基于DSP的EVRC声码器实现与软件优化技术

李　彦[1]，卢　虎[2]
(1.空军工程大学 电讯工程学院，陕西 西安 710077;2. 西北工业大学 电子信息学院，陕西 西安 710072)

摘　要:针对定点EVRC声码器的DSP实现,在充分理解TMS320C64xx CPU结构的基础上,对语音信号处理中大量出现的循环运算,进行软件深度优化,极大提高了CPU的利用率和CPU功能单元的并行处理程度,并在实际应用中得到满意的效果。

关键词:声码器; EVRC; DSP软件流水线
中图分类号:TN912.3　　**文献标识码**:A　　**文章编号**:1009-3516(2004)05-0057-04

TIA于1997年正式推出IS-127标准,它是IS-95的可选语音服务标准(Service Option 3)和第三代移动通信系统CDMA2000的语音编解码标准,此标准的核心是增强型可变速率语音编码器(Enhanced Variable Rate Codec,简称EVRC)[1-2]。就综合语音质量和编码速率,本文主要研究EVRC声码器的实现与优化,首先实现了EVRC声码器的功能,然后重点研究EVRC声码器的软件优化,使得此声码器具有较高的性价比。

1　EVRC声码器算法概述

EVRC的核心算法是基于贝尔实验室在1994年提出的RCELP(RelaxedCode-Excited Linear Prediction)算法，RCELP是一种广义的CELP语音编码算法。在传统CELP的合成分析法中,总是试图让合成的语音信号逼近原始信号,并由此选择最佳的激励;而EVRC逼近的是经过时域变形(Time-Warping)的残差信号。这个经过时域变形的残差信号包含了简化的基音信息,这些基音信息是在每帧中开环计算后进行线性插值得到的。这样会使得计算比较复杂,但它却使得在每帧的数据包中基音的比例减小,因此改进语音自然度的激励信号将得到更多的比特数,从而改善语音质量。

EVRC采用线性PCM信号(采样频率为8 kHz,16 bit量化)作为输入信号,以帧(20 ms、160个样点)为单位进行编解码。作为一种可变码率编码器,它能根据输入信号的特点来选择编码速率。这种特点不仅仅指语音信号的波形质量,还包括基音和相邻帧的信号样点。对每一帧信号,最后的编码速率为:码率1(8 kbps,171 bit/包)、码率1/2(4 kbps,80 bit/包)或码率1/8(1 kbps,16 bit/包)。正是由于码率可变,使得它的平均码率低于8 kbps[3]。

2　EVRC声码器的硬件和中断设计

这里充分利用TMS320C64xx系列DSPs的强大处理能力来设计多通道声码器,其硬件结构如图1所示。

编码时,码率为64 kbps的μ率PCM码流,通过TMS320C64xx的McBSP(多通道缓冲串行口)来接收,再转换成线性PCM码流,然后由DSP将其编码成最大码率为8 kbps的码包(Packet),最后通过HPI(主机口)发送出去;解码时,最大码率为8 kbps的码包,通过TMS320C64xx的主机口接收,经过DSP解码为线性PCM码流,再转换成64 kbps的μ率PCM码流,再通过McBSP发送出去。其中用到了3个中断,第一个是定时器中断,为每一个编码/解码通道平均分配时间;第二个是EDMA中断,用来标识McBSP口接收/发送1

收稿日期:2004-05-03
作者简介:李　彦(1963-),男,北京市人,副教授,主要从事信号与信息处理研究.

帧 PCM 数据的结束;第三个是 HPI 中断,用来标识 HPI 口接收/发送 1 个码包的结束。限于篇幅,我们只分析定时器中断。

由于是多通道声码器,所以需要给每一个通道平均的分配资源,这是通过定时器中断来实现的。设计中,采用定时器 0 来实现。对语音信号的处理是以帧为单位,每一帧有 160 个样点(8 kHz 采样,长度为 20 ms),假设 1 块 DSP 芯片上要实现 N 路声码器,则将 20 ms 分成 N 份,每一路语音的处理时间为 $(\frac{20}{N}$ ms。启动内部定时器 0,使其开始计时,并打开定时器中断。在中断服务程序中,首先将编码通道号加 1 作为当前的编解码通道号,并判断是否为最后一个通道,如果是最后一个通道,则关闭定时器中断,停止计时。然后设置当前通道编码时间的标志,启动该通道的编码。再从 EDMA 接收缓冲区中读取前向 PCM 数据以供编码,从 HPI 接收缓冲区中读取当前通道的反向语音包,为该通道的解码做准备,并设置相应标志;将前一通道的反向 PCM 数据写入 EDMA 发送缓冲区,将前向语音包写入 HPI 发送缓冲区,启动定时器,当一路语音的处理时间$\frac{20}{N}$ ms 到达时,触发中断,进入中断服务程序,首先将通道号加 1,判断是否为最后一个通道,如果是,则关闭定时器中断并停止计时。定时器中断中,根据通道号来设定通道状态,并重新启动定时器。在最后一个通道的定时器中断中关闭该中断,在 20 ms 中断中打开,防止 20 ms 中断和定时器中断同时到来。

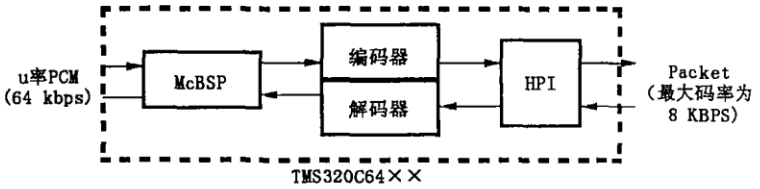

图 1 EVRC 声码器编码/解码示意图

3 软件设计与优化

图 2 和图 3 分别为 EVRC 声码器编码/解码软件程序流程图,各部分功能定义明确,对比分析协议即可具体实现代码[3]。

下面,我们主要讨论 DSP 程序设计的瓶颈——如何保证代码的实时性。大多数 DSP 程序由于无法满足该点,即使可以实现某种具体功能,也完全无法实际应用。下面就针对 DSP 软件的优化问题和注意事项作具体论述。

DSP 软件编程及优化流程大致可分为以下 3 个阶段:

第一阶段:脱离硬件,完全根据任务编写 C 语言程序;尔后,编译产生可在 C6000 内运行的代码,证明其功能正确;然后再用 CCS 的调试工具,如 Debug 和 Profiler 等,分析确定代码中可能存在的、影响性能的低效率程序段,为提高代码效率,进入第二阶段。

第二阶段:利用 C 语言的各种优化手段改进 C 语言程序,然后再检查所产生的代码的性能,如果性能仍不能达到所期望的效率,则进入第三阶段。

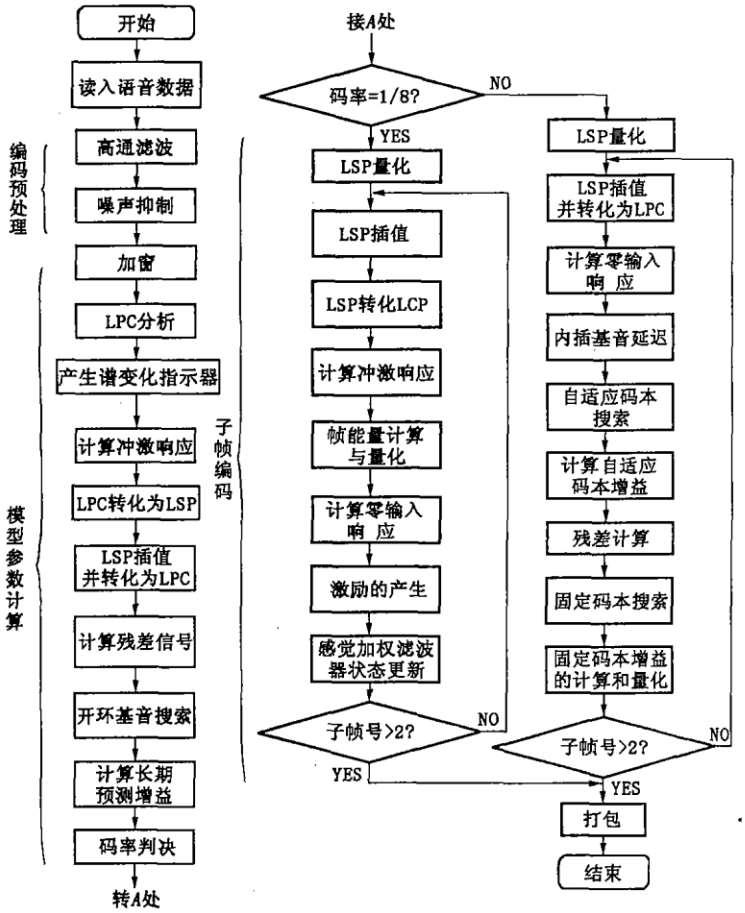

图 2 编码器软件流程图

第三阶段:从 C 语言程序中找出对性能影响较大的程序段,使用线性汇编重新编写这段程序,使用汇编优化器优化这段代码。直到性能符合要求为止。在开发 EVRC 声码器的过程中,关键是对一些重复调用且

耗时较多的模块进行优化,这些模块是否优化对程序的性能有很大的影响。

3.1　C语言级的优化

1)使用内联函数

C6000编译器提供了许多内联函数,可快速优化C语言代码。内联函数是直接与C6000汇编指令映射的在线函数,不易用C语言实现其功能的汇编指令都有对应的内联函数。每一个内联函数完成的功能与对应的汇编指令相同。如程序段:iSum0 = _sadd(iSum0, 0x800)。该程序段中,函数_sadd( )称为内联函数,和汇编指令sadd对应,它的功能是实现两个32 bit数据的饱和加法。

2)数据打包处理

C6000访问存储器很费时,为了提高C6000的数据处理率,应使1条Load/Store指令能访问多个数据。对C64xx来说,如需对一连串整型数据进行操作时,可以用双字长(64 bit)访问存储器。这种类型的优化,称为数据打包处理。如下程序段的功能是从存储器中一次读入64 bit的数据:

dFirMemory_3_2_1_0 = _amemd8(sFirMemoryBuffer + 2 * i)。

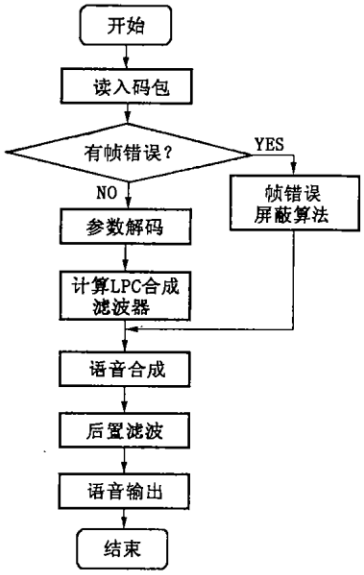

图3　解码器软件流程图

3)消除存储器相关性

为了使代码达到最大效率,C6000编译器会尽可能把指令安排为并行执行。为使指令并行操作,编译器必须确定指令间的相关性(如果一条指令必须发生在另一条指令之后,则它们是相关的),只有不相关的指令才可以并行执行。可以使用关键字restrict来表明一个指针是指向一个特定对象的指针。如:

```
void fun(short * restrict s_output,
short * restrict s_input,
short * restrict s_coef,
short * restrict s_memory)
{
程序体;
}
```

关键字"restric"表明指针 * s_output,与 * s_input、* s_coef、* memory是独立的, * s_output不会指向其它存储区,因而对各数据可并行读和写。程序编译后的代码效率极高。

4)软件流水

软件流水是一种用于安排循环内的指令执行方式,使循环的多次迭代能够并行执行的一种技术。在C6000的C语言编译器里,采用软件流水来优化程序代码是一项核心技术。在编译C语言程序时,使用-o2或-o3选项,编译器就能从程序中收集信息,尝试对程序循环实现软件流水。在用软件流水优化程序时,需注意以下几个方面的问题:

a)循环次数

循环次数指程序内循环执行的次数,软件流水结构都有一个最小安全循环迭代次数的要求,以保证用软件流水来执行循环程序的正确性。若编译器能确定循环迭代至少执行n次,n就是已知的最小循环迭代次数。这可以由程序中的Pragma Directive提供,如以下指令告诉编译器其后的循环至少执行的次数为20次,最多执行30次,且执行次数是2的倍数:

```
#pragma MUST_ITERATE(20,30,2);
```

b)循环展开

循环展开是在程序里把小循环的迭代展开,使得可能并行的指令数增加,从而改进软件流水编排,改善代码性能。

c)推测执行

在对循环实行软件流水时,循环填充(PIPED LOOP PROLOG)帮助建立流水,循环排空(PIPED LOOP EPILOG)使循环结束流水。为了提高代码性能,减小代码尺寸,有必要消除填充与排空。这种优化的原理是

把 Prolog 与 Epilog 部分全部编入到软件流水内核,这样既可以减小代码尺寸,又可以减小对软件流水安全执行的最小次数的要求。

3.2 汇编语言级的优化

当对 C 语言的优化不能满足需要时,就必须从 C 语言代码中抽出影响速度的关键部分,用线性汇编语言来改写这部分代码,并用线性汇编优化器来优化这些代码。线性汇编语言与 C6000 的汇编语言类似,两者有相同的指令集,但线性汇编语言可以不包含有关指令延迟间隙和寄存器使用的信息,这样做的目的是让汇编优化器来决定这些内容,以降低汇编难度,缩短软件开发时间。

在编写线性汇编代码的时候要注意以下问题:首先,线性汇编代码必须包含必要的伪指令,如".cproc"和".endproc"来限定线性汇编代码,另外还有一些伪指令,可以向优化器提供优化所需的信息,以便更好的优化程序。其次,线性汇编代码必须遵守一定的语法规则。最后,为了避免寄存器冲突。在进一步优化时可以指定处理单元。一般情况下,线性汇编代码的效率可达到汇编代码的 90% ~100%,在绝大数情况下,已经能满足用户需要,如果还不能满足需要,就只能对汇编代码进行手动优化。

4 小结

本文在深入理解 C64xx 的内核结构基础上,对 EVRC 声码器程序进行了较深的优化,如使用循环展开、数据打包处理和软件流水线等优化手段,使得程序高度并行,运算的速度大大加快,但也有一定的副作用,就是程序空间增大了。但是,对于内部 RAM 为 1 MByt 的 TMS320C64xx 芯片来说,该影响比起软件执行效率的显著提高,几乎可以不考虑。

参考文献:

[1] 刘湘雯,张 敏,赵世廉. ITU-T G.728 语音压缩算法的实时实现[J]. 空军工程大学学报(自然科学版),2002,3(2):53-55.

[2] 王尚武. 语音压缩中的线性预测编码技术[J]. 微机发展,2002,(6):40-42.

[3] 丁 琦,王炳锡. EVRC 语音编码算法的研究和实现[J]. 信息工程大学学报,2003,4(3):57-60.

[4] 王炳锡. 语音编码[M]. 西安:西安电子科技大学出版社,2002.

[5] TIA/EIA/IS-127. Enhanced Varialbe Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems[S].

[6] TIA/EIA/IS-718. Minimum Performance Specification for The Enhanced Varible Rate Codec, Speech Service Option 3 for Spread Spectrum Digital Systems[S].

(编辑:门向生)

The Implementation of EVRC Vocoder and Software Optimization Based on DSP

LI Yan[1], LU Hu[2]

(1. The Telecommunication Engineering Institute, Air Force Engineering University, Xi'an, Shaanxi 710077, China; 2. Electronic Information College, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China)

Abstract: This paper makes great efforts in the software optimization of EVRC vocoder. Based on the understanding of TMS320C64xx CPU structure, a deep optimization of software on the loop operations which appear frequently in voice signal processing is done. This greatly improves the utilization ratio of CPU and the parallel processing degree of CPU function cell. And the vocoder works well in practical application.

Key words: vocoder; EVRC; DSP software pipelining