, Protel 就是其中的一种, 可用它来制作电路原理图, 自动进行印制电路板设计, 但所生成的布线质量往往不能令人满意, 需人工修改直到获得满意效果。

2. 经验与技巧

PCB 设计是一个很具有工艺性、技巧性的工作, 设计者的经验对设计质量及对此后的装配、调试、生产与维修等都将起着重要作用。

下面提到的问题对缺乏经验的初学者来说值得注意。

(1) 元件布局要注意相互位置与疏密, 模拟电路与数字电路尽可能分开。

(2) 设计电路规模较大的双面板时, 尽量遵循印刷板的两面走线各取一个方向, 不要在同一面混用两种方向的走线。

(3) 预留测试杆位置, 以便调试时方便。对闭环电路, 预留开环断口或跳线插口座。

(4) 当一块板上有多个与外界连接的电缆线插头、插座时, 应采用不同形式、不同芯数的连接插头插座, 以免相互错位连接。

(5) 每个 IC 旁留有电源滤波电容安装空间, 以做电源去耦用。

(6) 注意印刷板中长距离并行引线的分布电容的影响。例如, 可在并行走线的总线之间加入地线起隔离作用。

- (7) 印刷板上的大面积地线可起到屏蔽作用,但应注意对信号引线所构成的分布电容参数的影响。
- (8) 对需要热拔插的插头座,可根据各芯片先后接通的顺序,将插针设计成不同的长度。

3. 信号完整性设计

在现代电子设计领域,由 IC 芯片构成的电子系统是朝着大规模、小体积、高速度的方向飞速发展的。这也给电子系统设计带来了新问题,即电子设计的体积减小导致电路的布局布线密度变大,而同时信号的频率还在提高,从而使得如何处理高速信号问题成为电子系统设计能否成功的关键因素。因此,对于一个相对复杂的电子系统,应尽可能使所完成的设计符合国家的 EMC 标准,在整个系统 PCB 设计的过程中仔细考虑电磁干扰(EMI)和信号完整性(Signal Integrity, SI)设计。具体设计细节阐述如下:

(1) 采用多层 PCB 结构,应仔细评估系统设计的复杂程度及产品要求等,以确定 PCB 的层数。

(2) 在电源模块中接磁珠来吸收 EMI,并在合适的位置如芯片供电引脚底部放置去耦电容,最后在 PCB 的顶层和底层大面积铺铜,并用较多的过孔将这些地平面连接在一起。此外,系统时钟线也是 EMI 问题的最大来源,因此在设计中尽可能把它放在电源层上。

(3) 考虑到在系统中存在着很多固有的噪声源,而系统 SI 的好坏又很大程度上决定了系统的抗干扰能力,在系统设计时必须考虑去除产生噪声的电路。虽然许多潜在的噪声源无法在设计上根本消除,但只要在设计过程中采取适当的方法,如在噪声源或耦合电路中,先将噪声尽量减小,设计出最佳的滤波和去耦布局。对于高速系统,在系统设计中必须要考虑互连延迟引起的时序问题及串扰、传输线效应等信号完整性问题。差的信号完整性不是由某一单一因素导致的,而是由板级设计中多种因素共同引起的,主要的信号完整性问题包括反射、时延、振荡、地弹、串扰等。通常情况下,在一个已有的 PCB 上分析和发现信号完整性问题是一件非常困难的事情,即使找到了问题,在一个已成形的电路板上实施有效的解决办法也会花费大量时间和费用。所以,设计时应从能够避免影响信号完整性因素出发,采用屏蔽和阻抗匹配,地址线 and 数据线采用等间距等线宽布置,尽量增大信号线间距、减少并行信号长度等一系列措施,最主要的是预先使用 EDA 信号完整性工具仿真实际物理设计中的各种参数,对电路中的信号完整性问题进行深入细致的分析。这样就能够在物理设计完成之前查找、发现并在电路板设计过程中消除或减小影响信号完整性的因素。

与传统的 PCB 设计方法相比,基于信号完整性分析的设计方法具有以下特点。

- 在 PCB 设计之前,首先建立高速信号传输的信号完整性模型。根据 SI 模型对信号完整性问题进行一系列的预分析,根据仿真计算的结果选择合适的元器件类型、参数和电路拓扑结构,作为电路设计的依据。
- 在电路的设计过程中,将设计方案送交 SI 模型进行信号完整性分析,并综合元器件和 PCB 参数的公差范围、PCB 版图设计中可能的拓扑结构和参数变化等因素,计算分析设计方案的解空间。
- 在电路设计完成后,各高速信号应该都具有一个连续的、可实现的解空间,即当 PCB 及元器件参数在一定的范围内变化、元器件在 PCB 上的布局及信号线在 PCB 上的布线方式具有一定的灵活性的情况下,仍然能够保证对信号完整性的要求。
- PCB 版图设计开始之前,将获得的各信号解空间的边界值作为版图设计的约束条件,以此作为 PCB 版图布局、布线的依据。在 PCB 版图设计过程中,将部分完成或全部完成的设计送回 SI 模型进行设计后的信号完整性分析,以确认实际的版图设计是否符合预计的信号完整性要求。若仿真结果不能满足要求,则需修改版图设计甚至电路设计,这样可以降低因设计不当而导致产品失败的风险。在 PCB 设计完成后,就可以进行 PCB 制作。PCB 制造参数的公差范围应在信号完整性分析的解空间的范围之内。

- 当 PCB 制造好后,再用仪器进行测量调试,以验证 SI 模型及 SI 分析的正确性,并以此作为修正模型的依据。在 SI 模型及分析方法正确的基础上,通常 PCB 不需要或只需要很少的重复修改设计及制作就能够最终定稿,从而可以缩短产品开发周期,降低开发成本。

(4) 对于系统带外噪声(分为输入带外噪声和输出带外噪声,前者由 A/D 变换前引入,后者由 D/A 变换后将系统内部噪声信号传输出去),这些噪声主要是系统内部的晶体振荡噪声及其谐波噪声,还包括系统内部 SDRAM 模块高速转换电路产生的噪声。消除这些噪声影响的根本办法是遵循上面提到的系统内部噪声处理方法,从 SI 和电源完整性两方面入手降低系统 EMI 水平。系统内部辐射水平降低后,耦合到模拟输出信号上的杂波信号电平也会随之降低,最后在使用低通滤波器对输出信号进行滤波,可以进一步对系统带外噪声进行衰减。消除输入带外干扰噪声显得更加重要,技术也更复杂。为了消除带外噪声的影响,同时又保证信号指标不受影响,可采用的另一做法是提高信号的取样频率,这样可以降低对信号输入端低通滤波器的设计要求。

已有很多的 EDA 工具用来进行 PCB 的信号完整性分析,如广泛使用的电路原理图制作和 PCB 版图制作软件 Protel DXP 就能够在 PCB 制作过程中进行 SI 的详细分析,有关具体的分析操作可以参考软件的帮助文档,这里不再赘述。

19.6.2 PCB的装配与焊接

虚焊、短路、开路是印刷板装配焊接中常遇到的故障问题。其中元件引脚的氧化、电路板焊接部分和连接部分的涂覆层质量及焊接工具、焊接温度和时间的掌握是关键。经验表明,应慎用助焊剂,特别是带有腐蚀性的助焊剂。

为避免焊接造成断路、开路等难以修复的故障,应在焊接之前对印刷板进行全面的检查。焊接过程对元件的损伤主要表现为

- (1) 半导体器件因受热太久导致损坏,应按安全要求掌握焊接温度与时间;
- (2) MOS 型器件因输入阻抗高,要求焊接工具无感应静电或将烙铁外壳接地;
- (3) 某些不耐高热的材料,如带有塑料构件的电子元件,不应在加热时引起变形,有时应采用扣接;
- (4) 焊接大电流的端子,可能因发热而熔脱,应改用绕焊、钩焊;
- (5) 对引线密度很高的大规模芯片和有特别要求的芯片,应使用专门的焊接设备和遵守专门的操作流程。

产品进行工业化批量生产时,常采用的焊接装配方式有:浸焊、波峰焊、再流焊、高频加热焊、激光焊等,了解这些焊接工艺,有助于更加合理地设计印制电路板。

19.7 电子系统的调试

一个新设计的装配好的电路,在通电之前,必须认真检查,特别应排除那些危及安全和可能导致损害元件与电路的问题。

19.7.1 通电调试之前的检查

重点应放在整体性、全局性连线的错误排查,如电源线的短路、错接等。通常采用万用表的欧姆挡对上述引线逐一检查其对地电阻。注意,由于半导体元件的存在,正、反向呈现不同的阻值。容易错误焊接的元件应仔细检查,如带极性的元件、不带定向标志的连接元件是否方向装反等。

19.7.2 调试的一般顺序与步骤

要求调试者对电路功能和性能指标有全面的了解,对常用的测量仪器功能和操作使用有一定的了解。在研制阶段,对由多级电路或多个功能模块构成的电路系统,应采用逐级逐块地安装、逐步调试的方法,而不宜先将电路全部插上或焊上,这样可避免因电路存在的潜在反馈渠道而使调试复杂。

19.7.3 做好调试记录

调试人员的工作经验十分重要,做好调试记录,培养良好的科研工作习惯,积累调试经验。电路的种类、功能、要测试的指标很多,很难对电路的调试提出一个固定的步骤和模式。以下就模拟电路、数字电路及带微处理器系统的软件调试中所遇到的带有共性的问题进行一般的介绍。

19.7.4 模拟电路的调试

(1) 晶体管电路若不正常工作,首先断开级联与反馈,检查工作点。

(2) 运放电路着重检查差分输入端电位。排除自激故障,可断开反馈,逐级将输入端交流短路接地,针对自激原因,采用旁路电容、负反馈等措施改善之。

(3) 模拟电路的输入阻抗的测量:采用电阻分压法测量(参见图19.11)。调节 R_{ref} ,使得 T 点所测得的信号为输入信号的 $1/2$,此时的 R_{ref} 即为输入阻抗值。这种方法只适用于低频电路。

(4) 输入短路等效噪声测量(参见图19.12)。将输入接地,测得放大器的输出有效值为 v_o ,放大器的短路噪声为 v_o/A ,其中 A 为放大器增益。

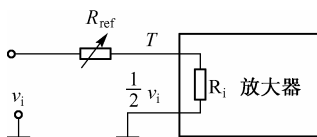


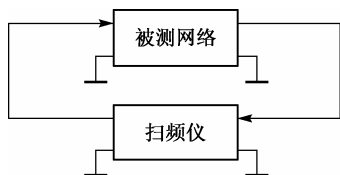
图 19.11 输入阻抗的电阻分压法测量



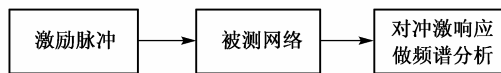
图 19.12 输入短路等效噪声测量

(5) 网络频率响应特性的测量:用扫频仪测量幅频特性,用矢量网络分析仪或信号源协同信号分析仪测试其幅频与相频特性(参见图19.13)。

(6) 失真度测量(参见图19.14)。可采用失真度仪直接测量,通常所测为总失真加噪声,即 $\text{THD} + N$ 。可由频谱分析仪分析失真分量的谐波结构,由于先经失真度仪滤去了基频分量,可充分利用频谱仪的测量动态范围。

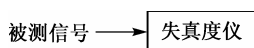


(a) 扫频法测试

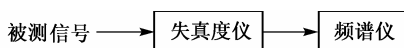


(b) 动态测试法

图 19.13 网络频率响应特性的测量



(a)



(b)

图 19.14 失真度测量

在组建测试环境时,应注意测试用的仪器对被测电路的影响,如引入噪声、改变其输入、输出阻抗等。由于模拟电路的信号流向较为简单清晰,逐级跟踪控制调试即可,常遇到的难点问题有排除自激和降低系统的噪声等。

噪声往往是接收机的前置放大器和高阻传感器的前置放大器的重要指标。噪声来源主要有外来电磁干扰(对于高阻输入端而言)和电路系统内部器件的工作噪声。前者主要靠改善电磁兼容工艺,提高其抗干扰性能来改善;后者要靠选用低噪声器件来改善。噪声问题还可以从整机系统的角度采用信号处理的方法改善。

19.7.5 数字电路系统的调试

对以中小规模数字集成电路搭成的系统,用万用表、逻辑笔和示波器等常规仪器即可完成调试。对大规模的复杂时序逻辑系统,可采用逻辑分析仪、特征分析的方法大大提高测试效率。

影响电路正常工作的常见问题有:①因电路延时或逻辑设计带来的竞争冒险;②组合逻辑电路输出中的“毛刺”脉冲;③负载过重而导致的 TTL 电平偏差。

上述现象在计算机仿真或实物调试时都能遇到。在仿真时出现的问题,可修改逻辑设计来解决。例如,当带有尖刺的组合逻辑输出信号作用于其后的电路时,将影响后段电路的正常工作。如果在设计时让该组合信号预先提前 $1/2$ 拍,然后再经 D 触发器延迟 $1/2$ 拍锁存,就能避开毛刺,获得良好的输出波形。

由于毛刺是一种很窄的脉冲,大多数情况下,在信号经过的路径上加一适当大小的对地电容滤波,即可使毛刺的幅度大大下降,不致形成危害,而有用的信号脉冲和电平并不受影响。滤除毛刺用的电容值的大小通常需在调试中用凑试法得到。

数字系统由于信号的逻辑复杂,因此需测试的输入、输出测点多。常用的数据域测试仪器有逻辑分析仪、特征分析仪等。可将数字电路系统分成两大类:一类为数字信号处理(DSP)系统,它可对模拟量的数字化运算,其中各数据流节点上的数字信号可通过 D/A 转换成模拟电量来观察;另一类是数字逻辑系统,各节点信号并无相应大小的模拟值与之对应。对于 DSP 系统调试时,可以用模拟信号经 A/D 转换后的数字量信号作为输入激励,再对运算后的各点数字量信号进行 D/A 转换,进行信号的模拟波分析,如同用示波器调试模拟电路过程一样。面对后者数字逻辑系统,常用特征分析仪来检查各节点数据信号以调试或故障诊断。当以特定的 PRBS(伪随机二进制序列)发生器作为特征激励输入时,一个可测试系统的各节点将产生稳定、专一的数据流,各点的特征数据流经压缩(除一个未知数所得余数——一个短的序列)后,将它与该点的参考值比较即可判定系统工作是否正常。可见,被测点的特征值并无物理意义,使用者也可在不知道被测系统的工作原理时,只根据设计所确定的该点特征值进行调试和故障分析。

图 19.15 为带微机电路系统的调试示意图。用 PLD 设计系统时,在电路中附加了多路选择器,可在片外 N 根选择地址线控制下,对 2^N 个节点信号进行观察,所选节点信号经 D/A 转换后即可用做 DSP 系统的调试。

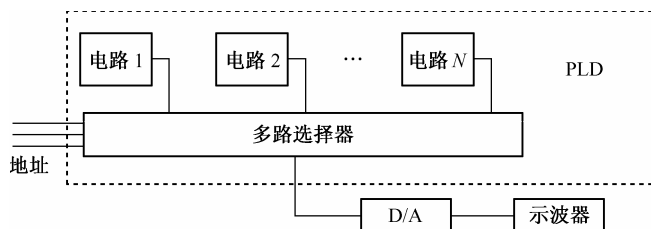


图 19.15 带微机电路系统的调试

19.7.6 带微处理器系统的软件调试

内带微处理器的电子系统,除了要进行硬件电路调试外,还应进行软件程序的调试。这类系统的调试过程为:

(1) 由 PC 和仿真器组成一套开发系统。

(2) 仿真器通过仿真头与目标板相连。通常仿真器可通过配用不同的仿真头或不同的软件环境配置来实现对不同型号的 CPU 仿真。所选适配头应与目标板所用的 CPU 类型相符。

(3) 在 PC 上运行与所用 CPU 相对应的开发系统软件。

开发功能的软件部分包括有监控程序、调试程序、编辑程序及仿真器的工作软件。通过单步运行,在程序的某处设立断点等方法,很容易找到程序中的问题。

大多数开发系统都支持用汇编语言、PL/M 语言及 C 语言进行开发,后两种皆为高级语言。其中,PL/M (Programming Language for Microcomputer) 语言是 Intel 公司为微型机的程序设计而开发的标准语言;使用汇编语言可节省程序空间容量,对单片机系统而言,这是很重要的;当涉及大量的数学运算时,采用高级语言可提高编程效率,开发者可根据具体情况选择使用。

组建好的开发系统在调试过程中,常遇到的故障现象有:

(1) 仿真头与目标板接触不良,必要时,应对簧片和插针逐一检查。

(2) 仿真头的负载能力不够,带不动目标板上的负载(多见于数据总线的负载太重),可将目标板上的有关元件拔去一部分,以减轻负载。

(3) 仿真器的工作速度达不到目标板的设计主频速度,此时应将降低时钟运行频率,可将目标板的工作主频切换到仿真器一边。

对于实现简单功能的小系统的软件开发,也可无须在线仿真,而仅在 PC 上离线情况下进行软件调试,再将调好的软件烧录到 CPU 片内存储器和/或外部 ROM 中。

19.8 本章小结

本章介绍设计模拟、数字及智能化电子系统时涉及到的有关工程实现方面问题,如抗电磁干扰、电磁兼容性、热特性、可测性、可靠性、系统调试方法及其他一些有特殊要求的电子系统在实现时应考虑的问题,既强调了工程设计理念,同时对电子系统实现过程中的技术与工艺问题进行了讨论。希望读者通过本章的学习,能够从工程化的角度去认识在电子系统实现过程中可能存在的问题和解决的方法。

参考文献

- [1] 田良. 综合电子设计与实践. 南京: 东南大学出版社, 2002.
- [2] 陆应华. 电子系统设计教程. 北京: 国防工业出版社, 2005.
- [3] 何小艇. 电子系统设计. 杭州: 浙江大学出版社, 2001.
- [4] 李朝清. 单片机原理及接口. 北京: 北京航空航天大学出版社, 2003.
- [5] 马忠梅. 单片机的 C 语言应用程序设计. 北京: 北京航空航天大学出版社, 1998.
- [6] 谢自美. 电子线路设计·实验·测试. 武汉: 华中科技大学出版社, 2000.
- [7] 全国大学生电子设计竞赛组委会. 第一届(1994)~第七届(2005)全国大学生电子设计竞赛题目[G/OL].
www.nuedc.com.cn, 2005.
- [8] 张永瑞. 电子测量技术基础. 西安: 西安电子科技大学出版社, 1994.
- [9] 古天祥. 电子测量原理. 北京: 机械工业出版社, 2004.
- [10] 高吉祥. 全国大学生电子设计竞赛培训系列教程. 北京: 电子工业出版社, 2007.
- [11] 黄智伟. 全国大学生电子设计竞赛系统设计: 数字系统与自动控制系统设计. 北京: 北京航空航天大学出版社, 2007.
- [12] 徐江丰. 相关计数法数字频率计的研究与实现. 电子技术, 2003, 30(4): 16~19.
- [13] 王皓. 低频相位测量系统的研究与实现. 电子技术, 2004, 31(9): 22~25.
- [14] 大学生电子设计竞赛湖北赛区组委会. 电子系统设计实践. 湖北: 华中科技大学出版社, 2005.
- [15] 黄智伟. 锁相环与频率合成器电路设计. 西安: 西安电子科技大学出版社, 2008.
- [16] 文卓然. 数字移相信号发生器设计中的一个误区. 电子世界, 2004, 5: 42~43.
- [17] 马明建. 数据采集与处理技术. 西安: 西安交通大学出版社, 1998.
- [18] 李小青. 高精度数据采集与回放系统的设计与实现. 电子技术, 2004, 31(7): 11~14.
- [19] 樊昌信. 通信原理. 北京: 国防工业出版社, 2001.
- [20] 金以慧. 过程控制. 北京: 清华大学出版社, 1993.
- [21] 张翼飞. 时滞系统控制发展历程综述. 控制工程, 2004, 11: 4~8.
- [22] 郑轲. 时滞过程控制算法分析. 北京科技大学学报, 2000, 22(5): 486~488.
- [23] 刘川来. 一类时滞系统的模型参考自适应控制. 仪器仪表学报, 1998, 4(19): 393~398.
- [24] 刘希远. 时变大时滞极点配置最优预报自校正 PID 控制器. 控制与决策, 1990, 5(5): 19~23.
- [25] 陈曦. 基于 CPLD 的温度自动控制系统的研制. 电子技术, 2003, 30(5): 7~9.
- [26] 王金明. 数字系统设计与 Verilog HDL. 北京: 电子工业出版社, 2002.
- [27] 陈赓. CPLD/FPGA 与 ASIC 设计实践教程. 北京: 科学出版社, 2005.
- [28] 夏宇闻. 复杂数字电路与系统的 Verilog HDL 设计技术. 北京: 北京航空航天大学出版社, 2002.
- [29] 黄智伟. FPGA 系统设计与实践. 北京: 电子工业出版社, 2007.
- [30] 仁爱锋. 基于 FPGA 的嵌入式系统设计. 西安: 西安电子科技大学出版社, 2005.
- [31] 李小青. 用 FPGA 设计大屏幕 LED 显示屏. 电子技术, 2005.
- [32] 萧世文. USB2.0 硬件设计(第2版). 北京: 清华大学出版社, 2006.
- [33] 廖济林. USB2.0 应用系统开发实例精讲. 北京: 电子工业出版社, 2006.
- [34] 田耘. 无线通信 FPGA 设计. 北京: 电子工业出版社, 2008.
- [35] 陈赓. CPLD/FPGA 与 ASIC 设计实践教程. 北京: 科学出版社, 2005.

- [36] 金朝辉. 采用 CY7C68013 芯片的 USB2.0 系统固件程序设计. 世界电子元器件. 2006. 7: 55~57.
- [37] 郑君里. 信号与系统(第二版)下册. 北京: 高等教育出版社, 2005.
- [38] 郑君里. 信号与系统(第二版)上册. 北京: 高等教育出版社, 2005.
- [39] Uwe Meyer-Baese 著, 刘凌译. 数字信号处理的 FPGA 实现. 北京: 清华大学出版社, 2002.
- [40] 程佩青. 数字信号处理教程(第二版). 北京: 清华大学出版社, 2002.
- [41] Keshab K. Parhi 著, 陈弘毅等译. VLSI 数字信号处理系统设计与实现. 北京: 机械工业出版社, 2006.
- [42] 王卫兵. Goertzel 算法的一种改进计算结构. 武汉大学学报, 2007, 53(3): 375~378.
- [43] 任淑艳. 应用 VHDL 语言的 FFT 算法实现. 哈尔滨理工大学学报, 2003, 8(6): 24~26.
- [44] 刘小明. 基于 FPGA 的基-8 FFT 模块的实现. 合肥工业大学学报, 2006, 29(5): 536~539.
- [45] 荣瑜. 一种高性能 FFT 蝶形运算单元的设计. 东南大学学报, 2007, 37(4): 565~568.
- [46] 王旭东. 一种新结构 FFT 算法及其 FPGA 实现. 无线通信技术, 2005, 14(3): 46~49.
- [47] 刘嘉新. 用 FPGA 实现快速傅里叶变换. 信息技术, 2006, 30(2): 57~60.
- [48] Roland E. Best 著, 李永明等译. 锁相环设计、仿真与应用(第5版). 北京: 清华大学出版社, 2007.
- [49] 邵帅. 基于 FPGA 的全数字锁相环性能改进的设计. 云南师范大学学报, 2008, 28(2): 37~39.
- [50] 单长虹. 基于 FPGA 的全数字锁相环的设计. 电子技术应用, 2001, (9): 58~59.
- [51] 蒲晓婷. 全数字锁相环的设计与分析. 现代电子技术, 2008, (5): 173~176.
- [52] 张益贞. Visual C++实现 MPEG/JPEG 编解码技术. 北京: 人民邮电出版社, 2002.
- [53] 张青贵. 人工神经网络导论. 北京: 中国水利水电出版社, 2004.
- [54] 阎平凡. 人工神经网络与模拟进化计算. 北京: 清华大学出版社, 2005.
- [55] 蒋宗礼. 人工神经网络导论. 北京: 高等教育出版社, 2001.
- [56] Simon Haykin 著, 叶世伟译. 神经网络原理(第2版). 北京: 机械工业出版社, 2004.
- [57] J. Volder. The CORDIC trigonometric computing technique. IRE Trans. on Electronic Computers, 1959, 330~334.
- [58] Wang, S. Hybrid cordic algorithms. IEEE Trans. on Computers, 1997, 46(11): 1202~1207.
- [59] Uwe Meyer-Baese 著, 刘凌译. 数字信号处理的 FPGA 实现(第2版). 北京: 清华大学出版社, 2003.
- [60] J. Walther. A unified algorithm for elementary functions. Spring Joint Computer Conf. Proc. 1971, 379~385.
- [61] Myers, D. J. Efficient implementation of piecewise linear activation function for digital VLSI neural networks. Electronics Letters, 1989, 25(24): 1662~1663.
- [62] Alippi, C. Simple approximation of sigmoid functions: realistic design of digital VLSI neural networks. Proceedings of IEEE Int Symp. Circuits and Systems. 1991: 1505~1508.
- [63] Amin, H. Piecewise linear approximation applied to nonlinear function of a neural network. IEE Proceedings Circuits, Devices and Systems, 1997, 144(6): 313~317.
- [64] Basterretxea K. Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons. IEE Proceedings of Circuits, Devices and Systems, 2004, 151(1): 18~24.
- [65] 陈曦. 高速集成电路中神经网络建模技术的研究与应用. 武汉: 武汉大学, 2007.
- [66] 邹彦. DSP 原理及应用. 北京: 电子工业出版社, 2007.
- [67] 彭启琮. DSP 集成开发环境——CCS 及 DSP/BIOS 的原理与应用. 北京: 电子工业出版社, 2004.
- [68] 汪春梅. TMS320C5000 系列 DSP 系统设计与开发实例. 北京: 电子工业出版社, 2004.
- [69] Jean J. Labrosse 著, 邵贝贝译. μ C/OS-II——源码公开的实时嵌入式操作系统. 北京: 中国电力出版社, 2001.
- [70] 任哲. 嵌入式操作系统基础 μ C/OS-II 和 Linux. 北京: 北京航空航天大学出版社, 2006.
- [71] 邬可军. DSP 实时多任务操作系统设计与实现. 北京: 电子工业出版社, 2005.

-
- [72] 孙琮. 嵌入式 Linux 应用程序开发详解. 北京: 人民邮电出版社, 2008.
- [73] 周维. 基于 S3C2410 的 ARM 开发平台. 电子技术, 2004, 31(6): 4~7.
- [74] 何明聪. 基于 ARM920T 微处理器的 IDE 硬盘接口设计与实现. 计算机工程与设计, 2005, 26(03): 768~769.
- [75] ARM Limited. ARM architecture reference manual [EB]. 1996—2000.
- [76] Samsung Electronics. Application note S3C2410X 32-Bit RISC microprocessor revision 1 [EB]. 2002.
- [77] 毛德操. Linux 内核源代码情景分析. 杭州: 浙江大学出版社, 2001.
- [78] 郭玉东. Linux 操作系统结构分析. 西安: 西安电子科技大学出版社, 2002.