ARM-Linux 嵌入式语音终端

李雄飞,黄 冰,梁 艳
(桂林电子科技大学 信息与通信学院,广西 桂林 541004)

摘 要:基于 ARM 体系构架和嵌入式 Linux 操作系统构建了算法平台的硬件和软件结构,并在该平台移植了 G.729 语音编解码算法。通过对软件优化设计以及采用基于 ARM 指令集的算法优化策略,对 G.729 编解码器进行优化,提高了系统运行速度。

关键词:ARM;嵌入式 Linux;G.729;语音处理

中图分类号:TN9 **文献标识码**:A **文章编号**:1674-7720(2011)05-0009-03

ARM-Linux embedded speech terminal

Li Xiongfei, Huang Bing, Liang Yan
(Department of Computer Science and Technology, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: In this paper, it described the structures of hardware and software needed by algorithm which is based on the architecture of ARM and embedded Linux operation system, and G.729 speech codec is transplanted in this platform. In the application of this algorithm, through optimizing design of the software and by the algorithm optimization strategies based on ARM command sets, the G.729 codec is optimized and improve the move speed of the system.

Key words: ARM; embedded Linux; G.729; speech processing

随着互联网的发展,基于网络的多媒体通信越来越引起人们的关注。多媒体通信的基础是语音通信,为此国际电信联盟电信组(ITU-T)创立了 G.711、G.723、G.729 等多个语音编码的标准,其中 G.729 以其较低的编解码复杂度、较高的语音质量和很低的编解码延时获得了人们的青睐。G.729 采用的是共轭结构的代数码激励线性预测算法 CS-ACELP (Conjugate Structure Algebraic Code Excited Linear Prediction),这是一种基于 CELP 编码模型的算法。由于 G.729 编码器能够实现高语音质量(MOS 分 4.1)和低算法延时,所以被广泛应用于 IP 电话、移动通信、多媒体网络等领域。

1 系统介绍

本文所研究的嵌入式语音终端要求能够实现点对点的实时语音通信,同时具有占用系统资源少、功耗低等特点。为了满足设计要求,系统在基于 ARM9 内核硬件体系构架及嵌入式 Linux 操作系统平台上移植了 G.729 编解码算法。应用 Linux 操作系统多进程与多线程编程相结合的方法以及管道通信方式来提高系统运行效率。通过 Linux 多进程编程机制实现编码模块和解码模块的同步运行,在编解码模块内采用 Linux 多线程编程机制实现了关键数据缓冲区的保护和共享、调度网络传输、音频数据处理等功能的系统资源分配。

2 系统硬件构架

本系统采用 Mini2440 开发板作为算法平台,Mini2440 是一款基于 ARM9 内核的开发板,它采用 Samsung S3C2440 作为微处理器,并采用专业稳定的 CPU 内核电源芯片和复位芯片来保证系统运行时的稳定性,最高主频达到 533 MHz。系统的硬件平台结构如图 1 所示。模拟音频信号通过音频接口 UDA1341 转换成数字音频信号后采用 DMA 方式交由 S3C2440 微处理器编码,处理后的数据通过网络接口与远端嵌入式语音终端进行语音交互,也可以与计算机网络的语音终端进行语音交互。系统使用触摸屏为人机交互介质,基于 qtopia 软件编写用户使用界面,控制软件终端的运行。

3 系统软件设计

Mini2440 开发板已经移植了基于 Linux-2.6.32 内核的嵌入式 Linux 操作系统,Linux 操作系统能够方便地构建交互性好、运行稳定、可扩展性强的专用通信设备。系

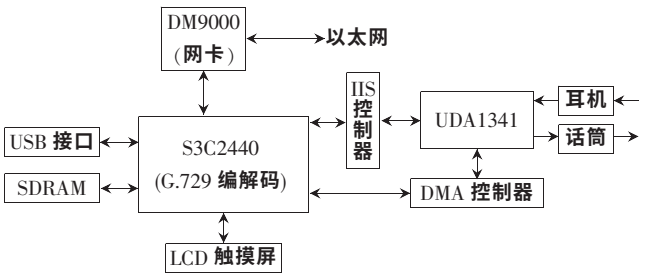

图 1 系统硬件构架

统软件设计主要分成三部分:(1)对源代码进行去串行化操作,以便适应于远程网络传输;(2)是对算法进行优化,提高运行效率;(3)进行系统软件设计以及优化,这是最重要的一步。系统软件设计采用服务器—客户端的模式,应用 Linux 多进程与多线程相结合的编程方法实现。软件主要由两个进程组成,即采样—编码—发送进程和接收—解码—播放进程,系统通过这两个进程实现点对点实时语音通信。在 Linux 环境下的项目开发中,一般使用 GNU make 工具来维护程序模块关系和生成可执行文件,对 G.729 算法进行移植。由于采用多线程的编程方法,所以在 Makefile 文件中需要加入对线程库的支持。

3.1 源代码去串行化

G.729 采用的是共轭结构的代数码激励线性预测算法,这是一种基于 CELP 编码模型的算法。编码器对 10 ms 长的语音帧进行处理,逐帧地提取 CELP 模型参数(LP 滤波器系数、自适应码本和固定码本索引和增益),然后对这些参数进行编码和传输。在解码器端,这些参数用来恢复激励信号和合成滤波器参数,重建语音是使激励信号通过线性预测(LP)滤波器滤波后得到的。

ITU-T 发布的 G.729 语音编码器的源代码在编码完毕后进行了“串行化”的操作,使得经过 G.729 算法处理后的数据流文件看起来比处理前的语音数据更大,这对语音信号远程网络传输并没有任何作用,因此不能直接应用于工程项目。ITU-T 为了在标准化方针中进行丢帧隐藏测试,对语音编解码器参考软件的码流格式一般要求为 ITU-T G.192 中规定的格式,即用 16 位的 0x007F 表示一个比特“0”,用 0x0081 表示一个比特“1”,每个帧头会有同步字和包的长度。对于同步字,0x6B20 表示该帧为坏帧,0x6B21 表示该帧为好帧,这导致了编码后数据的增大。

解决上述问题的方法是去掉串行化代码。在 BITS.C 文件中定义了 4 个函数:

```
static void bit2byte(Word16 para,int bitlen,unsigned char*bits,
int bitpos);
static Word16 byte2bit(int bitlen,unsigned char *bits,int
bitpos);
void prm2bits_ld8k(int prm[],INT16 bits[]);
void bits2prm_ld8k(INT16 bits[],int prm[]);
```

这个文件就是编码后流文件大小没有变化的关键所在,对这 4 个函数进行改写就可以得到压缩后的流文件,这样标准的 ITU-T 的 G.729 语音编码器和解码器就基本可以达到 1:16 的编码效率。

3.2 程序设计方案

系统软件设计采用服务器/客户端的模式,结合 Linux 多线程的编程方法,使整个系统能够流畅地实现多用户的接入和切换。在系统中使用 IP 地址作为服务器的编号,用来区分来自于不同用户的呼叫,软件启动后以服务器的形式在内存中运行,等待用户输入或者远程用户的接入,通信结束后重新返回此处等待。软件主要由两个进程组成,在进程之间采用 Linux 管道技术实现通信和数据传递,在线程之间采用互斥锁的机制对关键数据进行保护和实现线程之间的同步,以保证系统的平稳、流畅运行。系统软件流程如图 2 所示。在主进程采样—编码—发送进程中创建接收—解码—播放子进程,然后交由 Linux 操作系统进行调度。在两个进程交互执行的过程中,采用 Linux 管道通信的机制进行数据传递,同时使用 Linux 信号(signal)实现进行间的同步。如在语音通信中子进程接收—解码—播放进程与远端语音终端网络连接好以后将会发送一个同步信号给正在等待的主进程。

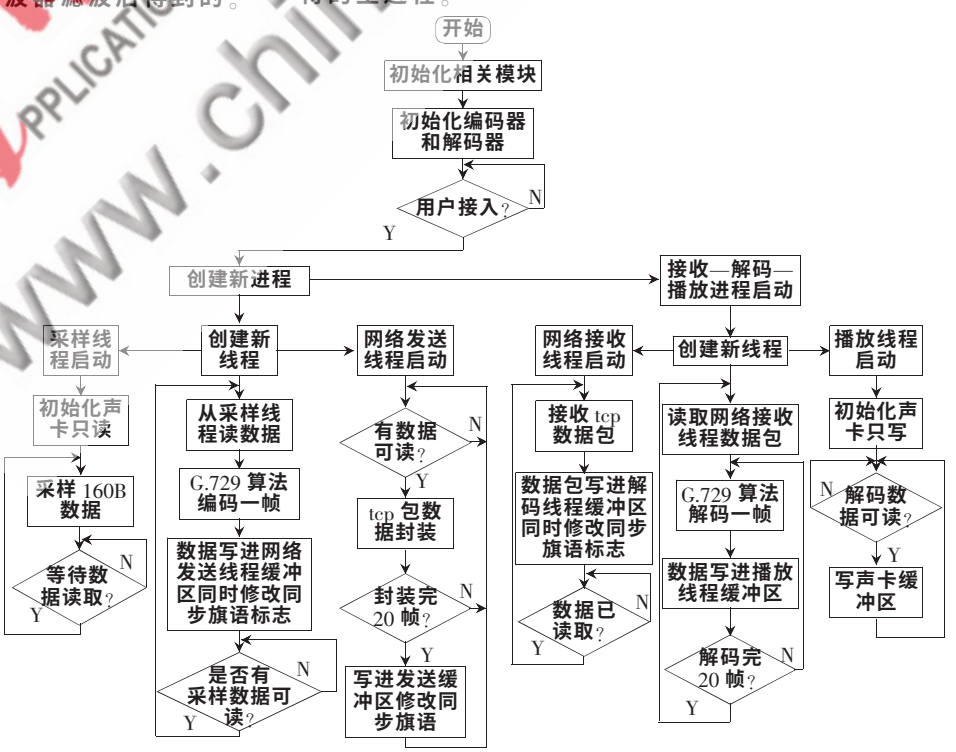

图 2 软件流程图

```
……
int  client_ready = 0;
……
//进程间同步
client_fd=accept(server_fd,(struct sockaddr*)&client_addr,
        &sin_size);
printf("accept a new client\n");
client_ready = 1;
mysigval.sival_int=real;
sigqueue(server_pid, SIGTERM, mysigval);
……
```

使用多线程编程方法实现进程中不同任务的同步运行,在主进程采样—编码—发送进程中,首先对编码器进行初始化,然后创建采样和网络发送线程。

```
……
//创建子线程
if(pthread_create(&thread_s,NULL,speechSample,NULL)) {
    printf("pthread_create failed.\n");
    exit(1);
}
if(pthread_create(&thread_s, NULL, streamServer, NULL)) {
    printf("pthread_create failed.\n");
    exit(1);
 }
……
```

为了降低网络发送线程的运行次数,在程序运行过程中对每20帧编码后的数据进行tcp包封装后发送,这样既提高了软件执行效率,又降低了程序的复杂度。由编码主线程维护两个缓冲区buf1和buf2,buf1缓冲区存放等待发送的20帧数据,buf2缓冲区存放编码线程正在对其进行操作的后20帧数据,使用流水线方式,循环地把buf2的数据拷贝至buf1。在数据拷贝时采用线程互斥锁对关键数据进行保护,对互斥锁静态赋值后使得在同一时刻仅有一个线程对这一部分关键数据进行操作。这样避免了线程间竞争导致数据丢失,同时定义了data_move_ready旗语可以实现线程之间同步。

```
……
data_move_ready=0;
……
//线程间同步
pthread_mutex_unlock(&mutex);
data_move_ready=1;
memmove(buf1, buf2, buf_count);
pthread_mutex_unlock(&mutex);
……
```

经过以上设计,使得网络数据的采集传输和编解码算法之间的依赖性降低到最低,保证了数据在系统中的流畅性。软件运行过程中,子进程接收—解码—播放进程的执行机制与主进程采样—编码—发送进程的执行机制一样,是主进程的一个逆过程。

4 编解码器的优化

G.729算法虽然有8 Kb/s的编码速率,是编码速率和合成语音质量综合效率最优的压缩算法之一,但是直接在ARM上运行的效率却相当低,因此需要对算法进行优化,提高编解码效率。编解码器的优化主要采用代码优化的方法,针对算法运算强度最大环节或函数,应用ARM指令集将该环节或函数重载,以实现对算法的优化。其次采用汇编语言的目的就是尽量避免ARM处理器流水线上的冲突,充分利用流水线的功能。

根据语音编码的特点,编解码的函数都是由一些基本的加减乘除的简单函数组织而成,G.729编码器的运算工作主要集中在LSP矢量量化、自适应码本搜索和固定码本搜索环节。通过对算法具体分析可知,算法运算量主要集中在L_mac()、L_mult()、L_add()、Sature()及L_sub()这几个函数。对这些函数进行优化时可以将其定义为内联函数,当该类函数被调用时,编译器自动在目标代码段展开该类函数,省去频繁调用函数的开销。G.729算法中包含大量的char和short类型变量,而32位定点ARM编译器在每次存储char和short类型变量时需要额外操作,如果将char和short类型局部变量改为int和unsigned int类型则会大大降低算法的运算量。大量的if语句判断增加了系统中跳转指令,影响了流水线的流畅性,所以尽量减少跳转指令的使用,通过填入其他非相关指令实现合理利用流水线的目的。

本文提出采用Linux多进程与多线程相结合的设计方案,并根据ARM处理器的特点,进行了系统性能的优化。系统延时为一帧数据处理时间和20帧数据Tcp封装时间,即10 ms+20×10 ms=210 ms,在算法处理过程中没有数据堆积,语音处理结果完全达到了预期效果。

参考文献

[1] 姚天任.数字语音处理[M].武汉:华中科技大学出版社,2007.
[2] ITU-T. Recommendation G. 729-Anneb B A silence scheme for G. 729 optimized for terminals conforming to Recommendation V7.0[S]. Geneva: ITU, 1994.
[3] WU Chienhsuan, Huang Chinyu, Chang Junru. Applation-specific RISC achitecture for ITU-T G.729 decoding processing[M]. IEEE, 2006.
[4] 杜春雷.ARM体系结构与编程[M].北京:清华大学出版社,2003.
[5] 吴海涛.G.729语音编码算法实现方法研究及DSP实现[J].哈尔滨理工大学学报,2005(6).

(收稿日期:2010-09-07)

作者简介:
李雄飞,男,1984年生,硕士研究生,主要研究方向:语音信号处理。
黄冰,男,1946年生,教授,主要研究方向:语音信号处理和光通信。