分类号_____ 密级 _____

UDC _____

学 位 论 文

基于 SOPC 的 MP3 编解码器的设计与实现

作 者 姓 名 ： 尚淼洪

指 导 教 师 ： 李晶皎 教授

东北大学信息科学与工程学院

申请学位级别： 硕士 学 科 类 别 ： 工学

学科专业名称： 计算机软件与理论

论文提交日期： 2008 年 6 月 论 文 答 辩 日 期 ： 2008 年 6 月

学位授予日期： 2008 年 7 月 答辩委员会主席： 高福祥

评 阅 人 ：

东 北 大 学

2008 年 6 月

A Thesis for the Degree of Master in Computer Software and Theory

Design and Implementation of MP3 codec based on SOPC

by Shang Miaohong

Supervisor: Professor Li Jingjiao

Northeastern University

June 2008

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：

日　　期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年□　　一年□　　一年半□　　两年□

学位论文作者签名：　　导师签名：

签字日期：　　签字日期：

基于 SOPC 的 MP3 编解码器的设计与实现

摘 要

目前国内外针对 MP3 编解码器的研究主要着重基于单片机的音频编解码，这样的系统有局限性，主要是扩展功能差，功耗大，很多主流功能都不兼容。在应用需求的牵引和 IT 技术的推动下，基于 SOPC 系统的开发在音频编解码处理领域的应用也日益增多。本课题的研究目标是设计出一个基于 SOPC 平台的 MP3 编解码器。

本文针对 SOPC 系统的特点和现实中的应用需求，研究实现了一套基于 Altera 的 CycloneII 2C35 处理器的 MP3 音频编解码器，并且可以挂载 USB 设备。以 CycloneII 2C35 为核心，通过开发音频处理的 SOPC 构件，将 MIC 采集的音频数据传到 NiosII 软核处理器上，再通过 μClinux 操作系统下音频编码程序进行编码，形成 MP3 文件存储在 USB 设备上，解码程序实现了 MP3 数字音频信息转化为 PCM 数据。该系统扩展性强，应用广泛，移植了 μClinux 操作系统，可以方便地进行后续开发增加用户所需功能。从而提高了设备的智能程度，具有一定的工程实用价值。

软件编解码而非硬件

本文从软件和硬件两方面实现了 SOPC 系统开发过程的相关模块。系统硬件控制平台的研究主要包括：基于 NiosII 软核处理器的 SOPC 系统的结构，音频处理模块的功能和结构，以及 DE2 开发板的应用要求；系统软件运行平台的研究主要包括：μClinux 操作系统介绍，MP3 编解码器的设计，以及 μClinux 环境下编解码程序的移植等工作。介绍了 Avalon 总线的结构，组成硬件平台的 NiosII 系统组成模块，MP3 编解码算法的原理等基于 SOPC 的 MP3 编解码器开发中所使用的主要相关技术。最后给出测试结果和列出了遇到问题的解决方案。

实验表明，该系统编解码准确，可操作性强，能满足系统的基本要求，而且移植了 μClinux 操作系统，可以方便地进行后续开发增加用户所需功能，具有广泛的应用前景。

关键词：SOPC；MP3 编解码器；Nios 软核；μClinux；DE2

Design and Implementation of MP3 codec based on SOPC

Abstract

At present, the research of MP3 codec is focused mainly on audio codec on MCU about at home and abroad. The systems have limitations. The main defects are the poor extended function, high loss of power and non-compatible with popular function. With the fast development of the application needs and IT, the research based on SOPC system grows fast in the fields of audio codec. The aim of the thesis is to develop an MP3 codec based on the SOPC platform.

In this paper, as the characteristics and application requirements in reality of SOPC system are considered, a set of MP3 codec based on Altera`s CycloneII 2C35 processor is developed, which may mount USB device. Taking the CycloneII 2C35 as the core, the audio processing component based on SOPC is developed, which transform the audio data collected by the MIC to NiosII soft-core processor, and then those data are coded into mp3 file which are stored in the USB device by audio coding program in the μClinux OS, and the transformation of MP3 digital audio information to PCM data is realized by the decoding program. The system has the strong expandability and wide application and μClinux is transplanted, which is convenient to continue subsequent development for adding user's needs. So it improves the intelligence degree of device and has the practical engineering value.

In this paper, the related modules of MP3 codec based on SOPC system are realized from two respects of software and hardware. The research of systematic hardware controlling platform mainly includes: the architecture of SOPC system based on NiosII soft-core processor, the function and structure of the audio process module, and the application requirements of DE2 development board; The research of systematic software operation platform mainly includes: the introduce of μClinux OS, the design of MP3 codec, and how to transplant codec program under the environment of μClinux etc. The paper introduced the structure of Avalon bus, NiosII system which constitutes hardware platform, and the MP3 algorithm principle, which are main related technology that are used in the development of MP3 codec based on SOPC. At last the paper gives the debuging results and lists the solutions of the problem.

The experiment results show that the developed system are accurate to code and decode the audio data, and are easy to be operated, and meet the basic system requirements. It has been transplanted with μClinux OS, which is easy to continue subsequent development for adding user's needs. So the system has a wide application prospect.

Key words: SOPC; MP3 codec; Nios soft-core; μClinux; DE2

目 录

独创性声明 I
摘 要 II
Abstract III
第 1 章 绪 论 1
1.1 课题研究背景 1
1.2 音频编码技术的发展及国内外现状 1
1.3 国内外 SOPC 在多媒体控制系统的应用现状 3
1.4 论文的组织结构 5
第 2 章 MP3 编解码器的相关技术介绍 7
2.1 MP3 标准简介 7
2.2 MP3 编码与解码算法的原理 8
2.2.1 MP3 编码算法原理 8
2.2.2 MP3 解码算法原理 9
2.3 基于 NiosII 软核的 SOPC 系统架构 10
2.3.1 简单的 SOPC 系统 11
2.3.2 NiosII 软核处理器 11
2.3.3 Avalon 总线 12
2.4 嵌入式 μClinux 操作系统 14
2.5 音频信号编/解码芯片 17
2.6 小结 18
第 3 章 MP3 编解码器的总体设计 19
3.1 MP3 编解码器的总体设计 19
3.2 MP3 编解码器的声音采样形式 20
3.3 MP3 编解码器的硬件环境 21
3.3.1 MP3 编解器的开发平台 21
3.3.2 QuartusII 和 SOPC Builder 22
3.4 MP3 编解码器的软件环境 24
3.5 小结 26
第 4 章 MP3 编解码器硬件的设计与实现 27
4.1 MP3 编解码器硬件的详细设计 27

4.1.1 编解码器的硬件结构.....27
4.1.2 Audio(音频处理)模块的硬件设计.....28
4.1.3 USB Flash 设备和接口.....33
4.2 基于 SOPC 的 MP3 编解码器系统硬件的实现.....33
4.2.1 NiosII 软件核处理器及控制模块.....33
4.2.2 SDRAM_PLL 模块.....35
4.2.3 Reset_Delays 模块.....35
4.2.4 I2C_AV_Config 模块.....35
4.2.5 FIFO 模块.....36
4.2.6 USB Flash 设备控制模块.....36
4.2.7 输入输出模块.....36
4.3 小结.....37
第 5 章 MP3 编解码器软件的设计与实现.....39
5.1 MP3 编解码器软件的详细设计.....39
5.1.1 μClinux 操作系统.....39
5.1.2 MP3 编码器软件设计.....40
5.1.3 MP3 解码器软件设计.....40
5.2 MP3 编码器软件的实现.....40
5.2.1 ShineFixed Point MP3 开源文件的说明.....41
5.2.2 MP3 编码器在 μClinux 上的实现.....41
5.3 MP3 解码器软件的实现.....44
5.3.1 MP3 解码软件的实现流程.....44
5.3.2 MP3 解码器在 μClinux 上的实现.....45
5.4 小结.....46
第 6 章 MP3 编解码器的测试与问题解决.....47
6.1 功能测试.....47
6.2 模块测试.....47
6.2.1 USB 存储设备模块测试.....47
6.2.2 MP3 编码模块测试.....48
6.2.3 MP3 解码模块测试.....48
6.3 遇到的问题及解决.....49
6.3.1 硬件——音频接口.....49
6.3.2 MP3 编码.....50

6.3.3 MP3 解码.....50
6.4 小结.....51
第 7 章 结束语.....53
参考文献.....55
致 谢.....59

第 1 章 绪 论

1.1 课题研究背景

随着科技进步，人们手中的家用电器不断地向小型化、多功能化发展，往往一个产品身兼多重任务。而在娱乐领域，人们的手中不再是那些单一的随身听、随身看，而发展成为可集媒体播放、数据交互采集甚至通信于一身的小型多媒体控制系统。这类系统可以自主进行编码解码、播放，同时拥有自己的处理器、存储器和多种对外通信接口。它可以装入实时操作系统，针对外界的触发条件完成相应工作，它还可以应用于工业控制或智能化的家用电器中。以往的此类播放器(如 MP3)主要是实现单一的功能，并且在板级实现，即将 CPU、DSP、外围接口这些独立的芯片集成在一块电路板上，体积很大，功能较少。而现今多媒体控制系统都在向着 SOPC 发展，即将 CPU、DSP、外围接口等数字逻辑集成在一块芯片内，同时与软件结合，成为一个小型的多功能片上系统[1]。相比以往，它们的功能更强，体积更小，功耗更低，可重构性强。

许多大型的国外厂商都相继推出了自己的多媒体控制 SOPC，像 PHILIP 公司的 Nexperia PNX5100 媒体处理器、Octasic 的 vocall 应用处理器、德州仪器 DM355 媒体处理器[5]等。它们可以运行微实时操作系统进行实时管理，并具有片上独立的 I/O 和协处理单元用来捕获格式化的 I/O 数据流以及完成多媒体加速算法，同时还支持动态电源管理从而降低功耗保存电能。这些产品可以对现今最流行及最新颖的多媒体文件进行高效率解码和编码工作，例如，DivX、MPEG-2、MPEG-4、MP3、AAC、JPEG 等文件。同时高集成度以及低功耗特征将会给这类 SOPC 带来广阔的市场前景。

而当今开放源代码运动的兴起，越来越多的新技术被共享，SOPC 也不例外，像 NiosII、LEON2、OpenRisc[2]以及大量的 IP 核，这样就有机会制作出自己所需的系统。本课题就是实现一个基于 NiosII 的嵌入式 SOPC 系统，并在移植了 μClinux 操作系统后，成功实现了 MP3 编解码器的工作。

1.2 音频编码技术的发展及国内外现状

自从 PC 支持多媒体以来，有几种音频编码标准得到了广泛的应用，数字音频编码技术无论对多媒体通信、多媒体广播、消费类电子都有其重要性。下面介绍一下音频编码技术发展中比较突出的几个标准。

(1) MPEG-1 音频编码标准

自从 1988 年以来，MPEG（运动图像专家组)承担了视频和音频压缩技术的

标准化工作。这个小组制定的音频编码标准是数字音频压缩领域中的第一个国际标准[3]。1989 年，MPEG 小组在征求了 14 种音频编码方案后，最后确定了 2 种：一种是 MUSICAM (Masking Pattern Adapted Universal Sub band Integrated Coding And Multiplexing，自适应掩蔽模式通用子带综合编码与多路复用)，另一种是 ASPEC(Adaptive Spectral Perceptual Entropy Coding，自适应频谱感知熵编码)。基于这两种算法于 1992 年制定了 MPEG-1 标准。MPEG-1 按照算法的复杂度和压缩比分为 I，II，III 三个层次。第 I 层的复杂度最低，是 MUSICAM 方案的简化形式，以每声道 192kbps 的速率提供高质量的声音，在不强调低码率的情况下应用。第 II 层具有中等复杂度，它使用比第 I 层更为精密的量化，与 MUSICAM 方案几乎完全相同，可在 128kbps 的码率下提供近乎 CD 质量的声音。第 III 层结合了 MUSICAM 和 ASPEC 的优点，复杂度最高，编码效果也最好，可在低于每声道 128kbps 的码率下获得极高品质的音频。第 III 层使用了心理学模型、可切换的混合滤波器组、比特池缓冲技术、先进的预回声控制、非均匀量化和熵编码技术。

MPEG-1 等级 III 在商业上获得了巨大的成功，这就是普遍使用的 MP3。MP3 是目前流传最广的一种音乐压缩格式，其 CD 般的音质、高压缩比、开放性和易使用性使之深受好评，尤其在 Internet 网络上广为流行，很多硬件厂商还推出了硬件设备 MP3 播放器，成为目前市场上音频播放器的主流。

(2) Dolby AC-2，AC-3

美国杜比(Dolby)实验室从 1980 年开始对数字声频技术进行研究，重点是降低比特率技术[4]。它先后研制了 AC-1 (Audio Coding-1)、AC-2 和 AC-3 技术。Dolby AC-2 是 60 年代中期开始发展的，用于立体声编码，具有 128kbps 和 192kbps 可变的数码率。它利用时域混叠抵消(TDAC)技术和 MDCT/IMDCT 变换。 1991 年 Dolby 发明了 AC-3，通道数由 1 到 5，再加一个低频增强通道。AC-3 算法允许数据率在 32~640kbps 内调整，可组合成各种通道组合，在低比特率时提供良好的音质，1993 年被美国 ATSC 确定为北美 HDTV 标准的声音编码系统，典型的 5.1 通道，数码率是 384kbps，每声道 64kbps。AC-3 还用于电影、DVD 等。

(3) DTS

DTS，数字影院系统(Digital Theater System)，是在杜比数字环绕声出现后的又一种数字环绕声系统。目前美国使用 DTS 作为其电影原声带数字音频编码方式的电影公司，已经超过了采用杜比数字系统的电影公司[5]。

DTS 之所以受到如此青睐，是由于其对高采样率、高量化精度的数字信号采用了灵活、先进的相干声学(Coherent Acoustics)编码技术。根据实际应用，DTS

具有单声道、双声道至 8 声道可供选用，分离式的 5.1 声道可以混音成为“矩阵式两声道”。DTS 每声道的采样频率最低为 8kHZ，最高为 192kHZ；量化精度范围为 16~24bit；压缩率范围为 l：1~40：1，总数据率范围为 32~4056kbps。另外相干声学算法还可以实现高达 138dB 的动态范围。

DTS 作为一种新型的数字环绕声技术，不但能胜任 AV 的重托，更能提高 CD 音乐的音质，完全可以和杜比数字制式在家庭视听方面想抗衡。

(4) MPEG-2 BC，MPEG-2 LSF 和 MPEG-2 AAC

针对 MPEG-1 只能进行单声道或双声道编码的局限性，MPEG 小组制定了多声道扩展的音频编码标准 MPEG-2 BC，它能够与已有的 MPEG-1 系统向下兼容。与此同时，MPEG 小组还制定了一个在较低采样频率(6kHZ，22.05kHZ，24kHZ)时效率高于 MPEG-2 的音频编码标准 MPEG-2 LSF。1994 年 11 月，MPEG 完成了 MPEG-2 BC 和 MPEG-2 LSF 的制定。对五个全带宽声道，MPEG-2 BC 在数据率为 640~896kbps 的情况下提供了高品质的音频[6]。

能够工作于 64kbps/channel 的系统有 MP3，AC-3 等，经严格测试在此码率时，它们的重建音质都达不到 ITU-R 和欧广联(EBU)关于听不出与 CD 有任何音质变差的作为无线广播的要求，所以在 1994 年 MPEG-2 通过的同时 MPEG 组织决定研究和制订新的要达到 ITU-R 和 EBU 要求的音频编码，新的音频编码不要求向下兼容，命名为 Advanced Audio Coding 即 AAC。有许多著名公司和大学参加了 AAC 标准的制订，其中有美国的 Dolby Lab，Lucent Bell Lab，AT&T Lab，德国的 Fraunhofer IIS University of Hanover，日本的 Sony，NEC 等。并很快地于 1997 年纳入到 MPEG-2 标准体系，成为其第七部分。AAC 的编码效率和商业价值被越来越多的国家广播机构和大公司青睐，日本无线电工业协会在 1998 年就决定日本的地面电视、有线电视、卫星电视一律采用 AAC 音频编码。Panasonic，Philips，Sanyo，Liquid 等公司已推出基于 AAC 的便携式产品或声源。可以预想 AAC 将有着广阔的发展前景。

(5) AVS

AVS 是我国具有自主知识产权的第二代音视频信源编码标准，编码效率比第一代标准(MPEG-2)高 2 到 3 倍，在技术上与 H.264 相似又比 H.264 实现复杂度低[7]。同时制定 AVS 标准的时候采取了“开放”、“自主”、“兼容”、“先进”的原则，侵犯其他专利的风险极小。此外 AVS 从系统的高度确立了内容传输、视频/音频编码格式和媒体版权管理，而 MPEG-2 和 H.264 仅仅只是视频编码标准。

1.3 国内外 SOPC 在多媒体控制系统的应用现状

当前 SOPC 在多媒体控制系统的应用倍受关注。多媒体在不断发展，越来越

多的影音格式流行起来，像 MP3、MP4。然而这些高质量低存储空间的影音格式给处理系统带来了越来越大的压力，因此各大公司都推出了自己的相关产品：

(1) PHILIP 的 Nexperia PNX5100 媒体处理器

Nexperia PNX5100 集成了媒体处理器核，提供强大的视频处理平台用于运动测定和向上转变。前视频和后视频处理起到了通用输入处理器和视频整合通道的作用，从而扩展了 PNX5100 的性能它还包括一个双通道 LVDS 接收器和两个双通道 LVDS 发送器。这一先进的后处理器还提供画中画和 OSD 图形插入功能。PNX5100 支持高端平板显示屏分辨率和包括 1366×7680@120Hz 和 1920×1080@120Hz 格式在内的刷新率[8]。

Nexperia PNX5100 提供一个强大的 C/C++环境的 TriMedia CPU，可以运行微实时操作系统来实现高效的、可预测的实时时间的响应。它具有片上独立的 I/O 和协处理单元用来捕获格式化的 I/O 数据流以及完成多媒体加速算法。它还具有一个复杂的存储结构来管理内部 I/O 和流水线访问外部存储器。PNX5100 同时还支持动态电源管理从而降低功耗、保存电能。

PNX5100 随同一个参考设计工具包一起发布，该工具包能够帮助电视制造商加快产品上市时间并降低开发风险。此外，它也能被用作单独的视频后处理器或主流电视处理器的辅助 IC，使得运动精确图像处理能够被作为一项性能提升来执行。Nexperiza PNX5100 已于 2008 年第一季度起量产。

(2) Octasic 的 vocall 应用处理器

Vocallo 可扩展媒体网关解决方案是同类产品中第一个支持全新且灵活的业务模式的 DSP 平台，可供 OEM 厂商用于设计媒体网关[9]。利用这种业务模式，OEM 厂商可以设定一种成本更低的网关，用于以可靠的性能处理当今的 IP 语音(VoIP)需求，从而为将来的应用做好准备。Vocallo 以 Octasic 特有的 DSP 核心体系结构 Opus 为基础，可以通过传统的编程摸式实现行业中最高的效能比。在高密度的应用中，Opus 是效率最高的 DSP 核心处理器.比同类产品高出两倍。此外，Opus 提供了一个丰富的指令集，这个指令集超越了可用于扩展到新应用的 DSP 函数。此外，Vocallo 还新增了一个优势，允许设计人员将他们自己的软件添加到 Vocallo 框架中以进一步突出自己产品的特色。Vocallo 的回声消除和音质增强(VQE)功能使用了 Octasic 经过了电信运营商所公认的算法。为了保持符合最新标准的纯正语音质量，Vocallo 采用了基于数据包的软件体系结构：该体系结构采用了全新的设计，可以支持宽带语音通信，并可以最小的延迟处理媒体连接。

(3) 德州仪器(TI)的 DM355 处理器

TMS320DM355 处理器是 TI 推出的面向便携高清视频应用的新型低成本

DaVnce 平台[10]。DM355 可以实现 720P 高清 MPEG4 编码或解码，支持 30fps 实时处理，编码解码能力可达每秒 5000 万像素。其内核包含频率为 216MHz 或 270MHz 的 ARM926EJ-STM，MJCP 协处理器和有前端和后端的视频处理子系统，可支持 CCD 控制器预览、图像缩放。ARM 端有 16kB 指令高速缓存、8kB 数据高速缓存、8kB ROM 以及 32kB RAM 存储器。外设包括：USB2.0 HS OTG 设备与迷你主机物理接口，可连接 DDR2 的外部存储器接口(EMIF) 。

集成式 MJCP(MPEG4 和 JPEG 协处理器)相当于 400MHz 的 DSP，预览处理引擎的等效 DSP 相当于 90MHz，图像缩放部分等效为 60MHz，OSB 等效为 90MHz，因此整体相当于 240MHz 的处理能力，而 VPSS(视频处理子单元)+MJCP 可以提供相当于 640M Hz DSP 处理能力，并针对 OEM 产品差异化提供 ARM，以实现整体系统的控制，也可以实现实时操作系统。

国内公司以及研究机构也有自己的相关产品：

(1) 方舟科技 Arca210

Arca210 核心是一款 32 位的微处理器，它不但具有 RISC 体系结构的典型特征，而且具有一套全新的高性能、低功耗的指令体系结构。同时系统集成了嵌入式产品所需的大量外设以及 PC 架构南北桥中的大部分功能，为嵌入式多媒体系统的设计提供了一个很好的选择[11]。

(2) 自主芯片“凤芯 2 号”

“凤芯 2 号”与“凤芯 1 号”都是由宁波中科集成电路设计中心负责开发，主要应用于高清晰度数字电视、高密度 DVD 机、数码摄像机、视频会议、视频监控等。“凤芯 2 号”与“凤芯 1 号”都是基于目前公布的 AVS 标准研制开发，兼顾 H.264 标准。同时“凤芯 2 号”完全兼容 AVS1.0，H.264/AVC main profile 标准，但在应用方面的技术远高于以前的技术。

1.4 论文的组织结构

作为一个嵌入式 SOPC 系统，基于 NiosII 的 MP3 编解码系统是软件与硬件的结合，而且还移植了 μClinux 操作系统，使其功能更加强大，扩展性能更为。

论文的安排如下：

第 1 章：绪论。主要描述了本课题的研究背景，介绍了音频编码技术的国内外现状，以及 SOPC 在多媒体控制系统中的应用现状。

第 2 章：MP3 编解码器的相关技术介绍。主要介绍了 MP3 标准，对 MP3 的编解码算法原理进行阐述，同时对硬件和软件相关技术进行介绍。

第 3 章：MP3 编解码器的总体设计。主要描述本设计的总体结构，介绍了

硬件和软件的开发环境，为后期的实现进行规划。

第 4 章：MP3 编解码器硬件的设计与实现。这部分中完成 Nios 软核处理器的设计与实现，重点描述了 Audio(音频处理)模块的设计与实现。

第 5 章：MP3 编解码器软件的设计与实现。分析了编解码器的软件流程，解释了 MP3 编解码器的软件实现过程。

第 6 章：MP3 编解码器的测试与问题解决。介绍了系统的软硬件测试与协同实现以及出现的问题解决。

第 7 章：结束语。

第 2 章 MP3 编解码器的相关技术介绍

2.1 MP3 标准简介

MP3 的全称为 MPEG1 Layer-3 音频文件[12]，MPEG 音频文件是 MPEG1 标准中的声音部分，也叫 MPEG 音频层。它根据压缩质量和编码复杂程度划分为三层，即 Layer1、Layer2、Layer3，且分别对应 MP1、MP2、MP3 这三种声音文件，并根据不同的用途，使用不同层次的编码。MPEG 音频编码的层次越高，编码器越复杂，压缩率也越高，MP1 和 MP2 的压缩率分别为 4：1 和 6：1~8：1，而 MP3 的压缩率则高达 10：1~12：1。一分钟 CD 音质的音乐，未经压缩需要 10MB 的存储空间，而经过 MP3 压缩编码后只有 1MB 左右。不过 MP3 对音频信号采用的是有损压缩方式，为了降低声音失真度，MP3 采取了“心理声学模型”，即编码时先对音频文件进行频谱分析，然后再根据心理声学模型把谱线分成若干个阈值分区，并计算每个阈值分区的阈值，接着通过量化和熵编码对每个谱线进行编码，最后形成具有较高压缩比的 MP3 文件，并使压缩后的文件在回放时能够达到比较接近原音源的声音效果。

表 2.1 各种音频的数据
Table 2.1 All kinds of audio data

音质	带宽	模式	码率	压缩因子
电话音质	2.5kHz	mono	8kbps	96：1
短波	4.5 kHz	mono	16 kbps	48：1
中波	7.5 kHz	mono	32 kbps	24：1
调频	11 kHz	stereo	56-64 kbps	26~24：1
接近 CD	15 kHz	stereo	96 kbps	16：1
MP3	>15kHz	stereo	112-128 kbps	14~12：1

从表 2.1 可以看出，采用 MP3 标准压缩的声音具有音质好、码率低的特点，而且根据国际上众多的视听测试，MP3 提供了非常好的性能，根据测试，在 10kHz 的通道中完全可以实现立体声广播。

人耳对于不同的声音的感觉是不同的，强的声音往往可以淹没弱的声音。所以在编码的时候就没有必要将所有的声音进行编码，这样就减小了数据量。

MP3 使用了非常经典的 Huffman 算法[13]，MP3 使用 Huffman 算法实现压缩的最后步骤。在编码过程中，Huffman 算法产生一个可变长的码流，并且可以根据一个相应的表格解决码流不等长的问题，而且解码速度非常快，同时压缩比也较高，平均可节省 20％的空间。MP3 虽然有压缩比大的优点，但因为它的编码原理的限制，同样也存在缺点。MP3 一个显著的缺点就是延迟时间长，它的最

小理论延时是 59ms，而实际上的值要比这大许多，而且同系统的实现方法有关，很难给出一个精确的值。对于一些特定的应用，比如全双工语音通信，这么长的延时将影响通话效果。

语音压缩的目的是使用尽可能少的比特的同时获得尽可能高的语音质量。在众多的语音压缩算法中，MPEG 的语音压缩标准得到了最为广泛的应用，即 MP3。MPEG 语音压缩算法属于有损压缩算法，因此它可以达到很高的压缩比，但同时也具有很高的语音质量。

MPEG-1 的 MP3 标准支持 48kHz，44.1kHz，32kHz 三种采样频率和 32kbps~320kbps 间的 14 种比特率。MPEG-2 LSF(低采样率模式)在不对原 MPEG-1 中 MP3 算法进行重大改动的条件下，通过降低采样率来达到降低比特率的目的。MPEG-2 LSF 支持 24kHz，22.05kHz，16kHz 三种采样频率和 8kbps~160kbps 间的 14 种比特率。由奈奎斯特采样定理可知，采样频率降低一半，信号在频域的带宽也降低一半，因此对 MPEG-2 而言，其信号的频宽比 MPEG-1 窄一半，但由于人耳对高频信号不甚敏感，这种损失是可以接受的。MPEG-2.5 在 MPEG-2 LSF 的基础上，把采样频率又降低一了半，它支持 12kHz，11.025kHz，8kHz。因此 MP3 音频所支持的采样频率从最低的 MPEG-2.5 的 8kHz 到最高的 MPEG-1 的 48kHz。对于 MPEG-1 单声道来说，每一帧(frame)中有 2 个颗粒(granule，而简写为 gr)，共 1152 个样本(sample)；而对于 MPEG-2 和 MPEG-2.5 单声道来说每个 frame 中只有 1 个 gr 共 576 个样本[14]。

2.2 MP3 编码与解码算法的原理

2.2.1 MP3 编码算法原理

MP3 是一种基于频域的子带编码，其压缩算法结构如图 2.1 所示。

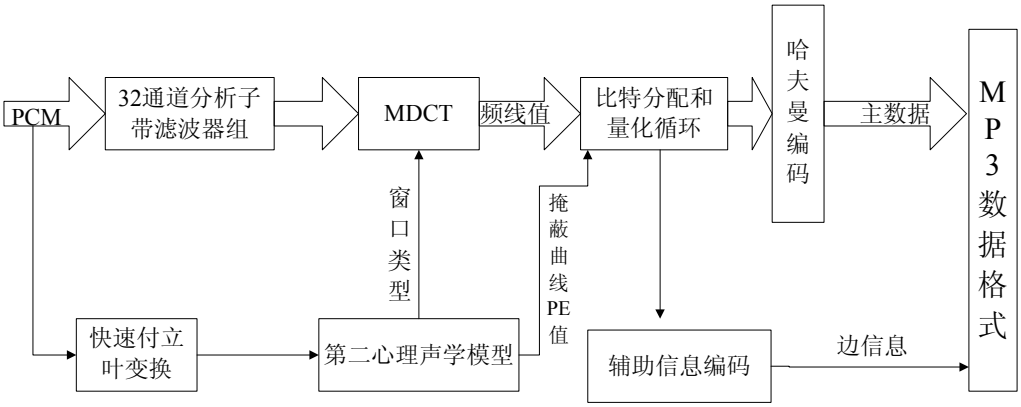

图 2.1 MP3 压缩算法结构图

Fig.2.1 MP3 compress algorithm structure diagram

原始的模拟信号经过 AD 采样器后，转化为 PCM 样本，作为 MP3 编码器的数据输入源。整个 MP3 编码是基于频域的编码，因此必须将原始的 PCM 做时频

转换。由于不可能将整个时域上的信号通过一次付立叶变换转换到频域上(音频信号是时变的)，因此只能在时域上逐段进行，于是将 PCM 样本分块进行处理。前面提到，对单声道 MPEG-1 而言，一帧包含 2 个 gr；而 MPEG-2 和 MPEG-2.5 一帧只包含 1 个 gro。MP3 编码则是以 gr 为基本单元的。首先 576 个 PCM 样本经过多相分析滤波器组(Polyphase Analysis Filterbank)，转换成 32 个等频宽的子带信号(Subband Signal)，然后通过改良离散余弦变换(Modified Discrete Cosine Transform，MDCT)，将子带时域信号转化为频线值，每个子带 18 条，一共 576 条频线[15]。在此过程中，会用到第二心理声学模型提供的窗口类型。对于快变信号做三次连续短窗(12 点)的 MDCT，以提高信号的时域分辨率和控制前回音；对于慢变信号做一次长窗(36 点)的 MDCT，以提高信号的频域分辨率。

同时，第二心理声学模型根据左右声道的相关性计算决定用 M/S(Middle /Side)还是用 L/R(Left/Right)编码方式。然后根据第二心理声学模型提供的 PE(perceptual entropy)值分配每帧编码比特数，对频线值进行量化；再将量化结果进行反量化计算量化噪声，看其是否在掩蔽曲线以下；否则反复调整比例因子(scalefactor)，重新量化，直到量化噪声达到最优，退出量化循环，对量化结果进行哈夫曼编码。最后按照 MPEG 标准规定的数据流格式输出。

以上简述了 MP3 压缩编码算法的全部过程，MP3 解码是 MP3 编码的逆过程，了解了 MP3 压缩编码算法后，解码问题自然迎刃而解。

2.2.2 MP3 解码算法原理

MP3 文件的解码过程是需要将数据比特流放入解码器中，解压帧，读取头信息来重建和转化映射表。解码的主要过程包括同步提取和错误校验、哈夫曼解码、反量化、重排序、立体声解码、反混叠、IMDCT 变换、频率反转补偿、子带合成，最后输出原始的 PCM 数据。如果错误检测被应用到编码中，那么在比特流解压过程中也要进行错误检测，同时比特流数据被解压成恢复为各种片信息。重排序模块将重构映射采样集的量化表，按照这个量化表将这些采样数据返回成统一的 PCM 数据[16]。MP3 解码的结构框图如图 2.2 所示。

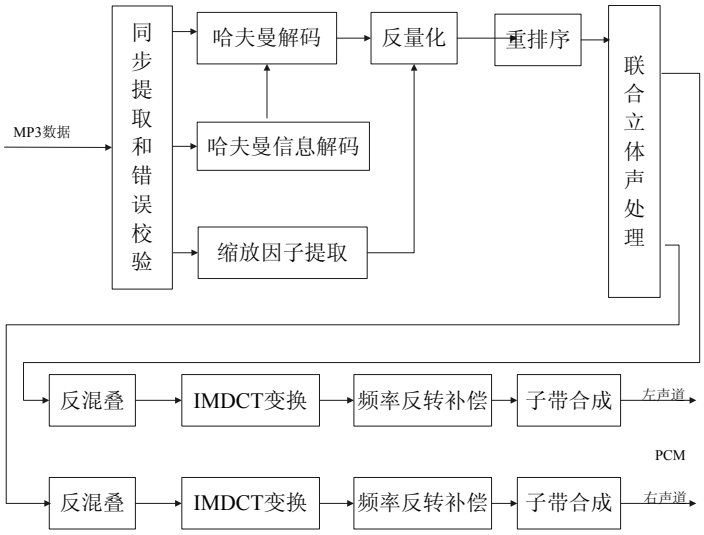

图 2.2 MP3 具体的解码结构框图
Fig.2.2 MP3 specific decoder diagram

2.3 基于 NiosII 软核的 SOPC 系统架构

Altera 的 SOPC 系统有两种嵌入式内核，一种在 FPGA 器件内嵌入 ARM922T 硬核，另一种使用 Altera 自己的 32 位嵌入式精简指令集软核 NiosII。NiosII 是一个基于流水线设计的通用 RISC 微处理器，拥有多级流水线和指令与数据内存分开的哈佛结构[17]。NiosII 支持 16 位和 32 位的数据总线并都使用 32 位的指令集。由于嵌入了 ARM922T 硬核的 Excalibur 系列器件价格非常昂贵，而 NiosII 几乎可以不受限制应用绝大多数 Altera 公司提供的 FPGA 器件中，所以 NiosII 软核处理器是目前世界上应用最广的嵌入式内核。本文所提及的 SOPC 系统也是基于 NiosII CPU。可以说 NiosII CPU 和 Avalon 总线加上外围 IP 一起构成了整个 SOPC 系统的主要内容。

SOPC Builder 开发 NiosII 系统的具体实现过程如图 2.3 所示。

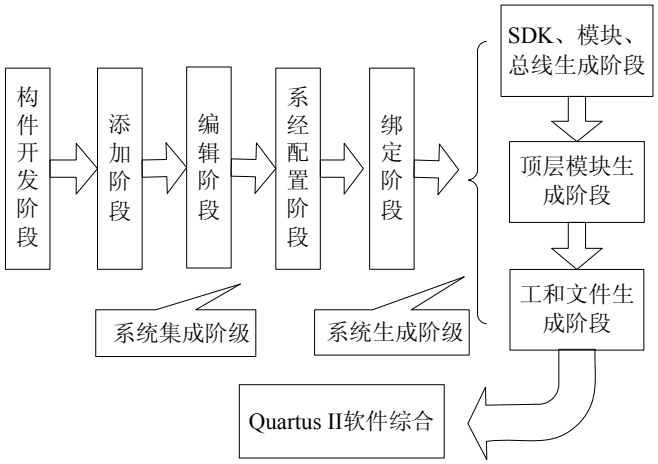

图 2.3 SOPC Builder 开发 NiosII 系统的具体实现过程
Fig.2.3 Specific implementation process of SOPC Builder developing NiosII system

2.3.1 简单的 SOPC 系统

图 2.4 显示的是一个简单的基于 NiosII 软核处理器的 SOPC 系统。从图中可以看出 Avalon 总线在 Cyclone II FPGA 中起到的是中间枢纽的作用[18]；NiosII 软核处理器与 Avalon 总线交换数据执行 CPU 的指令，同时通过控制 JTAG Debug 模块与主机相连，可以将运行数据反馈给主机；主机也可以通过 JTAG UART 接口下载程序到芯片中；而开关并行的输入输出接口则可以对于 NiosII 系统进行选择与显示[19]。

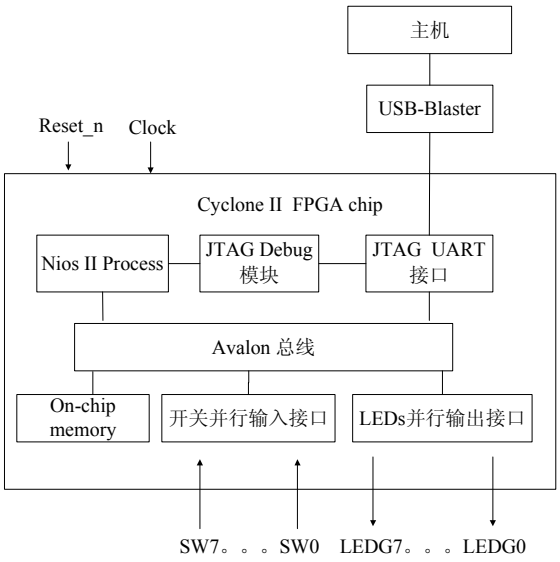

图 2.4 一个简单的基于 NiosII 软核的 SOPC 系统结构图
Fig.2.4 a simple SOPC system structure diagram based on NiosII soft-core

2.3.2 NiosII 软核处理器

NiosII 是 Altera 公司推出的一款非常经典的软核处理器，虽然没有 MMU，但是它能建立很多带有不同 UART、SDRAM 控制器、DMA 控制器、网络接口和普通 I/O 设备的微控制系统(包括 USB 设备)。作为一款常用的 32 位 RISC 处理器核，NiosII 拥有 32 位指令集、数据路径、地址空间、32 通用寄存器和 32 个外部中断源[20]。这种结构支持分离的指令和数据总线，NiosII 是依靠哈佛结构进行分类[21]。一个 NiosII 处理器系统等同于一个包含着一个处理器、复杂的接口和单芯片的存储器的微控制器或是 SOC。

NiosII 有三个不同的版本：NiosII-e、NiosII-s、NiosII-f[22]。

(1) NiosII-e 针对低成本应用，内部没有流水线结构，不带指令/数据 CACHE，所以实现所需要的逻辑较小(600-700 LEs)；

(2) NiosII-s 有五级流水结构，带指令 cache(size 根据用户配置)，实现需要 1200-1400 Les；

(3) NiosII-f 有六级流水结构，带指令和数据 cache，实现需要 1400-1800 Les。

2.3.3 Avalon 总线

NiosII 作为一种微控制器，它内部的重要部件是 Altera 的“Avalon-MM 总线”。

Avalon 是一种简单的总线体系结构，用来将处理器和周边设备集成到 SOPC。并规定了主设备和从设备的端口连接方式和时序关系[23]。其基本设计目标是：

(1) 简单性：提供一个易于理解的协议；

(2) 为总线逻辑提供优化的资源：节约可编程逻辑器件的逻辑单元；

(3) 基于同步操作：易于与片上的其他用户逻辑集成，避免了复杂的时序约束和分析过程。

基本的 Avalon 总线传输是在主设备和从设备之间传输一个字节、半字或字。一次传输过后，总线可以立刻进行下一次传输，而且与上一次传输的目的设备和源设备无关。Avalon 总线还支持外设等待执行时间、传输外设和多主设备总线等功能。这些传输模式使得在一次总线传输中，在外设之间能够完成多个数据单位的交换[24]。

Avalon 总线主设备和从设备的交互是采用“从端仲裁”技术，在多个主设备试图访问同一个从设备时，用于决定哪个主设备获得访问权。这使 Avalon 总线具有以下两个优点：

(1)仲裁的细节被封装到 Avalon 总线内，主设备和从设备的接口与总线上设备数目无关；

(2)多个主设备能够同时执行总线传输，只要它们不在同一时钟周期访问同一个从设备。

Avalon 总线结构的基本原则如下[25]：

(1) 到周边设备的接口同步于 Avalon 总线时钟，不需要复杂的异步握手/确认信号，Avalon 总线和整个系统的性能优劣，可以利用标准的同步时序分析技术来评测；

(2) 所有的信号都是高电平或低电平有效，有利于总线的切换(turn-around)。多路复用器(而不是三态缓冲器)决定哪一个信号驱动哪一个外设。即使外设没有被选中，此设备也不需要将其输出信号置为高阻态；

(3) 地址、数据和控制信号使用分开的专用端口，简化了外设的设计。外设不需要进行地址和数据译码，不需要判断当前总线周期的状态，即使没有被选中，也不需要关掉输出端口。

为了易于利用 SOPC Builder 软件自动生成系统，Avalon 总线还提供以下几个功能[26]：

(1)高达 4G 的寻址空间——存储器和设备可以被映射为 32 位地址空间中的

任何位置；

(2)同步接口，所有 Avalon 信号都被 Avalon 总线时钟同步。这可以简化和 Avalon 总线相关的时序，使得高速设备容易整合；

(3)分离的地址、数据和控制线，分离的地址、数据和控制通路保证了片内用户逻辑接口的简单性。设备不需要进行数据和地址译码；

(4)内建的地址译码，Avalon 总线对所有设备自动生成片选信号，极大的简化了 Avalon 设备的设计；

(5)多重主控制器总线结构，Avalon 总线自动生成仲裁逻辑；

(6)基于向导的参数配置，Avalon 总线架构可以基于用户的选择自动被生成；

(7)动态总线宽度调整，Avalon 总线自动处理在不同位宽的设备之间的数据传送。

Avalon 总线规范定义了通过 Avalon 总线模块在主从端口之间传送数据的信号和时序。不同的传达模式下，在 Avalon 总线模块和设备之间的接口信号是不同的[27]。以下介绍一种在该系统实际设计中用到的传送模式——从设备读写传送模式，如图 2.5 所示。

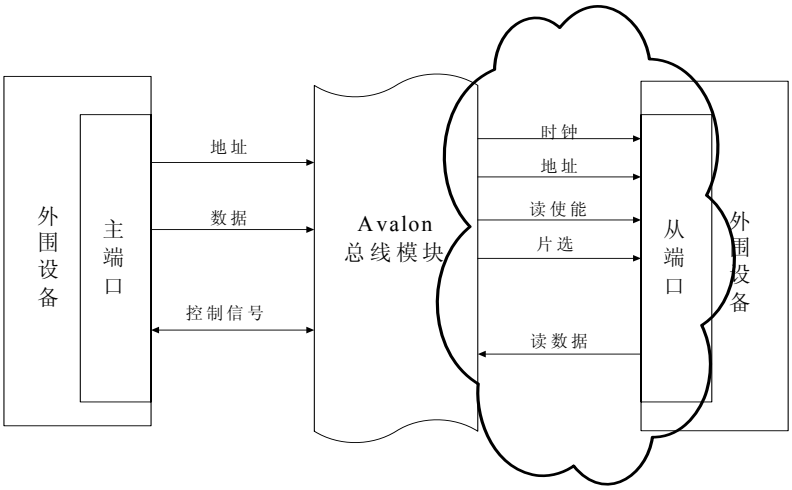

图 2.5 Avalon 总线从设备读写示意图

Fig.2.5 Schematic diagram of Avalon Bus reading and writing with slave peripheral

首先必须要认识到传送数据不是一件孤立的事情，往往数据是连续的传送。例如，一次从设备端读传送可能是发生在一次毫无相关的写传送的前后。在读传送过程中，目标设备的读使能和片选信号必须有效。但是在读传送结束后，如果在下一个时钟周期对同一个从端口的另一次总线传送发生，片选信号和读使能可能保持有效。

由设备控制等待状态的从端读传送允许目的设备根据自己的需要来拖延 Avalon 总线模块传送数据。利用这种传送模式，一个设备可以有充足的时间在总线上交换数据[28]。图 2.6 展示了由设备控制等待状态的从端读传送。设备通过从

端口输出的 wait_request 信号来控制等待状态。在输入到从端口的读使能有效之后，如果从端口希望延后读传送，那么它必须在紧接的时钟周期内返回 wait_request。当 wait_request 被设置成有效之后，它已拖延了 Avalon 总线模块，组织总线获取读到的数据。在 wait_request 被设置为无效之后的那个时钟周期的时钟脉冲的上升沿，Avalon 总线模块获取读到的数据[29]。

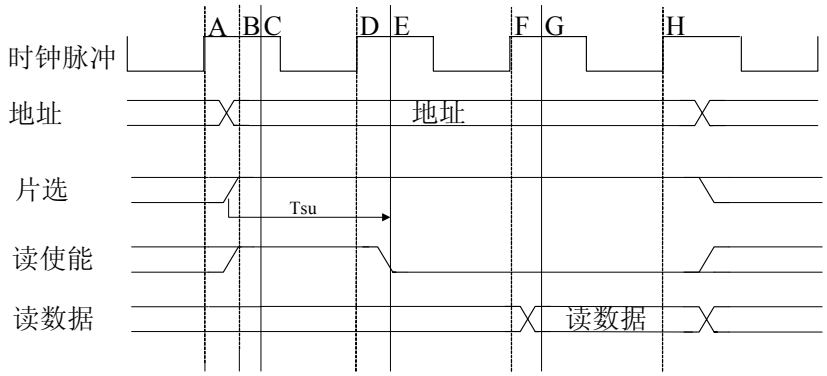

图 2.6 Avalon 总线从设备读写时序图

Fig.2.6 Time diagram of Avalon Bus reading and writing with slave peripheral

对于这种读传送模式，Avalon 总线没有超时的概念，也就是说可以想延迟多久就延迟多久。当 Avalon 总线模块被延时的时候，在系统模块内总有一个主控制器设备被延时，已在等待从端口数据的到来[30]。一个从端口可以永久的挂起这个主端口。因此，在设计时候必须小心的设计从设备的 wait_request 信号。

Avalon-MM 可以被作为传统的外围设备来使用，例如 SRAM，但它只支持简单的固定等待周期的读/写传输。另一方面 Avalon-MM 接口也能被用来描述一种更复杂的脉冲传输通道接口。

2.4 嵌入式 μClinux 操作系统

Linux 是一种很受欢迎的操作系统，它与 UNIX 系统兼容，开放源代码。它原本被设计为桌面系统，现在广泛应用于服务器领域。现在它正逐渐的应用于嵌入式系统领域，μClinux 正是在这种氛围下产生的。

在 μClinux 这个英文单词中 u 表示 Micro，小的意思，C 表示 Control 控制的意思，所以 μClinux 就是 Micro-Control-Linux，字面上的理解就是“针对微控制领域而设计的 Linux 系统”s。作为源代码公开的免费操作系统，μClinux 源代码可以从 www.uclinux.org 得到[31]。

μClinux 是针对控制领域的嵌入式 linux 操作系统，它从 Linux 2.0/2.4 内核派生而来，继承了主流 Linux 的绝大部分特性。适合不具备存储器管理单元(MMU)的微处理器/微控制器，没有 MMU 支持是 μClinux 与主流 Linux 的基本差异。

标准 Linux 是针对有 MMU 的处理器设计的。在这种处理器上，虚拟地址被

送到 MMU，把虚拟地址映射为物理地址。通过赋予每个任务不同的虚拟—物理地址转换映射，支持不同任务之间的保护。

对 μClinux 来说，其设计针对没有 MMU 的处理器，不能使用处理器的虚拟存储器管理技术。μClinux 仍然采用存储器的分页管理，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 μClinux 采用实存储器管理策略。μClinux 系统对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[32]。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。

同时，μClinux 有着特别小的内核和用户软件空间。熟悉主流 Linux 的开发者会注意到在 μClinux 下工作的微小差异，但同样也可以很快熟悉 μClinux 的一些特性[33]。对于设计内核或系统空间的应用程序的开发者，要特别注意 μClinux 既没有存储器保护，也没有虚拟内存模型，另外，有些内核系统调用也有差异[34]。

(1) 存储器保护

没有存储器保护(Memory Protection)的操作会导致这样的结果：即使由无特权的进程来调用一个无效指针，也会触发一个地址错误，并潜在地引起程序崩溃，甚至导致系统的挂起。显然，在这样的系统上运行的代码必须仔细编程，并深入测试来确保健壮性和安全。

对于普通的 Linux 来说，需要运行不同的用户程序，如果没有内存保护将大大降低系统的安全性和可靠性。然而对于嵌入式 μClinux 系统而一言，由于要运行的程序往往在出厂前已经固化的，不存在危害系统安全的程序侵入的隐患，因此只要应用程序经过较完整的测试，仍可以控制出现问题的概率。

(2) 虚拟存储器

没有虚拟存储器(Virtual Memory)主要导致下面几个后果：

首先，由内核所加载的进程必须能够独立运行，与它们在内存中的位置无关。

实现这一目标的第一种办法是一旦程序被加载到 RAM 中，那么程序的基准地址就“固定”下来；另一种办法是产生只使用相对寻址的代码(称为“位置无关代码”，Position Independent Code，简称 PIC)。μClinux 对这两种模式都支持。

其次，要解决存储器分配和释放问题。动态的存储器分配会造成存储器碎片，并可能耗尽系统的资源。对于使用了动态内存分配的应用程序，增强健壮性的一种办法是用预分配缓冲池(Preallocated buffer pool)的办法来取代 malloc()调用。由于 μClinux 中不使用虚拟存储器，进出存储器的页面交换也没有实现，因为不能保证页面会被加载到 RAM 中的同样位置。在普通计算机上，操作系统允许应用

程序使用比物理内存(RAM)更大的内存空间，这往往是通过在硬盘上设立交换分区来实现的[35]。在嵌入式系统中，通常用 FLASH 存储器来代替硬盘，很难高效地实现存储器页面交换的存取，因此，对运行的应用程序都限制其可分配空间小于系统的 RAM 空间。

最后，μClinux 目标板处理器缺乏存储器管理的硬件单元[36]，使得 Linux 的系统接口需要作些改变。有可能最大的不同就是没有 fork()和 brk()系统调用。调用 fork()将复制出进程来创建一个子进程。在 Linux 下，fork()是用 copy-on-write 页面来实现的。由于没有 MMU，μClinux 不能完整、可靠地复制一个进程，也没有对 copy-on-write 的存取。为了弥补这一缺陷，μClinux 实现了 vfork()，当父进程调用 vfork()来创建子进程时，两个进程共享它们的全部内存空间，包括堆栈。子进程要么代替父进程执行(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[37]。即使这个进程只是父进程的拷贝，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。

注意，多任务并没有受影响。那些旧式的、广泛使用 fork()的网络后台程序(daemon)的确是需要修改的。由于子进程运行在和父进程同样的地址空间内，在一些情况下，也需要修改两个进程的行为。

很多现代的程序依赖子进程来执行基本任务，使得即使在进程负载很重时，系统仍可以保持一种“可交互”的状态，这些程序可能需要实质上的修改来在 μClinux 下完成同样的任务。如果一个关键的应用程序非常依赖这样的结构，那就不得不对它重新编写了。

假设有一个简单的网络后台程序(daemon)，大量使用了 fork()。这个 daemon 总监听一个知名端口(或套接字)等待网络客户端来连接。当客户端连接时，这个 daemon 给它一个新的连接信息(新的 socket 编号)，并调用 fork()。子进程接下来就会和客户端在新的 socket 下进行连接，而父进程被释放，可以继续监听新的连接。

μClinux 既没有自动生长的堆栈，也没有 brk()函数，这样用户空间的程序必须使用 mmap()命令来分配内存。为了方便，在 μClinux 的 C 语言库中所实现的 malloc()实质上就是一个 mmap()。在编译时，可以指定程序的堆栈大小。

(3) 通用架构的内核变化

在 μClinux 的发布中，/linux/mmnommu 目录取代了/linux/mm 目录。前者就是修改后的内存管理子系统，去除了 MMU 硬件的依赖，并在内核软件自身提供基本的内存管理函数[38]。

很多子系统需要被重新修改、添加或者重写。内核和用户内存分配和释放进程必须重新实现。对透明交互/页面调度的支持也被去除。

内核中，加入了支持“位置无关代码(PIC)”的程序加载模块，并使用了新的二进制目标码格式，称为“扁平”格式(flat)，用来支持 PIC(具有非常紧凑的头部)。内核也提供了支持 ELF 格式的程序加载模块，用来支持使用固定基准地址的可执行程序。两种模式各有利弊，传统的 PIC 运行快，代码紧凑，但是有代码大小限制。例如 samsung S3C4510B 架构的 16 位相对跳转限制了 PIC 程序不能超过 32kbyte 大小[39]。而采用运行期固定基址的方法使得没有了代码大小限制，不过当程序被内核加载后有较多的系统开销。

(4) μClinux 的内核加载方式

μClinux 的内核有两种可选的运行方式：在 flash 上直接运行和加载到 RAM 中运行。

Flash 运行方式：把内核的可执行映像文件烧到 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 RAM 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。同时这也是标准 Linux 系统采用的启动方式[40]。

(5) μClinux 的文件系统

μClinux 系统采用 ROMFS 文件系统，这种文件系统相对于一般的 ext2 文件系统要求更少的空间。空间的节约来自于两个方面：首先内核支持 ROMFS 文件系统比支持 ext2 文件系统需要更少的代码；其次 ROMFS 文件系统相对简单，在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，对于系统需要动态保存的数据采用虚拟 RAM 盘的方法进行处理(RAM 盘将采用 ext2 文件系统)[41]。

应用程序如果需要以文件方式交换数据，可以将它存储在/tmp 目录下。这一目录实质上就是虚拟的 RAM 盘。不过在掉电时，这些数据就会丢失。

如果希望在掉电时，信息仍然可以保持，那么就要把它写到 FLASH 中[42]。这时，就可以使用 JFFS 这一文件系统，在 μClinux 的发布中，文件“/linux/drivers/block/flash.c”中提供的 JFFS 代码可以参考[43]。

2.5 音频信号编/解码芯片

音频信号编/解码采用的是 Wolfson Microelectronics 公司生产的一款低功耗的高品质双声道数字信号编/解码芯片——WM8731[44]。在本设计的硬件开发平

台——DE2 开发板中采用的是 28pin QFN 封装。它集成了耳机放大功能，因此 WN8731 也可以应用于 MD，DAT 等设备。它内建了 24bit 多位(multi-bit) sigma-delta 三角模数转换和数模转换，ADC 和 DAC 都使用了超采样数字插值技术，它支持的数字音频的位数可以是 16 位至 32 位，采样率从 8KHz 到 96KHz。立体声音频输出带有数据缓存和数字音量调节。WM8731 的内部有 11 个寄存器。该芯片的初始化以及工作时的工作状态和功能都是通过以 I2C 总线方式对其内部的这 11 个寄存器进行相应的配置来实现的。在 3.3V 信号电压时 ADC 可以达到 90dB 的信噪比，1.8V 信号电压时可以达到 85dB 的信噪比。3.3V 信号电压时的 DAC 信噪比可以达到 100dB，1.8V 信号电压时也有 95dB。ADC 和 DAC 的频率响应都在 8KHz 到 96KHz 之间，可以有选择的使用 ADC 的高通滤波。此外，WM8731 集成了一个片上时钟发生器，可以直接生成 44.1KHz、48KHz 和 96KHz 以其 MP3 标准定义的其他采样率，完全不需要独立的锁相环或晶振。

2.6 小结

本章主要讲解了 MP3 系统的相关技术，首先介绍 MP3 标准，着重讲述了 MP3 的编码和解码算法的原理，同时对于 NiosII 软核的 SOPC 系统和嵌入式 Linux 的基本结构进行说明，最后对本设计的关键编/解码芯片的工作原理进行的说明。通过本章的相关的技术介绍，对于系统以后的设计与实现起到了理论上的指导作用。

第 3 章 MP3 编解码器的总体设计

本设计融合执行一个声音记录系统所必须的软件和硬件技术。最有特色的是本设计的音频处理模块构件化，可以直接应用到基于 DE2 的 NiosII 软核系统中，另一特色是声音记录的 MP3 编码文件可以直接以“路径”的方式存储在外部存储器(USB flash 设备)上。下面将介绍 MP3 编解码器的总体设计。

3.1 MP3 编解码器的总体设计

本设计提供如下功能：

(1) 8KHz 采样、16 位录音；

(2) 在一个外部 USB flash 存储器中存储所记录的 wave 和 mp3 声音格式的声音片段，根据存储器的容量，可以支持不同的记录时间；

(3) 浏览外部存储介质的内容和播放已存在的声音片段；

(4) 基于 μClinux 操用系统下应用程序的开发；

从上面所述的功能可以看出，演讲人信号的带宽限制特性可以提供声波存储形式的选择，这样不但不影响声音的效果，而且降低系统设计的复杂性和生成文件的大小。

外置 USB flash 设备的选择提供了更大的存储空间，如果拥有 GB 大的空间，那么就可以扩展更大的存储容量，达到记录几百小时的声音片段，而且 USB 也是非常常用的设备。

本系统的总体方框图如图 3.1 所示。系统包括了 NiosII 硬件系统、硬件抽象层、软件应用层及外围设备组成。NiosII 硬件系统主要完成基于 Avalon 总线的各个硬件接口的开发，对外围设备进行控制，同时作为硬件抽象层与外围设备的接口，为 μClinux 操作系统的运行提供了相应的平台；而基于 μClinux 操作系统下的应用程序，则完成了对于 MP3 的编码和解码，并能支持 USB 存储设备。

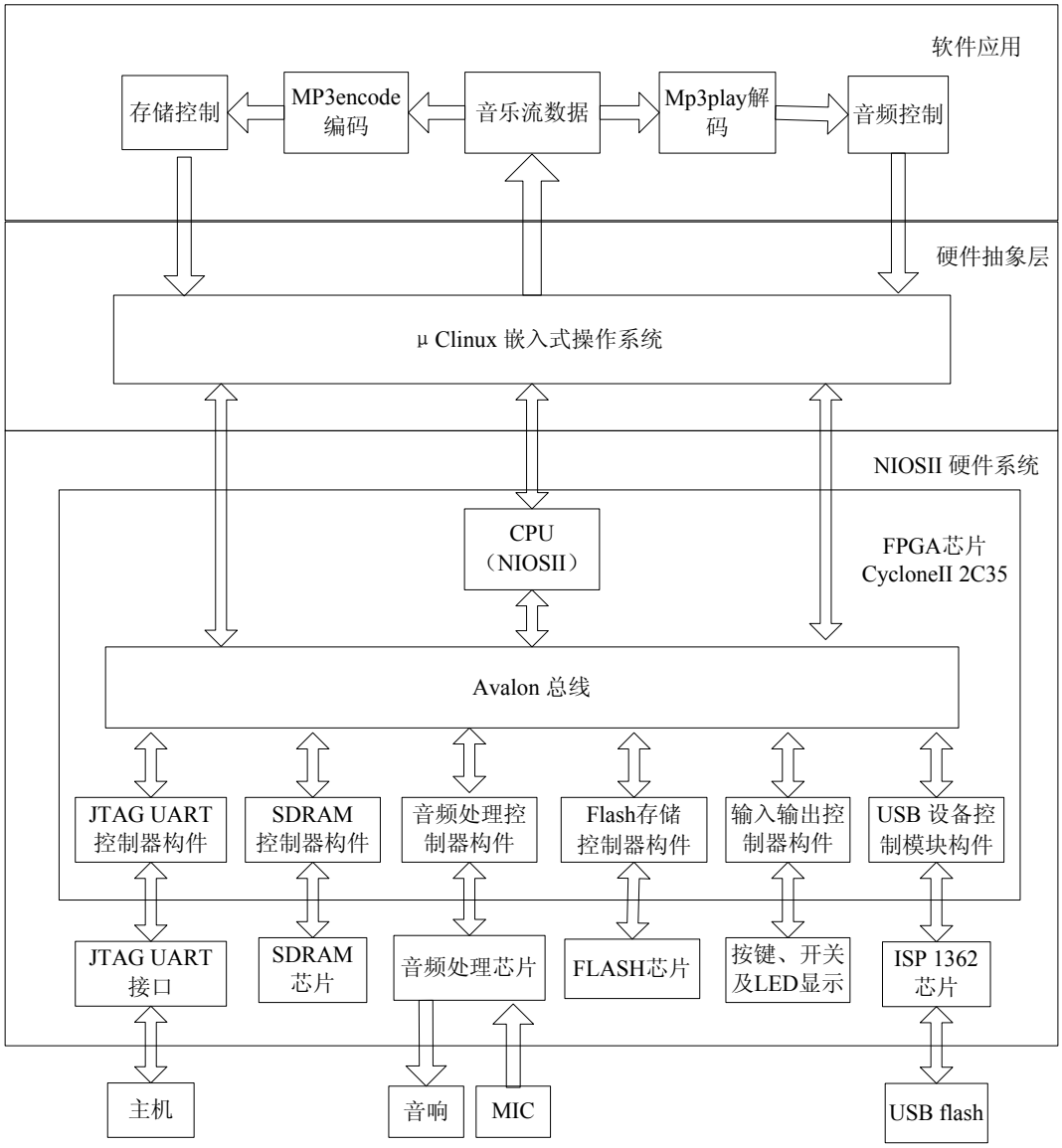

图 3.1 系统框图

Fig.3.1 System block diagram

3.2 MP3 编解码器的声音采样形式

正如上面所提到的那样，本设计的关键功能是声音信号的录音，存储及重新播放。本系统简化了声音的输入输出形式，这样设置虽然限制了系统的标准，但对于降低本系统的设计复杂性也是有益的。

更重要的是，声音信号是在一个 4KHz 的基带区域里，而且声音信号可能被限制在一个更小的区域(大约是在 3.3KHz)，这在声音质量上是有一些牺牲，但这些小的忽略是不会影响声音记录的效果的。因此本设计采用非常小的采样率为 8KHz，16 位的采样位(这样能提供更高品质的采样)。

采样率和量化水平的降低可以产生较低的采样比特率，因此本设计使用一种简单的记录形式—wave，而且这种记录声音的片段不会占有更大的空间。这样压缩文件大小，并且编码成 MP3 格式复杂性低和时间，同时也会使记录片段更加

接近实时系统。

3.3 MP3 编解码器的硬件环境

3.3.1 MP3 编解器的开发平台

硬件开发平台采用的是友晶(terasIC)公司开发的 Altera DE2 开发板。如图 3.2 所示。

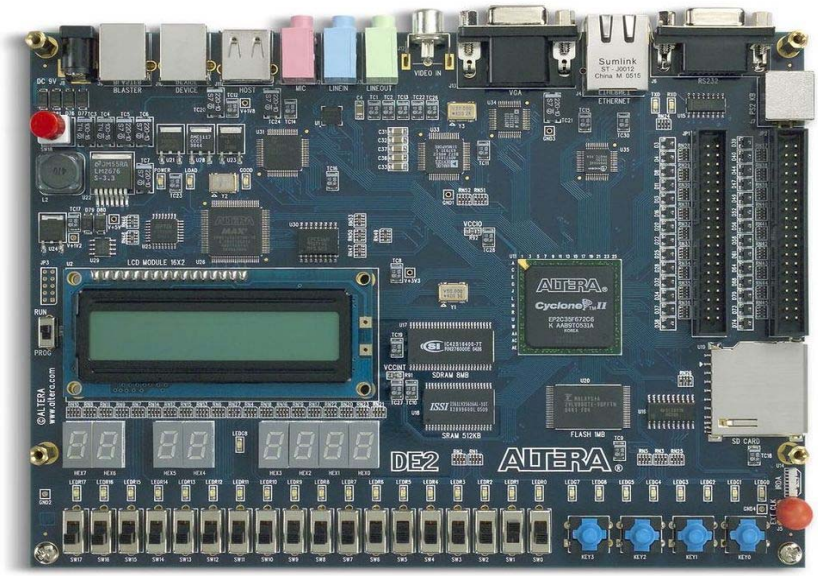

图 3.2 Altera DE2 开发板

Fig.3.2 Altera DE2 development board

其相关的属性如表 3.1 所示。从表 3.1 可知，DE2 开发板拥有更多的可编程逻辑单元和更丰富的接口，本设计选择 DE2 作为开发平台完全符合要求。

表 3.1 DE2 开发板属性
Table 3.1 The feature of DE2 development board

DE2 开发板板上资源	属性
Alter Cyclone II FPGA 类型	2C35
逻辑门数量	35000
按键/开关	4/18
LED 灯	9 个绿色/18 红色
七段数码管	8 个
SDRAM	8M
SRAM	512K
Flash	4M
9 针 RS-232 连接	支持
PS/2 鼠标/键盘	支持
24 位音频编解码芯片	支持(带有 line-in,line-out,MIC)
VGA 输出	支持
LCD	16×2 模块
TV 解码	NTSC/PAL
网络控制芯片	10/100M
USB 控制器	带用 USB A/B 的主/从连接控制
扩展插槽	2 个 40 引脚。

3.3.2 QuartusII 和 SOPC Builder

Altera 公司的 QuartusII 软件提供了 SOPC 设计的一个综合开发环境，是进行 SOPC 设计的基础。QuartusII 集成环境包括以下内容：系统级设计、嵌入式软件开发、可编程逻辑器件设计、综合、布局和布线、验证和仿真[45]。

本设计中使用的是 QuartusII 7.2 版本，其编辑调试界面如图 3.3 所示。

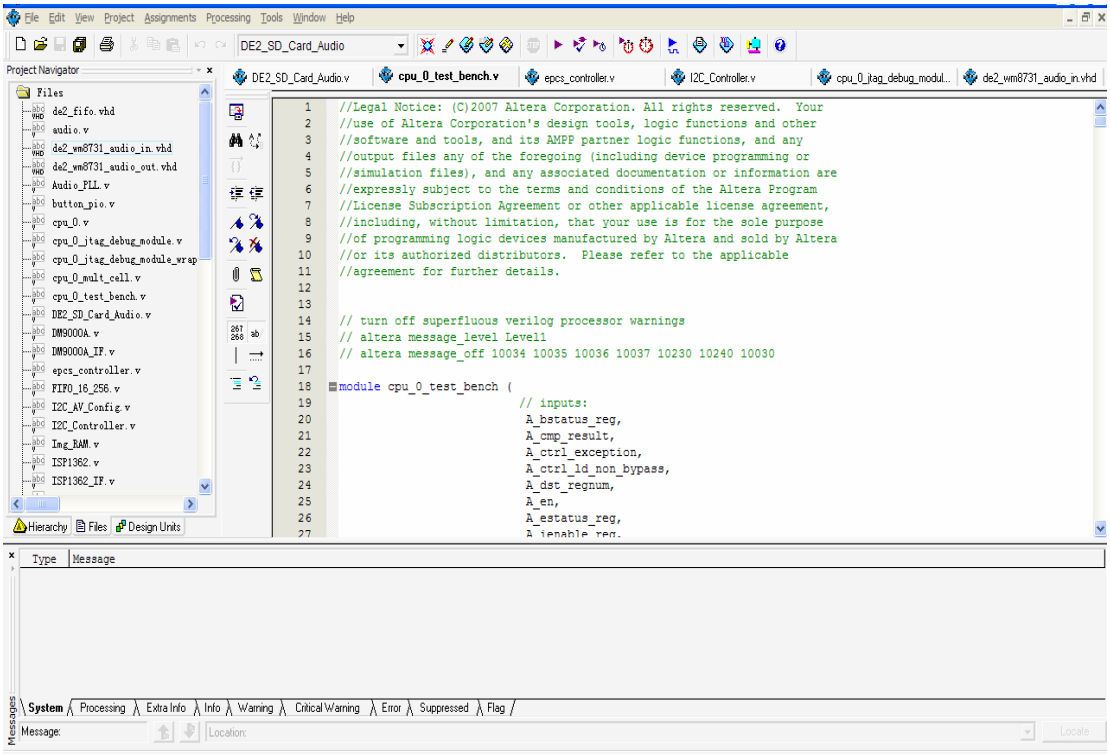

图 3.3 QuartusII 编辑调试环境

Fig.3.3 QuartusII editing and debugging enviroment

SOPC Builder 是 Altera 公司推出的实现 SOPC 概念的工具[46]。SOPC Builder 提供了一个强大的平台，用于组建一个在模块级和组件级定义的系统。SOPC Builder 的组件库包含了从简单的固定逻辑的功能块到复杂的、参数化的、可以动态生成的子系统等一系列的组件。这些组件是可以从 Altera 或其他第三方合作伙伴购买来的 IP 核，其中一些组件是可以免费下载用作评估的。用户还可简单地创建自己的定制的 SOPC Builer 组件。SOPC Builder 库中已有的组件包括：

(1)处理器：包括片内处理器和片外处理器的接口；

(2)IP 及外设：包括通用的微控制器外设、通信外设、多种接口(存储器接口、桥接口)、ASSP、ASIC、数字信号处理(DSP)和硬件加速外设等。通过这些接口，用户可以方便的实现各种外设[47]。

本设计中使用的是 SOPC Builder 7.2 版本，其开发环境如图 3.4 所示。

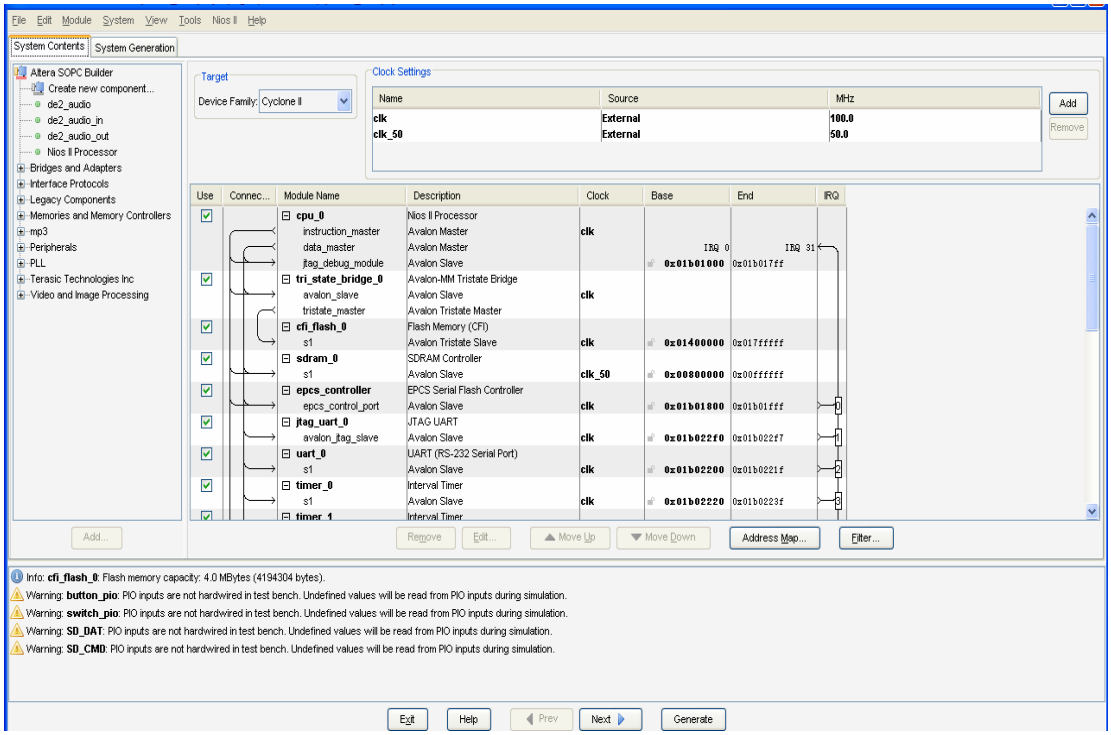

图 3.4 SOPC Builder 开发环境

Fig.3.4 SOPC Builder development enviorment

3.4 MP3 编解码器的软件环境

在本设计的软件编辑和调试是在 NiosII 集成开发环境(IDE)进行的的，然后在交叉编译环境中压缩为 zImage 镜像包，下载到 DE2 实验板上。下面介绍一下 MP3 编解码器的软件环境。

(1) NiosII IDE

Nios II IDE 是 NiosII 系列嵌入式处理器的基本软件开发工具，所有的开发任务都可以在 NiosII IDE 下完成，包括编辑、编译、调试程序和下载。图 3.5 是 NiosII IDE 编辑调试界面。

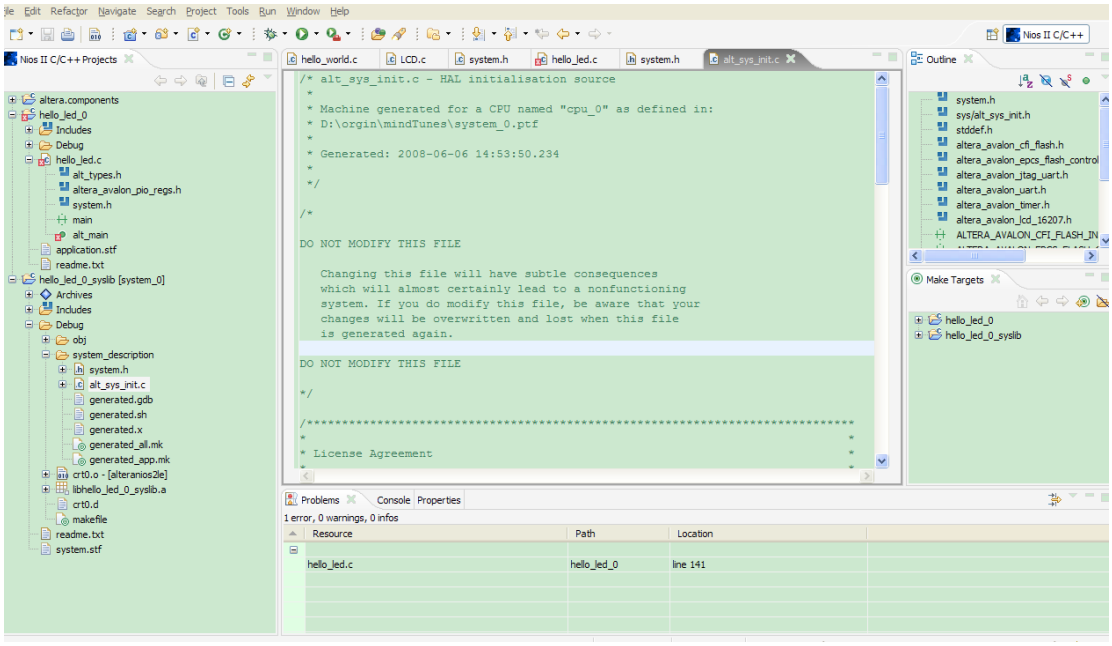

图 3.5 NiosII 编辑调试界面

Fig.3.5 NiosII editing and debugging interface

(2) 交叉编译环境

由于使用一台 PC，本设计通过在 Windows 操作系统上安装虚拟机来实现交叉编译环境。VMware Workstation 安装 Red Hat Linux9.0(内核为 2.4.20)。这样在安装 Windows 版的 Quartus II 與 SOPC Builder 產生硬體.sof 與.ptf，並在 VMWare 上安装 Linux 進行 μClinux 的 cross compiler，Windows 與 Linux 之間透過 VMWare Tools 傳檔，請參閱(原創) 如何在 CentOS 安裝 VMWare Tools? (OS) (Linux) (CentOS) (VMWare) 。这这 这这 Windows 下利用 Quartus II 与 SOPC Builder 产生硬件运行文件.sof 与.ptf，并在 VMWare 上安装 Linux 进行 μClinux(其内核为 2.6.1)的 cross compiler，Windows 与 Linux 之间透过 VMWare Tools 传递文件。由于在设计过程中，Red Hat Linux9.0 的 make 版本较低，通过升级解决问题。虚拟机工作站的调试环境如图 3.6 所示。

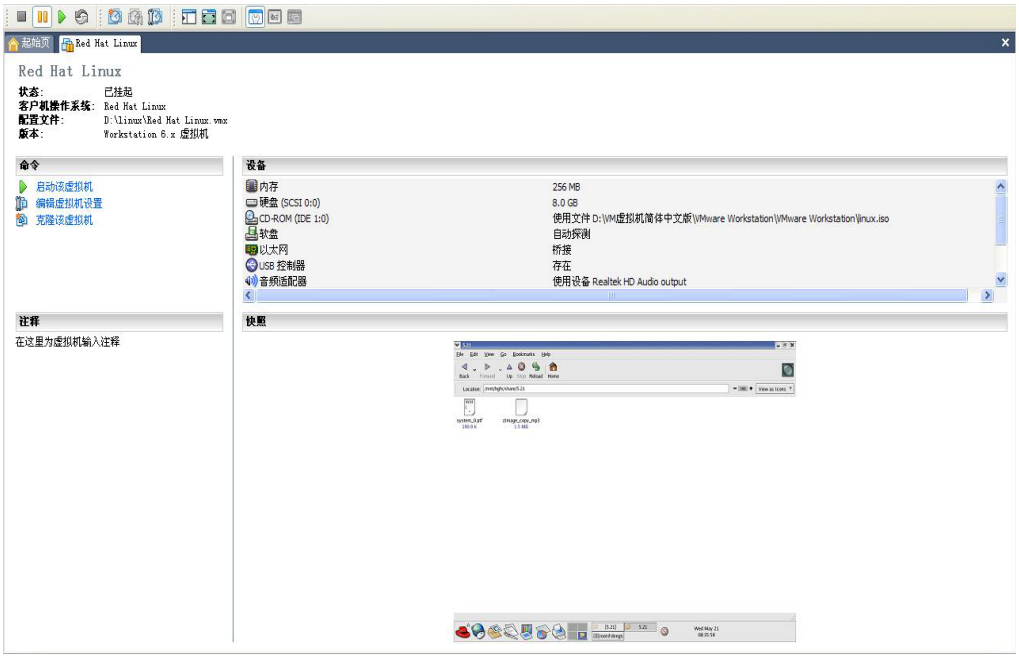

图 3.6 VMWare 虚拟工作站

Fig.3.6 VMWare virtual workstation

3.5 小结

通过对整个设计流程进行分析，给出了总体设计框图。同时了解了设计过程中声音的采样形式，介绍了 MP3 编解码器的硬件环境和软件环境，这为后面实现 MP3 解码器提供了编辑调试环境。

第 4 章 MP3 编解码器硬件的设计与实现

有了前面的相关技术介绍及系统的总体设计，下面就开始软硬件的详细设计与实现了。本章将介绍如何利用 SOPC 的设计方案构建 MP3 编解码器的硬件平台系统。包括基于 Avalon 总线的 Audio(音频处理)模块设计、NiosII 软核 CPU 的实现和其他模块设计，为 MP3 编解码器软件的设计和实现提供硬件平台。

4.1 MP3 编解码器硬件的详细设计

4.1.1 编解码器的硬件结构

开发板的核心芯片选用的是 Altcra 公司的 CycloneII 2C35 FPGA，主要外围设备包括 5l2Kbyte SRAM、8Mbyte SDRAM、4Mbyte Flash memory、USB Flash、l6/24 位 CD 音质的音频转换芯片、50MHz 和 27MHz 时钟。

IP 复用是 SOPC 技术的特点，利用 Altera 公司提供的标准 IP 库，在 QuartusII 软件的 SOPC Builder 中调用需要用到的 IP 核[48]，如 NiosII CPU、Timer、PIO 等，只要根据需求改变 IP 核参数就可以迅速构建 MP3 编解码器的硬件系统。MP3 编解码器的硬件系统结构图如图 4.1 所示。

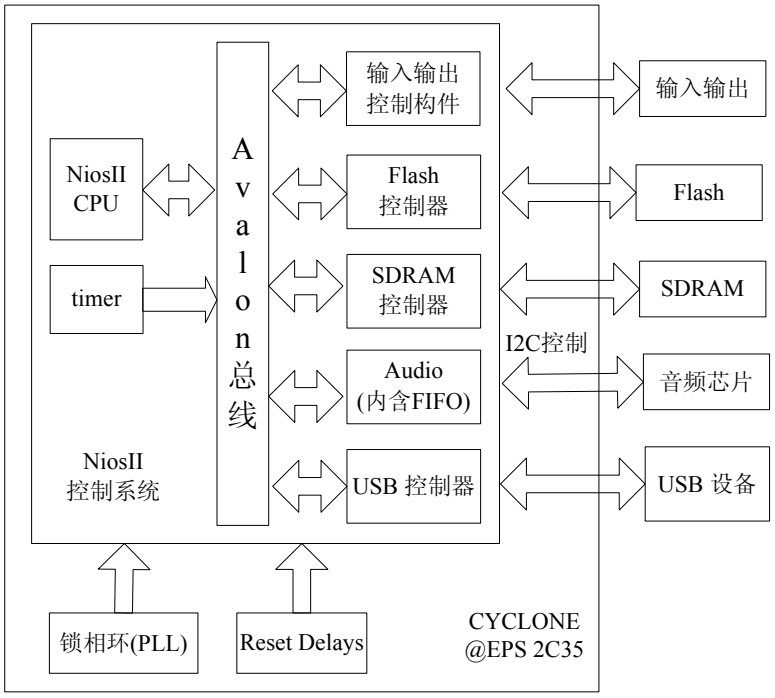

图 4.1 硬件系统结构图

Fig.4.1 Hardware system structure diagram

本设计是基于 SOPC 的 MP3 音频编解码系统，在设计和实现过程中最为重要的部分就是音频处理模块，下面介绍一下 Audio(音频处理)模块的硬件设计。

4.1.2 Audio(音频处理)模块的硬件设计

Audio(音频处理)模块是本设计的重要部分，是系统和外部音频设备相连的接口，其主要功能是实现声音记录和回放。它分为模数转换和数模转换两部分。它们将输入到系统的模拟形式的音频数据，转化为系统可以进行处理的数字形式，相应的数模转换处理就可以实现回放模拟声音的输出。这些构件的设置应当和其他已经存在的模块相配套，只有这样才能确保正确的数据传递给存储或编码结构。

(1) 音频处理模块的总体结构图

对WM8731芯片进行控制的音频处理模块结构如图4.2所示。从图中可以看出，在FPGA内部，设计了NiosII软核处理器以及挂在该NiosII系统的Avalon总线上的I2C配置接口模块和音频处理接口等。在FPGA外部，主要的部件是音频编/解码芯片WM8731。

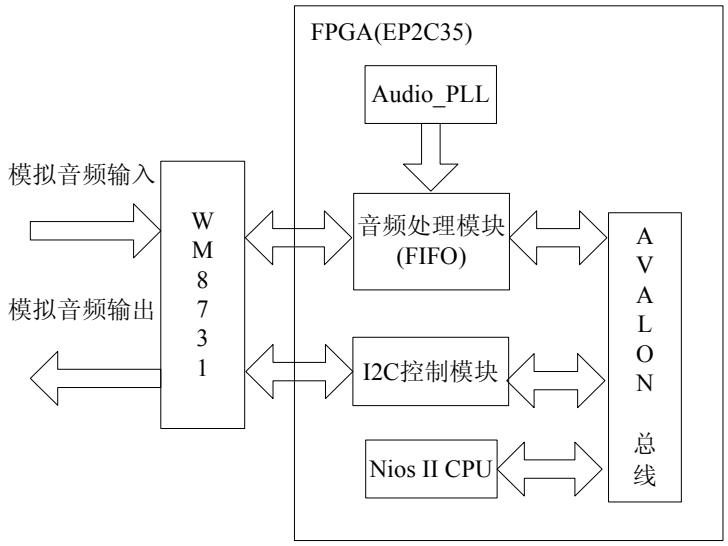

图 4.2 基于 WM8731 芯片的音频处理模块结构图

Fig.4.2 Audio process module structure diagram based on WM8731 chip

本设计在 NiosII CPU 及其相应的软件控制下，通过 I2C 配置接口模块对音频编/解码芯片 WM8731 进行配置后，由 WM8731 按照设计对输入的模拟音频信号进行 A/D 转换，对输出的数字信号进行 D/A 转换。转换前后的数字信号经过 FPGA 内的音频处理模块处理后，再交给 NiosII CPU 作后续的处理。

(2) 音频模块的时钟信号

音频处理模块在处理数据时要求有自己的时序，因此对音频处理模块要配置时钟信号，它使用锁相环 PLL(phase lock loop)来配置为 18.43Mhz，做为时序脉冲。其结构如图 4.3 所示。

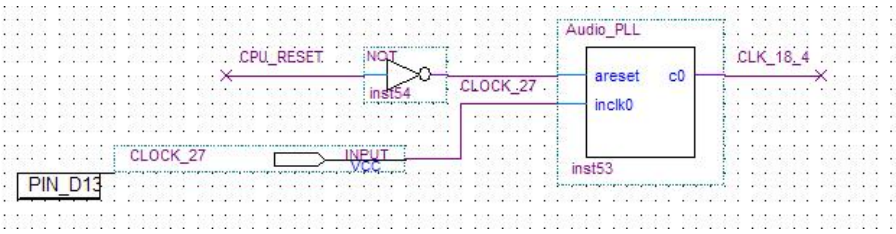

图 4.3 音频控制器的锁相环

Fig.4.3 The PLL of Audio controller

(3) 音频处理模块的 SOPC 构件的设计

Altera DE2 开发板提供了顶层级的 ADC 和 DAC(就像是编解码一样)，其功能应用可以通过软件控制来进行设定。这些设定包括模拟和数字的前端和后端信号滤波，同时也有 ADC 和 DAC 采样率和采样位数。这些参数具有传统的系统执行特性。举一个简单的例子，假定根据速度(主要是声音能量)和声音质量(通常指的是量化信噪比 SQNR(Signal to Quantization and Noise Ratio)或信噪比 SNR(signal to Noise Ratio))来设定采样位数的话，使用更多的位数将会导致更多的资源和能量的消耗。经音频处理模块对信号进行了处理后，虽然速度(声音能量)上发生了衰变，但此时声音质量方面却有很大的提高。

本设计主要关注存储方面，忽略了声音质量的降低。对于使用混合信号模块(ADC/DAC)来说，这样选择声音信号处理和存储的方式是非常明智的。本设计设定的采样率为 8kHz，采样位数为 16 位。8kHz 采样率是完全满足处理声音信号的，而带宽限制小于 4kHz(这是满足 Nyquist 定理)，采样位数采用 16 位将会更加符合设计要求。这样的选择有益于控制系统资源、运行速度和功率消耗，在编码处理的过程中将会达到最大的资源调动。很明显数据的串行化和 FIFO 结构所使用的空间也将是最小化。

音频处理模块的硬件设计需要有如下结构配置：

(1) 运行在从属模式的 Wolfson 音频芯片；

(2) 能处理数据串行向并行转化的串行化的转换结构；

(3) 先进先出结构(FIFO)作为一个缓存结构。

音频输入部分处理流程如图 4.4 所示。

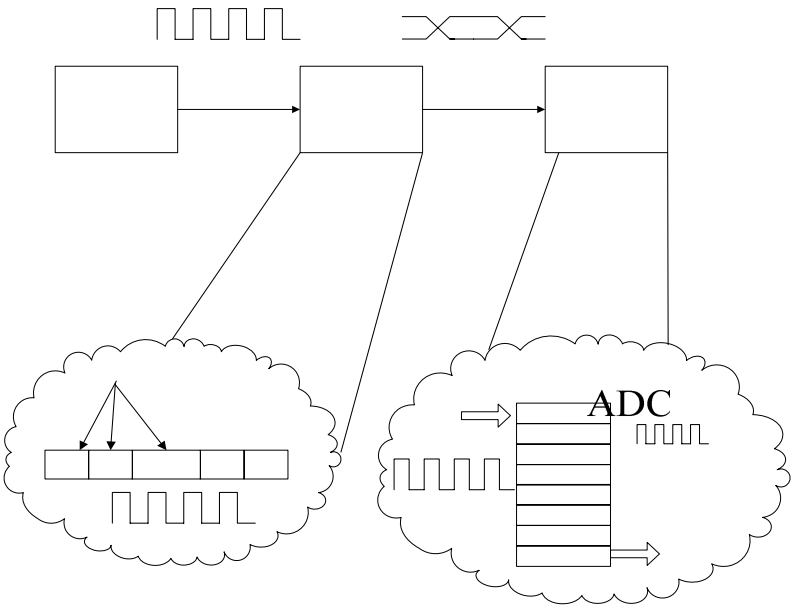

图 4.4 音频输入处理流程图

Fig.4.4 Audio input process flow chart

Wolfson 音频芯片里含有 ADC 和 DAC 结构。为了使芯片能提供更合适的采样信号(针对 ADC)，要对其设置正确采样率，同时回放存储声音片段时(针对 DAC)也一样。更为重要的是，声音接口结构的运行需要对应 ADC 为 8kHz 的采样时钟(如果立体声采样输入则为 16kHz)，而将这些音频数据并行化则时钟为 128kHz，即是$16\times 8kHz = 128kHz$ 时钟，这样才和串行输入相配套。这些时钟必须通过已经存在的音频时钟来创建(Altera DE2 开发板晶振时钟决定为 18.432MHz)，本设计使用了 2304 和 144 两个比率的分频来创建需要的时钟(18.432M/2304=8kHz，18.432M/144=128kHz) 。

但是对于 ADC 的输入和 DAC 的输出，音频芯片只处理 1 位的数据位流。无论是声音输入或是回放存储的声音片段，就要将 1 位的 ADC 串行采样转化为 16 位形式，而输出则是将 16 位的数据转化为 1 位，转化为一连串的比特流在 DAC 上输出。这就是并行化和串行化。并且对于音频处理模块来说，这些处理必须在下一帧声音采样数据来临之前完成。

考虑到从 ADC(如果探索结构的同步性，DAC 的实例更容易理解)得到的输入信号，在一个 ADC 采样时钟的并行化周期内，必须将 1 位数据转化为在后面的系统可以处理的单一 16 位“字”。很明显时钟必须比 16 倍的 ADC 采样速度更快。在“字”中，每一位的位置将根据 wav 和 mp3 定义来进行。这个功能如图 4.5 所示。

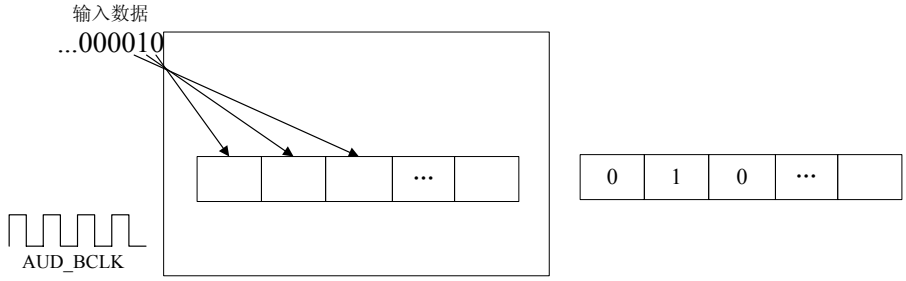

图 4.5 串行到并行的转换
Fig.4.5 Serial to parallel converter

音频处理模块最重要的部分是作为 FIFO 结构执行的“缓冲”部分。这样一个结构要求是很高的，因为系统的内部处理器以 100MHz 的速度运行(为了运行 μClinux)，这远远大于采样率。因为只有两个时钟，如果没有 FIFO 结构，那么内部处理器就要求有一个固定的时钟信号对输入进行采样，产生执行 ADC 的时序，显然这是不可能。为了解决这种执行问题，本设计使用了一个“等待”函数，它保持已存在的采样，并以串行的方式把采样数据传送给处理器，这就阻止了编码/解码在输入数据缺乏时运行(或者是对于 DAC 处理数据时，大量的采样数据将被传送到输出端口)，起到了缓冲作用。

本设计通过创建一个 Avalon 构件——de2_audio 来实现音频处理数据的功能。de2_audio 能执行两个 FIFO 结构，同时整合一些复杂逻辑来实现串行的逻辑控制处理器。但是这种结构有一些比较“紧”的时序定义。特别是本设计需要执行“暂停”功能，不管当前有没有数据，都需要处理采样，同时当需要播放采样时，有大量的采样数据已准备好(两者最好是同步的)。本设计选择 Legacy Mode 来运行 FIFO(Altera 文档中有描述)。FIFO 的结构如图 4.6 所示。

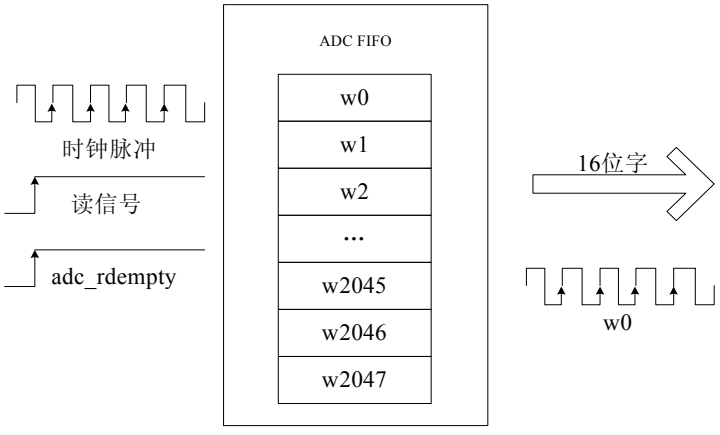

图 4.6 FIFO 结构
Fig.4.6 FIFO structure

因为控制逻辑的运行，本设计要执行一个 Moore 状态机，它最重要的功能是保持处理器系统的状态，这里假定当系统执行到上述时刻，总线就中止了“等待请求”。而且本设计需要在采样数据的第一次循环中，用“等待请求”来暂停处理器的功能，在合法数据到达总线前的延时内，完成 FIFO 结构的执行。

在转化过程中伴随的关键延时如下：

(1) 在 FIFO 中写入数据时，当 FIFO 为空或者几乎是满时，相应的信号之间中断产生的延时，这种延时是没有时钟循环的。

(2) 还有一种延时是在输出时表现出来的，是在 FIFO 中“读信号”的中断到指向合理数据之间的延时，它是“读时钟”循环的一个时钟周期。这种延时也是在“Legacy Mode”中 FIFO 结构的功能选择的一种效用，同时做为顶层执行结构提供给操作平台的。

状态机的设计是当预定的行为不能被完成时，使总线保持当时周期状态，例如当一个读命令被执行时，FIFO 所包含的从 ADC 传输过来的采样是空或者是 FIFO 从 DAC 中传出的是满。这个运行过程的关键点是和处理器的时钟对比是异步的，因此提出了一个 Moore 状态机。而且在一个功能命令(或读或写)被执行时，因为有等待响应信号，处理器将被暂停，一直等到数据在总线上重新合理出现时才恢复。这时提出了一个双处理时钟周期的时序间隔，在等待响应信号为高时，显示的是命令被成功处理，没有任何非法数据出现在编码过程中，状态机图如图 4.7 所示。

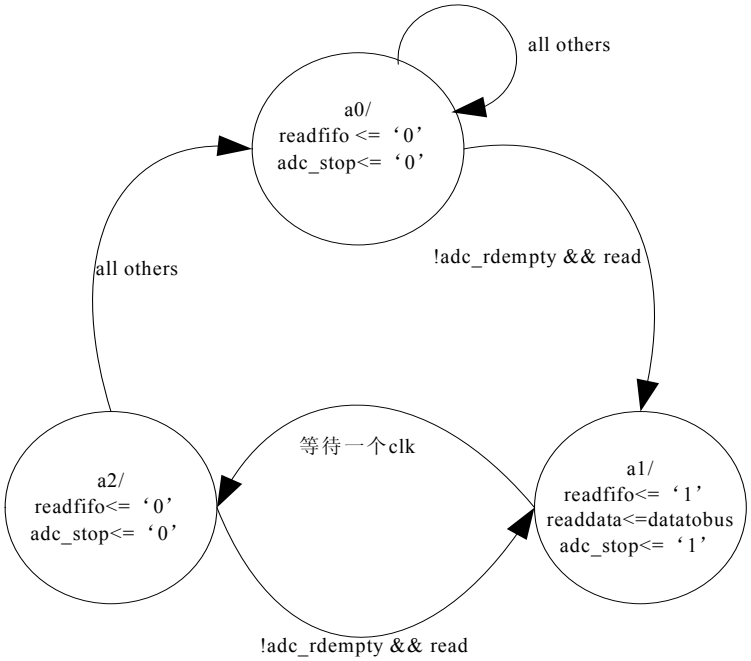

图 4.7 FIFO 所控制状态机

Fig.4.7 Function of FIFO controling state machine

如果系统复位，它将进行到状态 a0。在 a0 状态中，Avalon 总线读取 FIFO 的操作是被阻止的。状态机允许转换到下一个状态 a1，在 a1 状态中，rdempty 信号的功能是不让读响应执行。状态机处于状态 a1 一个时钟周期时，在检查 rdempty 和读取 Avalon 总线信号的响应以前，将声明“等待响应”(waitrequest)

和 rdreq。当前面提到的状态为真时，a1 将处于自循环。如果状态为假时，将进入到状态 a2，在这里总线再一次被 FIFO 的读取所阻止。在一个时钟周期后，状态由 a2 返回到 a0。

因为本设计包含两个独立的同时运行的状态机，在输出一侧的 Moore 状态机和输入一侧的相似。复位时系统进入到 d0 状态。之后，状态机只能进入到 d1 状态，wrfull 信号此时被取消，一个写响应将被执行。当状态机位于状态 d1 一个时钟周期，在检查 wrfull 和写 Avalon 总线信号的响应以前，此时声明等待响应和写响应。当前面提到的条件为真的条件下，在 d1 状态自循环。如果为假，将进入 d2 状态，此时总线将再一次因为写入 FIFO 数据而阻塞。在 d2 的一个时钟周期后，又返回到 d0 状态。

4.1.3 USB Flash 设备和接口

外部通用串行总线(USB)flash 设备被用作记录声音片段文件的存储介质。同时 flash 存储设备也将作为回放声音文件的存储介质。它不仅价格低，而且容量大(可以达到 GB)。DE2 开发板使用 Philips ISP1362 单芯片 USB 控制器作为 USB 主控和设备接口。flash 设备在操作系统(μClinux)上作为 USB 主设备挂载。正常执行 USB 部件需要一个设备驱动才能使用。幸运的是，μClinux 提供了一个 USB 设备驱动，本设计将使用它作为读写 WAV/MP3 的格式声音片段的设备。一旦挂载，USB 设备将作为一个软件编程的外围设备，将其定义为 PATH 目录来进行操作[49]。

4.2 基于 SOPC 的 MP3 编解码器系统硬件的实现

本设计的硬件系统部分包括 NiosII 软核处理器及控制模块、各种锁相环(PLL)、Reset_Delays 模块、I2C_AC_Config 模块、数据存取的 FIFO 模块和 USB Flash 设备控制模块和输入输出模块。

下面分别介绍一下各个模块的实现过程。

4.2.1 NiosII 软件核处理器及控制模块

本节使用 Quartus II 7.2 和 SOPC Builder7.2 来创建 NiosII 嵌入式处理器系统。

(1)创建 NiosII 系统模块。在 SOPC Builder 中，首先设置两个时钟频率，分别为 50MHz 和 100MHz，其中 50MHz 是专门给 SDRAM control 使用的。然后添加 Nios，选择标准处理器内核，由于本设计功能复杂，所以本设计选用的是 NiosII-f。指令 Cache 设置为 4Kbytes，数据 Cache 设置为 2Kbytes，数据 Cache 位数设置为 4Bytes。在选择 JTAG 调试模块时，选择 level 1。自定义指令时添加 Float Point Hardware 指令，使本系统支持浮点指令。

(2)添加调试接口、内部时钟和 Avalon Tri-State Bridge。SOPC Builder 使用 Avalon 接口来连接片上元件和 Avalon 主从端口。在 NiosII 的系列开发板上，要实现 NiosII 与 FPGA 片外存储器通信，用户就必须在 Avalon 总线和连接外部存储器的总线之间添加一个桥梁。

(3)添加外部存储器。本系统添加了 16bits-512KBytes 的 SRAM，64Mbytes 的 SDRAM(数据宽度 16Bits，1 个片选和地址为 ROW=12，COLUM=8)和 4Mbytes 的 Flash(地址宽度为 22，数据宽度为 8 位)。

(4)添加输入控制和输出显示模块。本系统添加了 4 个 Button 按钮和 18 个 Switch 开关作为输入控制，添加了 18 个红色 LED 显示、9 个绿色 LED 显示、8 个七段数码显示和一个 LCD 作为输出显示。

(5)添加 UART 控制器。UART 控制器波特率设为 115200，数据位 8 位，停止位 1 位，无奇偶校验。

(6)添加 ISP1362 USB 设备控制构件，它可以控制 USB 数据的输入。

(7)添加在硬件设计中实现的音频处理模块 de2_audio。这样本设计就可以实现了对音频设备的控制。

SOPC Builder 的外围模块添加完成以后，自动分配基地址，自动分配 IRQ，完整的 NiosII 系统配置及其地址映射如图 4.8 所示。

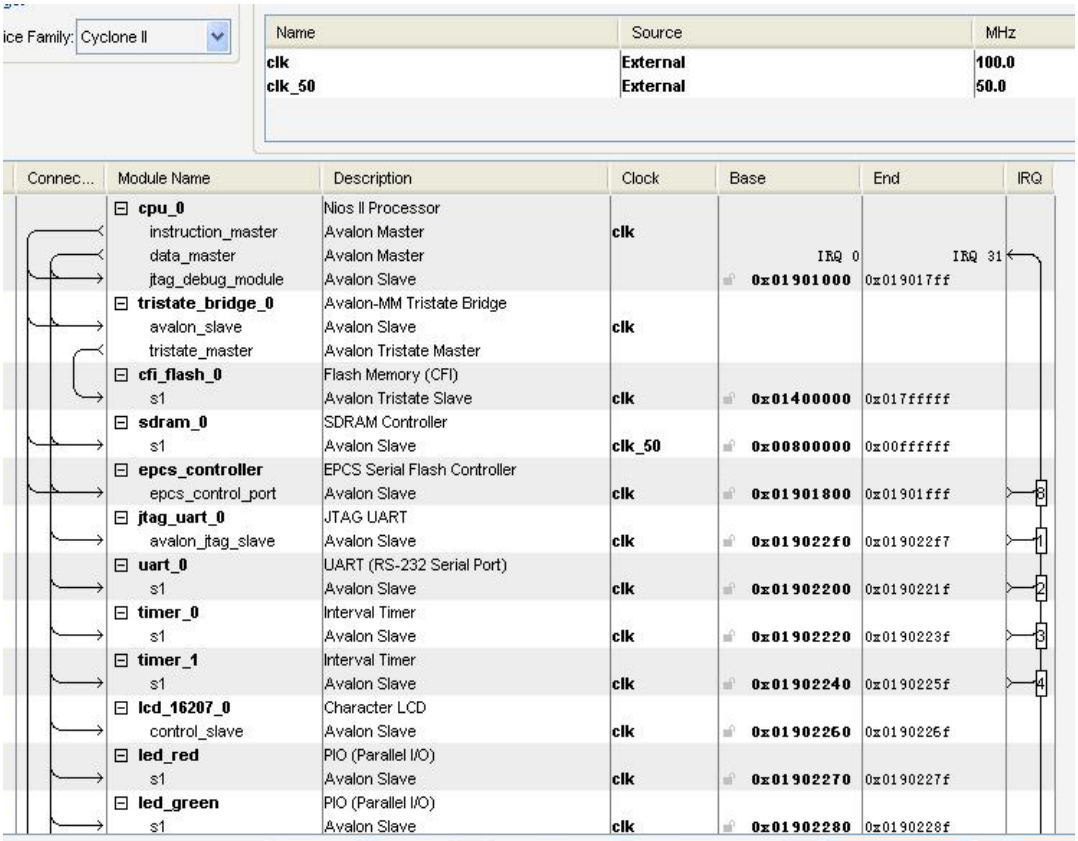

图 4.8 NiosII 系统定制结果

Fig.4.8 The result of NiosII system specify

需要注意的是在 linux 里面中断 0 是自动识别的，所以在硬件系统里面不要使用 0 号中断。

在产生 NiosII 硬件系统后，需要设计几个模块来使 CPU 正常运行，包括 SDRAM_PLL 模块，Reset_Delay 模块和 I2C_AV_Config 模块。

4.2.2 SDRAM_PLL 模块

因为要让 NiosII CPU 运行 100MHz，因此使用该模块。SDRAM_PLL 模块是由 MegaWizard 通过 MegaCore 产生的代码，它产生有三个时钟，c0 产生 SDRAM 所需要的 50MHz 相位延迟 60 度的 clock，c1 产生 100MHz 给 NiosII 使用，c2 产生分频的 25MHz。

该模块如图 4.9 所示。

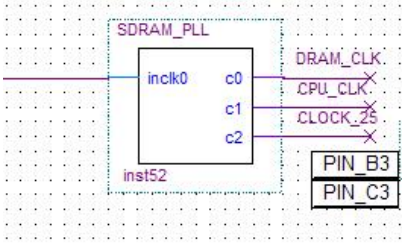

图 4.9 SDRAM_PLL 模块
Fig.4.9 SDRAM_PLL module

4.2.3 Reset_Delays 模块

Reset_Delays 模块主要起延缓 reset 时间的作用。其内容参考友晶(terasIC)公司的 Reference_Design 来完成，如图 4.10 所示。

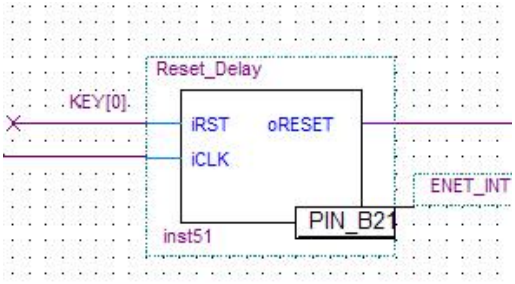

图 4.10 Reset_Delays 模块
Fig.4.10 Reset_Delays module

4.2.4 I2C_AV_Config 模块

这个模块主要是为了完成I2C总线对于音频Audio进行控制的模块。因为I2C同时对音频进行控制，所以在 I2C_AV_Config.v 中主要包含了一些对于不同情况下产生的 I2C 的逻辑控制值。如图 4.11 所示。

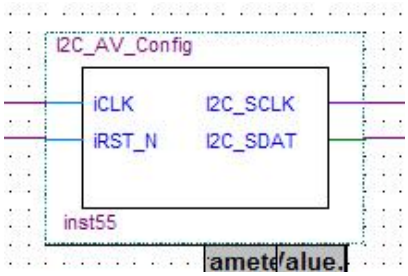

图 4.11 I2C_AV_Config 模块
Fig.4.11 I2C_AV_Config moudle

4.2.5 FIFO 模块

输入的 FIFO 由 Altera 的 MegaFunction 创建，其端口图如图 4.12 所示。输入和输出都是 16 位宽的。状态标志用来决定 Moore 机的各个状态。重要的状态信号是 rdempty、wrfull 和 wrusedw。在 rdreq 之后一个时钟周期被声明，一个 16 位数据将被挤出 FIFO，同时在输出 q 时被表达出，那时正是 usedw 下降的时刻。同样在 wrreq 之后的一个时钟周期被声明，一个 16 位数据将送入 FIFO 中，这时是 wrusedw 增加的时候。同一时钟周期里，FIFO 同时读和写允许的，wrusedw 将不会被更新。

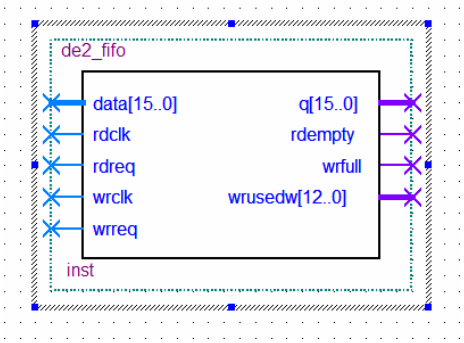

图 4.12 FIFO 的端口图
Fig.4.12 FIFO interface chart

4.2.6 USB Flash 设备控制模块

通过对 USB Flash 设备进行控制，实现了传送数据的功能。本设计在 SOPC Builder 实现 NiosII 系统时，通过导入友晶的 Reference_Design 中的 ISP 1362 构件来实现对 USB 设备的时序控制。要真正实现对 USB Flash 设备的存储与其他设备交换数据(如使用 USB 鼠标)，则需要在软件层次上根据硬件的设计来编写驱动[50]。

4.2.7 输入输出模块

输入输出模块使用了 1 个 Button 开关和 3 个 Switch 按钮，完成系统工作模式的选择和控制。

Button 开关因为直接连到了系统的 reset 端，所以复位键。

对于 Switch 的定义是：

S2，S1，S0=001：表示准备状态；

S2，S1，S0=011：表示开始；

S2，S1，S0=101：表示停止。

4.3 小结

本章主要实现了 MP3 编解码器的硬件平台，首先了解了 SOPC 的设计流程以及 NiosII 软核处理器的组成，然后通过 SOPC Builder 来创建 NiosII 硬件系统，完成了硬件上的对音频芯片和 USB 设备的控制模块的实现。

第 5 章 MP3 编解码器软件的设计与实现

在硬件平台搭好之后，将通过软件对于硬件接收到的数据进行处理。本设计是基于 μClinux 平台下运行 MP3 编解码器，对于采集到的声音数据进行编码，同时存储在 USB 设备上，而对于 MP3 进行软件解码后，又将 PCM 数据传到硬件平台，最终送到音频芯片，实现声音的回放。本章将介绍 MP3 编解软件的详细设计与实现。

5.1 MP3 编解码器软件的详细设计

本设计的 MP3 播放系统是基于 μClinux 嵌入式操作系统的，在其上运行 MP3 的编码和解码程序，同时本设计包含了 USB 驱动，在 μClinux 下运行编解码程序时，可以访问 USB 设备。编解码器的软件结构设计如图 5.1 所示：

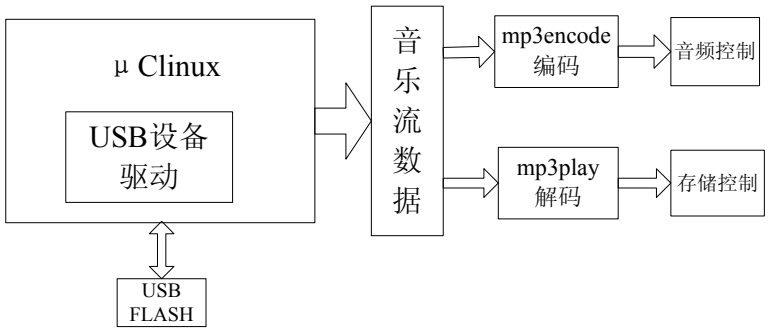

图 5.1 软件系统结构图

Fig.5.1 software system structure diagram

5.1.1 μClinux 操作系统

为了满足能浏览文件和存储在 USB flash 设备中的数据，本设计将使用一个操作系统，对于执行存储较大的数据库来说，文件系统有很大的益处。因此，本设计选择使用可以执行 MP3 编解码软件的 μClinux 系统，它的里面包含有很多的开源代码及硬件驱动。

μClinux 现在更广泛的使用，不仅 NiosII 可以实现，而且 ARM 单片机也有应用。在 μClinux 操作系统上编程有许多的资源可用，可以使用各种各样的开源的库，具有可再编程和更广泛的功能使用[51]。在本设计中添加了操作系统可以降低硬件的的复杂性。

本设计的程序作为在 μClinux 上的应用程序，要注意到本系统只是运行在用户空间。这意味着本设计不能使用中断或 NiosII 的硬件抽象层。本设计通过定义存储指针来对外设执行读和写命令的操作。

5.1.2 MP3 编码器软件设计

本设计对于 MP3 编码的过程，采用 DCT(Discrete Cosine Transform)来确定在时域音频采样的数量，同时要将其转化为频域。根据 Fourier 的分析理论，最低的几种频率总能很好的表达最初的信号和包含有主要的能量。因此本设计屏蔽了人耳不是很容易听到的声音，只留下了最重要部分的频率。然后进行 Huffman 信息编码，编码是采用最流行的方式——MP3 格式，而且是最小的采样率，这样就可以节省比特流的数量。解码也是按照同样的步骤，只不过是反向的。

为了实现对音频片段实现 MP3 的编码，又不影响实时性，因此首先在 USB 设备中将声音文件存储为.wav 文件，然后再将其编码为.mp3 文件。其执行流程如图 5.2 所示。

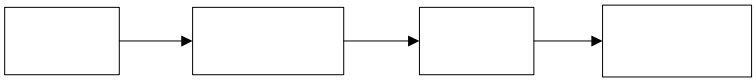

图 5.2 MP3 编码执行过程
Fig.5.2 MP3 coding implementation process

5.1.3 MP3 解码器软件设计

应用已经存在的 MPEG 解码库，尤其是使用定点解码库，不但可以提高解码速度，而且声音质量也有所保障的。本设计使用的是 μClinux 自带的解码库 mp3play，可以对采样为 8KHz、16 位的 MP3 文件进行解码。同时将其写到解码数据缓冲的 FIFO 地址，最后将其送到 LINE OUT， 这样就可以输出声音了。因为输出 DAC 是被设定 8KHz 采样率，如果想测试文件是否匹配，必须将其设定为 8KHz。

本设计中 MP3 具体的解码执行过程如图 5.3 所示。

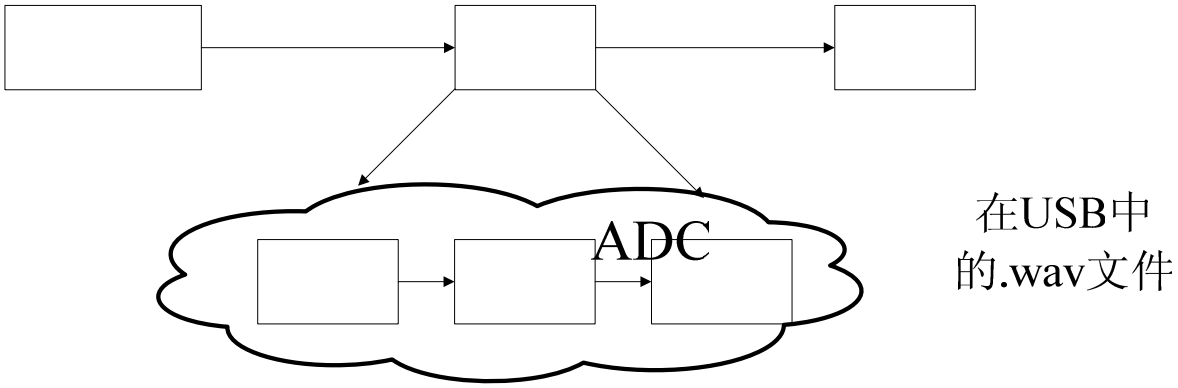

图 5.3 MP3 解码执行过程
Fig.5.3 MP3 decoding implementation process

5.2 MP3 编码器软件的实现

本设计在 Nios II CPU 上执行编码的库是基于 ShineFixed Point Mp3 V1.09 开

源编码进行修改的。这个程序最初是 ARM 的 RISC 操作系统上进行汇编操作优化的。因此为了使程序能在 Nios 运行系统上工作，本设计移除了汇编语言，并把他们转化为 C 语言。本设计在程序中未定义 RISC OS。通过在 μClinux 中增加用户应用，将程序放在 Nios 开发板上。最后设计将声音数据的重定义加入到主程序中。

5.2.1 ShineFixed Point MP3 开源文件的说明

ShineFixed Point MP3 开源代码，是定点的 MP3 压缩程序，特别适合在嵌入式系统中作为录音机使用，它无需心理声学模型，将 ShineFixed 源文件放入 mp3encode 目录下，可以看到下列文件：

(1) bitstream.c/.h：主要是提供了将格式信息写到 bit 流中去的功能；

(2) formatbits.c/.h：在编码时每一帧都将调入该文件，将组数据写入到帧文件中去，在 formatBitstream.h 文件中有更多的关于 MP3 的数据结构和音乐流的语法；

(3) huffman.c/.h：主要是哈夫曼表的相关数据；

(4) L3bitstrea.c/.h：程序中的函数主要是为了将被编码过的数据写入 bitstream；

(5) L3loop.c/.h：代码设置了缩放因子，选择最佳的量化尺寸；

(6) L3mdct.c/.h：主要功能是设置混叠的系统；

(7) L3SubBand.c/.h：计算分析滤波器的系数，使 MP3 编码符合 ISO 标准；

(8) Layer3.c/.h：Layer3 的数据结构；

(9) Main.c：主函数文件；

(10) reservoir.c/.h：由帧的开始所调用，功能是更新比特池的最大尺寸，同时检查主数据开始位是否按照格式进行设定；

(11) riscos.c：RISC OS 定义函数；

(12) tables.c/.h：包含有 MPEG 1 和 MPGE2 的缩放因子的表；

(13) types.h：描述了 wav 和 mpeg 的结构体；

(14) wave.c/.h：微软的 wave 文件储存音频数据的格式文件；

(15) mult_SA.s/mult.s：32 位乘法汇编代码，其中的 mult_SA 是为 ARM 单片机进行 long 整型乘法时设计的。

5.2.2 MP3 编码器在 μClinux 上的实现

为了使 ShineFixed Point 开源解码程序能够在 μClinux 上运行，对源文件进行了修改，同时编写适合 μClinux 系统的 makefile 文件，最后将编码程序

mp3encode 移植到 μClinux。

(1) mp3encode 文件的修改

该编码程序将录制的声音先存储为 wav 文件，在 NiosII 软核处理器上运行 mp3 编码程序。

最终修改的文件及其功能介绍如下：

a、main.c：本程序的主体，它能够从 ADC 中将声音片段记录为 wav 文件，同时它也调用了 mp3 编码函数 mp3encode()。

b、mp3encode.c：mp3 编码算法的主要“执行者”，它包含了 mp3encode()函数。

c、剩下的代码都是被 mp3encode()函数所调用的。

执行 mp3encode 程序时，先记录从 MIC 中的声音片段，存储为一个 wave 文件 TEST_0.wav，然后进行 mp3 编码，最终编码为 Mike_0.mp3。

(2) makefile 文件的编写

为了使 mp3encode 文件能够在 NiosII 下的 μClinux 系统下运行，将重新编写 makefile 文件。当执行 make 命令时，它就可以“指挥”make 如何对 mp3encode 进行编译。

下面是编写的 mp3encode 的 makefile 文件的内容：

```
# Project:    shine mp3 player

EXEC = mp3encode
OBJS = main.o mp3encode.o bitstream.o coder.o huffman.o layer3.o loop.o
wave.o

CFLAGS = -c -O5 -Wall -I/root/uClinux-dist/linux-2.6.x/include
LDFLAGSYEN = -o $@

# Rules

$(EXEC): $(OBJS)
    $(CC) $(LDFLAGSYEN) $(OBJS) -elf2flt="-s 16000"

$(OBJS): %.o: %.c
    $(CC) $(CFLAGS)    $<

romfs:
    $(ROMFSINST) /bin/$(EXEC)

clean:
    -rm -f $(EXEC) $(OBJS) *.elf *.gdb *.o

# Header dependencies
```

```
mp3encode.o: types.h
bitstream.o: types.h
huffman.o: types.h
layer3.o: types.h table1.h
coder.o: types.h
loop.o: types.h table2.h
main.o: types.h
wave.o: types.h
```

(3) MP3 编码软件加入 μClinux 操作系统

当编写好 makefile 文件后，下一步就是要把 mp3encode 作为应用程序加入 μClinux 系统了。

a、首先打开~/uClinux-dist/user 目录下 Makefile 文件，在时里面填加一行：

dir_$(CONFIG_USER_MP3ENCODE_MP3ENCODE) += mp3encode

这是表示当执行 make 编译命令时要将 mp3encode 作为一个应用程序进行编译。等号后面表示的是填加应用程序文件夹的名字。

b、在 linux 终端输入如下命令：

mkdir ~/uClinux-dist/user/mp3encode

或者直接在~/uClinux-dist/user 目录下建立一个文件夹，命名为 mp3encode。然后将 mp3encode 相关的.C 源程序和.h 头文件序拷贝到这个目录下，同时将 makefile 文件也拷贝过来。

c、编辑在~/uClinux-dist/user 下的 Kconfig，在 comment“Audio tools”里面加上如下命令：

```
config USER_MP3ENCODE_MP3ENCODE
        bool "mp3encode"
          help
          MP3 encode
```

这样在配置内核时，就可以把 mp3encode 作为一个应用程序填加到 μClinux 了。

d、在进行 make menuconfig 时进行如图 5.4 所示的选择，就可以将 mp3encode 作为一个应用程序加入到 μClinux 中了。

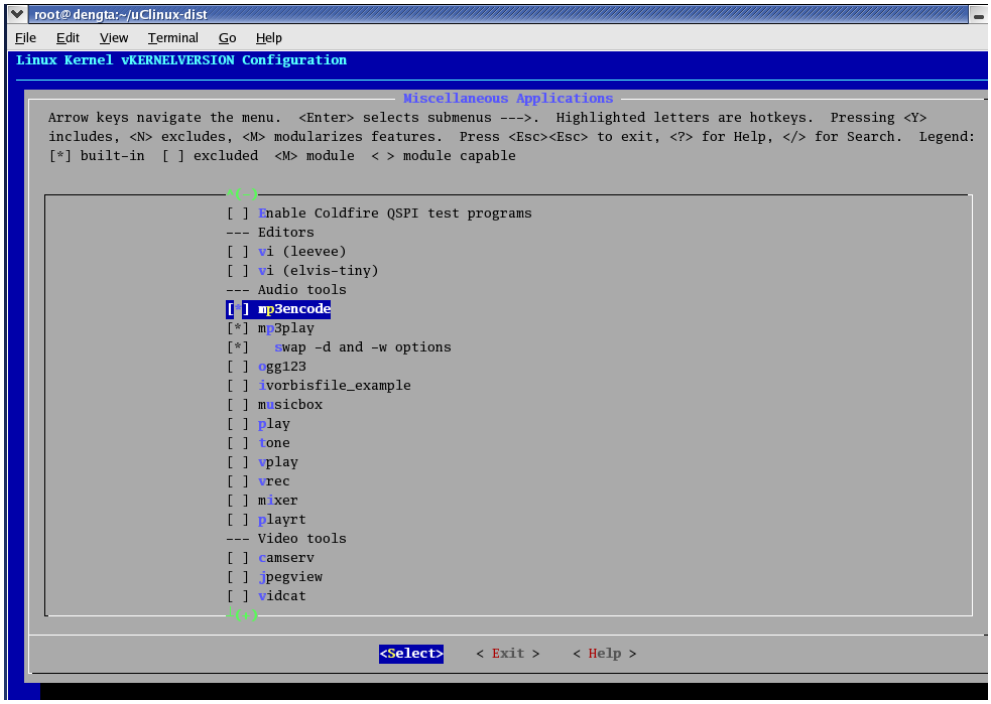

图 5.4 mp3play 和 mp3encode 作为应用程序加入 μClinux
Fig.5.4 Adding mp3play and mp3encode to μClinux as applictation

5.3 MP3 解码器软件的实现

5.3.1 MP3 解码软件的实现流程

本设计选择 mp3play 作为解码库，因为它是建立在 μClinux 系统内部的，同时支持各种 MPEG 标准(MPEG-1，2，2.5 在 Layer 1，2，3)，这是和编码设计完全匹配的。在 mp3play 库中，支持浮点和定点的操作。同时也有一些汇编优化，但这与本设计的硬件平台相冲突，需要将代码设定为不能运行在汇编代码模式。为使源代码能应用在 Nios2 中，本设计忽略了一些在其他设备上用的功能。mp3play 软件解码流程图如图 5.5 所示。

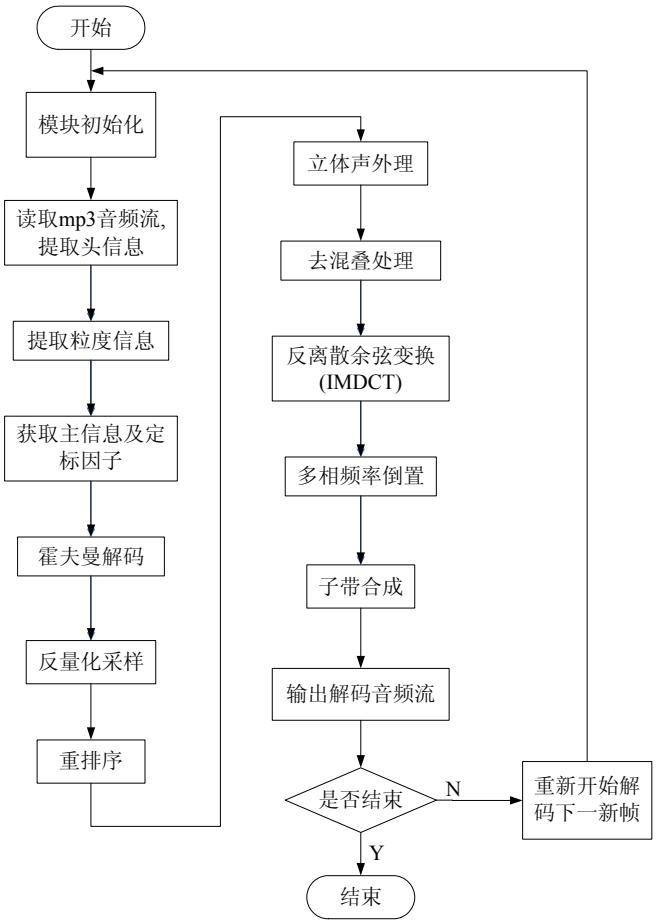

图 5.5 mp3play 软件解码流程图

Fig.5.5 mp3play software decoding flow chart

5.3.2 MP3 解码器在 μClinux 上的实现

(1)mp3play 程序

在 mp3play 目录下包含有 MP3 解码所需要的源文件和头文件，主目录下所有的文件的具体功能如下：

a、mp3play.c：解码程序的主体，它通过头文件里调用 mpgdec.c 中的函数。

在 mpeglib 目录下，包含有如下文件：

b、mpegdec.c：解压 mp3 的头信息和获取文件中的其他信息；

c、mpeg3dec.c：进行 mp3 解码；

d、mpegimdc.c：执行 DCT 功能的 Inverse，这里面包含有定点或是浮点运行方式；

e、mpegsub.c：包含定点或浮点运行方式；

f、huff.c：Huffman 解码表。

(2)makefile 文件的修改

为了使 mp3play 能在 μClinux 文件下运行，它所对应的 makefile 文件如下面所示。

```
EXEC = mp3play
OBJS  = mp3play.o http.o

CFLAGS += -Impegdec_lib -DMPEGAUD_INT -Irc4
-I/root/uClinux-dist/linux-2.6.x/include
MP3LIB    = mpegdec_lib/mpegdec.a
RC4LIB    = rc4/librc4.a
FLTFLAGS += -s 8192

ifdef CONFIG_USER_MP3PLAY_SWAP_WD
CFLAGS += -DSWAP_WD
endif

all: $(EXEC)

$(EXEC): mp3lib rc4lib $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $(OBJS) $(MP3LIB) $(RC4LIB) $(LDLIBS)

mp3lib:
    $(MAKE) -C mpegdec_lib
rc4lib:
    $(MAKE) -C rc4
romfs:
    $(ROMFSINST) /bin/$(EXEC)
clean:
    $(MAKE) -C mpegdec_lib clean
    $(MAKE) -C rc4 clean
    rm -f $(EXEC) $(OBJS) *.elf *.gdb
```

(3) mp3play 如何加入 μClinux 操作系统

同 mp3encode 一样，也要将 mp3play 加入到 μClinux 操作系统中去，作法同编码类似。但是由于在 μClinux 中已经包含了 mp3play 应用程序，只需要将 ~/uClinux-dist/user/mp3play 目录下的源文件更改为修改后的源文件就可。在进行 make menuconfig 时进行如图 5.4 所示的选择，就可以将 mp3play 作为一个应用程序加入到 μClinux 中了。

5.4 小结

本章首先讲解了 MP3 编码和解码器的详细设计结构，然后通过对 MP3 编码开源库(ShineFixed Point Mp3 V1.09)和 μClinux 下的 mp3play 解码库进行修改，按照 μClinux 系统的定义编写了相应的 makefile 文件，最后实现了 MP3 解码器在 μClinux 的移植。完成这一步就算是基本上完成了本系统的 MP3 编解码器的软件实现过程。

第 6 章 MP3 编解码器的测试与问题解决

在本系统进行设计和实现之后，就需要对系统进行必要的测试。本部分将详细介绍对于系统的软硬件的功能测试。本设计不只是进行理论研究，更侧重于实现过程，这部分也将介绍在完成设计过程中遇到的问题，同时提出解决方案。

6.1 功能测试

功能测试主要使用黑盒测试的方法，测试人员不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。

在本系统中，主要应该进行以下功能的测试：硬件驱动平台，μClinux 内核及文件系统，USB 设备驱动程序，声音录制，声音片断 MP3 编码和 MP3 解码。限于篇幅限制，这里简化了测试结果的分析，测试结果如表 6.1 所示。

表 6.1 功能测试结果表

Table 6.1 The result of function testing

测试项目	结果	分析
硬件驱动平台	正常	sof 文件配置到芯片，LED 灯正常工作
μClinux 内核及文件系统	正常	内核可以正常启动，并且可以快速的加载根文件系统
USB 设备驱动程序	正常	可以正确加载 USB，插入 USB Flash 后，识别为 sda
声音录制	正常	可以进行正常录音，存储为 wav 格式
声音片段 MP3 编码	正常	可以进行 MP3 编码
MP3 解码	正常	可以播放 MP3 文件

6.2 模块测试

6.2.1 USB 存储设备模块测试

在 μClinux 操作系统启动后，要测试 USB 存模块，输入如下命令进行挂载 USB 设备：

```
# mount –t vfat /dev/sda /mnt
```

如果挂载成功，有如下提示：

```
sd 0:0:0:0: [sda] 1013248 512-byte hardware sectors (519 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache :write through
sda:<7>usb-storage : queucommand called
unknown partition table
```

当 USB 存储设备调试好后就可以时行 MP3 编码和解码的测试了

6.2.2 MP3 编码模块测试

在 μClinux 下，输入如下命令：

#mp3encode

将 SW1 开关向上推去，这时在 mp3encode 中将识别出开始标志。将有如下显示：

```
/>mp3encode
Ready to Recode
Program Starts!!Start Recording ……
stereo 2 or mono 1:1
15 sec left
14 sec left
13 sec left
12 sec left
11 sec left
10 sec left
9 sec left
8 sec left
7 sec left
6 sec left
5 sec left
4 sec left
3 sec left
2 sec left
1 sec left
numSamplesWritten=131072
Done recording!
RAW DATA PCM,mono 16000Hz 16bit
MPEG 2 layer III,mono Psychoacoustic Model:none
Bitrate = 128 kbps De-emphasis : none
Encodeing “/mnt/test_0.wav” to “/mnt/Mike_0.mp3”
Done Encoding /mnt/Mike_0.mp3
```

进行完一次编码后，可重新开始下一次，只需要将 SW1 先向下再向上即可，如果要停止录音，则在 SW1 为向下时，将 SW2 向上推去，就可以停止录音。

6.2.3 MP3 解码模块测试

将录到的声音进行编码后，就可以进行 MP3 解码了，输入如下命令：

#mp3play Mike_0.mp3

将会有如下显示：

/>mp3play Mike_0.mp3

NewMike_0.mp3: MPEG2-III (8208 ms)

这时就会在耳机中听到声音，同时也可将 USB 存储设备接入 PC 机上，通过 PC 机上的普通音频播放器对录取的声音进行播放。这说明本设计的编码是完全正确的。

6.3 遇到的问题及解决

本设计在实现过程中不只是做在理论设计的，这部分将着重介绍在完成设计过程中遇到的最大问题。本设计的功能是有限的，这部分将列出所有在设计中遇到的问题，同时提出解决方案。

6.3.1 硬件——音频接口

关键的问题是在缓冲结构的执行对于硬件的影响。在运行实现时，它转化成状态机和相应的逻辑结构，相当复杂，主要是由于时间块的紧凑和大量的数据要检查。在不同配置可用性和大量的模块和控制信号的选择上，对于控制信号集更快更容易被理解的方法就是创建 FIFO/缓冲结构。

由于在“等待响应”信号不得不设置成异步方式，在每个 FIFO 结构运行时，两个(很大不同)时钟的同时出现，此时 FIFO 的可用性主要体现在大量数据的即时传递。尽管异步控制信号出现(不同于处理器的时钟)，大多数控制信号却不能在本系统中使用(就像数据流控制信号对于设计来说是可用的)，这主要是因为紧凑的 FIFO 结构的时序和双时钟的使用。在没有采用异步 Moore 状态机之前，没有很有效的暂停处理器，数据占有和发布被证明是大量错误的源泉。很明显，此时如果处理器的暂停能够被控制，处理器将会更缓慢的运行。

当试着验证本设计的功能性时，也有很大的问题——仿真。控制信号的本性使得建立简单的测试用例也变得困难，即使它常常被用在验证方面。而且，当系统所使用的 FIFO 进行读或者写时，问题也出现了。在第一个读响应来之前，Legacy 模式的 FIFO 需要从一个时间段中读取，即使在 FIFO 中第一次写的字将不会立即起作用。在 FIFO 显示头模式时，将不提供这种限制，但是却会对于存在的软件产生执行限制的警告。因此本设计选择 Legacy 模式为的是提高 FIFO 的执行和在 FIFO 输出时，简单的保存第一帧的非立即存在性。仿真显示了 FIFO 结构所有输入采样数据到总线是成功的，如图 6.1 所示。

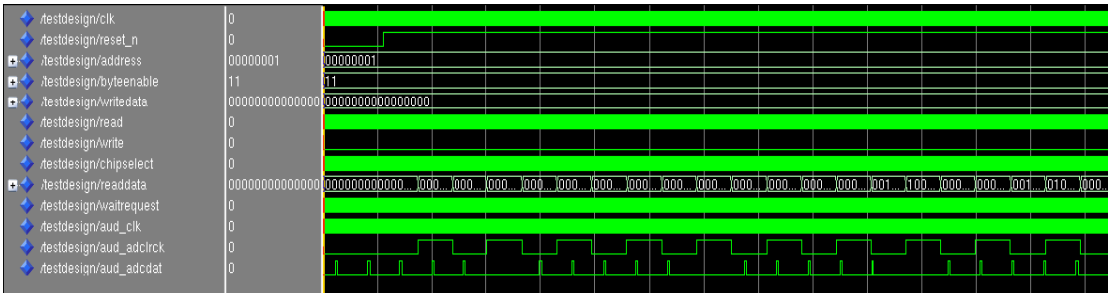

图 6.1 在 FIFO 处理过程的时序图

Fig.6.1 Timing diagram in the process of FIFO

在处理器同时读或者写时，边缘事件发生，这就需要用到异步逻辑来控制“等待响应信号”(waitrequest)信号。像期望的那样，Legacy 模式的 FIFO 属性对于本设计的功能性没有任何影响，这已经在编码过程中成功的得到验证。FIFO 的初始化也是反映了本设计的逻辑关系。不管 FIFO 的初始化输出是否被设定为 0 采样。

另一个限制因素是解码处理过程中，NiosII CPU 的 M4K 模块有限的大小，这是和 FIFO 结构对应的。本设计所执行的设备中，存在存储器的 90%被分配到 FIFO 结构中。

6.3.2 MP3 编码

编码程序是比实时大约慢 180%。通过改变滤波的宽度和 DCT 的位数，本系统能使其更接近 MP3 编码，但对应的是质量的下降。最初设想的是从 ADC 读取的前端到数据块然后送到编码器里。可是发现随着步骤的增加，编码程序不再是实时的。本设计采用编码处理 ADC 转换器直接获取的音频数据，该过程分为两个阶段：首先从 ADC 采样声音数据，然后将其做为 wav 文件放到 USB 中。生成 wav 编码是很容易实时的。然后 mp3encode 编码程序将 USB 设备中的 wav 文件编码成 mp3 文件。这种方式虽然不是实时的，但是实现 MP3 编码，而且获得了更好的声音质量。

6.3.3 MP3 解码

mp3play 库中的解码问题是为了本设计能成功地处理数据，写到 LINE OUT 中。但是本设计几乎听不到说话者的声音，同时背景有很大的干扰。曾试图写 PCM 数据到 WAV 文件代替写入到直接写入到 FIFO 来测试这个库是不是在工作。在 Matlab 上，这听起来像随机噪声。通过实时播放，本设计不能听到声音。

对于软件验证，这是非常难于在板子上调试的。曾经在没有下载 Linux zImage 时进行调试，直接通过 NiosII IDE 进行 MP3 解码调试，这样节省了很多的时间。

分清是硬件还是软件问题也是很重要的。首先在分析等待响应时，由于解码

进程是快于播放进程的，FIFO 没有发现等待响应，但等待响应仍然存在。mp3play 没有提供将 PCM 数据转化为 wav 的功能，这就难于证明 MP3 解码程序是否在正常工作。最好的验证库的方式是将解码音频变成一个 wav 文件，并在电脑上测试它，这样才能查错。

6.4 小结

本章说明了 MP3 编解码器的测试过程，详细的讲解了主要模块的测试；对于在设计和实现过程中遇到的问题，提出了解决方案。

第 7 章 结束语

本论文是基于 SOPC 的 MP3 编码解码器的设计与实现，研究了 SOPC 系统应用开发的相关知识，分析了 SOPC 系统的硬件和软件架构。本设计的 MP3 编解码器中用到的 Alter CycloneII 2C35 FPGA 处理器，对 SOPC 的特点及使用规则进行了研究，建立了基于该处理器的编解码器的硬件平台，设计并实现了 NiosII 软核处理器系统和音频数据处理的构件，同时在 NiosII 系统平台中实现了 USB 设备构件；研究了 μClinux 操作系统的工作机制，完成了 μClinux 操作系统在 SOPC 系统上的移植，同时针对 MP3 编码和解码软件设计进行了详细的分析设计，修改了开源的编码和解码库，编写 Makefile 文件，使编解码器的软件部分可以在 μClinux 上运行，完成了 MP3 编解码应用程序的硬件和软件的测试，最后对编解码器设计和实现过程中遇到的问题提出的解决方案。

论文研究了基于 SOPC 平台的 MP3 编解码器系统开发过程中使用的硬件平台设计方案，重点分析了 NiosII 软核处理器系统的建立，音频数据处理平台的构建，USB 控制设备构件的实现。

论文研究了基于 μClinux 操作系统的软件设计方案，对 μClinux 操作系统的移植过程做了介绍，重点研究了 MP3 编码程序的实现机制，对 μClinux 下解码功能的实现进行了分析。

虽然本系统实现了基于 SOPC 平台的 MP3 编码解码，可以在 USB 设备上存储音频片段，但中间存在一些不足，需要改进。

在硬件方面，在芯片控制时设定了采样率为 8KHz，这样在播放时也是只能播放这种采样率的录音的声音片段，不可以当作常用 MP3 播放器来使用。

在软件方面，因为没有直接对演讲者的声音直接进行 MP3 编码，虽然保证了实时性，但步骤繁琐，并且解码是基于软件平台建立的，解码速度比较慢。未来的研究可以直接在硬件平台上实现 MP3 硬件编码，增强实时性，简化步骤。同时没有良好的操作界面，不能显示当前的播放状态，这些应在以后的研发中实现。

参考文献

1. Nicolas Gac,PStephane Mancini,PMichel Desvignes. Hardware/software 2D-3D backprojection on a SoPC platform[J], symposium on Applied computing, 2006, 12(6):222–228
2. 王燕.基于OpenRISC的音频解码器软硬件协同设计[D],浙江大学,2007
3. 周颖,唐伦,陈前斌.MPEG标准中的音频编码与应用[J],电信工程技术与标准化,2007,19(7):25-28
4. 杜比实验室.杜比(Dolby Digital)高清音频技术漫谈[J],家庭影院技术,2007, 9(9):48-49
5. 高海鹏.新一代音频格式Dolby Digital plus/Dolby TrueHD 及DTS HS简介(六)[J],TWICE消费电子商讯,2007,7(9):16-18
6. 黄成辉,邓裕昭.MPEG-2中TS码流包结构和功能应用[J],中国有线电视,2008, 15(3):246-250
7. 刘艳,陈永恩.AVS数据视频解码中反量化/反变换的FPGA实现[J],现代商贸工业,2008, 20(3):293-294
8. 恩智浦技术创造完美的高清图像观赏体验[J],电子与电脑,2007,23(10):35
9. Octasic 多核心媒体网关 DSP 基于 IP 的语音视频和数据应用[J],基础电子, 2008,15(5):52
10. TI 公司.面向便捷高清视频应用的 DaVinci DM355 处理器[J],世界电子元器件，2008,7(4):62-66
11. 张丽杰,吕少中.方舟CPU体系结构及其嵌入式SOC[J],现代电子技术,2005, 12(6):48-50
12. 支锦扬.基于Leon2的MP3播放器的设计与实现[D],电子科技大学,2006
13. 宋奇刚,魏小义.霍夫曼编码器的设计及在MP3解码中的应用[J],今日电子, 2005,12(3):49-50, 52
14. Raut, Ketan J,Shiriramwar, Shashank S. MP3 Portable Player System-Level Glue Logic on FPGA[J], Microelectronic Systems Education,2007,12(8): 97-98
15. Jason Yu, Guy Lemieux,Christpher Eagleston.Vector processing as a soft-core CPU accelerator[J],Proceedings of the 16th international ACM/SIGDA sympo -sium on Field programmable gate arrays,2008,10(3):222-232
16. Martin Labrecque, PPeter Yiannacouras, PJ. Gregory Steffan. Custom code generation for soft processors [J], ACM SIGARCH Computer Architecture News, 2007,12(35):9-19

17. Li Nianqiang; Wang Yutai; Zhang Lu .Research on LCR Impedance Measurement Based on SOPC[J], Electronic Measurement & Instruments, 2007,6(l): 200-204
18. 王建校,危建国.SOPC设计基础与实践[M],西安:西安电子科技大学出版社, 2006,132-340
19. 徐婷婷.基于SOPC的MPEG4视频解码器的设计与优化[D],华东师范大学, 2007
20. 周学功,彭澄廉.SOPC Builder中IP构件的设计与实现[J],2005,26(2):293-295, 338
21. Rousseau, F. Petrot, and A. Jerraya.Prototyping Multiprocessor System-on-Chip Applications: A Platform-Based Approach[J], Distributed Systems Online, 2007,8(3):23-28
22. 高枫,王玉松.基于NiosII自定制Avalon设备的设计与实现[J],中国测量技术, 2007,33(4) :105-108
23. William Thies Vikram Chandrasekhar and Saman Amarasinghe. A Practical Approach to Exploiting Coarse-Grained Pipeline Parallelism in C Programs[J], International Symposium on Microarchitecture, 2007,12(4):23-34
24. Sascha Uhrig, Jorg Wiese. jamuth: an IP processor core for embedded Java real-time systems[J], Java technologies for real-time and embedded systems, 2007,7(4):230-237
25. 潘松,黄继业,曾毓.SOPC技术实用教程[M],北京:清华大学出版社, 2005,191-380
26. 周立功.SOPC嵌入式系统实验教程[M],北京:北京航空航天大学出版社,2006, 57-70
27. 李兰英.NiosII嵌入式软核SOPC设计原理及应用[M],北京:北京航空航天大学出版社,2006, 57-110
28. 余腊生,洪飞.μClinux在NiosII嵌入式平台上的移植研究[J],微计算机信息(嵌入式与SOC),2007,23(9):18-19,278
29. 蒋巍泉,王前,吴淑泉.基于NiosII的μClinux研究与应用[J],科学技术与工程, 2006, 6(8):1069-1071,1075
30. 陈福生.基于μClinux的网络摄像机软件设计与实现[D],电子科技大学,2006
31. 廖广益.基于FPGA的USB接口设计[D],昆明理工大学,2006
32. 唐如鸿.基于NiosII软核CPU嵌入式消息转发系统的设计与实现[D],西南交通大学,2006
33. Roger Moussali, Nabil Ghanem,Mazen A.R.Saghir.Supporting multithreading in configurable soft processor cores[J],Compilers,architecture,and synthesis for

embedded systems,2007,6(3):155-159
34. Michael Benjamin,David Kaeli,Richard Platcow. Experiences with the Blackfin architecture in an embedded systems lab[J], Computer architecture education, 2006,4(2):12-23
35. 刘六彬.嵌入式视频监控系统中μClinux的应用研究[D],浙江大学,2003
36. PRichard Hough, Praveen Krishnamurthy, Roger D. Chamberlain,Ron K. Cytron,John Lockwood,Jason Fritts.Empirical performance assessment using soft-core processors on reconfigurable hardware[J], Experimental computer science,2007,10(2):18-25
37. Hall,Tyson S.Hamblen,James O.Using an FPGA Processor Core and Embedded Linux for Senior Design Projects[J], Microelectronic Systems Education, 2007, 13(2): 33-34
38. 蔡英.嵌入式Linux下的MP3播放器的研究与实现[D],昆明理工大学,2007
39. Jordana Seixas, Edson Barbosa, Stelita Silva.Aquarius: a dynamically reconfigurable computing platform[J]，Integrated circuits and systems design, 2007,7(2):171-176
40. 乔光军.基于μClinux的语音编解码器的实现和优化[D],天津大学电子信息工程学院,2006
41. 袁海林.WM8731的I2C配置模块的FPGA设计[J],电脑知识与技术,2007, 7(6):1622-1623
42. 张君安.嵌入式指纹数据识别系统研究[D],华东师范大学,2007
43. Andrew R. Putnam, Dave Bennett, Eric Dellinger, Jeff Mason, Prasanna Sundararajan. CHiMPS: a high-level compilation flow for hybrid CPU-FPGA architectures[J], Proceedings of the symposium on Field programmable gate arrays, 2008,6(6):261-261
44. 曾璇.基于NiosII软核处理器的嵌入式MP3播放器[J],仪器仪表标准化与计量,2007,5(4):46-48
45. 周进.uC/GUI在基于Nios的嵌入式系统中的研究与实现[D],南京理工大学,2006
46. 陈小毛,陈尚松.32位软核处理器NiosII的以太网接口设计与实现[J],电子测量技术,2007,30(1):150-151,187
47. 蒋学鑫.MP3实时编解码系统的研究与开发[D],电子科学技术大学,2007
48. 张聪.MP3解码器系统在定点DSP上的实现[J],武汉工业学院学报,2007, 26(2):71-74
49. 毛利萍.MP3音频编解码运算中IMDCT算法研究及其FPGA实现[D],华东师范

大学,2007

50. 杜小平,周顺平,赵秋荣.一个Linux下makefile文件的分析[J],现代电子技术,2004,10(7): 61-63
51. 黎东涛.一种基于语音识别SoC调试的JTAG接口设计[J],扬州大学学报(自然科学版),2005,8(2):45-48

致 谢

时间过得真快，转眼间，两年的研究生生活即将结束。在此期间，我学到了很多专业知识和生活哲理，它将使我受益无穷。我的老师、师兄和师弟们给予我的支持和鼓励，伴随我走过每一步。在此，我忠心地向他们表示深深的谢意！

首先，我要衷心地感谢我的导师李晶皎教授。李老师为人朴实亲切，平易近人，营造了一种和谐、宽松的学术氛围。李老师，知识渊博、爱岗敬业，始终站在学科的前沿，把握研究方向，用先进的学术思想引导和激励着我。两年的师生相处，使我不仅在学业上有所收获，更重要的是他培养了我严谨的治学态度和精益求精的科学精神，使我受益终生。

我们实验室是一个融洽、欢乐的集体：非常感谢王爱侠老师、马学文老师、闫爱云老师对我的指导和帮助；感谢已经毕业的王文超、汪兆飞、王春雷、杨天杰、李海朋、邱方、郝建义、黄日文、姚猛等师兄和黄继娥、景巍、刘晓坤等师姐指点和帮助；感谢于伟光、吴秀丽、曹科研、王越、谢志豪、王坤、徐建华、范晓亮、郑旭等人的帮助，感谢你们在我的课题完成过程中给予我极大的指导和帮助。

感谢在我成长和课题的过程中给予我帮助的老师和朋友，这里就不一一列举，谢谢你们！

我还要真诚的感谢我的家人。感谢我的父母，感谢他们给予我生命、抚育我成长；感谢我的女朋友在课题完成过的支持，在我烦躁与困惑时给予我关怀和宽慰。

最后感谢所有参加论文评审的各位专家，感谢您们在百忙之中对我的论文给予批评指正。