码本搜索在 TMS320C54X DSP 上的优化实现

李章林

（ 南开大学信息学院机器人所，天津 300071）

摘要：码本搜索是 G.723.1、G.729 等语音编码中经常使用的程序结构，其运算量大，历来成为语音编码速度优化的重点。本文不同于其它学者采用改变码本结构的方式，而是在典型的 DSP TMS320C54x 上寻找实现码本搜索的最优的汇编程序结构上进行研究。本文根据 C54x DSP 的特点，提出了 3 种较优的码本搜索汇编程序实现方案，给出了 3 种方法的汇编代码，并从从搜索速度、适用的搜索类型、占用寄存器资源等方面比较了 3 种方法。本文将对在 C54x DSP 上快速实现带有码本搜索结构的编码算法提供有益参考。

关键词：DSP; 汇编; 码本搜索;

Optimization of Code- Book Search On TMS320C54X DSP

Li Zhanglin[1] , Wu Yue[1] ， Lu GuiZhang[1]

(1.Information Technology Department, Nankai University, Tianjin 300071)

ABSTRACT：Code-book searching program structure is often used in speech coding, for example in G.723.1, G.729. As code-book searching consume much calculation resource, it often becomes the key-point to optimize the speed of speech coding. Unlike other researchers, who optimize the code-book searching by reconstruct the code-book structure, we dedicated in finding the most optimum assembly language program structure to implement the code-book searching on A typical DSP device TMS320C54X. Based on the characteristics of C54x DSP, we promoted 3 comparatively optimum assembly language programs for code-book searching, and presented the program code. We also compared the 3 methods in searching speed, searching type and register consumption. The result will be a useful reference for implementing coding with code-book searching on C54x DSP.

Key word: DSP; assembly language; code-book

引言

矢量量化（VQ）技术在各类编码中占有重要的地位。所谓码本就是矢量量化的可能取值的列表，编码和解码方保存相同的码本，所以编码方只需要告诉解码方矢量量化值在码本表中的索引，而不需要传送量化以后的矢量本身，这样大大增加了编码效率。在 G.723.1、G723 等语音编码算法中都使用了该编码方法。

码本搜索的过程是：搜索整个码本表，在码本表中找到和需要量化的矢量（V）最接近（即两者点积最大）的量化矢量(Vj)的索引（j）。从 G.723.1 的实际优化经验来看，码本搜索占用很大的运算量，而且是一种经常出现的程序结构，所以有必要对其优化实现进行研究。

为了提高码本搜索的速度，以前的学者提出改善码本表结构的方法。例如采用分类预选算法[1]将相似的码本归为一类，码本搜索前先确定需要量化的矢量属于哪一类，然后只在该类码本中搜索，从而减少搜索范围；以及局部区域搜索法（PDS 法）等。本文与以前的优化方式不同，本文主要针对码本搜索的一般程序结构，寻找在 DSP 上的最优代码实现方法。

我们这里采用的是在语音压缩中应用广泛的 TI 公司的 C54x DSP，采用 C54xDSP 的汇编来实现码本搜索。为了使得码本搜索程序结构具有简单性而易于研究，且不失通用性，我们暂不考虑搜索过程中是求点积（V·Vj）的最大值，而改为求码本表（Vj）中的最大值。以后只要用 V·Vj 代替 Vj 即可。这样码本搜索一般地可描述为：在含有 SIZE 个数据的数组 array[SIZE]中搜索最大（或最小）值 array[Indx]，并且记录此时对应的索引 Indx。其 C 代码描述如下：

```
Acc1 = 0;
for(i=0; i<SIZE; i++)
{
    Acc0 = array[i];
    if(Acc0 > Acc1) ; 第 4 行
    {
        Indx = i;    ; 第 6 行
         Acc1 = Acc0;
     }
}
```

其中第 4 行有 4 种变化分别为①Acc0 > Acc1（极大值）②

Acc0<Acc1（极小值）③Acc0>=Acc1（最大值）④Acc0<=Acc1（最小值）。一下我们针对以上的程序结构给出 3 种 C54x DSP 汇编的实现方法，并比较 3 种方法的适用范围和效率。

1 实现方法一

以求最大值的索引为例。首先的一个问题是如何判断是最大值？通常可采用 MAX[2]指令。MAX 指令能够取 A 和 B 累加器中较大者放到目的累加器中。MAX 指令根据执行结果设置 C 标志位。设置方法为，如果 A>B，取 A 的值放到目的累加器，C=0；否则，取 B 的值放到目的累加器，C=1。例如我们若采用 MAX B 指令将最大值存放在 B 累加器中，则如果 C=0，则说明找到了一个更大的值。求最小值时，使用 MIN 指令，C 标志位设置方法为，若 A<B 则 C=0，否则 C=1。

然后根据条件存储索引。C54xDSP 的循环计数寄存器 BRC 在每次循环末自动减一，所以根据 BRC 可以用 i=SIZE-1-BRC 推得当前索引 i。利用 SRCCD 指令可以根据条件决定是否存储 BRC 的当前值到 Xmem（Xmem 指用双操作数寻址的 RAM 单元），判断和存储总共只需 1 个指令周期，效率非常高。这里用 SRCCD 存储 BRC 值，循环结束后通过 Xmem 中的值计算最大值的索引 Indx。但是使用 SRCCD 要注意流水线冲突，BRC 的减一操作在循环最后一条指令的解码阶段被执行，这要求 SRCCD 指令到循环末尾指令之间要有 3 个指令周期，否则用 SRCCD 存储的 BRC 是已经减一以后的值。如果在 SRCCD 之后加三个 NOP，则程序效率变低。一个可能想到的办法是不使用 NOP，而用修改计算公式 i=SIZE-1-BRC 为 i=SIZE-1-BRC+1 的方法，因为既然 BRC 是减一以后的值，使用 BRC+1 能恢复到原来的值。但是由于这样使得最后一次循环的 BRC 和倒数最后一次循环的 BRC 值都为 0，所以如果 BRC 为 0 则不知道是哪次循环。所以，这里使用“前向部分循环展开法”[3]，该法能够将循环体内前后语句交换位置，所以能够保证 SRCCD 指令到循环末尾指令之间有 3 个指令周期。

由于 SRCCD 指令不能用 C 标志位作为存储条件，所以这里采用相减的办法产生 SRCCD 需要的判断条件。以求最大值(Acc0>=Acc1)的索引为例，方法一的码本搜索程序如下（其中 AR_array, AR_Indx 等表示辅助寄存器，array 表示数组的首地址）：

```
LD      #0,          B    ;B 中将存放最值
STM     #(SIZE-1),   BRC  ;循环次数
MVDK    array,       AR_array  ;数组头指针
LD      *AR_array+, A    ;加载一个数进行比较
RPTBD   RotationEnd  ;循环黑体表示部分代码
MAX     B             ;A、B 地最大值放到 B 中
SUB     B,   A      ;判断 MAX 指令执行前 A>=B?
SRCCD  *AR_Indx, AEQ; A>=B 时存储 BRC 到
                  ;*AR_Indx 中。离循环末有 3 条指令
LD      *AR_array+, A
MAX     B
RotationEnd:
SUB     B,           A

LD  #(SIZE-1),  A; 用 i=SIZE-1-BRC 推得当前索引 i
SUB *AR_Indx,   A; 最后索引在 A 中
```

该方法，循环内部指令周期数为 4。由于采用“部分循环展开”，最后一次循环多运行了三条指令，这三条指令可能改变 B 的值，这样求得的 B 可能不再是最大或最小值，除非保证 array[SIZE]（array[SIZE]为 array[SIZE-1]后面的数）的值不可能比数组 array 中的其它值大，使得这三条指令不可能改变 B 的值，这可以在 array[SIZE]中存放最小负数来实现。

如果是求最小值，则只要将以上代码的 MAX 指令改为 MIN 指令即可。搜索极大（小）值，无法用该方法实现，因为方法一程序中 SRCCD 的存储条件 AEQ 只说明了新加载到 A 的数据大于等于当前最大（小）值 B，但是并不知道是大于还是等于，所以不能用于搜索极大（小）值。

2 实现方法二

由于方法一不能适用于极大、极小值的搜索，另外为了求得最大或最小值需要额外处理等问题，我们提出方法二。方法二不再有上述问题，但是循环内部指令周期数增为 5。方法二，不再使用相减产生判断条件，而使用 ADDC 指令将判断 C 或者 NC 转化为判断 ANEQ 或者 AEQ。以求最大值(Acc0>=Acc1)为例，码本搜索程序如下：

```
LD      #0,      A        ;A 中将存放最值
ST      #0,      m_Zero   ;m_Zero 将在下面使用
STM     #(SIZE-1),  BRC  ;循环次数
MVDK    array,      AR_array  ;数组头指针
LD      *AR_array+, B    ;加载一个数进行比较
RPTBD   RotationEnd  ;循环黑体表示部分代码
MAX     A            ;最大值放到 A
LD      #0, B        ;B 清零
ADDC    m_Zero, B    ;B==0 表示 C=0，表明 A>B
SRCCD   *AR_Indx,    BNEQ; B 不等于 0 时，即 A<=B
```

```
      ;时存储，即找到更大值。离循环末有 3 条指令
LD      *AR_array+,B
MAX     A
RotationEnd:
LD      #0,         B

LD  #(SIZE-1),  A; 用 i=SIZE-1-BRC 推得当前索引 i
SUB *AR_Indx,   A; 最后索引在 A 中
```

说明如下：当 Acc0>=Acc1 时，也就是 B>=A 时，认为找到一个最大值，此时 MAX A 指令取 B 放到目的累加器，标志位 C=1，于是 ADDC 以后 B 的值为 1，所以使用 BNEQ 作为存储索引的条件。对其他三种搜索类型只需要对程序稍加修改，修改的地方总结如表 1 所示：

表 1 四类码本搜索类型对应方法二程序的变化

搜索类型	累加器的使用	产生判断条件的语句	找到最/极值时 C 标志位的值	SRCCD 的存储条件
最大值：Acc0>=Acc1	A 存放 Acc1 B 存放 Acc0	MAX A	C=1	BNEQ
最小值：Acc0<=Acc1	A 存放 Acc1 B 存放 Acc0	MIN A	C=1	BNEQ
极大值：Acc0>Acc1	B 存放 Acc1 A 存放 Acc0	MAX B	C=0	AEQ
极小值：Acc0<Acc1	B 存放 Acc1 A 存放 Acc0	MIN A	C=0	AEQ

例如对于 Acc0>Acc1 类型搜索，需要将以上程序中的 A、B 调换位置，使用 MAX B 得到极大值和产生判断条件，并且使用 AEQ 作为存储条件。

3 实现方法三

由于方法一和方法二使用 SRCCD 指令，所以一般需要“部分循环展开”。如果使用 XC（条件执行）指令代替 SRCCD 指令则不必使用“部分循环展开”。方法三将索引 i 存放在 AR_i 辅助寄存器中，在循环中自增 AR_i，当找到一个最大（小）值时，用 XC 指令有条件执行 MVMM 指令来存储 AR_i 到 AR_Indx 中。用方法三求最大值（Acc0>=Acc1）的代码如下：

```
LD      #0,         A    ;A 中存放最值
STM     #(SIZE-1),  BRC  ;循环次数
STM     #(-1),      AR_i;索引初始值为-1
RPTBD   RotationEnd     ;循环黑体表示部分的代码
MVDK    array,  AR_array ;数组头指针
LD      *AR_array+, B     ;加载需要比较的数
MAX     A                 ;最值到 A
NOP     ; MAX 指令和 XC 指令之间需要 2 个指令周期
MAR     AR_i+              ;索引增一
XC      1,  C           ;如果 C 执行下面一条指令
RotationEnd:
MVMM    AR_i,   AR_Indx
```

由于 XC 指令要求判断条件至少在 2 个指令周期前产生，所以在 MAX 指令和 XC 指令之间需要插入 2 个周期的指令，这里还剩余一个 NOP 没有被替换。这样它的循环内部指令周期数为 6。该方法的好处是，不用部分循环展开，所以没有部分循环展开最后一次循环额外执行的指令，并且求得的 AR_Indx 直接是索引，不用公式转化。所以在能够替换循环中的 NOP 的情况下（此时循环内部指令可降为 5），可考虑使用方法三代替方法二。

对其他三种搜索类型需要对程序稍加修改，修改的方法见表 1。例如对于 Acc0>Acc1 类型搜索，需要将以上程序中的 A、B 调换位置，使用 MAX B 得到极大值和产生判断条件，找到极大值时 C 标志位为 0，所以用 NC 作为 XC 的执行条件。

4 结论

以上介绍的三种实现方法各有优缺点，在不同的场合可适当选择使用。表 2 为三种方法的比较。

表 2 三种码本搜索方法的比较

方法	搜索类型	循环内部指令周期数	是否求得索引	是否求得最值	部分循环展开额外指令周期数	用公式转化 Indx 的开销	使用辅助寄存器个数
方法一	最大（小）值	4	是	需要额外处理	3	2	2

方法二	最大（小）值、极大（小）值	5	是	是	3	2	2
方法三	最大（小）值、极大（小）值	6（替换 NOP 后为 5）	是	是	0	0	3

从表 2 可知：如果搜索类型为求最大（小）值的码本搜索，那么一般采用方法一，因为它的搜索速度最快。对于搜索类型为极大（小）值的码本搜索，首先尝试采用方法三，如果方法三循环内部的 NOP 能够通过调整语序被替换，那么方法三的循环内部指令周期数将降为 5，将和方法二有相同的指令周期数，而且方法三没有部分循环展开额外指令周期数，此时它比方法二要稍快，反之如果不能替换循环内部 NOP，则采用方法二。另外对于极大（小）值的码本搜索，如果想节省辅助寄存器的使用个数，而不考虑搜索速度，那么应该采用方法二。

本文提出的三种方法在实际的 G.723.1 语音编码算法的优化中已经得到了应用。

参考文献

[1] 一种新地快速码本搜索算法设计，马力波，朱淼良，贵州工业大学学报，第 28 卷 第 1 期，1999 年 2 月

[2] SPRU172:Mnemonic Instruction Set, 美国 TI 公司,http://www.ti.com

[3] VOIP 中 G.723.1 语音编码算法的 DSP 实现，李章林，南开大学通信与信息系统专业硕士论文，2004 年 6 月

作者简介

李章林，男，1979 年出生，正在南开大学信息学院控制理论和控制工程专业攻读博士研究生。主要从事信息自动化、嵌入式系统的研究开发。