学 位 论 文

MP3 软件解码器的研究与实现

（题　　目）

李 菁 菁

（作者姓名）

指 导 教 师　滕国库　副教授

申请学位级别　工 学 硕 士

专 业 名 称　计算机应用技术

学位授予单位　大 连 海 事 大 学

2006 年 3 月

摘 要

MPEG（Moving Picture Experts Group）是运动图像专家组的英文缩写。MP3是MPEG Audio Layer3的缩写，即MPEG第三层音频编码标准，使用MP3标准对音频数据编码既可以获得较大的音乐数据压缩比，又可以得到较好的音乐回放质量。因此，它在音乐迷中流行开来，正形成风靡全球之势。MP3 解码的实现有两种方案，硬件解码和软件解码。目前市场上流行的便携式MP3播放器都是硬件解码，采用专用的解码芯片。解码芯片是决定其音质好坏的关键，质量好的解码芯片所表现出来的音质是那些质量低劣的解码芯片难以相比的。但是高质量的解码芯片的价格也是高昂的，导致便携式MP3播放器的成本偏高。

基于以上背景，我们提出了基于 ARM（Advanced RISC Machines）微处理器的软件解码方案，在降低硬件成本的基础上保证高质量的播放效果。论文首先简要介绍MPEG系列国际标准音频编解码技术的发展现状。其次，介绍了MP3音频压缩标准所使用的子带编码、心里声学模型、自适应窗口切换、哈夫曼编码、比特池等关键技术。再次，介绍了MP3的解码流程、深入研究了解码流程中各模块国际标准解码算法、阐述了其软件实现。然后设计了MP3解码系统的硬件平台，包括微处理器的选型、CODEC（立体声解码电路）的选型、数据接口 IIS(Inter-IC Sound bus)总线的介绍等。最后分析了当前系统的完成情况和所存在的不足，以及今后可以改进和完善的地方。

关键字：MPEG；MP3；IIS总线；ARM

Study and Realization of MP3 Software Decoder

Abstract

MPEG is short for Moving Picture Experts Group. MP3 is short for MPEG Audio Layer3.We can obtain higher compression ratio of music data using MP3 technology, and at the same time we could receive good playback quality. So it becomes popular among music fans and form fashionable global tendency. In order to decode MP3 file, there are two schemes, decoding using hardware and decoding using software. At present the portable MP3 players that the market prevails are all decoding with hardware adopting special purpose decoding chip. The decoding chip determines the tone quality. Good quality decoding chip provides good tone quality, while bad quality decoding chip provides bad tone quality. But the price of good quality decoding chip is high, it causes that the cost of portable MP3 player is high.

Because of the above background, we have proposed the software decode scheme based on ARM (Advanced RISC Machines) microprocessor,that guarantees the high tone quality while reducing the hardware cost. The thesis firstly introduces the current situation of the development of MPEG series international standard and audio code technology briefly. Secondly, introduces MP3 audio compress standard including subband filter bank, psychoacoustic model, adaptive window switch over, Huffman coding, bit reservior, etc. Thirdly, recommends to decode procedure, and studies the international standard algorithms of each modules of the decoding procedure, and explains how to realize MP3 decoding procedure. Fourthly, we design the hardware platform of MP3 decoding system including how to choose the microprocessor and CODEC,and what is IIS bus .At last we analysis the performance of the system, the deficiencies existing at present and how to improve them.

Key Words: MPEG; MP3; IIS BUS; ARM

大连海事大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：本论文是在导师的指导下，独立进行研究工作所取得的成果，撰写成博士/硕士学位论文 “MP3软件解码器的研究与实现” 。除论文中已经注明引用的内容外，对论文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文中不包含任何未加明确注明的其他个人或集体已经公开发表或未公开发表的成果。

本声明的法律责任由本人承担。

论文作者签名：李菁菁　2006年3月15日

学位论文版权使用授权书

本学位论文作者及指导教师完全了解“大连海事大学研究生学位论文提交、版权使用管理办法”，同意大连海事大学保留并向国家有关部门或机构送交学位论文的复印件和电子版，允许论文被查阅和借阅。本人授权大连海事大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，也可采用影印、缩印或扫描等复制手段保存和汇编学位论文。

保密□，在_____年解密后适用本授权书。

本学位论文属于：　保密□

不保密□（请在以上方框内打“√”）

论文作者签名：李菁菁　导师签名：

日期：2006 年 3 月 15 日

第1章 绪论

1.1 MPEG 音频标准

MPEG (Moving Picture Experts Group) 是动态图像专家组织的英文缩写，是一个致力于数字视频、音频技术发展及标准化的杰出组织。它是ISO(international standard organization) 与 IEC(international electronic committee)在 1988 年联合成立的。该组织于 1992 年正式推出了具有 1.5Mbit/s 数据传输率的数字存储媒体运动图像及其伴音的 MPEG－Ⅰ的标准草案，即 ISO/IEC11172，于 1993 年正式通过。该标准规定了数字音视频编码的国际标准，主要包括三个方面：系统、视频和音频。视频压缩仅仅包含画面信息，音频压缩包含声音信息，系统实现音频和视频的同步。所有播放 MPEG 视频压缩数据和音频压缩数据所需的时钟信息都包含在系统流中，其中音频编码可以独立使用。[1]

MPEG－Ⅰ音频编码标准支持采样率为 32KHz、44.1KHz、48KHz 的单声道、双声道、立体声或联合立体声的两个声音通道的编码格式。由于该压缩算法可以把 CD 音质的两个通道共包含 1.4Mbit/s 的数据流压缩到 128kbit/s，且仍然保持高保真的声音，使其很快得到国际认可。随着多媒体技术和网络通信技术的进一步发展，ISO/IEC 在 1994 年又推出了 ISO/IEC13818，MPEG-Ⅱ运动图像及其伴音通用压缩标准。该标准最初的目的是在 MPEG-Ⅰ兼容的基础上实现低比特率压缩和多声道扩展，后来为了适应演播电视的要求，开始致力于定义一个可以获得高质量的多声道音频标准，它可提供左、右、中及两个环绕声道、一个加重低音声道，和多达 7 个伴音声道。同时可以将单声道编码比特率降低到 8kbit/s。

MPEG-Ⅰ和 MPEG-Ⅱ音频编码都提供了三个压缩层。层Ⅰ是简单压缩，它是一种听觉心理声学模型下的亚抽样编码。层Ⅱ加入了更高的精度。层Ⅲ，是现在流行的 MP3 音乐格式，加入了非线性量化、哈夫曼编码和其它实现低速率高保真音质的先进技术，它可以把一个 1.4Mbit/s 的立体声双声道数据流压缩为 32Kbit/s～384Kbit/s 且保持高保真的音质。依次下来的等级提供更高的质量和越来越高的压缩率，但要求计算机有越来越强的压缩计算能力。典型数据为，层Ⅰ的目标是每个通道 192Kbit/s，层Ⅱ的目标是每个通道 128Kbit/s，层Ⅲ的目标是每个通道 64Kbit/s。层Ⅱ要达到 64Kbit/s 时不如层Ⅲ效果好，而在 128Kbit/s，

层Ⅱ和层Ⅲ的效果接近，而且都比层Ⅰ效果好[2]。

MPEG-Ⅰ Layer 3 即 MP3 这种以压缩比和保真度出众的音频文件格式让许多个人用户青睐有加，同时也让 IT 厂商看到了潜在的市场。作为 MP3 文件格式最大优势一文件小的体现，MP3 随身听已经成为一种时尚的标志出现在街头人们的挎包里、腰带上或胸前。

1.2 课题的意义和本文所做的工作

目前市面上流行的 MP3 播放器一般都是采用专用的解码芯片对 MP3 文件进行硬件解码。硬件解码器在国内外已经是比较成熟的产品。硬件解码器解码质量的好坏必然受到解码芯片性能好坏的影响，性能良好的解码芯片所表现出来的音质是那些性能低劣的解码芯片难以相比的[3]。但是高质量的解码芯片的价格也是十分高昂的，这就是高保真音质 MP3 播放器价格居高不下的原因。在本系统中，我们采用基于 ARM 微处理器的软件解码方案，降低成本并保持高质量的声音效果，而且与硬件解码器相比系统升级具有更高的灵活性。

本文的研究工作主要包含两部分内容。

第一部分是对 MPEG 音频第三层解码算法的研究与软件实现。介绍了 MP3 音频压缩使用的几个关键技术：混合滤波器组、心里声学模型、自适应窗口切换技术、哈夫曼编码技术、弹性比特存储技术并剖析了 MP3 的帧结构。重点研究了解码算法，在实现过程中对音频解码（44.1KHz 采样频率，压缩到 64Kbit/s 每通道的编码速率，双声道）采用 MPEG－Ⅰ的标准，解码过程包括数据流同步及帧头、边信息的提取、主数据的提取、哈夫曼解码、反量化、立体声处理、重排序、混叠重建、逆向离散余弦变化、频率反转和子带合成几个步骤。

第二部分是系统硬件部分的实现。介绍了各功能模块的元器件的选型，各功能模块之间的接口设计，以及各功能部件拟定的工作方式。

第2章 音频压缩的基本原理和MPEG－Ⅰ音频标准

2.1 音频压缩的基本原理

音频压缩技术指的是对原始数字音频信号流（PCM 编码）运用适当的数字信号处理技术，在不损失有用信息量，或所引入损失可忽略的条件下，降低其码率，也称为压缩编码。它必须具有相应的逆变换，称为解压缩或译码[4]。

数字音频信号得以压缩的理论依据首先是原始信号本身存在着冗余度，其次是利用人类的听觉感知系统对某些失真不敏感的特性，即人耳的心理声学模型。信号的冗余度又包含两个方面：一是客观冗余度，它是可以计算的，同时用来确定音频信号的某些数字上可预测特性的数量，如周期波形；二是主观冗余度，指由于人耳的听觉特性，音频信号中包含着被人耳忽略的分量。人耳的听觉特性涉及心理声学和生理声学的问题。近代音频压缩编码的核心是依据心理学模型去除信号中的主观冗余，以及采用耦合和声道重组等技术消除客观冗余，同时结合复杂的近代数学信号处理算法，去除音频信号本身冗余度，从而达到音频压缩的效果。

2.1.1 心理声学模型

人的听觉系统是一个非常复杂的系统。对于人类听力感知的研究，其范围从人耳的生理设计，到大脑对听觉信号的解释。声音只是一个客观存在物质的学术概念，而心理声学模型则揭示了人们对于听到的一切的主观反应，是在有关听觉的关系中最终的仲裁者，因为它是人们对声音的一切反应。心理声学模型试图用所有与听觉感知有关的科学的、客观的和物理的特性所引起的心理和生理上的反应，使得听觉能感知这些特性，并和谐统一起来。

1. 掩蔽效应

一个声音的听觉感受受到另一个声音影响的现象叫做“掩蔽效应”。前者称为被掩蔽音，后者称为掩蔽音。被掩蔽音单独存在时的听阈分贝值称为绝对听阈。在掩蔽情况下，必须加大被掩蔽音的强度才能被人耳听见，此时的听阈称为掩蔽听阈。

2. 绝对听阈

图 2.1 绘出了人耳的听阈曲线，人耳对不同频率的敏感程度差别很大。其中，

图 2.1 绘出了人耳的听阈曲线，人耳对不同频率的敏感程度差别很大。其中，

第2章 音频压缩的基本原理和MPEG－Ⅰ音频标准

2.1 音频压缩的基本原理

音频压缩技术指的是对原始数字音频信号流（PCM 编码）运用适当的数字信号处理技术，在不损失有用信息量，或所引入损失可忽略的条件下，降低其码率，也称为压缩编码。它必须具有相应的逆变换，称为解压缩或译码[4]。

数字音频信号得以压缩的理论依据首先是原始信号本身存在着冗余度，其次是利用人类的听觉感知系统对某些失真不敏感的特性，即人耳的心理声学模型。信号的冗余度又包含两个方面：一是客观冗余度，它是可以计算的，同时用来确定音频信号的某些数字上可预测特性的数量，如周期波形；二是主观冗余度，指由于人耳的听觉特性，音频信号中包含着被人耳忽略的分量。人耳的听觉特性涉及心理声学和生理声学的问题。近代音频压缩编码的核心是依据心理学模型去除信号中的主观冗余，以及采用耦合和声道重组等技术消除客观冗余，同时结合复杂的近代数学信号处理算法，去除音频信号本身冗余度，从而达到音频压缩的效果。

2.1.1 心理声学模型

人的听觉系统是一个非常复杂的系统。对于人类听力感知的研究，其范围从人耳的生理设计，到大脑对听觉信号的解释。声音只是一个客观存在物质的学术概念，而心理声学模型则揭示了人们对于听到的一切的主观反应，是在有关听觉的关系中最终的仲裁者，因为它是人们对声音的一切反应。心理声学模型试图用所有与听觉感知有关的科学的、客观的和物理的特性所引起的心理和生理上的反应，使得听觉能感知这些特性，并和谐统一起来。

1. 掩蔽效应

一个声音的听觉感受受到另一个声音影响的现象叫做“掩蔽效应”。前者称为被掩蔽音，后者称为掩蔽音。被掩蔽音单独存在时的听阈分贝值称为绝对听阈。在掩蔽情况下，必须加大被掩蔽音的强度才能被人耳听见，此时的听阈称为掩蔽听阈。

2. 绝对听阈

图 2.1 绘出了人耳的听阈曲线，人耳对不同频率的敏感程度差别很大。其中，

对 2KHz 到 4KHz 范围内的信号最为敏感，幅度很低的信号都能被人耳听到。而在低频区和高频区，能被人耳听到的信号幅度要高得多。

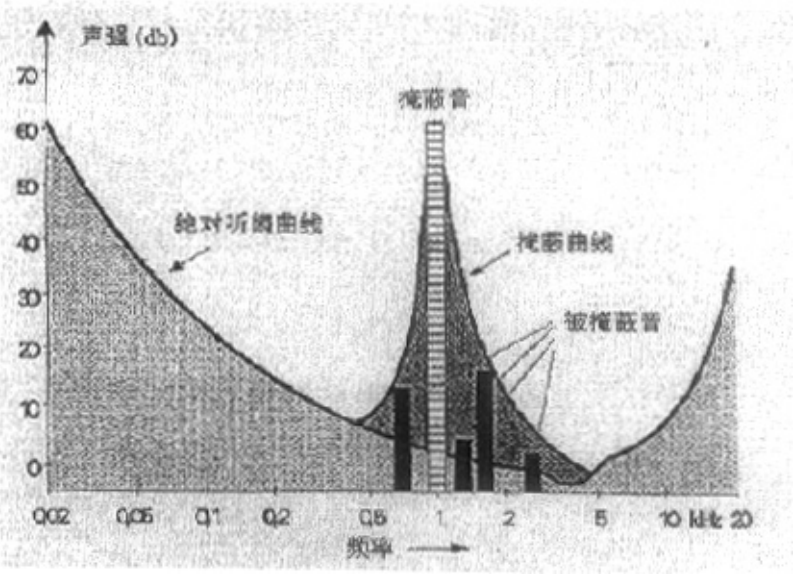


图2.1 掩蔽听阈图

Fig. 2.1 Masking threshold figure

3. 掩蔽听阈

掩蔽听阈分为频域掩蔽和时域掩蔽。

频域掩蔽：在同一时间内，频域中的一个强音会掩蔽与之同时发声的附近的弱音。它有如下的两个规律：一，低频纯音可以有效地掩蔽高频纯音，但高频纯音对低频纯音的掩蔽作用则很小。二，在距离强音较远处，绝对听阈比该强音所引起的掩蔽阈值更高。此时，噪声的掩蔽阈值应该取绝对听阈。

如图 2.1 所示，在 1KHz 处有一单频音，此处的掩蔽曲线如图所示，根据掩蔽曲线，位于 1KHz 单频音附近的强度较弱（位于掩蔽曲线包络下）的几个单频音被掩蔽，人耳无法听到这些声音，因此从心理学的角度讲，这些被掩蔽的声音应该是冗余的，在进行音频压缩编码时应该去除。

时域掩蔽：除了同时发出的声音之间有掩蔽现象以外，在时间上相邻的音之间也有掩蔽现象。我们称之为时域掩蔽。一个信号可以被以前发出的噪声（或另

一个信号）掩蔽，这就是前掩蔽。另外，一个信号也可以被以后发出的噪声（或另一个信号）掩蔽，这称为后掩蔽。当频率差别减小时，同时掩蔽增加，而当时间差异减小时，暂时掩蔽增加。图 2.2 给出了暂时掩蔽阈值随时间变化的曲线。由于声音同时出现，同时掩蔽要强于前/后掩蔽。

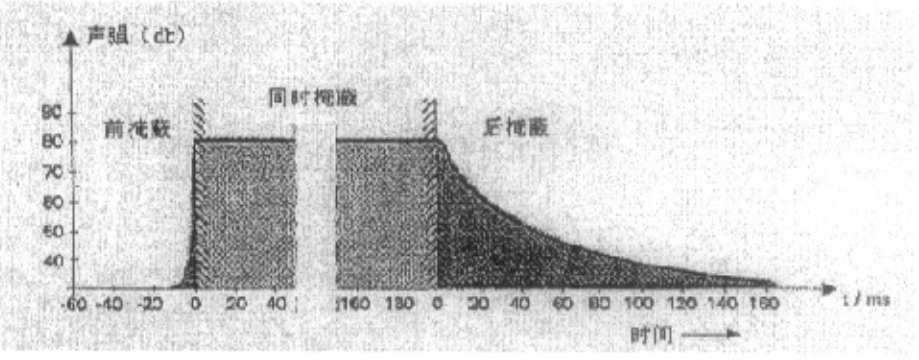


图2.2 时域掩蔽图

Fig. 2.2 Time domain masking figure

产生时域掩蔽的原因在于人的大脑处理信息需要花费一定的时间。[5]

MPEG－Ⅰ标准建议了两种决定最小掩蔽阈值的心理声学模型。只在编码器中使用该模型，而且简单的编码器不使用心理声学模型。对音频信号进行压缩编码时，在比特分配中利用最大信号电平和掩蔽阈值之间的差异来设定量化级，去除冗余信息，以达到高压缩比、低失真率的效果。

2.1.2 感知编码原理

任何数据压缩系统的目的都是降低数据传输速率。虽然降低采样率可以降低数据传输速率，但取样定理限制了采样频率的降低。另一种方法是减少字长，这样做的副作用是音频信号的动态范围减小了，同时也增加了量化噪声。比较可行的方法是利用心理声学模型。感知编码器在取样速率不变的情况下，根据信号的情况有选择的减少字长，并通过掩蔽及其他方法尽可能减少增加的量化噪声的影响。感知编码器首先分析输入信号的频率和振幅，然后将其与人的听觉感知模型进行比较。编码器用这个模型去除音频信号的不相干部分及统计冗余部分。尽管

这种方法是有损的，但是人耳却感觉不到编码信号质量的下降。

感知编码器的有效性部分源至采用了自适应的量化方法。在 PCM 中，所有的信号都分为相同的字长，感知编码器则是根据可听度来分配所使用的字长，重要的声音就分配多一些位数来确保可听的完整性，而对于轻言细语的编码位数就会少一些，不可听的声音就根本不进行编码，从而降低了比特速率。

一般感知编码采用两种比特分配方案。一种是前向自适应分配方案，所有的分配都在编码器中进行，这个编码信息也包含在比特流中。前向自适应编码的一个突出优点是在编码器中采用了心理声学模型，它只是利用编码数据完整的重建信号。当改进了编码器中的心理声学模型时，可利用现有的解码器来重建信号。在后向自适应分配方案中，比特分配信息可以直接从编码的音频信号中推导出来，不需要编码器中详细的分配信息，分配信息也不占用比特位。然而在解码器中的比特分配信息是根据有限的信息推导出来的，精度必然会降低。另外解码器相应也比较复杂，而且不能轻易地改变心理声学模型[6]。

2.1.3 数据压缩编码

数据压缩编码是以比较少的比特来表示音频信号，同时减少量化误差。现有的数据压缩编码主要有两种：时域编码方法和频域编码方法。时域编码方法对取样值采用预测的方法来表示音频信号的全带宽，导致量化误差的频谱覆盖整个音频带宽。同时，时域编码并没有充分挖掘掩蔽的潜力，未能达到最佳的压缩效果。频域编码方法在频域里分析、编码信号，利用基于人耳的心理声学模型来对量化噪声进行处理，虽然对于音频信号本身是有损的，但是人的听觉系统却感觉不到，因此具有很高的压缩质量。

MPEG－Ⅰ标准采用了两种频域编码器：子带编码器和变换编码器。子带编码器采用为数不多的子带，处理时间上相邻的取样值，而变换编码器使用很多子带处理频率上相邻的取样值。子带编码有着较好的时间分辨率和较差的频率分辨率，变化编码器正好相反。MPEG－Ⅰ标准结合了子带编码和变化编码两种方法，对音频信号进行压缩。

1. 子带编码

子带编码理论最早是由 Crochiere 等于 1976 年提出的。其基本思想是将信号

分解为若干子频带内的分量之和，然后对各子带分量根据其不同的分布特性采取不同的压缩策略以降低码率。

子带编码就是将一个短周期内的连续时间取样信号送入滤波器中，滤波器组将信号分成多个（最多 32 个）限带信号，以近似人耳的临界频段响应。通过分析每个子带的取样值并与心理声学模型进行比较，子带编码器基于每个子带的掩蔽阈值能自适应的量化取样值[7]。

子带感知编码器利用数字滤波器组将短时的音频信号分成多个子带。在 MPEG－Ⅰ算法中，利用快速傅立叶变换将信号变换到频域分析其能量，利用心理声学模型来分析这些数值，给出这组数据的合成掩蔽曲线，从而去除被掩蔽的冗余取样值。

2. 变换编码

变换编码与子带编码的不同之处在于该技术对一段音频资料进行“线性”的变换，对所获得的变换域参数进行量化、传输，而不是把信号分解为几个子频段。

在变换编码中，将时域音频取样值变换到频率域。编码器的变换方法可以采用离散傅立叶变换（DFT）或改进的离散余弦变换（MDCT）。这些变换根据信号的短时功率谱对变换域参数进行合理的动态比特分配可以使音频质量获得显著改善，而相应付出的代价则是计算复杂度的提高。从信息论的角度看，变换编码减少了信号熵，从而可以进行有效编码。块长度越长，变换编码的频率分辨率越高，但损失了时间分辨率。许多编码器将时间上连续的数据块重叠 50％来增加时间分辨率。MPEG－Ⅰ算法中采用改进的离散余弦变换（MDCT）进行信号的时域转换。

3. 哈夫曼编码

哈夫曼（Huffman）编码算法是一种熵编码，哈夫曼熵编码的基本思想是对于一组给定概率的符号，给出现概率较大的符号以长度较短的码字，概率小的符号以较长的码字，从而相对于等长编码，获得最短的平均码字长度，因此哈夫曼编码是可变长编码（VLC， variable-length coding）。它是一种无损压缩方法，因此可以在数据传送时减少数据量的同时而不会影响音质。

MPEG－Ⅰ音频算法在层 3 中采用了哈夫曼编码，针对每一个可能出现的采样值，给出了 32 个哈夫曼码表，在解码端只需根据样本值及边信息取得相应的哈夫曼码字。为了得到最高的压缩效率，通常在两个连续符号的编码结果之间并不插

入任何标记。因此 MP3 编码器端输出的是连续的哈夫曼码流，解码端必须从一维的比特流中分辨各个长度不同的哈夫曼码字，然后进行复杂的匹配过程，这给解码带来了难度。

2.2 MPEG－Ⅰ音频标准

2.2.1 MPEG－Ⅰ概述

在 MPEG－Ⅰ音频压缩算法标准中提供了以下模式：

1. 音频信号采样率可以是 32KHz，44.1KHz 或 48KHz。

2. 压缩后的比特流可以按照单声道、独立的两个声道、立体声、联合立体声等四种模式之一支持单声道或双声道编码。其中，联合立体声模式是利用了立体声通道之间的关联和通道之间相位差的无关性得到的。

3. 压缩后的比特率可以从 32Kbit/s 到 224Kbit/s（每声道），也可以使用用户定义的比特率。

4. MPEG 音频编码标准提供了三个独立的压缩层次，使用户可以在复杂性和压缩质量之间权衡选择。层Ⅰ最简单，最适于使用的比特率为 128Kbit/s（每声道）以上。层Ⅱ的复杂度中等，使用的比特率为 128Kbit/s（每声道）左右，其应用包括：数字广播，CD-ROM 上的声音信号以及 CD-I 和 VCD。层Ⅲ即我们通常所说的 MP3（MPEG－Ⅰ（or Ⅱ）Audio LayerⅢ），它最复杂，但音质最佳，比特率为 64Kbit/s（每声道）左右。尤其适用 ISDN 上的声音传输，本文将主要讨论这一层的解码算法。

5. 编码后的比特流支持 CRC 校验。

6. MPEG 音频编码标准还支持比特流中附带附加信息。

对于 MPEG－Ⅱ，增加了低采样频率的编码标准，采样频率可以扩展到 16 KHz、22.05 KHz、24 KHz。同时压缩后的比特率可以低到 8KHz（每声道），这有利于实现音质要求不高的现场语音录音。

2.2.2 MPEG－Ⅰ LAYER 3 编码的主要技术

MPEG－Ⅰ声音压缩算法是一种通用的声音编码技术，它对音源的性质没有作任何假设，而是利用人耳的听觉特性对声音进行压缩。一方面，去除声音信号本

身的相关性；另一方面，去除声音信号中人耳不可感知的部分。MPEG－Ⅰ LAYER 3 方案是综合 MUSICAM（掩蔽型通用子带综合编码和复用）算法和 ASPEC（自适应谱分析听觉熵编码）算法的优点提出的新的掩蔽型编码技术，采用了许多一二层未用到的技术，如多相/MDCT 混合滤波器组、前向回声控制（窗口切换）、比特池技术、非线性量化、熵编码等。

下面就对 MPEG－Ⅰ LAYER 3 的主要技术进行介绍。

1.多相/MDCT 混合滤波器组

多相滤波器对于 MPEG 音频压缩编码的各层来说是一样的。它将输入的音频信号分成 32 个等宽的频带，以相对低的复杂度，获得较好的时间分辨率和频率分辨率。但是，等宽的子带并不能准确地反映人耳听觉系统的频率特性，只有作为频率函数“临界带”的宽度才能准确地表示人耳的听觉特性。许多心理声学的效果是与临界带频率缩放比例一致的。例如，单个信号的响度和它在有掩蔽信号情况下的可听度是不同的，这是因为单个信号限制在一个临界带内，而有掩蔽信号时，它将扩展到其他的临界带中。在低频范围，一个子带覆盖若干个临界带，这时，量化器的比特不能针对个别临界带的噪声掩蔽来分配，而是将量化比特按最小噪声掩蔽的情况在整个子带上分配。

在各层中，编码器对子带滤波器的输出样值的处理是不一样的。Layer1 是对每个子带的连续 12 个输出样值进行处理，而 Layer2 和 Layer3 是同时对每个子带的连续 36 个输出样值进行处理。图 2.3 是他们的示意图。

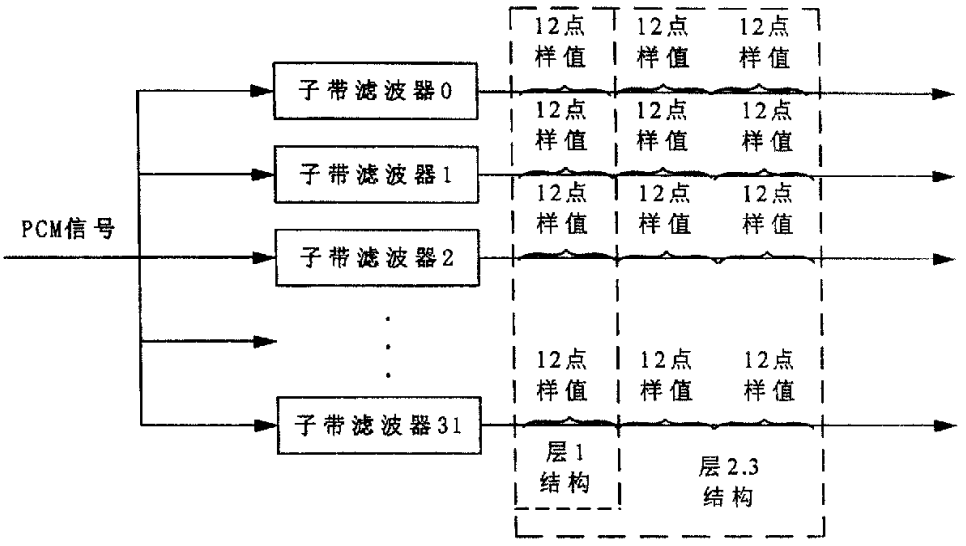


图2.3 层Ⅰ、Ⅱ、Ⅲ的子带样本（每输入32个PCM信号，各子带滤波器输出一个样值）

Fig. 2.3 Subband samples of layer1,2,3 figure

MPEG一Ⅰ采用多相/MDCT 混合滤波器组，以提高频率分辨率，增加编码增益。混合滤波器组的想法首先是由 K. Brandenburg 和 J. D. Jonhston 在第 88 届 AES 年会上提出的，把 Layer1 和 Layer2 使用的多相滤波器组的每路输出送到 18 信道的 MDCT 滤波器组，产生 576 条频率线。这在处理稳态信号时能获得最大的编码增益。必要时，MDCT 滤波器组能切换到低的频率分辨率和高的时间分辨率以减少前回声。为了保证时域混叠消除的特性，MDCT 的频率线必须为 4 的倍数，所以采用 3:1 的窗切换长度[8]。

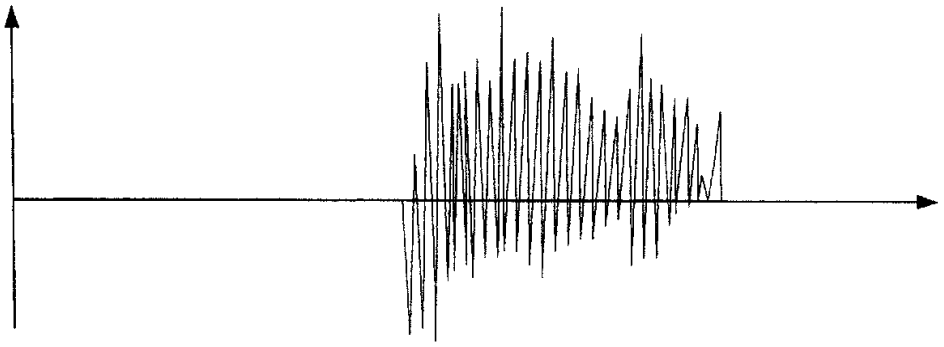
2. 自适应窗口选择技术

在频域编码中，当一个窗口中出现了一段安静的片段后紧接一个尖锐的声音的情况时，就会出现前回声，这将带来大量的量化噪声，严重影响编码的质量。

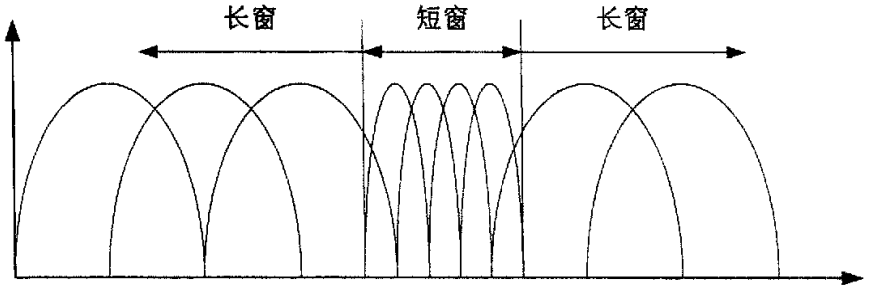
避免前回声的办法就是利用时域掩蔽效应中的前掩蔽效应。或在频域编码中采用短窗。前掩蔽效应的利用只能部分缓解前回声，而短窗的采用则可较好地抑制前回声。但是采用短窗必须使用大量的比特，这将影响整个算法的编码效率。

采用自适应窗口选择技术可以很好地在编码效率和编码质量之间取得折中，付出的代价就是算法复杂度的增加。所以，只在 MP3 中采用了自适应窗口选择技

术。这样，只需在需要抑制前回声的情况下采用短窗，而在平时则采用长窗，如图 2.4 所示。



(a)输入的音频信号



（b）窗函数

图 2.4 窗函数切换

Fig. 2.4 Window function switching

采用这种自适应的窗口切换技术，既满足了前回声抑制的要求，又不影响编码的效率。MPEG－Ⅰ音频中的窗口类型有起始窗，长窗（普通窗），短窗，结束窗（终止窗）四种，分别用于不同的情况，系统根据信号的实际情况决定采用哪种窗口。

下面解释各窗口类型的功能：

(1)长窗：用于稳态信号的正常窗口类型。

(2)开始窗：为了在长窗和短窗之间切换，使用这种混合窗。它的左边和长窗类型的左边具有相同的形状。右边的 1/3 长度的幅度是 1，1/3 和短窗的右边具有

相同的形状，剩余的 1/3 是 0。因此，与后面的短窗部分重叠可保证混叠抵消。

(3)短窗：短窗基本上和长窗具有相同的形状，只是长度是长窗的 1/3。它跟随着一个 1/3 的 MDCT。在 48KHz 取样频率时，时间分辨率增强为 4ms。在短窗情况下，混合滤波器组的频率分辨率是 192 线，而正常窗口频率分辨率是 576 线。

(4)结束窗：这种类型窗把短窗切换回正常窗，其形状与开始窗镜像。

MP3 中的窗口切换策略如图 2.5 所示。

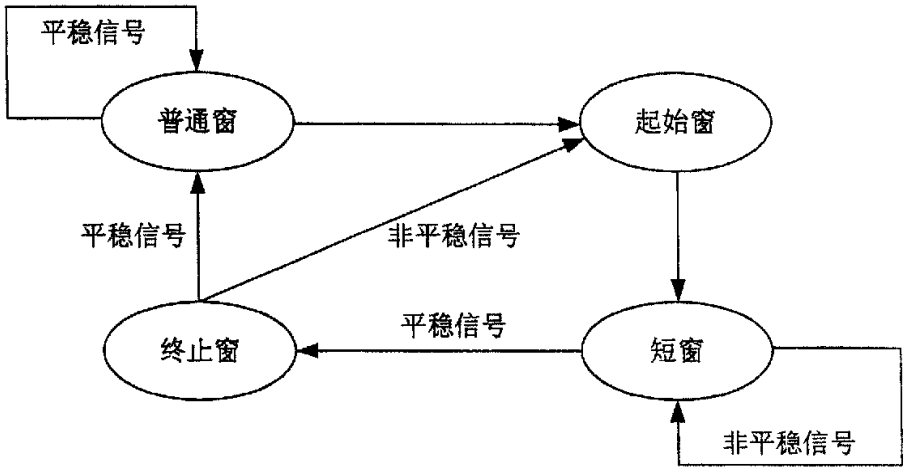


图2.5 窗口切换策略

Fig. 2.5 the policy of window switching

切换窗口类型的依据是必须得到前回声控制。使用混合滤波器组容易获得更好的前回声控制方案。在前回声情况下，如前描述的可把部分或全部的 MDCT 滤波器切换为短窗以获得更好的时间分辨率。如果在门限计算中完成前回声控制，前回声条件导致心理声学熵（PE），也就是信号编码所需的比特数大大增加。如果在某些区域要求的比特数超过了平均值，则假定前回声条件并启动窗口切换逻辑[9]。

3. 比特池技术

由于 MP3 采用了自适应窗口选择技术，这样在采用短窗时就会需要较多的编码比特。另一方面，在一些静音帧中，所需的比特数要小于一般的帧。这就会带来编码输出比特率不恒定的情况。

为了保持速率的恒定，在 MP1 和 MP2 中，针对所需比特数少于给定比特数的情况，采用了插入冗余比特的方法。由于没有采用自适应窗口选择技术，这两层不会出现所需比特多于给定比特数的情况。

在 MP3 中，保持比特率恒定的办法就是将某些帧（如静音帧）中多出来的比特保留下来，留给后面的帧使用。在每帧量化编码前，先根据信号的感觉熵（PE）预测其所需的比特数，若 PE 较大，即所需比特数较多，则从前面预留的比特中取出适当的比特加到可用的比特中。

采用了弹性比特存储技术之后，MP3 的帧结构就不同于一般数据流的帧结构了。在一般的帧中，每帧的数据都是紧接于帧同步码之后，而 MP3 帧的主数据则有可能先于该帧的同步码出现。为了准确定位每帧的主数据，MP3 在码流中设置了一个 9 比特的指针，用于指出主数据起始位置相对于帧同步码所超前的字节数。MP3 帧结构见图 2.6。

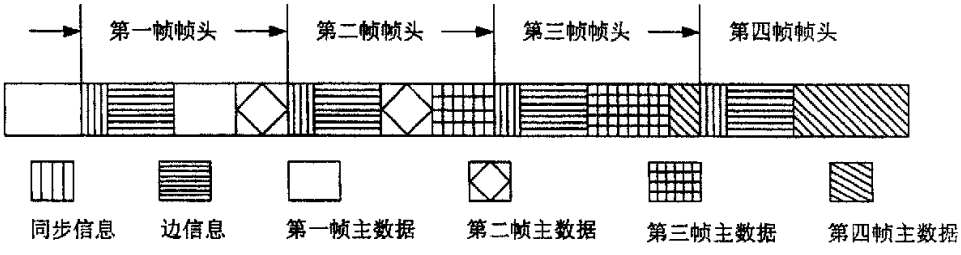


图2.6 MP3帧结构

Fig. 2.6 MP3 frame

由图可见，弹性比特存储技术实际上就是在每帧的主数据中以固定的间隔插入同步码和边信息，从而以固定波特率的格式实现了可变波特率的编码。在 MP3 的解码过程中，只需将每帧的主数据先暂存在缓冲区内，在收到同步码后，从缓冲区内取出相应的主数据即可。

4. 量化和编码

在 MP3 中，采用非线性量化和熵编码相结合的方法进一步压缩输出码率。非线性量化能起到一定的噪声整形作用，加上缩放因子对量化噪声的“着色”，可提高主观听觉质量。根据心理声学模型的输出，对 MDCT 的输出样本以颗粒为单位进

行比特位的量化和分配，再对量化结果进行哈夫曼编码。量化与编码主要是通过循环迭代来完成的。最高层称为帧循环，它调用外层迭代循环，而外层迭代循环又调用内层迭代循环。

帧循环对迭代循环所用到的迭代变量进行复位，计算传送每颗粒数据所能提供的最大比特数。然后调用外层迭代循环，跳出外层迭代循环后，在帧循环程序中计算存储每颗粒数据所用的比特数。

内层迭代循环完成对频谱线的量化，并给出量化阶的大小、哈夫曼编码信息等。

内层迭代循环对频谱值进行量化，将不可避免地产生量化噪声，而外层迭代循环通过给缩放因子频带分配缩放因子对量化噪声进行控制，尽量使量化噪声低于掩蔽阈值[10]。

熵编码对量化后的频率值使用哈夫曼码表进行可变长度编码。

综上所述，MPEG Layer 3 以增加少量复杂度为代价（特别是解码器算法复杂度几乎不增加），获得了比 Layer 1 和比 Layer 2 算法更高的声音质量或同样质量下更高的压缩率。在低比特率（如 64kbit/s 每声道）时效果更为明显。

第3章 MPEG LAYER 3数据流格式

PCM音频信号进行MP3压缩时，以1152个PCM采样值为单位，封装成具有固定长度的MP3数据帧，帧是MP3文件最小组成单位。在解码时，利用数据帧里的信息可以恢复出1152个PCM采样值。这1152个采样值被分成2个粒度组，每个粒度组包含576个采样值。一个MP3数据帧分为五个部分：帧头、CRC校验值、边信息、主数据、附加数据。如表3.1所示。

表3.1 MPEG LAYER 3的帧结构

Tab. 3.1 MPEG LAYER3 frame structure

帧头	CRC校验	边信息	主数据	附加数据

3.1 帧头

MP3数据流通过帧头来区分每一帧数据的开始。MP3的帧头有32比特长，包括同步字、标识符、扩展标识码、层、保护位、位率检索、取样频率、填充位、加密位、模式、模式扩展、版权位、原件/拷贝位和加重位。

1. 同步字：共12位，由位串“1111 1111 1111”所构成。
2. 扩展标识码（IDex）：用于指示算法规则的扩展的ID，它等于1，为MPEG1或MPEG2；它等于0为 MPEG2.5。
3. 标识码(ID)：表示算法规则的标识位，它等于1为MPEG1，它等于0为MPEG2或MPEG2.5。算法规则选择时由扩展标识码和标识码决定的，见表3.2。
4. 层：由2位所组成，用来表示哪一层被利用。第三层的位串是01。
5. 保护位：为了便于错误检测和隐藏。用于表示冗余信息是否已经加在音频位流中。位值为1表示未增加冗余信息，位值为0表示增加了冗余信息。

表3.2 算法规则的选择

Tab. 3.2 the choice of arithmetic

IDex	ID	算法规则
0	0	MPEG2.5
0	1	备用
1	0	MPEG2
1	1	MPEG1

6. 位率索引位：用 4 位来指示位率，所有的零值指示“自由格式”条件，在这种情形下可以使用不同于表中的固定位率。这里的固定意味着帧包含 N 或 N＋1 个槽，取决于填补位的值。位率索引是一个表的索引，见表 3.3。每层所用的表各不相同。总位率是由位率索引及标识码和扩展标识码决定的，而与模式（立体声、双通道、单通道）无关。通过开关位率索引支持可变位率。位率索引的开关既可以优化存储要求，又可以通过在位率表中的邻近值之间的开关插入任何方式数据比率的新语句。无论怎样，在自由格式中，固定位率是所要求的。在自由格式中，解码器也不需要支持位率高于 320kb/s、160 kb/s、64 kb/s 的位率。如果超过的话，就被禁用。

表3.3 位率选择表

Tab.3.3 Bit-rate selection table

位率索引	位率规格（kb/s）		
	ID=1 IDex=1	ID=0 IDex=1	ID=0 IDex=0
0000	自由	自由	自由
0001	32	8	8
0010	40	16	16
0011	48	24	24
0100	56	32	32
0101	64	40	40
0110	80	48	48
0111	96	56	56
1000	112	64	64
1001	128	80	禁用
1010	160	96	禁用
1011	192	112	禁用
1100	224	128	禁用
1101	256	144	禁用
1110	320	160	禁用
1111	禁用	禁用	禁用

7. 取样频率位：用于指定取样频率，见表 3.4。为了改变取样频率，音频解码器需要复位一下。

表3.4 取样频率选择

Tab. 3.4 Sampling frequency selection table

	MPEG1	MPEG2	MPEG5
00	44.1kHz	22.05 kHz	11.025 kHz
01	48 kHz	24 kHz	12 kHz
10	32 kHz	16 kHz	8 kHz
11	备用	备用	备用

8. 填充位：如果该位为 1，那么帧中包含一个额外槽，用于把中等位率调节为选定位率，否则该位必须为 0。在采样频率为 44.1KHz 时，填补是必要的，在自由格式中也可能需要填补。之所以要应用填补操作位到位流中，原因是在一定数量的音频帧后，编码帧的累计长度不应与下面的计算值偏离（＋、－1 槽）以上：

$$\text{累计帧长度} = \sum_{\text{第一帧}}^{\text{当前帧}} (\text{帧大小} * \text{位率} / \text{采样频率}) \qquad (3.1)$$

其中，帧大小＝384（层Ⅰ）/1152（层Ⅱ、Ⅲ）

9. 密码位：标志私有使用位。
10. 模式位：根据表 3.5 指定的模式，其中联合立体声是强度立体声与/或 ms 立体声。模式用 2 位来表示。

表3.5 模式选择

Tab. 3.5 mode selection

模式位	特定模式
00	立体声
01	联合立体声（强度立体声/或 ms 立体声）
10	双声道
11	单声道

11. 模式扩展位：这 2 位用于联合立体声模式，它们指示哪一种类型的联合立体声编码方式被应用。在算法规则中，强度立体声和 ms 立体声的频率范围是绝对的。如果模式位特指哪种立体声，那么哪种立体声就被利用。如果模式位指定联合立体声，那么相对应的联合立体声被利用。并且特定的联合立体声模式是由模式扩展位决定的，见表 3.6。

表3.6 联合立体声模式扩展

Tab. 3.6 Joint-stereo mode extension

Mode_extension	Intensity_stereo	Ms_stereo
00	off	off
01	on	off
10	off	on
11	on	on

如果模式位指定为立体声，则使用一种“立体声”（stereo）模式，或者当模式位指定联合立体声并且模式扩展位指定 intensity_stereo 为“off”、ms_stereo 为“off”时也使用立体声模式。

12. 版权位：用 1 位来表示。如果该位等于 0，那么位流没有版权保护，否则相应的位流受到版权保护。用户在使用时应注意。

13. 原件/拷贝位：用 1 位来表示。如果位流是拷贝的，那么这一位等于 0；如果位流是原件的，那么这一位等于 1。

14. 加重位：该位表明应使用的解码加重的类型。[11]

3.2 错误检测

一个16位长的CRC校验字，用于检测数据的传输过程中是否出现错误(任选)，它在编码位流中。

3.3 边信息

在每帧MP3数据中，对于单声道模式，边信息占用17字节的长度；其他模式中边信息占用32字节。边信息主要包含4个方面的信息：main_data_begin指针、2个粒度（granule）组共用的边信息、granule0的边信息及granule1的边信息。边信息的结构见表3.7。

表3.7 边信息的结构

Tab. 3.7 The structure of side information

	Main_data_begin	Private_bites	Scfis	Side_info_gr0	Side_info_gr1
单声道	9	5	4	59	59
双声道	9	3	8	118	118

主数据开始位（main_data_begin）:用于确定帧中主数据的第一位的位置值。这个 main_data_begin 值把位置用一负偏移量（相对于音频同步的第一个字节）表示，以字节为单位。属于帧头和辅助信息的字节数不考虑在内。例如，如果 main_data_begin=0，那么主数据从辅助信息后开始。

3.3.1 粒度组共用的边信息

两个粒度组共用的边信息包括保留位（private_bits）和缩放因子选择信息（scfsi）。

保留位：保留位的宽度由声道数决定，单声道为 5 比特，双声道为 3 比特。用户可以自己定义保留位的内容。

缩放因子选择信息：用来确定帧数据中的缩放因子是两个粒度组共用的还是针对单个粒度的。在 MP3 标准中，根据人体心理声学特性将 20Hz－20KHz 的频带分成 21 个（长窗的情况）或 12 个（短窗的情况）敏感带（即缩放因子带），这 21 个或 12 个敏感带又分成 2 个群，每个敏感带或群里的音频数据具有相似的特性，这给处理数据带来了方便，同时提高了压缩质量。在压缩时，同一个群中的敏感带都具有相同的缩放因子长度，每一个敏感带对应一个缩放因子。缩放因子选择信息字段每声道占用 4 比特，每个比特对应一个缩放因子群。‘0’表示此帧中该群的缩放因子是独立的（针对两个粒度）；‘1’表示缩放因子是两个粒度共用的，只有粒度 0 的缩放因子被发送。

3.3.2 每个粒度中的边信息

对于每个粒度，都有相应的边信息以对该粒度的数据进行压缩。每个粒度的结构如表 3.8 所示。下面逐一介绍每个变量的含义：

表3.8 每粒度的边信息结构

Tab. 3.8 the structure of granule information

	Part2_3_length	Big_values	Global_gain	Scale_fac_compress	Windows_sw_flag
单声道	12	9	8	4	1
双声道	24	18	16	8	2

(a)

	Block_type	Mixed_block_flag	Table_select	Subblock_gain
单声道	2	1	10	9
双声道	4	2	20	18

(b)

	Table_select	Region0_count	Region1_count
单声道	15	4	3
双声道	30	8	6

(c)

	Preflag	Scalefac_scale	Counttable_select
单声道	1	1	1
双声道	2	2	2

(d)

Part2_3_length：表示主数据中缩放因子和哈夫曼数据所占用的比特数。对于单声道分配 12 比特数据，双声道则分配 24 比特。由于每帧数据的边信息长度是固定的，所以可以从这个变量计算出下一个粒度的起始位置。

Big_values：在编码端，576 个采样值经过哈夫曼编码后，根据采样值的大小被重新排列，分成 3 个区域：rzero 区、count1 区和 big_values 区。rzero 区在高频区域，由一对一对的‘0’组成；count1 区在中频区，由+1、0、−1 的值组成，每 4 个值组成一组；big_values 区包含低频区幅度比较大的数据。big_values 的值表示这一区域所占的宽度，每声道用 9 个比特来表示。

Global_gain：全局缩放因子，表示编码器所采用的量化步长。在解码的反量化过程中，每个采样值都要乘以全局缩放因子。

Scalefac_compress：在编码时根据心理声学及所用的窗函数，每个粒度被分成 21 个（长窗）或 12 个（短窗）缩放带。这些缩放带又被分为两组：对于长

窗，由缩放带 0－10 和 11－20 构成两组；对于短窗，由缩放带 0－5 和 6－11 构成两组，这两组采用不同的缩放因子长度。Scalefac_compress 是一个 4 比特长的变量，结合 block_type 及 mixed_block_flag 的值，可以查表得出这两组缩放因子的长度 slen1 和 slen2。

若 block_type 的值为 0、1 或 3：slen1 表示缩放带 0－10 的缩放因子的长度；slen2 表示缩放带 11－20 的缩放因子的长度。

若 block_type 的值为 2 且 mixed_block_flag 的值为 0：slen1 表示缩放带 0－5 的缩放因子的长度；slen2 表示表示缩放带 6－11 的缩放因子的长度。

若 block_type 的值为 2 且 mixed_block_flag 的值为 1：由于 mixed_block_flag 的值为 1，所以该粒度由长窗和短窗混合组成，对于长窗部分，slen1 表示 0－7 缩放带的缩放因子的长度；对于短窗部分，slen1 表示 3－5 缩放带的缩放因子的长度。slen2 表示缩放带 6－11 的缩放因子的长度。

Scalefac_compress 对于 slen1 和 slen2 的选择见表 3.9。

表3.9 Scalefac_compress对于slen1和slen2的选择

Tab.3.9 the selection of scalefac_compress for slen1 and slen2

Scalefac_compress	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
slen1	0	0	0	0	3	1	1	1	2	2	2	3	3	3	4	4
slen2	0	1	2	3	0	1	2	3	1	2	3	1	2	3	2	3

Window_sw_flag:窗口切换标志。用来表示粒度中是否出现了特殊窗，‘1’表示出现了特殊窗，‘0’表示没有特殊窗。

Block_type:窗类型。用来表示出现特殊窗时的窗类型，包括起始窗、短窗、结束窗等。

Mixed_block_flag:混合窗标志。用来表明低频进行变换时所用的窗口类型与高频进行变换时所用的窗口类型不同。如果 Mixed_block_flag 为 0，那么所有的子带都按照 Block_type 表明的方式进行变换；如果 Mixed_block_flag 为 1，那么用正常窗口对对应于两个最低频率子带的频率进行变换，而剩余的 30 个子带按照 Block_type 指定的方式进行变换。

Table_select:在编码时用到了 32 个哈夫曼码表对 Big_values 区域的数据进行编码。在解码端，根据这一变量的值来选择 32 个码表进行哈夫曼解码。

Subblock_gain：短窗中的缩放因子。在反量化的过程中使用。

Region0_count：为了提高哈夫曼编码的效率，编码器的 Big_values 区域又被划分成 3 个小区域：区域 0、区域 1、区域 2，每个区域中运用不同的哈夫曼码表。Region0_count 表示处于区域 0 的缩放带个数。

Region1_count：表示处于区域 1 的缩放带个数。

Preflag, Scalefac_scale:在反量化过程中对压缩数据还原时用到的变量。

Count1_table_select:该变量选择 count1 区域哈夫曼解码的码表。在对 count1 区域的数据编码时，只用到了 2 个哈夫曼码表，该变量的长度为 1 比特。

3.4 主数据

MP3 数据帧的主数据部分包括缩放因子（scale factors）和哈夫曼编码数据两部分。

缩放因子：在 MP3 数据中，缩放因子以缩放带为单位，即缩放带中的每个样点具有相同的缩放因子。这些缩放因子是否为两个粒度共用由 scfsi 变量决定。缩放因子所占用的空间由压缩时采用的窗类型决定。

哈夫曼编码数据：在处理 MP3 数据的时候，把每粒度中的 576 个样点分成了 3 个区域（big_values、count1、rzero)。rzero 区域的数据不进行编码。对于 big_values 区域中的样点，又分成了 3 个区域（region)。在该区域中，对两个样点进行统一编码；对于 count1 区域中的样点，4 个样点进行统一的哈夫曼编码。

3.5 附加信息

附加数据是可选的，用户可以在该字段中添加信息[12]。

第4章 MPEG LAYER 3音频解码算法及软件实现

MPEG LAYER3 标准规定了对 MP3 音频数据进行解码的算法。在 MP3 解码的流程中，涉及到很多复杂的解码算法。要设计 MP3 解码器，无论是软件解码器还是硬件解码器都需要对这些算法有很深入的理解。在这一章中先给出 MP3 解码的流程，然后分别介绍每一步的算法。

MPEG LAYER3 的解码流程图如图 4—1 所示。在进行 MP3 解码时，首先要检测数据流中的同步字来正确确定一帧数据的开始，提取帧头信息，从而得到相应的解码参数，同时分离边信息和主数据。通过对边信息数据解码可得到哈夫曼解码信息和反量化信息，主数据就可以根据哈夫曼解码信息解码出量化之后的数据，量化后数据结合反量化信息就可以得到频域中的数据流。结合帧头中的立体声信息，对反量化结果进行立体声处理后，通过反混叠处理、反离散余弦变换、合成滤波器处理就可以得到原始的 PCM 音频信号。解码流程见图 4.1。

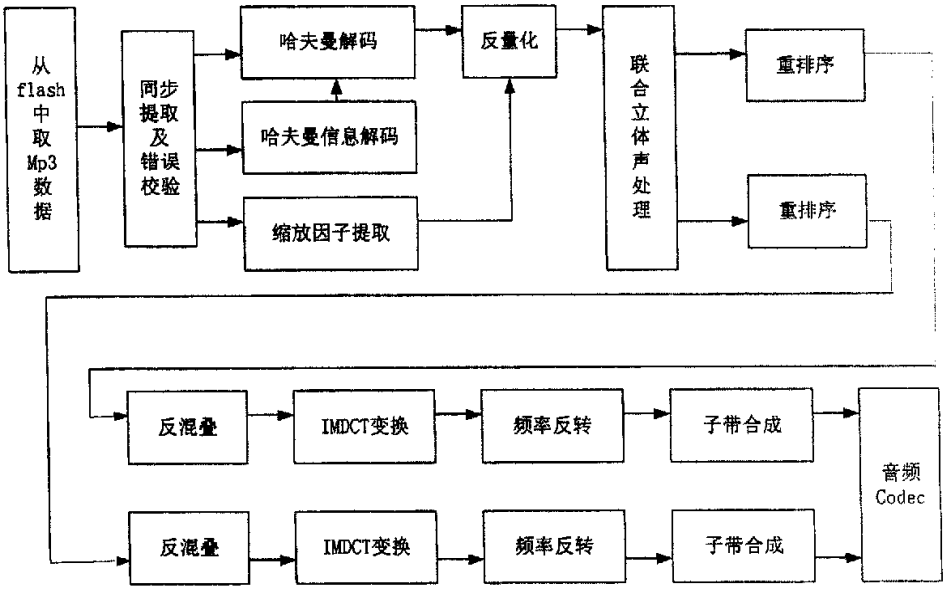


图4.1 MPEG LAYER3的解码流程图

Fig. 4.1 MPEG LAYER3 decoding float chart

4.1 数据流的同步及帧头信息、边信息的读取

4.1.1 数据流的同步及帧头信息的读取

MP3 数据流的同步以帧为单位，每一帧的帧头都包含同步信息。这个同步信息是连续的 12 比特的‘1’组成。MP3 音频解码过程中的第一步就是使解码器与输入数据流同步。在启动解码器后，可以通过搜索数据流中的 12 比特长的同步字来完成。

在软件实现上，我们编写 int seek_sync(SYNC_WORD)这个函数在 MP3 比特流中搜索同步字以实现帧同步。其中 SYNC_WORD 被宏定义为 0xfff，当成功搜索到同步字时，返回值为 1，否则为 0。

帧头信息中包含采样率、比特率、填充位等主要的用于解码的信息。比特率和填充位信息用来确定每帧的帧长。在得到每帧的帧头信息之后，帧的帧长可由下面的式子来确定：

$$length = (144*bitrate/\ sampfreqs) + padding\_bit \tag{4.1}$$

其中 bitrate 代表比特率；sampfreqs 代表采样率；padding_bit 为填充位的值[13]。

帧头中包含的这些信息在后续的解码过程中要用到。在设计软件的时候，把帧头信息定义为一个结构，每帧的帧头信息就存放在定义为这个结构类型的变量中，后面的流程中要用到头信息，只需访问结构中的成员变量即可。在程序中，这个数据结构的定义如下：

```
typedef struct {
    int version;                    //算法规则扩展标识位
    int lay;                        //算法规则标识位
    int error_protection;           //保护位
    int bitrate_index;              //位率索引位
    int sampling_frequency;         //采样频率位
    int padding;                    //填充位
    int extension;                  //密码位
    int mode;                       //模式位
    int mode_ext;                   //模式扩展位
```

```
    int copyright;                  //版权位
    int original;                   //原件/拷贝位
    int emphasis;                   //加重位
} layer;
```

完成读取帧头信息功能的函数是 void decode_info( )，它从 MP3 比特流中获得帧头信息，并存储在帧头信息结构类型变量中。

4.1.2 边信息的读取

紧跟在帧头信息后面的是边信息，边信息包括两个粒度组共用的边信息和每个粒度组的边信息。其中两个粒度组共用的边信息包括主数据开始、保留位和缩放因子选择信息。和对帧头信息的处理一样，把两个粒度组共用的边信息定义成结构：

```
typedef struct {
   unsigned main_data_begin;     //主数据开始指针
   unsigned private_bits;        //私有位
   unsigned scfsi[2][4];         //两个粒度共有选择信息
} III_side_info_t;
```

每个粒度的边信息提供哈夫曼解码和反量化时所需要的参数，如 big_values 、 table_select 、 region0_count 、 region1_count 、 以 及 count1_table_select 等，它的结构定义如下：

```
struct Granule {
        unsigned part2_3_length;        //主数据位数位
        unsigned big_values;            //大值数位
        unsigned global_gain;           //全局增益位
        unsigned scalefac_compress;     //比例因子压缩位
        unsigned window_switching_flag; //窗口切换标志位
        unsigned block_type;            //窗口类型位
        unsigned mixed_block_flag;      //混合窗口标志
        unsigned table_select[3];       //哈夫曼码表选择位
```

```
        unsigned subblock_gain[3];        //子块增益位
        unsigned region0_count;           //区域0计数位
        unsigned region1_count;           //区域1计数位
        unsigned preflag;                 //预标志
        unsigned scalefac_scale;          //比例因子缩放位
        unsigned count1table_select;      //count1区表选择位
    }grle[2][2];
```

在后面的解码流程中如果要用到粒度信息，只需要访问这个结构的数组成员即可。

在软件设计中，完成边信息的读取并存储起来功能的函数是 void III_get_side_info(III_side_info_t *si)，它从MP3比特流中读取两个粒度共用的边信息和每个粒度的边信息，存放在结构III_side_info_t和Granule的变量中。

4.2 主数据的读取

由于MPEG LAYER 3标准中采用了比特池（bit_reservoir）技术，所以当前帧的主数据不一定全部都在当前帧中。在解码过程中，必须结合主数据开始指针(main_data_begin)的值来确定主数据的开始位置。因此，解码器需要开辟一个缓冲区作为比特池的存储空间，处理完当前帧后，把此帧中的缓冲数据存到缓冲区中供后续帧使用。

主数据中包含的数据有缩放因子、哈夫曼数据及附加数据。这些字段在主数据中有固定的格式。以双声道为例，如图4.2所示。在解码过程中，应对这些数据顺序提取。哈夫曼解码后输出的数据是经过量化的频域样本值，和缩放因子结合就可以进行反量化处理了。

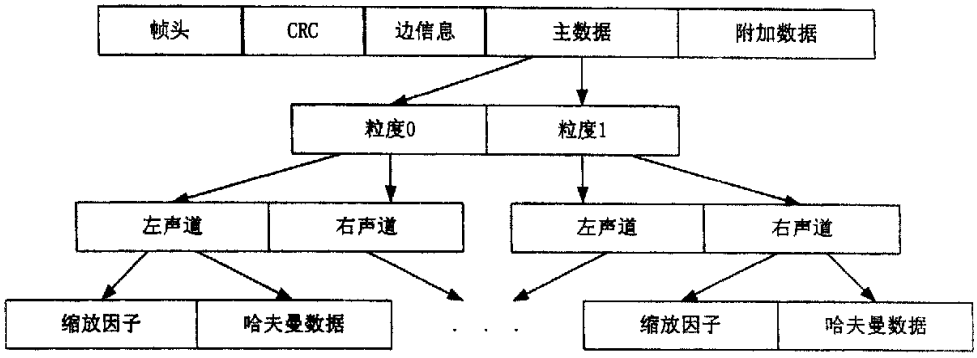


图4.2 双声道帧数据结构图

Fig. 4.2 the frame structure of dual_monophonic

在读取主数据之前首先要计算主数据的长度，此功能由 int main_data_slots() 这个函数完成，此函数的返回值为主数据的长度的字节数。然后根据主数据的长度从 MP3 比特流缓冲区读取该帧的主数据到 part2_3 缓冲区中。

缩放因子位于主数据中，读取出主数据后，就可以开始提取缩放因子。缩放因子是以缩放带为单位的，每一粒度中缩放带的个数由编码时所用的窗类型决定，因此在程序中需要首先判断所用的窗类型；同时每一缩放带编码的缩放因子长度不一样，这个长度根据 scalefac_compress 变量值查表得出，根据第三章的介绍，长窗和短窗的缩放因子长度分别记为 slen1 和 slen2。在程序中，定义了结构体来存放缩放因子：

```
typedef struct {
 int l[22];                    //存储长窗的缩放因子
 int s[3][13];                 //存储短窗的缩放因子
} III_scalefac_t[2];
```

长度定义为 2 是对应于一个粒度组中左右两个声道。其中数组 l[22]对应于长窗的缩放因子，s[3][13]对应于短窗的缩放因子。通过函数 void III_get_scale_factors(III_scalefac_t *scalefac, III_side_info_t *si, int gr, int ch)从 part2_3 缓冲区中获得缩放因子。

4.3 哈夫曼解码和反量化

在 MP3 编码过程中，根据心理声学模型的输出，对离散余弦变换的输出样本以粒度为单位进行比特位的量化和分配，再对量化结果进行哈夫曼编码。量化和编码主要是通过循环迭代完成的，循环模块分三层描述。最高层称为帧循环，它调用外层迭代循环，而外层迭代循环又调用内层迭代循环。[14]但是在解码过程中，哈夫曼解码和反量化过程是分开实现的。下面分别对这两个步骤进行说明。

在主数据中，紧跟在缩放因子之后的就是哈夫曼编码数据了。每个粒度组的频率线都是用不同的哈夫曼表来进行编码的。编码时，把整个从 0 到奈奎斯特频率的频率范围（共 576 个频率线）分成几个区域，然后再用不同的表编码。划分过程是根据最大的量化值来完成的，它假设较高频率的值有较低的幅度或者根本不需要编码。从高频开始，一对一对的计算量化值等于“0”的数据，此数目记为“rzero”。然后 4 个一组的计算绝对值不超过“1”的量化值（也就是说，其中只可能有－1，0 和 1 共 3 个可能的量化级别）的数目，记为“count1”，在此区域只应用了两个哈夫曼编码表，即四元组的哈夫曼表 A 和 B（hA，hB）。最后，剩下的偶数个值的对数记为“big_values”，在此区域值应用了 32 个哈夫曼编码表，即哈夫曼表 0～31（h0～h31）。

此后，为增强哈夫曼编码性能，进一步划分了频谱。也就是说，对 0～big_values*2 的区域（姑且称为大值区）再细化，目的是为了得到更好的错误健壮性和更好的编码效率。在不同的区域内应用了不同的哈夫曼编码表 0～31。具体使用哪一个表由 grle[gr][ch].table_select [region]给出。从帧边信息表中可以看到：当 window_switching_flag＝“1”时，只将大值区再细分为 2 个区，此时 region1_count 无意义，此时 region0_count 的值是标准默认的；但当 window_switching_flag＝“0”时，由 region0_count 和 region1_count 再将大值区分为 3 个区。但是由于 region0_count，region1_count 是根据从 0～576 个频率线划分的，因此有可能超过了 big_values*2 的范围，此时以 big_values*2 为准。region0_count 和 region1_count 表示的只是一个索引值，具体频带要根据标准中的缩放因子频带表来查得[15]。

在大值区中超过 15 的值，都以 15 和一个余数来表示，余数我们称之为 ESCAPE

值，和 ESCAPE 的值大小相等的 bits 称为“linbits”，具体公式如下：

ESCAPE＝量化值－15，当量化值>15 时

Linbits＝字长（ESCAPE 值）

图 4.3 表示的三个哈夫曼编码的区域和比例因子之间的关系。

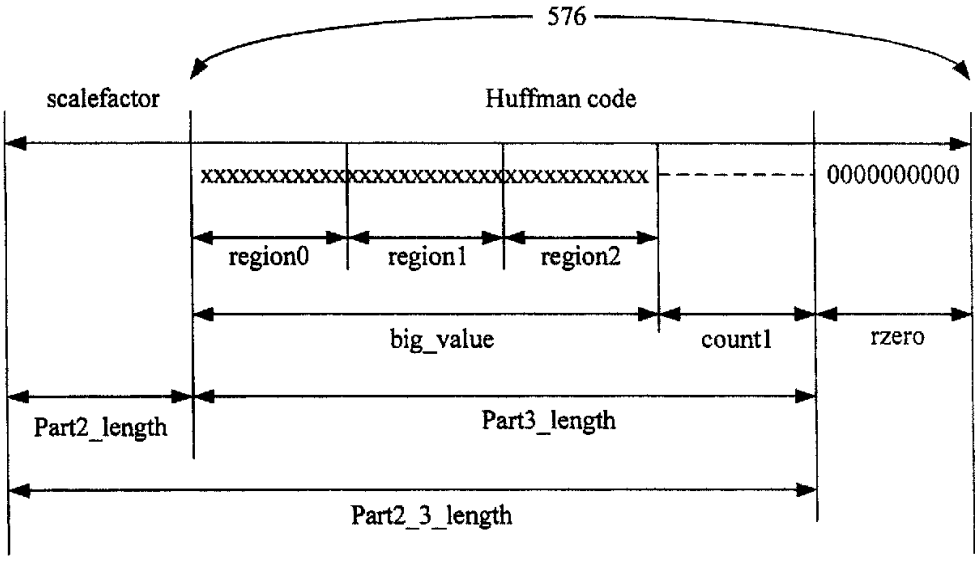


图4.3 主数据结构图

Fig. 4.3 the structure of main data

哈夫曼解码所需要的信息在前面的步骤中已经得到。由于在编码时将样本点分成了 3 个区域，这三个区域采用不同的编码方法，所以在编写软件时要区分这三个区域并选择不同的解码方式。

程序的一开始首先要确定要解码的样本值属于哪个区域，这个区域可以从 big_values 的值确定，big_values 变量的值就是属于 big_values 区域的样本数；对 big_values 区的所有样本解完后，主数据中剩余的就是 count1 区的哈夫曼编码数据，因为 rzero 区是不占用比特数的。对上述两个区域的样本解码完毕后，剩下的样本点数就是 rzero 区的数，可以用‘0’直接填充到满 576 个样本。哈夫曼的解码流程图如图 4.4 所示。

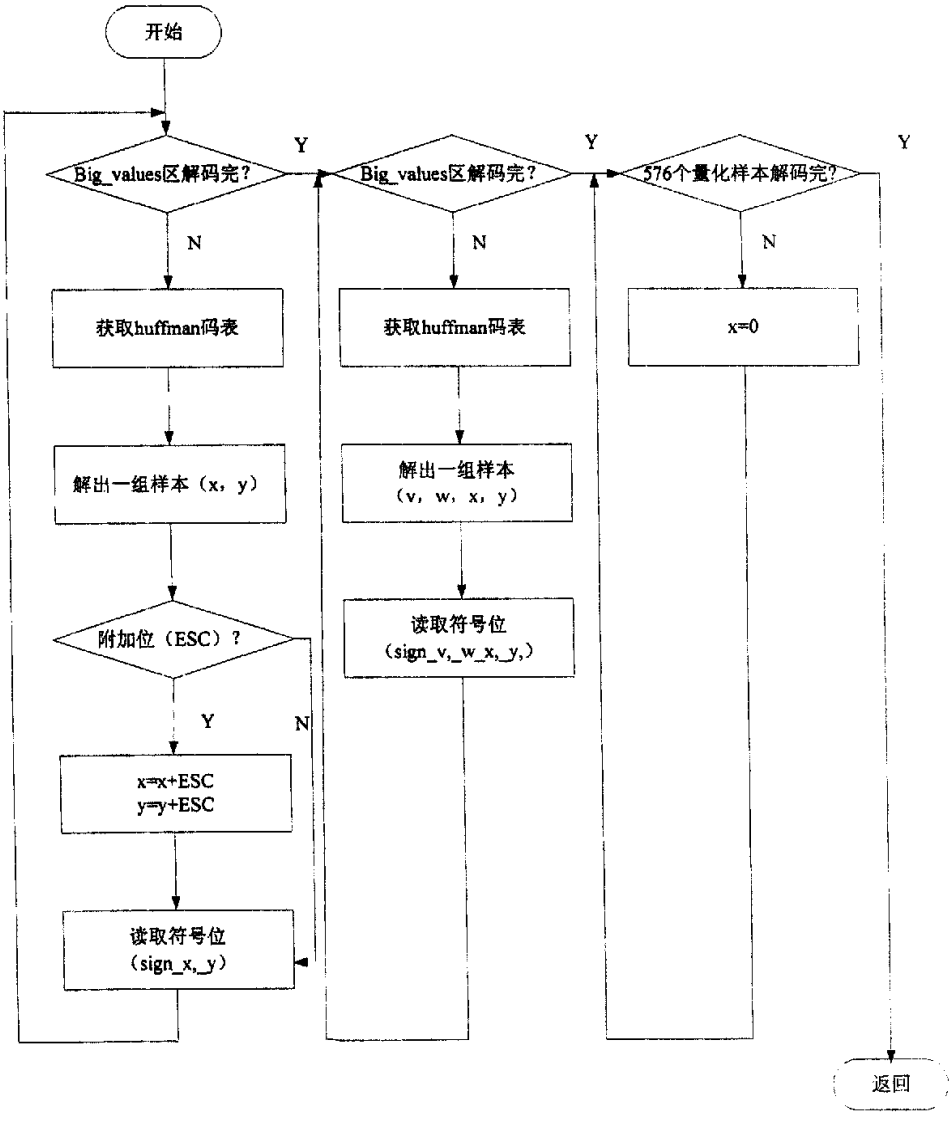


图4.4 哈夫曼解码流程图

Fig. 4.4 Huffman decoding float chart

在本系统中编写函数 void III_hufman_decode(struct Granule *gr, int part2_start, int freqline[SBLIMIT][SSLIMIT])来完成哈夫曼解码。其中函数入口参数*gr 中存放用于哈夫曼解码的边信息，part2_start 用来计算主数据中哈夫曼编码数据的结束位置。freqline[SBLIMIT][SSLIMIT]用来存放哈夫曼解码后的

量化后的样本值。由于哈夫曼码表是按照哈夫曼码值大小排列的顺序表，所以把 Big_values 区和 Count1 区样本值与哈夫曼码值相匹配的解码过程，用对于顺序表查找效率很高的折半查找算法来实现。从而提高了哈夫曼解码这一步骤的效率。

逆量化频谱过程就是基于前面步骤中所得到的哈夫曼解码数据，根据逆量化全缩放公式和帧边信息，对于不同的窗类型采用不同的公式以恢复 576 个频率线的真实值。在编码过程中进行 MDCT 变换时，针对不同信号为同时得到较好的时域和频域分辨率定义了 2 种不同的窗长：长窗的块长为 18 个样本，短窗的块长为 6 个样本。这使得长窗对于平稳的声音信号可以得到更高的时域分辨率。由于在短窗模式下，3 个短窗代替 1 个长窗，而短窗的大小恰好是一个长窗的 1/3，所以 MDCT 的样本数不受窗长的影响。对于给定的一帧声音信号，MDCT 可以全部使用长窗或全部使用短窗，也可以长短窗混合使用。因为低频区的频域分辨率对音质有重大的影响，所以在混合窗模式下，MDCT 对最低频的 2 个子带使用长窗，而对其余的 30 个子带使用短窗。这样，既能保证低频区的频域分辨率，又不会牺牲高频区的时域分辨率。长窗和短窗之间的切换有一个过程，一般用一个带特殊长转短（起始块，block_type=1）或短转长（终止块，block_type=3）数据窗口的长窗来完成。

哈夫曼解码数据记为 x_i，长窗中的数据用下面的公式进行反量化，反量化后的值记为 y_i：

$$y_i = sign(x_i)\cdot abs(x_i)^{\frac{4}{3}} \cdot \frac{2^{\frac{1}{4}(global\_gain[gr][ch]-210)}}{2^{(scalefac\_multiplier(scalegac\_l[gr][ch][sfb]+preflag[gr][ch]\cdot pretab[sfb]))}} \quad (4.2)$$

短窗中的样本的反量化公式为：

$$y_i = sign(x_i)\cdot abs(x_i)^{\frac{4}{3}} \cdot \frac{2^{\frac{1}{4}(global\_gain[gr][ch]-210-8\cdot subbolck\_gain[gr][ch][sfb][window])}}{2^{(scalefac\_multiplier\cdot(scalegac\_s[gr][ch][sfb][window]))}} \quad (4.3)$$

在反量化公式中，频谱值 x_i 首先提高到原来的 4/3 幂，用来补偿编码时的幅度衰减。之后再乘以符号位 sign（x_i）。Global_gain 变量是每声道中的全局量化步长。常数‘210’是一个系统常数，用来保证合适的量化步长，同时也保证

编码过程中不会出现全‘1’而扰乱同步字。在编码器中缩放因子采用指数量化的形式，步长为2或$\sqrt{2}$，通过scalefac_scale来标识，若scalefac_scale=0则scalefac_multifier=0.5，否则scalefac_multifier=1。Preflag和pretab只在长窗中有效。Preflag是高频预加重标志，‘1’表示采用高频预加重，pretab[sfb]用来查表得出每个缩放带的预加重值，其中sfb是当前样本所处的比例因子带。Scalefac_l和scalefac_s对应缩放因子解码中所得出的长短窗的缩放因子。当运用短窗时，subblock_gain变量对应子带中更细的量化[16]。

在软件设计，编写函数void III_dequantize_sample(int is[SBLIMIT][SSLIMIT], double xr[SBLIMIT][SSLIMIT], III_scalefac_t *scalefac, struct Granule *gr_info, int ch)来完成反量化。函数的入口参数is[SBLIMIT][SSLIMIT]是哈夫曼解码后的样本值，*scalefac是缩放因子，*gr_info是边信息，ch标明当前反量化的样本是属于左右哪个声道。经过反量化后的576个样本值存放在xr[SBLIMIT][SSLIMIT]中。在这个函数中，对于反量化公式的实现做了一些数学上变换处理：对公式两边取以e为底的自然对数，这样就避免了对频谱值x_i的4/3幂的计算，然后调用C语言库函数中的exp()这个函数对反量化后的值y_i进行恢复。

4.4 立体声处理

MPEG I LAYER 3中除了支持简单的单声道和立体声（相互独立的两个声道）外，还支持更为复杂的联合立体声模式：MS－立体声（MS－stereo）和强度立体声（intensity stereo）。

MS立体声模式：传送的是规格化的中间/旁边声道M_i/S_i值，而不是左右声道L_i/R_i。值M_i在左声道中传送，值S_i在右声道中传送。当两个声道高度相关时，这种模式比较适合，这就意味着“和”信号中要比“差”信号中包含较多的信息。一般左右声道的值是很接近的，采用MS立体声模式后，只需要传送一个均值和一个小值，这样可以减少传输的比特数。在解码时，左右两个声道L_i/R_i可以通过下面的公式来重建，其中i表示的是频率线的指数：

$$L_i=\frac{M_i+S_i}{\sqrt{2}} \quad 和 \quad R_i=\frac{M_i-S_i}{\sqrt{2}} \tag{4.4}$$

MS 立体声模式处理是无损的。

强度立体声模式：在层 3 中，强度立体声不是像第一层和第二层那样通过使用一对比例因子来完成，而是左声道仍传送缩放因子，右声道传送立体声位置 is_pos[sfb]。编码器把一些高频子带的输出编码为单个的“和”信号 L+R，而不是分别独立的传送左和右的子带信号，左右声道的平衡可以通过比例因子来传输。解码器通过单个 L+R（$=L_i^{'}$）信号来重构左右两个声道的信号，右声道的平衡比例因子用is_pos_{sfb}来表示。解码时利用下面两个公式解出左右声道信号：

$$\text{is}\_ratio_{sfb} = \tan(is\_pos_{sfb} \cdot \frac{\pi}{12}) \tag{4.5}$$

$$L_i = L_i^{'} \cdot \frac{is\_ratio_{sfb}}{1+is\_ratio_{sfb}} \quad 和 \quad R_i = L_i^{'} \cdot \frac{1}{1+is\_ratio_{sfb}} \text{[17]} \tag{4.6}$$

根据帧头信息中的模式位(mode)和模式扩展位(mode_extension)的值可以确定何时应用 M/S 立体声和强度立体声解码公式：

1. 若（mode! = 01），左右声道使用各自对立体声边信息对主要数据解码，获得通道独立的样本数据。
2. 若（mode = 01）&(mode_extension = 01），则对右声道 Zero_part 部分使用强度立体声解码(直至 $is\_pos_{sfb}=7$)，其余部分为独立的左右声道数据。
3. 若（mode = 01）&(mode_extension = 10），则对全部频率线用 M/S 立体声方式解码。
4. 若（mode = 01）&(mode_extension = 11），则对右声道 Zero_part 部分使用强度立体声解码(直至 $is\_pos_{sfb}=7$)，其余使用用 M/S 立体声方式解码。

在软件设计中，编写函数 void III_stereo(double xr[2][SBLIMIT][SSLIMIT],III_scalefac_t *scalefac, struct Granule *gr_info)来实现立体声的处理。其中 xr[2][SBLIMIT][SSLIMIT]为前面反量化步骤的输出结果，两个声道的频率线值，它既作为入口参数存放反量化后的样本值，也作为出口参数存放立体声处理后的样本值。*scalefac 存放缩放因子，*gr_info 存放边信息。在程序中，我们取 1.41421356 作为$\sqrt{2}$的近似值。对于公式中的$\tan(is_pos_{sfb} \cdot \frac{\pi}{12})$调用 c 语言的库函数 tan（）来实现。

4.5 重排序和反混叠

反量化过程中得出的频谱值并不是按相同顺序排列的。在编码的 MDCT 过程中，对于长窗产生的频谱值先按子带然后按频率排列；对于短窗，产生的频谱值时按子带、窗、频率的顺序排列的。为了提高哈夫曼编码效率，短窗中的数据被重新排序，按照子带、频率、窗的顺序排列。解码时，重排序过程就是将短窗中的频谱值重新排列。变量 window_switch_flag 和 block_type 可以用来确定是否重排序。

在软件设计中，编写函数 void III_reorder(SS xr , struct Granule *gr_info)来实现重排序。入口参数 xr 是立体声处理后某一声道的样本值，gr_info 是该声道的边信息。xr 也作为出口参数，存放重排序后的样本值。

在编码的 MDCT 过程中，为了得到更好的频域特性对长窗对应每个子带进行了去混叠处理。为了得到正确的音频信号，在解码时必须对长窗对应的子带进行混叠重建。在数据帧中存在长窗有两种情况：

1. gr_info->window_switching_flag && gr_info->mixed_block_flag && (gr_info->block_type == 2)，即混合窗口模式下最低频的两个子带。
2. 整个数据帧全部都为长窗的情况。

每个对应长窗的子带的混叠重建由 8 个碟形运算组成。如图 4.5。

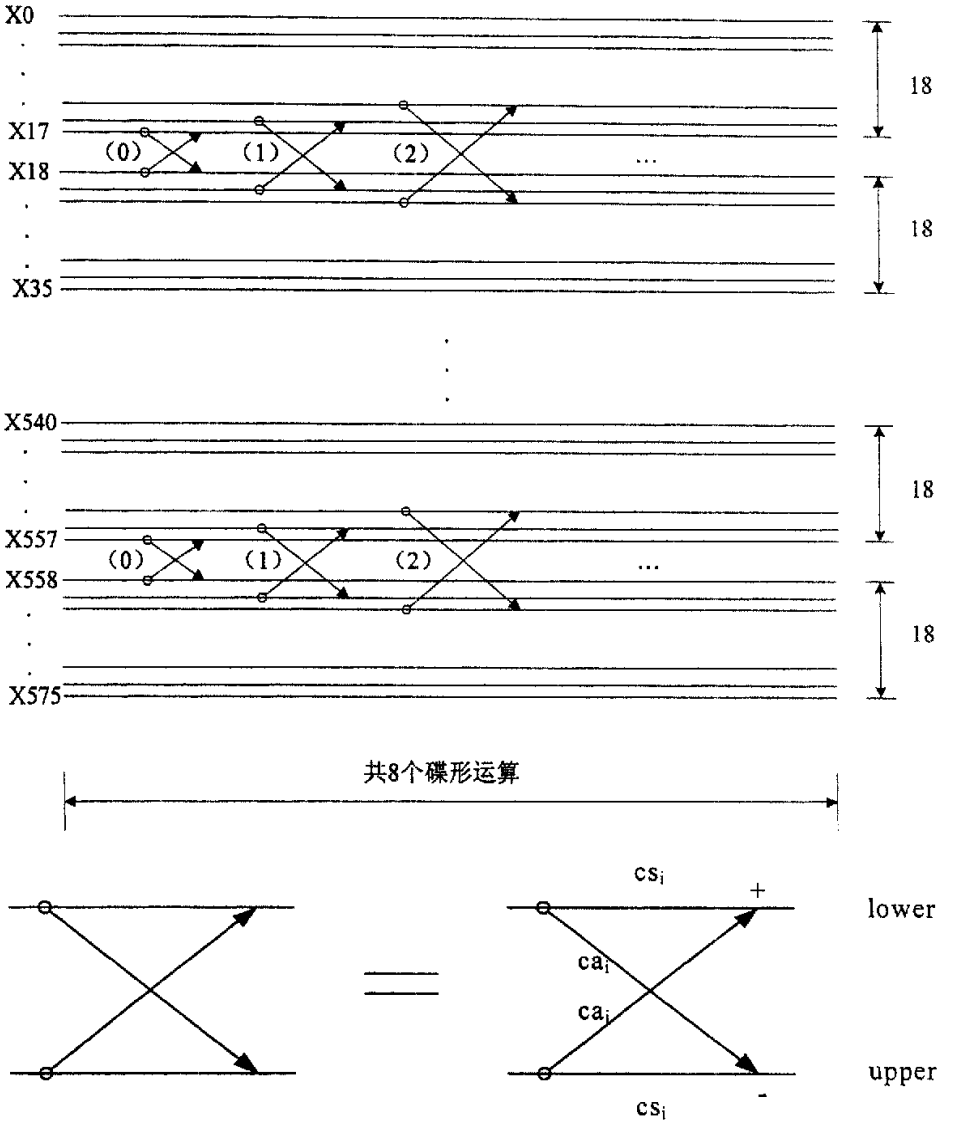


图4.5 混叠重建示意图

Fig. 4.5 the sketch map of alias reduction

图中的每一个碟形运算的上面输入样本值记为 bu，下面的输入样本值记为 bl，上面的输出样本值记为 bu'，下面的输出样本值记为 bl'。则碟形运算用公式可以表示为：

$$bu' = bu * cs_i - bl * ca_i \tag{4.7}$$

$$bl' = bl * cs_i + bu * ca_i \tag{4.8}$$

其中 cs_i 和 ca_i 的值由 ISC/IEC11172-3 给出。

在系统中用函数 void III_antialias(SS hybridIn, struct Granule *gr_info)来实现混叠重建。入口参数*gr_info 是边信息，SS 定义为 typedef double SS[SBLIMIT][SSLIMIT](SBLIMIT 宏定义为 32， SSLIMIT 宏定义为 18),hybridIn 既作为入口参数来存放混叠重建前的样本值，又作为函数输出存放混叠重建后的样本值。

4.6 逆向离散余弦变换

经过混叠重建后的信号便要进行逆向的离散余弦变换。

在编码过程中进行 MDCT 变换时，针对不同信号为同时得到较好的时域和频域分辨率定义了两种不同的块长：长块的块长为 18 个样本，短块的块长为 6 个样本。这使得长块对于平稳的声音信号可以得到更高的频率分辨率，而短块对跳变信号可以得到更高的时域分辨率。由于在短块模式下，3 个短块代替 1 个长块，而短块的大小恰好是一个长块的三分之一，所以 MDCT 的样本数不受块长的影响。对于给定的一帧声音信号，MDCT 可以全部使用长块或全部使用短块，也可以长短块混合使用。因为低频区的频域分辨率对音质有重大影响，所以在混合块模式下，MDCT 对最低频的两个子带使用长块，而对其余的 30 个子带使用短块。这样，既能保证低频区的频域分辨率，又不会牺牲高频区的时域分辨率。长块和短块之间的切换有一个过程，一般用一个带特殊长转短或短转长数据窗口的长块来完成这个长短块之间的切换。因此长块也就是包括正常窗，起始块和终止窗的数据块；短块也包含 18 个数据，但是是由 3 组 6 个数据独立加窗后在经过连接计算得到的。

逆向离散余弦变换的公式为：

$$x_i = \sum_{k=0}^{k<\frac{n}{2}} x_k \cdot \cos\left(\frac{\pi}{2n}(2i+1+\frac{n}{2})(2k+1)\right), \quad \text{for i =0 to n-1（短窗 n=12，长窗 n=36）} \tag{4.9}$$

x_k 表示经去混叠处理输出的频谱值，x_i 表示逆向离散余弦变换的输出值，最终的输出为子带样本 s_i。根据 block_type 的值，x_i 还要进行不同的加窗运算。

block_type=0（正常窗）：

$$w_i = \sin(\frac{\pi}{36}(i+\frac{1}{2})), \quad \text{for i=0 to 35} \tag{4.10}$$

block_type=1（起始窗）：

$$w_i = \begin{cases} \sin(\frac{\pi}{36}(i+\frac{1}{2})) & for\ i=0\ to\ 17 \\ 1 & for\ i=18\ to\ 23 \\ \sin(\frac{\pi}{12}(i-18+\frac{1}{2})) & for\ i=24\ to\ 29 \\ 0 & for\ i=30\ to\ 35 \end{cases} \tag{4.11}$$

block_type=3（结束窗）：

$$w_i = \begin{cases} 0 & for\ i=0\ to\ 5 \\ \sin(\frac{\pi}{12}(i-6+\frac{1}{2})) & for\ i=6\ to\ 11 \\ 1 & for\ i=12\ to\ 17 \\ \sin(\frac{\pi}{36}(i+\frac{1}{2})) & for\ i=18\ to\ 35 \end{cases} \tag{4.12}$$

block_type=2（短窗）：

$$w_i = \sin(\frac{\pi}{36}(i+\frac{1}{2})), \quad for\ i=0\ to\ 11 \tag{4.13}$$

当 block_type！=2 时，x_i 与 w_i 直接相乘，结果记为 z_i：

$$z_i = x_i * w_i \tag{4.14}$$

当 block_type=2 时，x_i 与 w_i 相乘后要经过如下运算，才能得到 z_i，y_i 为中间变量：

$$y_i^{(j)} = x_i^{(j)} \cdot \sin(\frac{\pi}{12}(i+\frac{1}{2})) \quad for\ i=0\ to\ 11, for\ j=1\ to\ 3 \tag{4.15}$$

$$z_i = \begin{cases} 0 & for\ i=0\ to\ 5 \\ y_{i-6}^{(1)} & for\ i=6\ to\ 11 \\ y_{i-6}^{(1)} + y_{i-12}^{(2)} & for\ i=12\ to\ 17 \\ y_{i-12}^{(2)} + y_{i-18}^{(3)} & for\ i=18\ to\ 23 \\ y_{i-18}^{(3)} & for\ i=24\ to\ 29 \\ 0 & for\ i=30\ to\ 35 \end{cases} \tag{4.16}$$

上述计算的结果是每个子带的 z_i。

以上步骤计算出该子带的前半部分样本值与上次计算结果的对应子带的后半部分样本值经过重叠运算得到新的样本值 s_i，当前块的后半部分被保留起来，用在下一个块中。

$s_i = z_i + prev_i$　　对于 i = 0 到 17　　(4.17)

$prev_i = z_{i+18}$　　对于 i = 0 到 17 [18]　　(4.18)

但是直接进行 IMDCT 的计算量很大，因此运用高效算法实现 IMDCT 在实时解码系统中十分重要。目前已经提出的快速算法有基于 FFT 的 IMDCT 蝶形快速算法（见参考文献[19]）、IMDCT 的递归实现（见参考文献[20]）等。在本系统中采用一种利用三角函数性质推导出来的快速余弦变换算法来实现 IMDCT。它的运算量更少，而且算法简单。对于这种快速算法的简单推导如下：

$$x_i = \sum_{k=0}^{\frac{N}{2}-1} X_k \cdot \cos\left(\frac{\pi}{2N}(2i+1+\frac{N}{2})(2k+1)\right) \tag{4.19}$$

式子(4.19)是 ISO/IEC Standard No. 11172-3 中规定的，其中 i = 0,1,2,…, N-1（短窗 N=12，长窗 N=36）。

令 $x_i^{'} = \begin{cases} x_{i+3N/4}, i = 0,\ldots,N/4-1 \\ x_{i-N/4}, i = N/4,\ldots,N-1 \end{cases}$　　逆运算为：$x_i = \begin{cases} x_{i+N/4}^{'}, i = 0,\ldots,3N/4-1 \\ x_{i-3N/4}^{'}, i = 3N/4,\ldots,N-1 \end{cases}$

易证：$x_{N/2+i}^{'} = \begin{cases} -x_{N/2-1-i}^{'}, i = 0,\ldots,N/4-1 \\ x_{N/2-1-i}^{'}, i = N/4,\ldots,N/2-1 \end{cases}$

令 $x_i^{''} = \begin{cases} -x_i^{'}, i = 0,\ldots,N/4-1 \\ x_i^{'}, i = N/4,\ldots,N/2-1 \end{cases}$　　逆运算为他本身

根据三角函数的特性，可以推导出：

$$x_i^{''} = \frac{1}{2\cos(2*i+1)\frac{\pi}{2N}} \sum_{k=0}^{N/2-1} (X_{k-1} + X_k)\cos(2i+1)k\frac{\pi}{N}, i = 0,\ldots,N/2-1 \tag{4.20}$$

令 $x_i^{'''} = x_i^{''} * 2\cos(2*i+1)\frac{\pi}{2N}$　$i = 0,\ldots,N/2-1$

令 $X_k^{'} = X_{k-1} + X_k, k = 0,\ldots, N/2-1$

令 N = 2M

（4.20）式变形为：

$$x_i^{''} = \sum_{k=0}^{M-1} X_k^{'} \cos(2i+1)k\frac{\pi}{2M}, i = 0,...,M-1 \qquad (4.21)$$

这样式子（4.19）可以通过N/2点的DCT运算（还需要一个因子）和N/2点的数据拷贝运算来完成。把式子(4.21)分成奇数列h_i和偶数列g_i如下：

令 $g_i = \sum_{k=0}^{M/2-1} X_{2k}^{'} \cos(2i+1)k\frac{\pi}{M}, i = 0,...,M/2-1$

令 $h_i = \sum_{k=0}^{M/2-1} X_{2k+1}^{'} \cos(2i+1)(2k+1)\frac{\pi}{2M}, i = 0...M/2-1$

利用Lee’s镜像概念可推导出：

$$x_i^{''} = g_i + h_i, i = 0,...,M/2-1 \qquad (4.22)$$

$$x_{M-1-i}^{''} = g_i - h_i, i = 0,...,M/2-1 \qquad (4.23)$$

利用三角函数的特性，把式子h_i变换成易于计算的形式：

令：$h_i^{'} = 2\cos(2i+1)\frac{\pi}{2M} * h_i, i = 0,...,M/2-1$

$$X_k^{''} = X_{2k-1}^{'} + X_{2k+1}^{'}, k = 0,...,M/2-1$$

$$h_i^{'} = \sum_{k=0}^{M/2-1} X_k^{''} \cos(2i+1)k\frac{\pi}{M}, i = 0,...,M/2-1 \qquad (4.24)$$

对于以上推导的详细过程见参考文献[21]。

观察以上的推导结果可知，对于长窗的情况，窗长为36的IMDCT运算可以最终简化成两个窗长为9的类DCT变换（即计算g_i和$h_i^{'}$的公式）。观察计算g_i和$h_i^{'}$的式子，可知这两个窗长为9的类DCT运算的变换核是相同的，这样使计算更加简化。同样对于短窗的情况，窗长为12的IMDCT运算简化成两个窗长为3的类DCT变换。在软件实现上，逆向利用上述推导过程：首先计算出g_i和$h_i^{'}$的值，再由$h_i^{'}$值算出h_i的值，从而可以根据公式4.22和4.23计算出$x_i^{''}$的值，再根据$x_i^{''}$、$x_i^{''}$、$x_i^{'}$和x_i之间的关系，最终计算出x_i。并且把变换核的值事先计算出来，计算时用到变换核时使用查表操作，这样又大大少了计算量，提高了程序的效率。

这种快速算法对于N＝36的长窗情况，乘法次数由648次减少到81次，加法次数由612次减少到149次，计算减少量分别为88%和76%；对于N＝12的短窗情况，乘法次数由72次减少到13次，加法次数由60次减少到27次，计算减少

量分别为82%和55%。

在本系统中，我们编写函数void Granule_imdct(struct Granule *gr, int ch, SS XX)使用以上推导的快速算法来完成逆向离散余弦变换、加窗运算和重叠相加运算。XX既作为入口参数来存放混叠重建后的样本值，又作为函数输出存放逆向离散余弦变换、加窗运算和重叠和相加运算后的样本值。入口参数*gr 存放边信息，ch表明当前正在处理的样本值所属的声道。

4.7 频率反转和子带合成

频率的反转是对逆向离散余弦变换的输出值中的奇数号子带（0到31号子带中的1，3，5，7，…，31号子带）中的奇数号样本值（每个子带中的0到17号样本值中的1，3，5，7，…，17号样本值）进行反相处理，用来补偿编码时为提高离散余弦变换效率而进行的频率反转。

在本系统中，编写子函数void Granule_freqinverse(SS X)来完成频率的反转。X 既作为入口参数来存逆向离散余弦变换后的样本值，又作为函数输出存放频率反转后的样本值。

子带合成滤波器将32个带宽相等的子带中的频域信号反变换成时域信号。子带合成是逆向离散余弦变换后的一个通道中32个子带的样值，经过一系列运算还原出32个PCM数字音频信号的过程。子带合成过程先将32个子带样值进行逆向离散余弦变换，生成64个中间值，将这64个中间值转入到一个长为1024点的类似先进先出FIFO的缓存，再在这1024个值中抽取一半，构成一个512个值的矢量，进行加窗运算，最后将加窗结果进行叠加生成32个时域输出。

在子带合成滤波器中，通过逆向离散余弦变换将32个等频带宽内的频域信号反变换成64个样值。公式如下：

$$x_i = \sum_{k=0}^{31} X_k \cdot \cos((16+i)(2k+1)\frac{\pi}{64}), \quad for\ i = 0\ to\ 63 \quad [22] \qquad (4.25)$$

子带合成的流程如图4.6：

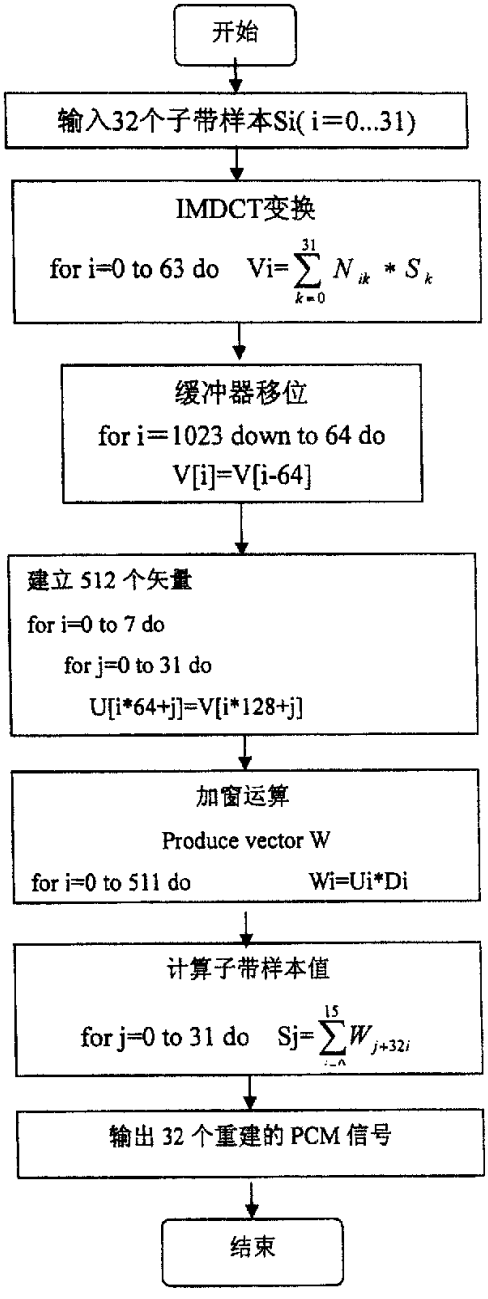


图4.6 子带合成流程图

Tab. 4.6 subband synthesis float chart

子带合成中的逆向离散余弦变换是MP3解码过程中计算量最大的部分，因此对于这部分的软件实现我们采用基于 ISO/IEC Standard No. 11172-3 给出的标准

公式推导出的一种快速算法。它采用 2 级 lee’s 快速 IDCT 算法来实现。在第一级利用奇蝶形运算和偶奇蝶形运算把长度为 32 的输入样本值分离成两个长度为 16 的矢量。在第二级中利用同上的方法把一级运算结果中的偶序列再分离成两个长度为 8 的的矢量。然后分别进行 16×16 和 8×8 的 IDCT 变换。从而减少了变换块的长度，提高了算法的效率。这种快速的推导过程见参考文献[23]。它的具体的实现过程如下：

(1)将 N（=32）点的输入矢量变成 3 个长分别为 N/2（=16）、N/4（=8） 和 N/4 （=8）的矢量。设输入矢量为 in[i], i=0, 1, …, 31，分解后形成 3 个矢量 ee[8], eo[8], odd[16]。

$$
\begin{aligned}
&even[i] = in[i] + in[N-1-i] \\
&odd[i] = in[i] - in[N-1-i] \qquad i = 0,1,\cdots,N/2 - 1 \\
&ee[i] = even[i] + even[N/2-1-i] \\
&eo[i] = even[i] - even[N/2-1-i] \qquad i = 0,1,\cdots,N/4 - 1
\end{aligned} \tag{4.25}
$$

(2)构造变换矩阵。偶矢量（ee[8]）的变换矩阵为正常的 IDCT 变换矩阵，而奇矢量（eo[8]、odd[16]）的变换矩阵则要在正常的 IDCT 变换矩阵的基础上做适当的调整。具体做法如下：设 N 点 IDCT 的变换矩阵为 DCT_N，则偶矢量的变换矩阵 $A_N = DCT_N$，奇矢量的变换矩阵 $B_N = G_N DCT_N H_N$，其中各个变换矩阵中的元素为：

$$
\begin{aligned}
&dct_{ij} = \cos[(2j + 1)i\pi/2N] \\
&g_{ij} = 1, \qquad j = i \text{ 或 } j = i + 1 \\
&h_{ij} = 1/2\cos[(2i + 1)\pi/4N], \quad i = j
\end{aligned} \tag{4.26}
$$

其余的 g_{ij}, h_{ij} 均为 0。

(3)进行矢量与矩阵相乘。将 3 个输入矢量分别与相应的矩阵相乘，得到 32 个输出矢量。设 odd，eo，ee，为别是 odd[16], eo[8], ee[8]的矢量形式，计算公式如下：

$$
\begin{aligned}
&out.odd = B_{16} * odd \\
&out.eo = B_8 * eo \\
&out.ee = A_8 * ee
\end{aligned} \tag{4.27}
$$

将 3 个输出矢量重新排序，即可得到 32 点 IDCT 的结果 out[32]：

$$out[2i+1] = out.odd[i] \qquad i = 0,1,\cdots,15$$

$$out[4i+2] = out.eo[i] \qquad i = 0,1,\cdots 7$$

$$out[4i] = out.ee[i] \qquad i = 0,1,\cdots 7 \tag{4.28}$$

(4)调整成最终的 IMDCT 结果。设 x[i]为 IMDCT 的最终结果，它和 out[i]之间的关系如下：

$$x[i] = out[i+16]$$

$$x[i+17] = -out[31-i]$$

$$x[i+32] = -out[16-i]$$

$$x[i+48] = -out[i]$$

$$x[16] = 0 \tag{4.29}$$

其中 i = 0,1,..,15。

该快速算法在运算量上的改进:N 点的 IDCT 需要 N×N 次乘法与 N(N-1)加法。因此,如直接计算 32 点的 IDCT,则需要 1024(32×32)次乘法与 992(32×31)次加法。而如果采用上述的快速算法,则只需要 1 个 16 点的 IDCT 与 2 个 8 点的 IDCT,即 384 次乘法与 352 次加法,另外再加上分解输入矢量所需的 24 次加法,总共为 384 次乘法与 376 次加法。运算量约为原来的 1/3。

对于单声道模式编写了函数 void Granule_subband_synthesis(int ch, SS s, short SAM[2][SSLIMIT][SBLIMIT])来实现子带合成。其中 ch 作为入口参数，标识是左声道单声道还是右声道单声道模式。s 作为入口参数存放频率反转输出的样本值。SAM[2][SSLIMIT][SBLIMIT]作为出口参数存放子带合成输出 PCM 数据。

对于双声道或立体声模式编写了函数 void Granule_subband_synthesis2(SS s1, SS s2, short SAM[2][SSLIMIT][SBLIMIT]) 来实现子带合成。其中 s1 作为入口参数来存放左声道频率反转输出的样本值，s2 作为入口参数来存放右声道频率反转输出的样本值，SAM[2][SSLIMIT][SBLIMIT]作为出口参数存放子带合成输出 PCM 数据。

4.8 MP3 的播放

从子带合成模块输出的数据就是 PCM 数据,这种数据格式需要经过音频 CODEC 转换成模拟信号然后播放出来。把 PCM 数据传送到 CODEC 我们采用 DMA 方式，这

样可以节省 CPU 资源，使同时解码和播放成为可能。为了保证音频信号能够随着解码过程实时的、连续的播放出来，我们需要对缓冲区进行设置。

双声道的一帧 MP3 数据能够还原出 1152*2 个 PCM 样本值，每声道 1152 个样本。为了能及时播放解码后的 PCM 数据，在程序中设置了两个 1152*4 大小的缓冲区。一个缓冲区用来存放已经解码完毕的一帧 PCM 数据，另一个缓冲区存放正在解码的 PCM 数据。对于每一个缓冲区用一个标志位来表示它的四种状态：空（free）、满（full）、正在被解码程序占用（filling）、正在在 DMA 方式下向 CODEC 传送 PCM 数据（sending）。

首先，解码程序查看是否有状态为空的缓冲区，如果没有则等待，否则占用状态为空的一块缓冲区，开始把解码得到的 PCM 数据存放到该缓冲区中，此时该缓冲区的状态为 filling。当缓冲区被填满之后，它的状态被设置成满（full），此时 DMA 操作被允许，如果 DMA 操作启动，那么它的状态为 sending。同时，解码程序可以占用另外一个缓冲区，继续进行解码操作。当 DMA 操作结束后，产生 DMA 中断，我们在中断服务子程序中把该缓冲区的状态该为空（free），则它又可以被解码程序所占用了。就这样，两个缓冲区被交替使用，实现了实时解码播放。

第5章 系统硬件设计

5.1 微处理器选型

ARM（Advanced RISC Machines），既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。[24]

1991 年 ARM 公司成立于英国剑桥，主要出售芯片设计技术的授权。目前，采用 ARM 技术知识产权（IP）核的微处理器，即我们通常所说的 ARM 微处理器，已遍及工业控制、消费类电子产品、通信系统、网络系统、无线系统等各类产品市场，基于 ARM 技术的微处理器应用约占据了 32 位 RISC 微处理器 75%以上的市场份额，ARM 技术正在逐步渗入到我们生活的各个方面。

目前非常流行的 ARM 内核有 ARM7TDMI、StrongARM、ARM720T、ARM9TDMI、ARM920T、ARM10TDMI、ARM11 等。其中 ARM7TDMI 是 ARM 公司最早为业界普遍认可且赢得了最广泛应用的处理器核。[25]

Samsung 公司出品的 S3C44B0X 16/32 位微处理器片内集成 ARM7TDMI 核，采用 0.25μm CMOS 工艺制造，并在 ARM7TDMI 核基本功能的基础上集成了丰富的外围功能模块，便于低成本设计嵌入式应用系统。片上集成的主要功能如下：

- 在 ARM7TDMI 基础上增加 8KB cache;
- 外部扩充存储器控制器（FP/EDO/SDRAM 控制，片选逻辑）；
- LCD 控制器（最大支持 256 色 DSTN），并带有一个 LCD 专用 DMA 通道;
- 两个通用 DMA 通道，两个带有外部请求引脚的 DMA 通道;
- 两个带有握手协议的 UART，一个 SIO;
- 一个多主的 IIC 总线控制器;
- 一个 IIS 总线控制器;
- 五个 PWM 定时器和一个内部定时器;
- 看门狗定时器;
- 71 个通用可编程 I/O 端口，8 个外部中断源;
- 电源控制方式：正常，低，休眠和停止;
- 8 路 10-bit ADC;
- 具有日历功能的 RTC（实时时钟）；

- PLL 时钟发生器。[28]

这款微处理器最高支持 66MHz 的频率，能够满足 MP3 软件解码的速率要求。它提供的 IIS 总线控制器和支持的 DMA 数据传输方式为连接立体声解码电路 CODEC 提供了极大的方便。而且 S3C44B0X 的价格十分低廉，只有几十元，与那些高端处理器和 MP3 专用解码芯片相比，作为本次研究性毕业设计的微处理器是非常适合的。它的引脚设置如图 5.1。

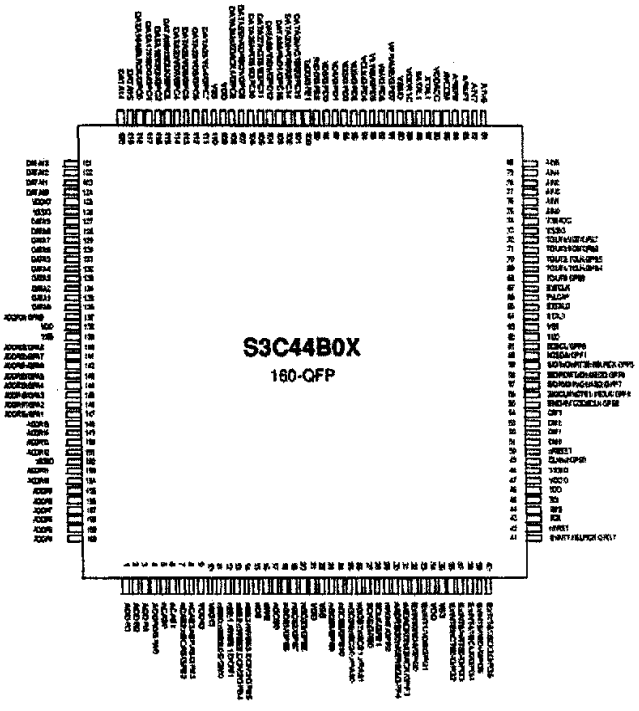


图5.1 S3C44B0X引脚图 (160 LQFP)

Fig. S3C44BOX pins assignment(160 LQFP)

5.2 存储模块设计

在现代的片上系统设计中，为了实现高性能，微处理器核必须连接一个容量大、速度高的存储器系统。设计一个足够大又足够快的单一存储器，使高性能处理器充分发挥其能力，是有一定困难的。一般的解决方法是构建一个复合的存储器系统，这就是普遍使用的多级存储器层次的概念。多级存储器包括一个容量小

但速度快的从存储器以及一个容量大但速度慢的主存储器。[27] ARM 的存储器包括寄存器组、片上 RAM、片上 Cache、主存储器等。

5.2.1 片上 Cache 的使用

S3C44B0X 芯片上有一个 8KB 的 Cache。它有三种使用方式。第一种是全部 8KB 空间作为指令/数据 Cache，这样将减少取指时间，大大提高程序的执行效率。第二种是将全部 8KB 空间作为内部 SRAM 使用。内部 SRAM 与外部 RAM 之间的区别在于：在 DMA 过程中，外部 RAM 由 DMA 控制器锁定，处理器没有权利访问，但是可以访问内部 RAM。第三种是 4KB 空间作为内部 SRAM 使用，另外 4KB 空间作为指令/数据 Cache。在本系统中我们传输 MP3 解码后的 PCM 数据到立体声解码电路 CODEC 采用 DMA 方式，为了防止在 DMA 过程中，当执行到访问内存指令时，处理器只能停下来等待 DMA 结束才能继续执行这种情况的发生，我们要使用内部 SRAM。同时为了提高取指效率，我们折中采用第三种使用方式来分配片上 Cache。

5.2.2 外部存储器的设计

外部存储器包括三部分：存储程序、存储 MP3 数据和程序运行时需要的 RAM。

Flash存储器有高可靠性读写、非易失性、可擦写性、操作简便和低成本等特点，所以本系统选用Intel TE28F320 FLASH芯片，容量为4MB，16位数据宽度，用于存放代码和一些只读的数据，包括系统的配置信息，MP3解码需要的一些参数表等等。其地址映射到S3C44B0X的BANK0。选用三星的大容量与非存储器KM29U128T[28]来存储mp3数据，容量为128M。其地址映射到BANK1、BANK2、BANK3和BANK4。选用的RAM是Winbond 986416DH，它是8M，16位数据宽度的SDRAM存储器。地址映射到S3C44B0X的BANK6。地址对应关系见表5.1。

表5.1 外部地址映射

Tab. 5.1 Memory map

Address Map	Chip	Start Address	End Address	Size
nGCS0	Flash	0x00000000	0x00400000	4MB
nGCS1- nGCS4	Flash	0x02000000	0x0a000000	128MB
NGCS6	SDRAM	0x0c000000	0x0c800000	8MB

5.3 播放模块设计

要播放 MP3 数据解码后的 PCM 数据需要立体声解码电路 CODEC。CODEC 与 S3C44B0 之间可以使用多种数据接口，如 SPI、IIS 等。三星 S3C44B0 芯片内嵌一个 IIS 总线控制器，它支持 IIS 数据格式，所以我们选择 IIS 数据接口。菲利浦公司出品的立体声解码芯片 UDA1341TS 支持 IIS 数据接口，而且支持 PCM 这种数据格式作为输入，所以本系统选择它作为 CODEC。

5.3.1 IIS 数据接口的选择

IIS(Inter-IC Sound bus)又称 I^2S，是菲利浦公司提出的串行数字音频总线协议。IIS 总线只处理声音数据。其他信号(如控制信号)必须单独传输。为了使芯片的引出管脚尽可能少，IIS 只使用了三根串行总线。这三根线分别是：提供分时复用功能的数据线（在三星公司的 ARM 芯片中，为了实现全双工模式，使用了两条串行数据线，分别作为输入和输出）、字段选择线(声道选择)、时钟信号线。

1. IIS 总线概述

在 IIS 总线中，串行数据以 2 的补码形式表示，首先将 MSB（最高有效位）发送出去。MSB 首先被发送是因为发送器和接收器可能有不同的字长，没有必要让发送器知道接收器能处理多少位，也没有必要让接收器知道有多少位正在被发送。当系统字长大于发送器的字长时，为了能够传输，该字被删除一部分（最低有效位被置 0）。如果发送的数据大于接收器的字长，最低有效位之后的位被忽略。另一方面，如果接收器的字长大于它接收的数据长度，不足的位将被在内部置 0。因此，最高有效位（MSB）有一个固定的位置，而最低有效位的位置取决于字长。发送器总是在声道选择信号改变后的下一个时钟周期发送一个字的 MSB。

串行数据可在时钟信号的上升沿或下降沿被同步。然而，串行数据必须在连续时钟信号的上升沿被锁存到接收器。因此，当发送数据用上升沿被同步时，会有一些限制。

LR 通道选择信号指示正在发送数据的通道。声道选择信号在连续时钟的上升沿和下降沿均可改变，但是不需要同步。在从模式下，该信号在时钟的上升沿被锁存。声道选择信号在 MSB 被发送前改变一个时钟周期，这就允许从发送器和即将发送的数据同步。此外，它还通知接收器存储前一个字，并为下一个字清空输

入。[29]图 5.2 为 IIS 总线数据接口格式。

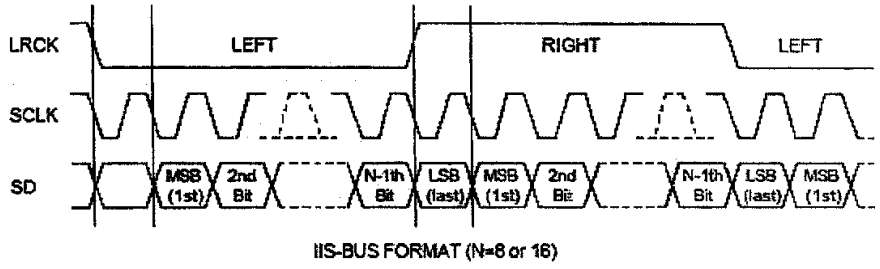


图5.2 IIS总线数据接口格式

Fig. 5.2 Timing for IIS bus data interface format

S3C44B0X 的 IIS（Inter－IC Sound）总线接口可作为一个编解码接口与外部 8/16 位的立体声的立体声音解码电路（CODEC IC）相连，它支持 IIS 数据格式和 MSB-Justified 数据格式。IIS 总线接口为先进先出度列 FIFO 的访问提供 DMA 传输模式来取代中断模式，可同时发送和接收数据，也可以只发送或接收数据。

S3C44B0X IIS 特性：IIS、MSB-Justified 格式兼容；每通道 8/16 位数据格式；每通道有 16fs、32fs、48fs（fs 为采样频率）的串行位时钟；具有 256fs 和 384fs 的主时钟；具有为主时钟和编解码时钟分频的可编程分频器；支持 32（2*16）字节发送和接收（FIFO）；具有正常和 DMA 两种传输模式。

- 正常传输模式

对于发送与接收 FIFO，IIS 控制寄存器有队列（FIFO）就绪标志位。如果发送队列非空，当队列发送数据准备就绪时，就绪标志位被置 1；如果发送队列为空，就绪位被置 0。在接收队列未满的情况下，队列接收数据的就绪标志位被置 1，指示队列可接收数据；如果接收队列满，就绪标志位被置 0。这些标志位决定了 CPU 读或写队列的时间。通过这种方式，当 CPU 访问发送或接收队列（FIFOs）时，串行数据能够被发送或接收。

- DMA 传输模式

在 DMA 传输模式中，发送或接收队列（FIFO）的访问是由 DMA 控制器来完成的。在发送或接收模式中，DMA 服务请求由队列的就绪标志位自动给出。

2. 基于 S3C44B0X 的 IIS 接口配置

MP3 数据的压缩比一般在 10:1 到 12:1 之间。通常使用的一帧 MP3 数据将近 500 字节，那么解压之后就要有将近 5000 字节。这样 CPU 要将这将尽 5000 字节的数据从内存搬到 CODEC，会有很大的时间开销，所以在本 MP3 解码系统中，采用 DMA 内存访问模式。

S3C44B0X 芯片，它提供了 4 个 DMA 通道，2 个 ZDMA 和 2 个 BDMA。ZDMA 就是 General DMA，它是用在 SSB (Samsung System Bus)总线上，主要适用于在 SSB 总线上的高速设备的数据传输，如内存，I/O 内存，高速的设备。BDMA 是 Bridge DMA，它是用在 SPB (Samsung Peripheral Bus)总线上，主要用于内存和外设之间的数据传输，如 SIO, IIS 以及 UART 等。系统是通过 IIS 接口将数据传给 CODEC 的，所以 BDMA 将作为本系统的数据传输模式。

UDA1341TS 只能应用在从模式下，这意味着在所有的应用中，系统设备必须要提供系统时钟。3C44B0X 的 IIS 接口会提供 3 个时钟，分别为 CODECLK，IISLRCK，IISCLK。其中 CODECLK 为立体声解码电路的工作频率，定义为 256fs 或 384fs(fs:sampling frequency)。IISLRCK 是送 PCM 数据是切换左右声道的频率，在数值上等于 fs。IISCLK 是送 PCM 数据是每个比特的打入频率，数值上为 16fs，32fs 或 48fs。所有的这些频率都是由 CPU 的主频 MCLK 来产生。IIS 的分频系统见图 5.3。

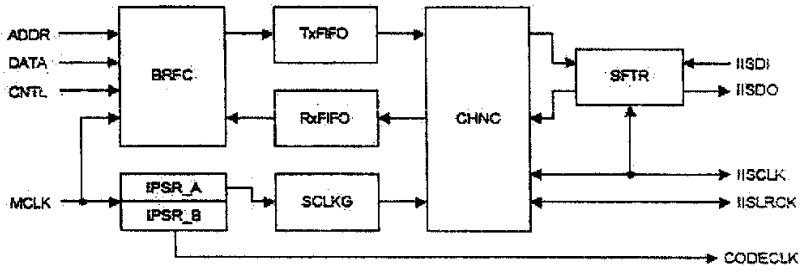


图5.3 IIS的分频系统

Fig. 5.3 IIS bus block diagram

由图 5.3 可知，CODECLK 是微处理器的主频由 IPSR_B 直接分频得到的，而

IISCLK 和 IISLRCK 是由主频通过 IPSR_A 分频后再经 SCLKG 处理后的到的。因为 MP3 采样率 fs 为 44.1KHz，所以播放的频率也为 44.1KHz，IISLRCK 在数值上和采样频率相等，所以 IISLRCK=44.1KHz。IISCLK 是采样频率的 16 倍，32 倍或 48 倍，这和采样的数据宽度有关，MP3 一般采用的是 16bit 的编码，左右声道加起来为 32bit，所以 IISCLK 为 32fs，即 IISCLK=1.4112MHz。CODECLK 是采样频率的 256 倍或 384 倍。它是由主频 MCLK 分频而来的。我们选择 CODECLK=256fs，是主频的 6 分频，则 CODECLK=11.287879MHz，主频为 67.727273MHz。

5.3.2 CODEC 芯片的选择

菲利浦公司出品的UDA1341TS是一个单片立体声模数和数模转换器，它带有提供比特流转换技术的信号处理特征。它支持IIS数据格式；支持256fs、384fs，512fs的系统时钟；可以通过L3-interface控制芯片功能。以上功能完全能满足本系统的需求。它的引脚排列见图5.4。

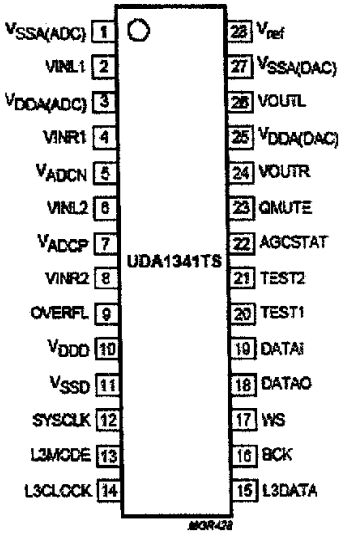


图5.4 UDA1341引脚图

Fig. 5.4 UDA1341 pins assignment

在微控制器和UDA1341TS之间控制信息的交换是通过一个串行硬件L3接口完成的。这个接口是由以下引脚组成：L3DATA(微控制器接口数据线)；L3MODE（微

控制器接口模式线）；L3CLOCK（微控制接口时钟线）。

通过 L3 接口传输的信息遵照所谓的‘L3’格式。在这种格式下，可以分为两种不同的操作模式，地址模式和数据传输模式。

当要通过 L3 总线选择一个器件或为数据传输模式定义一个目标寄存器时需要使用地址模式。数据传输可以是双向的：输入到 UDA1341TS 中，规划声音的处理和系统控制；从 UDA1341TS 输出来提供峰值。地址模式是由 L3MODE 持续为低，和 L3CLOCK 上连续 8 个脉冲来标识，同时传输 8bit 的数据。时序见图 5.5。

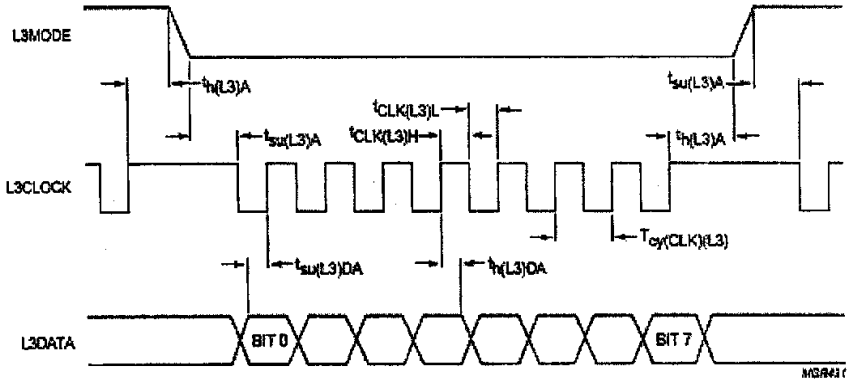


图5.5 地址模式时序图

Fig. 5.5 Timing address mode

图 5—5 中，数据的 7-2bit 代表一个 6 位的器件地址，bit7 是 MSB，bit2 是 LSB，1-0bit 表明了后面要传输的数据类型。UDA1341TS 的地址是 000101。如果 UDA1341TS 收到了一个不同的地址，它会取消选择它的控制器接口逻辑。在地址模式下选择的地址在后面数据的传输过程中会一直保持有效，直到 UDA1341TS 接收到了一个新的地址命令。[30]

数据模式的时序和地址模式的时序基本相同，字节传输的示意图见图 5.6。

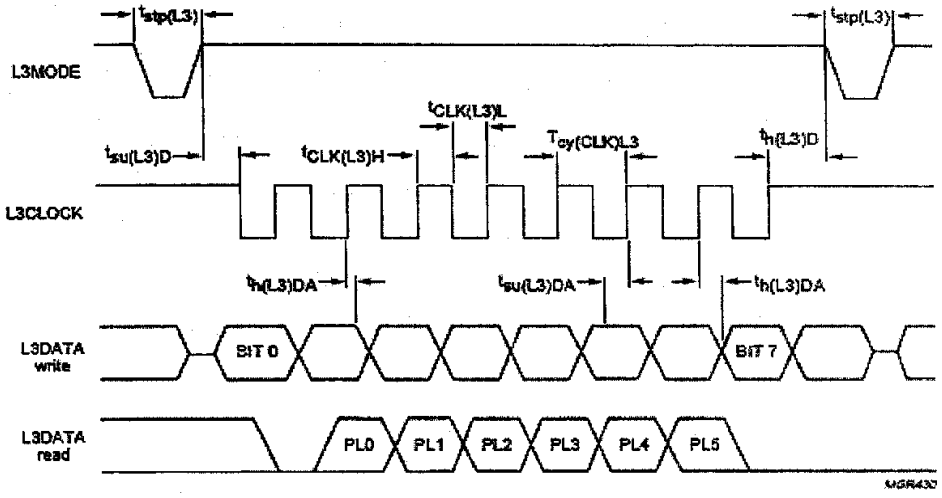


图5.6 字节传输示意图

Fig. 5.6 Timing for data transfer mode

微处理器S3C44B0与UDA1341TS音频芯片的连接见图5.7。

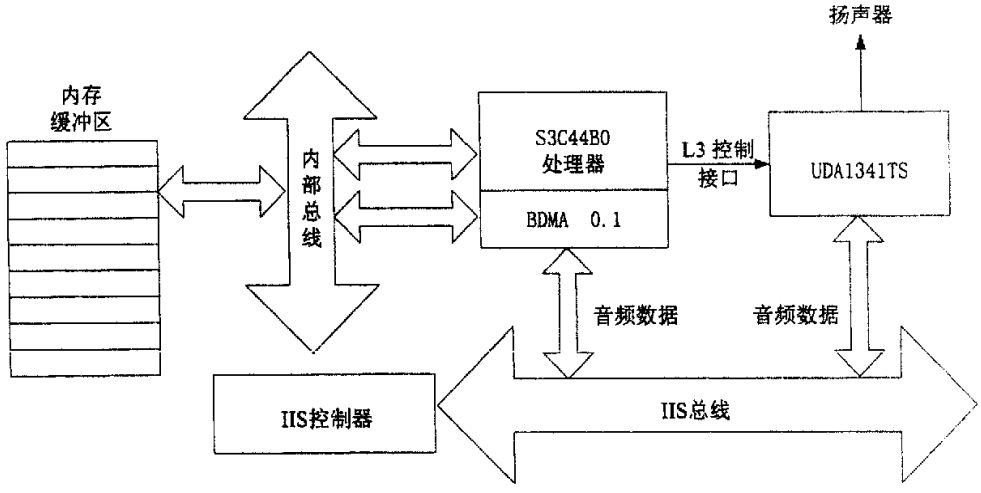


图5.7 S3C44B0与CODEC的连接示意图

Fig. 5.7 The connection of S3C44B0 and CODEC

第6章 结论与展望

MPEG LAYER 3 的解码算法十分复杂，所以以往所设计的 MP3 解码器主要是利用 DSP 芯片来实现。然而 DSP 芯片的成本通常比一般的微处理器高出许多，其代价包含了电力消耗以及 DSP 成本等。随着 ARM 技术的成熟，ARM 微处理器的性能的提高，以及 MPEG LAYER 3 解码算法的优化，本系统采用了 ARM7 家族中的 S3C44B0X 这款微处理器来实现 MP3 文件的实时播放。

MP3 解码运算量主要集中在哈夫曼解码、反量化、IMDCT 变换和子带合成这四个步骤。在解码软件的实现上，主要对这些步骤进行了优化：在哈夫曼解码过程中，采用了折半查找算法来提高查找效率；在反量化过程中，对反量化公式进行变形，从而避免了复杂的指数运算；在 IMDCT 变换过程中，利用三角函数的性质和公式中变换结果的相关性，采用了一种新的快速算法，减少了大约 1/3 的运算量；在子带合成中的 IMDCT 变换过程中，采用了 lee’s 快速算法来减少运算量。

在系统调试过程中，使用 ARM 公司提供的 ADS 集成开发环境来仿真存储器的访问时间，从而量化 MP3 的解码效率。在主频为 67.727273MHz 的情况下，以一段采样率为 44.1kHz、比特率为 128kbit/s 的立体声 MP3 数据为例，对其中 500 帧进行解码，共耗时 9600ms。平均每帧解码时间为 19.2ms。每帧解码过程各个步骤的平均解码时间如表 6.1：

表6.1 MP3解码各个步骤解码时间表

Tab. 6.1 Time consume of each step of MP3 decoding

哈夫曼解码	反量化	IMDCT 变换	子带合成	其余步骤
3.84ms	1.77ms	3.40ms	9.01ms	1.18ms

这样的解码性能达到了设计指标，是可以满足解码要求的。此外，解码后通过音频CODEC播放出来的声音，音质良好，效果令人满意。

由于时间有限，在此次毕业设计期间，实现了对于单段MP3音乐的解码与播放，如果要实现真正的实用化，使它成为一种便携式的消费品，还有很多地方有待于改进和扩展：

1. 目前只实现了对一段MP3数据的解码和播放，不能对多文件进行解码和播放。可以通过添加文件系统来实现多文件的播放。
2. 在这个系统中，存储MP3数据的存储器采用的是FLASH，其容量有限，可以考虑用硬盘作为存储器，其容量可大大增加。
3. 与PC机之间的数据传输这部分功能还没有实现，可以采用目前比较流行的USB方式或者网络传输。

参 考 文 献

[1] 许盈.MPEG 与 MP3 技术的发展.技术市场，2005，5：96.

[2] 杨俊，蔡宣平，颜飞翔.数字音频技术及其应用与发展（二）.电声技术，2001，6：12.

[3] 李静芳.MP3 播放器，路在何方？.电子技术，2005，5：3.

[4] 何业军，陈永泰.数据编码与压缩技术.电信快报，2001,6:37.

[5] 余永聪，尹俊勋.音频压缩编码中的参数比特分配技术.电声技术，2001，7：6-7.

[6] Ken C.Pohlmann 著.苏菲译.数字音频原理与应用（第四版）.电子工业出版社，2002.

[7] 王炳锡.语音编码.西安电子科技大学出版社，2002.

[8] 陈海燕.MPEG1 低码率音频压缩算法.通信世界，2000，5：14.

[9] 高智衡，韦岗.MP3 宽带音频压缩中的核心技术.电声技术，2000,9:10.

[10] 付中华,王大炜,赵荣椿等.一种基于 MP3 框架的低采样频率音频压缩方案.计算机工程，2003，11：146-149.

[11] http://www.dv.co.yu/mpgscript/mpeghdr.htm#MPEG.

[12] 王文林.MP3 播放机维修技术.新时代出版社，2004.

[14] 付轩，陈健，徐盛.MP3 编码量化模块的改进.电声技术，2004，11：2-3.

[13] 江巍,杨军，罗岚等.MP3 定点解码算法的设计与实现.计算机工程，2004，8：83

[15] 许林.MP3 解码及技术难点.电声技术，2003,04:64-65

[16] 郝育闻.MP3 解码器软件实现学位：（硕士论文）.大连理工大学，2000.

[17] ISO/IEC 11172-3. Coding of Moving Picture and Associated Audio for Digital Storage Media at up to About 1.5Mbit/s (Part 3) Audio.1993.

[18] 苏祖辉.IMCDT 在 MPEGAudio-1 LayerⅢ中的递归实现：（硕士论文）.安徽：合肥工业大学，2004.

[19] 窦维蓓，刘若珩，王建昕等.基于 DSP 的 IMDCT 快速算法.清华大学学报，2000，Vol.40,No.3:99-102.

[20] 邓宁.运用递归算法实现共用的 MDCT 和 IMDCT 结构.电声技术，2005，1：47-48.

[21] ScottB.Marovich,Fast MPEG-1 Ⅲ Audio Decoding,June.2000

[22] 计丹. 基于定点 DSP 的 MP3 解码系统设计与实现：（硕士论文）.华中师范大学，2002.

[23] 王剑虹，吴海华，陈健.MPEG 音频解码中子带合成滤波器的快速算法及定点 DSP 实现.上海交通大学学报，2000，6.

[24]http://www.arm.com.

[25] 华锡锋，蒋忠平，罗明清等.基于 ARM7TDMI 的 SoC 中 MP3 子系统的设计.电子工程师，2004，12：1.

[26] S3C44B0X RISC Microprocessor Datasheet , Version 1.0 , SAMSUNG Electronics ,2000.

[27] 田泽.嵌入式系统开发与应用.北京：航空航天大学出版社，2005.

[28] 窦振中著.单片机外围器件实用手册一存储器分册.北京：航空航天大学出版社，1999.

[29] I^2S bus specification . Philips Semiconductors,1986：1-7

[30] UDA1314TS Product specification. Philips Semiconductors,2001：1，11

攻读学位期间公开发表的论文

[1] 李菁菁，滕国库. 高速公路车辆自动识别系统一车载机和终端设备的硬件设计. 中国计算机学会，2004全国企业管理控制一体化和软件技术研讨会论文集. 2004年6月， pp310-312.

[2] 李菁菁. 基于ARM的MP3解码优化算法. 大连海事大学，科技创新论文集. 已录用.

致　谢

在本课题的研究和论文的写作期间，我得到了很多老师和同学的热心帮助，在此向他们表示真挚的感谢。

首先，感谢我的导师滕国库老师，在毕业设计阶段一直关注课题的进展情况，在技术难点上给我很多启发性的参考意见。感谢滕老师给予我的悉心教诲和指导，从滕老师那里我不仅学习了知识技能，还学会如何为人处事。这些都将使我终身受益。同时感谢计算机学院所有给予我鼓励和帮助过我的老师们。

还要感谢三年来与我同窗的同学。正是和你们的交流才给了我更多更好的完成论文的灵感。一路上有你们的陪伴，我的生活才更加丰富多彩。

最后谨以此文献给我在抚顺的父母。当我遇到困难时，是你们永远支持、鼓励我，总能让我充满信心的面对生活！在以后的日子里，我会更加努力的工作、学习，让你们因我而更加骄傲。

研究生履历

姓　名	李菁菁
性　别	女
出生日期	1981 年 3 月 31 日
获学士学位专业及门类	工学
获学士学位单位	大连海事大学
获硕士学位专业及门类	工学
获硕士学位单位	大连海事大学
通信地址	大连海事大学 9 舍 622 室
邮政编码	116026
电子邮箱	jing@dlmu.edu.cn

MP3软件解码器的研究与实现

作者：　　李菁菁
学位授予单位：　　大连海事大学

参考文献(30条)

1. 许盈 MPEG与MP3技术的发展 2005(05)

2. 杨俊. 蔡宣平. 颜飞翔 数字音频技术及其应用与发展(二)[期刊论文]-电声技术 2001(6)

3. 李静芳 MP3播放器,路在何方?[期刊论文]-电子技术 2005(5)

4. 何业军. 陈永泰 数据编码与压缩技术[期刊论文]-电信快报 2001(6)

5. 余永聪. 尹俊勋 音频压缩编码中的参数比特分配技术[期刊论文]-电声技术 2001(7)

6. Ken C Pohlmann. 苏菲 数字音频原理与应用 2002

7. 王炳锡 语音编码 2002

8. 陈海燕 MPEG1低码率音频压缩算法 2000(05)

9. 高智衡. 韦岗 MP3宽带音频压缩中的核心技术[期刊论文]-电声技术 2000(9)

10. 付中华. 王大炜. 赵荣椿. 谢磊 一种基于Mp3框架的低采样率音频压缩方案[期刊论文]-计算机工程 2003(19)

11. 查看详情

12. 王文林 MP3播放机维修技术 2004

13. 付轩. 陈健. 徐盛 MP3编码量化模块的改进[期刊论文]-电声技术 2004(11)

14. 江巍. 杨军. 罗岚. 胡晨 MP3定点解码算法的设计与实现[期刊论文]-计算机工程 2004(15)

15. 许林 MP3解码及技术难点[期刊论文]-电声技术 2003(4)

16. 郝育闻 MP3解码器软件实现[学位论文]硕士 2000

17. ISO/IEC 11172-3.Coding of Moving Picture and Associated Audio for Digital Storage Media at up to About 1.5Mbit/s (Part 3) Audio 1993

18. 苏祖辉 IMDCT在MPEG/Audio-1 LayerIII中的递归实现[学位论文]硕士 2004

19. 窦维蓓. 刘若珩. 王建昕. 董在望 基于DSP的IMDCT快速算法[期刊论文]-清华大学学报（自然科学版） 2000(3)

20. 邓宁. 周源华. 郭凯 运用递归算法实现共用的MDCT和IMDCT结构[期刊论文]-电声技术 2005(1)

21. ScottB Marovich Fast MPEG-1IIIAudio Decoding 2000

22. 计丹 基于定点DSP的MP3解码系统设计与实现[学位论文]硕士 2002

23. 王剑虹. 吴海华. 陈健 MPEG音频解码中子带合成滤波器的快速算法及定点DSP实现[期刊论文]-上海交通大学学报 2000(6)

24. 查看详情

25. 华锡锋. 蒋忠平. 罗明清. 凌明 基于ARM7TDMI的SoC中MP3子系统的设计[期刊论文]-电子工程师 2004(12)

26. S3C44B0X RISC Microprocessor Datasheet,Version 1.0 2000

27. 田泽 嵌入式系统开发与应用 2005

28. 窦振中 单片机外围器件实用手册--存储器分册 1999

29. I2S bus specification 1986

30. UDA1314TS Product specification 2001

相似文献(5条)

1.期刊论文 熊赟.于鸿洋.Xiong Yun.YU Hong-yang H.264实时软件解码器的实现 -中国有线电视2005,""(6)

对H.264测试模型JM8.4进行优化和改进.从分析模型结构入手,对解码流程进行调整.针对函数模块的耗时情况,采用SIMD技术对重要解码函数进行改写,使解码速度提高,在主流的PC机上可以对VGA和cif格式的H.264序列实时解码.

2.学位论文 王燕 基于OpenRISC的音频解码器软硬件协同设计 2007

本文以在OpenRISC上设计实现低功耗音频解码器为研究目标，根据实际设计的需求，运用不同的软硬件协同设计方法，分别实现MPEG-1/2 BC纯软件解码器、硬件协处理器加速的面向DVB标准的音频解码器，并提出以更加灵活高效的扩展指令集方式实现多标准低功耗音频解码器。

本文对MPEG-1/2 BC核心编码技术子带分析滤波和MDCT进行分析研究。在对原理理解的基础上，优化标准子带合成滤波算法的流程，只需计算一半滤波系数，并且采用FFT快速算法，减少运算量和中间数据的存储，改变加窗过程的运算顺序，减少软件解码器频繁读取Memory次数。针对传统Huffman解码流程繁琐冗余的缺点，本文根据Huffman码值的前缀码特性，改造码表，利用读入的码字作为码表的索引，最多只需搜索两次就能解出码值。

本文对MPEG-1/2 BC软件解码器从三方面进行优化：解码流程、算法、汇编级，分析解码器系数的动态范围，对动态范围大的变量采用不同精度定点化，从而在OpenRISC上实现纯软件低功耗高品质音频解码器。

在面向DVB解码芯片的设计中，CPU处理完基本任务有富余的处理能力，本文安排音频解码的主体部分由CPU完成，软件不方便实现的getbits模块(不定比特读数据)采用硬件协处理器实现，该设计不仅仅完成getbits功能，还相当于给音频解码软件加了一个DMA数据通道。解码器的运算瓶颈子带合成滤波也由硬件加速完成。本文改变传统做法，把一直由音频播放实现的音视频同步控制交给CPU完成，使得音视频同步控制的灵活性增大，音频播放模块设计简化。

针对设计多标准低功耗的音频解码器的设计目标，文中以MPEG-1/2 BC、 MPEG-4 AAC和AC-3音频标准为例子，阐明了基于OpenRISC的扩展指令集的设计方法。通过分析它们的解码复杂度，提取多条在各个标准中都频繁调用且需多条汇编指令才能完成的关键指令，用以在后续工作中扩展特殊指令，实现高效率的多标准音频解码器。

3.学位论文 武晓燕 H.264/AVC解码器的软件实现及优化 2006

H. 264是开放统一的视频编解码标准，对于解码兼容性有明确的规定，它能够把提高视频图像的传输效率和为用户提供更高质量的画面两个目标结合起来，所以该标准被认为技术先进，代表未来发展的方向。特别是IPTV成为中国通信产业发展的一个新的热点，H. 264以其各方面的优势成为业界看好的标准之一。这就必然要求解码端能够支持各方面的应用，特别是在H. 264的高压缩性能是以高复杂度为代价的，为满足视频通信终端的应用要求必须对其进行性能上的优化。而以JVT推出的JM模型是开放的最具有代表性的软件模型，对其的研究和优化具有实际的意义。

本文围绕H. 264 / AVC标准展开，首先介绍了H. 264 / AVC编解码原理，着重对其新发展FREXt和SVC扩展集分别进行了详细说明。

其次对H. 264 / AVC基类解码器实现的空间复杂度和关键子函数实现的时间复杂度分别进行了理论分析，并且讨论了硬件实现时影响解码子函数时间复杂度的因素。存储复杂度由实现一个算法需要的近似内存的大小来测量，时间复杂度由执行一个算法的特定实现需要的近似基本操作数目来评估。在分析结果中给出了在PHILIPSTriMediaPNX1502为核心的硬件平台上进行软件实现的可行性分析及影响各子函数设计的因素。

最后，针对JM9. 6模型软件解码器进行优化。优化过程主要分为算法的优化和程序级的优化。算法的优化是分别针对解码器实现的各个模块，如宏块解码、环路滤波等，分别进行解码流程和算法实现的优化。而程序级别的优化主要是利用通用的程序优化方法，如用&和移位来分别代替取模和乘除运算，循环展开等。同时给出了优化后各个函数模块占用时间百分比，并且与优化前相比，性能有明显提高。

4.期刊论文 王华明.陈健 MPEG-2 AAC音频编码技术及其软件解码器的实现 -计算机工程2001,27(6)

MPEG-2 AAC(ISO/IEC13818-7)是一种新型高效多声道Hi-Fi音频编码标准,在每声道64kbps时重建音质优于MP3和AC-3.分析了它的编码技术特点,着重研究了它的解码流程,讨论了对其关键模块的优化措施.同时提出了基于PC的AAC软件解码播放器的实现方法,对其他类型的音频解码器也具有参考价值.实现了在Pentium Ⅱ 350、64MB内存的PC机上对AAC(Main Profile)压缩文件的软件解码播放.

5.学位论文 施澍 基于UniCore32体系结构的MPEG视频解码优化 2007

随着多媒体应用重要性的不断提升，对多媒体处理的支持也逐步成为各处理器设计重要的指标。MPEG 视频解码作为多媒体应用中的重要组成部分，以其计算复杂、数据密集以及严格的实时性对处理器的处理能力提出了很高的要求。UniCore32 是北京大学微处理器研究开发中心自主研发的微处理器体系结构，在实际应用中，为更好地加强其对 MPEG 视频解码的支持，需要对基于UniCore32体系结构的视频处理计算进行优化。

本文通过系统分析MPEG-1、MPEG-2和MPEG-4标准，研究软件解码器的解码流程，调研多媒体处理的发展历史，提出了基于UniCore32体系结构的优化策略，并从软件和硬件两个方面进行了优化。在软件方面，一是利用UniCore32指令系统中的特殊指令和体系结构的特点，结合SIMD的思想对解码器进行优化；二是在解码过程中压缩视频的分辨率，通过损失部分分辨率和图像质量来提升解码性能。在硬件方面，通过设计支持MPEG视频解码的加速器，采用基于硬件加速器的软硬件协同处理可以进一步增强 UniCore32体系结构的视频处理性能。本文着重分析了硬件设计前的软硬件划分和硬件实现后期的软硬件协同两部分工作。通过实验数据评测，经过优化后的UniCore32体系结构的视频解码性能提升可以超过200%，能有效地支持MPEG-1、MPEG-2 MP@ML和MPEG-4 AsP@L5视频的实时解码。