并行处理 JPEG 算法的优化

薛永林, 刘 珂, 李凤亭
(清华大学电子工程系, 北京 100084)

摘 要: 本文结合指令级并行 32 位定点处理器的结构特点, 对 JPEG 算法中 DCT、量化及 Huffman 编码等步骤, 提出一些适于并行处理算法和数据结构的优化方法, 以有效发挥其高速并行的性能. 模拟结果表明本文的方法显著提高了 JPEG 算法并行实现的效率, 适于图像压缩的一些实时应用.

关键词: 图像压缩; 并行处理; 流水线; JPEG 算法; 优化
中图分类号: TN919.81 文献标识码: A 文章编号: 0372-2112 (2002) 02-0160-03

Optimization for a Parallel JPEG Algorithm

XUE Yong-ling, LIU Ke, LI Feng-ting
(Department of Electronic Engineering, Tsinghua University, Beijing 100084, China)

Abstract: This paper describes some methods on optimizing parallel processing and data structure for DCT, quantization, and Huffman coding of JPEG algorithm with parallel instruction of 32 bits fixed-point processors, in order to develop its parallel processing capability effectively. The result of simulation indicates that the methods obviously speed up parallel implementation of JPEG Algorithm, being applicable to the real-time image compression.

Key words: image compression; parallel processing; pipeline; JPEG algorithm; optimization

1 引言

在移动平台图像压缩传输的场合, 信道衰落严重, 为避免误码扩散引起图像质量劣化, 有时希望采用无帧间预测的 Motion JPEG 方法, 而且希望图像分辨率和帧率可根据信道情况改变. 为同时满足图像压缩的实时性、可靠性和灵活性要求, 需用可编程 DSP 作为压缩处理器.

TMS320C62X 是美国德州仪器公司(Texas Instrument, 简称 TI) 90 年代后期的高速数字处理器(Digital Signal Processor, 简称 DSP)产品. TMS320C62X 为定点运算芯片, 它采用 VelociTI™ 体系结构. 这种高性能、先进的超长指令字(very long instruction word, 简称 VLIW)体系结构通过增加指令级并行流水处理赋予芯片高性能. 它采用程序代码段和数据段分开的 Harvard 结构, 芯片核心 CPU 有 32 个 32 位机器字(Word)长的通用寄存器和 8 个功能单元(2 个乘法器和 6 个算术逻辑单元), 故可以在每个时钟周期内并行运行多达 8 条 32 位指令. 所有 DSP 指令均可以条件执行, 该特点有利于减少程序跳转, 增加程序的并行性. 采用这样的 DSP 实现 Motion JPEG 算法, 还需要结合其结构特点进行处理算法和数据结构的优化, 才能有效的发挥其高速并行的性能.

(1) 尽量充分利用芯片的 32 位数据操作, 减少存取访问次数. 例如, 对于矩阵运算, 可以采用有效的数据结构, 改 Byte 结构为 Word 即 4 个 Byte 字结构, 这样不但减少了存取操作, 而且还减少了循环次数, 加快了处理速度.

(2) 利用指令流水线特点, 合理安排指令执行顺序. 采用 VLIW 的指令流水线设计, 指令的执行具有延迟特性, 即指令执行的结果需自指令执行的时间延迟某一时间(Delay Slot)才能得到. 充分利用这些 Delay Slot 是优化的重要步骤.

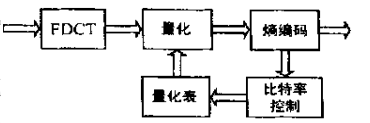

图 1

(3) 合理安排处理流程, 尽量做到并行化处理, 这是提高运行效率的关键步骤.

2 JPEG 算法的优化方法

JPEG 算法编码、解码的一般流程如图 1、2 所示.

2.1 采用整数 DCT 和 IDCT 快速算法

DCT 和 IDCT 的计算公式如下:

DCT:

$$S_{uv} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

IDCT:

$$S_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{uv} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

当 $u, v = 0$, $Cu, Cv = 1/\sqrt{2}$;
否则, $Cu, Cv = 1$.

收稿日期: 2000-04-10; 修回日期: 2001-10-10

对于定点处理, 采用整数运算来实现 FDCT 的 Chen 氏算法, 并分别对行和列运算循环进行展开, 展开后进行指令并行处理. 通常情况下, 8×8 的 FDCT 运算是对每一个 8× 8 块, 先做列方向上的 FDCT, 再做行方向上的 FDCT, 如此来完成该块的二维 FDCT 运算之后, 再顺序做下一块的二维 FDCT. 为了在水平与垂直方向的循环中实现流水线操作, 我们把原来在垂直方向的嵌套循环展开为一个单级循环, 具体实现为:

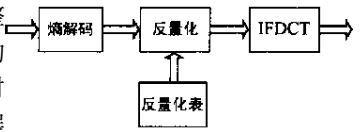

图 2

首先, 对所有 8× 8 像素块, 实行垂直方向的一维 FDCT 运算. 即在每一个 8× 8 像素块, 对每一列的 8 个像素进行一维 FDCT 运算, 并将运算结果存到该列相应的位置上; 然后对所有 8× 8 像素块, 实行水平方向的一维 FDCT 运算, 即在每一个 8* 8 像素块, 对每一行的 8 个元素进行一维 FDCT 运算, 并将运算结果存到该行相应的位置上. 所有运算均为同址运算.

同样在进行 IFDCT 运算时, 所有嵌套循环也都被展开成为单级循环, 并且是流水线操作. 对每一个 8× 8 DCT 系数块, 由于需要转置操作, IDCT 不可能完全同址运算. 为了节约存储空间, 水平方向的路径, 是按从末尾到开始的顺序进行的, 将运算结果行列转置保存在对应于该 DCT 系数块的下一个块内. 因而, IDCT 运算是准同址运算. 垂直方向的路径, 执行 IDCT 操作时, 同水平方向相反, 是按从数组的开始到末尾的顺序进行的, 将运算结果转置保存在对应于该 DCT 系数块原来的位置上. 这样带来的一个好处是, 无论进行多少 IDCT 块的运算, 水平循环的结束地址, 相对于垂直循环中的地址, 有一个固定的偏移量, 这使得地址的重置相当简便.

采用上述方法分别对行和列运算循环进行展开后, 为了充分发挥指令高速并行的性能优势, 即可以在每个时钟周期内并行运行多达 8 条 32 位指令, 利用并行处理部件进行指令流水线处理, 将程序代码安排为尽可能紧凑的并发执行的指令包, 避免指令包内不必要的空操作; 对于循环外操作也可与循环条件等组合为并发执行的指令包.

经过优化, 可在 226 个 DSP 时钟周期(每个时钟周期为 5ns) 内完成 8× 8 短整数型亮度矩阵的 DCT 变换, 在 230 个 DSP 时钟周期内完成 8× 8 矩阵的反 DCT 变换, 而优化前时间分别为 758 个时钟周期和 862 个时钟周期.

2.2　压缩数据储存单元扩展

JPEG 标准中, 压缩数据按 8 比特为单元存放, 即压缩数据依次按从最高位(MSB) 到最低位(LSB) 的顺序填充存储器每字节(Byte) , 填满一个字节后, 再填充下一个字节. 但对于 32 位并行处理器来说, 因为流水延迟使存取操作需几个时钟周期, 而绝大多数的数学运算仅需 1 个时钟周期. 故将压缩数据存储单元扩展为 32 比特, 即一个机器字(Word) 长. 这样比原来的 Byte 型储存单元减少了 3/ 4 的存取访问次数.

经过 Word 型码流储存单元扩展, 原来的 JPEG 中的 8 位压缩码流中标识码 0xff 符号扩展成 32 位标识码 0xffffffff. 可作如下证明:

(1) JPEG 每个压缩码元可以表示如下为(Huffman 码, Value 码) , 对于直流 DCT 系数 Huffman 码长度范围为 2~ 9 位, 交流 DCT 系数 Huffman 码长度范围为 2~ 16 位. Huffman 码不出现全 1 码, 且 Huffman 码中起始处连 1 的位数最大为 15, 即 Huffman 码 F/ A(1111111111111110) .

(2) 设 Huffman 码结尾处连 1 的位数为 S, Value 码中连 1 的位数为 T, 可以发现 $S+T$ 的最大值为 14, 即 Huffman 码 A/A(1111111111001111) 和 Value 码为 1023(1111111111) 时, 这里 $S=4$, $T=10$.

如果 32 位存储单元中包含一个完整的 Huffman 码, 则由(1) 知, 这个 Huffman 码中必出现 0, 所以在这种情况下, 该存储单元不会出现标识码 0xffffffff, 即 32 位全 1.

如果 32 位储存单元不包含一个完整的 Huffman 码, 由(1) 知 C 部分最大的连 1 位数为 15; 由(2) 知 A 与 B 部分合起来最大连 1 位数为 14. 所以, 该码流存储单元中最大的连 1 位数为 14+ 15= 29. 而标识码 0xffffffff 连 1 位数为 32, 故标识码不会在码流存储单元中出现. 同时如果压缩码流未填满一个完整的存储单元, 则对该单元剩余位填 0.

这样就证明标识码 0xffffffff 在压缩码流中不会出现, 从而免去了 Byte 型压缩码流存储单元中因码流可能出现标识码而进行的补 0 操作.

2.3　量化运算优化

设 DCT 变换后 8× 8 系数矩阵的元素为 Dxy, 相应的量化矩阵元素为 Qxy, 则量化后的系数矩阵相应元素为 $Cxy=$ Round(Dxy/Qxy). 但是对于定点整数运算处理器, 一般不提供整数除法指令, 除法运算只能调用数学仿真功能, 而这样的整数除法往往需要几十个时钟周期. 由于 8× 8 的系数矩阵需要 64 次除法操作, 而且对每个 MCU(Minimum Coded Unit) 都进行, 故对量化运算进行优化, 将耗时的除法改为乘法和移位运算, 将显著减小运算量, 具体方法如下.

将 Dxy/Qxy 进行改写:

$$\begin{aligned} Dxy/Qxy &= (Dx \times \mathrm{MAXSCALE})/(Qxy \times \mathrm{MAXSCALE}) \\ &= (Dxy \times (\mathrm{MAXSCALE}/Qxy))/\mathrm{MAXSCLAE} \\ &= [Dxy \times Q'xy]/\mathrm{MAXSCALE} \end{aligned}$$

取 MAXSCALE 为 2 的幂, 即 MAXSCALE= 2^m, 则上式简化为$[Dxy \times Q'xy] \gg m$. 这里 $Q'xy$ 为预处理后的量化表, $Q'xy=$ MAXSCALE/ Qxy. 可见对每一量化表只需进行 8× 8= 64 次除法, 生成预处理后的量化表, 以后对每个 MCU 的量化只需做快速的乘法(2 个 DSP 时钟周期) 和快速的符号数右移位操作(1 个 DSP 时钟周期) , 这样量化运算速度提高了 8 倍多.

此外还应合理选取 MAXSCALE 的值, MAXSCALE 应较大, 从而提高 MAXSCALE/ Qxy 即 $Q'xy$ 的精度; 但又要保证 Q'_{xy}的值在 16 位整数范围内(因为 C6201 的乘法指令为两 Short 型整数乘法) . 这里取 MAXSCALE 为 $2^{14}=$ 16384.

2.4　Huffman 解码的优化

按 JPEG 算法, 一个压缩码元可以表示为: (Huffman 码) (Value 码) 的形式. 这里的 Huffman 码为变长码, 传统的解码方法是将压缩码流逐位移入, 查 Huffman 表直至找到正确的 Huffman 码. 由于处理器存取访问比较耗时, 逐位解码的方法

将需要大量的存取操作,降低模块的处理速度.

经过对 JPEG 标准推荐 Huffman 编码表研究,我们发现一种定长解码的方法,现介绍如下:

将每个Huffman 码扩充为一个固定的位长 L(L 应不小于 Huffman 编码集中最大的码长). 设某一Huffman 码 Ca 为 S 位长,表示为 $(B_{S-1}B_{S-2}\cdots B_0)$,扩充后表示为 $(B_{S-1}B_{S-2}\cdots B_0X_{L-S-1}X_{L-S-2}\cdots X_0)$,这里 B_{S-1}为最高位且 X 为 0 或 1. 其实,扩充后的结果为整数域$[0, 2^L-1)$(记为 F)的一个子域$[(B_{S-1}B_{S-2}\cdots B_000\cdots 0), (B_{S-1}B_{S-2}\cdots B_011\cdots 1)]$(记为 Fa). 这样,构造了一种从 Huffman 码 Ca 到整数域 F 的子域 Fa 的映射 H. 而且设另有一个 Huffman 码 Cb 为 W 位长,表示为 $(B_{W-1}B_{W-2}\cdots B_0)$. 由于 Huffman 码为唯一可解码,则两个码一定在从最高位记起的第 d 位出现不同, $d\leqslant$Minimum(S, W). 故 Ca 与 Cb 对应的子域 Fa 和 Fb 不重叠,这样我们就构造了一个从Huffman 码 C 到整数子域 F 的一一映射 H.

进一步,将这些两两互不重叠的子域排列在整数数轴$[0, 2^L-1)$上,注意因为Huffman 码不可能出现全 1,故 2^L-1 不可能为Huffman 码扩充至 L 位后的结果. 并且如果编码集为完全Huffman 码集,则这些子域将完全覆盖该数轴,即 $\sum Fi = F$. 如下图:

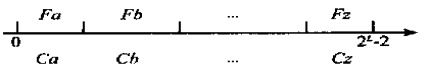

图 3

于是构造一个数组,按上图从下标 0 到下标 (2^L-2)依次存放对应的 Huffman 码元,在 JPEG 中是 Size(对于直流 DCT 系数)或 Run/Size(对于交流 DCT 系数)以及该 Huffman 码本身的码长 Len. 一次从压缩码流中读入 L 位码流 $(B_{L-1}B_{L-2}\cdots B_0)$,视为一个 L 位整数,并将它作为上述数组的下标,便可立即查到这 L 位中前 Len 位为一个 Huffman 码,且获得了该 Huffman 码对应的 Size 或 Run/Size 信息,用于解出紧跟Len 位 Huffman 码的Value 码.

一般处理器提供位段提取指令,该指令仅需 1 个 DSP 时钟周期. 对比逐位读入查表的方法,此优化方法只需一次访问内存读取 L 位码流,就可解出其中包含的 Huffman 码,从而极大的加快了Huffman 解码的运算速度,提高了近 5 倍.

然而须指出的是,本方法是以增加存储容量的代价来换取速度的提高. 一个长度为 2^L-1 的码元信息数组将消耗大量的存储空间.

为了减少解码数组占据的内存,对于交流 DCT 系数Huffman 编码表,选取 $L=12$. 将 127 个码长为 16 的 Huffman 码分离出来,建立一个长为 127 的补充解码表,并将码长为 2 的两个Huffman 码单独解码. 故只建立了 2K 长的解码表. 对于直流 DCT 系数 Huffman 表,模块选取 $L=9$. Huffman 解码消耗了约 5K 字节存储空间.

3 实验结果

对于每秒 25 帧分辨率为 512×512×8bits 视频图像,为了达到实时处理要求,每一个 8×8 块的处理速度计算如下:

由于 TMS320C6201 处理器时钟频率为 200MHz,为达到实时要求,1 秒内需处理 25×64×64= 102400 个 8×8 块,即平均每处理一个 8×8 块最多用 200M/102400= 1953 个处理器时钟周期.

采用独立 JPEG 组织的简化程序进行仿真,平均每处理一个 8×8 块约需要 5142 个时钟周期,不能满足实时要求. 经过优化,平均每处理一个 8×8 块只需 910 个时钟周期,已经达到实时性要求速度的 2 倍.

4 结论

根据实时应用要求和所采用 DSP 的结构特点,本文提出的适于 JPEG 算法指令级并行处理的优化方法,可以显著提高其运算速度. 优化方法虽然是针对 TMS320C62X 实现的,对于类似其 VLIW 指令结构的处理器也可采用. 此外,本文的方法也可推广到与 JPEG 有同类运算的基于 DCT 的图像压缩算法如 MPEG、H.263 等.

参考文献:

[1] International Standards Committee International Standard ISO/IEC DIS 10918-1. Digital Compression and Coding of Continuous-tone Still Images [S]. American National Standards Institute, New York, 1992.

[2] V Bhaskaran, K Konstantinides. Image and Video Compression Standards-Algorithms and Architectures [M]. Kluwer Academic Publishers, USA, 1996.

[3] Texas Instruments Incorporation. TMS320C62XX CPU and Instruction Set Reference Guide [M]. TI Co., Texas, USA, 1998.

[4] 张宪超,李宁,陈国良. 离散余弦变换的改进的算术傅立叶变换算法 [J]. 电子学报, 2000, 9: 88-90.

[5] Willam B Pennebaker, Joan L Mitchell 著,黎洪松,成实译. JPEG 静止图像数据压缩标准 [S]. 北京: 学苑出版社, 1996.

作者简介:

薛永林 男 1965 年出生于陕西. 1989 年在清华大学电子工程系获工学学士学位, 1992 年在中科院空间中心获硕士学位. 现为清华大学电子工程系讲师,目前研究方向是图像压缩、数字视频通信、遥感图像处理.

刘 珂 男. 1976 年出生. 1999 年在清华大学电子工程系获工学学士学位,现赴美留学.