用超长指令实现 DCT 的新算法

李学明, 李 继
(北京邮电大学信息工程学院, 北京 100876)

摘 要: 本文介绍一种新的 DCT 计算方法, 它以现有的 DCT 快速算法为基础, 利用超长指令的并行特征来提高 DCT 计算的性能. 仿真结果表明: 该方法的运算速度比普通的 DCT 计算方法提高 73%, 即便同快速算法相比, 也可以提高 24% 的运算速度.

关键词: 离散余弦变换; 超长指令字; 并行算法
中图分类号: TN951.52 文献标识码: A 文章编号: 0372-2112 (2003) 07-1074-04

New DCT Computation Algorithm for VLIW Architecture

LI Xue-ming, LI Ji
(School of Information Engineering of BUPT, Beijing 100876, China)

Abstract: In this paper, a new DCT computation algorithm is presented, which is based on existing fast DCT algorithm and has fully utilized the parallelism of very long instruction word (VLIW) architecture to achieve high speed performance. Simulation results show that this method can reduce more than 73% and 24% computing cycles, compared with normal and fast DCT computation algorithm respectively.

Key words: discrete cosine transform; very long instruction word; parallel algorithm

1 引言

离散余弦变换(Discrete Cosine Transform)(DCT)是一种常见的变换方法[1], 在图像处理、数据压缩、滤波和其它领域均有非常广泛的运用, 现有的图像处理国际标准如 H.261, H.263, MPEG-1/2/4 等均采用了 DCT 变换. 普通的 DCT 运算量比较大, 为了提高运算速度, 人们提出了很多种快速计算 DCT 的方法[4~7], 其中以 Loeffler, Ligtenberg 和 Moschytz 在 1989 年提出的算法最有代表性[8]. 这种方法由 4 个连续的步骤来完成, 对 8 点的一维 DCT, 只需要 11 次乘法和 29 次加法. 图 1 给出了该算法的计算流程. 图 2 解释了图 1 中使用的各个符号的含义.

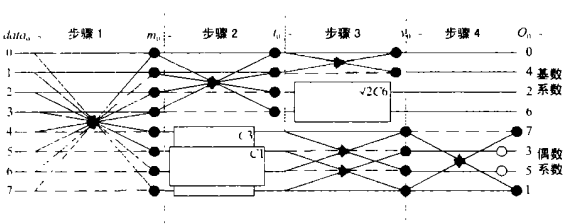

图 1 8 点一维 DCT 的计算流程

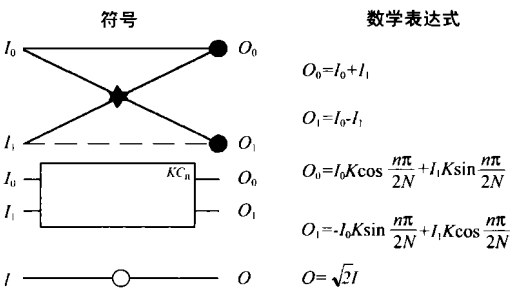

图 2 8 点 DCT 计算中符号及其含义

从图 1, 2 可以看出, 在每个步骤内部很多运算是相似的且相互独立的, 因此可以并行进行. 本文提出的算法充分利用了这一特征, 达到提高计算速度的目的. 下面我们将以快速算法为基础, 介绍如何利用 DSP 支持超长指令的特点并行完成每个步骤内的计算任务, 进一步提高运算速度.

2 基于超长指令的 DCT 计算方法

支持超长指令(Very Long Instruction Word, VLIW) 是近年来微处理器设计的热点之一. 这种类型的 DSP 通常具有较长的固定字长(比如 32 比特), 用户可以将几条比较短的指令组合成一条长的指令放入 CPU 的寄存器中. 由于多条指令可以并行执行, 因此可以大幅度地提高运算速度. 通常情况下,

收稿日期: 2001-12-12; 修回日期: 2002-10-20
基金项目: 国家自然科学基金(No. 60172050)

DSP 只负责超长指令的执行, 而用户使用的编译程序则负责将多条短的指令合并成为一条长的指令. 我们知道, 一般的编译工具均是为普通应用而设计的, 比如普通的 C 编译器. 即便是根据某种类型的 DSP 专门设计的编译程序也只能在一定程度上利用 DSP 的某些特性. 对于某个具体的算法, 如果用户希望充分利用 DSP 的特性, 通常情况下需要结合 DSP 的特性来进行专门的设计, 以提高算法的运算速度.

前面我们提到, 在 DCT 快速算法中, 乘法和加法是最主要的运算, 而普通的 DSP 一次只能完成一次加法或一次乘法. 可以想象: 如果 DSP 支持 VLIW, 那么我们可以将多个加法或多个乘法合并成为一条指令, 那么我们在一条指令周期中就可以完成多次加法或乘法, 从而提高 DCT 的运算速度. 下面我们以 Philips 公司的 TM1300 为例子, 说明如何利用 DSP 提供的超长指令提高 DCT 的运算速度.

2.1.1　计算 DCT 要使用的特殊超长指令

(1) DSPIDUALADD

这条指令可以计算两对有符号数(用 16 比特表示) 的和. 假定在 32 比特的寄存器 rsc1 的高 16 位和低 16 位分别保存了两个有符号数. 寄存器 rsc2 的高 16 位和低 16 位也保存了两个有符号数. 那么对它们执行 DSPIDUALADD 指令时, rsc1 和 rsc2 的高 16 位和低 16 可以对应相加, 相加的结果经过截尾处理后分别保存在 32 位的寄存器 rdest 的高 16 和低 16 位中. 也就是说, 这条指令可以在一个指令周期中完成两次加法.

(2) DSPIDUALSUB

DSPIDUALSUB 指令完成的操作与 DSPIDUALADD 指令非常相似, 唯一不同的地方是: 它完成的是减法操作.

(3) IFIR16

假定在 32 比特的寄存器 rsc1 的高 16 位和低 16 位分别保存了两个有符号数. 寄存器 rsc2 的高 16 位和低 16 位也保存了两个有符号数. 对它们执行 IFIR16 时, 首先是 rsc1 的高 16 位和低 16 位分别同 rsc2 的高 16 位和低 16 位对应相乘, 接下来是相乘得到的两个输出再相加. 也就是说 IFIR16 指令可以在一个指令周期内完成两次乘法和一次加法.

(4) PACK16MSB

PACK16MSB 可以将寄存器 rsc1 和 rsc2 的高 16 位合并成一个新的 32 位操作数并存入寄存器 rdest 中, rdst 的高 16 位来自 rsc1 的高 16 位, 它的低 16 位来自 rsc2 的高 16 位.

2.1.2　计算一维 DCT

我们知道在图图像编码中, 图像数据通常是以 8@8 的块为单位来处理的, 所使用的 DCT 变换也是二维 DCT 变换. 但从理论上说, 二维 DCT 变换可以通过一维 DCT 变换来实现, 所以在本文中我们主要介绍一维 DCT 计算方法.

(1) 组织数据

假定我们要计算 8@8 图像块的 DCT. 通常情况下每个像素的值需要 8 比特来表示, 这里我们统一用 16 比特的整数来表示. 由于 TM1300 的字长是 32 比特的, 所以我们可以将两个像素的值组合后放入一个 32 比特的寄存器中. 这样 64 个输入的像素值需要用 32 个 32 比特的寄存器来表示. 数据结构如图 3 所示.

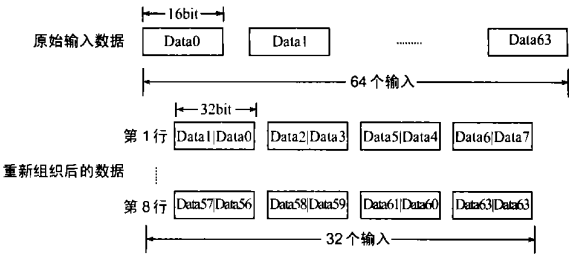

图 3　输入数据的结构

对于 64 个输入数据, 根据图 1 的流程图, 我们可以得到并行计算的蝶型流程图, 如图 4 所示.

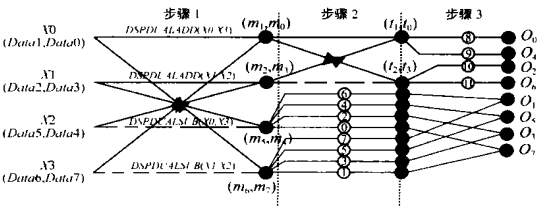

图 4　新的 DCT 计算流程图

(2) 蝶型运算

蝶型运算的第一步由四条指令完成, 两条是加法, 两条是减法. 这四条指令可以完成 4 次加法和 4 次减法. 这一步的操作可以用下式来表示:

$$(m_1, m_0) = \mathrm{DSPDUALADD}(x_0, x_3) = (data_1 + data_6, data_0 + data_7) \quad (1)$$

其中 m_1, m_0 分别表示一个寄存器的高 16 位和低 16 位.

(3) 计算偶数系数

经过第一步之后, 奇数系数和偶数系数将用不同的方法来计算. 对于偶数部分, 我们用两条 DSPDUALADD 指令来实现 4 次加法, 从而得到中间结果(t_1, t_0) 和(t_2, t_3).

根据图 1 的计算流程图, 我们可以导出 t_i 和 O_i 的关系如下式所示:

$$O_2 = \sqrt{2}\cos\frac{3P}{8}t_2 + \sqrt{2}\sin\frac{3P}{8}t_3 = 0.54t_2 + 1.31t_3 \quad (2)$$

$$O_6 = -\sqrt{2}\sin\frac{3P}{8}t_2 + \sqrt{2}\cos\frac{3P}{8}t_3 = -1.31t_2 + 0.54t_3 \quad (3)$$

从这里我们可以看出, 如果我们直接计算 O_2 和 O_6, 那么我们需要进行浮点运算, 这会耗费大量的时间. 为避免进行浮点运算, 我们对式(2) (3) 中的浮点系数进行标量化, 即每个系数乘 2^{14}, 并截去小数部分. 比如: 系数为$\sqrt{2}\cos\frac{P}{8}$] 1.31@2^{14} = 0X539E (16 进制表示), 同理, $\sqrt{2}\cos\frac{3P}{16}$] 0.54@2^{14} = 0X22A3. 这样我们有:

$$O_2 = \sqrt{2}\cos\frac{3P}{8}t_2 + \sqrt{2}\sin\frac{3P}{8}t_3 = \mathrm{IFIR16}(t_{2,3}, C_{10}) \quad (4)$$

$$O_6 = -\sqrt{2}\sin\frac{3P}{8}t_2 + \sqrt{2}\cos\frac{3P}{8}t_3 = \mathrm{IFIR16}(t_{2,3}, C_{11}) \quad (5)$$

其中 C_{10}= 0X22A3539E, C_{11}= 0XAC6222A3, 它们的高 16 位表示一个系数, 低 16 位表示另一个系数.

(4) 计算奇数部分

相对而言, 奇数系数的计算比较复杂. 首先我们需要导出输出 (1, 5, 3, 7) 与中间结果 m_i 的关系. 参考图 1 我们知道 O_1 可以用下列步骤来计算.

首先计算 t_{4-7}:

$$t_4 = m_4\cos\frac{3P}{16} + m_7\cos\frac{5P}{16},\ t_7 = -m_4\cos\frac{5P}{16} + m_7\cos\frac{3P}{16} \quad (6)$$

$$t_5 = m_5\cos\frac{P}{16} + m_6\cos\frac{7P}{16},\ t_6 = -m_5\cos\frac{7P}{16} + m_6\cos\frac{P}{16} \quad (7)$$

接下来计算 y_{4-7}

$$y_4 = t_4 + t_6 \quad y_6 = t_4 - t_6 \quad y_7 = t_7 + t_5 \quad y_5 = t_7 - t_5 \quad (8)$$

由此可以知道如何计算 O_{4-7}, 比如:

$$O_1 = y_7 + y_4 = t_4 + t_5 + t_6 + t_7$$
$$= m_4(\cos\frac{3P}{16} - \cos\frac{5P}{16}) + m_5(\cos\frac{P}{16} - \cos\frac{7P}{16})$$
$$+ m_7(\cos\frac{3P}{16} + \cos\frac{5P}{16}) + m_6(\cos\frac{7P}{16} + \cos\frac{P}{16}) \quad (9)$$

从式(9)知道, 奇数系数是由相乘相加来完成的, 因此我们可以用 IFIR16 和加法运算来实现, 即:

$$O_1 = IFIR16(m_{5,4}, C_6) + IFIR16(m_{6,7}, C_7) \quad (10)$$

上式中的常数(C_6, C_7) 也需要采用上面的方法进行处理.

(5) 组织一维 DCT 计算的结果

我们知道, 二维 DCT 可以通过先对输入数据的每一行做 DCT, 然后再对每一列做 DCT 来实现. 为此, 我们需要对水平方向 DCT 运算的结果进行组织, 为垂直方向的 DCT 运算做准备. 图 5 给出了数据的组织方式.

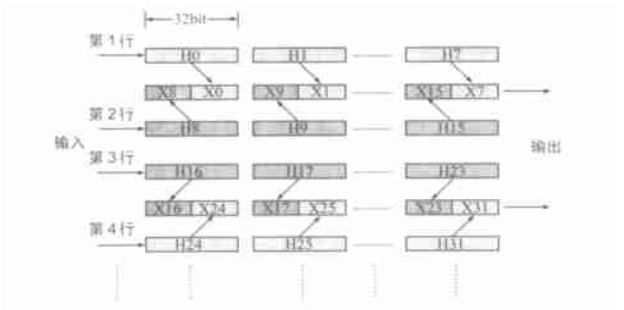

图 5 12D 系数的组织方式

对于 8@8 的输入数据, 我们先对它的每一行做 DCT, 假定 H0~H7 表示第一行 DCT 运算的结果, H8~H15 表示第二行 DCT 运算的结果, 以此类推. 从前面的步骤可以知道, 在计算 12D DCT 时, 所有的系数均被扩大了 2^{14} 倍并进行了截尾处理以便用整数运算取代浮点运算, 运算的结果保存在 32 位的寄存器中. 从理论上说, 在进行垂直方向的 DCT 之前, 我们应该将第一次运算的结果缩小 2^{14} 倍才能得到正确的 DCT 运算结果. 在实际应用中我们对第一次运算的结果缩小了 2^{-16} 倍, 这样得到的结果比理论值要小 4 倍, 这个常数倍数我们可以在第二次 DCT 运算结束后进行补偿, 以保证最终运行结果的正确性. 将一维 DCT 的结果缩小 2^{-16} 倍意味着我们可以简单地只取每个系数的最高 16 位做为输出. 从图 5 可以看出, 每个 32 位的 DCT 系数只取高 16 位后, 我们可以将两个 DCT 系数合并成存入一个 32 位的积存器中(这可以用 PACK16MSB 指令来实现), 数据的组织方式与前面组织数据的方式完全相同. 这样做的好处是: 可以用相同的 DCT 算法来计算垂直方向的 DCT, 而不需要重新设计算法.

利用上面的方式组织好一维 DCT 的运算结果后, 我们只要用同样的方法对该结果在计算一次就可以得到二维 DCT 的运算结果.

213 计算 IDCT

一维 IDCT 的运算与一维 DCT 非常相似, 图 6 给出了它的运算流程图.

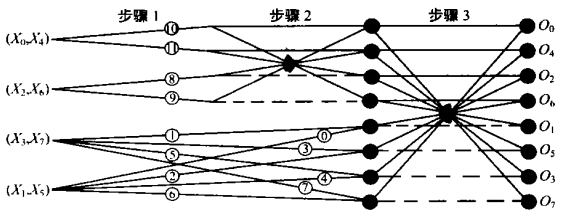

图 6 一维 IDCT 的计算流程

在每一行中, 我们需要将第 0 和第 4 个 DCT 系数存入一个 32 比特的寄存器中, 将第 2 和第 6 个系数存入另一个寄存器中, 依此类推. 计算过程也需要分为几个连续的步骤. 在第一个步骤中我们需要计算浮点乘法, 同样的我们将每个系数扩大 $2^{14}@\sqrt{2}$ 倍以便能用整数计算来替代浮点运算. 在完成第一次 IDCT 运算后我们也只保留 32 比特输出的高 16 位作为 IDCT 运算的输出. 只要再在垂直方向做一次 IDCT 我们就能得到二维 IDCT 的输出.

3 仿真结果与分析

为验证该算法的性能, 我们在 Philips 公司生产 Trimedia DSP 仿真板上进行了测试. 仿真板上安装有主频为 133MHz 的 DSP 芯片 TM1300. TM1300 是一款 32 比特的支持超长指令的 DSP. 我们利用仿真板提供的性能分析工具来估计二维 DCT 算法在 TM1300 上运行时所需要的指令周期, 以便评估各个算法运算性能. 此外我们还利用该工具软件来计算某段代码的指令级的并行程度(instruction level parallelism, ILP), ILP 值越大, 表示算法的并行程度越高.

表 1 是给出了我们测试的结果. 其中, 第二列表示总的指令周期(Total Instruction Cycles, TIC), 第三列给出了同第一种方法相比节省指令的比例(Saved Instruction Cycles, SIC), 最后一列是 ILP.

表 1 不同 DCT 算法的性能比较

算法	TIC	SIC(%)	ILP
标准算法	487539	)))	2.001864
快速算法	170395	65%(同标准算法比较)	2.030982
我们的方法	129529	73%(同标准算法比较) 24%(同快速算法比较)	2.078364

在这里我们测试了三种不同的 DCT 运算方法: 标准算法, 即根据根据 DCT 的定义直接计算; 传统的快速算法(如图 1 所示)和我们提出快速算法.

由此可以看出, 标准的 DCT 算法需要较多的指令周期, 快速算法可以节省 65% 的指令周期, 即: 运算速度是标准算

法的 2.8 倍; 我们提出的算法同标准算法相比可以节省 73% 的指令周期, 即: 运算速度是标准算法的 3.7 倍, 即便同快速算法相比我们提出的算法也可以提高 24% 的运算速度.

4　结论

本文提出了一种利用 DSP 的超长指令来计算 DCT 和 IDCT 的并行算法. 这种方法以 Loeffler 等人提出的经典快速算法为基础, 并利用超长指令的特性来提高运算速度, 主要步骤是: 合理组织输入数据, 利用超长指令进行并行计算, 将浮点运算转换为定点运算等. 利用该方法可以节省大量的指令周期, 提高运行的并行特性. 仿真结果表明, 同经典的快速算法相比, 它可以提高近 24% 的运算速度, 因此在图像压缩编码, 特别是实时视频编码中有非常广泛的应用.

参考文献:

[1] N Ahmed, T Natarajan, K R Rao. Discrete Cosine Transform [J]. IEEE Transaction on Computer. 1974, C223: 90- 93.

[2] M Vetterli, A Ligtenberg. A Discrete Fourier Cosine Transform Chip [J]. IEEE Journal on Selected Areas of Communications. 1986, SAC24 (1): 49- 61.

[3] A Ligtenberg, J H O'Neill. A Single Chip Solution for an 8 by 8 two Dimensional DCT [A]. Proceedings IEEE International Symposium on Circuits and Systems [C]. USA: IEEE, 1987. 11128- 1131.

[4] W A Chen, C Harrison, S C Fralick. A Fast computational Algorithm for the Discrete Cosine Transform [J]. IEEE Transactions on Communications, 1977, COM225(9): 1004- 1011.

[5] Z Wang. Fast Algorithms for the Discrete W2Transform and for the Discrete Fourier Transform [J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1984, ASSP232(4): 803- 816.

[6] Byeong Lee. A new algorithm to Compute the Discrete Cosine Transform [J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1984, ASSP232(6): 1243- 1245.

[7] H S Hou. A Fast Recursive Algorithm for computing the Discrete Cosine Transform [J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1987, ASSP235(10): 1455- 1461.

[8] Christoph Loeffler, et al. Practical Fast 12D DCT Algorithms with 11 multiplications [J]. Acoustics, Speech, and Signal Processing, 1989, ASSP289(12): 988- 991.

[9] ITU2T H. 263 Recommendation. Video coding for low bit rate communication [S].

作者简介:

李学明　男, 1969 年生于四川资阳, 北京邮电大学信息工程学院副教授, 1992 年毕业于中国科学技术大学无线系, 1997 在北京邮电大学获工学博士学位, 同年进入北方交通大学信息所做博士后, 2002 年在德国卡尔斯鲁厄大学计算机学院多媒体中心做访问学者, 主要研究方向包括: 图像处理, 视频编码与传输, 计算机网络和数据通信, 目前已在该领域发表论文十多篇, 多篇论文被 EI 检索.

李　继　男, 1977 年生于北京, 北京邮电大学信息工程学院硕士研究生, 1999 年毕业于北京邮电大学电信工程学院, 2002 年在德国卡尔斯鲁厄大学计算机学院学习, 主要研究方向包括: H26L 图像编码算法研究, VOD 系统和远程教育.

(上接第 1073 页)

[3] Beidas B F, Weber C L. Higher2Order Correlation2Based Approach to Modulation Classification of Digitally Frequency2Modulated Signals [J]. IEEE Journal on Selected Areas in Com, 1995, 13(1): 89- 101.

[4] Kim K, Polydoros A. Digital Modulation Classification: The BPSK versus QPSK Case [A]. IEEE, Inc: MILCOM 88 Conference Record [C]. USA: IEEE, 1988. 2: 431- 436.

[5] Lay N E, Polydoros A. Modulation Classification of Signal in Unknown ISI Environments [A]. IEEE, Inc: MILCOM 95 Conference Record [C]. USA: IEEE, 1995. 170- 174.

[6] 梁昆淼, 刘法, 缪国庆. 数学物理方法[M] (第三版). 北京: 高等教育出版社, 1998.

[7] Van Trees H L. Detection, Estimation, and Modulation Theory, Pt. I [M]. New York: John Wiley&Sons, Inc. 1968.

[8] Srinath M D, Rajasekaran P K. An Introduction to Statistical Signal Processing with Applications [M]. New York: Wiley &Sons, 1979.

[9] Poor H V. An Introduction to Signal Detection and Estimation[M]. New York: Springer2Verlag, 1994.

[10] Kay S M. Fundamentals of Statistical Signal Processing, Vol. II: Detection Theory [J]. New Jersey: Prentice Hall PTR, 1998Bennett W R. Method of solving noise problems. Proceedings of IRE, 1956, 44(5): 633.

[11] Raheli R, Polydoros A. Per2Survivor Processing: A General Approach to MLSE in Uncertain Environments [J]. IEEE Trans. on Com, 1995, 43 (2/3/4): 354- 364.

[12] Meyr H, Moeneclaey M, Fechtel S A. Digital Communication Receivers2 Synchronization, Channel Estimation, and Signal Processing [M]. New York: John & Sons, NC, 1998.

[13] Viterbi A J. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm [J]. IEEE Trans on IT, 1967, 13(12): 1967. 260- 269.

[14] Forney G D, Jr. Maximum2Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference [J]. IEEE Trans on IT, 1972, 18(3): 363- 378.

[15] Forney G D, Jr. The Viterbi Alogrithm [J]. Proceedings of IEEE, 1973, 61(3): 268- 278.

[16] 罗利春. 几种调制分类方法的原理与仿真试验研究. 系统工程与电子技术 [J]. 2002, 24(11): 87- 90.

[17] 罗利春. 无线电侦察信号分析与处理[M]. 北京: 国防工业出版社, 2003.