基于 DSP的实时图像压缩软件优化技术研究

孟占红, 赵保军
(北京理工大学电子工程系 551室, 北京 100081)

摘 要: 本文提出了一种基于 DSP(Digital Signal Processing)核及其数据链路特征的优化方法, 并验证了该优化方法可以使软件效率提高 5~15倍. 高效快速的算法程序, 不但使图像的实时传输变为可能, 还可以节约硬件成本, 使系统变得简单, 提高系统的稳定性. 为解决软件效率问题, 本文主要研究了去相关、内联函数、短整型数据用整型处理和软件流水技术, 并把这些优化技术应用在基于 C6416平台的实时图像压缩系统中, 在输入数据率 82M byte/s~110M byte/s, 输出码流 2.9M byte/s的条件下, 使该系统达到了实时处理的能力.

关键词: 软件流水; 去相关; 内联函数; 实时处理
中图分类号: TN957 文献标识码: A 文章编号: 0372-2112 (2006) 09-1558-04

The Study on Software Optimizing Technology of Real-Time Image Compression System Based on DSP

MENG Zhan-hong, ZHAO Bao-jun
(551 Lab of Dept of Electronic Engineering, Beijing Institute of Technology, Beijing 100081, China)

Abstract This paper studied the optimizing technology based on the character of the DSP (Digital Signal Processing) and its data route way, and proved that these optimizing technologies could improve the software's efficiency from five times to fifteen times. High efficient programmer can make the system real-time, and it can reduce the cost of hardware, make the system simple and improve its reliability. To improve the software efficiency, this paper studied the removing relativity, intrinsic, short data processed as int data and software pipeline technologies, and these technologies are applied in the real-time image compression system based on C6416, with the inputting data rate 82M byte/s~110M byte/s and the outputting data rate 2.9M byte/s, the system achieved the goal of real-time processing.

Key words software pipeline; remove relativity; intrinsic; real-time processing

1 引言

在现代社会里, 视觉信息在很多领域都扮演着重要的角色. 但是数字图像信息需要大的存储容量和宽的传输信道, 尤其是在传输高分辨率实时图像序列的场合, 如果原始图像数据不经过压缩处理就直接传输是不现实的, 因为信道带宽有限, 这就需要快速高效的压缩系统来解决问题[1].

常规的图像压缩解压缩处理器多使用以 MPEG4和 JPEG为标准的专用芯片, 主要应用于时滞较大、数据率不是很大的电视直播以及声音等的编解码[2]; 而本实时系统中的输入数据率为 82M byte/s~110M byte/s, 用传统的方法进行压缩是不可能的, 在系统硬件资源有限的情况下, 这就需要研究和开发快速高效的软件来解决问题. 高效快速的算法程序, 不但使图像的实时传输变为可能, 而且还可以节约硬件成本, 所以软件优化就显得尤其重要了.

2 硬件系统概述

受体积的限制, 该实时图像压缩系统由 3片 C6416 (DSP1、DSP2和 DSP3)、FPGA、同步 RAM、大容量 SDRAM、视频显示等模块组成. 其中 C6416是业界目前性能最高的 DSP处理器, 主频达到 720MHz, 峰值处理能力为 5.76G IPS (5.76G Instruction Per Second)[3]. 系统输入数据率为 82M byte/s~110M byte/s, 输出码流 2.9M byte/s, 前端由 FPGA进行数据的预处理 (数据重排序), DSP1进行四级小波变换, 变换出来的小波系数通过同步双口缓存以乒乓倒的方式传给 DSP2和 DSP3, 之后由 DSP2和 DSP3完成数据编码.

收稿日期: 2005-07-28; 修回日期: 2006-12-26

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3 算法优化

3.1 算法概述

C64X系列 DSP有 6个逻辑运算单元和 2个乘法器, 当程序中进行大量的乘加运算时, CCS(和 DSP配套的代码编译软件)的 C编译器可以生成高效的代码, 以实现在 DSP上的并行流水处理, 可极大地提高 DSP的运算效率; 而当进行大量的判断时, 流水处理将被阻塞, 此时 DSP运算效率大大降低. 基于 DSP的这种情况, 本文开发了一种以乘加运算为主、判断为辅的基于小波变换放入压缩算法 ——类零树算法, 有效地提高了 C6416的运算效率. 该算法中, 离散小波变换采用的是线性相位双正交 5/3小波, 计算量相对比较小, 变换效果又比较好[4].

3.2 软件流水处理的硬件资源优化配置

对于 110M byte/s的数据输入速度来说, 类零树算法还是太慢了, 不能达到实时处理, 这就需要针对 C64X的硬件资源优化配置对程序进行优化. 既然是针对 C6416的程序优化, 首先要知道 C64X的 CPU数据链路的下列特点:

(1) C64X片内有两个数据通道、8个功能单元 (. L1, . L2, . S1, . S2, . D1, . D2, . M1和. M2, 其中. M1和. M2为两个专用的硬件乘法器)和两个通用寄存器组 (A和 B), 每个寄存器组包括 32个通用寄存器, 这为软件流水提供了硬件基础.

(2) C64X DSP采用超长指令字 (VLIW, Very Long Instruction Word), 即在每个时钟周期最高可提供 8条 32位指令, 总字长为 256位的指令包同时分配到 8个并行处理单元[5]. 在 720MHz的时钟频率下, 当片内 8个处理单元同时运行时, 其最大处理能力可以达到 5.76GIPS, 这是软件流水最重要的硬件资源优化配置.

(3) C64X DSP具有双 16位扩充功能, 芯片能在一个周期内完成双 16位的乘法、加减法、比较、移位等操作, 这为后面的短整型数据用整型处理提供了硬件基础.

3.3 DSP程序的优化技术和方法

程序的优化阶段分为 C语言级的优化和汇编语言级的优化, 如果优化了 C代码之后, 还不能满足效率要求, 则再进行汇编级优化. 优化之前, 需要先用 CCS内的代码检测工具检测出比较耗时的循环代码段, 然后对这些循环体进行优化, 优化的目的是为了让它流水起来执行. 具体的优化方法如下:

(1)选用 C编译器中提供的优化选项

根据实际编译的程序特征, 选择合适的优化选项, 对源程序进行优化.

(2)消除存储器相关性

为使指令并行操作, 编译器必须知道指令间的关系, 因为只有不相关的指令才可以并行执行. 优化时, 可以用关键字 restrict(专用的)说明两个目标指针是相互独立的, 即访问的地址没有任何相关性, 一但编译器得到此信息, 它就会流水存取数据. 例如下面一段小波逆变换的程序段 1采用了加 restrict关键字去相关的技术, 优化后, 效率提高到原来的 4.3倍, 程序段 2是程序段 1对应的汇编代码, 由此可以看出该程序段的指令并行性很高.

程序段 1:

```
void Syn_Row_LDWT_ ( short* restrict in_data,
short* restrict out_data)
{ int i, j, k= 0, h= 0,
for( j= 0; j< 288; j+ + )
for( i= 0; i< 1024; i+ + )
{ out_data[ h+ + ]= in_data[ k];
out_data[ h+ + ]= ( in_data[ k] + in_data[ k+ 1] ) /2,
k+ = 1; } }
```

程序段 2:

```
||    [! A1]    STH. D2T2    B6, * + + B4[ 0x2]
||    [ B0]     BDEC. S2     L96, B0
||              SHRU. S1     A7, 0x1f, A5
||              OR. L2X      0, A8, B6
||              LDH. D1T1    * + + A3[ 0x1], A6
      [ A0]     MPYSU. M1    2, A0, A0
||    [! A0]    STH. D2T2    B5, * - B4[ 0x3]
||              SHR. S2X     A4, 0x1, B5
||              ADD. S1      A5, A7, A4
||              ADD. L1      A8, A6, A7
||              LDH. D1T1    * + A3[ 0x0], A8
```

要想正确运用去相关优化技术, 就要知道 C64X的片上 1M bytes的内存是由 5个 block(存储块)组成的, 具体 block大小和区间见参考文献 [3], 只有当两个指针所指的内存在不同的 block时, 用 restrict去相关才是合法的.

(3)短整型数据用整型处理或者更长的数据类型进行处理

C64X具有双 16位扩充功能, 芯片能在一个周期内完成双 16位的乘法、加减法、比较、移位等操作. 在优化时, 当对连续的短整型数据流操作时, 应该转化成对整型数据流的操作, 这样一次可以把两个 16位的数据读入一个 32位的寄存器, 然后用内联函数来对它们处理, 充分运用双 16位扩充功能, 一次可以进行两个 16位数据的运算, 速度将成几倍的提高. 例如下面的进行 Z字型逆排序的程序段 3采用了短整型数据用整型处理和去相关技术, 效率提高到了原来的 9倍; 而程序段 4使用了短整型数据用整型处理和用内联函数两种优化技术, 效率提高到原来的 6.7倍, 这里效率的提高是因为充分运用了 C64X的双 16位扩充功能.

程序段 3:

```
void Data_Conver3_De ( const int* restrict idata, int* restrict
                       odata)
```

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

```
{ int i, j, k, m;
for( k= 0; k< 2; k+ + )
{m= k* 1024; j= 0;
for( i= 0; i< 1024; i+ = 4)
{ odata[ j+ m ] = idata[ i+ m ];
odata[ j+ 1024+ m ] = idata[ i+ 1+ m ];
odata[ j+ 1+ m ] = idata[ i+ 3+ m ];
odata[ j+ 1024+ 1+ m ] = idata[ i+ 4+ m ];
j+ = 2; }}
```

程序段 4

```
for( i= 0, j= 1792, m= 3584, n= 5376; i< 1792;
i+ +, j+ +, m+ +, n+ + ) {
p_ int1[ i] = _ add2( p_ int2[ i], p_ int3[ i]);
p_ int1[ j] = _ add2( p_ int2[ j], p_ int3[ j]);
p_ int1[ m ] = _ add2( p_ int2[ m ], p_ int3[ m ]);
p_ int1[ n] = _ add2( p_ int2[ n], p_ int3[ n ]); }
```

(4)使用内联函数和循环展开

内联函数是 C64X编译器提供的专用函数, 它们是直接与 C64X 汇编指令映射的在线函数, 其目的是快速优化 C程序, 循环体内加入内联函数不影响程序的流水执行, 这是和一般函数的最大区别; 而循环展开可以使 CPU 内的功能单元和寄存器得到充分的利用, 使循环体达到最佳流水状态. 程序段 5采用了内联函数和循环展开两种技术, 效率提高到原来的 11. 4倍.

程序段 5

```
FX( int* ) 0x0020000;
for( i= 0; i< 11468; i+ = 2)
{* (FX+ i) = _ max2(* FX+ i), 0);
* (FX+ i) = _ min2(* (FX+ i), 0xff00ff);
* (FX+ i+ 1) = _ max2(* (FX+ i+ 1), 0);
* (FX+ i+ 1) = _ min2(* (FX+ i+ 1), 0xff00ff); }
```

(5)使用逻辑运算代替乘除运算

在 DSP指令里, 乘除运算都是多周期指令, 优化时, 可以根据实际情况, 尽量用逻辑移位运算来代替乘除运算, 这样可以加快指令的运行速度, 也有助于循环体的流水执行.

(6)软件流水技术的使用

在图像压缩的 DSP算法中, 存在大量的循环操作, 因此充分地运用软件流水技术, 能极大地提高程序的运行速度, 但使用软件流水线还有下面几点限制:

◆循环结构中不能包含一般的函数调用, 但可以包含内联函数.

◆循环结构中不能有 break和 goto语句, if语句不能嵌套, 条件代码应尽可能的简单.

◆循环结构中不要包含改变循环计数器的代码.

◆循环结构代码不能过长, 因为寄存器的数量 (64)有限, 太长的话, 可以分解为多个循环体.

在编译器优化级别为 -o2的基础上, 针对以上的代码段总结一下 C代码优化方法和测试的优化结果, 见下面的表 1:

表 1 优化测试结果及其采用的技术

程序段序号	优化前时间 (us)	优化后时间 (us)	效率比较 (倍)	采用的技术
1	390	90	4. 3	去相关技术
3	362	40	9. 0	短整型数据用整型处理和去相关技术
4	168	25	6. 7	短整型数据用整型处理和内联函数技术
5	2608	228	11. 4	内联函数和循环展开技术

(7)汇编代码级的优化

在经过 C代码的优化之后, 还不能满足性能上的要求, 则可以通过 CCS的时间测试工具 profile clock 找出效率比较低的部分, 使用线性汇编重新改写. 用线性汇编改写程序时, 实现流水执行的要求主要有以下几点:

◆寄存器的数目不能超过 64个, 条件寄存器的数目不能超过 6个;

◆程序中不能含有分支语句, 可能的话尽量使用条件语句[3];

◆程序所使用的 CPU 左右两边的资源尽量平衡, 这样可以使流水周期比较短;

◆尽量把不相关的指令安排在一起并行执行, 采用指令乱序技术, 减小指令的相关性;

满足上述要求之后, 再通过汇编优化器的优化, 一般可以满足性能上的要求.

4 结论

以往人们只是追求快速的图像压缩算法, 而不去对自己的算法进行深层次的优化, 导致硬件系统过分复杂、成本过高、可靠性得不到保证. 如果只是简单的针对语法进行优化, 程序的效率也能得到一定的提高, 但很难达到 1倍; 只有把软件算法和硬件的结构特点融为一体, 才能大幅度地提高程序的速度. 本文提出的基于 DSP核及其数据链路特征的优化方法有效地解决了上述不足, 实验证明本文所提出的优化技术和方法可以使程序效率提高 5~ 15倍, 本算法已在某国家重点项目中成功应用.

参考文献:

[1] 沈利军. 基于 DSP的高速实时图像编解码系统研究 [D]. 北京: 北京理工大学博士论文, 2005.
Shen Lijun. Researches on DSP based High-Speed Real-Time image Encoding and Decoding System [D]. Beijing: Beijing Institute of Technology, 2005. (in Chinese)

[2] 赵保军, 史彩成, 毕莉, 安建波, 毛二可. 基于 FPGA和 DSP实现的实时图像压缩 [J]. 电子学报, 2003, 31(9): 1318-1319.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

ZHAO Bao-jun, SHI Cai-cheng, BI Li, AN Jian-bo, MAO Erke. Implementation of real-time 2D-DCT with FPGA and DSP [J]. Acta Electronica Sinica, 2003, 31(9): 1318- 1319. (in Chinese)

[3] 李方慧, 王飞, 何佩琨. TMS320C6000系列 DSPs原理与应用[M]. 北京: 电子工业出版社, 2003. 6.
Li Fang-hui, Wang Fei, He Pei-kun. The Principles and Applications of TMS320- C6000 DSPs[M]. Beijing: Publishing House of Electronics Industry, 2003. 6. (in Chinese)

[4] Julien Reichel, Gloria Menegaz, Marcus J Nadenau, Murat Kunt. Integer wavelete transform for embedded lossy to lossless image compression [J]. IEEE Transactions on Image Processing, 2001, 10(3): 383- 384.

[5] Jian Wang, Bogong Su. A scalable loop optimization approach for scalable DSP processors[J]. IEEE ASSP[C]. Istanbul, Turkey: IEEE, 2000.

作者简介:

孟占红　女, 1979年出生于河南省开封市, 硕士. 1999年进入北京理工大学电子工程系, 2003年又被录取为本系信号与信息处理专业研究生, 主要研究方向为图像压缩、DSP程序优化、离散小波变换.
E-mail: mengzhanhong_2005@yahoo.com.cn

赵保军　男, 1960年出生于陕西省西安市, 教授, 博士导师. 承担并完成近40项科研项目, 发表学术论文100多篇, 主要研究方向为图像处理、信号处理、神经网络与模糊控制、DSP与ASIC设计.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net