独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。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了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：李琴琴　　日期：2010.3.12

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。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，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：李琴琴　　日期：2010.3.12

导师签名：　　日期：2010.3.12

VoIP 系统中 AGC 算法的研究与 DSP 实现

摘 要

随着互联网的快速发展，互联网语音通信应用日益普及，与传统电话相比，IP 电话以其网络带宽利用率高、通话费用低、承载业务多而得到广泛应用。与传统电话网不同，IP 电话网将所有的话音打包成 IP 包通过 IP 寻址的方式传送到对端，因此就产生了一些问题，包括延时、抖动、回声等现象。为了消除这些影响，需要对语音信号进行预处理，通过处理可以大大改善原系统在外界环境干扰条件下的性能，提高语音通信质量。

实用语音预处理系统主要包括降噪系统、回声控制系统、语音激活检测模块和自动增益控制模块等。自动增益控制模块能稳定信号传输电平，通过信号增益控制的处理，能根据输入信号电平大小和指定的输出电平，自动调整电平变化，并且不影响传输信号尤其是语音的质量，保证经过控制后的语音可懂度不会发生波动。

本论文的主要研究内容是语音信号预处理系统中的自动增益控制模块。重点学习基于 VAD（Voice Activity Detection）的 AGC 算法和基于能量比较的 AGC 算法，主要工作是将 Speex 语音编码算法中用到的自动增益控制算法在 DSP 芯片上实现，应用于 VoIP 系统的话机终端。

在本课题中，采用 24 位的 AR1688 芯片，综合实际应用的优势，定点 DSP 芯片功耗小，价格低，运算时间比浮点 DSP 芯片短，更适用于实时语音传输应用及大规模生产，因此本课题选择定点 DSP 实现。课题的实现过程分为两步，首先对算法进行了定点 C 语言的实现，然后把定点代码转化为 DSP 汇编代码。在 AGC 算法通过调试后，为了降低运算复杂度，适应低速处理器的要求，利用了 AR1688 芯片自身的硬件特点和指令特点，对算法进行了优化。最终在以 AR1688 为核心的设备上成功地实现了自动增益控制的要求。

关键词：语音预处理 自动增益控制 AGC 算法 AR1688 芯片 DSP 实现

RESEARCH AND IMPLEMENTATION OF AUTOMATIC GAIN CONTROL ALGORITHM IN VoIP

ABSTRACT

With the rapid development of Internet, Internet voice communications applications become increasingly popular.Comparing with traditional telephone, IP telephone has been widely used because of its advantages of high utilization ratio of network bandwidth, low talking cost and multiple bearing services. IP telephone network is different from the traditional one for its packing the voice into IP packets and sending them to the end through IP address. Thus, it causes some problems such as time-delay, jitter and echo. To eliminate these influences, the speech signals should be preprocessed, and it can greatly improve the system performance under the condition of outside environment interference and the speech quality.

Practical voice preprocess system mainly includes noise reduction system, echo control system, voice activity detection module and automatic gain control module and so on. The automatic gain control module can make signal transmission level stable.Through the process of signal gain control,it can adjust level variation according to input signal level and specified output level,and can make no influence to transmission signal, and make sure that the intelligibility of controlled speech will not fluctuate.

The main research content of this thesis is automatic gain control module in the voice preprocess system .The focuses are VAD-based AGC algorithm and energy comparison-based AGC algorithm, and the main work is to implement the AGC algorithm used in Speex speech code algorithm on the DSP chip, and then apply it to the telephone terminal in VoIP system.

In this thesis, the 24-bit AR1688 chip is used. The fixed-point digital signal processor chip has the advantages of practical application because of the little power consumption, inexpensive price and shorter operation time than the float-point one, and it is more suitable for realtime voice transmission application and large scale production, so the fixed-point DSP chip is adopted. The implementation in this thesis is in two steps, first make the floating point to fixed-point conversion in C with optimization. Then translate the C to DSP assembly codes. After the AGC algorithm passing the testing sequence, the next work is the optimization of the AGC algorithm using the hardware features and instruction characteristics of AR1688 chip, in order to reduce the complexity of algorithm and adapt the requirements of processor. Eventually, we successfully achieved the requirements of automatic gain control on equipment of which the core element is AR1688 chip.

KEY WORDS: voice preprocess automatic gain control AGC algorithm AR1688 chip DSP implementation

目 录

第一章 引 言

互联网的出现给人们的工作和生活带来了巨大的影响和改变，通过网络人们可以更方便地了解外界的信息，更快捷地完成各种业务。随着互联网与其它领域的结合，也产生了各种新的应用，这将进一步影响和改变人们的生活方式。

VoIP 技术正是网络技术与多媒体技术结合产生的，基本目标是通过分组交换网络来传输语音数据。由于与计算机技术的结合，VoIP 可以提供比传统的电话网络更多、更好的服务，如统一消息、查号业务、虚拟语音/传真邮箱等。本章将从 VoIP 提出的背景、系统组成及原理、VoIP 系统中的语音预处理模块等几个方面介绍 VoIP 系统，从而引出本文的研究课题。

1.1 VoIP 提出的背景

1995 年，全球第一个可以通过 Internet 打长途电话的软件产品“Internet Protocol”由以色列 Vocal Tec 公司率先推出。IP 电话的出现不仅大幅度减少了用户的通讯费用，而且提供了一个全新的通讯方式，迅速成为全球热门的新业务，也是当今世界上发展最快、普及最快的应用服务技术之一[1]。

VoIP 是 Voice over IP 的缩写，即把语音技术集成在 IP 传输协议中，通过 Internet 网络进行语音传输的一种全新的通讯方式。随着 Internet 在全球范围内的兴起和语音编码技术的发展，VoIP 获得了突破性的进展和实际的应用，而且正在逐步占领传统电话业务的市场。

IP 电话之所以迅速如此发展，是因为它比传统电话具有一定的优势：

（1） IP 网传输交换。IP 电话的传输媒介是 Internet 网络，IP 电话在传输过程中，信息根据 IP 协议被分成组进行传输，每个分组上都有目的地址与分组序号，分组可以沿不同的路径到达目的地，在目的地将分组重新组成原来的信号，可以实现信道的统计复用，提高信道利用率。而传统电话采用电路交换方式，信道利用率低。

（2） 语音压缩技术成熟。传统电话一般采用 64kbit/s 的语音编码速率，而 IP 电话使用的压缩技术可以将语音信息压缩到 10Kbit/s 以下，占用带宽达到传统带宽的 1/8。

（3） IP 电话占用带宽低，利用率高。传统电话给每一个成功的呼叫都提供 64kbit/s 的固定信道，只要不挂机，即使没有人说话、没有语音信息传输的情况下，这一信道始终不能被别的呼叫使用。IP 电话的语音信息不占用固定的信道，

只在有信息的时候才进行传送，而且采用语音压缩技术，大大提高了带宽的利用率。

（4） 语音通信费用低廉。由于 IP 电话占用的网络资源比传统电话低，成本也较低，所以 IP 电话资费比传统电话低。

（5） 终端号码具有可携带性。传统电话有固定的地点、固定的号码，而 IP 电话没有固定的地点及号码，只要可以接入 Internet 网络就可以使用号码实现语音通信。

1.2 VoIP 技术简介

VoIP 技术通过语音压缩算法对语音信号进行压缩编码处理，然后把这些语音数据按照无连接的 UDP 协议（用户数据报协议）标准进行打包，经过网络把数据包发送到接收端；接收端将这些数据包经过解码、解压缩处理后恢复成原来的语音信号，从而达到由互联网传送语音的目的[2]。目前，VoIP 的实现结构主要有四种形式，包括电话到电话，电话到 IP 终端，IP 终端到电话，IP 终端到 IP 终端，其中 IP 终端既可以是 PC 机，也可以是专门的 IP 电话。

1.2.1 VoIP 系统的组成

目前国内可以支持 VoIP 业务开展的主要有 3 种实现形式，即基于 H.323 协议的 IP 电话网络、基于会话启动协议（SIP 协议）的 IP 电话网络和基于软交换的网络。

（1） 基于 H.323 协议的 IP 电话网络

H.323 协议的 IP 电话网络由网守、网关、应用服务器和后台管理等主要模块组成，其中网守负责用户呼叫的地址解析和资源的管理，网关负责不同通信网络间媒体流的转换和呼叫通道的建立，应用服务器负责在基本的语音业务的基础上扩展增值业务或进行呼叫策略的管理，后台管理设备则负责计费、认证、网管等功能。

（2） 基于 SIP 协议的 VoIP 网络

SIP 协议最早是由 IETF 的 Iptel 工作组提出的一种 VoIP 实现方式，其基本思想是在互联网环境中，组建一个平面结构的、可用于点到点对话需求的系统。

基于 SIP 协议所构建的系统具有协议结构简单、设备易于开发等特点，也能很好地承载语音和图像等多种业务。该系统是由 SIP 终端（客户机）、代理服务器和重定向服务器等功能模块组成。SIP 终端负责发出呼叫、媒体流的编解码；代理服务器负责接受终端的呼叫请求信息，并根据重定向服务器给出的地址信息将呼叫请求消息转发给下个代理服务器，直到送至最终的用户终端；而重定向服

务器用于给代理服务器指出转发消息下一次应该送达的代理服务器的地址。

（3） 用软交换设备实现的 VoIP 网络

软交换一词最早见于综合交换机的研究中，是 NGN 的核心设备，国内最早用于商用 VoIP 业务的软交换设备采用的是 SIP 协议，其只负责地址解析而不管理呼叫状态。而随着发展，软交换也逐渐吸收了许多其他 VoIP 技术的内容，将网关分解协议中的媒体网关控制器、H.323 协议中网守以及其他设备的功能逐步地融合进来，形成了目前的软交换设备。

1.2.2 VoIP 的基本传输过程

传统的电话网是以电路交换方式传输语音，所要求的传输带宽为 64kbit/s。而 VoIP 是以 IP 分组交换网络为传输平台，对模拟的语音信号进行压缩、打包等一系列的特殊处理，使之可以采用无连接的 UDP 协议进行传输。为了在一个 IP 网络上传送语音信号，要求几个元素和功能。最简单形式的网络由两个或多个具有 VoIP 功能的设备组成，这些设备通过一个 IP 网络连接。VoIP 模型的基本结构如图 1-1 所示。

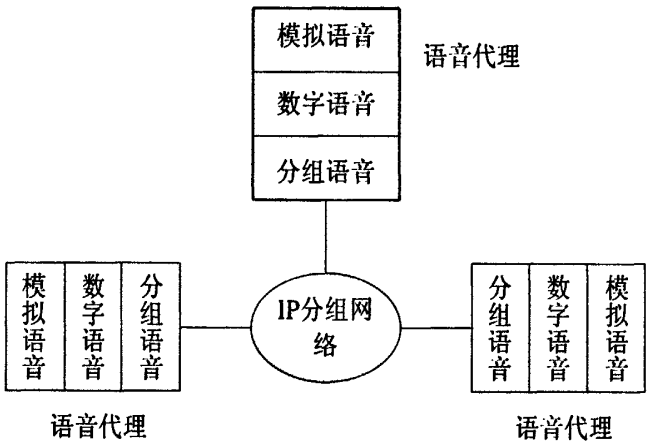


图 1-1 VoIP 的模型结构

从图可以发现 VoIP 设备是如何把语音信号转换成 IP 数据流，并把这些数据流转发到 IP 目的地，IP 目的地又把它们转换回到语音信号。两者之间的网络必须支持 IP 传输，且可以是 IP 路由器和网络链路的任意组合。

简而言之，语音信号在 IP 网络上的传送要经过从模拟信号到数字信号的转换、数字语音封装成 IP 分组、IP 分组通过网络的传送、IP 分组的解包和数字语音还原到模拟信号等过程[3]。整个过程如图 1-2 所示。

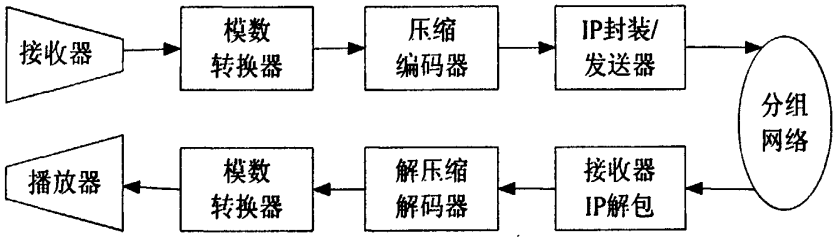


图 1-2 VoIP 传输的基本过程

1.2.3 VoIP 系统中需要解决的技术问题

多种原因促使 IP 电话（以及在其他数据网上传送语音）的产生和发展，然而大规模使用 IP 电话并不是一件容易的事。主要原因是 IP 协议族并不是为像语音这样的同步、实时的业务所设计的，因此在实现 VoIP 系统时，存在一些需要解决的问题。

（1） QoS 问题

由于分组网络尽力而为的传输特性，则影响 VoIP 系统服务质量最主要的因素是丢包和网络时延抖动。由于语音通信的实时性要求，在现有的情况下必须保证最低标准的通话质量的时延应小于 250 毫秒，其传输过程中的丢包率应小于 5%，否则语音质量无法接受。另外，影响 VoIP 通信质量的还有回音处理，静噪抑制，IP 包分割，语音数据优先级和前向纠错等因素。

（2） VoIP 与 PSTN，IN（智能网）的无缝连接

IP 电话不仅要与 PSTN 之间完成基本的通话服务，而且对于各种智能业务也要求做到互通。但是由于目前 IP 电话的信令标准化程度还有待进一步的完善，卡用户的漫游还没有得到很好的解决。

（3） VoIP 安全问题

目前，VoIP 面临的安全问题主要有四个方面：拒绝服务（DoS）攻击、非法接入、话费诈欺或窃听等威胁。信息安全专家警告，如果对 VoIP 部署不当，VoIP 会受到黑客和恶意代码的攻击，从而可能破坏网络的安全措施。对于企业网络而言，VoIP 的威胁则更大，因为企业会急于部署这一技术而忽视了安全。

1.3 VoIP 中的语音信号预处理

1.3.1 语音信号概述

语音是人们讲话时发出的话语，它既是一种声音，又包含人们进行交流的信息。因此，语音是语言和声音的组合体[4]，目前语音信号处理的内容主要集中在声音的处理上。

根据声音所在的频率范围，可以将声音分类为：

（1） 亚音(subsonic)：频率低于 20 Hz 的信号；

（2） 音频(Audio)：频率范围为 20Hz~20kHz 的信号；

（3） 超音频(ultrasonic)：频率高于 20 kHz 的信号。

人类能够听到的声音是音频（强度为-5~130dB），但能够发出的声音却低于 4kHz。根据香农采样定理[5]，数字语音信号采样率应该是信号最高频率的两倍，即 8kHz。我们通常所说的语音信号处理，实质就是对每秒至少 8000 个声音数据进行分析与变换。

完整的语音信号的数学模型可用三个子模型：激励模型、声道模型和辐射模型的串联来表示[6]，如图1-3所示。

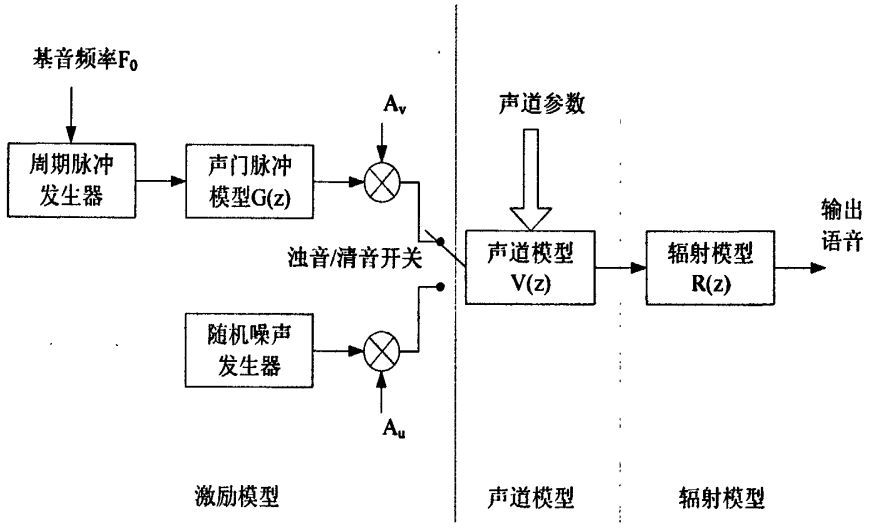


图 1-3 语音信号产生的离散时域模型

其传输函数可以表示为：

$$H(z)=A\cdot U(z)\cdot V(z)\cdot R(z) \tag{1-1}$$

其中，$U(z)$是激励信号，浊音时$U(z)$是声门脉冲即斜三角脉冲序列的z变换；清音时，$U(z)$是一个随机噪声的z变换。$V(z)$是声道传输函数，既可以用声管模型，也可以用共振峰模型来描述，但本质就是全极点模型[7]：

$$V(z)=\frac{1}{1-\sum_{k=1}^{N}a_k z^{-k}} \tag{1-2}$$

而辐射函数$R(z)$则可以近似为：

$$R(z)=R_0(1-z^{-1}) \tag{1-3}$$

需要指出的是，这些模型都是“短时”的，因为一些语音信号的变化是较缓慢的，例如元音在10~20ms内其参数可以假设不变。另外，模型中用浊音和清音

这样简单的划分方法是有缺陷的，这对于某些摩擦音是不适用的，而且也无法用叠加的方式得到，对于这些音可以用一些修正模型来模拟。

1.3.2 语音分析方法

语音信号分析是语音信号处理的前提和基础，只有分析出可以表示语音信号本质特性的参数，才有可能利用这些参数进行高效的语音通信、语音合成和语音识别等处理。根据所分析参数不同的性质，可将语音信号分析分为时域分析、频域分析、同态分析、线性预测分析等方法[8]。

（1） 时域分析方法

语音信号的时域分析就是分析和提取语音信号的时域参数，这是一种比较直观的分析方法。时域分析通常用于最基本的参数分析及应用，如语音的分割、预处理与分类等，实现简单，运算量也较小。

语音信号的时域参数包括有短时能量、短时过零率、短时自相关以及短时平均幅度差等，这是语音信号的一组最基本的短时参数，在各种语音信号数字处理中都要应用到。为了使语音信号的短时能量与幅度的变化相对平滑，在计算这些参数时使用的一般都是矩形窗或汉明窗。

（2） 频域分析方法

从广义上来讲，语音信号的频域分析包括语音信号的频谱、功率谱、倒频谱、频谱包络分析等。常用的频域分析方法包括傅立叶变换法等等。因为语音信号是一个非平稳过程，因此适用于周期、非瞬变或平稳随机信号的标准傅立叶变换不能用来直接进行分析，而应该用短时傅立叶变换进行频谱的分析，相应的频谱称为“短时谱”。

对第 n 帧语音信号 $x_n(m)$ 进行傅立叶变换，定义如下：

$$X_n(e^{jw}) = \sum_{m=0}^{N-1} x_n(m) e^{-jwm} \qquad (1\text{-}4)$$

其中 N 为变换点数，短时傅立叶变换实际上就是窗选信号的标准傅立叶变换。选取不同的窗函数，就会得到不同的傅立叶变换结果。

如令角频率 $w = 2\pi k / N$，则可得到离散的短时傅立叶变换 $X_n(k)$。在语音信号数字处理中，一般采用 $x_n(m)$ 的离散傅立叶变换来替代 $X_n(e^{jw})$，并且可以用高效的快速傅立叶变换算法完成出 $x_n(m)$ 到 $X_n(k)$ 的转换。为了符合人耳的听觉特性，提高语音信号处理系统的性能，还可以进一步将实际的线性频谱转化为临界带频谱矢量，从而可根据人耳对频率高低的非线性心理感受反映语音短时幅度谱的特征。

（3） 同态分析

同态分析实现将卷积关系变换为求和关系的分离处理，即解卷。对语音信号进行解卷，可将语音信号的声门激励信息及声道响应信息分离开来，从而求得声道共振特征和基音周期，可用于语音编码、合成与识别等。

许多语音信号并不是加性信号，而是声门激励和声道冲击响应的卷积信号，而同态信号处理可以将这类非线性问题转化为线性问题，在线性空间完成运算后再逆变换为卷积信号。

对卷积信号 $x(n)=x_1(n)*x_2(n)$ 进行如下的运算处理：

$$\begin{cases} Z[x(n)]=X(z)=X_1(z)\cdot X_2(z) \\ \ln X(z)=\ln X_1(z)+\ln X_2(z)=\hat{X}_1(z)+\hat{X}_2(z)=\hat{X}(z) \\ Z^{-1}[\hat{X}(z)]=Z^{-1}[\hat{X}_1(z)+\hat{X}_2(z)]=\hat{x}_1(n)+\hat{x}_2(n)=\hat{x}(n) \end{cases} \quad (1\text{-}5)$$

由于 $\hat{x}(n)$ 是加性信号，所以可对其进行线性处理。例如在两个信号互不交替的情况下，将声门激励信号与声道冲击响应分离开来，最后只需要对分离信号进行逆变换与指数运算即可恢复出原来的卷积信号。

（4） 线性预测分析

线性预测分析的基本思想是：利用语音样点之间存在的相关性，可以用过去的样点值来预测现在或者未来的样点值，即一个语音的抽样能够用过去若干个语音抽样或者它们的线性组合来进行逼近，然后通过使实际的语音抽样和线性预测抽样之间的误差在某个准则下达到最小值来决定唯一的一组预测系数。现代语音编码的声道模型的参数估计大多都是基于线性预测分析的方法。

1.3.3 语音信号预处理

在数字语音通信中，背景噪声的干扰、信号传输的损耗以及语音信号的正反馈所引起的信号不稳定，使得很多语音处理系统的性能急剧下降。例如语音编解码系统中，信道噪声与线路噪声污染的影响是很大的，又例如线路电平的不匹配造成音量大小不一等。为了消除现实环境中的数字语音对人们的主观听觉所造成的负面影响，对语音预处理技术及其实用化的研究是非常有必要的。语音预处理技术是数字语音信号处理的重要分支，已经广泛应用于无线电话、电话会议和场景录音等领域。通过各个方面的预处理可以大大改善原系统在外界环境的干扰条件下的性能，从而提高语音通信质量。

在电话网中，为了把模拟的话音转换为数字信号进行传输需要采用一定的信号转换技术，具体过程如图1-4所示。话音在话机中由声音信号转换为电信号后，在数字电话交换机中先是对信号进行了采样，得到一系列的离散信号，然后对这些离散信号进行量化，得到相应的量化值，最后把这些值进行编码。编码后的数字信号就可以在数字通信网中传送。在信号的接收端，数字信号被译码为一系列

的值，进而通过类似低通滤波器的电路恢复原有的模拟话音信号，送到被叫用户的电话设备上。

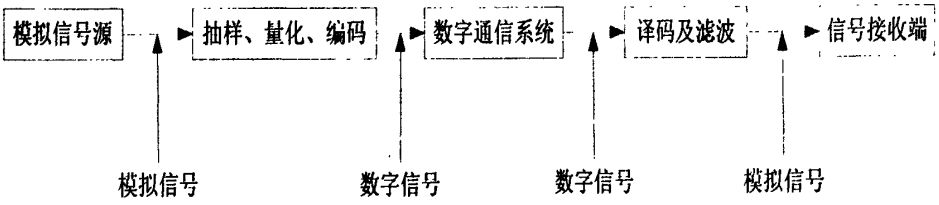


图1-4 语音信号在数字通信系统上的传输

IP电话与传统电话网相比，优势在于话音编码技术，通过一定的话音编码技术，IP电话能在一个2M带宽的信道中传输比传统电话技术多几倍的话音，由此大大节约了每路电话所需的传输带宽。在长途电话业务中能减少很多的长途传输设备，优势尤其明显。

在IP电话网中，话音不像传统电话网中占用一个固定的时隙，而是把所有的话音打包成IP包通过IP寻址的方式传送到对端，因此就产生了一些问题，包括延时、抖动、回声等现象。为了消除这些影响，需要对语音信号进行预处理。

语音预处理的目的是为了在保持语音可懂度和清晰度的前提下，对语音信号进行时域或频域的变换与处理，从而使语音在音强、音长、音质与纯净度等方面得到一定程度的提升。实用语音预处理系统主要包括降噪系统、回声控制系统、语音激活检测模块和自动增益控制模块等，如图 1-5 所示。

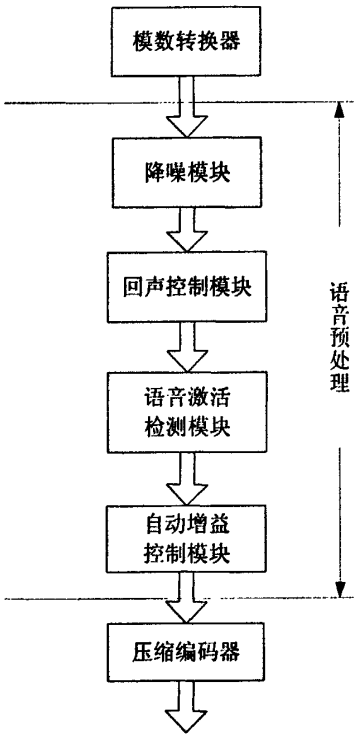


图1-5 语音预处理系统结构

其中，降噪模块的作用是检测并且抑制语音信号中的背景噪声，提高语音的纯净度；长话通信中回声影响较大，因此需要配备回声控制模块，针对扩声系统中回声所引起的正反馈放大现象，采用自适应滤波等方法进行回声对消。发送线路上的信号经过降噪后再进行回声消除处理，可减少噪声和回声对系统的干扰。

语音激活检测的目的是检测语音通信时是否有话音存在，在两个用户通话时，从统计上来说，用户发声的时间只占总时间的一半，用户在不说话时如果不进行话音处理，不传送语音包，可以大量节约网络的带宽。同时，在接收端，如果系统发送给最终用户的信号是什么也没有，用户会误以为信号中断了，所以一般的IP电话系统中如果处于静音状态，接收端就要向用户发送一个音量很小的噪声，使用户感觉到系统仍然在工作，这种噪声叫做舒适噪声。

自动增益控制模块能够稳定信号传输电平，使双端或多端语音的音强与音质维持在一定的水平上。通过信号增益控制的处理，能够根据输入信号的电平大小和指定的输出电平来自动调整电平变化，并且不影响传输信号尤其是语音的质量，保证经过处理后的语音可懂度不会发生波动。

1.4 工作流程与论文结构

本论文的主要研究内容是语音信号预处理系统中的自动增益控制模块。重点研究基于 VAD（Voice Activity Detection）的 AGC 算法和基于能量比较的 AGC 算法，主要工作是将 Speex 语音编码算法中用到的自动增益控制算法在 DSP 芯片上实现，应用于 VoIP 系统的话机终端。这两种 AGC 算法，其制定者只给出了基于 C 语言的浮点算法参考代码，因此在课题研究与实现的过程中，采用以下两个步骤完成工作：

1. 把给出的浮点 C 语言参考代码转化为定点 C 语言代码。一般情况下，把浮点运算转换为定点运算会带来一定程度的精度的损失，因此需要采用合理的定点化方法在保证数据动态范围的情况下把精度损失控制在一定的范围内。在对 AGC 算法进行定点化的同时，需要进行一些优化的工作，以减小算法的计算量，主要包括以下两个方面的优化：

（1） 算法级的优化：用更优的算法来取代参考代码中的实现方式，从而降低算法的时间复杂度和空间复杂度，改进算法的运算效率；

（2） C 语言级的优化：对定点化后的代码进行优化，提高代码的执行效率。

2. 把相应的定点 C 算法进行 DSP 环境的搬移，即将 C 语言代码改写成 DSP 代码，然后对计算量较大的模块进行汇编语言级的优化，尽可能地降低代码的运算复杂度，充分利用 DSP 的特殊运算单元和运算指令以及并行计算的能力，对汇编代码进行优化。

图 1-6 是定点化过程的流程图：

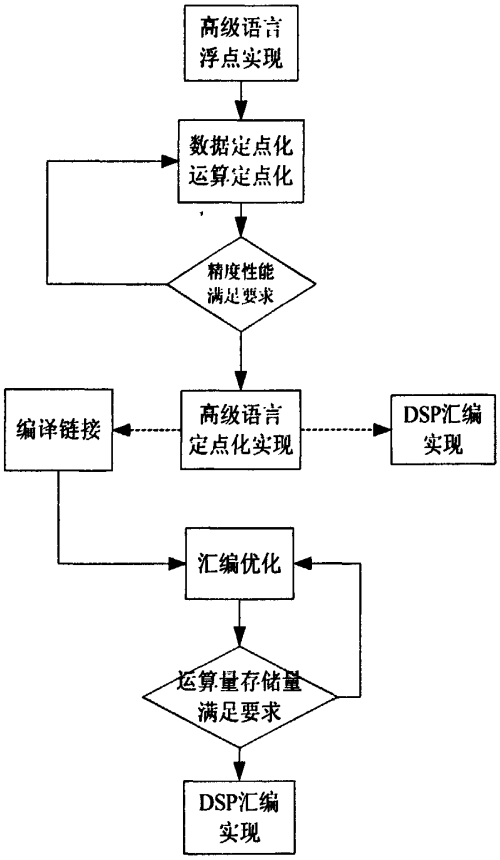


图 1-6 定点化过程的流程图

基于上面的流程，本论文接下来的各章安排如下：第二章简要介绍几种自动增益控制算法原理，并对本课题中用到的两种 AGC 算法进行详细的分析，第三章介绍定点理论、定点运算和基本函数库的实现，第四章主要介绍开发平台及 DSP 芯片的结构，第五章包含了自动增益控制算法的 DSP 实现及优化，并给出了仿真的结果及调试情况，第六章则对整个论文工作进行总结。

第二章　算法原理

2.1　自动增益控制方法的比较

自动增益控制技术在数字通信、语音处理、测试设备等方面的应用是十分广泛的[9]。通信中传输信号的幅度经常发生较大波动，通过 AGC 可以优化信号电平，从而提高通信质量。传统的自动增益控制都是用模拟电路实现[10]，其性能很大程度上受电路本身如响应时间、动态范围等的限制[11]。在 AGC 系统中使用数字信号处理方法[12],[13]，可避免控制电路的影响，设计灵活、精度高、控制范围大[14]，更有效地提高自动增益控制的性能。

自动增益控制是语音信号预处理系统的核心部分，下面是几种假设。

首先，通过选取较长时间的语音为一帧来将静音部分或噪音部分与语音部分的能量平均，这样可以减小每一帧与标准值的差值波动，达到稳定效果。然而这种做法，每帧要处理的数据太多，会有很大的延迟，不能很好地达到实时处理，而且对语音的估计也不准确，不能达到标准值的补偿效果。

其次，可以直接用能量进行判断，如果遇到静音帧或者噪音帧，能量就会很小，这时就跳过这一帧，不作为参数计算的帧，而对下一帧沿用上一次有效的调整参数。这样做，虽然计算简单，时延小，但也存在风险。如果噪音能量很大，则可能无法准确判断参数计算帧。

第三，采用 VAD 检测，先判断出语音帧，再根据语音帧之间的相关性，利用前一语音帧算得的调整参数帧，调整当前帧。如果判断出不是语音帧，则可使用上一次存储的参数调整，直到下一个有效的语音帧更新参数。这个方法很容易使调整参数变化太快而使语音信号失真。

接下来介绍几种常见的 AGC 算法以及本课题中所用到的两种 AGC 算法的原理，本课题中分别使用了一种采用 VAD 检测的 AGC 算法和一种直接用能量进行判断的 AGC 算法，通过进行 DSP 实现及仿真比较各自的性能。

2.2　几种常见的自动增益控制算法

2.2.1　音频 AGC 算法

音频 AGC 是音频自动增益控制算法，更为准确的说是峰值自动增益控制算法，是一种根据输入音频信号水平自动动态地调整增益的机制。当音量（无论是捕捉到的音量还是再现的音量）超过某一门限值，信号就会被限幅。限幅指的是

音频设备的输出不再随着输入而变化，输出实质上变成了最大音量位置上的一条水平线，当检测到音频增益达到了某一门限时，它会自动减小增益来避免限幅的发生，另一方面，如果捕捉到的音量太低时，系统将自动提高增益。当然，增益的调整不会使音量超过用户在调节向导中设置的值[15]。图 2-1 是音频 AGC 算法的结构框图。

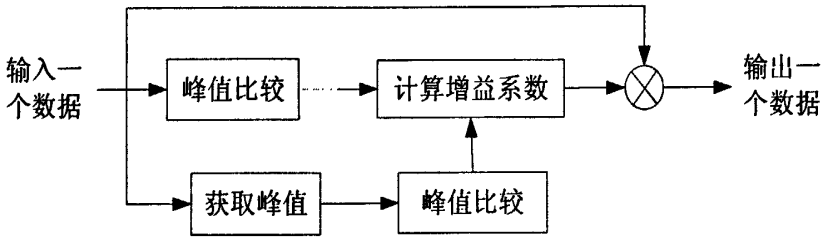


图 2-1 峰值 AGC 结构框图

该算法的程序流程图如图 2-2 所示。

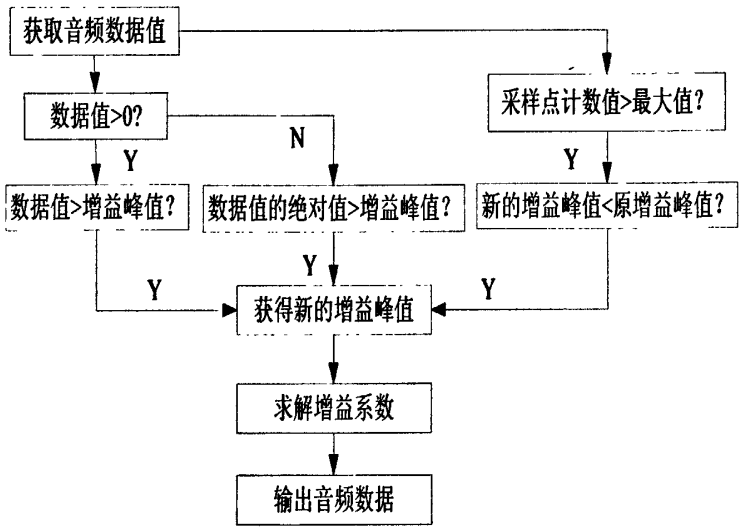


图 2-2 峰值 AGC 程序流程图

首先获取音频数据，一般来说，这些数都是比较小的，通过 AGC 算法将输入的音频数据投影在一个固定区间内，从而使得不论输入的数据点数值大小都会等比例地向这个空间映射。一方面将获得的音频数据最大值与原来的峰值进行比较，如果有新的峰值出现就计算新的增益系数；另一方面在一定的时间周期内获取一个新的峰值，这个峰值就具有检测性能，又与原峰值比较，然后就计算新的增益系数。这个增益系数是相对稳定的。当音量加大时，信号峰值会自动增加，从而增益系数自动下降；当音量减小时，新的峰值会减小并且取代原来的峰值，从而使峰值下降，使增益系数上升。最后输出的数据乘以新增益系数后映射到音频信号输入的投影区间内。

2.2.2 符合 G.169 协议的 AGC 算法

使用一种基于 RMS 功率比较的 AGC 算法，可以保证语音质量。

首先，AGC 对每帧语音作 RMS 功率估计[16]：

$$\bar{X}_k^2 = \frac{1}{N}\sum_{n=1}^{N}\left[x_k(n)\right]^2, k = 1,2,... \qquad (2\text{-}1)$$

N 是每帧语音的样点总数，$x_k(n)$是信号样点幅值，k 代表输入的帧序号。式(2-1)计算的单帧语音功率用于更新长时 RMS 功率估计值$S(k)$

$$S(k) = \alpha \cdot S(k-1) + (1-\alpha)\cdot \bar{X}_k^2, S(0) = 0 \qquad (2\text{-}2)$$

功率平滑因子$0 < \alpha < 1$，α 值越大，最近输入的一帧语音能量对$S(k)$的影响越小，此时$S(k)$变化趋于平缓，更利于反映较长时间内的信号能量均值。但α取值非常接近 1 时，$S(k)$的变化相当缓慢而无法体现语音的瞬时强弱变化，反而造成控制处理失真。

利用目标电平与功率估计均值的偏差，可计算 AGC 的增益因子

$$g(k) = \beta \cdot g(k-1) + (1-\beta)\cdot 10^{[T-dBS(k)]/20} \qquad (2\text{-}3)$$

其中，$g(k)$是当前一帧语音的增益。目标电平T以$dBov$为单位，$dBS(k) = 10\cdot \lg S(k)$，$T$与$dBS(k)$两者的差值以及增益平滑因子$0 < \beta < 1$都会影响增益的变化速度。式（2-3）计算得到的增益可能在某段时间内迅速变化，语音经过增益控制后容易产生失真。为此设定增益最大变化速率W使之满足$\left|20\lg\frac{g(k)}{g(k-1)}\right| \le W$，以保证 AGC 不影响语音的传输质量。

采用该算法进行 AGC 处理，可有效地根据输入信号幅度控制输出电平的变化，使输出电平控制在目标值附近，并且信号波形基本不变，各频率特性变化不大，保证语音失真小。此外还可对输出增益进行再次平滑，使 AGC 增益变化满足 G.169 的要求，实现 AGC 系统的功能。

上述是比较有效的 AGC 算法，但其控制能力和准确度还能再提高。例如在功率估计过程中，如果运行时间太长，估计的平均能量就不能准确反映较近时间内的信号。对此可经过一定时间，将该时间段内的信号能量取代总的估计能量，这样 AGC 能更有效地利用估计功率进行信号调整。

2.3 本课题所采用的自动增益控制算法

2.3.1 基于 VAD 检测的 AGC 算法

采用语音激活检测（Voiced Activity Detection, VAD）技术的目的是检测语音通信时是否有话音存在，检测到静音时加以抑制，使其不占用或极少占用信道带宽，检测到话音时才对其进行压缩编码与传输。

如图 2-3 所示，经过加窗和 FFT 变换的话音传入 VAD 和 AGC 模块，两者配合实现自动增益控制功能。其中 VAD 用于判断信号是否为纯语音段，其输出结果是布尔值，输出“1”表示当前信号帧是话音帧，输出“0”表示非话音帧。AGC 根据 VAD 的结果，仅对检测到的纯语音作自动增益控制，将其电平调整至理想值，而对非语音信号直接输出 0，不进行编码与发送，从而节省了带宽。

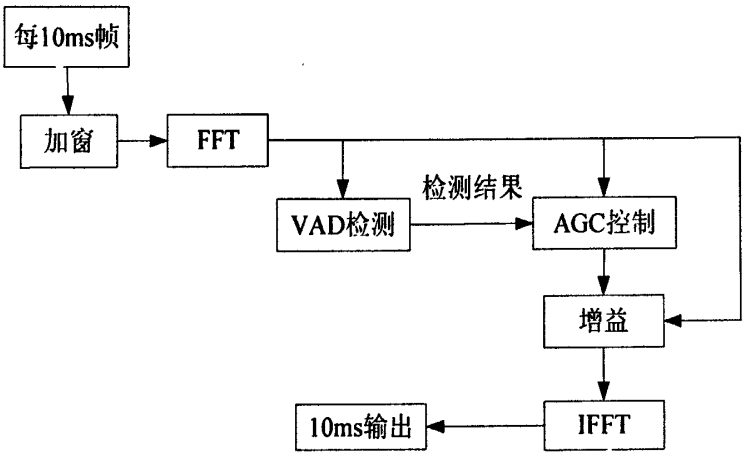


图2-3 基于VAD的AGC算法原理图

本课题采用的是一种基于能量比较的 AGC 算法，在保证语音质量的同时实现增益控制。

VAD 需要对语音进行分帧检测，AGC 可利用该条件对每帧语音各样点作能量估计：

$$ps_k(i) = |x_k(i)|^2, i = 1,2,...,N \qquad (2\text{-}4)$$

其中 N 是每帧语音的样点数，$x_k(i)$ 是信号样点幅值，是输入语音信号经过加窗、快速傅立叶变换（FFT）后得到的带噪语音的幅值，k 代表输入帧序号。式（2-4）计算的单帧语音各样点能量用于更新长时能量估计值 $S_k(i)$：

$$S_k(i) = \alpha \cdot S_{k-1}(i) + (1-\alpha) \cdot ps_k(i),$$

$$S_0(i) = 0, i = 1,2,...,N \qquad (2\text{-}5)$$

其中 $S_{k-1}(i)$ 为第(k-1)帧中第 i 个样点的平滑能量值，能量平滑因子 $0 < \alpha < 1$，α

值越大，最近输入的一帧语音能量对 $S_k(i)$ 的影响越小，此时 $S_k(i)$ 变化趋于平缓，更利于反映较长时间内的信号能量均值。但 α 取值非常接近 1 时，$S_k(i)$ 的变化相当缓慢而无法体现语音的瞬时强弱变化，反而造成控制处理失真。本算法中取 α =0.8。

根据每一帧的 $S_k(i)$ 值，记录各样点长时能量的最小值 $S_{k\,min}(i)$，通过比较 $S_k(i)$)和 $S_{k\,min}(i)$ 的值来确定是否需要进行增益的调整，本算法中采用的判决条件是

$$S_k(i) > 20 \cdot S_{k\,min}(i) + 1000 \tag{2-6}$$

满足该条件时记录该样点，当该帧中满足该条件的样点数占总样点数的百分比大于一定的值时，进行增益调整，不满足时增益系数保持不变，即为前一帧的增益系数。

在进行增益调整时，设定两个调整平滑因子 *rate* 和 *rate*2：

$$loudness(k) = (1 - rate) \cdot loudness(k-1) + rate \cdot ps(k),$$

$$loudness(0) = 3.6 \times 10^7 \tag{2-7}$$

$$loudness2(k) = (1 - rate2) \cdot loudness2(k-1) + rate2 \cdot \left[loudness(k)\right]^{1/2},$$

$$loudness2(0) = 6000 \tag{2-8}$$

$$gain(k) = 8000 / loudness2(k) \tag{2-9}$$

其中 *rate* 初值为 1，随着处理帧的个数增加而逐渐减小；*rate*2 为 0.2；$ps(k)$ 表示第 k 帧语音能量和的加权值；$gain(k)$ 表示第 k 帧语音的增益系数。

计算出增益系数后，将当前帧各样点幅值与增益系数相乘，然后进行 IFFT（快速傅立叶逆变换）后输出，当 VAD 检测到是非语音帧时，由于增益系数为 0，则输出为 0。

另外，VAD 采用的是基于统计模型的话音激活检测算法（STAT-VAD）[17]。传统 VAD 算法都假设背景噪声是平稳的，这样无论是否有话音都可估计时变的噪声统计值，利用特定的判决准则比较当前帧信号的统计量与所估计的噪声统计量，由此可进行话音激活检测。Sohn 和 Sung 指出针对由最大似然准则估计出的未知参数进行似然比检测可得到该算法的判决准则[18]。鉴于实际背景噪声的非平稳性，该算法在此准则的基础上根据观测噪声的能量调整判决门限以达到在低信噪比下的鲁棒性，从而进一步优化判决准则。

首先假设 2 个事件：

H_0：无话音 即：$X = N$

H_1：有话音 即：$X = N + S$

S，N 和 X 分别表示话音、噪声和带噪语音，其第 k 个分量为 S_k，N_k 和 X_k，话音和噪声的频谱分量都服从高斯分布，且加性噪声与话音不相关，则带噪语音谱在 H_0 和 H_1 下的条件概率分别为

$$p(X_k \mid H_{0,k}) = \frac{1}{\pi\lambda_{N,k}} \exp\left(-\frac{|X_k|^2}{\lambda_{N,k}}\right) \tag{2-10}$$

$$p(X_k \mid H_{1,k}) = \frac{1}{\pi(\lambda_{N,k} + \lambda_{S,k})} \exp\left(-\frac{|X_k|^2}{\lambda_{N,k} + \lambda_{S,k}}\right) \tag{2-11}$$

$\lambda_{N,k}$ 和 $\lambda_{S,k}$ 分别表示加性噪声和话音谱的方差，第 k 个分量的似然比 Λ_k 定义为

$$\Lambda_k = \frac{p(X_k \mid H_{1,k})}{p(X_k \mid H_{0,k})} = \frac{1}{1+\varepsilon_k} \exp\left[\frac{(1+\gamma_k)\varepsilon_k}{1+\varepsilon_k}\right] \tag{2-12}$$

其中，先验信噪比：$\varepsilon_k = \frac{\lambda_{S,k}}{\lambda_{N,k}}$，后验信噪比：$\gamma_k = \frac{|X_k|^2}{\lambda_{N,k}}$。

噪声方差 $\lambda_{N,k}$ 可通过噪声自适应求得，而信号的方差未知，因此，第 n 帧的 ε_k 用直接判决方法的估计：

$$\overline{\varepsilon}_k^{(n)} = \alpha \cdot \frac{\left|\overline{S}_k^{(n-2)}\right|^2}{\lambda_{N,k}^{(n-1)}} + (1-\alpha) \cdot \max(\gamma_k^n, 0) \tag{2-13}$$

其中，α 为加权因子，通常取 0.98。$\hat{S}_k$ 为增强谱幅度，由短时幅度的最小均方误差计算得到。

存在话音的判决准则由贝叶斯准则建立，带噪语音中存在话音的概率为

$$p(H_{1,k} \mid X_k) = \frac{1}{\left[1 + \frac{p(H_{1,k})}{p(H_{0,k})}\right]\Lambda_k} \tag{2-14}$$

先验概率 $p(H_{0,k})$ 和 $p(H_{1,k})$ 由实验分析得到，将每帧中由式（2-11）求得的条件概率与通过实验选定的门限进行比较，如果条件概率大于门限，则该帧判断为有话音帧，否则为静音帧。为增强检测对噪声变化的鲁棒性，算法中如果判断该帧为话音帧，更新噪声谱。

带噪语音中不存在话音的概率为

$$p(H_{0,k}|X_k)=\frac{1}{\left[1+\frac{1-p(H_{0,k})}{p(H_{0,k})}\right]\Lambda_k} \tag{2-15}$$

用当前帧和前一帧的参数迭代更新噪声的方差：

$$\lambda_{N,k}^{(n)}=\eta\lambda_{N,k}^{(n-1)}+(1-\eta)E\left(\left|N_k^{(n)}\right|^2\middle|X_k^{(n)}\right) \tag{2-16}$$

其中η为平滑因子，能量谱的计算公式为

$$E\left(\left|N_k^{(n)}\right|^2\middle|X_k^{(n)}\right)=p(H_{0,k}|X_k)\left|X_k^{(n)}\right|^2+\left[1-p(H_{0,k}|X_k)\right]\lambda_{N,k}^{(n-1)} \tag{2-17}$$

2.3.2 改进的基于能量比较的 AGC 算法

对于直接用能量进行判断的 AGC 算法，当语音能量较大时，可以调节增益减小输出的语音幅值，当语音能量较小时，可以提高增益增大输出的语音幅值，但在实际的话机测试中，当双方通话过程中处于静音时，会有小噪音进入，则可能判断为语音引起误判，进而会提高增益，放大噪音。

因此，对该算法进行改进，引入一个门限值用来判断语音帧和非语音帧，从而减小误判的概率，防止小噪音被放大进而影响通话质量。该门限值是基于先验信噪比和后验信噪比计算出来的，具体步骤如下：

（1） 如上述 AGC 算法中提到的，先对每帧语音各样点作能量估计：

$$ps_k(i)=|x_k(i)|^2, i=1,2,...,N \tag{2-18}$$

同样的，N 是每帧语音的样点数，$x_k(i)$是信号样点幅值，是输入语音信号经过加窗、快速傅立叶变换（FFT）后得到的带噪语音的幅值，k 代表输入帧序号。

（2） 式（2-18）计算的单帧语音各样点能量用于对噪声能量$S_noise_k(i)$进行估计：

$$S\_noise_k(i)=(1-\beta)S\_noise_{k-1}(i)+\beta\cdot ps_k(i)$$

$$S\_noise_0(i)=0, i=1,2,...,N \tag{2-19}$$

其中$S_noise_{k-1}(i)$为第（k-1）个噪音帧中第 i 个样点的能量值，能量平滑因子$0<\beta<1$，且β的值随着处理帧个数的增大而逐渐减小。

（3） 后验信噪比$postSNR_k(i)$定义为：

$$postSNR_k(i)=\frac{ps_k(i)}{S\_noise_k(i)}-1 \tag{2-20}$$

即本帧中有用信号与噪声信号的能量比；而先验信噪比定义为：

$$priorSNR_k(i) = \gamma \cdot postSNR_k(i) + (1-\gamma) \cdot \frac{old\_ps_k(i)}{S\_noise_k(i)} \qquad (2\text{-}21)$$

其中$old_ps_k(i)$为前一帧中各样点的能量值，γ为更新系数，且$0<\gamma<1$。

（4） 求先验信噪比的递归平均值：

$$zframe_k = \frac{1}{N}\sum_{i=1}^{N} zeta_k(i) \qquad (2\text{-}22)$$

其中$zeta_k(i) = \alpha \cdot zeta_{k-1}(i) + (1-\alpha) \cdot priorSNR_k(i)$，$i = 1,2,...,N$，平滑因子$\alpha$取0.7。

（5） 门限值*pframe*则可求出：

$$pframe_k = \mu + (1-\mu) \cdot zframe_k \qquad (2\text{-}23)$$

$pframe_k$表示第 k 帧的判决门限值，因子μ值越小，本帧的先验信噪比对门限值的影响越大，更利于语音和噪声的判别，因此本算法中取μ=0.1。

利用该门限值，可以判别语音和非话音，当该值较大时，语音的概率较大，当连续几十帧都大于固定的值，则可初步判定为语音信号，否则判定为非话音信号。对语音信号进行相应的增益调节，而对于非话音，不进行增益调节，直接输出，这样可以防止小噪音信号被放大。

当门限值确定之后，设定当大于门限值时判定为话音帧，当连续输入 30~50 帧为话音帧时，则进入话音状态，该状态下需要进行增益系数的调节，改善输出话音信号电平；同样地，小于门限值时判定为非话音帧，当连续输入 30~50 帧为非话音帧时，则进入非话音状态，此状态下不进行增益调节，直接输出该帧信号，防止将小噪音放大。

而对于需要进行增益系数调节的话音帧来说，具体的增益系数的计算方法如下：

首先，对该帧各样点的能量求得能量和：

$$\text{loudness} = \sqrt{\sum_{i=1}^{N} [ps_k(i) \cdot loudness\_weight(i)]} \qquad (2\text{-}24)$$

其中，$loudness_weight(i)$为设定的各样点的权值，由此得到一个能量权值的乘积和，为防止能量过大，进行开方运算。

然后，对该能量和进行平滑，得到值$st_loudness_k$：

$$st\_loudness_k = (1-rate) \cdot st\_loudness_{k-1} + rate \cdot loudness \qquad (2\text{-}25)$$

*rate*为平滑因子，且随着$pframe_k$的值改变而改变，从而更有利于反映不同

帧的影响，通过该平滑因子得到另外一个值 $loudness\_accum_k$：

$$loudness\_accum_k = (1-rate)\cdot loudness\_accum_{k-1} + rate \qquad (2\text{-}26)$$

$rate$ 值越大，表明 $pframe_k$ 值越大，更能反映当前帧的影响，利用这些值可以得到增益系数

$$gain_k = \left[\frac{st\_loudness_k}{loudness\_accum_k}\right]^{-1/5} \qquad (2\text{-}27)$$

对于每帧不同的话音信号，由式（2-27）得到的增益系数可能偏大或偏小，因此当 $pframe_k$ 值大于一定的值或者与上一帧的增益系数相比本帧得到的增益系数过小时，需要对增益系数设定一个上限和下限。

在本算法中，具体的操作是：

（1） 当本帧增益系数与前一帧的增益系数相比大于一定的比值时，需要减小本帧的增益系数，因为此时 $pframe_k$ 是大于一定的值，是话音的可能性较大，如果增益过大，放大后的幅值则可能过大；

（2） 当本帧增益系数与前一帧的增益系数相比小于一定的比值，而且当前帧的各样点能量和与上一帧所得到的样点能量和相比很小时，则说明此时的话音信号幅度较小，可适当增大增益系数。

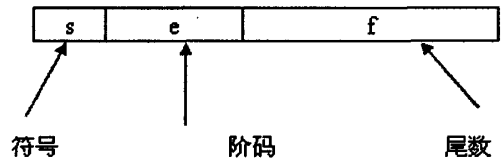
第三章　定点理论与运算的实现

由于定点 DSP 芯片的功耗小，价格低，运算时间比浮点 DSP 芯片短，且更适合用于实时语音传输应用以及投入大规模生产，实际工程应用中通常采用定点 DSP 芯片。因此，将 AGC 算法制定者给出的浮点 C 语言参考代码转换成定点 C 语言代码，是将该算法应用于工程实践的第一步。转换后的 C 代码再经过到 DSP 代码的翻译过程，完成大量的优化和测试工作后再写入芯片，完成最终的从参考代码到话机终端自动增益控制模块的实现。本章重点研究和探讨了定点理论及定点化方法，为下一步的定点化工作提供理论指导，并实现所用算法中常用的基本函数，完成初步的定点化工作。

3.1　定点化理论概述

现在的计算机中，一般采用浮点数表示实数，对于软件编程者来说，采用浮点数可以把数学公式中的实数直接用高级语言描写，编译器把这些数转化成计算机里用来存储的浮点数，IEEE 规定了单精度浮点数及双精度浮点数的存储格式。

浮点数表示的方法是类似于科学计数法的一种表示方法。一个浮点数被分为符号、阶码和尾数三个部分进行存储和运算，IEEE754 单精度数[19]表示方法如下：



$$V=(-1)^{s}\times 2^{e-127}\times 1.f$$

图 3-1 IEEE754 单精度浮点数的存储结构

例如：机器码为 0 10000001 00011000000000000000000

其表示的数为

V = (-1)0×2129-127×1. 00011000000000000000000

= 4.375（10）

然而浮点数在方便了程序编写的同时，却大大增加了机器的执行时间，对于相同的整数乘法，浮点运算的执行时间可能会是定点算法的十几倍，因而在对运算时间要求比较高的场合，例如在实时语音、视频的传输中，定点 DSP 由于速

度较快，有着广泛的应用。

3.1.1 定点数的 Q 值表示方法

在定点运算系统中，数据都是用二进制的补码数[19,20]进行存储的，数据按照二进制补码的运算法则进行运算。对于小数，我们是对这个二进制的补码数加入一个定标值来表示的，然而，这个定标值并不实际存储在存储单元中，而是完全根据程序员的理解来确定定标的位置，这个定标的位置我们可以用 Q 值来表示，例如 Q4 表数该数的最后四位表示小数。

一般地，如果 x 用$[(b_N b_{N-1} b_{N-2}..b_0)2c, Q]$来表示，其中下标 2c 表示格式为二进制补码，则$x=(-1)^{b_N}\cdot 2^{N-Q}+\sum_{k=0}^{N-1} b_k\cdot 2^{k-Q}$，比如当 N=15, Q=0 时，每一位的权值如下图 3-2(a)所示，简记为 Q0 格式；再比如当 N=15, Q=15 时，序列 b 中每一位的权值如图 3-2(b)所示，记为 Q15 格式。

-2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

(a) N=15，Q=0

-2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

(b) N=15，Q=15

图 3-2 定点数的 Q 值与权值的关系

需要说明的是，Q 值表示法也存在其它的的表达形式，例如在 ADI 公司的文档[21]中，用的是 M.N 的格式来表示定点数，其中 M 表示整数的位数，N 表示小数的位数，QN 与 M.N 表达的定点数是等价的。在下文的讨论中，主要使用 M.N 的表示法，目的是与 ADI 文档中的运算表示法一致。

有了 Q 值的概念，就可以方便地对实数变量进行定标，从而找到合适的表达法。具体方法是估计出该变量的最大值，然后找到大于或等于这个值的最小的 2 的幂的数，这样就可决定 Q 值表示法中整数的位数，从而也决定了该数的表达法。

为了更便于定标，一般利用 Q 值表示法的数据范围表[22]来指导定标工作。见表 3-1。

表 3-1 数据的 Q 值与其数值范围

Q 值	M.N 表示法	数值范围
Q15	1.15	-1=X=0.9999695
Q14	2.14	-2=X=1.9999390
Q13	3.13	-4=X=3.9998779
Q12	4.12	-8=X=7.9997559
Q11	5.11	-16=X=15.9995117
Q10	6.1	-32=X=31.9990234
Q9	7.9	-64=X=63.9980469
Q8	8.8	-128=X=127.9960938
Q7	9.7	-256=X=255.9921875
Q6	10.6	-512=X=511.9804375
Q5	11.5	-1024=X=1023.96875
Q4	12.4	-2048=X=2047.9375
Q3	13.3	-4096=X=4095.875
Q2	14.2	-8192=X=8191.75
Q1	15.1	-16384=X=16383.5
Q0	16.0	-32768=X=32767

如果数据的值超出了 Q0 值所能表示的范围，可以用整数位扩展的双精度数或者多精度数来表示。类似地，如果 Q15 不足满足变量的精度要求，则可以用小数位扩展的双精度数来表示。

3.1.2 定点数的运算法则

由于定点 DSP 是对定点补码数进行运算的，因而了解定点数的运算规则是非常重要的，下面就对定点运算的基本规则[20,23]进行介绍。

（1） 加减法运算规则：

只有定标值相同的数，加减法运算才有意义，否则，进行运算之前应该先进行定标值的转化。

两个定标值相同的数进行加减法运算，小数的位数不变。即

$$M.N+M.N \rightarrow (M+1).N \tag{3-1}$$

（2） 乘法运算规则：

两个定点数进行相乘，乘积的 Q 值为两数的 Q 值之和。

$$A.B * C.D \rightarrow (A+B).(C+D) \tag{3-2}$$

（3） 移位法则 A(数据放大或缩小，定标值不变)：

如果数 x 的定标值为 M.N，则：

x>>L 位后，x 缩小 2^L 倍，定标值不变；

x<<L 位后，x 放大 2^L 倍，定标值不变。

（4） 移位法则 B(数据大小不变， 定标值改变)：

如果数 x 的定标值为 M.N，则：

x>>L 运算后，定标值变为(M+L).(N-L)；

x<<L 运算后，定标值变为(M-L).(N+L)。

在这种情况下，要注意数值的取值范围，避免 M-L 小于零，出现溢出情况。

3.1.3 定点化的原则与方法

与浮点器件相比，定点器件具有运算部件简单、乘法运算速度快、功耗和成本低等优点，因而很多数字信号处理器都采用了定点结构。不过，定点数的动态范围和精度都没有浮点数好，所以在算法软件实现时就带来了一些困难，主要表现在两个方面：数据的定标和运算的定点实现。

在定点化过程中，必须始终考虑数据的两个指标，即数据的动态范围和数据的精度。数据的动态范围是指数据在数轴上的可能的取值范围，比如 $\cos(x)$ 的取值范围为[-1,1]，16 比特语音信号的取值范围为[-32768,32767]。数据的精度是指数据的真实值与实际参与运算的数据取值之间的误差大小，比如 π 在实际运算中常取为 3.1415926，精度为 10^{-7}。

在定点数的表示中，不同的 Q 值对应的数据的动态范围和数据的精度是不一样的。比如，Q0 格式数据值对应的动态范围为[-32768,32767]，精度为 1，而 Q15 格式数据值对应的动态范围为[-1,1]，精度则为 2^{-15}。Q 值越大，小数点后面的位数就越多，精度就越高。

数据定标的一个原则是：在满足动态范围的前提下，使数据的精度尽可能地高。根据数据定标的原则，对于一个给定的变量，我们只需找到满足动态范围且 Q 值最大的数（即小数位数多，整数部分相应少）。因而核心问题便是估计变量的绝对值的最大值，记这个最大值为 $|\max|$。通常我们采用两种方法来估计 $|\max|$，一种是理论分析法，另一种是统计分析法[24]。

1. 理论分析法

有些变量的动态范围通过理论分析是可以确定的。例如：

（1） 三角函数。$y=\sin(x)$ 或 $y=\cos(x)$，由三角函数知识可知，$|y|\le 1$。

（2） 汉明窗。$y(n)=0.54-0.46\cos n\pi_n(N-1),0\le n\le N-1$。

因为 $-1\le\cos[2\pi n/(N-1)]\le 1$，所以 $0.08\le y(n)\le 1.0$。

（3） FIR 卷积。$y(n)=\sum h(k)x(n-k)$，设 $\sum h(k)=1.0$，且 $x(n)$ 是模拟信号 12 位量化值，则有 $|x(n)|\le 2^{11}$，则 $|y(n)|\le 2^{11}$。

2. 统计分析法

对于理论上无法确定取值范围的变量，一般采用统计分析的方法确定其动态范围。所谓统计分析，就是用足够多的输入信号的样本值来确定程序中变量的动态取值范围，这里的输入信号一方面要有一定的数量，另一方面必须尽可能地涉及各种情况。例如，在语音信号分析中，统计分析时就必须采集足够多的语音信号样值，并且所采集的语音样值应尽可能地包含各种情况，如音量大小，声音的种类。只有这样，统计出来的结果才具有典型性。

当然，统计分析时毕竟不可能涉及所有可能发生的情况，因此对统计得出的结果在程序设计时可采取一些保护措施，比如适当牺牲一些精度，取比统计值稍大些的 Q 值，使用 DSP 芯片提供的溢出保护功能等。

3.2 定点化函数库的实现

由于最终需要在定点 DSP 上对算法进行实现，参考代码中的浮点运算必须用相应的定点函数来代替。AR1688 提供了一些常用的定点运算函数[21]，但还不够完善，在模块分析时可以得到要实现各模块必须用到的子函数信息，在进行定点化工作之前，应当把一些基本的运算，如双精度的乘法、求指数、求对数运算进行定点的实现。

表 3-2 列出了程序中用到的定点函数库的内容，其中用白底色表示的部分是已有的定点化函数，灰底色部分是自行编写的模块。

表 3-2 定点化函数库的实现

函数	返回值	功能说明
DIV32_16(Word32, Word16)	Word16	双单精度除法
DIV32 (Word32, Word32)	Word32	双精度除法
Inv_sqrt(Word32)	Word32	开平方并求倒数
MULT16_32_Q15(word16,Word32)	Word32	单双精度乘法
spx_fft(Word16, Word16)	Word16	FFT 运算
spx_ifft(Word16, Word16)	Word16	IFFT 运算
MULT16_16_P15(word16,Word16)	Word16	单精度乘法（带进位）
MULT16_16_Q15(word16,Word16)	Word16	单精度乘法
Pow2(Word16, Word16)	Word32	以 2 为底的指数函数
Log2(Word32, Word16, Word16)	void	以 2 为底的对数函数
Div_L(Word32, Word32)	Word32	双精度除法
Div_S(Word16, Word16)	Word16	单精度除法
Pshr32_7(Word32)	Word32	32 位右移 7 位
Pshr32_15(Word32)	Word32	32 位右移 15 位

Pshr32(Word32)	Word32	32 位右移位
PSHR16(Word16)	Word16	16 位右移位
FLOAT_ADD(Word32, Word32)	Word32	双精度加法
div48(Word32, Word32)	Word32	双精度除法
MULT16_48_Q15(word16,Word32)	Word32	单双精度乘法
powx_y(word16,word16)	Word32	求 0.2 次方

一些基本的运算，如单(双)精度加/减法，单(双)精度乘法，单精度除法，移位运算，归一化运算等，在芯片的说明手册中已有介绍，下面对扩展精度除法及一些常用的非线性运算函数的定点实现方法做一些说明。

3.2.1 扩展精度除法的实现

通常实现的除法除数与被除数都是 16bit 的定点数，结果也只能有 16bit 的精度。在 AGC 算法中，有些算法（如后验信噪比的运算）需要两个 32bit 的定点数相除，如果把这两个数的低 16 位舍去进行 16 位的除法运算，那么得到的精度较低，因此需要有精度较高的除法运算。

本文中实现了一种 32 位的定点除法运算，精度为 2^{-28}，可以较好地满足 AGC 算法中对除法运算的要求，文中称为扩展精度的除法。

假设定点除法的除数与被除数为 32 位的定点数 A 与 B，两者的定标值相同，并且有 $A \le B$ 且 $2^{30} \le B \le (2^{31}-1)$（实际中先对 B 进行归一化运算，再把 A 左移归一化时的移位数值，如果 $A > B$，则可以求 $(A/2)/B$，最后将结果左移一位），把 A，B 用 DPF(双精度)格式表示，记 A，B 的高 16 位为 A_H，B_H，低 16 位分别为 A_L，B_L，则高精度除法算法为：

a) 先对被除数 B 进行归一化运算，然后对除数 A 作相同位数的移位运算。

b) 把被除数 B 分为高低字节 B_H 和 B_L，用 $approx = (2^{14}/B_H)/2^{30}$ 得到商的近似值。

c) 计算 $T = approx(1 - B_L * approx)$ 求得 B_L 最后的商。

式（3-3）给出了只取高 16 位的算法和高精度算法精度的推导。

除法 1：

$$\frac{A}{B} = \frac{A_H \cdot 2^{16} + A_L}{B_H \cdot 2^{16} + B_L} = \frac{A_H + A_L/2^{16}}{B_H + B_L/2^{16}} = \frac{A_H}{B_H}\left(\frac{1 + \frac{A_L}{B_H}\cdot\frac{1}{2^{16}}}{1 + \frac{B_L}{B_H}\cdot\frac{1}{2^{16}}}\right) \qquad (3\text{-}3)$$

$$\underline{\text{泰勒级数}}\ \frac{A_H}{B_H}\left(1 + \frac{A_L}{B_H}\cdot\frac{1}{2^{16}}\right)\left(1 - \frac{B_L}{B_H}\cdot\frac{1}{2^{16}} - \left(\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}\right)^2 - \cdots\right)$$

若取$\frac{A_H}{B_H}$为商，则误差为

$$\frac{A}{B}-\frac{A_H}{B_H}\approx\frac{A_H}{B_H}\cdot\frac{A_L-B_L}{B_H}\cdot\frac{1}{2^{16}} \tag{3-4}$$

理论上，上式最大的取值为$1/2^{16}$，即精度为$1/2^{28}$。

除法 2：

$$\frac{A}{B}=A\cdot\frac{2^{30}}{B}\cdot\frac{1}{2^{30}} \tag{3-5}$$

注意到在定点算法中$1/2^{30}$可以通过数据的定标值的改变来实现，不需要进行实际的运算。

$$\frac{2^{30}}{B}\cdot\frac{1}{2^{30}}=\frac{2^{30}}{B_H\cdot 2^{16}+B_L}\cdot\frac{1}{2^{30}}=\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\cdot\left(\frac{1}{1+\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}}\right) \tag{3-6}$$

$$\underline{\underline{\text{泰勒级数}}}\ \frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\cdot\left(1-\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}-\left(\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}\right)^2-\cdots\right)$$

取$A\cdot\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\cdot\left(1-\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}\right)=A\cdot\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\cdot\left(1-B_L\cdot\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\right)$为商，则误差为

$$\frac{A}{B}-A\cdot\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\cdot\left(1-B_L\cdot\frac{2^{14}}{B_H}\cdot\frac{1}{2^{30}}\right)\approx\frac{A}{2^{30}}\frac{2^{14}}{B_H}\left(-\frac{B_L}{B_H}\cdot\frac{1}{2^{16}}\right)^2 \tag{3-7}$$

理论上，上式的最大值总小于$1/2^{28}$，可见精度值得到了很大的提高。

3.2.2 非线性函数的实现方法

在定点运算中，一些非线性运算的实现是很重要的，这些运算的精度及运算量对程序的时间复杂度有着很大的影响。

通常用三种方法实现非线性函数，包括级数展开法，查表法及混合法[24]。

（1） 级数展开法的优点是占用较少的存储空间，运算精度较高，缺点是每次运算都要进行一系列的乘累加的运算，适用于非线性运算较少的情形。

（2） 查表法的优点是速度较快，对每个自变量值，只要得到对应的索引就可以得到相应的函数值。其缺点在于当要求的精度较高时，内存中要保存较大的表。

查表法适用于周期函数或函数取值范围较小的情况。

（3） 混合法是对查表法的一种改进的方法，采用查表法加线性内插法计算。对于每一个自变量，混合法将它分为两部分处理，第一部分是可以确定函数值的大致范围，第二部分再进行线性内插，得到细节值。混合法有两个条件，第一是自变量的取值范围要求较小，第二是函数在该区间内是单调的。

以下将对 AGC 算法中用到的非线性函数的实现进行详细的描述。

1. 指数函数、对数函数的定点实现

在数字信号处理相关的算法中，指数函数和对数函数以及开方运算是经常使用的非线性函数。在具体实现之前，我们先利用数学公式把求任意底数的指数函数和对数函数的问题转化为以 2 为底的对数运算，在二进制数的计算机系统中，这样的转化可以简化问题。

根据对数的换底公式，可以得出：以任意数为底的对数运算可以转化为以 2 为底的对数运算，相应地，以任意数为底的幂运算可以转化为以 2 为底的幂运算。

$$a^z = 2^{\log_2 a^z} = 2^{z\log_2 a} \tag{3-8}$$

在具体实现时，如果我们已知 a，b 的值，就可以在编写 DSP 代码时先对 $\log_2 b$ 和 $\log_2 a$ 计算并存储起来，这样可以减少 DSP 的运算量。

进一步利用二进制运算的特性，可以进一步地简化运算。

对于 $\log_2 z$，$z \geq 1$(如果 z<1，可以求 $-\log_2(1/z)$)，可以得到 z 的具体数值，就可以利用归一化的运算将它改写为 $z = 2^k \cdot x$ 的形式，其中 $1 < x \leq 2$，从而 $\log_2 z = \log_2(2^k \cdot x) = k + \log_2 x$，于是我们只要计算 $\log_2 x$，$1 < x \leq 2$ 就可以了。

对于 2^z，$z > 0$ 的运算（如果 $z < 0$ 则可以求解 $1/2^{-z}$），如果 z 是整数，简单地利用移位就可以了，否则，令 $z = \lfloor z \rfloor + x$，其中 $\lfloor z \rfloor$ 是小于 z 最大整数，$0 \leq x \leq 1$，于是，$2^z = 2^{\lfloor z \rfloor} \cdot 2^x$，只要计算 2^x，$0 < x < 1$ 就可以了。

对于 $\log_2 x$，$1 < x \leq 2$ 及 2^x，$0 < x < 1$ 的计算，由于在给定区间内，两个函数均满足自变量范围较小、函数值单调上升的特性，因此可以采用混合法进行求解，下面以 $\log_2 x$，$1 < x \leq 2$ 为例进行说明。

实现步骤如下：

1） 建立函数表

将区间 [1,2] 分成 32 等分(根据实际需要决定等分份数)，等分点为 $\{x_i = 1 + i/32,\ 0 \leq i \leq 32\}$，共有 33 个点。计算 2^{x_i} 的数值，得到一个数据 yTable[33]，其中 yTable[i] = $\log_2 x_i$，以 2.14 的定标格式表示。

```
Word16 tablog[33] = {
0, 1455, 2866, 4236,  5568,  6863,  8124,  9352, 10549, 11716, 12855,
13967, 15054, 16117, 17156, 18172, 19167, 20142, 21097, 22033, 22951,
```

23852, 24735, 25603, 26455, 27291, 28113, 28922, 29716, 30497,

31266, 32023, 32767}

2) 函数求解

为了扩大函数自变量的范围，自变量用双精度的数输入，采用以下步骤来求解函数值。

a) 对输入数进行归一化得到L_x，根据归一化值求函数值的整数部分；

b) 提取L_x中的第 31-25 位，作为查表运算的部分，得到小数值的估计值；

c) 提取L_x中的第 24-10 位，作为线性内插的值，得到小数值的修正部分。

对上述算法进行了测试，与浮点算法计算值进行对比，可以得知算的精度是很高的。对于2^x，$0 < x < 1$可以用类似的算法，可得到精度较高的定点算法。

2. 开方函数及相关运算

在各种数字信号处理算法中，$\sqrt{z},(z>1)$以及$1/\sqrt{z},(z>1)$经常出现，对这两种运算的快速实现也是定点算法的要求之一。

由于$\sqrt{z}=z\left(1/\sqrt{z}\right)$，因而可以只实现$1/\sqrt{z}$的定点算法，$\sqrt{z}$可再进行一次乘法运算即可得到。

$1/\sqrt{z}$同样可以用混合法进行求解。

$$1/\sqrt{z}=1/\sqrt{2^{2k}x}=2^{-k}\left(1/\sqrt{x}\right),\quad \text{其中} 1<x<4 \tag{3-9}$$

建立表格，令

1) $x_i=1+3i/48,\quad 0\le i\le 48$，并令$Table[i]=1/\sqrt{x_i}$，得到函数表

2) 查表计算

对输入变量L_x进行归一化操作，得到指数 exponent；

a) 若 30-exponent 为偶数则L_x右移一位；

b) 指数部分变成 exponent = (30-exponent)/2 +1；

c) 根据L_x的第 31-25bit，求得索引 i，查表求得估计值；

d) 根据L_x的第 24-10bit，求得插值部分 a，线性内插，求得修正值；

e) 对修正值右移 exponent 位。

利用上述方法实现定点开方函数的运算。

第四章　开发平台及 DSP 芯片结构介绍

为了将自动增益控制算法投入到实际的 VoIP 应用中，需要将算法移植到 DSP 芯片上并完成实现及优化。在本课题中，我们根据实际需求选用了价格低廉的定点 DSP 芯片，因此需要首先对算法进行定点化的处理，然后转化为汇编程序，再进行仿真和下载调试。本课题中采用与 Analog Devices 公司（ADI）的定点数字信号处理器 ADSP2181 指令集兼容的 24 位 AR1688 芯片，它是 PalmMicro 公司为低成本 VoIP 市场量身定做的单芯片(SOC)方案，用于替代较早的 PA1688 芯片。和 PA1688 相比，AR1688 具有更强大的功能、更高的集成度、更低的功耗和更小的体积等优势，为用户实现性能更佳、语音质量更好、价格更低的网络电话终端提供了基本保证，将满足广大客户未来几年网络终端市场的需要。本章将详细阐述 PA1688 和 AR1688 芯片的硬件结构和软件设计方法。

4.1　PA1688 介绍

PA1688 是一款入门级的双处理器集成芯片，其片内集成了控制器（MCU）、信号处理器（DSP）以及必要的外围接口电路，提供单一芯片的网络电话解决方案，它的设计思路是产品容易开发、生产成本低廉，配套容易，是一款应用广泛的 SOC 网络电话芯片解决方案[25]。

4.1.1　PA1688 的芯片特点和硬件结构

PA1688 是一个双核处理器集成芯片，其片内集成 8 位控制器，16 位数字信号处理器以及其他必要的接口电路，如 RS-232、USB、SDRAM、AC97codec、SRAM 和 KeyPad，提供单一芯片的网络电话解决方案。芯片特点包括：160 pin PQFP/LQFP 封装，最高运行速度 50MHz，内置高速 8 位 8051/32 兼容内核，兼容 ADSP2181DSP 内核，预留 AC'97 2.1Codec 接口，图像 CMOS 传感器接口，键盘接口等，工作电压为 2.5V 和 3.3V。图 4-1 表示出了 PA1688 芯片硬件的整体结构。

PA1688 芯片的核心主要包括两部分：

（1） controller：增强的 Intel MCS8051 指令集兼容控制器，进行系统控制、处理系统接口等外围工作，以及各种协议处理（如 TCP/IP，H.323 等）。PA1688 芯片一条指令的运行时钟为 4~8 个时钟周期（平均 6 个时钟周期），其最高运行速度为 50MHz。

（2） DSP：ADSP2181 指令集兼容的数字信号处理器，主要包括语音、图像编

解码的运算，最高速度相当于 33MHz 标准的 2181。

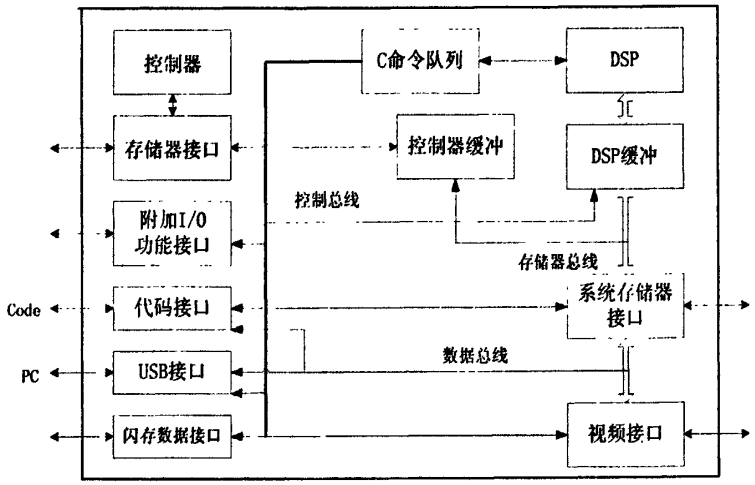


图 4-1　PA1688 芯片硬件整体结构

4.1.2　PA1688 芯片控制器系统结构

图 4-2 表示 PA1688 芯片控制器系统结构，该控制器系统主要由三部分构成：

1）控制器核心（Controller Core）：目前 PA1688 采用 8 位高性能处理器，进行系统控制和处理外围工作，其完全和 Intel MCS8051 兼容；

2）存储器接口；

3）静态存储器。

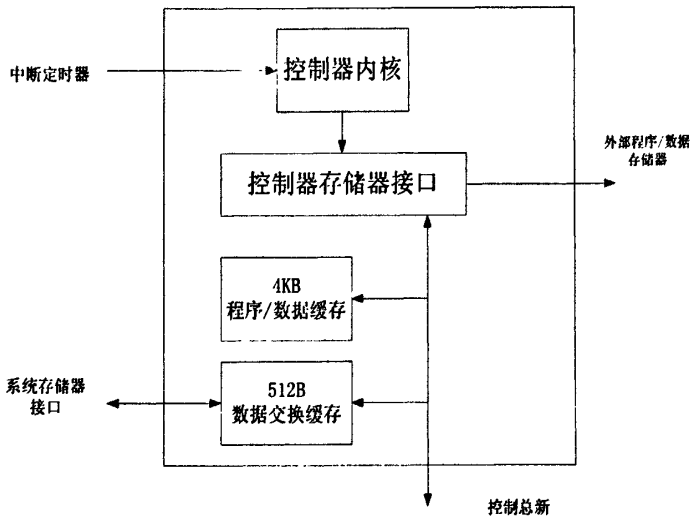


图 4-2 PA1688 芯片控制器系统结构

MCS8051 包括 8 位的数据总线、256 字节的内部数据存储器（DataRAM）和 64KB 的存储数据空间，其中 256 字节 DataRAM 为 Controller Core 使用。芯片使用的 64KB 的数据存储空间分成几部分，具体的空间分配如表 4-1 所示。

表 4-1 芯片使用的 64KB 数据存储器空间分配

地址范围	内容	说明
0x0000-0xFFFF	16KB of program upgrade memory	用于程序的升级
0x4000-0x5FFF	8KB of reserved memory	预留给仿真程序和连接外设用
0x6000-0x6FFF	4KB of controller memory	作为 8051MCU 的 xdata 使用
0x7000-0x71FF	512Byte controller exchange buffer	MCU 和 SDRAM 数据交换缓冲区
0x8000-0xBFFF	Registered memory space	寄存器存储空间
0xC000-0xFFFF	DSP Memory locations	DSP 的存储空间
0xC000-0xC7FF	PM Internal Buffer	DSP 代码空间的内部存储区
0xD000-0xDFFF	PM Exchange Buffer	DSP 代码空间的交换存储区
0xE000-0xE3FF	DMX Internal Buffer	DSP 数据空间的内部存储区
0xE800-0xEFFF	DMX Exchange Buffer	DSP 数据空间的交换存储区
0xF000-0xF7FF	DMY	

4.1.3 PA1688 DSP 子系统结构及外设

PA1688 集成的 DSP 是兼容 ADI 公司的一种低价格、高性能的 16 位定点运算 DSP——ADSP2181。它集成度高，主要包括语音、图像编解码的运算。DSP 的执行完全受 MCS8051 控制器的控制。

DSP 子系统结构如图 4-3 所示：

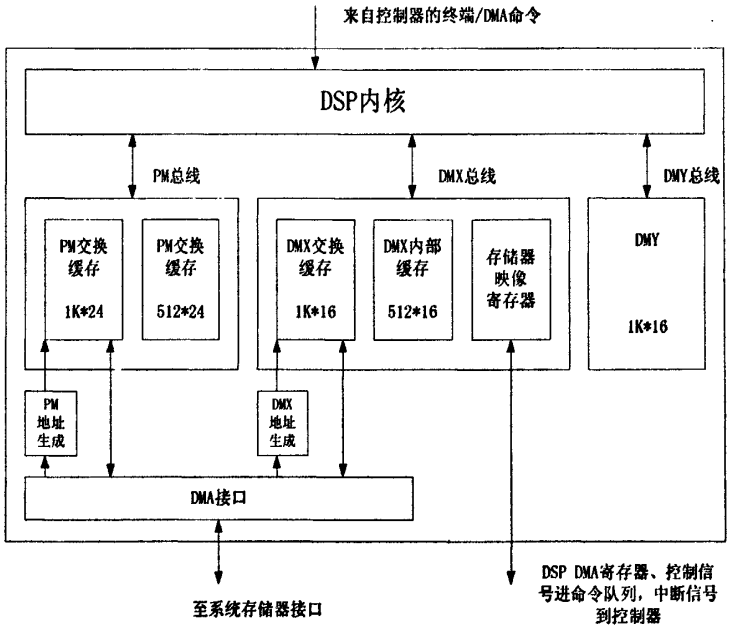


图 4-3 PA1688 DSP 子系统结构

DSP 的总线结构和 MCS8051 不同，它有专门的程序空间（PM）和数据空间（DMX 和 DMY）。DSP 与 SDRAM 之间的程序和数据的操作，需要使用 PM Exchange Buffer 和 DMX Exchange Buffer 缓冲区。当 DSP 与 SDRAM 之间进行

程序或数据操作时，DSP 会将数据首先读入上面的缓冲区，再从缓冲区中取得。图 4-4 表示了这两个存储器缓冲区的结构。

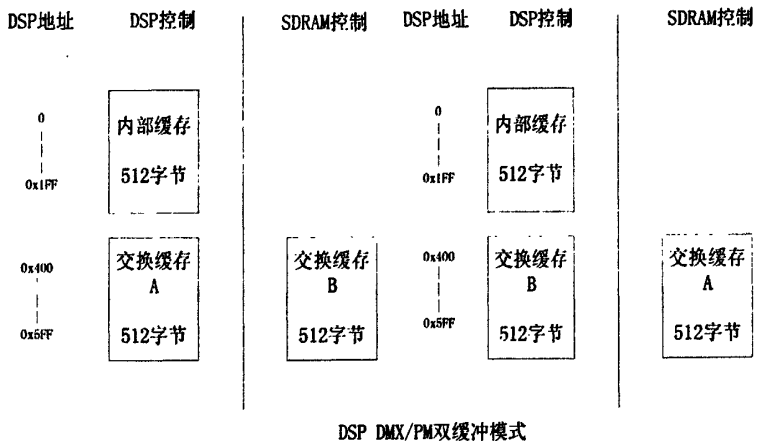


图 4-4 两个存储器缓冲区结构

PA1688 芯片集成了程序 Flash 接口，可以与数据总线宽度为 8bits 的 Program Flash 连接，外接的 Flash 最大容量为 16MB，可以编程修改的为前 8MB。Program Flash 即可编程的 Flash，是程序和初始设置数据的存储区，程序都放在 PFlash 中。MCU 在运行程序的时候，每次从 Flash 中取一条指令，其功能相当于 ROM，但与其有一个不同点，PFlash 由若干的 Sector（扇区）组成，删除和编程的操作都是针对单独一个 Sector 进行的，而不是对整个的 Flash。

由于 PA1688 片内的存储器空间不大，大量的数据都是通过 SDRAM 暂时存放和交换的，即动态存储区。PA1688 集成了 SDRAM 控制器，可以直接连接单片的 1M×16bit、4M×16bit、8M×16bit 和 16M×16bit 的 SDRAM，最大外接 4 片 SDRAM。在 IP Phone 应用中，我们只使用一片 SDRAM，因此最大可以为 16MB。

4.2 AR1688 介绍

和 PA1688 相比，AR1688 功能更强，集成度更高。其芯片特点和优势如下：AR1688 的 MCU 典型运行频率为 24.576MHz，最高运行至 60MHz，内部集成 116KB SRAM，无需外部扩展 SDRAM，最大支持 2MB 外部存储器，集成 24bit DSP 内核，DSP 内核运算能力达 72MIPS，外部接口有 UART 接口、键盘接口等，最大音量输出为 11MW（16 Ω）且低功耗。

4.2.1 AR1688 话机系统结构

AR1688话机系统由AR1688主芯片、网络单元、Flash单元、电源单元、键盘和显示单元组成，其框图如图4-5所示。

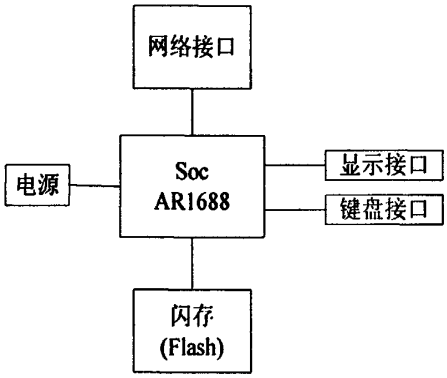


图4-5 AR1688系统框图

整个话机共分成以下几个部分的单元设计。

1. AR1688主芯片及语音单元

AR1688主芯片是整个网络电话系统的核心，它负责控制所有其他单元电路。

AR1688主芯片包含以下一些功能单元：

（1）存储器接口电路：包括地址线、数据线、读写控制和片选信号；

（2）键盘接口电路：可以直接连接按键系统；

（3）系统时钟电路：主芯片的时钟源；

（4）实时时钟电路：为AR1688主芯片提供时钟基准；

（5）ADC电路：实现A/D转换；

（6）UART电路：无硬件流控握手的标准串口；

（7）DC/DC电路：产生内部需要的内核电源电压；

（8）语音接口：提供语音输入输出接口。

语音单元包括：

（1）话机手柄接口；

（2）免提MIC电路；

（3）免提功放电路和侧音电路。

2. Flash单元

如果AR1688系统在更新软件或设置的过程中断电，那么更新的操作就不能保证正常结束。这时，系统可能需要从“安全模式”中恢复。

3. 电源单元

AR1688系统的主要供电电压是5V，系统所需的其他电压，可以由5V经过LDO或者DC/DC变换得到，整个系统大概需要的功耗如表4-2所示。

表4-2 系统功耗表（待机状态）

系统配置	5V工作电流/mA
单网口	95
双网口	270

根据外部电源的输入方式不同，在 AR1688 系统上需要实现不同的电源电路：

（1） 外部是 5V 直流输入，此时 AR1688 系统的电源电路最简单，可省略产生 5V 的 DC/DC 电路；

（2） 高于 5V 的直流或交流输入，这时需在 AR1688 系统上实现一个输入电压到 5V 的 DC/DC 电路；

（3） 外部采用 POE 供电，这时需要在 AR1688 系统上实现 POE 受电设备的 DC/DC 变换电路。

4. 其他单元

（1） 网络单元：AR1688 系统支持两种网络连接：单网口系统和双网口系统。单网口和双网口的网络结构不同，所以网络特性也不同，单网口的网络连接是 10Mb/s；双网口的网络连接是 100Mb/s。

（2） 复位电路：高版本的复位电路具有更好的复位波形。

（3） 安全模式：安全模式（Safe Mode）的作用是让 AR1688 系统能够进入 Page0 模式，以便能对系统进行特殊操作，如系统软件的恢复。

（4） UART:在 4.0 版本以前是做为单独的 UART 串口使用,以后作为 GPIO 接口电路来使用。

（5） LCD 接口：系统支持 16×2 的字符型 LCD，并带背光控制接口，采用 Motorola 的 M6800 标准接口，支持的 LCD 模块有：SED1278（SEIKO EPSON）、KS0066（SAMSUNG）、NJU6408（NER JAPAN RADIO）。

（6） LED：AR1688 系统一共带有 7 个 LED 指示，所有的 LED 指示的控制信号全部是低电平有效，即低电平亮，高电平灭。

（7） 键盘：AR1688 配置可 29 按键的键盘接口，其中 16 键是专用键盘实现，13 个是采用 ADC 电路实现。一般来说，由专用的键盘电路实现的键盘，具有比较高的抗干扰能力，而 ADC 电路实现的键盘，是通过检测不同的电平区分不同的按键，所以容易受电源电压精度、分压电阻精度、按键接触电阻大小、接插件及连线电阻等因素的影响，为保证按键的可靠，必须满足下列条件：

- 保证 3.3V 的电源精度不低于 2%；
- 所有的分压电阻采用 1%精度；
- 尽可能缩短键盘连线长度，减少连线阻力；
- 提高键盘 PCB 键盘触点的防氧化能力，避免触点氧化导致接触电阻增大；
- 注意键盘导电橡胶性能，保证按键的接触电阻尽量小。

（8） 键盘接口单元:厂家所提供的参考设计的键盘接口采用了 20pin 的 FPC 连接器，除了引出键盘信号外，还引出了所有的指示灯信号，方便用户安装。

4.2.2 AR1688 和 PA1688 的比较

AR1688与PA1688的性能对比如表4-3所示。

表 4-3 AR1688 和 PA1688 性能比较

性能	PA1688	AR1688
控制器 MCU 速度	22MHZ	48MHZ
DSP 运算速度	33MIPS	60MIPS
音频编解码器	集成，高性能	需额外的编解码芯片，WM9707
存储器	集成大容量 SRAM	需要额外的 SDRAM
1472 字节的 ICMP 响应	21ms	10ms
MD5 48 字节加密运算	9.9ms	4.8ms
开发环境	开源的 SDCC 编译器	商业版的 Keil C51

AR1688的硬件结构与PA1688基本一致，主要由一个控制模块和一个DSP模块组成，控制模块负责通信流程，主要包括协议栈的实现如IAX2、SIP协议、用户输入响应和显示输出等。DSP模块主要负责与语音处理相关的内容如语音编解码，DTMF产生与检测，基本课题的回声抵消等。此外AR1688芯片还包含了片内ADC/DAC，FLASH接口（这部分与PA1688相同），RS232接口等。AR1688芯片的DSP模块与ADSP2181芯片指令集兼容，采用哈佛结构、内部存储空间大、运算功能强、接口能力强。

（1）控制器（MCU）

采用8位高性能处理器进行系统控制和处理外围工作，它与Intel MCS8051完全兼容。AR1688芯片一条指令的运行时种为4-8个时钟周期（平均6个时钟周期），其最高运行速度为50MHz，数据总线为8位，提供256字节的内部数据存储器（Data RAM）。

（2）数字信号处理器（DSP）

AR1688片内集成的DSP是一个与ADSP2181指令集兼容的数字信号处理器，主要完成语音和图像编解码的运算功能。DSP的执行完全受MCU的控制。DSP的总线结构和MCU不同，它有专门的程序空间（PM）和数据空间（DMX和DMY）。DSP对SDRAM的访问需要使用PM Exchange Buffer和DMX Exchange Buffer缓冲区，当DSP与SDRAM之间进行程序或数据操作时，系统先将数据读入上述缓冲区，然后再由DSP从缓冲区取得。

（3）Controller总线、DataMem总线和SmartMedia总线

系统采用当今嵌入式系统通常采用的三组总线结构。通过Controller总线实现对外围设备（如PFlash、网卡等）的控制和通信；通过DataMem总线来扩展系统内存；通过SmartMedia总线提供SmartMedia（智能媒体）插槽接口，接入可抹

写功能的IC记忆卡。在本系统中用于连接LCD。作为升级功能，将来也可接入数码相机、DataFlash等SmartMedia设备。

（4） 程序Flash和SDRAM存储器

AR1688芯片集成了程序Flash接口，可以与数据总线宽度为8bits的Program Flash连接。MCU在运行程序的时候，每次从PFlash中取一条指令并执行。PFlash是程序和初始设置数据的存贮区，其功能相当于ROM。PFlash由若干的sector组成，删除和编程写入操作都是以一个完整的sector进行，而不是对整个PFlash。

由于AR1688片内的存储器空间小，大量的数据都是通过SDRAM暂时存放和交换。包括DSP程序和数据、声音数据、话机配置参数、TCP备份数据、文件系统、IVR数据、铃声数据、语音缓冲区以及语音留言数据。

（5） AC97 CODEC

AC97主要是完成听筒和免提的声音采集和播放。芯片采用8KHz采样、48KHz放音。收到的数据经解码后得的语音信号为8KHz，再通过数字滤波器变换，生成48KHz的语音信号。语音数据的交换是通过数据交换中枢SDRAM实现的。

发送语音（编码）：在MCU的控制下，首先将采集到的语音数据先放到SDRAM的语缓冲区中。DSP从SDRAM中取得语音数据进行编码，然后将编码结果放回SDRAM中。MCU从SDRAM中将编码结果读出，加上包头，如：RTP、UDP、IP、MAC，打包发出。

接收语音（解码）：对从网络上接收的数据包，首先在MCU中分析MCU头、IP头、UDP头、RTP头的数据。将拆分完头部后剩下的语音数据放到SDRAM的远端数据缓冲区中，然后由DSP从SDRAM中取数据进行语音算法的解码，并将解码结果放回SDRAM的播放语音缓冲区。由MCU控制直接从播放语音缓冲区中取数据，送到AC97进行播放。

4.3 AR1688 软件设计及实现

4.3.1 AR1688 软件结构

AR1688 系列的网络电话都采用了 2MB 的程序闪存，它们被分为 32 个页，每页 64KB，其具体描述如下：

1. 第一部分（Page0~Page11）

Page0：安全恢复功能和启动功能代码；

Page1：存放话机设置选项信息、呼叫规则和电话本数据，分别占用 16KB、16KB 和 32KB；

Page2~Page4：保存话机铃声数据信息，大小为 192KB（3×64KB）；

Page5~Page7：呼叫保持音乐数据信息，大小也为 192KB；

Page8~Page11：字库数据，其大小为 256KB，对应文件为 font.dat。

2. 第二部分（Page12~Page31）

这部分为主程序，HTTP 数据和 DSP 数据。

Page12~Page31 分为两段，程序和功能都相同，这是 AR1688 的安全特性。

Page12~Page21 为段 1，Page22~Page31 为段 2，通过标志位"system_page"区分，当 system_page=12 时，话机程序运行在段 1，此时更新话机程序更新至段 2；当 system_page=22 时，话机程序运行在段 2，此时更新话机程序更新至段 1。这种方式解决了话机更新新程序是因异常情况被迫停止或死机的情况。

4.3.2 AR1688 程序软件开发应用及工具

AR1688 采用 SDCC 开放源码编译器进行软件的编译，SDCC（Small Device C Complier）是专为 8 位微控制器开发的免费 C 编译器，兼容多种不同体系结构，对 8051、Z80 内核比较适合。SDCC 是命令行固件开发工具，含预处理器、编译器、汇编器、连接器和调试器，安装文件还捆绑了 SDCDB，类似于 gdb（GNU 调试器）的源码级调试器，程序采用 SDCC 编译，生成一个 Intel 十六进制格式的加载模块，之后将该文件加载到指定内存。SDCC 支持两种存储器模式：小模式和大模式，默认为小模式。SDCC 目录下的文件夹如下：

（1） bin 目录：该目录包含 SDCC 开放源代码编译器二进制文件和我们自己专用工具的二进制文件。

（2） doc 目录：

- txt 文件：我们将一些说明性文件放在此目录。
- include 目录
- .h 文件：我们只使用 SDCC 编译器，没有使用任何包含文件和库文件。所有的.h 文件都是 PalmMicro 提供的，同标准的 C 语言运行库函数相比有一点参数上的不同。

（3） lib 目录：它不是真正意义上的"库"，而是目标文件。这些目标文件是从源码编译而来的，该源码不是应用程序接口源码。这些目标文件同其他文件链接阶段被链接在一起。同"include"一样，在工程中没有标准的 SDCC 库。

（4） mcs51 目录：基于 AR168M VoIP 模块的 IP 电话参考设计一个外部 MCS8051 的 UI 控制器是为演示使用的，实际上它可以用于实际生产并保持一个非常低的 BOM 成本，在这里列出源代码的详细说明，其软件目录为 SDCC/MCS51.

- 使用较小的存储模式，开发源码的 SDCC 编译器编译；
- 在板子上不能通过软件进行升级，如果需要则要用编程器重新烧写；
- 界面包括 2×16 液晶显示屏、8×6 键盘和 4 个 LED；
- 如果 P4 端口可用还有额外的四个 LED；
- 采用 2797 字节的代码空间，其中包括扩展了 ISO-8859-1 字体的 2×16 液晶显示屏；
- 19200b/s 的 UART，8 位数据，一个中断，无校验；
- 振荡器在 22.1184MHz。

（5） src 目录：开放源码文件，包括 makefile 和批处理文件，用于生成升级的二进制文件。所有相关的用户接口和 VoIP 协议的实现都写入源代码。从代码数量来看，在用户接口程序中超过 80%是开放源码。

（6） tool 目录：专用工具的源代码，运行在一个 MS Visual C++6.0 工程中。

（7） dsp 目录：该文件夹中存放与语音处理的相关源程序，如回声抵消，舒适噪声生成，自动增益控制等。

4.3.3 程序仿真调试环境

本课题的自动增益控制程序采用的仿真调试环境是PalmMicro公司的调试软件PalmADSP，该软件能够加载由SDCC编译器生成的可执行文件，模拟AR1688芯片的指令执行情况，在调试中可以清楚地看到各个变量及程序代码在数据存储区、程序存储区的分布，数据寄存器与状态寄存器的值。

如图4-6的软件界面截图所示，界面内有三个窗口，分别显示寄存器，DM和PM的值，可以方便地查看程序，分配存储空间，并对代码进行调试。通过与定点C代码中的数据相对比，进行单步调试。DM中的值可以通过文件而导入和导出，对于单个需要改动的可通过双击DM的地址，直接进行修改等。

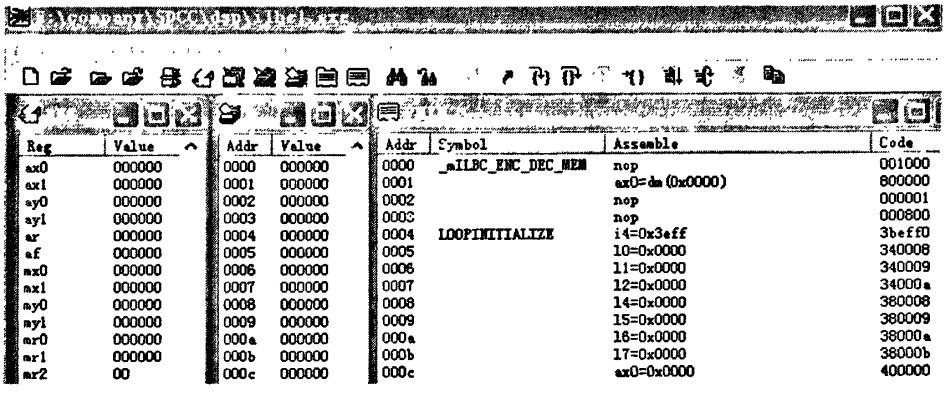


图 4-6 PalmAdsp 调试界面

第五章　自动增益控制算法的 DSP 实现及优化

学习了定点化的相关理论和DSP芯片的结构特点之后，本章重点介绍将AGC算法进行DSP代码实现的详细过程与步骤，本课题所实现的AGC算法是以开源的Speex语音编解码算法当中的compute_agc模块为基础，Speex提供了浮点C代码的实现，因此实现工作主要包括以下几个方面：首先在浮点C代码的基础上进行定点化实现，并对定点化的代码进行一致化比较以及初步的优化；再以定点C代码为参考，在24位定点DSP上实现该算法；最后对其进行汇编语言级别的优化，尽可能地减小代码的运算复杂度，使其更加适合在DSP上实际的应用。

5.1　AGC 算法代码浮点 C 到定点 C 的实现

第二章中对自动增益控制算法的原理进行了探讨，并对各模块的功能及实现进行了分析，第三章给出了定点化的要求及常用运算的实现方法，在此基础上，以改进型的 AGC 算法为例，本节对该算法的定点 C 语言实现进行了详细的描述。

在第二章对算法原理进行介绍时可以看出，整个算法包括一些相互独立的模块，模块之间通过参数的传递来完成整个算法处理的过程，因此在对算法进行定点 C 实现之前，要先对算法进行模块分析，分模块完成定点化工作，这样不仅有利于定点化工作的进行，同样有利于测试工作的开展。

5.1.1　模块分解

模块分析的第一步是子模块分解，这项工作的意义如下：

1) 对每一部分的实现方式可以单独考察，为在 DSP 芯片上的实现提供依据；
2) 对每一部分的实现代价(时间复杂度和空间复杂度)进行评估，提供算法优化的方向；
3) 可以发现可重用的模块，作为子函数提取出来，为利用原来实现过的模块提供了可能。

表 5-1 提供了改进型的 AGC 算法分解的结果，同时也给出了 C 语言的实现算法作为对 DSP 芯片上实现的参考。对于其他的 AGC 算法，可根据各自的处理流程及特点用类似的方式进行分解。

表 5-1 自动增益控制模块的分解

模块	实现方法	重点
能量估计值计算	对输入语音序列做平方运算	存储方式
噪声能量值估计	功率平滑	运算数据的中间格式

后验信噪比运算	能量比值	数据算法
先验信噪比均值运算	能量比值，平滑，递归平均	递推算法，数据格式
门限值 pframe 的计算	均值平滑	数据格式
能量和运算	乘累加，开方，平滑	存储方式，运算速度
增益系数计算	数据比值，幂运算	数据格式

根据第三章中提到的理论分析法和统计分析法这两种确定数据动态范围的方法，可以确定 AGC 参考代码中的各个变量的动态范围，从而进一步可以对各个变量进行定标。表 5-2 给出了 AGC 算法中一些主要变量的动态范围和定标值。

表 5-2　改进型 AGC 算法中主要变量的动态范围与定标值

数据所在模块	数据名	数据物理意义	数据理论范围	定标值	备注
	x[]	输入语音帧	[-32768,32767]	16.0	数据输入
能量估计值计算	window[]	加窗函数	[-1,1]	1.15	
	ps[]	能量估计值	[0, 2^32]	32.0	经放缩
噪声能量值估计	st_noise[]	能量估计值	[0, 2^32]	32.0	经放缩
后验信噪比运算	st_post[]	信噪比	[0,256]	8．8	
先验信噪比均值运算	old_ps[]	上一帧的能量估计值	[0, 2^32]	32.0	经放缩
	gamma	平滑因子	[-1,1]	1.15	
	st_prior[]	先验信噪比	[0,256]	8．8	
门限值 pframe 的计算	st_zeta[]	平滑后的先验信噪比	[0,256]	8．8	
	zframe	先验信噪比的递归平均	[0,256]	8．8	
	pframe	语音信号的概率	[-1,1]	1.15	
能量和运算	loudness_weight[]	能量权值	[-1,1]	1.15	
	loudness	能量和开方值	[0, 2^32]	32.0	经放缩
	rate	能量平滑因子	[-1,1]	1.15	
	st_loudness	能量和平滑值	[0, 2^32]	32.0	经放缩
增益系数计算	loudness_accum	平滑因子累积值	[-32768,32767]	16.0	
	init_max	中间比较值	[-32768,32767]	16.0	
	gain	增益系数	[-32768,32767]	16.0	

5.1.2　定点化 C 代码的实现

本文采用下面的流程来进行具体的实现工作。

首先是定点代码的框架建立工作，包括以下几个步骤：

（1）　新建一个 C 语言的工程，加入数据结构的定义文件及定点函数库的实现

文件；

（2） 把浮点代码中的全局变量根据定标值转化成相应的定点数据结构，文件的安排结构与参考的浮点算法要保持一致；

（3） 实现一组将中间数据输出的函数，作为调试的一种手段；

（4） 将参考代码中的函数定义用定点函数代替，函数体可以先简单地用空函数(对应于原来的函数无返回值)或者返回简单值的函数代替；

（5） 建立定点算法的框架后，就可以按照浮点代码中的执行顺序对各个子函数进行算法的定点化工作了。

整个定点化工作的过程如图 5-1 所示。下面将对子模块的实现方法，测试方式及集成方法作详细的说明。

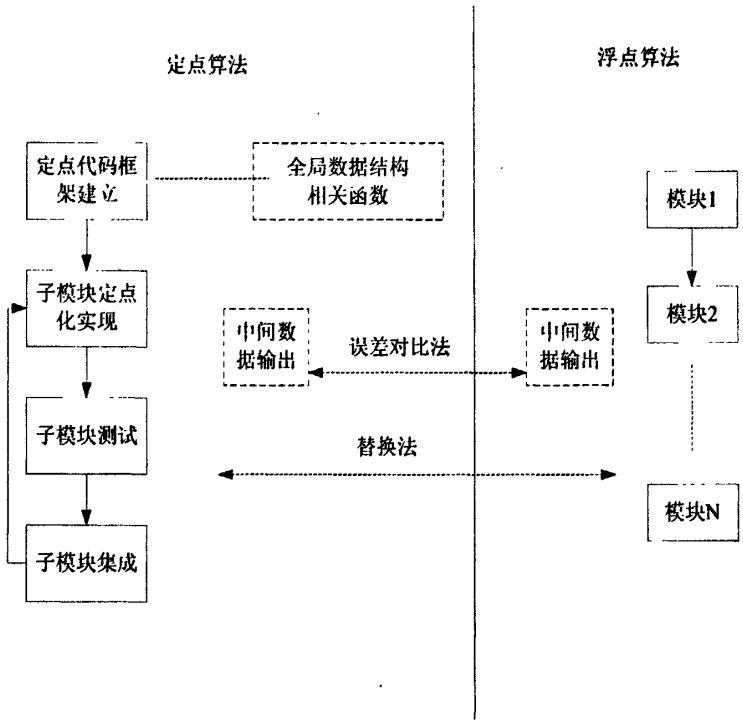


图 5-1 定点化工作的流程说明

1. 子模块的定点实现方法：

下面将以 AGC 算法中对每帧语音各样点作能量估计为例，具体说明定点化实现时采用的方法。图 5-2 是浮点算法的 C 语言代码，代码中的数据和运算都采用浮点的形式。

```
static void preprocess_analysis(SpeexPreprocessState *st, spx_int16_t *x)
{
   int i;
   int N = st->ps_size;
   int N3 = 2*N - st->frame_size;
   int N4 = st->frame_size - N3;
   spx_word32_t *ps=st->ps;

   /* 'Build' input frame */
   for (i=0;i<N3;i++)
      st->frame[i]=st->inbuf[i];
   for (i=0;i<st->frame_size;i++)
      st->frame[N3+i]=x[i];

   /* Update inbuf */
   for (i=0;i<N3;i++)
      st->inbuf[i]=x[N4+i];

   /* Windowing */
   for (i=0;i<2*N;i++)
      st->frame[i] = MULT16_16_Q15(st->frame[i], st->window[i]);

#ifdef FIXED_POINT
   {
      spx_word16_t max_val=0;
      for (i=0;i<2*N;i++)
         max_val = MAX16(max_val, ABS16(st->frame[i]));
      st->frame_shift = 14-spx_ilog2(EXTEND32(max_val));
      for (i=0;i<2*N;i++)
         st->frame[i] = SHL16(st->frame[i], st->frame_shift);
   }
#endif

   /* Perform FFT */
   spx_fft(st->fft_lookup, st->frame, st->ft);
    for (i=0;i<160;i++)
        x[i] = st->frame[i];
   /* Power spectrum */
   ps[0]=MULT16_16(st->ft[0],st->ft[0]);
   for (i=1;i<N;i++)
      ps[i]=MULT16_16(st->ft[2*i-1],st->ft[2*i-1]) + MULT16_16(st->ft[2*i],st->ft[2*i]);
   for (i=0;i<N;i++)
      st->ps[i] = PSHR32(st->ps[i], 2*st->frame_shift);

   filterbank_compute_bank32(st->bank, ps, ps+N);
}
```

图 5-2 能量估计运算的浮点 C 语言代码

其中，对各样点作 FFT 变换的函数代码如图 5-3：

```
void spx_fft(void *table, spx_word16_t *in, spx_word16_t *out)
{
   int i;
   float scale;
   struct kiss_config *t = (struct kiss_config *)table;
   scale = 1./t->N;
   kiss_fftr2(t->forward, in, out);
   for (i=0;i<t->N;i++)
      out[i] *= scale;
}
```

图 5-3 FFT 变换的浮点 C 语言代码

首先，我们需要把代码中的所有的浮点类型转化成定点类型，所有的浮点运算转换为定点运算，代码的结构可以不变。在之前的介绍中，我们已经得到了下面的定标表，这里我们可以直接利用。

表 5-3 能量估计运算的数据定标结果

数据所在模块	变量名	数据物理意义	数据理论范围	定标值
能量值估计	x[]	输入语音	[-32768,32767]	16.0
	window[]	窗函数系数	[-32768,32767]	16.0
	ps[]	能量估计值	[0, 2^32]	32.0

因此，可根据上图对该函数进行定点化实现，其中 FFT 变换经定点变换后的代码如图 5-4 所示。

```
void spx_fft(void *table, spx_word16_t *in, spx_word16_t *out)
{
   int shift;
   struct kiss_config *t = (struct kiss_config *)table;
   shift = maximize_range(in, in, 32000, t->N);
   kiss_fftr2(t->forward, in, out);
   renorm_range(in, in, shift, t->N);
   renorm_range(out, out, shift, t->N);
}
```

图 5-4 FFT 变换的初步定点化代码

在上面的代码中，用定点数代替了浮点数，在对两个数相乘时，浮点代码中定义为

#define MULT16_16_Q15(a,b)　　((a)*(b))

即两个数直接相乘，而定点后的 C 代码中则将其定义为

#define MULT16_16_Q15(a,b) (SHR(MULT16_16((a),(b)),15))

其中，#define MULT16_16(a,b)

(((spx_word32_t)(spx_word16_t)(a))*((spx_word32_t)(spx_word16_t)(b)))

因为在定点过程中，x[]和 window[]都变成了 Q15 的数，所以将两数相乘之后还有一个 Q 值的变化，即移位的问题。

对于其它的模块，可以用同样的方法进行定点化。在实际实现的过程中，应该按照程序的执行顺序进行，方便测试和集成。

2. 子模块的测试

由于定点数的特点，数据表示的精度和定点运算的精度通常都没有浮点数高，当数据用单精度表示时这一特点更加明显。在子模块的定点化实现完成后，必须对定点化的结果进行测试，以保证精度能够满足程序的要求。如果某一个模块与浮点数之间的误差比较大，则需要对数据的精度进行调整，或者使用更高精度的定点运算。

对于定点模块的测试，本文采用了两种方法[26]，第一种方法是误差对比法，第二种方法是代入替换法。

误差对比法就是对于某个特定的模块，用浮点算法得到一组输出的结果，然后用定点算法也得到一组输出结果，由于定点算法的输出 Q 值在定标的过程中已经确定，我们可以把浮点输出结果转化为定点数，通过对比两组数的相对误差就可以确定定点算法的精度要求了。图 5-5 表示了误差法的测试过程。

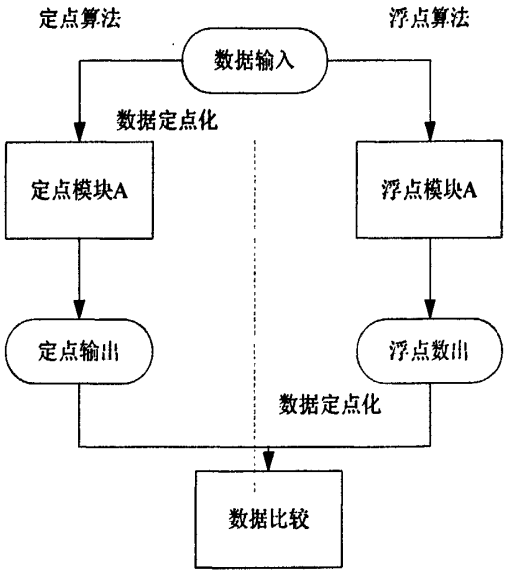


图 5-5 误差对比法的图示

下面以改进型 AGC 算法中能量估计值和后验信噪比的测试为例，说明利用误差法进行测试的过程。

1) 调试浮点代码，得到能量估计和后验信噪比模块的输出变量结果；

2) 把浮点代码中的能量估计模块和后验信噪比模块的输入数量化为定点模块对应的输入变量，目的是保证浮点模块与定点各模块的误差很小；

3) 调试定点代码，得到定点输出；

4) 进行浮点输出与定点输出的数据比较。

表 5-4 是对某个数据帧进行调试的结果，从调试中可以看出定点算法的精度。

表 5-4 能量值和后验信噪比的定点-浮点结果对比

ps[]		st_post[]		
浮点结果	定点结果(Q0)	浮点结果	浮点结果转为 Q8 格式	定点输出(Q8)
13.3252	15	5.66261	1450	1664
135.389	139	66.6943	17073	17536
202.527	204	100.264	25668	25856
66.748	66	32.3740	8288	8192
59.5654	61	28.7827	7368	7552
198.054	196	98.0271	25094	24832
186.060	186	92.0298	22559	23552
98.4823	98	48.2411	12349	12288

73.5604	74	35.7802	9160	9216
106.242	106	52.1212	13343	13312
211.747	213	104.874	26848	27008
234.123	234	116.061	27912	29696
99.7026	100	48.8513	12506	12544
16.2375	16	7.11877	1822	1792
119.701	119	58.8506	15065	14976
222.809	221	110.445	28273	28032
225.571	225	111.785	28617	28544
200.938	201	99.4690	25464	25472
142.101	141	70.0505	17933	17792
65.6628	65	31.8314	8148	8064
76.3461	76	37.1730	9516	9472
191.149	191	94.5744	24211	24192

由于能量估计值 ps[]的取值范围较大，对于不同幅值的语音信号其值相差较大，所以定标时采取 Q0 格式，因此这样会造成一定的误差和精度损失，从而对后续的数值产生影响。在测试过程中，通过反复的测试和定标值的改变来使后续的数值的误差尽量小。

误差对比法的优势在于可以明确地得到该模块与浮点模块之间的误差大小，对于改进原有的定点算法有着重要的意义。该方法也有其局限性，即在定点化完全实现之前，不知道这部分算法的精度对整个算法的影响。

第二种进行定点模块测试的方法是代入替换法，该方法可以更好地测试模块的性能。把浮点算法中的模块用定点化后的模块代替，并且在模块与其它模块的接口处进行相应的数据类型的转换，这样就可以测出这个定点化模块对整个算法的影响程度了。图 5-6 给出了替换法的实现图示。

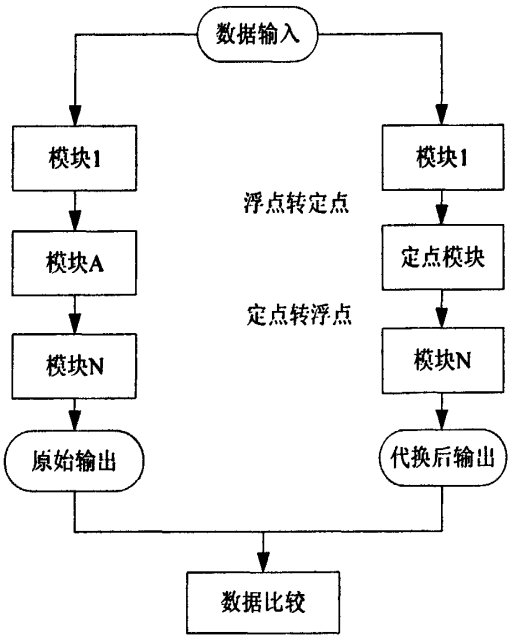


图 5-6 替换法进行精度测试图示

一般情况下，只对误差比较大的模块进行替换法的测试，以便检查定点模块引起的误差是否对整个系统的性能产生了较大的影响，如果是，则需要改进该模块的算法以提高精度。

当替换法的结果与未替换前对比没有较大的误差时，可以认为这部分的精度已经达到了要求，可以进行下个模块的实点化实现工作。

3. 模块集成与性能测试

在经过了对定点模块的实现与测试之后，就可以将这部分代码集成到定点化代码中了。通过对各个模块的测试、集成及修改，可以对整个定点 C 语言代码进行集成测试，即测试各模块组成的整体是否能够正确地执行功能。

我们需要对这时的定点化的结果进行测试，可以多选取一组语音数据，对整个定点代码进行测试。

对于 AGC 算法，发布者并没有提供标准的测试序列，在实际的工作中，是利用参考算法的结果作为参考标准进行的。测试时把相同的语音文件同时作为浮点算法和定点算法代码的输入，得到两份不同的输出文件，然后对文件中的数据进行对比，观察两者在数值上是否接近，或者波形是否吻合，如果有误差，观察误差是不是在可允许的范围内。同时，为了模拟网络环境和测试 AGC 的增益控制效果，除了对各组语音的输出效果进行测试之外，还对该算法的收敛速度进行了测试。

当定点化工作完成以后，一般来说，向汇编语言的转化是比较直观的。一方面，数据的定标值已经在定点化代码中体现出来，另一方面，定点化函数库与 DSP 芯片本身提供的函数库是有着明确的对应关系的。很多情况下，只需通过相

关运算符的替换就可以得到汇编语言的代码。

5.2 AGC 算法定点 C 代码到 DSP 汇编的实现

一般的 DSP 开发环境，都可以对 ANSI C 语言进行编译。为了使 DSP 开发变得更容易，一般情况下，我们都是利用编译器对定点 C 语言代码再根据编译器的特点进行一些小的改写，然后经过编译链接，最后对形成的代码进行运算量的统计分析，得到运算量较大的模块，用手工编写汇编代码的方式将这部分代码进行改写。然而，由于 DSP 芯片[27]是一个片内芯片，内部的 RAM 容量会受到芯片面积的限制，数据 RAM 和代码 RAM 分别为 16KB，因此将 C 语言先编译再优化的做法是行不通的，编译出来的代码长度将远远大于 16KB。

为了使代码长度尽可能的小，本课题中代码全部是用 DSP 代码直接编写的。因此，经过浮点 C 代码到定点 C 代码的转换之后，下一步的工作就是根据所采用的 DSP 芯片的实际情况，将定点 C 代码转换成相应的 DSP 代码，完成从算法到工程应用的过程。

5.2.1 设定文件结构

进行具体的代码编写工作之前，首先设计整个程序的框架，定义程序空间和数据空间的布局以及文件的相互依赖关系，使整个程序有一个紧凑、良好的架构，这样不仅可以提高代码的执行效率，合理利用芯片的存储空间，而且方便了后期的代码测试。

本课题将所有与 DSP 有关的文件和目录均放在已经设置好的目录 c:\sdcc\dsp 下面。其中，根目录下存放 .ach 文件，链接用到的文件以及 mk.bat 编译的批处理文件等，Common 目录存放公共的子函数和常量及变量定义文件，data 目录用来存放编程中用到的数据初始化的文件，agc 目录存放与 agc 有关的文件。

5.2.2 课题编程规则

在完成结构设定后，考虑到程序的可读性和可移植性，同时为了方便之后的一致化和调试测试工作的顺利进行和程序完成后的维护工作，在课题开展过程中还制定了统一的编程规则。

（1） 函数返回值

返回单精度数值，返回值在 SR0。返回双精度数值，返回值在 SR0，SR1。和单精度情况下低位保持一致，高位在 SR1 中。如果需要改动其它的值，用数组在参数中传递。

（2） 函数参数

只能传递 4 种参数，分别是单精度数值、双精度数值、DM 中的数组、PM 中的数组，且全部用寄存器传递。函数定义中参数次序依照上面，即先列出所有单精度数值，然后是所有双精度数值，再然后是所有 DM 中的数组，最后是所有 PM 中的数组。

单精度数值的值依次放在 AX0, AX1, AY0, AY1, MX0, MX1, MY0, MY1 中。双精度数值的值依次放在 MR，SR 中，高低位次序跟（1）中的函数返回值一致。

DM 中数组的地址依次放在 I0, I1, I2, I3 中，PM 中的数组的地址依次放在 I6, I7 中。

函数参数的寄存器值在调用函数过程中会改变。用函数 MULT16_48_Q15 举例，其中

int MULT16_48_Q15(int a, int b)

调用之前，MX0 中是 a，MY1 和 MY0 中是 b，MY1 是高 24 位，MY0 是低 24 位，调用之后，结果保存在 AY1 和 AY0 中。

（3） 栈(stack)和局部变量

所有局部变量都放在栈中， I4，M4 专门用于栈维护，其它地方一律不能使用。栈从 0x3bff 开始，往更小的地址使用。

固定的 M 寄存器值有 M0 = 0、 M1 = 1、M7 = -1，初始化后不允许再改动。M2、 M3、M6 可以在非中断处理程序中随意使用，M5 可以在中断处理程序中随意使用。

（4） L 寄存器值

除仅在中断中使用的 L5 外，其它初始化都要为 0。如果程序中改变了任何 L 值，返回前要把它改回到 0。在调用任何其它函数前都要保证 L 寄存器是 0。

（5） 地址空间

为了调试程序将地址空间分配如下：

DM 0000-0fff: 各个算法需要的全局变量数据

DM 1000-1fff: 公共算法和公共子程序需要的全局变量数据

DM 2000-3bff: 栈和局部变量

PM 0000-0fff: 各个算法需要的程序和 PM 参数数据

PM 1fff-3fff: 公共算法和公共子程序需要的程序和 PM 参数数据

总之，为了在取得汇编程序运行时的高效率的同时，又能保持 C 语言清楚、易读的特点，编写的原则就是对于程序主框架部分和没有大量运算的函数，尽量对 C 语言不做修改；而对于包含大量运算和数据处理的函数，则以高效为原则进行汇编语言的编写，达到提高效率、减少运算量的目的。

5.2.3 通用子函数编写

转换到 DSP 代码后，需要用到很多公共的函数，为了尽量节省程序存储空间以及提高代码重用性，同时也为了提高程序可读性从而便于调试和维护，尽可能地将每个函数拆分成小的功能函数，将公共的函数提取到统一的源文件中。

本次课题实现的函数有：DIV32、DIV32_16、Inv_sqrt、MULT16_32_Q15、spx_fft、spx_ifft、MULT16_16_P15、MULT16_16_Q15、Pow2、log2、Div_L、Div_S、pshr32_7、pshr32_15、pshr32、PSHR16、FLOAT_ADD。下面举例说明：

（1） DIV32

说明：双精度除法运算 {ay1,ay0}/{mr1,mr0}，ay1 和 mr1 表示高字节

输入：ay0,ay1,mr0,mr1

输出：{sr1,sr0}

用到的寄存器及初始值:

需包含文件<speex\basicop.h>

例子：

```
call Inv_sqrt;
ena m_mode;
mr1 = sr1;
mr0 = sr0;
ar = 1;
sr = lshift ar by 22(hi);
ay1 = sr1;
ay0 = 0;
call DIV32;
```

（2） Inv_sqrt

说明：双精度数值开方运算

输入：{mr1,mr0}，mr1 表示高字节

输出：{sr1,sr0},以 2.46 格式表示

需包含文件<common\basic_op.h>

例子：

```
mr0 = ar, ar = sr1 + ay1 + C;
mr1 = ar;
dis m_mode;
call Inv_sqrt;
ena m_mode;
```

```
mr1 = sr1;
mr0 = sr0;
ar = 1;
sr = lshift ar by 22(hi);
ay1 = sr1;
ay0 = 0;
call DIV32;
```

（3） MULT16_32_Q15

说明：单、双精度乘法运算 mx0*{mr1,mr0}

输入：mx0, {mr1,mr0}，mr1 表示高字节

输出：{mr1,mr0}

需包含文件<speex\basicop.h>

例子：

```
mx0 = sr0;
ar = dm(i0, m1);
sr = ashift ar by 3(lo);
ay1 = 0;
ay0 = 0x800;
ar = sr0 + ay0;
ay0 = ar,ar = sr1 + ay1 + C;
mr1 = ar;
mr0 = ay0;
call MULT16_32_Q15;
```

上面三个步骤完成之后，可以开始 DSP 代码的编写工作，即对模块的各个部分进行设计和实现。事实上，设计好的程序架构往往需要在程序编写过程和调试优化过程中作出调整，以优化存储空间的分配、节省芯片程序和数据存储空间，同时提高代码的执行效率和可维护性。

5.2.4 模块设计

对于不同的 DSP 芯片来说，其内部资源和指令集是不同的，本小节针对课题所采用的 AR1688 芯片，完成模块的设计。其他的定点 DSP 芯片在实现细节上会有所不同。

下面是对应的模块的具体实现过程：

（1） 能量估计值计算（preprocess_analysis）

位置：agc.asm

功能：对输入语音帧作能量估计，包括加窗，FFT 变换，能量运算等过程

输入：i0——inbuf_ptr（输入缓冲区指针）

i1——st_inbuf（输入语音帧各样值）

i2—— st_frame1（FFT 变换的输入）

i6—— st_window1（加窗系数）

输出：st_ps（输入语音帧的能量估计值）

其中，函数 power_spectrum 的功能主要是作能量运算，即对输入语音序列做平方运算：

输入：i0——st_frame（输入语音帧经加窗，FFT 变换后的语音序列）

输出：i1——st_ps（该语音帧的能量估计值）

实现程序：

```
ay0 = ax1;
ax0 = 14;
ar = ax0 - ay0;
dm(st_frame_shift) = ar;
seti(i2, st_frame1);
se = ar;
cntr = 160;
do frameshift1 until ce;
mr0 = dm(i2, m0);
sr = ashift mr0(lo);
frameshift1:dm(i2, m1) = sr0;
seti(i0, st_frame1);
seti(i1, st_frame);
call spx_fft;
seti(i0, st_frame);
i1 = ^st_ps;
call power_spectrum;
```

（2） 噪声能量值估计（update_noise）

位置：agc.asm

功能：完成能量估计值运算

输入：i0——st_ps（本语音帧的能量估计值）

i1——st_noise（本语音帧的噪音能量估计值）

i2——st_update_prob（对每个样点值求得的噪音概率值，0 或 1）

输出：i1——st_noise（本语音帧的噪音能量估计值）

程序实现：

```
call preprocess_analysis;
call update_noise_prob;
i0 = ^st_ps;
i1 = ^st_noise;
i2 = ^st_update_prob;
call update_noise;
```

（3） 先验、后验信噪比运算（compute_SNR）

位置：agc.asm

功能：完成对本帧各样点的先验信噪比和后验信噪比的运算

输入：i0——st_noise（本语音帧的噪音能量估计值）

i2——st_ps（本语音帧的能量估计值）

i3——SNR_post_ptr（本语音帧的后验信噪比的指针）

i6——st_prior（本语音帧的先验信噪比值）

i7——st_old_ps（上一帧语音的能量估计值）

输出：ay0——st_post（本语音帧的后验信噪比值）

sr0——st_prior（本语音帧的先验信噪比值）

程序实现：

```
ax0 = dm(st_nb_adapt);
ay0=1;
ar=ax0-ay0;
if ne jump no_adapt;
i0 = ^st_ps;
i1 = ^st_old_ps;
cntr = 208;
do update_oldps until ce;
ar= dm(i0, m1);
update_oldps: dm(i1, m1)=ar;
no_adapt: seti(i0, st_post);
i1 = ^st_prior;
call compute_SNR;
```

（4） 门限值 pframe 的计算（compute_pframe）

位置：agc.asm

功能：计算门限值 pframe

输入：i0——st_prior（本语音帧的先验信噪比值）

i1——st_zeta（平滑后的先验信噪比值）

输出：ax0——zframe（先验信噪比的递归平均）

ar——pframe（语音信号的概率）

程序实现：

```
dm(zframe) = ax0;
ay1 = 0;
ay0 = ax0;
mr1 = 0;
mr0 = 24;
call div32_16;
ax0 = sr0;
call qcurve;
mx0 = sr0;
my0 = 0x7312;
call MULT16_16_Q15;
ay0 = 0xccc;
ar = sr0 + ay0;
dm(pframe) = ar;
```

（5） 增益系数的计算（compute_agc）

位置：agc.asm

功能：计算增益系数

输入：i1——st_ps（本语音帧的能量估计值）

i6——st_loudness_weight（能量权值）

ax0——pframe（语音信号的概率）

输出：sr1,sr0——st_loudness（能量和平滑值）

ar——st_loudness_accum（平滑因子累积值）

ax0——st_agc_gain（增益系数）

i0——st_frame（语音序列与增益系数相乘之后的值）

程序实现：

```
seti(i0, st_post);
i1 = ^st_prior;
seti(i2, st_gain);
```

```
call compute_old_ps2;
i6 = ^st_loudness_weight;
i1 = ^st_ps;
call compute_agc;
```

5.2.5 DSP 代码编译和链接

模块的各部分设计完成后，接下来要将代码进行编译，生成可执行文件。课题编译环境使用了 DOS 下的 ADI_DSP，主要的命令行有 asm21，ld21。其中 asm21 主要是查找.asm 源代码的语法错误，如果没有语法错误则生成目标文件.obj；ld21 则是将生成的.obj 文件生成可执行.exe 文件。整个 AGC 算法作为一个完整的模块进行编译，图 5-7 是整个 AGC 算法的编译过程。

```
asm21 -2181 common\basic_op.asm
asm21 -2181 common\defines.asm
asm21 -2181 agc\agc.asm
asm21 -2181 echo\spxfft.asm
asm21 -2181 echo\spxifft.asm
asm21 -2181 speex\basicop.asm

ld21 -a ar1688 -e agc -g -i agc.txt -x

..\bin\adsptool -e -f -l agc.txt -x agc.exe -m agc.sym -d agc.dat
```

图 5-7　AGC 算法的编译命令

出图 5-7 可以看出，对于算法中的每个模块都需要进行编译，由于涉及到 ADSP 有许多型号的 DSP，为了进行语句的识别，在 asm21 后面加一参数-2181，将芯片的型号强制定为 ADSP-2181。各模块的程序是以.asm 为扩展名，最后两条语句的意思是按照文件 agc.txt 的内容进行链接，从而生成可执行文件 agc.exe，-x 表示产生一个.map 的文件，此文件能够将每个模块所占用的 PM 和 DM 详细地列举出来，便于代码的优化。

5.3 DSP 代码调试工作

在完成定点化 C 代码转化为 DSP 汇编代码的过程之后，为了验证转化后的结果是否正确，首先需要比较 C 代码和 DSP 代码实现的结果是否一致。

5.3.1 一致化验证方法

进行一致化的验证，主要使用以下三种方法：

1. 内存数据对比法

对定点 C 代码和 DSP 代码分别进行调试，将相应的数据在数据调试窗口中显示出来，通过对比数据是否一致，则可以知道 DSP 编程是否正确。

内存数据对比法的一个例子如图 5-8 如示，其中图左边的部分是用 Visual C++ 6.0 调试定点 C 代码的结果，图右边是 PalmADSP 仿真环境下观察到的结果。

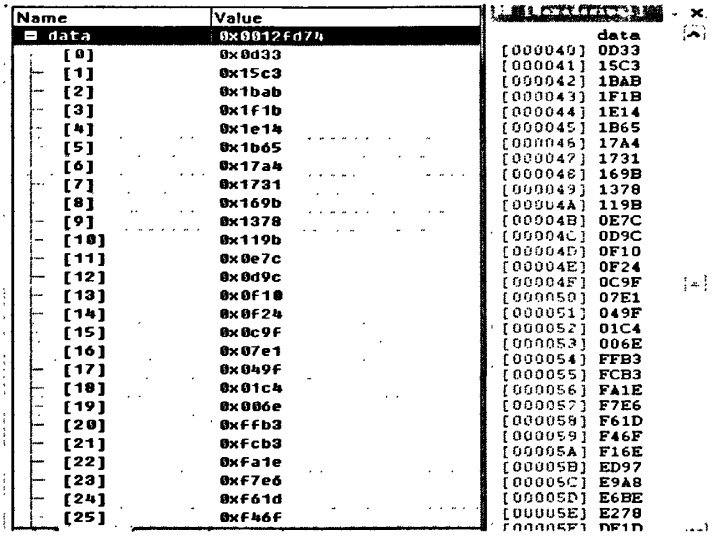


图 5-8 汇编代码的调试方法-内存数据对比法

这种方法适用于子模块的调试，以及将各个模块进行集成之后对于整体算法的前几帧的调试。程序最初开始进行调试时，不一致的情况出现的概率比较大，因此采取单步调试可处理大量的错误。而随着调试的进行，不一致的情况会越来越少，采用单步调试对内存数据进行对比的效率也会随着降低，因此针对集成调试需采用文件对比法。

2. 图形观察法

对于较小的子模块的调试，用数据对比法可以知道 DSP 程序的编写是否正确，但是，当涉及到较大的子模块时，内存数据对比法则不够理想，因为这种方法是以内部存储区的某一段为处理对象的，内存区的长度限制了可对比的内容的长度。这时，我们可以借助开发环境的图形输出功能[28]来观察实验的结果。

3. 输出文件对比法

利用图形进行观察比较直观，但是只能表征信号波形大体上的相似，还不够准确，当完成了所有模块的 DSP 汇编工作以后，对于系统的集成测试，经常采用的是文件对比法，即仿真器每次从输入文件中读入一段数据到内存区，进行处理后，把处理的结果输出到输出文件中，通过这种方法，可以对不同的语音片断的结果进行分析，从而发现代码中与定点代码不一致的地方，然后进行修改。如图 5-9。

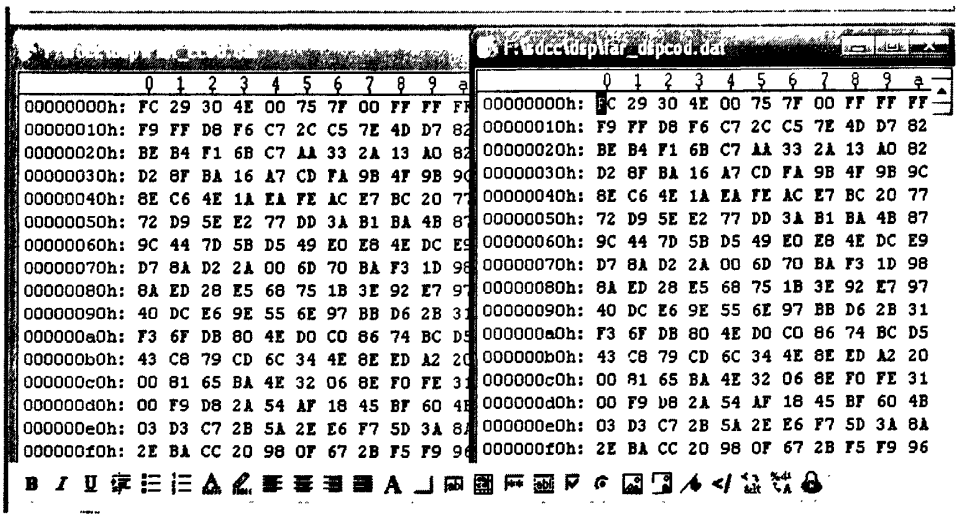


图 5-9 比较两个输出文件是否一致

5.3.2 一致化调试

在一致化过程中，我们首先根据上面的方法判断结果是否一致，本课题主要采用第一和第三种方法来判断一致化过程的错误量，然后根据输出的十六进制的运行结果来确定错误的位置，然后进行调试。在调试过程中，根据 AGC 算法自身的程序结构确定错误存在于哪个子程序，从而设置断点，进入子程序内部进行分步调试，直到最终确定错误所在的具体位置，最后进行改正。

在本课题的调试过程中，常见的错误举例和修改过程如下。

1. 变量溢出

这种情况往往出现在进行乘累加计算时，数据过大造成的。一般的处理方法是，扩展用于保存数据的寄存器数量，对数据分高低位存储，对有可能出现这种情况的地方提前进行溢出的判断和保护。如在计算能量和平滑值 st_loudness 时，需要进行乘累加运算，将 st_loudness 扩展成 64 位，其中 st_loudness_hi 表示高 32 位，st_loudness_lo 表示低 32 位。

2. 数据长度不一致

由于 AR1688 中 DSP 的数据长度是 24 位，对于 C 代码中 16 位的数据就存在数据精度不一致的问题，在对这类数据进行计算时就要注意最后数据的饱和处理。虽然编写 DSP 程序的原则是尽量与 C 语言保持一致，但算法最后是应用在 DSP 平台上，为了使运算更为准确，在处理这类情况时，应尽量提高 C 程序的精度。

3. DSP 汇编命令使用错误：

compute_agc 中对能量权值和 loudness 开方时，调用了公共函数 Inv_sqrt，而这个函数的返回值是开方的倒数，因此在调用该函数后，还应该进行一次倒数

运算，但 Inv_sqrt 结果是 2.46 格式的数据，若要进行倒数运算，需使用相同 Q 值的数据，即：

```
ar = 1;
sr = lshift ar by 22(hi);
ay1 = sr1;
ay0 = 0;
call DIV32;
```

这样才能得到正确的开方值。

4. 其他错误：

在进行除法运算时，要注意 Q 值的变化，例如，在求后验信噪比 st->post[i] 时：

```
st->post[i] = SUB16(DIV32_16_Q8(ps[i],tot_noise), QCONST16(1.f,SNR_SHIFT));
```

能量值是 Q0 的数，st->post[i]是 Q8 的数，因此在进行除法运算时，要先进行 Q 值的变换。

另外，在调试的过程中，还有几个方面的问题也是应该注意的。

（1） C 语言中的循环语句在 DSP 程序中要借助循环计数器 CNTR 来实现，除非使用条件跳转指令，否则只能是当循环计数器为零时，循环才会终止；

（2） DSP 程序的跳转判断条件只能是加法寄存器 AR 或 AF 状态的改变，即程序只能通过判断 AR 或 AF 的值是否为 0 来判断是否跳转，如

```
ar = pass ax0;
if ne jump out1;
```

就是通过判断 AR 是否为 0 来决定是否跳转到分支 out1。因此 C 语言程序的条件判断都要转换成 AR 或 AF 与零值的比较；

（3） 一般的，如果 C 代码中的函数参数不是很多，则 DSP 中对应的函数仍然通过指针寄存器和数据寄存器来传递，但如果函数参数过多，则要设置入口参数变量，用数据变量的形式传入参数。为了便于程序的调试，DSP 函数的定义必须明确地指出参数传递的方式以及变量名，如函数 power_spectrum，入口参数地址存放在 I0 中，输出结果地址存放在 I1 中，参数需要正确地进行传递；

5.4 代码的优化

为了提高算法的运行效率以及考虑到 DSP 芯片的容量和处理能力，还需对 DSP 代码进行优化。一般的，将代码占用的内存空间（包括 PM 和 DM）和 PalmADSP 中 MIPS（Million Instruction Per Second）的数值作为优化的两个指标。

5.4.1 程序组织结构的优化

要对程序的组织结构进行优化，需要从这两个方面考虑：

（1）多层循环的处理

程序中的多重循环是运算效率的瓶颈。当遇到多重循环时，最内层循环的一条指令，都有可能对整个程序的运算带来较大的影响。所以，碰到这种多层循环时，要尽可能地减少循环内部的指令数。而且这种减少对程序的总运算量的减少也是非常有效的。

（2）循环的展开与合并

对于循环次数较少的小循环，如 4 次以下，循环体内指令如果有比较简单的循环，我们可以将其展开，而不是用循环来完成。对于前后两个独立的循环，如果循环条件一致，则可以直接合并为一个循环，这样在省去循环条件判断的同时，有时也可以避免数据的重复赋值。

对于内层循环里的函数，为了减少指令，我们可以根据函数的调用情况，编写针对这种情况的程序，这样，就省去了程序执行时的一些不必要的判断及跳转。例如，在算法中会经常用到很多基本的移位等函数，这些函数都有一个共同的特点就是函数的规模小，有的甚至只有几行，但是调用的次数却非常频繁。这样，调用函数时，输入参数的赋值、函数调用、进入函数后参数的传递都需要额外的指令。对于这样的函数，优化的方法就是将函数展开，这样不仅省去了函数的调用，参数的传递也可以根据前后程序的实际情况灵活地进行设计。

5.4.2 基于 DSP 指令的优化

DSP 芯片提供了辅助的硬件单元和指令来提高算法的执行效率，在本课题的实现过程中，使用了以下方法来优化 DSP 代码。

1. 利用并行执行指令

利用并行执行指令可以在一个时钟周期里完成多个操作，这样就可以将多条指令合并为一条，提高程序的运行效率。利用并行指令也可以在进行运算的同时完成数据的存取。

执行并行指令的基本原则是先读后写，例如：

AR = AX0 + AY0, AX0 = DM(I0, M0)；与 AX0 = DM(I0, M0), AR = AX0 + AY0；的作用是一样的。

另外，执行多指令的规则如下：

(1) $\begin{vmatrix}\langle \mathrm{ALU}\rangle^{*+}\\ \langle \mathrm{MAC}\rangle^{*+}\end{vmatrix}$, $\begin{vmatrix}\mathrm{AX0}\\ \mathrm{AX1}\\ \mathrm{MX0}\\ \mathrm{MX1}\end{vmatrix}$=DM($\begin{vmatrix}I0\\ I1\\ I2\\ I3\end{vmatrix}$, $\begin{vmatrix}M0\\ M1\\ M2\\ M3\end{vmatrix}$), $\begin{vmatrix}\mathrm{AY0}\\ \mathrm{AY1}\\ \mathrm{MY0}\\ \mathrm{MY1}\end{vmatrix}$=PM($\begin{vmatrix}I4\\ I5\\ I6\\ I7\end{vmatrix}$, $\begin{vmatrix}M4\\ M5\\ M6\\ M7\end{vmatrix}$);

(2) $\begin{vmatrix}\mathrm{AX0}\\ \mathrm{AX1}\\ \mathrm{MX0}\\ \mathrm{MX1}\end{vmatrix}$=DM($\begin{vmatrix}I0\\ I1\\ I2\\ I3\end{vmatrix}$, $\begin{vmatrix}M0\\ M1\\ M2\\ M3\end{vmatrix}$), $\begin{vmatrix}\mathrm{AY0}\\ \mathrm{AY1}\\ \mathrm{MY0}\\ \mathrm{MY1}\end{vmatrix}$=PM($\begin{vmatrix}I4\\ I5\\ I6\\ I7\end{vmatrix}$, $\begin{vmatrix}M4\\ M5\\ M6\\ M7\end{vmatrix}$);

(3) $\begin{vmatrix}\langle \mathrm{ALU}\rangle^{*}\\ \langle \mathrm{MAC}\rangle^{*}\\ \langle SHIFT\rangle^{*}\end{vmatrix}$, $dreg=\begin{vmatrix}DM(\begin{vmatrix}I0\\ I1\\ I2\\ I3\end{vmatrix}, \begin{vmatrix}M0\\ M1\\ M2\\ M3\end{vmatrix})\\ PM(\begin{vmatrix}I4\\ I5\\ I6\\ I7\end{vmatrix}, \begin{vmatrix}M4\\ M5\\ M6\\ M7\end{vmatrix})\end{vmatrix}$;

(4) $\begin{vmatrix}DM(\begin{vmatrix}I0\\ I1\\ I2\\ I3\end{vmatrix}, \begin{vmatrix}M0\\ M1\\ M2\\ M3\end{vmatrix})\\ PM(\begin{vmatrix}I4\\ I5\\ I6\\ I7\end{vmatrix}, \begin{vmatrix}M4\\ M5\\ M6\\ M7\end{vmatrix})\end{vmatrix}$ = $dreg$, $\begin{vmatrix}\langle \mathrm{ALU}\rangle^{*}\\ \langle \mathrm{MAC}\rangle^{*}\\ \langle SHIFT\rangle^{*}\end{vmatrix}$;

(5) $\left|\begin{matrix}\langle \mathrm{ALU}\rangle^{*} \\ \langle \mathrm{MAC}\rangle^{*} \\ \langle SHIFT\rangle^{*}\end{matrix}\right|$，$dreg = dreg$；

其中，<ALU>表示除 DIVS，DIVQ 指令外的任意 ALU 指令；

<MAC>表示任意乘累加指令；

<SHIFT>表示除移位立即数之外的任意移位指令；

*表示非条件指令；

+表示必须使用 AR 和 MR 寄存器，而不是 AF 和 MF 反馈寄存器。

2. 利用地址生成器的特性

在访问存储单元的同时改变地址值，加快访问的速度。

3. 在单指令中完成转移指令与赋值运算

4. 使用第二套运算寄存器

另外，还存在其它方面的一些优化，主要包括：

1. 利用空间换时间的优化

对于一些规模比较大的函数，在编写程序时会出现寄存器不够的情况，这时就需要我们开辟堆栈来存储这些变量。但是，在 DSP 程序中，堆栈的读取是用宏操作来实现的，这就要比寄存器的读取多出 4~5 条指令。如果是位于内层循环中，可以考虑开辟 DM 来存储这些数据，以减少指令、提高效率。

2. 存储区空间的节省

在 C 程序中，对于函数体中的一些局部变量，当函数调用完成后，其生命周期也就结束了，而在 DSP 程序中，变量都是需要手工分配，也不会对变量所占用的空间自行释放，因此就需要我们仔细考虑并计算好变量的长度和生命周期，对变量的空间进行重复利用。首先，充分利用寄存器来存储局部变量，完成参数的传递；其次，对于较长的数组，在其生命周期结束后，可以对它分配的空间进行重复利用，而不必在程序开始时另行开辟空间，但须做好完备的注释；最后，分析数据的作用时间，尽量减少中间不必要的变量传递。

以上两方面究竟是要用空间换取时间，还是牺牲时间换取空间，要根据具体的情况进行分析，综合考虑。

5.4.3 代码的优化效果

最终，对本课题所采用的两种 AGC 算法的 DSP 代码性能进行分析，结果如表 5-5 所示，其中对于基于 VAD 的 AGC 算法，如果不选用 VAD 模块，单独的 AGC 模块占用 8MIPS。

表 5-5 最终实现的 DSP 代码性能

AGC 算法	基于 VAD 的 AGC 算法	改进型基于能量比较的 AGC 算法
代码空间(K)	6.887	4.818
数据空间(K)	1.92	1.3
运算复杂度(MIPS)	20	10

本课题所采用的 DSP 芯片最大的速率是 72 MIPS，而这 72 个 MIPS 不能都用来做语音编码和预处理方面的，它还要分配一些 MIPS 语音编码的外围主程序，包括字节顺调整、格式转换等，而且当控制芯片读取处理后的数据时会暂停 DSP 芯片的操作，这也是会影响 DSP 芯片的处理速度的。其中外围主程序占用约 3 个 MIPS，而控制芯片读取数据时对 DSP 芯片速度造成的影响虽然现在还不能精确地通过仿真测量，但是按平时经验来看影响不大，也在 3 个 MIPS 左右，所以大约还有 66 个 MIPS 可以留给语音编码和预处理模块。

我们平时所说的 MIPS 都是平均的复杂度，但是在硬件上应该保证的是程序复杂度的峰值要小于硬件所允许的大小，所以一般程序的复杂度要求小于 56 个 MIPS。针对不同的语音编解码算法，如 G.729 算法，其优化后的复杂度约为 15MIPS，因此有约 41 个 MIPS 可以用来进行 VAD 和自动增益控制等预处理；又如 Speex 算法（15kbps 模式），其复杂度约为 34MIPS，因此有约 22 个 MIPS 可以用来进行自动增益控制等预处理。

另外，该 DSP 芯片最大可使用的 DM（数据存储区）和 PM（程序存储区）均为 16K，对于不同的语音编解码算法，如 G.729 算法，其 DM 和 PM 分别约为 9.3K 和 11.1K；又如 Speex 算法，其 DM 和 PM 分别约为 9.4K 和 11K。因此，由表 5-5 可知，硬件空间是足够用的，该 AGC 算法能够实际应用于该平台。

在实现完定点汇编代码以后，进行了仿真器上完整的输入文件的自动增益控制效果测试，对于 AGC 算法，我们采用了大量不同的输入语音文件，尽量覆盖可能多的语音情况。在此基础上将算法成功地写入 AR1688 芯片，进行了实际的通话测试，结果表明，可以实现较好的自动增益控制效果，长时通话效果良好。

5.5 AGC 算法效果测试

5.5.1 基于 VAD 的 AGC 算法效果测试

以一段断续的语音为例来说明该算法的性能。图 5-10 是断续的原始无噪语音，图 5-11 是经过 AGC 模块处理后的语音输出波形，图 5-12 是使用基于 VAD 的 AGC 模块处理后的语音输出波形。从波形可以看出，只使用 AGC 模块时，

语音输出电平得到调整，且保持在一定范围内，但静音信号同样被增强；同时使用 VAD 和 AGC 模块时，在放大语音的同时保持了原有的信号波形，失真小并具有一定的鲁棒性，且静音信号段输出为 0，很好地实现了自动增益控制功能。

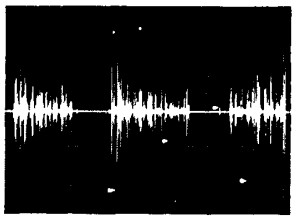


图 5-10 原始无噪语音

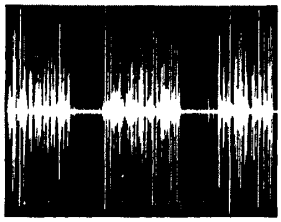


图 5-11 经 AGC 处理后的语音

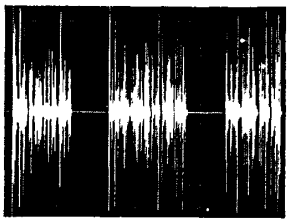


图 5-12 经 VAD 和 AGC 处理后的语音

5.5.2 改进型 AGC 算法效果测试

对基于 VAD 的 AGC 算法来说，当语音能量较大时，可以调节增益减小输出的语音幅值，当语音能量较小时，可以提高增益增大输出的语音幅值，但在实际的话机测试中，当双方通话过程中处于静音时，会有小噪音进入，则可能判断为语音引起误判，进而会提高增益，放大噪音。因此，改进型的 AGC 算法中引入一个门限值用来判断语音帧和非语音帧，从而减小误判的概率，防止小噪音被放大进而影响通话质量。

一、门限值 pframe 的确定

对于改进后的 AGC 算法的 C 程序，首先输入一段幅值较小的白噪声，对各个帧得到的 pframe 值进行输出并在 matlab 中画出曲线图，如图 5-13 所示。再输入一段较平稳的语音序列，如图 5-14 所示，对各帧得到的 pframe 值进行输出并在 matlab 中画出曲线图，如图 5-15 所示。

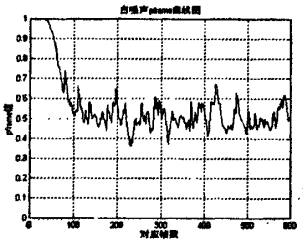


图 5-13 白噪声的 pframe 曲线图

图 5-14 一段较平稳的语音序列

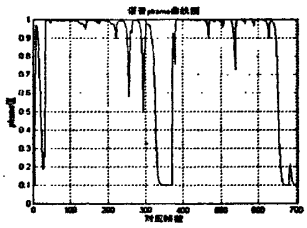


图 5-15 语音的 pframe 曲线图

由图可以观察出，噪音的 pframe 值一般在 0.4~0.7 附近，而对于语音，pframe 的值集体在 0.9~1.0 附近，因此取该门限值为 0.9

二、改进型 AGC 算法的仿真及分析

以一段幅值变化较大的语音为例来说明该算法的测试性能。图 5-16 是原始无噪语音，图 5-17 是经过 AGC 模块处理后的语音输出波形，从波形可以看出，经过 AGC 处理后，语音输出电平得到调整，且保持在一定范围内，防止声音忽大忽小不均衡。

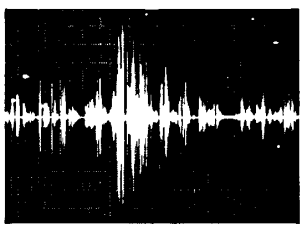


图 5-16 原始无噪语音

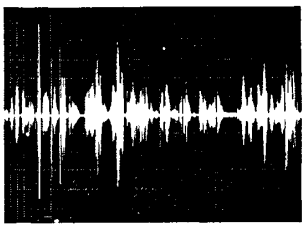


图 5-17 经 AGC 处理之后的语音

5.6 实际的自动增益控制效果测评

选择一段较平稳的语音来测试 AGC 程序的效果，在 CoolEdit2000 语音工具软件中查看语音，同时可以直观的看到语音波形图，通过图形的对比说明该程序的作用。图 5-18 是平稳的原始语音，图 5-19 是设定的每帧对应的增益值，将对应帧的语音信号与之相乘，得出的语音波形如图 5-20 所示。将该信号通过 AGC 模块进行处理，记录每帧对应的增益值如图 5-21 所示。根据对图 5-18 的原始语音进行观察，分析得出：

（1）0~100 帧，设定的增益值由 0.5 逐渐增大，语音信号刚开始较小，所以 AGC 得到的增益值逐渐增大，随着语音信号的先增大后减小，AGC 增益值先减小后增大。

（2）100~300 帧，设定的增益值保持不变，得到的 AGC 增益值基本上保持与原始语音对应，当语音减小时，增益值变大，当语音增大时，增益值减小。

（3）300~345 帧，设定的增益值急剧增大，最大值为 6.0，语音信号先减小后增大，则对应的 AGC 增益值先增大后逐渐减小到最小值。

（4）345~700 帧，设定的增益值逐渐减小，AGC 增益值对应地呈上升趋势。

（5）700~900 帧，设定的增益值逐渐增大，初始语音较小，AGC 增益值增大，接着语音信号增大，AGC 增益值呈下降趋势，由于在 800 帧开始时的语音信号幅度小，增益值不断上升，随着语音信号稍微变大，设定的增益值也不断增大，AGC 增益值呈下降趋势。

图 5-22 为经过 AGC 模块处理得到的语音波形，经过以上分析可以得出，该 AGC 程序达到了自动增益控制的目的。

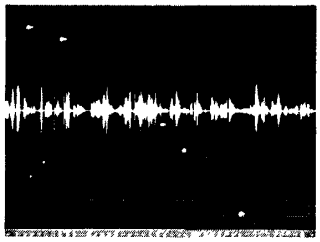


图 5-18 平稳的原始语音

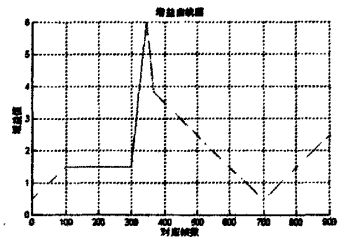


图 5-19 设定的增益值

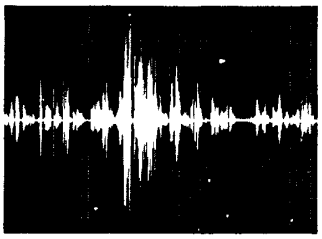


图 5-20 原始语音与增益值相乘

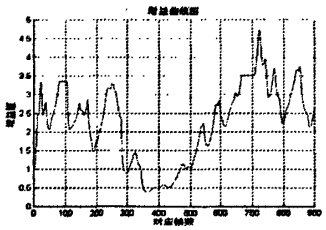


图 5-21 AGC 模块得到的增益值

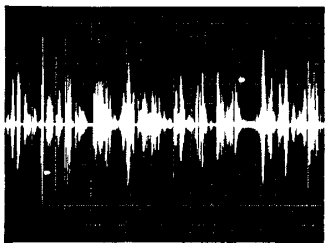


图 5-22 AGC 模块输出语音

第六章　总结与展望

6.1　本文工作总结

随着互联网的快速发展，互联网语音通信应用日益普及，IP 电话以其网络带宽利用率高、通话费用低、承载业务多而得到广泛应用。为了消除 IP 电话网传输过程中产生的延时、抖动、回声等现象的影响，需要对语音信号进行预处理，改善原系统在外界环境干扰条件下的性能。实用语音预处理系统主要包括降噪系统、回声控制系统、语音激活检测模块和自动增益控制模块等。其中自动增益控制模块能稳定信号传输电平，根据输入信号电平大小和指定的输出电平，自动调整电平变化，并且不影响传输信号尤其是语音的质量，保证经过控制后的语音可懂度不会发生波动。

本文在研究自动增益控制算法原理的基础上，重点研究了基于 VAD（Voice Activity Detection）的 AGC 算法和基于能量比较的 AGC 算法，主要工作是将 Speex 语音编码算法中用到的自动增益控制算法在 DSP 芯片上实现，应用于 VoIP 系统的话机终端。重点和难点在于算法的优化和定点的 DSP 汇编实现，通过处理得到比较平稳的输出信号，达到较理想的增益控制、平稳输出的目的，从而实现自动增益控制的目的。

首先，借助 PalmADSP 等仿真工具，基于算法提出者给出的参考浮点代码，以基于能量比较的 AGC 算法的最终实现为目标，完成了 AGC 算法浮点 C 语言代码的定点化工作。随后，将 C 语言定点化程序转化为 DSP 汇编语言并进行一致化调试，成功地进行了代码搬移。最后，完成了优化工作和测试结果分析，并将代码写入 AR1688 芯片，应用于话机终端。

综合起来，本文主要分成以下几个部分进行研究。

第一部分：研究课题背景，在了解了 VoIP 系统中语音信号预处理的结构的前提下，研究了自动增益控制模块所采用的算法，并在此基础上确定了开发平台和实现流程。

第二部分：比较了几种常用的自动增益控制算法，重点分析了基于 VAD 的 AGC 算法原理，并研究了改进型的基于能量比较的 AGC 算法，研究其浮点算法，深入理解算法的思想及优点，深入分析 C 语言源代码库，理解算法流程中各模块的细节。

第三部分：介绍了定点理论、定点运算和基本函数库的实现，为下一步的定点化工作提供理论指导，并实现所用算法中常用的基本函数，完成初步的定点化

工作。

第四部分：主要介绍了开发平台，详细地介绍了 AR1688 芯片的结构和特点，并学习了软件设计方法。

第五部分：首先把标准浮点 C 代码转化为定点 C 代码，再利用 Palmmicro 公司开发的调试工具，用汇编语言完成 AGC 算法 C 语言代码到 DSP 代码的搬移，并在与 ADSP2100 系列指令兼容的芯片上进行一致化调试。然后，完成代码优化并进行测试及结果分析。最后，把算法成功地写入 AR1688 芯片，进行了实际的通话测试。

第六部分：将本课题进行总结，并提出下一步的工作和展望。

6.2 下一步的工作

在自动增益控制算法实现的过程中，对其中的一些算法模块进行了研究，发现某些算法和代码还有进一步优化的余地，主要还有两个方面的工作可以做：

第一方面：基于代码量上的优化工作，可以将定点 C 代码进一步优化，使得定点 C 代码的效率更高，可移植性更强，最终可以较容易地移植到新的平台上；

第二方面：基于算法本身的优化，把参考代码中的一些模块用更优化的算法来代替，如对增益系数的调节，可结合预处理模块中其他模块如降噪模块一块来实现，可以得到更好的效果。

6.3 技术展望

在当前通信网络全 IP 化的趋势下，各大运营商和设备商都投入大量人力物力进行研究，利用 IP 网络传送语音数据的 VoIP 系统发展很快。本文研究的目标是在话机终端的 24 位定点 DSP 芯片上实现适合于 IP 网络传输环境的自动增益控制模块，文中所依据的理论根据和采用的工程方法同样适用于其他应用的开发。最可预见的有两方面的技术展望：

第一方面：通过自动增益控制，优化可视通话和电话会议等应用的效果，并且终端可以由话机发展为移动手机。

第二方面：随着网络技术的发展，也可以在如 WIMAX，WIFI，蓝牙等无线网络中适用。

今后，随着芯片处理能力的增强以及更多高效实用的软件的开发，在终端实现方面将有更加丰富的应用，将会为消费者提供新的服务，为企业提供新的盈利点。

参考文献

[1]李建峰. IP 电话的实现及其关键技术[J].山西电子技术, 2006(12).

[2]苏苇,毛宏艳. 浅谈 VoIP 技术[J]. 沈阳工程学院学报, 2006(04).

[3]黄永峰. 因特网语音通信技术及其应用. 北京:人民邮电出版社, 2002.

[4]文成义译. 语音处理. 北京:国防工业出版社, 1996.

[5]C.E.Shannon, A mathematical theory of communication.Bell System Technical Journal,1948,27:379-423,623-656.

[6]易克初. 语音信号处理. 北京:国防工业出版社, 2000.

[7]何振亚. 自适应信号处理. 北京:科学出版社, 2002.

[8]朱民雄,闻新,黄健群等. 计算机语音技术. 北京:北京航空航天大学出版社, 2002.

[9]蔡凌云. 自动增益控制技术应用[J]. 电子工程师, 2002,28(4):22-23,37.

[10]Young P. Electronic Communication Techniques 4th. Ed.,Prentice Hall, 1999.

[11]毕无敌,张洪君. 自动增益控制(AGC)电路特性研究[J]. 山东师大学报, 1993, 8(1):31-35.

[12]Steber G.R.. Digital Signal Processing in Automatic Gain Control Systems. Industrial Electronics Society. IECON-88. Proceedings, 14 Annual Conference of,1998,2:381-384.

[13]Lovrich A., Troullinos G., Chirayil R.. An All Digital Automatic Gain Control. Acoustics, Speech, and Signal Processing, ICASSP-88., 1988 International Conference on,1988,3:1734-1737.

[14]徐济仁,陈家松,徐屹. 语音信号预处理技术综述[J]. 电子工程师, 2001,27(6): 26-27,53.

[15]李夕红,祝忠明,谢兴红. 音频信号采集与 AGC 算法的 DSP 实现[J]. 电子产品世界, 2007(04).

[16]何志远,李显文,韦岗. G.169 自动电平控制装置标准及其实现[J]. 数字声频, 2003,03(11).

[17]Jongseo Sohn, Wongyong Sung. A Statistical Model-Based Voice Activity Detection. IEEE Signal Processing Letters, 1999(1):1-3.

[18]Jongseo Sohn, Wongyong Sung. A Voice Activity Detector Employing Soft Decision Based Noise Spectrum Adaptation. IEEE International conference on Acoustics, Speech and Signal Processing, 1998-01:365-368.

[19]李亚民. 计算机组成与系统结构. 北京:清华大学出版社, 2004.

[20]Erick L.Oberstar, "Fixed Point Representation and Fractional Math", Oberstar Consulting, 2005.

[21]ADI Corporation. Using the ADSP-2100 Family Volume1(Rev 1.0), 1990.

[22]张雄伟,陈亮,徐光辉. DSP 集成开发与应用实例. 北京:电子工业出版社, 2002.

[23]徐科军,黄云志. 定点 DSP 的原理、开发与应用. 北京:清华大学出版社, 2002.

[24]张雄伟,曹铁勇. DSP 芯片的原理与开发应用（第二版）. 北京:电子工业出版社, 2000.

[25]刘洪林,蒋昌茂,张建勇. IP 语音通信原理、设计及组网应用[M]. 北京:电子工业出版社, 2009.

[26]A.G.M Cilio, H.Corporal. Floating Point to Fixed Point Conversation of C Code. Delft University of Technology: Computer Architecture and Digital Techniques Dept, 2004.

[27]ADI Corporation, ADSP-218x DSP instruction Set Reference, 2004.

[28] ADI Corporation, Visual DSP++ 2.0 User's Guide for ADSP21xx DSPs, 2001.

[29]郭莉,殷南,王炳锡. 语音业务中鲁棒性 VAD 算法分析[J]. 电声技术, 2005(09).

[30]Shan T.J., Kailath T.. Adaptive algorithms with an automatic gain control feature. IEEE Transactions on Circuits and Systems, Volume 35, Issue 1, 1988 Page(s):122-127.

[31]Lovrich A., Troullinos G., Chirayil R.. An all digital automatic gain control. ICASSP-88., 1988 International Conference on Acoustics,Speech,and Signal Processing, 1988.

[32]Ramirez Barajas J., Dieck Assad G., Soto R.. A fuzzy logic based AGC algorithm for a radio communication system. The Ninth IEEE International Conference on Fuzzy Systems, 2000.

[33]Heitkamper P., Walker M.. Adaptive gain control for speech quality improvement and echo suppression. IEEE International Symposium on Circuits and Systems, 1993.

致 谢

在北京邮电大学的硕士学习生活即将结束，回顾这两年多的时光，感慨良多。尊敬的师长和亲爱的同学们给予了我极大的帮助和鼓励，他们教会我如何学习，如何工作，如何生活，更教会我如何保持良好的心态，积极地应对各种困难和挑战。在此，我要向所有关心和帮助过我的老师和同学们表示最诚挚的谢意！

首先，我要感谢我的导师黄孝建教授，在这两年多的时间里，黄老师严谨认真的工作态度，对学术研究孜孜以求的精神、勇于开拓创新的作风和平易近人的师者风范指引着我，教会我为人处事的正确态度和方法，为我今后的工作和生活树立了良好的榜样。

其次，要感谢微掌公司的林蓉榕、李敬老师，在课题的工程实践开展过程中给予了我大量的指导和帮助，耐心地教会我熟悉开发环境，详细地讲解工作原理和工作流程，在对我理论研究能力和工程实践能力的培养上起了非常重要的作用，是不可多得的良师益友。

再次，还要感谢李莉、吴晓宇同学，感谢她们在工作中的相互合作和帮助。

另外，还要特别感谢我的家人，感谢他们一直以来给我的无私的爱和无限的关怀，他们始终是我不断奋斗和进步的最坚强的后盾。

最后要感谢各位对论文进行评审的专家学者，他们在百忙之中对我的论文不吝给予指正，在此表示由衷的感谢。

作者攻读学位期间发表的学术论文目录

[1]李菁菁,黄孝建,李敬. DTMF 信号的产生与检测在 ADSP-2181 中的实现. 中国科技论文在线, 2009,12.