TMS320C67x DSP Library 在程序开发中的应用

时间：2010-11-18 19:39:50 来源：国外电子元器件 作者：马克雷

0 引言

美国德州仪器(TI)公司的数字信号处理器(DSP)以其处理速度快，功能强大，易于使用，且有开发软件支撑等优点而广泛应用于通信、电子、自动控制等领 域。一个完整的 DSP 系统应当由硬件和软件两部分组成。在硬件(主要是 DSP 芯片)相同时，系统的性能将主要取决于软件部分的效率。而在相同的硬件平台 上，不同程序员编写的软件效率相差很大，软件效率已成为影响 DSP 系统性能的一个重要因素。要充分发挥 D-SP 芯片的性能，就必须编写高效率的程序。一般 在 DSP 进行运算时，有些操作会频繁出现(如卷积、FFT、FIR 滤波等)，完成这些操作的程序的效率直接影响整个软件部分的效率。为此，TI 公司提供了 一系列库函数(TI DSP Library，以下简称 DSP Lib)来完成这些操作。这些库函数既可减少程序员的工作量，又可提高程序效率，因此，在程序开发中，合理地使用这些库函数，将大大提高系统性能。

1 TI DSP Library 简介

1．1 TI DSP Library 的特点

DSP Lib 的核心实际上是一系列经过手工优化的汇编程序代码，这些代码封装在后缀名为．lib 的文件中，可用于完成各种运算。它们对外是不可见的。这些程序 (库函数，routines)可被 C 程序调用。由于经过了手工优化，它们的效率都非常高。由于不同系列 DSP 芯片的指令集不同，因此，不同系列 DSP 芯片 的 DSP Lib 也是不同的，如 TMS320C5000 的 DSP LIb 就不能用于 TMS320C6000。但是，各个系列 DSP Lib 的基本组成是相同的，一个完整的 DSP Lib 通常由 Lib 文件夹、include 文件夹和其它辅助文件组成。其中 lib 文件夹用于存放*．lib 文件，其内部封装着手工优化的汇编程序代码，是 一个 DSP Lib 的核心部分。有的 DSP Lib 还有*．src 文件，这些*．src 文件主要是用 C 语言和汇编语言编写的程序源代码。使用归档器可从中提取出这些源代码；而 include 文件夹用 于存放各个库函数的头文件，通常这些文件分为 C 程序头文件和汇编程序头文件两部分。

1．2 TI DSP Library 的下载和安装

由于 DSP Lib 种类繁多，且属可选模块，通常的 DSP 开发环境(CCS，Code Composer Studio)并没有配备 DSP Lib。因此，使用一个 DSP Lib 之前，必须进行 DSP Lib 的下载和安装。

所谓下载，就是在 TI 公司网站 WWW．ti．com 上免费下载各种 DSP Lib；而所谓安装，就是在 DSP Lib 下载完毕后，双击安装文件，以将它安装在计算机中选定的位置(默认位置为 C：＼ti)。安装之后，即可在程序开发中使用 DSP Lib 的库函数。

1．3 TI DSP Library 的使用

按处理数据类型的不同，TI DSP 分为定点(fixed-point)DSP 和浮点

(floating-point)DSP。由于浮点 DSP 既有定点指令集，又有浮点指令集，因此，本文选取浮点 DSP 系列 TMS320C67x 的 DSP Lib，并且选取了 TMS320C67x DSP Library 和 TMS320C67x FastRTS Library 两个 DSP Lib，前者主要针对数字信号处理的常用操作，后者则针对一般数学运算的通用操作。

2 TMS320C67x DSP Library 的应用

当 DSP 进行数据处理时，卷积、FFT、FIR 滤波等操作频繁出现，故在程序开发中，使用 DSP Lib 来完成这些操作将大大提高整个程序的效率并简化编程。TMS320C67x DSP Library 就是这样的一个 DSP Lib，它的 lib 文件夹内含库文件 dsp67x.lib 和源文件 dsp67x.sr、dsp67x_C.sr-c、dsp67x_sa.src。TMS320C67x DSP Library 主要用于 TMS320C67x 系列 DSP 芯片的程序开发，使用它可完成 FFT 运算。

2．1 TMS320C67x DSP Library 的使用

使用 TMS320C67x DSP Library 的第一步是将其核心文件“dsp67x. lib”加入到当前工程中，相关编译链接参数为“-ldsp67x. lib”；接着，将存储头文件 的 include 目录所在路径添加到工程搜索路径中，其相关编译链接参数为“-i pathname”，具体操作可参考 TI 公司的有关文献。选取该 DSP Lib 中的库函数“DSPF_sp_cfftr2_dit()”可完成 FFT 运算，它使用的是基 2 的时间抽取算法，具体形式如下：

库函数　Void DSPF_sp_cfftr2_dit　(float *x, float *w, short n)

参数　x 输入复数组，同时存储输出的频域值，输入顺序，输出位倒叙；

w旋转因子，位倒叙存储；

n输入数组的长度 (复数的个数)；

同时，该库函数还有一个对应的头文件“dspf_sp_cfftr2_dit. h”，使用时可将其包含到调用该库函数的程序中。此时，该库函数就可以像一般子程序一样被其他程序调用，具体使用代码如下：

```
#include <dspf_sp_cfftr2_dit.h> // 包含头文件
    ……
void main ()  {
    ……
    gen_w_r2 (w, N) ; // 产生FFT运算所需的旋转
因子
    bit_rev (w, N>>1) ;  // 旋转因子进行位倒叙操
作
    DSPF_sp_cfftr2_dit (x, w, N) ; //调用库函数进
行FFT运算
    ……
}
```

为了便于比较，可使用归档器指令“ar6x”从该 DSP Lib 的源文件“dsp67x_c.src”中提取出库函数的源代码，以得到文件“sp_cfftr2_ dit.c”。所有归档器指令的命令文件都存储在 CCS 的安装目录下，这里，“ar6x”的使用格式为：

ar6x-x dsp67x_c. src sp_cfftr2_dit. c

从“sp_cfftr2_dit. c”中可得到库函数“DSPF_sp_cfftr2_dit()”的 C 语言源代码，相应的 C 程序为“void sp_cfftr2_dit(float*x，float*w，short n)”，该程序可以像一般子程序一样被主程序调用。源函数和库函数的形式完全相同。实际上，库函数就是对源函数的程序代码进行手工优化的结果。

2．2 性能分析

分别使用库函数和源函数可完成 FFT 运算。并可用 CCS 自带的剖析工具“Profiler”来分析两个函数由于编程方式的不同所带来的运 行时间上的差异。改变输入数组的长度，可得到如表 1 所列的一组数据。由表 1 可以看出，库函数的效率远远高于源函数，其效率的提高量随着输入数据长度的变化 而变化，最高的效率可提高 40 倍(40. 98-1=39. 98)，最低仍在 25 倍左右，而且该 DSP Lib 的其他库函数也有相近的测试结果。虽然用该 DSP Lib 的库函数后，程序效率可以提高一个数量级，对于时间限制较为严格的系统，特别是实时系统，这仍然是非常有用的。

库函数和源函数相比，其效率有了很大提高，但这种提高是有代价的。它主要表现为通用性降低。其原因是为了最大限度的提高效率，在对代码进行手工优化的过程 中，引入了一些强假设，同时，使用了大量的操作合并、并行处理等简化手段，这必然导致库函数的通用性降低。例如，库函数 “DSPF_sp_cfftr2_dit()”使用时就会受到以下条件的限制：

(1)输入数组的长度必须是 2 的幂级数，且不得小于 32；

(2)输入数组 x 和旋转因子数组 w 必须按双字对齐方式存储，即数组起始地址的末 3 位必须是零；

(3)数据的存储格式必须是小端模式(Little Endian)；

(4)执行期间可接收中断，但不予响应，这可能导致一些实时事件得不到及时响应。

表1 使用库函数和源函数完成FFT的测试数据

输入数组的长度	完成操作所需时钟周期个数		二者所花时间比
	库函数	源函数	库函数：源函数
32	499	15663	1.00：31.39
64	987	36224	1.00：36.70
128	2067	82599	1.00：39.96
256	4523	185350	1.00：40.98
512	11752	413505	1.00：35.19
1024	36142	917224	1.00：25.38
2048	79277	2002627	1.00：25.26

如果使用“DSPF_sp_cfftr2_dit()”时不考虑到这些限制，就有可能导致程序运行异常。因此，库函数的效率虽然高，但不能盲目的滥用，在程序开发时，必须根据实际情况在通用性和效率之间进行折衷，以合理的使用库函数。

3 TMS320C67x Fast RTS Library 的应用

在 DSP 进行数据处理时，除了一些典型的操作外，还存在大量常规的操作，如除法操作、对数运算、三角函数等，这些操作也是很费时的，提高 这些操作的代码效率，也能显著提高整个软件的效率。TMS320C67x FastRTS Library 就是这样的一个 DSP Lib，它通常由 Lib 文件夹、include 文件夹和 doc 文件夹组成。其中 lib 文件夹内含库文件 fastrts67x. lib(Little Endian)、fastrts67xe. lib(Big Endian)和源文件 fastrts67x. src；include 文件夹内含头文件 fastrts67x. h 和 recip. h；而 doc 文件夹内含帮 助文件。

3. 1 TMS320C67x FastRTS Library 的使用

TMS320C67x FastRTS Library(以下简称 FastRTS Library)主要用于处理一些常规的操作。由于在通常情况下，CCS 已经有一个 RTSLib-rary 来完成这些操作(例 如，“rts6700.lib”就是一个适用于 TMS320C67x 的 RTS Library 文件)，因此，如果要使用 FastRTS Library，就必须在编译链接过程中先于“rts6700. lib”来编译链接“fastrts67x. lib(或 fastrts67xe. lib)”，相应的编译链接命令选项为：

-l fastrts67x. lib－rts6700. lib 或 -l fastrts67xe. lib－rts6700. lib

FastRTS Library 同样需要注意头文件的使用，它有两个头文件：“fastrts67x.h”和“recip.h”。如果使用 FastRTS Library 中的特殊函数(三角函数，对数函数等)，则必须包含“fastrts67x.h”；而如果使用求倒数操作，则必须包含“recip. h”。 FastRTS Library 的使用方式如下：

```
#include fastrts67x.h        // 包含头文件
#include recip.h
        ……
void main ()  {
        float pi=3.14159;
        float a;
        ……
        a=cossp (pi) ;    // 调用三角余弦函数cossp ()
        ……
}
```

3．2 性能分析

分别使用 FastRTS Library 和 RTS Library 可完成一些常用操作，使用剖析工具可得到各个操作所需的时钟周期个数，具体如表 2 所列(所有的操作均处理单精度浮点数)。对比表 2 中的数据 可以发现，和 RTS Library 相比，FastRTS Library 大大提高了程序的效率。

表2　FastRTS Library和RTS Library性能对比

操作	所需的时钟周期个数		二者所花时间比
	FastRTS Lib	RTS Lib	FastRTS Lib：RTS Lib
sqrt ()	50	159	1.00：3.18
cos ()	75	182	1.00：2.43
exp ()	81	210	1.00：2.59
求倒数	39	180	1.00：4.62

4 DSP Lib 的编写

事实上，程序员并非只能被动的使用 DSP Lib。只要遵循相应的规则，程序员也可以自己编写一个 DSP Lib。编写一个最简单的 DSP Lib 的步骤如下：

(1)新建一个工程 newLibrary，将其属性设为“Library(. lib)”，图 1 所示是新工程设置示意图；

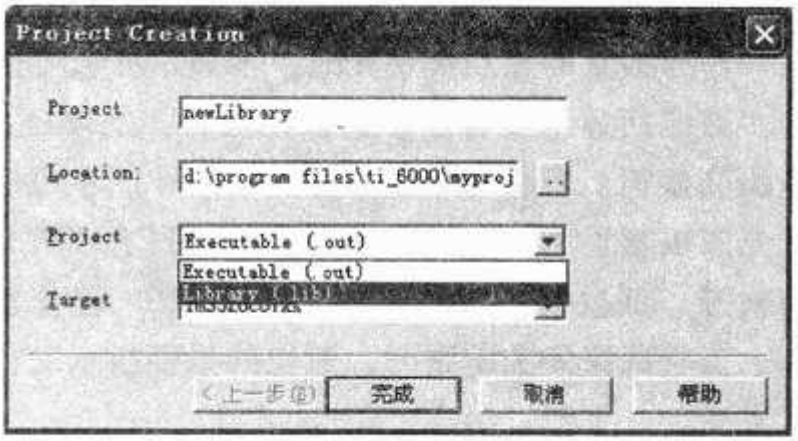

图1　新工程属性设置界面

(2)编写高效率代码文件 myLib1. asm、myLib2. asm、myLib3. asm，……

(3)将 myLib1. asm、myLib2. asm、myLib3. asm，……等文件添加到工程 new Library 中；

(4)编译链接工程 new Library；

完成上面 4 步后，工程中就会出现库文件 newLibrary.lib，这样，一个 DSP Lib 就制作成功了。为了使 DSP Lib 具有保密性，通常情况下，只需保留工程中的 newLibrary. lib 文件，而将其他文件，特别是源代码文件*. asm 删除或保密存放。这样，用 户就只能使用库文件，而无法从中得到源代码的信息。

5 结束语

本文以 TMS320C67x DSP Library 和 TMS320C67x FastRTS Library 为例，详细介绍了如何在程序开发中使用 TI DSP Library，并分析了使用 TI DSP Library 所带来的程序效率的提高。最后，还给出了编写 TI DSP Library 的一个应用实例。