AMR-WB 编码中线谱频率量化的 DSP 优化与实现①

Implementation and Optimization of the Line Spectrum Frequency Quantization Based on DSP

刘祥明　王　玲　(湖南大学 电气与信息工程学院 长沙 410082)

摘　要： 语音编码算法的实时实现一直是研究的重要课题。AMR-WB 是一种高品质的宽带语音编码标准，但计算复杂度颇高。在研究 AMR-WB 语音编码算法优化过程中，从改变搜索方法和码书结构研究了 AMR-WB 编码中线谱频率量化的优化，并阐述了基于 TMS320C64X 的 C 语言和线性汇编混合编程的接口标准，给出了线谱频率量化的线性汇编实现的应用实例。实验结果表明，运算时间得到了很大的改善，且语音质量没受影响。

关键词： 线性汇编 AMR-WB S-MSVQ 线谱频谱

AMR-WB(Adaptive Multi-Rate Wideband Speech Codec，宽带自适应多速率语音编码)是由 3GPP/ETIS 提出的新一代宽带语音编码标准，也是第一次被无线通信和有线通信同时选定的语音编码标准，在移动通信、IP 电话、多媒体通信等领域有很好市场前景。对它的优化研究均发现，AMR-WB 编码的算法复杂度高，线性预测模块、线谱频率量化模块、自适应搜索模块和固定码本搜索模块占了编码时间的 60%，实现对这 4 个模块的优化是整个编码优化的关键[1,2]。其中线谱频率量化模块的计算复杂度占 9%~13%，本文着重讨论了线谱频率量化模块，从改变搜索方法和码书结构对线谱频率量化模块做优化研究。

TMS320C64X 系列 DSP 是 TI(德州仪器)公司推出的高性能的定点数字信号处理芯片，本文结合它的硬件特点，研究了使用 C 和线性汇编混合编程在优化算法实现中的应用。

1　AMR-WB编码中频谱参数量化的原理

线谱频率(Line Spectrum Frequency, LSF)因具有很好的内插特性和量化特性，被广泛应用于语音编码和语音识别等领域。在 AMR-WB 编码中，先把每帧语音 16 个 LSP(Immittance Spectral Pair)参数通过(1)式转换为 ISF 参数。

$$f_i=\begin{cases}\frac{f_s}{2\pi}\,ar\cos(q_i), i=0,\ldots,14\\ \frac{f_s}{4\pi}\,ar\cos(q_i), \quad i=15\end{cases} \tag{1}$$

式中，f_i 是 ISF，f_s=12.8KHZ 采样频率，q_i 为余弦域中的 ISP 参数,则 ISF 系数的矢量可表示为 f^t= [$f_0 f_1 \cdots f_{15}$]，t 表示转置

其次，求出当前帧去掉均值后的 ISF 矢量 Z(n)：

$$Z(n)=\left| f-\frac{1}{16}\sum_{0}^{15} f_i \right| ,\ i=0,\cdots,15 \tag{2}$$

然后，用一阶滑动平均预测法求出当前的 ISF 预测残差矢量 r(n)：

$$r(n)= Z(n) - p(n) \tag{3}$$

其中，$p(n)=(1/3)\hat{r}(n)$ 是当前帧的 ISF 矢量的预测值，$\hat{r}(n-1)$为上一帧量化残差矢量。r(n)的量化是采用 S-MSVQ(Split-Multistage Vector Quantization, 分裂多级矢量量化)系统，它结合了分裂矢量量

① 收稿时间:2008-12-31

化和多级矢量量化的特点。MSVQ 系统在 AMR-WB 中量化过程如图 1 所示。LSF 参数的量化分两阶段进行，第一阶段分裂成 9 维和 7 维的两个子矢量以 8bits 进行量化，按欧氏距离公式(4)作为失真测度，即：

$$E = \sum_{i=m}^{n} \left[r_i - \hat{r}_i^{\,k} \right]^2 \qquad (4)$$

式中 E 加权误差，m 和 n 是第一个和最后一个子矢量，r_i 是残差子矢量，$\hat{r}_i^{\,k}$ 是索引值为 k 的矢量量化值。

采用公式(4)(即全搜索法)找出最佳码字，然后计算输入 ISF 参数于该码字的失真，作为第二级输入。第二阶段又把 9 维子矢量再分成 3 个 3 维的子矢量分别以 6，7，7bits 进行量化，把 7 维子矢量分一个 3 维和一个 4 维的子矢量都以 5bits 进行量化，采用第一阶段同样的方法寻找最佳码字。

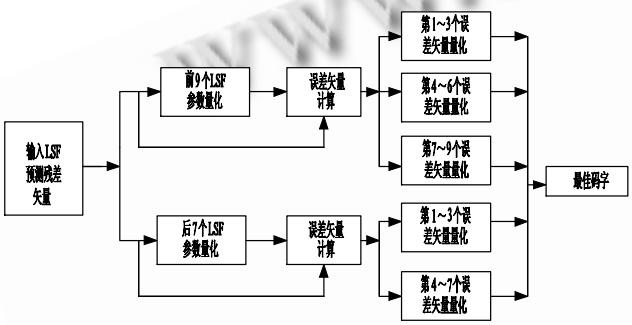


图 1　AMR—WB 中 S-MSVQ 量化过程图

2　改进方案

矢量量化计算复杂度常以乘法的次数来衡量，如何减少乘法的运算次数也就成为了矢量算法设计的研究重点。对于矢量量化的算法的改进一般有两种方法：i)改进码本结构，ii)改进搜索策略。在上述 S-MSVQ 系统中，其矢量失真测度是基于欧氏距离平方，码字搜索采用全搜索法。通过该算法实现研究发现，以下两方面有待改进：①第一阶段分裂成 9 维和 7 维子矢量，相对第二阶段的子矢量，其维数比较高，采用全搜索法计算量比较大。②第二阶段子维数多为奇数维，对于一些支持字读取的硬件而言，将是造成程序运行效率不高的瓶颈。

结合 TMS320C64X 的硬件特点，对以上两方面分别做了改进：对于第(1)方面，本文采用了一种快速的码字搜索方法[4]，把待量化矢量和码字分割成的两个子矢量，它依据准则 1 和准则 2 达到快速搜索的目的，且不影响量化效果。

准则 1．对于待量化的 K 维矢量 X 和码字 Y，若 $\sum_{i=1}^{2} (\sqrt{k_i} m_{Xi} - \sqrt{k_i} m_{Yi})^2 \geq D_{\min}$ 则码字 Y 可以排除。

准则 2．对于待量化的 K 维矢量 X 和码字 Y，若 $\sum_{i=1}^{2} \{(\sqrt{k_i} m_{Xi} - \sqrt{k_i} m_{Yi})^2 + (V_{Xi} - V_{Yi})^2\} \geq D_{\min}$ 则码字 Y 可以排除。

(其中 K_1，K_2 为子矢量的维数 且 $K_1+K_2=K$，m_{Xi}，m_{Yi} 为子矢量的均值，V_{Xi}，V_{Yi} 为子矢量的方差，$D\min = \sum_{i=1}^{k} (X_i - Y_i)^2$)

按照准则 1 和准则 2，本文改进的具体实现步骤如下：

① 把 9 维码字分割成 $K_1=4$ 维和 $K_2=5$ 维两个子矢量，把 7 维码字分割成 $K_1=4$ 维和 $K_2=3$ 维的两个子矢量。计算每个码字的 4 个参数，即 $\sqrt{k}_{1mY1}$、$\sqrt{k}_{2mY2}$、V_{Y1}、V_{Y2}，并事先存储。

② 计算待量化矢量的 4 个参数，$\sqrt{k}_{1mX1}$、$\sqrt{k}_{2mX2}$、V_{X1}、V_{X2} 。

③ 选择第一个码字为初始码字，计算 Dmin；

④ 选取下个码字参数，按准则 1 进行码字排除，若排除，则跳⑦；

⑤ 按准则 2 进行码字排除，若排除，则跳⑦；

⑥ 若不能排除，则计算该码字与待量化矢量的 Dmin，并与原来的 Dmin 比较，如果小于原来的 Dmin，则更新 Dmin；

⑦ 若全部码字搜索完，则跳⑧，否则，跳④执行；

⑧ 搜索结束；

与文献[4]中的搜索算法框架相比，少了一个码字参数，少了一条 ENNS(equal-average nearest neighbor search，等均值最近邻搜索算法)判决准则。这是基于 TMS320C64X 硬件特点考虑，如能对码字或码字参数都能按字或双字进行读取的方式，则能实现程序高效运行。4 个码字参数正好按 8 字节对齐存储，同时准则 1 和准则 2 排除能力比 ENNS 判决准则强，因此并不影响量化效果。只是搜索时间从理论上会比文献[4]中略长点。但相对全搜索法则能提高很多倍。

对于第②方面，主要是通过加 0 把 3 维码书扩为 4 维，相应的待量化矢量在量化之前行加 0 扩展为 4

维。这样主要是为了实现对码字实现按字或双字读取方式，便于实现软件流水操作，降低运行时间。

3　线性汇编及在优化中的应用

为了是程序代码的执行具有尽可能高的执行效率，往往需要考虑使用混合编程来实现。下面着重讨论了线性汇编代码的编程。

3.1 线性汇编接口标准

线性汇编是一种跟汇编相似的语言，但它并不要象汇编那样需指明并行指令，指令延时以及寄存器的具体使用情况。这些都可以交给汇编优化器完成，因而大大减轻了编程的难度。

线性汇编程序是 C 语言能够直接调用的函数，它符合 C/C++的函数调用规则和寄存器使用规则[5]。线性汇编函数的接口语法如下：

```
.global  _Lable    //用.global 声明的函数名 C 可调用
.global  _Var, …  //在线性汇编中要用到的全局变量
_Lable .cproc [varible1 [,variable2, …] ]
          //函数的形式参数个数，
   …                  //函数体
.endproc     //线性汇编结束
```

按以上接口编写的函数，.cproc/.endproc 之间的代码能被汇编优化器当成 C/C++可调用的函数进行优化编译。其编译出的代码比直接用 C 写的效率高。

对于 varible1(可选的函数参数)必须注意以下几点：

如果设置为机器寄存器名，在变量表的位置必须与 C 中的变量传递规则一致。即前 10 个参数必须依次存入寄存器 A4,B4,A6,B6,A8,B8,A10,B10,A12,B12。

对于高于 32 位字长的参数类型，要用寄存器对来表示。如对 long 型为第一个参数，可用 A5:A4 表示或用变量名 var1:var2。

鉴于上面两点，在编写线性汇编时，尽量不直接使用机器寄存器名，可使用变量名直接代替，这样就可以即可防止破坏编译环境，也使程序可读性增强。

3.2 基于 TMS320C64X 的线谱频率量化的混合编程实例

TMS320C64X 具有很强的并行处理能力，且支持数据打包处理。如能利用编程语言实现数据打包处理，程序的执行效率将达到最大化。由 3GPP 提供的标准 AMR-WB 的 C 源代码中，实现线谱频率量化模块的函数有 VQ_stage1()和 Sub_VQ()。本文以实现第二阶段量化的 Sub_VQ()为例，其的 C 语言函数如下：

```
Word16 Sub_VQ( Word16 *x, Word16 *dico,
Word16 dim,Word16 dico_size,
          Word16 *distance   )
{   ……
      p_dico=dico;
    for( i=0;i<dico_size;i++ )
        {   dist=0;
            for( j=0;j<dim; j++ )
            {   temp=x[j]-*p_dico++;
          dist=L_mac(dist,temp,temp);
            }
          }
      ……
}
```

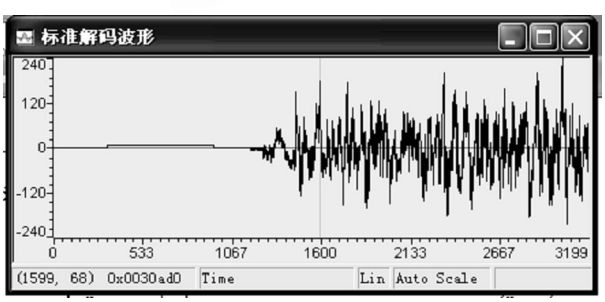
对该函数，首先进行了内循环展开，使用内联函数等基于 C 语言级的优化手段，编译后发现执行代码不能总是按 LDW(字)进行读取，且没有利用数据打包处理技术，代码显得冗长，效率也不高。考虑到其重要性，因而采用线性汇编实现。做为说明 C 与线性汇编混合编程的例子，采用线性汇编实现了这个函数的优化，这里只出了核心代码：

```
.global _Sub_VQ //函数名，由 C 直接调用
_Sub_VQ: .cproc x,dico,dim, dico_size ,
distance//函数参数，注意与 C 语言中函数声明顺序一致
.reg index,index1,temp,temp1
.reg dist,dist1, p_dico, dist_min
.reg y1,y2, contr,if,isf,isf1
//函数体中用到的中间变量但不允许有和.cproc 后面变量有同名的
… …
LDDNW *x,isf:isf1  //读取 x[0]到 x[4]
loop:.trip 64   //循环执行的最少次数，便于优化器实现软件流水
LDDW *p_dico++,y1:y2 //每次读取 2 个字
SUB2 isf,y1,temp //实现两次减操作
DOTP2  temp,temp,dist //实现两次乘 和一次加操作
```

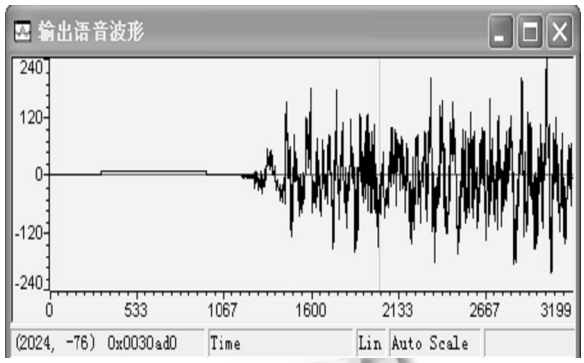
```
SUB2 isf1,y2,temp1
DOTP2 temp1,temp1,dist1
        //以上几步实现了数据打包处理操作
ADD dist1,dist,dist
CMPLT dist,dist_min,if //量化失真比较
[if] MV dist,dist_min
[if] MV index1,index
//用条件寄存器实现 C 语言中 IF~ELSE 语句
ADD index1,1,index1
BDEC loop,contr//实现循环判断和跳转
… …
        .return index //返回最佳码字的索引号对有返回值的函数用.return 伪指令
        .endproc  //返回到函数调用处
```

4 优化结果分析

按照前述的改进方案，通过线性汇编实现代码优化后，把代码移植到标准 AMR-WB 的 C 源代码中，采用了由3GPP为AMR-WB定点算法录制的标准语音测试序列T08.inp，以 14.25Kb/s 速率对优化后语音编码后，再解码和利用它提供的标准编码流 T08_1425.cod 进行解码后波形对比，利用 TI 公司提供的 CCS3.3 软件平台的 View—>Graph 功能观察其结果如图 2 显示了前 10 帧(一帧时间为 20ms,320 个采样点)语音的波形，可以发现语音波形几乎没有发生变化，说明改进后的搜索方法并没有影响合成的语音质量。同时也利用 CCS3.3 软件平台的 profile 工具对优化前后的线谱参数量化部分耗时进行了测试。如表 1 所示，例出了执行该功能的 VQ_stage1()、Sub_VQ() 两个函数优基于 C 语言级优化和线性汇编优化后执行一次的的平均指令周期数。从表 1 可以看出使用线性汇编后 VQ_stage1 的执行时间提高了 8.5 倍，而 Sub_VQ 提高了 12 倍。



(a) 标准解码语音波形



(b) 输出语音波形

图 2　标准解码语音波形与输出语音波形的比较

表 1　优化前后耗时比较

函数名称	优化前执行周期数(cycle)	C语言级优化后执行周期数 (cycle)	线性汇编优化后执行周期数(cycle)
VQ_stage1	25402	7671	2987
Sub_VQ	2125	258	175

5 结语

通过新的搜索方法和改变码书结构，并用线性汇编实现优化之后，AMR-WB 编码中线谱频率模块执行速度得到了极大提高。该方法的不足是增加了小量的存储空间，但对于实时性要求严格的领域，这是值得的。本文的优化经验对于 AMR-WB 在 TI 的 C6000 系列 DSP 上实时实现，有参考价值。

参考文献

1 林奕琳,李巧玲,李江源,等.AMR—WB 语音编码算法及仿真.计算机工程与应用, 2003,29:67－69.

2 Byun KJ, Eo IS, Jeong HB. Real-time Implementation of AMR and AMR-WB Using the fixed-point DSP for WCDMA Systems. IEEE Consumelectronics, 2006, 6:1－6.

3 3GPP.TS 26.171 V6.0.0 AMR Wideband Speech Codec:General description, 2004.

4 赵铭,唐昆,崔惠娟,等.矢量量化快速搜索算法研究.清华大学学报, 2004,44(10),1407－1409.

5 田黎育.TMS320C6000 系列编程工具与指南.北京:清华大学出版社, 2006.